In: C. Alvegård (Ed), *Proceedings of CALISCE'98*. Göteborg: Chalmers University of Technology, 381-389

Error Characterisation in Problem Solving Tasks

J-B. Auriol - J-L. Dessalles

Ecole Nationale Supérieure des Télécommunications
46 rue Barrault - 75013 Paris - France
{auriol, dessalles}@inf.enst.fr

Abstract

Students' errors become manifest through erroneous behaviours noticed by the teacher. However, addressing behavioural deviation alone is not sufficient to design appropriate feedback. We propose here a model of student error, based on a separation between procedural and logical knowledge. This model is tested through its ability to predict the observed behaviour of subjects solving the Tower of Hanoi problem. Using this model, we are able to propose a "deep" error classification, based on the observation of the internal representations of the system when it generates deviant behaviours. From this characterisation of errors, we aim at designing a critiquing system. Such a system will deliver more elaborate feedback to the learner, from which we hope better pedagogical efficiency and better acceptability.

Keywords: Error, error repair, misconception, procedural learning, overlay

1. Importance of offering an appropriate error detection and repair

One of the main purposes of teaching activity is to detect errors and to correct them. Error repair was one of the main concerns of the Skinnerian tradition in C.A.I. (Skinner, 1969), and it has been addressed, from a very different perspective though, by initiators of the ITS trend (Stevens et al., 1979). This issue seems to be less central now, because of the absence of general error handling procedures, and because of a concern shift towards new simulation and browsing possibilities offered to students. However, we still believe that much added value is still to be gained from a proper error analysis and from relevant correction.

The observation of students performing tasks leads to an obvious distinction between conceptual and procedural errors (Inhelder & Piaget, 1979; Stevens et al., 1979; Richard, 1990; Sleeman, 1989; Ohlsson, 1991). Surprisingly, this fundamental distinction was strongly rejected in the Skinnerian tradition, for external reasons. This attitude may explain the poor efficiency of the "conditioning" approach in Computer Assisted Learning. Our aim here is to build on this concept / procedure distinction, by analysing the interplay of both kinds of knowledge and by suggesting a more precise classification of the origin of errors.

What is at stake in a correct qualitative assessment of the student's performance lies at the core of the pedagogical act: deliver a message or a new learning situation that will help the student acquire the new task. The issue is not to oppose directed *vs.* non-directed learning. Rather, it is to avoid tedious and inefficient recurring trials, wild guesses and random repairs, while favouring creative exploration and understanding.

Giving relevant help is not an easy task, however. It has been shown that a conceptual explanation may be strictly useless when students are confined to procedural behaviour (Sleeman, 1989; VanLehn 1989). For instance, giving a rationale for proscribing a given action (*e.g.* moving a

term from right to left in an algebraic equation without changing sign) is irrelevant: students do not perform better than their fellows who are just given procedural instructions (*e.g.* "This is wrong. Here is the correct way to do it.") (Sleeman, 1989). Conversely, avoiding any conceptual feedback may be nonsense in some contexts, as illustrated in (Ohlsson, 1991).

As a consequence, it is of crucial importance to make the right diagnostic about the origin of error, since conceptual, logical, strategic or procedural mistakes require specific processing. Giving an inappropriate feedback is disturbing and makes learning boring. Conversely, if the computer shows relevant reactions, it will be perceived as smarter, C.A.L. sessions will be more attractive and learning efficiency is expected to improve significantly.

In what follows, we present our project, which is to design a C.A.L. system able to deliver relevant feedback to the student. Then we present results concerning the first phase of this project: based on an "overlay" principle, our system offers a characterisation of "surface" errors in terms of "deep" errors that are claimed to underlie observed errors. Lastly, we will make suggestions about the potential value of such a characterisation of problem solving behaviour for the generation of relevant critique.

2. First steps toward a critiquing system

Our aim is to design a critiquing system (Fischer, 1991) based on the "overlay" approach: the student's behaviour is compared to the system's handling of the problem (Vassileva, 1990). We are not analysing differences between expert and novice behaviour. Our method consists rather in tuning the system so that its behaviour matches the learner's actual actions. Then, by looking at the system's internal representations, we may infer the student's motivation for any faulty action, and bring the appropriate feedback if necessary.

The results presented in this paper correspond to the first phase of this project, which presumably is the most difficult. We show how the observed problem solving behaviour of novice subjects can be reconstructed using general, reusable techniques. We simulate the observed behaviour of students performing a complex task, in order to infer aspects of their internal processes from the actual functioning of the simulating program. We chose to work on the five discs version of the Tower of Hanoi problem. We were offered the opportunity to work on human protocols collected by Josiane Caron-Pargue at the University of Poitiers[1] (Caron-Pargue et al. 1996). The task is to move disks of increasing size from peg **A** to peg **C**, by moving one disk at a time, without ever putting a bigger disk on a smaller one. The disks are numbered in increasing size order. At the beginning of the resolution, all disks are on peg **A**, with both pegs **B** and **C** free (which is **noted** `(1 2 3 4 5)()()`).

3. Surface errors

There are few opportunities to make genuine errors in the Tower of Hanoi task. When involved in problem solving tasks, subjects take many sub-optimal moves that are often not to be considered as errors, especially in a trial and error strategy. But when teaching a skill, sub-optimal moves have to be detected as such by the teacher who decides whether s/he should give a feedback. In what follows, sub-optimal moves are thus listed among errors.

Sub-optimal and erroneous actions can be classified according to their form, as we propose now. Such a classification of what we call *surface error* cannot be sufficient, however, since it says little about the origin of errors and about the type of feedback that would be appropriate. Our aim then will be to use our model to infer *deep errors* that presumably caused those surface errors.

---

[1] In the work presented here, all aspects concerning the Tower of Hanoi task were carried out in the context of the PROVERB group managed by Josiane Caron-Pargue.

3.1. Rule Violation

The most basic error is rule violation: the subject does something illegal. The problem is not a calculation error, but rather an attempt to make a step that is not allowed in the current state of the problem. As example, we consider a move performed by Olivier, a 10 years old boy, when solving the four ring Tower of Hanoi problem. The state is `(4)(1 3)(2)` (translated from french):

Olivier: I take the green disk [disk 2], and I put it on peg B.	`(4)(2 1 3)()`
Experimenter: You cannot, Olivier.	
Olivier: I put it on A.	`(2 4)(1 3)()`

We note, in this situation, that the experimenter mentions the problem just for Olivier to correct it. He does not need to remind Olivier of the rule.

3.2. Calculation Error

The next possible error is the calculation error. The subject attempts something legal, but the result given is not correct. This is the easiest mistake to detect, provided the student gives the solution step by step.

3.3. Stuck in Dead-End

The subject feels that s/he cannot do anything further, whereas legal moves still exist. As an example, consider the typical comment (translated from french), given here by an adult subject in the situation `(1 3)(4)(2 5)`:

> Well, here I'm stuck. I cannot put disk 4 anywhere.

An extreme case of this situation is the "blank page syndrome". The subject does not initiate any consistent plan, expressing again the feeling of being blocked, as none of the legal steps look satisfactory. As an example, we consider the hesitation showed by an adult subject. In the following excerpt (translated from french), digits in parenthesis indicate time in second during which the subject says and does nothing. The situation presented here still last for 181 seconds. Afterwards, the subject resumes a more effective solving behaviour.

Here I am blocked. (13) Oh (11)	`(1 2 3)(4)(5)`
I put back disk 1 on B(4) but I am still blocked (4)	`(2 3)(1 4)(5)`
Disk 2 on A (24) Well, at the end [the subject laughs] (22)	`(2 3)(1 4)(5)`
No, I put back disk… it's the disk 3 the disk 2… that…	
Disk 2 on…Disk 1 Disk 2… Disk 2 on C (4)	`(3)(1 4)(2 5)`
But it will do the same thing (30)	

4. A Model of problem solving activity

In this section, we present a model that we implemented to account for the observed behaviour of subjects involved in a problem solving activity. This model is based on a strong hypothesis: we make a distinction between procedural and conceptual abilities. This distinction may appear as classical (*e.g.* similar to the procedural / declarative separation (Anderson, 1983)). We mean it as radical, however. Knowledge is represented differently in both cases: calculation capabilities are based on *operators*, while evaluation capabilities are based on *logical* competence. Both representations are kept strictly apart in two different modules.

4.1. Procedural skills

Following (VanLehn, 1989), procedural skill representation is based on task specific operators. An operator takes the following form:

(State 1, Operation, State 2)

where **State 1** is the state of the problem before applying **Operation**, and **State 2** is the state resulting from applying **Operation** to **State 1**. Correct operators are able to propose all existing legal steps from a given situation. They can take one of the resulting states they have proposed as a new starting state. Operators can thus be applied recursively. The allowed depth of this recursive search is one of the parameters that can be tuned to match the performance of novice subjects.

To account for the observed fact that, when a subject encounters a given state twice, s/he often chooses the same move, we introduced a contextual preference for operators. In other words, in a given context, the operator will propose legal steps in a given order. These preferences, which may evolve through learning, are another parameter that has to be adapted.

An operator, when prompted, returns (legal, preferred) moves. However, it ignores the current goals; it does not evaluate in any way the states resulting from suggested moves. This will be the role of the conceptual capabilities.

Possible errors

There are many ways to make errors with operators. An operator may propose illegal steps; it may also start from a given state, choose a legal step, but compute a wrong resulting state. Lastly, an operator may fail to propose a legal step in a given situation.

4.2. Conceptual capabilities

Rules

Conceptual knowledge is represented by first-order logical rules, in an extension of the negative conjunctive normal-form. Rules take the following form:

List of terms ⇒ Mod

where each term in the list is in conjunction with the rest of the list, and where the modality **Mod** is either **Und** (undesirable) or **Par** (paradoxical).

The functioning of logical competence in this model is to systematically detect any rule saturation (Dessalles, 1991). We say that a rule is saturated when all its terms are known true. Thus:

[A, not(B), C] ⇒ Mod

will be saturated if **A** is true, **B** false and **C** true. Depending on the modality, an undesirable or paradoxical situation will be detected.

Representation of constraints

Logical rules are used to represent constraints of the task, by using the modality **Par**. For example, the Tower of Hanoi rule saying that disk X cannot be put on a smaller disk Y can be expressed as follows:

[X is on Y, X is greater than Y] ⇒ Par

Representation of goals

Logical rules are also used to represent goals (and sub-goals) during the solving task, by using the modality **Und**. For example, in the Tower of Hanoi case, the sub-goal ‘put disk **5** should be on peg **C**’ can be expressed this way:

[not (5 on C)] ⇒ Und

Possible errors

Errors related to rules can only occur when we have a bad set of rules. A rule can be false, unknown or partially known. If the erroneous rule modality is **Par**, errors appear as constraint violations. If the modality is **Und**, the student might get stuck in dead-ends or use poor strategies.

4.3. Coupled capabilities

Problem solving

Given logical and procedural capabilities, a problem solving activity can be carried out easily. The operator proposes resulting states of legal steps to the logical apparatus, which evaluate whether or not the current undesirability rule is still saturated in the proposed state. The proposed step is accepted when the undesirability rule is no longer saturated, and is rejected otherwise.

Subgoaling

Generation of sub-goals are carried out by a coupling between logical and calculation capabilities. Because generating a subgoal for a goal that can be solved on his own is inefficient, we postulate that subgoaling activity arises only when the subject is stuck in a dead-end. A subject is stuck in a dead-end when none of the reachable steps solves the current undesirability rule[2]. In such a situation, the logical apparatus generates a counter-factual (inverting the truth value of a known term to see what would be the consequences of this inversion; It can be seen as a mechanism to generate hypotheses). The counter-factual is chosen in order to make the current undesirability rule no longer saturated. The counter-factual is then given to the operator, which is called in a reverse way, in order to generate a situation where a step that would reverse the truth value of the counter-factual is legal. The new situation is then considered as the new goal (hence, the *subgoal*) by the logical apparatus.

5. Experimentation

In order to check the "overlay" accuracy of our model, we compared the resolution given by our model and the result given by human subjects in a given task, the Tower of Hanoi problem.

We performed a step by step comparison. In order to be able to compare resolutions after the first difference in move, our resolution is bound to follow the human one. At each step, our model computes its next step, which is then compared to the subject's move. The human step is always the one played. If there is a difference, and if the step chosen by our model resulted from operator preferences, the involved preference is inverted. Seven subjects were involved in the protocol set on which we worked, totalling 40 protocols and 1462 steps.

We also put a few other basic models to trial: random only, no preferences[3], random without moving the same disk twice, and a model inspired by (VanLehn, 1991). In the VanLehn inspired model, three steps out of four are forced steps. Each time the model moves Disk 1, the next allowed move is to take the only other legally moveable disk and to put it on the only legal peg. Each time the model moves Disk 2, the next allowed step is to put Disk 1 back on it. The model chooses the optimal move for each unresolved move (without correction, this algorithm gives the optimal solution[4]).

For each model, we computed the percentage of correctly predicted moves out of the total number of moves. The results obtained after these trials are:

---

[2] Actually, before entering the subgoal generation, the model takes a few moves which do not solve the current undesirability rule, but which correspond to the preferences. We call this the free space exploration mode. This point does not make any difference in this discussion, however.

[3] That is, our model where the operator preferences have been replaced by random choice.

[4] The reader will note that in his paper, VanLehn did not intend to duplicate human performance at the solution level. Rather, his purpose was to study how learning takes place during problem solving. We chose to use this model as reference because it was simple to implement and gives good results.

Random:	33.7 %
Random without backtrack:	68.9 %
Our model without preferences:	73.7 %
Inspired by (VanLehn, 1991):	78.6 %
Our model:	80.7 %

The VanLehn inspired model generates by itself only the optimum solution, and makes heavy assumptions on what subjects know. It gives good results, but this is because it takes advantage of task specific constraints. These constraints are very unlikely to be known by the novice subject, and the model is thus, as such, not plausible. Its good performance comes merely from the fact that this model takes good decisions, as subjects often do.

The results given by models involving random may vary by 1%. Results given by our model also vary by 0.7% around the value we give. This last variation is due to the fact that initial preferences are fixed at random. The results of the VanLehn inspired model do not vary.

The differences between the first three models and ours are significant ($chi^2 = 15.49$, $p < 0.0005$, for the comparison between our model and our model without preferences). The difference between our model and the VanLehn inspired model is not significant ($chi^2 = 1.51$, $p < 0.25$).

The comparison with the three random models is interesting. Our model without preferences is much better that random alone and is significantly better than random without backtrack. This confers an independent validation to the logical part of our model.[5]

A few unexpected differences (less than 4% of the differences) still occur, as sometime the human subject does not play a step solving the model current goal. Those differences could probably be explained by difference between our model's and the subject's goal.

6. Deep error classification

We are now able to propose a "deep" error classification, based on what our model designates as possible source of errors. We will link each class to typical "surface" errors, previously listed, that are to be observed on human protocols.

6.1. Paradox violation

Effect: the student makes 'obvious' errors, such as an explicit constraint violation (*cf.* 3.1).

The first possible cause of this type of mistake is that the student ignores a paradoxical rule. Reminding the student about the violated rule may be sufficient in this case. Sometimes, however, a rule seems to be ignored, but in fact is not. This is possible when a rule appears as violated (for the teacher) while not saturated (for the student). This is due to the fact that the subject ignores another rule:

$$\mathbf{[A, B] \Rightarrow Par}$$
$$\mathbf{[not(B)] \Rightarrow Par}$$

Here, if **A** is true but the subject ignores the first rule, the second rule will not be saturated. This makes the teacher believe that the subject ignores this second rule. In such cases, it is better to engage in a Socratic dialogue with the student, by pointing to the contradiction and by asking him/her to change the truth-value of one of the terms involved in the saturated rule. This strategy is used in SAVANT-3, a Socratic system dedicated to conceptual learning (Dessalles, 1991).

6.2. Calculation errors

Effect: the student makes an incorrect move (*cf.* 3.2).

---

[5] Also, the results given by the complete model are better than those obtained by replacing preference by random, which indicates that preferences better account for human behaviour than random choices do, especially when our model switch to free space exploration mode.

According to the initial state and the operator used, the final result is not the result expected by the student. In other words, the student's operator does not give reliable results. This situation has been modelled with 'malrules' (Sleeman, 1982) or bugs (VanLehn, 1989). As discussed earlier, when we arrive at such a diagnostic, it may be preferable to simply give a procedural correction to the student.

6.3. 'Strategic' errors

Among the most difficult errors to address are strategic errors. The student's moves are legal, and yet the student is not able to solve the problem because s/he does not take steps in appropriate order. This kind of mistake is hard to correct because it may result from either part of the problem solving apparatus, as we see now. It is also a positive aspect of our model to point to this distinction.

Incomplete operator

Effect: the student does not see that a legal step, or a short legal step combination, allows to reach the local goal.

In the terms of our model, this means that the operator is incomplete, as it is unable to propose the appropriate combination of legal steps required by the logical module. What would be an appropriate help in such case ? The student can be allowed to search further, or may be given a hint, *e.g.* the first step of the combination. The best remedial would be to give opportunities to practice, as the student is still unfamiliar with the operator.

Poor strategy or no goal generation

Effect: the student is stuck in a dead-end. S/he cannot solve the problem without taking back some moves (*cf.* 3.3). Another possible, extreme, manifestation is that the student is lost; s/he expresses the feeling that there is nothing interesting to be done (*cf.* 3.3). Although legal steps are at hand, s/he tries none of them.

This means that the student does not see how to reach the current goal, and that s/he is unable to compute easier ones. The problem is similar to what we called paradox violation (*cf.* 6.1.), where the **paradoxical** modality has been replaced by the **undesirable** modality.

This is a typical behaviour of novice subjects. Addressing this problem is quite easy. The student may be encouraged to try something anyway (*i.e.* to explore the problem space). A goal may also be computed and a Socratic dialogue initiated to make the student realise it. It is important to note that a novice student should not be immediately corrected when s/he shows poor strategy. Learning operators requires going through a trial and error stage. During this necessary phase, the student should not be systematically corrected, otherwise s/he looses confidence in his/her skill.

7. Conclusion and future work

We presented here the first phase of our project of critiquing system based on an "overlay" approach. Our aim in this phase is to build a reliable, psychologically plausible model of student performance. We presented such a model, based on a strict separation between logical competence and procedural skill. Using this model, we are able to propose a method to go back from the observed erroneous behaviour (surface errors) to its cause (deep errors). We are indeed convinced that an efficient feedback must address this deep, cognitive level where actions are motivated and plans are initiated. Our model aims at being generic, *i.e.* not task-specific. We need to establish this generality by evaluating the model on other tasks. We are currently working on trigonometric problem solving task. Then we will develop the critiquing part of the model, which will convert error diagnostic into a relevant teaching act.

References

Anderson, J. R. (1983). The Architecture of Cognition. Cambridge, USA : Harvard University Press.

Caron-Pargue, J. & Gillis, S. (1996). *Verbal production and problem solving*. Antwerp papers in linguistics 85.

Dessalles, J-L. (1991). "Conversation Assisted Learning: The SAVANT3 Dialog Module". In E. N. Forte (ed), *Proceedings of Calisce'91*. Lausanne : Presses Polytechniques et Universitaires Romandes, 159-165.

Fischer, G., Lemke, A. C. & Mastaglio, T. (1991). "Critics: An Emerging Approach to knowledge-based human-computer interaction". *International Journal of Man-machine Studies*, *35*, 695-721.

Inhelder, B. & Piaget, J. (1979). "Procédures et structures". *Archives de Psychologie*, *XLVII*(181), 165-176.

Ohlsson, S. (1991). "Interview, by J.Sandberg and Y.Barbard". *AI communications*, *4*(4), 137-144.

Richard, J-F. (1990). *Les activités mentales*. Paris : Armand Collin.

Skinner, B. F. (1969). *La révolution scientifique de l'enseignement*. Bruxelles : Dessart.

Sleeman, D. (1982). "Assessing aspects of competence in basic algebra". In D. Sleeman & J. S. Brown (ed), *Intelligent Tutoring Systems*. Londres : Academic Press, 185-199.

Sleeman, D., Kelly, A. E. & Martinak, R. (1989). "Studies of Diagnosis and Remediation with High School Algebra Students". *Cognitive Science*, *13*, 551-568.

Stevens, A., Collins, A. & Goldin, S. E. (1979). "Misconceptions in student's understanding". *International Journal of Man-machine Studies*, *11*, 145-156.

VanLehn, K. (1989). "Problem solving and cognitive skill acquisition". In M. I. Posner (ed), *Foundations of Cognitive Science*. Cambridge : MIT Press, 527-579.

VanLehn, K. (1991). "Rule acquisition events in the discovery of problem-solving strategies". *Cognitive Science*, *15*.

Vassileva, J. (1990). "A Classification and Synthesis of Student Modelling Techniques in Intelligent Computer-Assisted Instruction". In D. H. Norrie & H-W. Six (ed), *Lecture Notes in Computer Science 438 - Computer Assisted Learning*. Berlin : Springer-Verlag, 202-213.

Appendix : experiment

All the goals are represented by undesirable rules, in the following form:

(not (Disc X on peg Y)) ⇒ Und

We use two kinds of paradoxical rule in our model, which play a role only in the subgoal computing part, as the operator used does only compute legal moves. The first ones were rules stating that a given disc cannot be located on two different pegs as in the following:

(Disc 1 on peg A, Disc 1 on peg B) ⇒ Par

The second ones were rules stating that a disc has to be located somewhere (i.e. that a disc cannot vanish) as in the following:

(not (Disc 1 on peg A), not (Disc 1 on peg B), not (Disc 1 on peg C)) ⇒ Par

The operator we used was able to propose all the possible moves up to a depth of two steps. Preferences context included the starting peg and whether or not the disc 5 was already on peg C, as in the following:

(*context*: [starting peg: A, 5 not on C] *preference*: [First C, then B])