Backward Model Tracing: An Explanation-Based Approach for Reconstructing Student Reasoning

Danilo Fum (*), Paolo Giangrandi(°), Carlo Tasso (°)

(*) Dipartimento di Psicologia, Università di Trieste
Via dell'Università 7, I-34123 Trieste, Italy - fum@uts882.units.infn.it.bitnet
(°) Laboratorio di Intelligenza Artificiale, Università di Udine
Via Zanon 6, I-33100 Udine, Italy - tasso@uduniv.infn.it.bitnet

Abstract

An original methodology, called *backward model tracing* to model student performance which features a profitable integration of the bug collection and bug construction techniques is presented. This methodology has been used for building the modelling module of a new version of ET (English Tutor), an ITS aimed at supporting the learning of the English verb system. Backward model tracing is based on the idea of analyzing the reasoning process of the student by reconstructing, step by step and in reverse order, the chain of reasoning (s)he has followed in giving his/her answer. In order to do this, both correct domain specific knowledge and a catalogue of stereotyped errors (malrules) are utilized. When the system is unable to explain the student behavior by exploiting its previous knowledge, new malrules are generated dynamically, by utilizing explanation-based learning techniques. The overall process is based on a deep modelling of the student problem solving and the discrimination among possible explicative hypotheses about the reasons underlying the student behavior is carried on non-monotonically through a truth maintenance system. The proposed approach has been fully implemented in a student modelling module developed in PROLOG.

1. Introduction

One of the most important features an intelligent tutoring system (ITS for short) should provide, is the capability to adapt its behavior to the specific traits of the student. To this purpose, a fundamental contribution is given by the component aimed at building and maintaining the student model. The student model describes the knowledge and beliefs of the student in the specific subject domain and is used for designing and taking appropriate tutorial and remedial actions, tailored to the peculiarities of the student.

Building an ITS with a student modelling component is hindered by several problems concerning both theoretical and practical issues. There is a sufficiently general agreement on the fact that the modelling activity cannot be based only on the knowledge provided by an expert in the domain but it is better performed by relying on an explicit description of possible student (mis)behaviors.

Unfortunately, no agreed-upon solution exists on this topic and the three main approaches adopted for building student modelers (i.e., overlay, bug collection and bug construction: Wenger 1987) directly reflect this situation. In fact, each technique has been generally used alone and no clear ways have been proposed to combine these techniques in order to exploit their respective advantages. Considered by itself, each technique has known drawbacks and falls short of constituting an ideal tool for building cognitively adequate and computationally sufficient student models: overlay is generally considered as not sufficiently powerful to perform sophisticated modelling, collecting catalogues of bugs is notoriously a dull and labor-intensive endeavor, while bug construction has not yet proved to be a reliable and sufficiently comprehensive approach.

In this paper we present an original methodology, called *backward model tracing* to model student performance which features a profitable integration of the bug collection and bug construction techniques. This methodology has been used for building the modelling module of a new version of ET (English Tutor), an ITS aimed at supporting the learning of the English verb system. Backward model tracing is based on the idea of analyzing the reasoning process of the student by reconstructing, step by step and in reverse order, the chain of reasoning (s)he has followed in giving his/her answer. In order to do this, both correct domain specific knowledge and a catalogue of stereotyped errors (malrules) is utilized. When the system is unable to explain the student behavior by exploiting its previous knowledge, new malrules are generated dynamically, by utilizing explanation-based learning techniques.

The overall process is based on a deep modelling of the student problem solving and the discrimination among possible explicative hypotheses about the reasons underlying the student behavior is carried on non-monotonically through a truth maintenance system.

Backward model tracing seems a promising approach to tackle the hard problem of student modelling (Self 1988) for the following reasons:

- it shares the benefits of the model tracing methodology (Anderson 1987);
- it exploits the respective advantages of bug collection and bug construction without the limitation of the exclusive usage of a single technique;

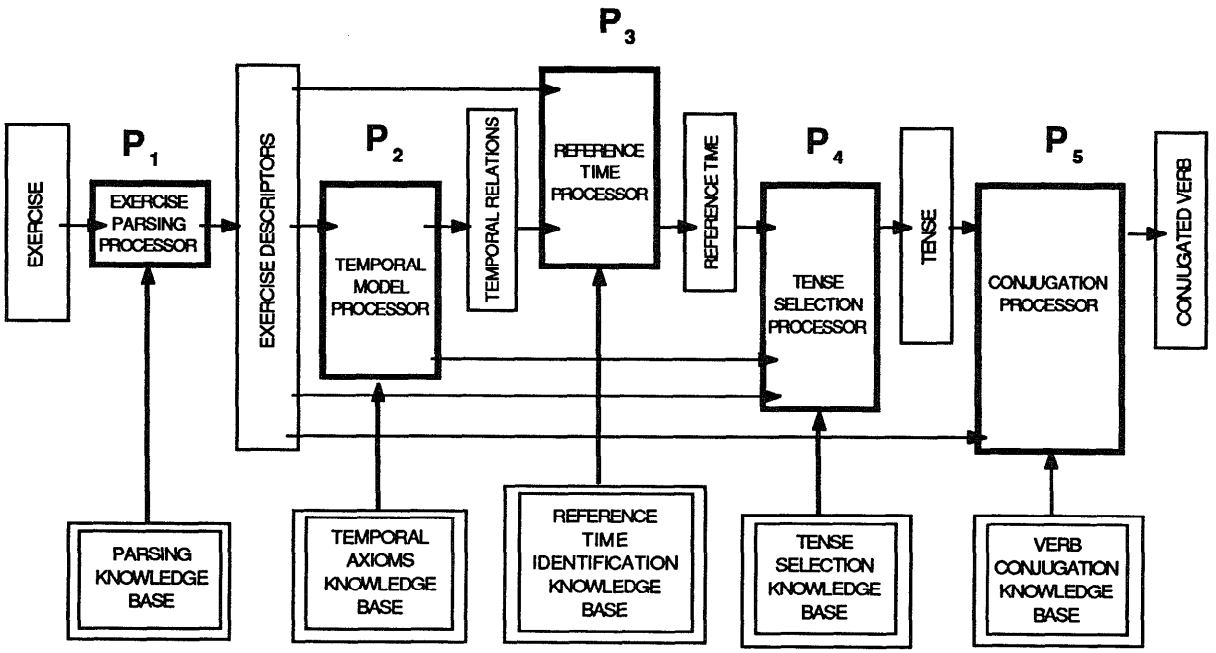

Figure 1. The tense generation process.

- it exploits a sophisticated technique such explanation-based learning to push the limits of the bug construction methodology.

The paper is organized as follows: The following section briefly presents the ET system where backward model tracing has been applied. Section 3 illustrates the general criteria upon which the methodology is grounded. Section 4 and 5 describe the technical details of the modelling process focusing on the generation of explanatory hypotheses about the causes of the student erroneous behavior and on the discrimination among the formed hypotheses, respectively. Section 6 presents the conclusions of the work.

2. Second Language Tutoring in ET

The backward model tracing methodology has been utilized in the new version of the ET system (Fum, Giangrandi, & Tasso 1988), an ITS aimed at supporting Italian students in the learning of the English verb system. ET comprises an articulated knowledge-based domain expert (Fum, Giangrandi, & Tasso 1989), devoted to tense generation, capable to solve fill-in exercises constituted by one or more sentences in which some verbs have to be conjugated into the appropriate tenses. The operation of the domain expert is organized around a sequence of five phases (illustrated in figure 1), each devoted to a specific (sub)process, namely:

- parsing the exercise sentence(s);
- recognizing the temporal relations among the events described in the sentence(s);
- identifying the reference time for every clause in the sentence(s);
- selecting the correct tense to be used for each verb;
- conjugating the verb(s) into the appropriate tense.

Each process is carried out by a dedicated processor, which is supported by a knowledge base encoding the knowledge usually exploited by humans for that subtask. The main representation paradigm utilized in the knowledge bases is constituted by production rules. The generation expert produces also a precise track of the reasoning performed and of the knowledge utilized for solving the exercises.

The second fundamental module of ET is the tutor which is devoted to define the modalities of the teaching activity. More particularly, the tutor assigns the exercises according to a given syllabus, cooperates with the modeler in order to discriminate among alternative hypotheses about the student erroneous behavior, and manages the dialogue with the student.

The student modeler is the module which implements the backward model tracing methodology. The general goal of this module is to discover the domain specific knowledge the student has utilized in order to derive her/his answer(s). The modelling process aims at constructing a model of the students beliefs in which both correct knowledge and misconceptions are explicitly represented. The student model, which is partitioned into different parts, one for each computational phase, contains therefore a collection of facts and rules that have been utilized in order to interpret the student behavior and that are supposed to mirror what the student knows/beliefs about the subject domain. The student model supplies the basis for remedial activity which is aimed at correcting the student misconceptions. The structure and the functioning of this module constitutes the subject of this paper.

3. Modelling Student Performance

Backward model tracing relies on two basic assumptions:
(i) in achieving the solution, the student follows a process akin to that used by the expert module i.e., it goes through the same phases and performs essentially the same kind of computations, and
(ii) the student mistakes can be modelled by appropriately perturbing the knowledge utilized by the expert module.

Some words are needed in order to justify these assumptions. It could be claimed that assumption (i) is unrealistic: in fact it is easy to find evidence that novices solve problems by using strategies that are different from those utilized by experts (e.g., diSessa 1982; Reif 1987). While this cannot be denied, it is also true that, in order to be able to model possible student misbehaviors, a model of the correct performance is required. The student behavior needs thus to be compared with that of an 'expert', being this a real domain expert, a teacher or an 'ideal student' (Anderson 1987). The general philosophy followed in ITS development is to locate the domain specific knowledge in the expert module which plays therefore a twofold role: it acts as the *source* for the knowledge to be presented and, at the same time, it serves as the *standard* for evaluating the student's performance" (Wenger 1987). In our context we assume that a student, trying to figure out the correct answer for an exercise, follows the same reasoning steps of the expert module, i.e. after interpreting the meaning of the exercise sentences (s)he computes the temporal relations between the states/events described in the sentence, calculates the reference times for every clause of the sentence, chooses the tense for the verb and, finally, conjugates the verb in that tense. Assumption (i) implies that the errors the student makes could derive only from the fact that some of the rules (s)he applies are 'bugged', not from the fact that (s)he can follow computational paths different from those of the expert. Assumption (ii), (shared by other authors: Brown & Burton 1980; Sleeman 1983; Bonar & Soloway 1985) states that it is possible to model these bugs by assuming more or less severe deviations from the knowledge base actually utilized by the expert module.

Backward model tracing is grounded on the idea of trying to reconstruct, step by step and in reverse order, the chain of reasoning the student has followed in building the answer. Backward model tracing is triggered by the discovery of a mismatch between the answer given by the student and that provided by the expert module. The goal of the modelling process is to identify the phase(s) where the reasoning process of the student and of the expert differ, and the specific erroneous rules *(malrules)* applied by the student. Backward model tracing analyzes the reasoning process performed by the student starting from the last phase and going back toward the first ones. For each phase, the modeler tries to determine the input to the phase and the knowledge the student has utilized in order to produce the corresponding output. If a mismatch between the student and the expert output is discovered, it could mean that: (a) some of the rules contained in the knowledge base utilized by the student in that phase, or (b) some of the data utilized as input for that phase or, (c) both some of the rules and the data differ from those utilized by the expert.

The goal of the diagnostic process performed by the modeler is to realize which of the above alternatives holds. Case (a) is true when both the expert and the student work with the same input data but their output is different because some of the rules contained in the student knowledge base are actually bugged. In this case a remedial activity could be planned in order to clarify the student misconceptions. As an example of case (a) let us consider the case when both the student and the expert have to form the present perfect of 'to study' and the student produces as an answer 'has studyied'. If (b) is true, then at least one of the previous phases should be blamed for producing the erroneous data and the diagnostic process is repeated focusing on the phase immediately preceding the current one. As an example of (b) let consider the case of the student answering 'has studied' when in fact the correct verb tense is the past perfect. If (c) is true, then both the above mentioned activities occur, i.e., the malrules responsible for the mistake made in that phase are identified and the diagnostic process continues with the preceding phases. An example of case (c) is the answer 'has studyed' given when the past perfect is required.

Backward model tracing shares all the features of the model tracing methodology (Anderson 1987), i.e. it tries to simulate dynamically a student's problem solving and uses that simulation to interpret the student behavior. Differently from Anderson's approach:
- it does not rely only on a-priori established catalogue of correct and incorrect productions but it is able to dynamically generate the malrules necessary to explain the student performance;
- the tracing occurs *after* the student has produced his/her performance and it is not used to monitor the student during the solution process just to assure that the correct path will be followed.

As a result, backward model tracing possibly represents a less intrusive modelling methodology and a more general diagnostic procedure.

Having established the general features of our approach to student modelling, we concentrate now on the technical details of the diagnostic process as it is performed within a single phase.

4. Explaining the Student Answer

The diagnostic process aimed at reconstructing the student reasoning for each single phase can be divided in two activities: the first is aimed at generating all the possible lines of reasoning which constitute putative explanations of how the student determined his solution in that phase; the second is aimed at discriminating among the different candidate explanations trying to discover the one(s) that best account(s) for the actual student reasoning. The present section explains how the first activity is performed; section 5 deals with the hypotheses discrimination activity.

In order to formulate possible explanations for the student behavior, the modeler operates in backward chaining and generates a derivation tree containing the rules and facts used to produce the output of the phase. More precisely, the output is considered as explained if it is possible to prove it by exploiting the production rules contained in the knowledge base the student utilizes for that phase. As a result of the diagnostic process, the modeler will be able to identify the input data for that phase and the possible malrules utilized by the student.

In order to give a more comcrete idea of the diagnostic process, let us follow the system operation through a worked out example, restricted for simplicity only to the last verb conjugation phase. The student, requested to conjugate into the appropriate tense the verb 'to study', gives as an answer 'is studyed' while the output given by the expert module is 'has studied'. As a first step, the modeler transforms the student answer into the following clause

verb(Verb, Tense, Person, Number, [is, studyed]/[])

which represents the goal to be proven and whose first four arguments will be instantiated at the end of the diagnostic process with the input data for the phase. The modeler then tries to construct one or more derivation trees for that goal. To this purpose, the system tries to find, among the rules contained in the student knowledge base, that whose right hand side matches the current goal (if several rules match the goal, the modeler constructs several derivation trees in parallel) and it tries to prove each antecedent clause in the left hand side of the rule. If a clause constitutes a primitive goal (for example a fact contained in the dictionary) it is considered as proven and the modeler goes on to analyze a new clause, otherwise the modeler proceeds recursively by trying to build a derivation (sub)tree for that clause.

The main improvement of our algorithm in comparison with similar approaches (e.g. classic backward chaining, the resolution plus oracle method reported in (Costa, Duchenoy, & Kodratoff 1988), and the technique utilized in (Sleeman 1983)), concerns the treatment of the failing situations. In fact, when the modeler finds a subgoal which is unprovable (i.e. which is neither a primitive goal nor can be demonstrated by applying the rules contained in the student knowledge base) it tries to recover from this situation by exploiting two different modelling strategies. First, it can resort to the bug collection technique by selecting in a catalogue of malrules, representing instances of stereotyped errors students generally make, an appropriate malrule which could be used to prove the current goal. This malrule is then imported as a hypothetical misconception into a/the derivation tree for the student answer. Second, if none of the available malrules is suitable to prove the current goal, the modeler tries to generate a new malrule by perturbing an expert rule; in other words a rule representing certain domain knowledge is purposely modified and made incorrect in order to use it to prove the original goal. This malrule is also imported into the derivation tree.

While the first strategy simply utilizes the classic bug collection approach, the second strategy (bug construction) exploits machine learning techniques in order to generate new malrules. Among the possible machine learning techniques usable for generating new malrules, we have adopted an explanation-based approach (Mitchell, Keller, & Kedar-Kabelli 1986; DeJong & Mooney 1986). This approach, which has been already exploited in the field of ITS (Bar-On & Or-Bach 1988), seems particularly promising to be utilized for student modelling for the following reasons:

- differently from other machine learning techniques, it requires only a few training instances (possibly only one) to be applied and it seems therefore particularly suitable for modelling when only a few student answers are available;
- it is an intensive knowledge-based technique and it seems therefore suitable in a field where a lot of domain-dependent knowledge is available in terms of both the expert knowledge base and of diagnostic knowledge (i.e. knowledge about possible student errors).

Bug construction thus constitutes the major tool when no a-priori information about a specific student (mis)behavior is available, i.e. when a student makes an error not contained in the bug library. The generative capability of the bug construction methodology, however, can be improved by taking into account other useful kinds of diagnostic knowledge. For this reason, we have generalized the bug catalogue approach in order to be able to represent any kind of misconception a student could possibly have about expert knowledge. The solution adopted is to allow the bug library to specify not only the malrules but also to describe possible perturbations of them (Hirsh 1985). To perturb a rule, the modeler uses the knowledge contained in the so-called *meta-bug library* of rules which specifies the kinds of perturbations that could be applied to a rule. This knowledge, which forms a theory of the possible errors students can make, is represented with general diagnostic rules whose condition part specifies a failing situation (a general pattern for a goal that cannot be proved) and whose conclusion part specifies a clause that could be used to replace the failing condition.

Coming back to our example, the spelling error contained in the student answer can be explained by the following stereotyped malrule (MR1):

IF add_string(V,ed,V_ed)
THEN form_ed(V,[V_ed \X]/X).

which says that to form the ed-form of a verb it is sufficient to add the suffix 'ed' to the root form of the verb.
Two rules of the meta-bug library (MB1 and MB2) utilized by the system are the following:

IF verb(have,T,P,N,V1/V2) *% Failure*
THEN verb(be,T,P,N,V1/V2). *% Repair*

IF regular(Verb, Tense) *% Failure*
THEN true. *% Repair*

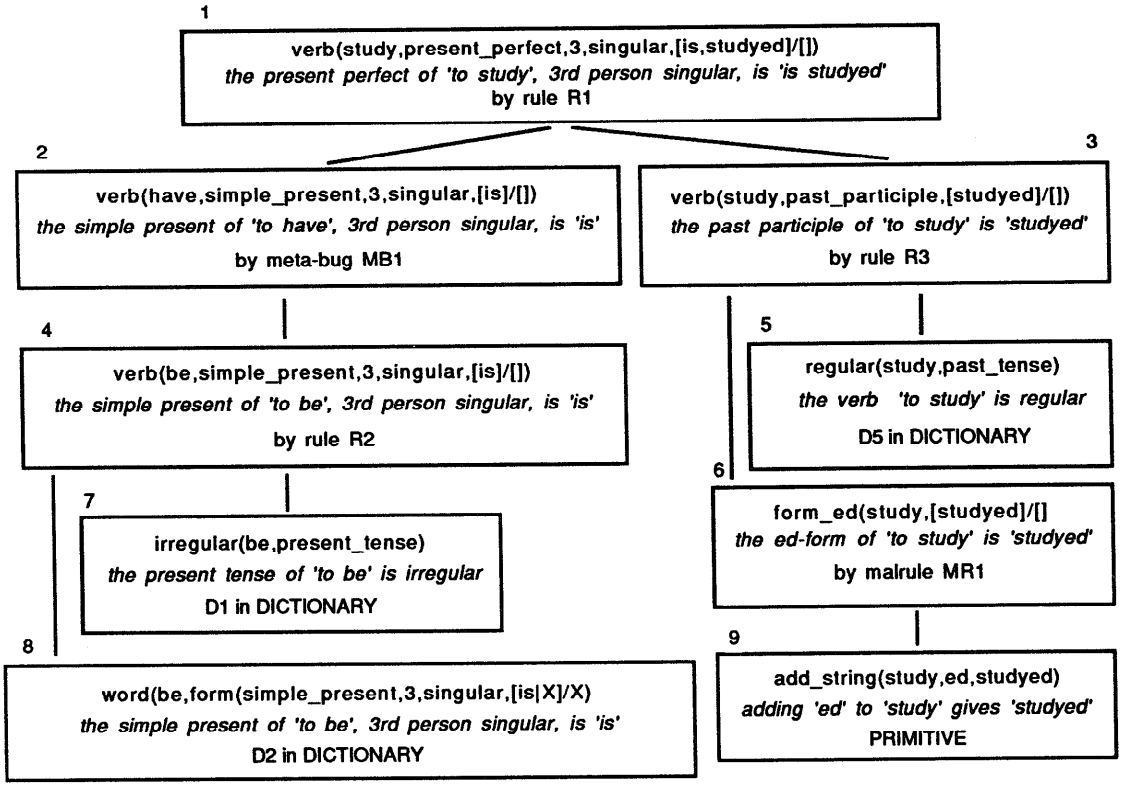

Figure 2. A derivation tree.

The first rule takes care of the case in which a student could use the auxiliary 'to be' instead of 'to have' (auxiliary inversion represents a typical error made by Italian students) by saying that, in order to prove a failing clause containing the verb 'to have', it is possible to replace it with a clause containing 'to be'. The second rule takes care of the case in which the student may forget the constraint that, in order to use a given conjugation rule, a regular verb is required, by relaxing that constraint and considering the clause to be proven as true.

In our example, the first meta-rule allows to explain the incorrect use of the auxiliary. In fact, when the modeler tries to prove the original goal

verb(Verb, Tense, Person, Number, [is, studyed]/[])

it finds a match with the consequent of the following conjugation rule (R1):

IF verb(have,simple_present,Person,Number,V1/V2),
verb(V,past_participle,V2/V3)
THEN verb(V, present_perfect,Person,Number, V1/V3).

(the present perfect of a verb is formed by joining the simple present of 'have' and the past participle of the verb). To prove this goal it tries to prove the two antecedent clauses. It fails with the first one since nowhere it can find a rule that enables it to deduce that

verb(have, simple_present, Person, Number, [is]/[])

(i.e. that the simple present of have is 'is'). But now the meta-bug MB1 can be applied to this clause thus obtaining

verb(be,simple_present,Person, Number, [is]/[])

which can be proven by looking at another rule (R2). By introducing this new clause it is possible to positively conclude the building of the derivation tree. Figure 2 illustrates a complete example of a derivation tree for the above discussed goal while figure 3 shows both a PROLOG and a natural language version of the knowledge utilized. Each node of the derivation tree (with the exception of the root) specifies a (partial) conclusion produced during the hypothesized reasoning process of the student, while the root represents the final output. In the specific case, the student built incorrectly the present perfect as a combination of the simple present of the verb 'to be' (rule R1) and the ed-form of the verb 'to study'. His wrong answer has been caused by the mistakes made in producing the present perfect and the ed-form of 'to study' (this mistake is represented in the bug catalogue by the malrule MR1).

Since each derivation tree describes the reasoning of the student applied to a very specific situation, in order to be able to extract more general malrules it is necessary to generalize the derivation tree containing the mistake.

CONJUGATION RULES

```
%% R1: The present perfect is formed with the simple present of the verb
%% 'to have' and the past participle of the verb.
  IF verb(have,simple_present,Person,Number,V1/V2),
     verb(Vb,past_participle,V2/V3),
  THEN verb(Vb,present_perfect,Person,Number,V1/V3).
%% R2: The simple present of an irregular verb is contained in the dictionary.
  IF irregular(Vb,present_tense),
     word(Vb,form(simple_present,Person,Number,V1/V2)),
  THEN verb(Vb,simple_present,Person,Number,V1/V2).
%% R3: The past participle of a regular verb is formed with the ed-form
%% of the verb.
  IF regular(Vb,past_participle),
     form_ed(Vb,V1/V2),
  THEN verb(Vb,past_participle,V1/V2).
```

VERB DICTIONARY

```
%% D1: The simple present of the verb 'to be' is irregular.
  irregular(be,present_tense).
%% D2: The simple present of the verb 'to be', 3rd singular, is 'is'.
  word(be,form(simple_present,3,singular,[is|X]/X)).
%% D3: The simple present of the verb 'to have' is irregular.
  irregular(have,present_tense).
%% D4: The simple present of the verb 'to have', 3rd singular, is 'has'.
  word(have,form(simple_present,3,singular,[has|X]/X)).
%% D5: The verb 'to study' is regular.
  regular(study,past_tense).
```

BUG LIBRARY

```
%% MR1
  IF add_string(V,ed,V_ed)
  THEN form_ed(V,[V_ed|X]/X).
```

META BUG LIBRARY

```
%% MB1
  IF verb(have,T,P,N,V1/V2)
  THEN verb(be,T,P,N,V1/V2).
%% MB2
  IF regular(Verb,Tense)
  THEN true.
```

Figure 3. Domain and diagnostic knowledge.

For this purpose, the modeler considers a more general structure, called *explanation structure*, which highlights the various rules (domain rules and possible diagnostic rules) applied during the construction of the derivation tree. Figure 4 illustrates the explanation structure originating from the derivation tree of figure 2. Each box contains the pair of rule clauses unified during the construction of the derivation tree.

At this point, while the standard explanation-based learning technique works on the whole structure in order to extract a single general concept, our algorithm proceeds instead focussing on the subtrees containing some meta-bugs; more precisely, it considers those rules who have clauses linked to the meta-bugs. From a single derivation tree it is thus possible to infer more than one malrule. In our example, illustrated in figure 4, the modeler concentrates upon the subtree constituted by rule R1 and the meta-bug MB1. For each subtree, the modeler unifies the rule clauses to which a meta-bug has been applied with the left hand side of the meta-bug and then it applies the substitution to the right hand side of the meta-bug. In our example, the antecedent clause of R1

verb(have, simple_present,P1,N1,V1/V2)

is unified with the left hand side of meta-bug MB1

verb(have, T2, P2, N2, V4/V5)

and the substitution {T2 ⇔ simple_present, P2 ⇔ P1, N2 ⇔ N1, V4 ⇔ V1, V5 ⇔ V2} is then applied to the right hand side of the meta-bug. The resulting right hand side of meta-bug MB1 is thus:

verb(be,simple_present,P1,N1,V1/V2)

Now, the final step consists in substituting this clause to the antecedent of rule R1. The final resulting malrule is therefore:

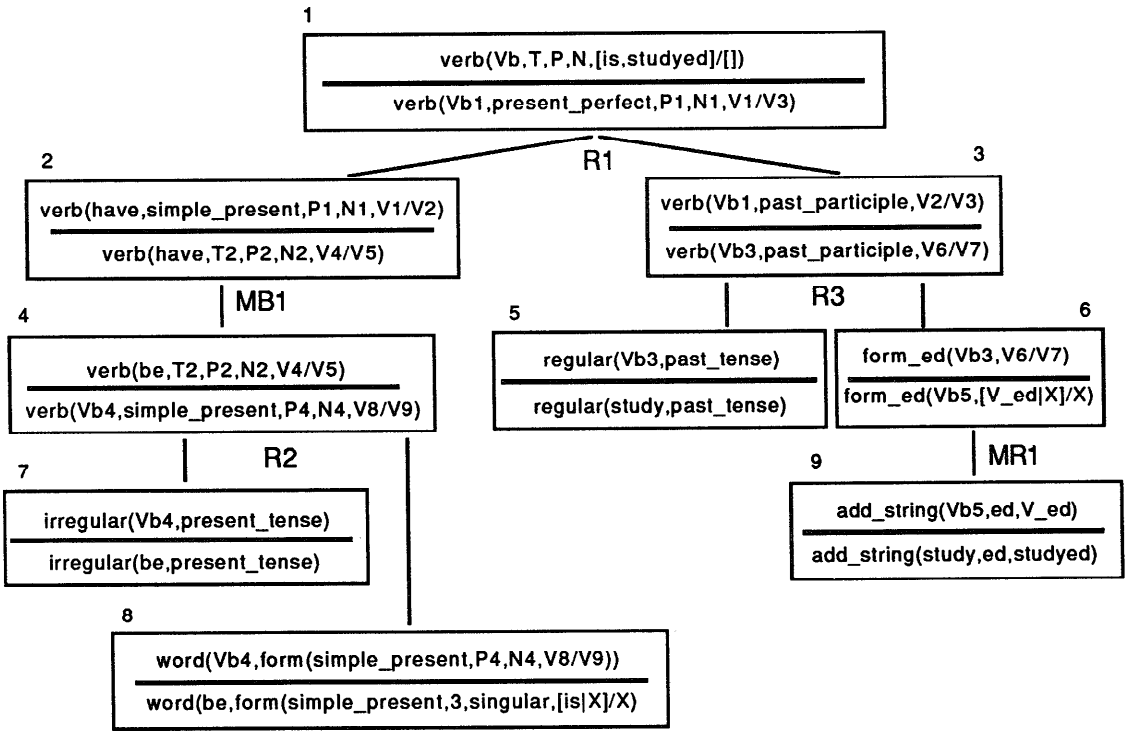

Figure 4. An explanation structure.

IF verb(be,simple_present,P1,N1,V1/V2),
verb(Vb1,past_participle,V2/V3)
THEN verb(Vb1,present_perfect,P1,N1, V1/V3),

meaning that the present perfect could be erroneously formed with the simple present of the verb 'to be' and the past participle of the verb.

5. Discriminating among Explanations

In general, the result of the first modelling activity is a collection of derivation trees which represent different possible reasoning processes followed by the student. The second basic activity of the modelling process has the goal to test all the generated hypotheses in order to select the best one(s) which account(s) for the student performance. During this operation the modeler behaves non monotonically discarding contradictory or very unlikely hypotheses, retracting the disconfirmed assumptions and transforming hypotheses into certified facts.

More particularly, in order to discriminate among the different candidate explanations the modeler has built, two strategies are followed. On one hand, some heuristic criteria are utilized to eliminate unplausible hypotheses or to derive new assertions about the student. On the other hand, the hypotheses about the reasons of the student behavior are maintained by utilizing a justification-based truth maintenance system (Doyle 1979; McAllester 1982).

Some of the heuristic criteria utilized to discriminate among the different hypotheses are the following:

- *If the student answer is explained by a single derivation tree, the rules and the facts contained in the tree are inserted into the student model.*
- *If a hypothesized malrule appears in all the alternative derivation trees, it can be taken for sure and inserted into the student model.*
- *If a derivation tree contains a number of perturbations which is greater than an established threshold, the derivation tree and its hypothesized malrules are discharged.*
- *If a hypothesized malrule, applied to simulate the student behavior in an exercise different from that by which it was originally derived, produces an answer different from that given by the student, then it is considered as unplausible and discharged.*

The hypotheses the modeler makes about the student are managed through a justification-based truth maintenance system. In particular, the hypotheses are organized into a network (called *dependency network*) which describes the dependency relations among different hypotheses. For example, the rules and the dictionary facts R1, R2, R3, D1, D2, and D5 support the derivation tree of figure 2 and this derivation tree supports both the stereotyped malrule MR1 and the newly constructed one. If some node of the network is deleted during the modelling process, then all the nodes that are dependent on it are also cancelled. This operation is achieved by following the edges of the dependency network. On the other hand, when a previously constructed

hypothesis is confirmed, it is introduced into the student model and, furthermore, the consequences of this change are propagated through the dependency network thus influencing the truth of the other hypotheses supported by it.

The diagnostic process requires generally also the cooperation of the tutoring module. When the information the system has about the student is insufficient to discriminate among the different alternative hypotheses generated by the modeler, the tutor can (i) assign new exercises in order to gather data to confirm or disconfirm them or (ii) can start an interactive (menu-driven) dialogue with the student. More precisely, the tutor selects from the exercise database the exercises related to the hypothesized misconceptions and, for each of them, the system determines the answers corresponding to these misconceptions. An exercise is considered as discriminating if it gives different answers for every different hypothesized misconception. The tutor then can select the most discriminating exercise. If no discriminating exercises are available, it is necessary to use the dialogue in order to establish which misconception reflects the student knowledge. It must be pointed out that the tutor considers not only the derivation tree(s) related to the last exercise but it also analyzes the derivation trees of the previous exercises in order to examine the remaining hypotheses. The analysis of a student misconception cannot therefore generally be completed in a single exercise but it requires a sequence of exercises.

6. Conclusions

A new methodology for modelling student performance has been presented which could be used for making diagnosis in those systems that, like ET are based on processes decomposable into a finite number of subtasks related to each other through producer-consumer dependencies. It is claimed, in other words, that the deep modelling process illustrated in the paper has general significance beyond the domain of second language teaching and can constitute an attempt to deal with the problem of finding the reasons of and giving a satisfactory explanation to the student performance. Some features of the proposed approach that we find particularly original are: (i) the integration of the bug collection and bug construction techniques, (ii) the use of diagnostic knowledge contained in the meta-bug library in order to help the process of bug construction, and (iii) the use of explanation based learning techniques in the domain of modelling students' misconceptions. The proposed approach has been fully implemented in a student modelling module written in QUINTUS PROLOG and running on a SUN 3 workstation.

References

Anderson, J. 1987. Production Systems, Learning, and Tutoring. In D. Klahr, P. Langley, and R. Neches eds. *Production System Models of Learning and Development.* Cambridge, MA: The MIT Press.

Bar-On, E.; and Or-Bach, R. 1988. Explanation-Based Learning in Intelligent Tutoring Systems. In P. Ercoli, and R. Lewis eds. *Artificial Intelligence Tools in Education.* Amsterdam, NL: Elsevier Science Publ.

Bonar, J.G.; and Soloway, E.M. 1985. Pre-Programming Knowledge: A Major Source of Misconceptions in Novice Programmers. *Human-Computer Interaction* 1(2): 133-161.

Brown, J.; and Burton, R. 1980. Diagnostic Models for Procedural Bugs in Mathematical Skills. *Cognitive Science* 4: 379-426.

Costa, E.; Duchenoy, S.; and Kodratoff, Y. 1988. A Resolution Based Method for Discovering Students' Misconceptions. In J.A. Self ed. *Artificial Intelligence and Human Learning.* London, UK: Chapman and Hall.

DeJong, G.F.; and Mooney, R.J. 1986. Explanation-Based Learning: An Alternative View. *Machine Learning* 1: 145-176.

diSessa, A.A. 1982. Unlearning Aristotelian Physics: A Study of Knowledge-Based Learning. *Cognitive Science* 6: 37-75.

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

Fum, D.; Giangrandi, P.; and Tasso, C. 1988. ET: An Intelligent Tutor for Foreign Language Teaching. In Proceedings of ITS-88, 462-468. Montreal, Canada.

Fum, D.; Giangrandi, P.; and Tasso, C. 1989. Tense Generation in an Intelligent Tutor for Foreign Language Teaching: Some Issues in the Design of the Verb Expert. In Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics, 124-129. Manchester, UK: Association for Computational Linguistics.

Hirsh, H. 1985. Modeling Problem Solving Performance. Master's thesis, Stanford Univ.

McAllester, D.A. 1982. Reasoning utility package user's manual. Technical Report, AIM-667, Artificial Intelligence Laboratory, M.I.T.

Mitchell, T.M.; Keller, R.; and Kedar-Kabelli, S. 1986. Explanation-Based Generalization: A Unifying View. *Machine Learning* 1: 47-80.

Reif, F. 1987. Interpretation of Scientific or Mathematical Concepts: Cognitive Issues and Instructional Implications. *Cognitive Science* 11: 395-416.

Self, J.A. 1988. Bypassing the Intractable Problem of Student Modelling. In Proceedings of ITS-88, 18-24. Montreal, Canada.

Sleeman, D. 1983. Inferring (Mal) Rules from Pupils' Protocols. In Proceedings of the Second International Machine Learning Workshop, 221-227. Chicago, IL.

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