From I.T.S. to I.C.S. : Learning with an Intelligent Critic, Not with a Tutor

jean louis Dessalles
Télécom Paris - Département Informatique
46 rue Barrault - 75634 PARIS Cedex 13 - France
E-) : dessalles@enst.fr

Summary

Intelligent Tutoring Systems (ITS) seem very attractive as learning tools, because they ideally replace a personal tutor, keeping in their records (student model) many characteristics of each individual learner: what the learner knows, what he ignores or wrongly knows, his preferred way of learning (inductive vs deductive, autonomous vs guided, ...), and so on.

Unfortunately I.T.S. are to be found mainly in research laboratories, not in classrooms! The reason for this is that their design is very difficult, especially on university level topics.

In TELECOM-Paris we are trying to design another type of intelligent teaching aid which should be considered rather as a critic than as a tutor. This approach should be indeed more appropriate in a university context where students are able to learn mostly by themselves, but need specific help when experiencing a particular problem (conceptual puzzle, abnormal behavior of some device, repeated failure).

Tutoring vs "Critiquing"

In traditional tutoring systems, the expertise to be conveyed is contained in prestored presentation blocks which are designed by an expert teacher and are simply displayed to the student under given conditions. In the so-called Intelligent Tutoring Systems, sometimes also called Knowledge Communication Systems [Wenger 1987], an expert module is able to deliver its expertise dynamically. In most cases, this expertise is not only a description of the various concepts that the student is to acquire, as in a curriculum, but an actual domain-model which can perform some of the tasks asked of the student and thus provide him with a dynamic guidance.

ITS have been presented as offering better guidance, not only because of the ability of their expert module to generate and compare solutions, but also because they may have at least three other kinds of knowledge :

- a pedagogical expertise : decisions about sequencing, interventions, types of explanations, immediate or delayed remediations, types of interaction (from coaching to mere monitoring through mixed-initiative) are taken during the interaction, by taking the student's behavior into account [Woolf 1987]
- a student model : what the student correctly knows, what he can do, his misconceptions, some of his preferences [Self 1988].
- complex interactional capabilities : conversational capabilities [dessalles 1991], language processing [Brown et al. 1982], graphical representations, etc.

This features of Intelligent Tutoring Systems make them quite "heavy", in such a way that they cannot perform their tutoring task if authors do not provide them with all these expertises. Such a system must be infallible in order to give the right diagnostic and the right guidance. Unfortunately, this is not always possible, especially for domains that cannot be totally represented through rules. Moreover, the system knowledge is hardly reusable for other domains : the pedagogical expertise is not fully generic since it depends on specific actions performed by the student and on specific decisions that can be taken ; the student model depends crucially on the domain, especially for the analysis of bugs and misconceptions [VanLehn 1988] ; most of the interactional capabilities which are generally implemented, especially natural language processing, are also dependent on the domain knowledge base (e.g. for NLP : vocabulary, anaphora processing, understanding of meaning through recognition of intention).

Compared to a tutor, a *critic* has several distinctive features. It does not supervise the whole teaching system, but is rather a part of it. It is not omniscient, it only criticizes what it is able to. Critics do not give validity judgments that classify answers or actions as right or wrong. Their remarks should rather be considered as warnings that may help students repair their misconceptions.

Systems which are able to deliver such criticisms are presented by G. Fischer [Fischer et al. 1991] as *cooperative* systems, that serve as cognitive amplifiers of the person. For this author, the major difference between expert systems and cooperative problem-solving systems involves the roles of the user and computer. Most expert systems ask the user for input, make all decisions and then return an answer. In a cooperative system, the user is an active agent empowered by the system's knowledge.

In order to be a good cooperative partner, the system does not need to have a comprehensive knowledge base covering all aspects of the task being performed, contrary to what is required in an ITS (e.g. the PROUST system [Johnson & Soloway 1987] needs to know exactly the function to be computed by the student's Pascal program ; it also needs to know about typical bugs in order to give a sound diagnostic about the program).

G. Fischer and his team designed several artificial critics. For instance ACTIVIST is an active help system for a text editor. It "looks over the shoulder" of a user an infers goals from observed actions. After having observed three actions that it analysed as suboptimal executions of a given type of task, ACTIVIST informs the user of a better procedure for the task.

LISP-CRITIC is another critic, designed to support programmers. It suggests transformations that make the Lisp code more readable, smaller or more efficient. For instance it may suggest replacing a single conditional *cond* expression *(cond (C a) (t b))* by a simpler expression *(if C a b)* in a certain context.

For the student, an interaction with such critics are intermediary between learning with a tutor and with an open learning environment. The former cannot adapt to tasks defined by the user, while the latter does not sufficiently help students who are stuck in a problem-solving activity or who have reached a suboptimal plateau in their competence. Critics are able to point out wrong actions or shortcomings in students' solutions, and suggest ways to correct them.

We strongly believe that the "critiquing" approach can be valuable in many contexts where teaching systems are used. A teaching system typically involves a knowledge presentation device (possibly with hypertext or hypermedia), simulations (from exercises to microworlds), and also some guidance. A critic could intervene on two occasions :

- on a learner active request, to criticize the learner's beliefs or actions (passive critic),
- during exercises or simulations, when the learner comes upon a situation that can be considered as abnormal or undesirable in the context (active critic).

The critic will criticize what he is able to, and will not require from the student that he accepts its diagnostic. This approach is particularly well suited in the context of higher scientific education for at least two reasons :

- most of the time, scientific subject matters are complicated. No system is able to cope with the full complexity of, say, computer programming, differential equations, or chemical reactions. By contrast, critics may often have something useful to say that may help the student, even if they are unable to provide a complete solution.
- adult students are perhaps more prone to engage in a cooperative relation with the machine. They will be able to integrate the system's remarks into their own conceptual framework and then they can adapt to it or reject it.

Our own work at TELECOM-Paris with the development of SAVANT3 is based on the critiquing paradigm : our system tries to help the student revise his knowledge by making relevant criticisms. But SAVANT3 has some distinctive features, compared to other critiquing systems.

SAVANT 3 as a critic: from criticism to argumentation

When trying to make a criticism, SAVANT3 does more than delivering a single comment. It is actually able to initiate a whole conversation, with both the learner and the system arguing in turn. This is actually a way to go further in the "critiquing" paradigm. Most of the time, SAVANT3 is unable to calculate the definitive criticism that helps the student understand his misconception. It is only able to point out an abnormal (or undesirable) situation.

SAVANT3's functioning is straightforward. It always tries to find an inconsistency in the student's statements or actions. During a simulation, for instance, it is able to point out an abnormal situation and to present it as the consequence of inconsistent choices. The student generally has several ways to escape from this inconsistency. From one inconsistency to the next, a whole conversation may ensue. The purpose of this kind of well-argued criticisms is to help the student discover possible misconceptions by himself.

In the following example, SAVANT3 criticizes the choice of a student who decided to represent the information contained in a typed text by coding only printed characters (student in italics) :

According to what I understood: the points (pixels) in the page are not taken as symbols. What else in the page could we code, if we except printed characters?
----- *you mean if there are drawings?*
But there may be a problem ... If we consider that one wants to code for figures (schemas, photos, signature)...
----- *but this is a printed page*
yes... According to what I understood: one chooses a TELETEXT-like coding. Printed characters are not the only thing to be coded...
----- *no*
could you say more?...
----- *margins, paragraphs*
yes... mmm... But wait... According to what you said: one wants to code for positions (paragraphs, etc.). I must say I'm confused !
----- *you use special codes*
well, well!... Okay...

Let us give another example. SAVANT3 is used here to make criticisms about a simple Prolog program written by the student. The student was asked to write a small program that computes the following relation *taxes(X,Y)* :

for X less than 4000, Y must be 0;
for X between 4000 and 10000, Y must be 10;
for X greater than 10000 Y must be 30.

X represents Earnings of a person working in a company, and Y stands for the taxes she has to pay for her car. The student is provided with a simple editor that allows only a limited number of different programs. SAVANT3 is then able to criticize the program written by the student, as in the following excerpt :

```
taxes(Earnings, 0)   :-  Earnings < 4000, ! .
taxes(Earnings,10)   :-  Earnings >= 4000, Earnings < 10000, ! .
taxes(Earnings,30) .
```

According to what I understood : Earnings >= 4000 is used in clause 2. But perhaps you think that the program is the most efficient one ----- *why?* But there is a problem anyway... unless you consider that clause 2 is called for people earning less than 4000 ...	*SAVANT3 detected that* Earnings >= 4000 *is superfluous in clause 2*
----- *yes* According to you, clause 2 is called for people earning less than 4000 . It's surprising, if we admit that clause 1 succeeds for people earning less than 4000 !	*For the student, the instruction* Earnings >= 400 *is not superfluous*
----- *why is it surprising?* According to what I understood: - clause 2 is called for people earning less than 4000 - clause 1 succeeds for people earning less than 4000 - cut is used in clause 1 Which solution do you opt for?	*SAVANT3 must make the inconsistency explicit*
----- [clause 2 is not called for people earning less than 4000]	*The student changes his min*
But wait... According to what you said: - Earnings >= 4000 is used in clause 2 - clause 2 is not called for people earning less than 4000 I must say I'm confused !	*We are back to the previous problem*
----- *Earnings >= 4000 is no longer used in clause 2* well, well!... Okay...	*The student modifies the program.*

In order to make SAVANT3 able to argue with the student, we have to provide it with logical knowledge about the subject. This is precisely where the "critiquing" approach is the most valuable. Only a few logical rules are necessary for SAVANT3 to be able to create relevant criticisms : about 15 rules for the first dialogue, 100 rules deriving from about 20 prototypes for the second dialogue. With this knowledge, SAVANT3 can compute several ways of arguing, depending on the student's reactions.

Such rules are used either to establish that some terms are locally true in the current context, or to "trap" the student in an inconsistency. SAVANT3's functioning is thus very simple: it looks for a rule which is violated, in order to express surprise. If none can be found, but if a rule is "almost" violated, it makes a suggestion that may lead the student to be inconsistent [dessalles 1991].

SAVANT3 is a critic. After one of its utterances, there are no right or wrong answers. There are just many possibilities in order to remain consistent. The basic idea is that if there is any misconception in the student's mind, then such a misconception will probably have unsound results, as shown in both excerpts above. The system, without being able to diagnose the misconception in the first place, points out the inconsistency, and then, through the dialogue, it gives the student the opportunity to find out a possible flaw in his reasoning.

Theoretical background of SAVANT3

In most ITS, students are asked to perform *tasks*. This is perhaps the consequence of the stress laid by many psychological and pedagogical theories on the acquisition of procedures. However we have some reasons to consider that the importance of *concepts* mastered by the student before and during learning has been overlooked [dessalles 1990]. Very often the knowledge to be taught is purely conceptual by nature (as was the case in the first excerpt above), and conceptual knowledge is also essential to carry out complex procedures like computer programming.

Two kinds of causes have been invoked to explain errors during action: procedural flaws (bugs [Van Lehn 1981, 1988] and "mal-rules" [Sleeman 1982]) on one hand, and conceptual flaws (misconceptions [Stevens et al. 1979]) on the other hand. When the task becomes complex enough, so that it cannot be performed by merely following a given set of rules (e.g. when writing a computer program), it seems that students have to *understand* their errors in order to progress. In other words they have to correct their misconceptions.

Our approach does not consist in trying to diagnose a possible student's misconception directly from the actions performed. The basic principle underlying SAVANT3 is that any misconception will become apparent through its logical unsound consequences, and that it will be corrected by the student himself when confronted to some contradiction and after an argument exchange.

The basic principle underlying this approach is that people are particularly sensitive to their own inconsistencies and are ready to re-examine their knowledge to find a solution, as it commonly occurs during spontaneous conversations [dessalles 1992a]. Some conversations seem indeed to emerge from this perception of logical unsoundness, and from the willingness of interlocutors to get out of it, i.e. to *explain* it [dessalles 1992b]. In didactic contexts, one can also observe that teachers most of the time do not give immediate feed-back to their students, but prefer to show that the student's statements or actions have absurd consequences.

Conclusion

This approach, which promotes the use of a critic "looking over the student's shoulder" is not in contradiction with the I.T.S. approach. It makes use of many results which have been established in the I.T.S. field. However it is less ambitious, and in certain contexts, the "critiquing" approach is perhaps more justified from a pedagogical point of view, insofar as criticisms may be perceived by the student as relevant and useful. SAVANT3's way of arguing has been designed in order to mimic some aspects of spontaneous arguing as they occur during conversations. Our hope is that students will feel concerned by SAVANT3's remarks, because these are always uttered in context, and that they will accept them, after discussion, the way they do in everyday conversations.

Bibliography

Brown John Seely, Burton Richard R., de Kleer (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I,II,III. In Sleeman Derek, Brown J.S., *Intelligent Tutoring Systems*, Academic Press, Londres 1982, pp. 227-282

Dessalles jean louis (1990). Computer Assisted Concept Learning. In Norrie D.H., Six H.-W., *Lecture Notes in Computer Science 438 - Computer Assisted Learning*, Springer-Verlag, Berlin 1990, pp. 175-183

Dessalles jean louis (1991). Conversation Assisted Learning: The SAVANT3 Dialog Module. In Forte Eddy N., *Proceedings of Calisce'91*, Presses Polytechniques et Universitaires Romandes, Lausanne 1991, pp. 159-165

Dessalles jean louis (1992a). *Logical Constraints on Spontaneous Conversation*. TELECOM-Paris technical report 92-D-011, Paris 1992

Dessalles jean louis (1992b). Model-Based Surprise and Explanation: a way to negotiate concepts. In Brezillon Patrick, *Proceedings of the ECAI-92 Workshop on Improving the Use of KBS with explanations*, Rapp. LAFORIA 92/21 Univ. Paris VI, Paris 1992, pp. 107-113

Fischer Gerhard, Lemke Andreas C., et al. (1991). The Role of Critiquing in Cooperative Problem Solving. *ACM Transactions on Information Systems*, Vol.9, n°3, 1991, pp. 123-151

Johnson W. Lewis, Soloway Elliot (1987). PROUST: An Automatic Debugger for Pascal Programs. In Kearsley Greg P., *Artificial Intelligence & Instruction - Applications and Methods*, Addison-Wesley Publishing Company, Menlo Park, USA 1987, pp. 49-67

Self John (1988). Student Models: What Use Are They ?. In Ercoli P., Lewis R., *Artificial Intelligence Tools in Education*, North-Holland, Amsterdam 1988, pp. 73-86

Sleeman Derek (1982). Assessing aspects of competence in basic algebra. In Sleeman Derek, Brown J.S., *Intelligent Tutoring Systems*, Academic Press, Londres 1982, pp. 185-199

Stevens Albert, Collins Allan, Goldin Sarah E. (1979). Misconceptions in student's understanding. *Int. J. Man-Machine Studies* 11, 1979, pp. 145-156

VanLehn Kurt (1981). *On the Representation of Procedures in Repair Theory*. CIS-16 (SSL-81-7), Xerox Palo Alto Research Center, 1981

VanLehn Kurt (1988). Toward a Theory of Impasse-Driven Learning. In Mandl Heinz, Lesgold Alan, *Learning Issues for Intelligent Tutoring Systems*, Springer Verlag, Berlin 1988, pp. 19-41

Wenger Etienne (1987). *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann Publishers, INC., Los Altos, Cal.,USA 1987

Woolf Beverly P (1987). Theoretical Frontiers in Building a Machine Tutor. In Kearsley Greg P., *Artificial Intelligence & Instruction - Applications and Methods*, Addison-Wesley Publishing Company, Menlo Park, USA 1987, pp. 229-268