Reasoning about Student Knowledge and Reasoning

Luigia Carlucci Aiello, Marta Cialdea, Daniele Nardi
Dipartimento di Informatica c Sistemistica
Universita di Roma "La Sapienza"
Via Salaria, 113 1-00198 Roma, ITALY

Abstract

A basic feature of Intelligent Tutoring Systems (ITS) is their ability to represent domain knowledge that can be attributed to the student at each stage of the learning process. In this paper we present a general (first order logic) framework for the representation of this kind of knowledge acquired by the system through the analysis of the student answers. This represantation makes it possible to describe the behaviour of well known ITSs and to provide a direct implementation in a logic programming language. Moreover, we point out several improvements that can be easily achieved by exploiting the features of a declarative approach. In particular, we address the representation and use of the knowledge that the system knows not to be possessed by the student.

1 Introduction

The goal of Intelligent Tutoring Systems (ITSs) is to support the learning process of a student on a specific sub ject domain. This is usually achieved through a conversational system which aims at providing the student with the knowledge and skills required for solving problems in the domain of interest. The learning status reached by the student is also tested, mistakes are discovered, tlieir causes are diagnosed and eliminated by appropriate remediation actions.

ITSs usually embody a knowledge base built by analyzing the student's responses to tests and problems. This knowledge base, called student model, represents the knowledge attributed by the system to the student and can therefore include explanations of the student's mistakes. Such explanations are built by tak ing into account empirical knowledge about students[1] mistakes derived from the study of typical student behaviours. The idea is that the ITS attempts to single out matters that the student may have not well understood and therefore need further attention (for a detailed description of the typology and role of student modelling in ITSs see for instance [Self, 1988, VanLehn, 1988]).

A first part of the work presented in this paper is concerned with a rational reconstruction of the process of building and maintaining the knowledge base representing the system knowledge about the student knowledge in existing ITSs, with the aim of identifying a domain independent formalization schema that accommodates apparently different approaches. Previous results of this work are presented in [Cialdea *et* al., 1990, Cialdea, 1991] and show that it is indeed possible to provide a common framework for the specification of some well-known systems and that such a formal specification can be directly translated into a run liable logic program.

A further important outcome of this formalization, winch is in the spirit of Self's Computational Mathetics [Self, 1991], is that within a simple declarative framework some previously neglected aspects of the representation and use of the system knowledge about the student become easily available.

The main goal of our work is to improve the representation and use of the system knowledge about student, knowledge. We take a deductive approach to representing knowledge, where the information manipulated by the system is represented in knowledge bases (for generality, we consider them to be sets of first-order logic formulae, called *contexts),* arranged in a met.a-level architecture. This approach originates from our work on reasoning about knowledge and reasoning in a multi-agent scenario [Aiello *et* al., 1991]. In the case of ITSs, we have two agents the system and the student. The interaction between the system and the student can therefore be regarded as a process that enables the system to acquire new knowledge about the student.

The process is activated by the answers provided by the student to specific problems. They can be definite answers (either correct or incorrect) or "I don't know'[1] answers. Typically, existing ITSs do not analyze "I don't know" answers, that, conversely, can provide useful information about the student's learning status.

The analysis of student answers (diagnosis) leads to what we call an explanation. We compute it by finding a derivation for the answer. An explanation can be formed by looking at the student knowledge and at two separate knowledge sources: the expert knowledge (i.e. the knowledge of the teaching domain that the system typically uses to find the correct solution to a problem) and the buggy knowledge (i.e. information about student mistakes, generally expressed in the form of mal

rules). The kind of explanation built by the system depends on the answer: correct, incorrect and "I don't know". A correct answer requires to find all possible explanations and select a preferred one. An incorrect answer requires to identify all possible explanations by using buggy knowledge and then to select a preferred one. A "1 don't know" answer requires to identify which knowledge the student is lacking, that would enable him to provide a correct answer. Therefore, explanations of "I don't know" answers are given by considering every explanation of the corresponding correct answers. In the rest of the paper we show how this analysis can be phrased in terms of operations on knowledge bases.

Once the system has found an explanation for the student's answer, the representation of the knowledge the system has about the student should be updated accordingly. The above sketched analysis suggests that the system maintains a representation of what it knows *to be known* by the student, as well as of what it knows *not to be known* by the student.

The paper is organized as follows. We first discuss the structure of TTSs in terms of a deductive approach and present a simple meta-level framework based on first-order logic. We then show how the diagnostic behaviour of three well-known TTSs proposed in the literature, namely SEDAF [Aiello and Micarelli, 1990], LMS [Sleeman and Smith, 1981, Sle eman, 1983], and the BUGGY/DEBUGGY system [Brown and Burton, 1978, Burton, 1982], can be rephrased. We finally address some features that can be considered in order to improve the performance of ITSs: the exploitation, on the side of the system, of the informative content of the student ignorance.

2 A deductive approach to the representation of student knowledge

This section describes the meta-level architecture underlying the formulation of the diagnostic principles of ITSs: the fundamental object-level contexts, the meta-language and some basic definitions and axioms shared by any met a-theory.

2.1 Object-level structures

The fundamental object-level structures representing the system's knowledge about the teaching domain and the student's learning status are here provided an abstract description. Such structures are what the meta-theory reasons about.

As already mentioned, we assume that any statement about the teaching domain is represented as a first order formula. Such a simplification is made only for methodological reasons and it does not impose an a priori restriction to the approach. For example, domain knowledge can itself be organized into meta-levels, where heuristic principles and reasoning strategies are represented at higher levels, following [Vivet, 1988]; or else, the cumbersome logical expression of some operations on object-level entities can be avoided by using semantic attachments [Weyhrauch, 1980].

According to the above assumption, the object-level language is a first order language containing constants, predicates and functions denoting corresponding types of objects in the teaching domain. The expert knowledge and the student's beliefs can thus be represented by sets of first order formulae, called contexts.

At the object-level there are two contexts, corresponding to the Expert Module and the Student Model of ITSs: E, that contains formulae denoting correct statements about the teaching domain, and S, containing formulae believed by the student.

In a malrule approach, a "bug catalogue" is also considered: at the object-level there is a context, B, containing formulae corresponding to the incorrect beliefs that could possibly be ascribed to a student. It plays the role of a database which the teacher consults when trying to explain a wrong answer given by the student. Usually, $\mathbf{E} \cap \mathbf{B} = \emptyset$ and $\mathbf{S} \subset \mathbf{E} \cup \mathbf{B}$.

Contexts E and S can be considered as standard logical theories, in the sense that they contain correct logical inference rules (even if possibly incomplete) and proper axioms representing domain knowledge. For the sake of generality, we do not make any assumption on the kind of logic in such contexts, even if, in actual implementations, it is not full first-order logic (for example, S and E may be production systems or Horn clause theories). However, we assume that the two contexts have the same logic. Even if such an assumption, that the student and the expert have the same reasoning capabilities, looks quite strong, it seems necessary in order for the system to be able to tell anything about the student's reasoning in the teaching domain. Context B has no logical inference rule at all: deductions within B make no sense. Hence, the only set of object-level inference rules we consider is that of context E. We do not make any assumption about the consistency of contexts: B is very likely to contain contradictory formulae. The same may happen to S, although its logic is correct.

In the sequel, any finite set Σ of object-level formulae will be considered as a context, as well as the union of any pair of contexts. Formulae in any context will be called axioms of that context. Proofs in any context are built by means of the inference rules in E.

2.2 Meta-Theory

We now introduce the basic meta-language and definitions that will be shared by any meta-theory (the generic meta-theory will be called MT). The meta-language contains suitable constants, functions and predicates allowing the object-level entities to be referred to. Among them, the constants *E, B,S* will be used, to denote respectively contexts E, B and S. Moreover, the language of MT contains a functional symbol allowing the representation of finite sets of object-level formulae and the predicate $\in$, defining the membership relation. MT contains a functional symbol denoting union of contexts, so that, if C_1 and C_2 d e n $\mathbf{C_1}$ e a n $\mathbf{C_2}$, h e n $C_1 \cup C_2$ denotes their union.

The meta-language contains constants denoting object-level symbols and functional symbols allowing the meta-level coding of object-level complex formulae. If a is an object-level formula, $'\alpha'$ denotes the representation

of α at the meta-level, and $mknot('\alpha')$ the representation of the formula $\neg\alpha$. Similarly, if α and β are object-level formulae, $mkand('\alpha','\beta')$ is the representation of the formula $\alpha \wedge \beta$. The meta-level representation of the finite set of formulae $\Sigma = \{\alpha_1, \ldots, \alpha_n\}$ will be denoted either by $'\Sigma'$ or $\{'\alpha_1', \ldots, '\alpha_n'\}$. The symbols $\alpha, \beta, \gamma, \phi, \Sigma$ (possibly with subscripts) will be used as meta-variables, when they are not quoted.

Answers given by the student are coded by the meta-level predicate *answers*(ϕ). We consider only positive answers, since a question requiring a value as an answer, is turned into a relation instantiated with the answered value, and negative answers can be obtained by negating the question.

In order for MT to be an adequate meta-theory for the object-level contexts, the derivability relation in any context has to be definable in MT. As we are interested not only in knowing whether a formula is a theorem of a given context, but also which axioms are effectively used to derive it, the key meta-level formula linking the object-level context C to the meta-theory is the formula *derivablcc{x,y)* with two free variables, such that for any finite set of formulae Σ and formula α, if there exists a (non trivial) proof of *a* in C using exactly the formulae in Σ as axioms, then $\mathbf{MT} \vdash derivable_C('\Sigma','\alpha')$.

We assume that such a notion of derivability excludes trivial cases, i.e. Σ is a minimal set of axioms from which *a* can be derived in $\mathbf{C}$ and Σ is not contradictory. The consistency condition may seem a strong requirement in computational terms, because it is a generally non-decidable predicate. However, in actual systems, the proof procedure used at the object-level usually guarantees both the consistency and the minimality requirements, so that there is no need to check for non triviality of proofs. Or else, as it is the case of the three systems described in Section 3, consistency can be defined in a more constructive way. In fact, the object domain of such systems is formalized as a production system, where each malrule is associated with a correct rule, so that a set of axioms is contradictory only if it contains either both a correct rule and one of its buggy versions, or two buggy versions of the same correct rule. In such cases, if the predicate *buggyversion* defines the relation holding between a rule and one of the associated malrules, consistency can be defined as:

$$\forall\Sigma(consistent(\Sigma) \Leftrightarrow \neg\exists\beta_1,\beta_2(\beta_1 \in \Sigma \wedge \beta_2 \in \Sigma \wedge (buggyversion(\beta_1,\beta_2) \vee \exists\beta(buggyversion(\beta,\beta_1) \wedge buggyversion(\beta,\beta_2)))))$$

Other basic formulae defining object-level relations in MT are the following: *subconjunct*(α_o, α) defines the relation holding between a subconjunct and the whole conjunction; for any context C, $axiom_C(\alpha)$ defines the set of axioms of C.

We assume that the meta-theory contains also simple axioms ruling the relations among contexts, such as, for any context C_1, C2:

$$\forall\Sigma,\phi(derivable_{C_1}(\Sigma,\phi) \Rightarrow derivable_{C_1 \cup C_2}(\Sigma,\phi))$$

3 The diagnostic principles of some existing systems

In this section we show how to express in our meta-language the diagnostic principles adopted by three existing ITSs, namely SEDAF [Aiello and Micarelli, 1990], the two versions of LMS [Sleeman and Smith, 1981, Sleeman, 1983], and the BUGGY/DEBUGGY system [Brown and Burton, 1978, Burton, 1982]. We abstract from some features that are strictly peculiar to a system, in order to point out how the general diagnostic principles can be expressed and compared in our formalism.

The meta-theory of each system contains the predicate *ascribes* representing the result of a meta-level reasoning. Different hypotheses on the student's way of reasoning can be expressed by means of the meta-level axioms defining the predicates *ascribes*-

Let us assume that the student admittedly believes ϕ to be true and that we already know how to find out the set Σ of the axioms in $\mathbf{E} \cup \mathbf{B}$ that best explains ϕ, then the system can conclude that every formula in Σ must be in S. So, each of the meta-theories corresponding to the diagnostic principles of the systems considered below contains the following axiom:

$$(D1)\ \forall\phi,\alpha,\Sigma(answers(\phi) \wedge explains(\Sigma,\phi) \wedge \alpha \in \Sigma \Rightarrow ascribes(\alpha))$$

The hard task is the definition of the predicate *explains.* As expected, the theories we are now going to consider differ fundamentally in the axioms defining the predicate *explains.* However, all of them are patterned in the same style.

3.1 The system SEDAF

SEDAF teaches students to graph mathematical functions by solving for the characteristics of the function (singular points, asymptotes, maxima, etc). It embeds an expert module consisting of a set of correct rules, a database of malrules and a student module consisting of a set of correct rules and malrules.

During each phase in the study of a function, SEDAF asks questions to the student. If the student's response is correct, the system assumes that it has been correctly derived and that the student knows all the facts and rules used in the corresponding deduction built by the expert module. So, a first, axiom defining *explains* is:

$$(E1)\ \forall\Sigma,\phi(derivable_E(\Sigma,\phi) \Rightarrow explains(\Sigma,\phi))$$

This axioms assumes that there is always a single correct way of proving a correct statement. This corresponds to a design choice made in SEDAF, where, like in many other ITSs, the focus of the diagnostic activity is on finding out reasons for wrong answers, while taking the correct ones more or less for granted. A more articulated version of EI could cope with the case of multiple explanations for a correct answer (see for example [Cialdea, 1991]).

If the response provided by the student is incorrect, SEDAF takes the student's solution as a goal, and tries to build a deduction that leads to the goal, using also malrules. During this process the system interacts

with the student in order to discriminate among different inference chains, by asking him to confirm which rule/malrule he has actually used. This is described by the following axiom, where *test student*('α') represents the fact that the student confirms he believes a.

$$(E2)\ \forall \Sigma, \phi (derivable_{E \cup B}(\Sigma, \phi) \wedge \forall \alpha (\alpha \in \Sigma \Rightarrow teststudent(\alpha)) \Rightarrow explains(\Sigma, \phi))$$

These formal statements compactly describe the diagnostic principles of SEDAF. In addition, they point out an incoherence in their application: SEDAF assumes that correct responses have been correctly derived (axiom EI); but this principle is not recursively applied when building subproofs of correct statements used to reach a wrong conclusion, as axiom E2 shows; in fact, in such cases, the student is questioned to confirm his use of any rule and not malrules only.

3.2 The system LMS

LMS models students who are learning to solve linear algebraic equations in one variable. The domain knowledge is represented as a production system and both the expert and the student's procedure to solve a problem are represented as ordered sets of production rules. The order in which rules are applied may be relevant; in fact, precedence of operators is modelled by the ordering of rules (e.g., compute products before sums), rather than by modifications of the rules' preconditions. Bugs are represented either as variants of correct rules or as an incorrect order of application of correct rules. The fact that control errors are represented by an incorrect ordering of rules is irrelevant, for our purposes, so we assume that the object domain of LMS is represented as an unordered set of production rules, where control is expressed by additional requirements for the applicability of relevant rules and corresponding malrules drop such additional requirements.

In LMS-1 [Sleeman and Smith, 1981], a first version of LMS, the student is presented with a sequence of problem sets of growing complexity; each of them requires the application of one new rule and it is assumed that if the student has applied a given sequence of rules (either correct or incorrect) in solving a problem set, he will apply the same rules, plus one more, to the probems at the next level. So, the student model is incrementally built: once a partial student model has been inferred, the subsequent one is obtained by the addition of the next correct rule or one of its incorrect variants.

It is also assumed that, the set of problems presented to the student at each level is discriminatory, so that it can never happen that two different extensions of the existing student model explain the student's answer.

In terms of the meta-language proposed in this work, the diagnostic meta-theory of LMS-I is characterized by the following axiom:

$$(E3)\ \forall \alpha, \phi (axiom_{E \cup B}(\alpha) \wedge \exists \Sigma (\alpha \in \Sigma \wedge derivable_{S \cup \{\alpha\}}(\Sigma, \phi)) \Rightarrow explains(\{\alpha\}, \phi))$$

Note that ϕ here represents the student's answers to a set of problems (at a given level).

The second version of LMS [Sleeman, 1983] takes into account the fact that, for instance, the ability to apply a correct rule may depend on the complexity of the problem. LMS-II thus rejects the assumption that partial student models are never contradicted. In order to reduce the search space of all possible (ordered) subsets of the rule and malrule set, in an off-line phase, the system generates a complete and non reduntant set of models, retaining only those which give unique results with the predefined problem set. Proper subsets of a given model are considered as different models, because they pertain to a different level. So, in particular, the system has a set of expert models, one for each level. In the on-line phase, the student's performance is compared against such models. It is still assumed that the set of problems for each level is adequate to discriminate among all possible models at that level.

The on-line phase of LMS-II could be described by introducing a set of Expert and Buggy theories, $\mathbf{C_1}, \ldots, \mathbf{C_n}$, with $\mathbf{C_i} \subset \mathbf{E} \cup \mathbf{B}$, which are assumed to result from the off-line phase. In that case, a set of rules Σ explains the answer ϕ given by the student to a problem set if, for some i, ϕ is a theorem of $\mathbf{C_i}$ and Σ is the set of axioms of Cj. However, if we abstract from the computational problems dealt with in the off-line phase, we see that the subset of $\mathbf{E} \cup \mathbf{B}$ corresponding to the model entailing ϕ can equally well be seen as the set of axioms used to prove ϕ in the context $\mathbf{E} \cup \mathbf{B}$. So, the meta-theory embedding the diagnostic principles of LMS-II is characterized by the following axiom for explains:

$$(E4)\ \forall \Sigma, \phi (derivable_{E \cup B}(\Sigma, \phi) \Rightarrow explains(\Sigma, \phi))$$

Note that EI is a derivable from E4.

Of course, the simplicity of E4 is due to the assumption that ϕ is a discriminatory set of problems, so that, in practice, there is not a real diagnostic problem (the emphasis of LMS is rather on the reduction of the search space of possible models).

3.3 BUGGY and DEBUGGY

Brown and Burton [Brown an d Burton, 1978] proposed "the Buggy model", a descriptive theory of bugs, where students' mistakes in simple procedural skills are seen as symptoms of local modifications (bugs) to the correct procedure. The model was developed and explored in the domain of place-value subtraction.

The object domain is represented by a set of goals and methods for satisfying these goals, linked together in a procedural network, where links represent goal-subgoal relations. Corresponding to each goal, there are correct and incorrect methods. A student's incorrect behaviour can be reproduced by replacing some correct methods by one of their buggy variants in the basic (correct) network. The same structure can of course be represented by other means and, for our purposes, we consider it as represented by a set of rules.

A first naive diagnostic system based on the buggy model compares the student's answers on a given set of problems with the output of each network obtained by replacing a single method with one of its bugs. So, if one single bug can explain the student's answers, it is assumed to be possessed by the student. BUGGY

diagnostic system assumes, like SEDAF, that, if part of the solution is correct, then the student achieved it correctly; and, like LMS, that diagnosis is fired on the results of a discriminatory test: there can be at most one bug explaining the student's behaviour on the entire test.

Abstracting from some details, BUGGY met a-theory is characterized by:

$$(\mathrm{E5})\ \forall\Sigma,\phi(\exists\beta(axiom_B(\beta) \wedge derivable_{E\cup\{\beta\}}(\Sigma,\phi)) \Rightarrow explains(\Sigma,\phi))$$

Note that EI is derivable from E5 and E5 is derivable from E4.

DEBUGGY [Burton, 1982], a development of BUGGY, is a much more sophisticated diagnostic system, that takes into account both the fact that more than one bug can cause the student's errors and the fact that sometimes there is no hypothesis explaining the student's behaviour completely, so that the system has to find out the model that best explains it. The diagnostic technique in DEBUGGY is quite heavy and we shall only sketch it here, just to show that it can be expressed in the same style as the others.

DEBUGGY assumes that simple bugs that form multiple ones can also be detected individually. So, an initial set of hypotheses is generated, containing all single bugs that explain at least one of the student's incorrect answers (for effectiveness reasons, the initial set of hypotheses is then reduced by eliminating bugs subsumed by others and bugs that can be considered as casual matchings. We shall not describe here the criteria used for the reduction). To represent this fact, we emulate the dynamic construction of a new context, $\mathbf{HS}_\phi$, the initial set of hypotheses containing simple bugs, by defining:

$$\forall\beta(axiom_{HS_\phi}(\beta) \Leftrightarrow axiom_B(\beta) \wedge \exists\Sigma,\phi_0(subconjunct(\phi_0,\phi) \wedge derivable_{E\cup\{\beta\}}(\Sigma,\phi_0) \wedge \beta\in\Sigma))$$

This definition allows us to define the predicate *derivable* $_{E\cup HS_\phi}$ in the obvious way.

The elements of the set, of hypotheses are then combined and compound bugs (up to a given depth) are added to the set of hypotheses if they explain more of the student's behaviour than their constituents. A compound bug that is a good candidate to explain ϕ can be defined as a set of axioms in $\mathbf{E}\cup\mathbf{HS}_\phi$ that entails a subconjunct of ϕ and is more explicative than any of its subsets:

$$\forall\Sigma,\phi(candidate(\Sigma,\phi) \Leftrightarrow \exists\phi_0(subconjunct(\phi_0,\phi) \wedge derivable_{E\cup HS_\phi}(\Sigma,\phi_0)) \wedge \forall\Sigma'(\Sigma'\subset\Sigma \Rightarrow explainsmore(\Sigma,\Sigma',\phi)))$$

Here, $\subset$ stands for proper set inclusion; the predicate *explainsrnore* (Σ,Σ',ϕ) defines the fact that, if E U £ entails the subconjunct ϕ_0 of ϕ and $\mathbf{E}\cup\mathbf{\Sigma'}$ entails the subconjunct ϕ_1 of ϕ, then ϕ_1 is a proper subconjunct of ϕ_0. If we abstract from computational problems, in the above definition $\mathbf{HS}_\phi$ can be replaced by B.

After the generation of compound bugs (i.e. good candidates), each bug is compared with the others, by taking into account not only the number of predicted answers, but also the number and type of mispredictions and a simplicity criterion. Finally, the best of them is chosen as an explanation of the student's behaviour. If we assume that the predicate *wins*(Σ,Σ',ϕ) represents the fact that Σ is a better candidate than Σ' according to the above quantitative and qualitative criteria, then diagnosis in BUGGY is performed according to the following:

$$(\mathrm{E6})\ \forall\Sigma,\phi(candidate(\Sigma,\phi) \wedge \forall\Sigma'(candidate(\Sigma',\phi) \Rightarrow wins(\Sigma,\Sigma',\phi)) \Rightarrow explains(\Sigma,\phi))$$

Various considerations can be made after the analysis we have just performed. The most relevant one is that the diagnostic principles followed by the three ITSs can be accommodated in a unique framework that is independent of the domain and largely independent of the particular diagnostic strategy adopted by each system. The three systems essentially differ in the criteria used to choose the best explanations for given answers. This last aspect is somewhat more domain dependent and often neglected. It deserves a more analytical investigation and the use of a formal tool such as the one we propose could be of great value and effectiveness.

4 Representation and use of "I don't know" answers

Most existing intelligent tutoring systems concentrate their diagnostic effort on the justification of incorrect answers provided by students, in terms of bugs present in their procedures or misconceptions in their knowledge. Correct answers are rarely analysed in some depth. "1 don't know" answers, even when admitted, only trigger the system to give help to the student, and no information about the student's knowledge is inferred from them. The logical treatment presented in the previous sections accommodates the diagnostic principles coping with both correct and incorrect answers provided by the student. In this section we briefly show how a tutoring system can gather information about what the student does not know and how it can be used. So, we assume that the student is given the possibility to tell the system "I don't know". Of course, as shown in the sequel, information about the student's ignorance can also be obtained indirectly, by met a- reasoning.

4.1 An explicit representation of the student ignorance

Both during the diagnostic and teaching-remediation phase, the system has to access what the student knows and what he doesn't know. It must, be remarked, however, that the system cannot be assumed to have complete knowledge about the student beliefs[1]. So if a formula α is not contained in S, the system cannot infer that the student does not believe α. It only means that the system does not know whether the student believes α or not. In other words, S contains what the system knows that the student believes. For this reason, we add

[1]The system's knowledge may also be partially incorrect, in the sense that a previous hypothesis can reveal to be inconsistent with a new assertion of the student. This problem will not he touched upon within this work.

a fourth component to the object level structure: context U, containing what the system knows the student does not know[2].

Considering the nature of the diagnostic task in this wider view, two main cases can be distinguished:

1. The system has to discover the reasons why the student believes something (either correctly or not);
2. The system has to explain where the student's ignorance of some facts originates from.

In the first case the system has to deal with an abductive task: if the student believes ϕ and $\pi_1, \ldots, \pi_n$ are all the possible (non- trivial) proofs of > that can be constructed in $\mathbf{E} \cup \mathbf{B}$, then only one of them has to be chosen. Let's say that π_j is the preferred one; if $\alpha_1, \ldots, \alpha_k$ are all the axioms of E U B used in π_j, then it can be concluded that the student believes all of $\alpha_1, \ldots, \alpha_k$. Point 1 above will be called the abductive aspect of diagnosis.

The second case represents the dual situation. If the student does not believe ϕ, and if $\pi_1, \ldots, \pi_n$ are all the possible proofs of ϕ that can be constructed in $\mathbf{E} \cup \mathbf{B}$, then the system can reasonably conclude that the student is not able to construct any of them. In this case there is no abductive problem, being the system's task-purely deductive. However, there is again a choice problem, at the level of the subsets of axioms used in the proofs: if $\alpha_1, \ldots, \alpha_k$ are all the axioms of $\mathbf{E} \cup \mathbf{B}$ used in π_j, for $1 \leq j \leq n$, then surely the student does not believe the conjunction $\alpha_1 \wedge \ldots \wedge \alpha_k$. But this does not mean that the student lacks all of $\alpha_1, \ldots, \alpha_k$, so an extra investigation should be performed in order to single out which of them the student actually does not know or possibly believes to be false. This second aspect of diagnosis will be called deductive.

It is worth noting that the abductive aspect of diagnosis is treated very roughly in SEDAF (just ask the student what he thinks) and it is absent in both versions of LMS and in BUGGY. The difficulty of finding good criteria for abductive diagnosis supports our conviction that it is important to have a formal language to state them clearly and explicitly. Note that standard techniques for abduction only consider sets of assumptions, while here we are interested in general axioms used in the derivation.

4.2 The deductive aspect of diagnosis

Answers given by the student are now coded by the meta-level predicate *answers*(ϕ, t), where t is either the constant *yes* or *dontknow.* The axioms dealing with the deductive aspect of diagnosis allow the system to conclude that the student does not believe something, i.e. *ascribe* $_U(\alpha)$.

[2]Usually, existing ITSs identify the student's ignorance with the systems's ignorance about the student. Mizoguchi [Mizoguchi *et* al., 1988] recognizes the importance of the distinction, by use of four truth values for statements a about the object domain: true (the student knows α), false (the student knows $\neg\alpha$), unknown (the student does not know α), and fail, which is a meta-truth attribute, referring to the system's knowledge (the system cannot attribute any of the previous three values to α).

The first axiom states that if the student answered "1 don't know" to a question ϕ, then he knows no way to derive either ϕ or the negation of ϕ. So, axiom D2 is:

$$\begin{aligned}(\mathrm{D2})\ \forall\phi,\alpha(&\mathit{answers}(\phi,\mathit{dontknow})\wedge\\ &\exists\Sigma((\mathit{derivable}_{E\cup B}(\Sigma,\phi)\vee\\ &\quad \mathit{derivable}_{E\cup B}(\Sigma,\mathit{mknot}(\phi)))\wedge\\ &\quad \mathit{conjunction}(\Sigma,\alpha))\\ &\Rightarrow \mathit{ascribe}_U(\alpha))\end{aligned}$$

where *conjunction*(Σ, α) defines the relation between a set $\Sigma = \{\alpha_1, \ldots, \alpha_k\}$ and the formula $\alpha = \alpha_1 \wedge \ldots \wedge \alpha_k$.

Note that a is a conjunction of axioms in $\mathbf{E} \cup \mathbf{B}$, so axiom D2 can lead to conclude that the student does not have a misconception. Even if such information is usually of no importance for the teaching and remediation activities, it can be useful when the student model is used during the diagnostic process itself. In fact, knowing that the student cannot have used a given misconception can help to reduce querying/testing the student about her or his beliefs, when an incorrect answer has to be explained.

The second axiom of the deductive aspect of diagnosis concludes that the student does not believe something from the fact that he believes something else. It states in fact that if the student believes ϕ to be true, then he does not know any way to derive the negation of ϕ. In other words, axiom D3 states a minimal consistency hypothesis about the student's knowledge.

$$\begin{aligned}(\mathrm{D3})\ \forall\phi,\alpha(&\mathit{answers}(\phi,\mathit{yes})\wedge\\ &\exists\Sigma(\mathit{derivable}_{E\cup B}(\Sigma,\mathit{mknot}(\phi))\wedge\\ &\quad \mathit{conjunction}(\Sigma,\alpha))\\ &\Rightarrow \mathit{ascribe}_U(\alpha))\end{aligned}$$

The problem of reducing the size of an unbelieved conjunction has been addressed in [Cialdea, 1991].

The explicit representation of what knowledge the student has shown to lack can guide the choice of a suitable teaching action. For instance, if the system realized that the student has not mastered some basic abilites yet, it tries to remediate before going on with more complex concepts and problems.

Information about what the student does not know can also help in the diagnostic task. For example, if the student believes ϕ to be true and ϕ can be derived using exactly the axioms in $\Sigma \subset \mathbf{E} \cup \mathbf{B}$ then Σ can be considered as a good explanation of ϕ only if it does not contain any statement that the system has checked to be unknown to the student. So, the following requirement could be added to the preconditions of any axiom enabling a conclusion of the form *explains*(Σ, ϕ):

$$\neg\exists\alpha(\alpha \in \Sigma \wedge \mathit{axiom}_U(\alpha))$$

A complete example showing how to state axioms for the abductive aspect of diagnosis that take into account such principle can be found in [Cialdea, 1991].

5 Conclusions

In this paper we have presented a formal framework for the description of diagnostic principles in ITSs. It consists in a set of object level (first order) theories along with a meta-theory, where reasoning about reasoning is

described. Our formal framework has been used to rationally reconstruct the diagnostic process of some well known existing ITSs and to propose further extensions to the diagnostic capabilities of ITSs, namely by allowing them to make also a sense out of "1 don't know" answers provided by a student, i. e. answers that point out student's ignorance.

There are several good reasons supporting the usefulness of a formal approach to the problem of specifying general principles for student modelling in educational systems. First, the meta-language we have proposed is a design tool for ITSs: it can be used as a high level specification language, when conceiving the student modelling features of a concrete system.

Second, the high level specifications thus obtained can be directly implemented in a (logic) programming language; the prototypes constructed that way help to test the validity of the specifications and give the possibility of modifying them rapidly, at low cost, as described in [Cialdea *et ai,* 1990].

Furthermore, the proposed framework can be an instrument for a comparative study of existing systems: the analytical study of the student modelling guidelines adopted in existing systems leads to point out enhancements and enlargements of such principles. To support this last claim, in the paper we have shown how the analytical study and logical formulation of the principles informing meta-reasoning in the student modelling component of educational systems can lead to take into account something new: the possible uses of the student model during the diagnostic process itself, in order to reduce querying/testing the student and the possible uses of an explicit representation of what the system knows the student does not know.

Our proposal can be extended and refined in many directions. As a first example, different strategies for treating the set of malrules can be easily accommodated. For instance, it would be easy to organize misconceptions into subsets, possibly ordered according to a likelihood criterion. A further important issue that can be addressed is the treatment of student's contradictory beliefs. And more: what happens if the student is allowed to have incomplete and possibly incorrect reasoning capabilities?

It is of great importance to deal with such issues in a uniform way, by exploiting both the theoretical and software engineering advantages of a logic specification.

Acknowledgements. The work reported here has been partially supported by Esprit Basic Research Action COMPULOG and by Progetto Finalizzato "Sistemi Informatici e Calcolo Parallelo" of the Italian National Research Council.

References

[Aiello *et* a/., 1991] Luigia Aiello, Daniele Nardi, and Marco Schaerf. Reasoning about Knowledge and Reasoning in a Meta-Level Architecture. *International Journal on Applied Intelligence,* in press.

[Aiello and Micarelli, 1990] Luigia Aiello and Alessandro Micarelli. SEDAF: An Intelligent Educational System for Mathematics. *Applied Artificial Intelligence - An International Journal,* 4: 15-37, 1990.

[Brown and Burton, 1978] John S. Brown and Richard R. Burton. Diagnostic Models for Procedural Bugs in basic Mathematical Skills. *Cognitive Science,* 2: 155-191, 1978.

[Burton, 1982] Richard R. Burton. Diagnosing Bugs in a simple Procedural Skill. In D.II. Sleeman, J.S. Brown, editors, *Intelligent Tutoring Systems,* pages 157-183, Academic Press, London, 1982.

[Cialdea *et al,* 1990] Marta Cialdea, Alessandro Micarelli, Daniele Nardi, James C. Spohrer, and Luigia Aiello. A Rational Reconstruction of the Diagnostic Process in Intelligent Tutoring Systems. In *Proc. of Pacific Rim International Conference on Artificial Intelligence PRICAI '90,* Nagoya, Japan, November 1990.

[Cialdea, 1991] Marta Cialdea. Meta Reasoning and Student Modelling. In E. Costa, editor, *New Directions for Intelligent Tutoring Systems,* NATO ASI Series F, Springer-Verlag, in press.

[Mizoguchi *et ai,* 1988] Riichiro Mizoguchi Mituru Ikeda, Osarnu Kakusho An Innovative Framework for Intelligent Tutoring Systems. In P. Ercoli and R. Lewis, editors, *Artificial Intelligence Tools in Education,* North-Holland, Amsterdam, 1988.

[Self, 1988] John A. Self. Student models - what use are they? In P. Ereoli and R. Lewis, editors, *Artificial Intelligence Tools in Education,* North-Holland, Amsterdam, 1988.

[Self, 1991] John A. Self. Computational Mathetics: the missing link in Intelligent Tutoring Systems research? In E. Costa, editor, *New Directions for Intelligent Tutoring Systems,* NATO ASI Series F, Springer-Verlag, in press.

[Sleeman and Smith, 1981] Derek H. Sleeman and M.J. Smith. Modelling Students' Problem Solving. *Artificial Intelligence, \G:* 171-187, 1981.

[Sleeman, 1983] Derek H. Sleeman. Inferring Student Models for Intelligent Computer Aided Instruction. In R.S. Michalski, J.G Carbonell, P.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach,* pages 483-509, Morgan Kaufmann, Palo Alto, California, 1983.

[VanLehn, 1988] Kurt VanLehn. Student Modeling. In M. C. Poison, J. J. Richardson, editors, *Foundations of Intelligent Tutoring Systems,* pages 55-78, Lawrence Erlbaum Associates Publishers, Hillsdale, New Jersey, 1988.

[Vivet, 1988] Martial Vivet. Reasoned Explanations need Reasoning on Reasoning and Reasoning on the Student. In P. Ereoli and R. Lewis, editors, *Artificial Intelligence Tools in Education,* pages 121-128, North-Holland, Amsterdam, 1988.

[Weyhrauch, 1980] Richard W. Weyhrauch. Prolegomena to a Theory of Mechanized Formal Reasoning. *Artificial Intelligence,* 13: 133-170, 1980.