Plan inference and student modeling in ICAI

Y.M. Visetti
CNRS / LIMSI . Universite Paris-Sud
(Batiment 508) ORSAY 91406 FRANCE

P. Dague
IBM Research Center . 36 av. R. Poincare
PARIS 75016 FRANCE

ABSTRACT

This paper adresses the problem of building user models within the framework of Computer Assisted Instruction (ICAI), and more particularly for systems teaching elementary arithmetic or algebra. By "model building" we mean the understanding of the student's performances, as well as a global description and evaluation of his/her ability (competence), including a representation of some errors.

As an application domain we have here retained the learning of "calculus" in the field of rational numbers, as an intermediate area between arithmetic and algebra. The aim of our system is to control the way in which the pupil solves exercises.

In the light of the particular nature of the chosen application, the main points to be stressed are the following :

- calculations are described as plan generation and execution ; consequently the student's modelling consists primarily in plan inferencing
- the system takes into account the non deterministic nature of the task, and recognizes valid variants of expert calculation plans
- numerous errors are detected and categorized
- the system accepts that the student write the calculations in a more or less elliptic manner ; whenever ambiguities occur, the student is precisely asked about implicit steps of his calculations, and the system uses the answers given to reduce the uncertainties
- a global model of the student is generated, which incorporates observations and appreciations ; this model, in turn, determines the subsequent interpretations.

All these questions are discussed both at the fundamental and the methodological levels.

0. INTRODUCTION

It is generally recognized (see [Sleeman et al. 82] or [Zissos 85]) that an Intelligent Tutoring System (ITS) should be composed of at least 4 components respectively in charge of :

- the domain knowledge to be taught
- the understanding of the student's performances and the progressive building of a certain image (or model) of his/her competence
- the pedagogical rules
- the control of the dialogue with the student (including linguistic and graphic interface).

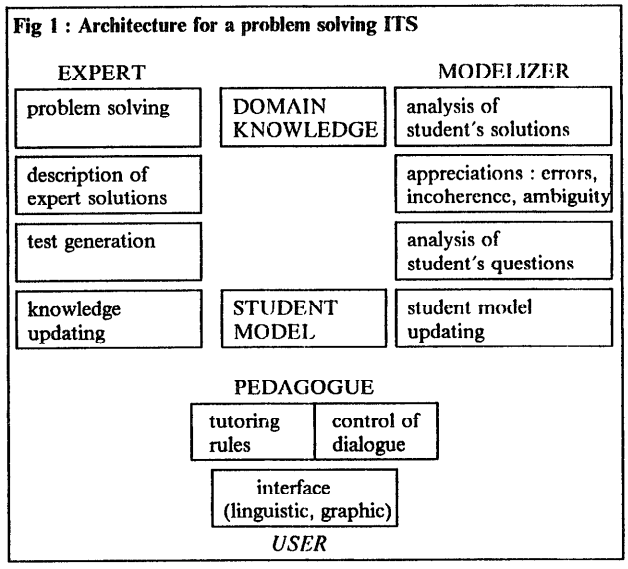
A minimal architecture for a problem solving oriented ITS is presented in Figure 1.

The following two examples will illustrate all these functions (and particularly the modelling activity) of an ITS teaching the calculus in the field of rational numbers. We assume here that the student has already a basic knowledge of fractions and operations on fractions. The aim of the system is then to test the student's ability to perform operations at a purely technical/procedural level.

Exercise 1

System1 : Compute 4/6 + 5/7 !
User1 : ...I find : 29/21

Fig 1 : Architecture for a problem solving ITS



S2 : that's right, but show me how you have simplified
U2 : 4/6 + 5/7 = 58/42 = 29/21
S3 : that'right, but you could have simplified before adding up, as follows :

4/6 + 5/7 = 2/3 + 5/7 = (2x7 + 3x7)/(3x7) = 29/21

This example shows that the system must not only be capable of generating (like in S1) and solving exercises. It also must accept from the pupil, and of course understand, more or less developed solutions (compare answers U1 and U2). Because of the non deterministic nature of the task, uncertainties unavoidebly occur, concerning certain implicit steps in the actual calculations of the pupil. The system must then be able to recognize the possible alternatives, and if it deems it important, to ask the pupil a question focusing on the specific step, or task involved (see S2). The system will then use the new answer (U2) in order to reduce its uncertainties and thus to trigger off the appropriate comments (see S3).

Exercise 2

S4 : compute 4/6 + 2/5 , and don't forget to simplify as soon as possible !
U3 : 4/6 + 2/5 = 2/3 + 2/5 = 2/(3x5) = 2/15
S5 : let's take your first equality : that's good, you have properly simplified this time
now let's consider the second equality : the addition of 2/3 + 2/5 is wrong ! There is nothing particular here, you should have done this addition exactly in the same way as you did in Exercise 1. You should have written :

2/3 + 2/5 = (2x5 + 3x2) / (3x5) = 16/15

It is to be noted that, when formulating the second exercise, the system takes into account the previous performances of the pupil (see S4), and it refers to them in its comments (S4 and S5). On the other hand, the pupil has made a mistake, possibly because of the particular pattern of the fractions which he had to add up (after simplification both fractions have the same numerators !). Nevertheless the system recognizes it as an "attempted addition" carried out with a deviant procedure (like : a/b + a/d --> a / (bxd)), so that it can categorize it in its comments as being a "wrong addition" (see S5).

The problem arises now of knowing what memory the system should keep of these performances, how it uses this memory in order to build up a global image of the pupil's competence, and finally how the global image is to influence the interpretations of future calculations.

We shall now proceed to explain some partial answers among all those which are required to build up a system capable of reacting exactly as stated in the previous examples. We underline that we are only concerned here with the problem of the student's modelling, which we view both as a local and a global activity : analysis of the performances and synthesis (description and evaluation) of the pupil's competence. We do not deal with the process of dialogue at the level of natural language generation or comprehension ; we shall only indicate how calculations must be analyzed in order to permit clarification dialogues such as those presented in the above examples.

1. AN APPLICATION DOMAIN

We tackle the problem of student's modelling within the framework of a particular area : calculus in the field of rational numbers. Although this calculus may seem simple, it nevertheless presents important processing difficulties, both for the learning pupil and for the teaching system. First the calculation processes are not completely deterministic : for instance, one can perform an addition of fractions either before or after possible simplifications. Secondly objects and rules are not realy accessible outside the symbolic (and not only numeric) framework in which they are defined. Thirdly the pupil is given the opportunity of making a number of stereotyped errors, which an ITS cannot completely ignore (as shown in the Introduction).

To reflect the non deterministic nature of the calculations, we have to specify which tasks are obligatory, and which are optional. Optional tasks (like simplifications, factorizations, etc.) may be ignored or postponed without entailing a complete failure of the main task which consists in evaluating the proposed expressions. However whenever an optional task has not been performed while it was possible, the system will have to notice this significant fact. On the other hand, obligatory tasks are those which must mandatorily be performed when the right moment comes : the process of reducing the expressions would otherwise be blocked. For instance if we want to reduce a sum of two fractions, we have to perform their addition ; or if we want to reduce a ratio A/B we must reduce A into A', B into B', and only then A'/B' into the final answer (but possible simplifications of the successive ratios can be performed or not at different moments).

We turn now to the problem of errors. One can find in an article by Matz ([Matz 82]) a list of errors commonly made by beginners in elementary algebra. As an example :

1a $\frac{AX + CY}{BX + DY} = \frac{A + C}{B + D}$; $\frac{A}{BX} + \frac{CX}{D} = \frac{A}{B} + \frac{C}{D}$

1b $\frac{A + C}{B + D} = \frac{A}{B} + \frac{C}{D}$

1c $\frac{A}{B} + \frac{A}{C} = \frac{A}{B + C}$; $\frac{A}{B} + \frac{A}{C} = \frac{A}{B \times C}$

The same type of errors is to be found currently in the calculus of rational numbers Let's take examples similar to those quoted above :

2a $\frac{2 + 5}{5 + 3} = \frac{2}{3}$; $\frac{2}{5} + \frac{5}{3} = \frac{2}{1} + \frac{1}{3}$

2b $\frac{2 + 5}{3 + 7} = \frac{2}{3} + \frac{5}{7}$

2c $\frac{2}{3} + \frac{2}{5} = \frac{2}{3 + 5}$; $\frac{2}{3} + \frac{2}{5} = \frac{2}{3 \times 5}$

The similarity of these errors is not surprising, since the pupil facing the "concrete" numerical expressions, tries to recognize them as instances of general symbolic patterns which he thinks he can transform according to some general rules. This means that certain situations, which we can legitimately describe in a formal symbolic manner (like in 1a, 1b, 1c), induce some pupils to "apply" non valid calculation methods : a wrong simplification like in 2a, a strange splitting like in 2b, or a wrong performance of the addition like in 2c. Some of these actions may be considered as wrong executions of legitimate tasks (which could be otherwise properly performed) ; other actions must be considered as the execution of totally illegitimate "tasks" (which should in no case whatsoever be performed). Actually this categorization into legitimate and illegitimate tasks is up to a certain point arbitrary. It essentially depends upon the manner in which the system will use it in its comments : sometimes it may be preferable to indicate to the pupil the similarity between his errors and right procedures ; some other times it is better to consider them as thoroughly absurd.

Several authors (see [Brown 78/80], [Sleeman 84], [Resnick et al. 85]) have endeavoured to explain the psychological nature of these erroneous cognitive processes. We shall not go here into this aspect of the matter. In our system we use a purely formal/procedural description of errors, made in the same style as the description of valid processing rules.

2. KNOWLEDGE REPRESENTATION AND THE GLOBAL MODEL OF THE STUDENT

To sum up, calculations are for the system the succession of a certain number of tasks carried out on various expressions. This means that we adopt the plan paradigm in order to describe their execution (quite in the same line as [Genesereth 82]). When the system analyzes one of the equalities, whether valid or not, stated by the pupil, let us say E1 = E2 , it infers one or may be several plans which, applied to E1, result in E2.

We call *context* a pair composed of a *task* ascribed and a *situation* where the task is to be applied. In fact in our system the situation is completely specified by formal characteristics of the processed expressions, such as their symbolic pattern. In each context (understood as task + situation), occurring during the analysis of an equality, the system looks into the global model of the student where all the possible methods for this context are represented. It selects some of these (according to certain heuristics which we shall explain later), and executes them. Of course, due to the recursive nature of plans, some of these methods call upon the execution of several other tasks, while other methods consist of one single final procedure which the system executes whitout analyzing it further.

This process is non deterministic in that, in a given context, all possible methods will be tried. If none is successful, or if the heuristic does not allow to test any method at all, the system reports failure in the case of an obligatory task ; but for an optional task, the process will go on and the next task in the plan will be examined. Furthermore, execution contexts

for a given task can be ordered according to the generality of their input and output filters, so that execution methods are inherited along this *particularization* link.

There is now an abundant AI literature about plan generation and inference. The representations we have used are simple, and the best we can do is to give now some examples. For more fundamental informations on plan representation, the reader is invited to consult [Allen 80] or [Charniak and Mc Dermott 85].

Figure 2 shows the general form of a context frame. The slots of the frame define the task to be executed and the relevant situation, ie. the constraints ruling the input and output expressions (the latter information being particularly useful for plan inference). They also specify all valid methods on the one hand, and on the other hand all the possible methods (valid or not) which might be used by the pupil. Each of these methods is assigned a certain level from 1 to 4, the signification of which is as follows :

- level 1 : in the course of the most recent occurrences of the context, the method has been predominantly used by the pupil among all those which are declared for that context
- level 2 : the method has been occasionaly used in the course of the most recent occurrences
- level 3 : the method has been used, but not recently
- level 4 : the method has never been used .

These "most recent occurrences" of the context define the scope of the short- term memory of the system (for us this scope has been set at 5 occurrences). Level 3 represents the long-term memory of the system. If the context has not yet occurred for a given student, levels are however assigned to the possible methods ; but they only represent the general expectation of the system concerning a typical student.

The distribution of all possible methods in four levels is immediately obtained from the list (with repetition) of the methods the most recently used by the pupil. This list is the value to be found in the OCCURRENCE-TRACE slot of the context frame. Another slot, called LAST-CHANGE, contains an integer, which is incremented each time the pupil uses a method belonging to levels 1 or 2. If he uses some other method, this counter is reset at value 0.

Furthermore, in order to be able to trigger off its tutoring rules, the system must also find in the global model some qualitative evaluations of the pupil's behavior in each context. For that reason the slot COHERENCE contains an association list pairing session identificators with some appreciations. Briefly the coherence is appreciated as being all the higher as there are fewer methods (valid or not) declared at levels 1 or 2 (contexts should be defined with enough accuracy so that this be meaningful). A similar information is given in the QUALITY slot where used methods are evaluated on the basis of their validity.

Figure 3 presents some typical contexts concerning the reduction of a sum of 2 fractions.

The context frames set forth above are part of the model of the student, because they are progressively individualized. But the model also contains other frames which we shall mention here briefly. The pupil's competence is tested by numerous exercises which are generated from types. Each type addresses particular contexts, and tries to avoid some other ones : for instance one can test the addition of fractions with or without the possibility of simplifications. For each exercise type the global model contains various counters such as the number of exercises already done, the ratio of successes of the pupil, etc. Lastly the system produces for each session a frame containing, among other slots, the list of the appeared contexts, and the list of contexts where the pupil's behavior has changed during the session.

In summary the "student's model" contains, in a redundant way, local (episodic) and global (qualitative) information about the pupil's behavior, which is observed and evaluated

Fig. 2 : representation of a task execution context

< context-name >
  TASK : < task-name >
  TYPE : obligatory/optional
  INPUT-FILTER : < pattern of exp > < additional conditions >
  OUTPUT-FILTER : < pattern of exp > < conditions >
  VALID-METHODS : a list of < methods >
  METHODS : a list of pairs < level,method >
    where < level > = 1/2/3/4
    and each < method > has the following format :
      ((< method-name > , < input > , < output >) :
        (< task1 > , < input1 > , < output1 >),
        ...
        (< taskn > , < inputn > , < outputn >))
  OCCURRENCE-TRACE : list of < method-name >
  LAST-CHANGE : < integer >
  COHERENCE : a list of < session-identificator,appreciation >
  QUALITY : a list of < session-identificator, appreciation >
    where < appreciation > = good/average/bad ...

Fig.3 : some typical contexts for the sum of 2 fractions

Context68 (a context for the reduction of a pair of fractions)
  TASK : TRANSFORM
  INPUT-FILTER : a/b ope c/d ,
    with operator(ope), fraction(a/b), fraction(c/d)
  OUTPUT-FILTER: result ,
    with fraction(result) or integer(result)
  METHOD : ((level1,METHOD1))
    whith ((METHOD1, a/b ope c/d , result) :
      (SIMPLIF, a/b , a'/b') (SIMPLIF, c/d , c'/d')
      (CROSS-SIMPLIF, a'/b' ope c'/d' , a"/b" ope c"/d")
      (OPERATE, a"/b" ope c"/d" , num/den)
      (EVAL, num , n) (EVAL, den, d)
      (SIMPLIF , n/d , result))

Context 31 (a context for the simplification of a fraction)
  TASK : SIMPLIF
  TYPE : OPTIONAL
  INPUT-FILTER : a/b , fraction(a/b)
  OUTPUT-FILTER : res , fraction(res) or integer(res)
  VALID-METHODS : SIMP , NON-EXEC
  METHODS : ((level1,SIMP) (level1,NON-EXEC))
    where SIMP(a,b) is the name of the simplification procedure for a pair (a,b) and NON-EXEC is the "empty" procedure representing non-execution of optional tasks

Context32
(a context for the erroneous cross-simplification of a pair of fractions)
  TASK : CROSS-SIMPLIF
  TYPE : OPTIONAL
  INPUT-FILTER : a/b + c/d , fraction(a/b) , fraction(c/d)
  OUTPUT-FILTER : ...
  VALID-METHODS : NON-EXEC
  METHODS : ((level1,NON-EXEC) (level4,CROSSIMP))
    where CROSSIMP(a,b,c,d) is the name of the procedure (here erroneous) for "cross-simplifications" of a quadruplet (example : 2/9 + 6/5 = 2/3 + 2/5 !)

Context6 (a context for the addition of 2 fractions)
  TASK : OPERATE
  TYPE : OBLIGATORY
  INPUT-FILTER : a/b + c/d , ...
  OUTPUT-FILTER : ...
  VALID-METHODS : A1
  METHODS : ((level1,A1) (level4,D1) (level4,D2) ...)
    where A1(a,b,c,d) = (ad + bc) / (ad)
      D1(a,b,c,d) = (a + b) / (c + d)
      D2(a,b,c,d) = (a + b) / (cd) ...

context by context. The updating of this model takes place at different moments of a session.

3. PLAN INFERENCE

When confronted with an equality E1 = E2 asserted by the pupil, the system first examines wether the equality is valid. To do that, the plan interpreter executes on the expression E1 a general TRANSFORM task, by selecting only those context frames whose input and output filters match E1 and E2 respectively. Once such a frame has been chosen, only the valid methods are tried and the possible results are compared with E2.

In all cases, whether the equality is found valid or not, the system starts again the whole analysis, this time using wider heuristics (indeed, certain valid equalities may be obtained through erroneous operations). If the analysis succeeds, the system now possesses one or several plans explaining the equality. It is on the basis of these plans that the system focuses its comments. In the following section we shall see how it is possible to interrogate the pupil when there are uncertainties about the plan he has adopted. For the moment we present the heuristics used by the system for plan inference.

For any one of these heuristics, there is no restriction to the choice of a relevant context : any frame whose task and filters match will be tried. Restrictions are imposed only upon the accessible methods. We have defined 3 heuristics for the choice of methods. Going from the first to the third, we progressively put in question the previously observed behavior.

Heuristic1 assumes the pupil's regularity, ie. the plan interpreter tries to apply in each context the methods declared at level 1 or 2, or the methods already applied in the current session (they possibly have not yet been recorded at level 1 or 2 of the model). Moreover the system always tries to apply the valid methods in the assumption that the pupil has benefited from the teaching ! All the possible plans compatible with this heuristic are inferred. If no plan is found, and only in that case, the system tries heuristic 2.

Heuristic 2 makes available all the methods of heuristic 1, plus the methods mentioned at level 3 in the global model. Furthermore it gives access to certain erroneous patterns which have possibly never been used before, but could be at any moment by an inadvertent pupil (for instance, forgetting a sign '-' in the result). If no plan is found with heuristic 2, and only in that case, the system goes on to heuristic 3.

Heuristic 3 simply gives access to all possible methods.

In fact, to avoid a combinatorial explosion, and also to avoid inferring totally absurd plans, the set of possible methods has to be filtered (especially for the heuristic 3). This filtering takes into account the already inferred part of the ongoing plan. Criteria for this filtering are :

- maximum number of errors mentioned by a plan (one has here to distinguish between errors made "sequentially" or "in parallel")
- size of the analyzed data
- local coherence (if the same context recurs within the ongoing plan, the same method is applied)
- use of certain "crazy" errors only if the rest of the plan is correct
- execution in a normalized order of certain universally commuting actions (but then if the plan succeeds, the system must find out all the previously inhibited variants).

Note that the internal structure of a plan is that of a tree, whose nodes are instances of context frames, labelled by a context name, the input and output expressions, and the name of the executed method.

At the end of this first phase of the analysis, it may happen that severalplans are candidates to the explanations of the current equality. Figure 4a presents a simple example of such a situation. We shall now show how the system reacts to this uncertainty.

Fig.4 : an example of ambiguity

Fig. 4a The pupil is asked to reduce 4/6 + 5/7 ; he answers by the equality : 4/6 + 5/7 = 29/21 ; The plan interpreter infers 2 possible plans for this valid result :

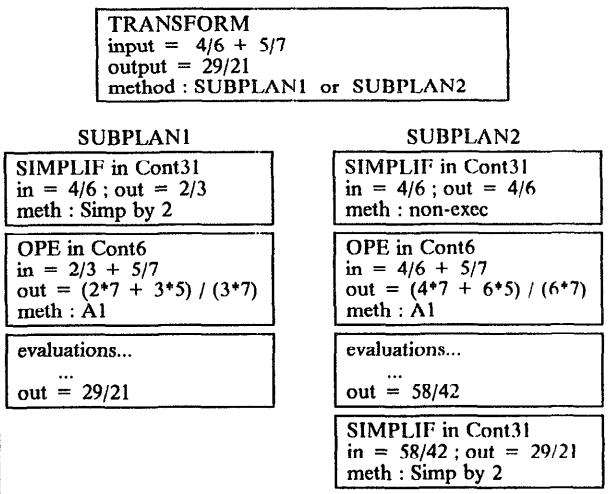


Fig. 4b Both plans are plausible ; the system knows that an addition and a simplification have been performed, but does not know in what order they were effected ; it wants to explain to the pupil that it is better to simplify first. To clarify the situation, the system can focus on either of the 2 following calculation steps :

Step 1
Task : Simplif ; Expression : 4/6 ; Appreciation : valid
Possible <context,method> : (Cont31,sd) (Cont31,non-exec)
Step2
Task : Addition ; Expression : * ; Appreciation : valid
Possible <context,method> : (Cont6,a1)

If Step1 is selected, the system asks the pupil to explicit his simplification. He answers by the equality : 58/42 = 29/21 . This equality is analyzed, and by cross-checking with the two candidate plans, the system is now sure of the simplification time.

In a similar way, the pupil could have been asked to explicit Step2, thus writing : 4/6 + 5/7 = (4x7 + 6x5) / (6x7). In this case also the system would have reached the same conclusion.

4. AMBIGUITIES

The existence of alternative plans simply indicates that the analyzed equality is ambiguous : it may be interpreted in several ways. This ambiguity however may not affect all the task execution contexts mentioned by the different plans. To be able to comment upon the pupil's performances, the system needs to know which are the contexts affected by the ambiguity, and for each of these contexts, it must have the most accurate posssible description of the (implicit) calculation steps whose complete clarification would eliminate the uncertainty concerning the context.

Firstly, it is easy to define the contexts affected by the ambiguity : theyare those which are not mentioned by all candidate plans, or those which, although mentioned in all plans, are processed by different methods depending upon the plan.

Secondly, what are the (implicit) calculation steps about which the system can on good grounds decide they have actually been processed by the pupil, and on which it could focus a question ?

In order to answer this question, let us first recall that the tree structure of a plan reflects the logical and temporal structure of a task execution. Assuming that we climb down along a branch starting from the root, we note at each node the task

name Ti and the input expression Ei. We obtain a sequence :

< T1,E1 > , < T2,E2 > , ... , < Ti,Ei > , ... , < Tn,En >

In other words the execution of T1 on E1 has required (among other tasks) the execution of T2 on E2, ... , Ti on Ei, ... , Tn on En . If all candidate plans possess in common the same sequence, we can take for granted that this hierarchy of situations has actually occurred. The system can "develop" this sequence for the pupil and designate without ambiguity the step corresponding to the lowest node < Tn,En > . This gives rise to a first type of questions : " how did you execute Tn on En ? ".

It is yet possible to go further down in the trees, but questions then become less precise. Let us suppose for example (with the previous notations) that all plans mention, among all the sons of the node < Tn,En > , a node < Tp,* > . This means that the execution of Tn on En has required the execution of Tp ; but depending on the plan, Tp has been executed on different expressions. The system can however, after having designated the "higher" step < Tn,En > , ask the question : " show me how you did Tp ".

An obvious variant of this second type of question is obtained by assuming that it is not the processed expression which varies from one plan to another, but the task. So the system could similarly ask : " show me what you did when you obtained expression Ep ".

Of course this investigation into the implicit (but "accessible") calculation steps can only be meaningful if it is carried out on a small number of not very deep trees.

Thus, given a certain task execution context affected by the ambiguity of the current equality, the system extracts the "accessible" calculation steps corresponding to nodes as close as possible to those labelled by the context under examination. Anyone of these steps may be an opportunity to ask clarifications from the pupil. The pupil gives this clarification in the form of new equalities expliciting some details of his former equality. These new equalities are in turn analyzed, and the inferred plans are cross-checked with the previous candidate plans. Thus the system reduces its uncertainties (see Fig. 4b).

5. CONCLUSION AND FUTURE DIRECTIONS

In this paper we have addressed the problem of modelling a student who performs non deterministic calculations (meaning that the calculations are not absolutely constrained by the assigned task), and who addresses them to the system in a more or less elliptic way. Calculus being a well structured activity, the plan formalism is adequate for its representation. Non determinism is reflected by the existence of optional tasks and/or the variety of possible methods in a given context. Since the drafting of calculations is not entirely normalized either, a certain ambiguity is unavoidable. We explained in Section 4 what are the only chances, according to us, for the system to manifest to the student a partial comprehension of his calculations, and to ask relevant questions in order to improve this comprehension.

The approach presented here has been the basis for an implementation in Vmprolog carried out at the IBM Research Center in Paris. Only a part of the architecture given in Figure 1 has been achieved, namely an expert problem solving module (covering the four operators +, - *, /), a knowledge base (including incorrect knowledge), a student model, and a modelizer analyzing and appreciating equalities asserted by a (simulated) pupil. The modelizer also updates the student model.

Even without mentioning all the discourse understanding and tutoring strategies problems, there is still obviously much to be done, especially for the local modelling of the student's calculations, even if they are analyzed from a strictly "procedural" point of view. Here we have given methods for analyzing *one* equality, but we have not said how to process complex calculations which may spread over many equalities. Between those two abilities, there is, if a metaphor may be permitted here, as long a distance as between understanding a sentence and understanding a discourse.

A last word of caution regarding the concept of plan which we have used in this paper. We do not pretend that the plans in our system exactly reflect the intentions of the pupil. They are only a way to describe his actions. He is not supposed to acknowledge completely this description, more particularly when this description mentions what we have called "erroneous methods", which by definition have never been taught to him. The general relationship between an "execution method", written in symbolic form as in the knowledge base of the system, and a "computation act" performed on numbers, must not be a priori considered as the intentional application of a rule, but only as a resemblance relation between two patterns. Similarly we think that it is perhaps better not to use (as we did here) the terms *task* and *method* which carry too much intentionality, but rather to speak of *actions* and *decomposition of actions*.

So, unless the same pattern of error returns several times, or unless the pupil is prompted to express this pattern in a symbolic (litteral) form to justify his calculation, the system should merely categorize the observed performance as a "wrong addition", "wrong simplification", etc., without going into further details regarding the origin of the mistake.

Note : work presented here is part of first author's thesis [Visetti 86]

References

[Allen 80] J. Allen , C. Perrault. Analyzing intention in utterances. *Artificial Intelligence* 15 (1980), 143-178

[Brown 78] J.S. Brown , R. Burton Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science* 2 (1978), 155-192

[Brown 80] J.S. Brown , K. Van Lehn Repair theory : a generative theory of bugs in procedural skills. *Cognitive Science* 4 (1980), 379-426

[Charniak-Mc Dermott 85] E. Charniak , D. Mc Dermott *Introduction to Artificial Intelligence* Addison-Weslay 1985

[Genesereth 82] M.R. Genesereth. The role of plans in intelligent teaching systems. *Intelligent tutoring systems* (eds. Sleeman and Brown). Academic Press 1982

[Matz 82] M. Matz. Toward a process model for high school algebra errors. *Intelligent tutoring systems* (eds. Sleeman and Brown). Academic Press 1982

[Resnick et al. 85] L. Resnick, E. Cauzinille-Marmeche, J. Mathieu. Understanding algebra. *International Seminar on cognitive processes in maths and maths learning*, University of Keele, 1985

[Sleeman et al. 82] D. Sleeman , J.S. Brown (editors) *Intelligent Tutoring Systems*. Academic Press 1982

[Sleeman 84] D. Sleeman An attempt to understand student's understanding of basic algebra. *Cognitive Science* 8 (1984), 387-412

[Visetti 86] Y.M. Visetti . *User modeling in ICAI* . Doctorat de l'Universite Paris 6 , 1986

[Vmprolog 85] Vm/Programming in Logic *IBM Programmer's Manual*, 1985

[Zissos 85] A. Zissos , I. Witten. User modelling for a computer coach. *Int. J. of Man Machine Studies* 23 (1985), 729-750