Concepts and Procedures in Engineering Education: Designing Specific Teaching Aids*

jean louis Dessalles
martin Rajman
Télécom Paris - Département Informatique
46 rue Barrault - 75634 PARIS Cedex 13 - France
Tel.: (33 1) 45 81 75 29 - Fax: (33 1) 45 81 31 19
E-✉: dessalles@inf.enst.fr
rajman@inf.enst.fr

march 1992

abstract

Most developments in the area of computer assisted teaching address the problem of helping students acquire new procedures or skills. Acquisition of new concepts, and correction of misconceptions detected during problem solving, is often considered as irrelevant or un important.

However engineers have to adapt continuously to a rapidly changing world. Merely mastering a few skills proves to be insufficient. Precise knowledge of technical concepts is also essential.

Presenting conceptual knowledge in a context of skill acquisition or of problem solving may be of course a good way to teach concepts. But this approach is not the only one, and should involve a qualitative measure of student's performance: errors are indeed important and complex events that are often linked to underlying misconceptions.

We will present the SAVANT 3 system that was specifically designed to teach concepts. This system is able to argue with the student about technical notions. Our aim is to reproduce the kind of relation that may exist between a student and a human private teacher. We believe that a good way to achieve this is to try to mimic argumentation as we observe it in natural conversations.

SAVANT 3 is still in development. Its main feature, compared to other ICAL systems, is that authoring time is reduced: less than 20 logical rules are typically necessary for SAVANT 3 to discuss about a given topic.

keywords: ICAL - egineering education - concepts - conversation - learning

---

* This research has been done in TELECOM-Paris which is part of FranceTelecom University

Bugs and misconceptions

In most of 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 underestimated (see [dessalles 1990a]). Very often the knowledge to be taught is purely conceptual by nature. Considering that negotiation, as it occurs in spontaneous conversation, is a good way to transmit conceptual knowledge, we designed the SAVANT 3 program and gave it some abilities to sustain conversations with the learner about new notions.

It is of crucial importance to understand the origin of errors occurring when the student is performing a task, in order to prevent them to occur again. Two kinds of causes have been invoked to explain errors during action: procedural flaws (bugs and mal-rules) and conceptual flaws (misconceptions). The former are often a good tool to understand and even predict errors in procedural learning: the behavior to be acquired can be modeled by a procedural graph ([Van Lehn 1980, 1988]) or a set of production rules (see [Sleeman 1982], [Payne 1990]), and errors are then explained by bad rules (mal-rules) or correct rules improperly applied (bugs). Misconceptions, on the other hand, have been mentioned more often in conceptual contexts ([Stevens 1979]).

After having gone through a purely procedural learning process, the student has acquired automatisms (e.g. integer division algorithm), but is unable to justify any of his actions (why do I pull down the next digit, why do I subtract,...) by explaining why it is right to do so and why a given alternative action may be wrong. In such a context, it seems, as shown by Sleeman [1989] and mentioned by Anderson [1992], that providing conceptual details about a given error is inefficient compared to mere procedural reteaching ! Procedural skills seem thus to be disconnected from any conceptual construction.

However when the task becomes complex enough, so that it cannot be performed by merely following a given set of rules (e.g. writing a computer program having a given behavior), it seems that students have to understand their errors in order to progress. In PROUST for example errors are explained in conceptual terms like "sentinel test" ([Johnson 1987]). Nowé and Jonckers [1991] mention the use of "abstract programming concepts" to report high level diagnoses to the student.

Our approach does not consist in trying to diagnose a possible student's misconception directly from the actions performed. The basic principle underlying SAVANT 3 is that any misconception will become apparent through its unsound logical consequences, and that it will be corrected by the student herself when confronted to some contradiction and after an argument exchange.

Teaching conceptual knowledge with SAVANT 3

Let us consider the following excerpt which illustrates the kind of interaction that can occur between student and machine.

[interaction with SAVANT3]
context: this dialog is proposed as an introduction to a digital communication course. Some concepts like digital symbols were presented, and then a printed page is displayed.

S1- Can we consider a printed text as a digital message ?
E1- sure
S2- According to what I understood: this typed page can be considered as a digital message. We have to consider symbols allowing a complete description of the page
E2- characters
S3- yes... mmm... 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?
E3- you mean if there are drawings?
S4- But there may be a problem ... If we consider that one wants to code for figures (schemas, photos, signature)...
E4- but this is a printed page
S5- yes... According to what I understood: one chooses a TELETEXT-like coding. Printed characters are not the only thing to be coded...
E5- no
S6- Could you say more?...
E6- margins, paragraphs
S7- yes... mmm... But wait... According to what you said: one wants to code for positions (paragraphs, etc.). I must say I'm confused !
E7- you use special codes
S8- well, well!... Okay...

The functioning of SAVANT3 is quite simple, and reminds of the entrapment strategy used in WHY ([Collins 1976]). The knowledge given to SAVANT3 on a specific subject consists of a set of propositions (e.g. "*one chooses a TELETEXT-like coding*") and of several logical constraints on these propositions. These constraints are stored as incompatibilities:

- one chooses a FAX-like coding
- one chooses a TELETEXT-like coding
- one does not choose a mixed coding (text + pictures)

- one wants to code for figures (schemas, photos, signature)
- the points (pixels) in the page are not taken as symbols

- one chooses a TELETEXT-like coding
- one wants to code for positions (paragraphs, etc.)
- word processing control marks do not belong to the symbols

etc.

SAVANT 3 typically needs between 15 and 25 such incompatibilities to handle a dialogue with students on a given topic. In the previous excerpt, the program knows after E2 that characters are taken as symbols. It can deduce that pixels are not taken as symbols, using the first incompatibility given above (and a few others). It thus asks the question S3 and makes the suggestion S4 in order to "trap" the student into the second incompatibility. After E6, another incompatibility (the third one given above) is almost verified. But the student could escape with E7, which denies its last term.

Two points are worth noting here. First the central role played by logical argumentation. Teaching concepts cannot be limited to the display of definitions or to simple exercises. SAVANT 3 continuously attempts to trap the student in one of its incompatibilities, trying another one if the current incompatibility becomes invalidated. But since the same strategy has been observed in natural conversation (cf. [dessalles 1990b]), we hope that the student will not experience it as a kind of chase, but on the contrary will perceive the machine's replies as relevant.

The second point here is that the author has not to bother about the linking of replies. The simple entrapment strategy used by SAVANT 3 produces many different potential dialogues. Here for instance, the student could have chosen to perform a FAX-like coding, and the system would have argued differently.

Acquiring new skills with SAVANT 3 *

In order to study the need of conceptual knowledge when performing a task - here writing a simple program in Prolog -, we develop a system which gives the student the opportunity to run a simulation on her/his program, but also to present the program to the critical look of SAVANT 3, which is used here to argue about the correctness and the efficiency of the student's program. The student may thus try her/his program and observe its execution, and then "discuss" with SAVANT 3 about correctness and efficiency of the written clauses.

The student is asked to write a small Prolog program that computes the following relation *Car(X,Y)* :

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

The program is supposed to perform the calculation of the amount of taxes someone has to pay for his car (second argument), given his earnings (first argument). The student can write a program to solve this exercise (the number of possible programs he may write has been cut down by offering a very limited editor), and at every moment, he can ask to execute it: the student supplies the first argument (*Earnings*) of *Car(Earnings,Taxes)*, after which a Prolog interpreter finds all the possible solutions for *Taxes* while a trace window shows which clauses are called and if they succeed or fail.

---

* This part describes a work done in collaboration with Pascal Meyers of Catholic University o Leuven (KUL), Belgium

The Prolog interpreter may just point to abnormal behavior of a program, but it does not link this to possible misconceptions about the fact that backtracking occurs and that some programs may give multiple solutions, or about the meaning of cuts, etc. This is the role devoted to SAVANT 3.

The student can ask SAVANT 3 to look at his program at any moment. He is first asked if he considers his program as correct and efficient, and then SAVANT 3 tries to detect logical inconsistencies. The conversation stops as soon as the student's program is modified, or when no further argument can be found.

student's program:

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

----- *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 !

----- *why is it surprising?*

But there is a problem anyway... 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?

----- [clause 2 is not called for people earning less than 4000]

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 !

----- *Earnings >= 4000 is no longer used in clause 2*

well, well!... Okay...

This kind of interaction is quite far from what can be found for example in [Anderson 1989], principally because no attention is paid here to the "principle" of immediate feedback. SAVANT 3 points possible misconceptions out to the student by highlighting their inconsistent consequences. But it never tries to elaborate or suggest any diagnosis directly in terms of bugs or misconceptions. 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.

Experimenting with SAVANT 3

The first complete course: "Second Order Systems" was distributed to Telecom Paris students in 1989, but it contained only three dialogs. For two of them, the machine could play its role correctly and many students engaged into a well-balanced and enriching conversation. The third dialog encountered less success: it began with the question: "What do you think of the comfort of the suspension, in your car?", and the students answered using analogies, names of specific car models, or even slang! The first conclusion is that we must restrict the dialogs to technical subjects for which the context is limited and can be anticipated. Since this first experiment, SAVANT 3 has been greatly improved but not yet used by students.

One of the most interesting features of SAVANT 3 is that the system has not to be redesigned or even modified when used on different topics. Only the knowledge base is content-dependent. It is usually very small: usually around 20 rules, generated by authoring programs. But when dealing with procedural learning as was the case above with Prolog programming, SAVANT 3 had to work on larger knowledge bases (more than 100 rules). Here, the conversion of the small Prolog program into a set of rules was made empirically, and proved to be tricky. We have to think about ways to generate rules systematically.

SAVANT 3 was designed to help students master new concepts, either in a purely conceptual context, or during procedural performance.

In the first case, our hope is that SAVANT 3 will prove to be an efficient way to learn concepts, but also a pleasant one, so that it may become a usual tool for students. That is why we tried to reproduce in SAVANT 3 some of the argumentation capabilities that we use in our daily conversations.

But our aim is also to explore if SAVANT 3 can be accepted by the student as a criticizing tutor during problem solving. We hope first to be able to prove the feasibility of our approach: eliminate misconceptions in the student's knowledge by noticing inconsistencies and generating argumentation, and thus emphasize the relevance of conceptual knowledge in procedural learning.

references

Anderson John R., Conrad Frederick G, Corbett A.T. (1989): *Skill Acquisition and the LISP Tutor*, Cognitive Science 13, 1989, pp. 467-506

Anderson John R. (1992): *Interview, by J. Sandberg and Y. Barnard*, AIcommunications vol. 5 n° 1, 1992, pp. 28-31

Collins Allan (1976): *Processes in Acquiring Knowledge*, in Schooling and Acquisition of Knowledge, Anderson, Spiro and Montague eds, Erlbaum Assoc., Hillsdale NJ 1976

Dessalles Jean-Louis (1990a): *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 (1990b): *The simulation of conversations*, in Kohonen Teuvo, Fogelman-Soulié Françoise: *COGNITIVA 90 - Proceedings of the Third Cognitiva Symposium (Madrid)*, North Holland, Amsterdam 1991, pp. 483-492

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

Nowé Ann, Jonckers V. (1991): *The Use of Very High Level Clichés in PATAT: a Program Analysis Tool Using Algorithm Transformation*, in Forte Eddy N.: *Proceedings of Calisce'91*, Presses Polytechniques et Universitaires Romandes, Lausanne 1991, pp. 209-216

Payne Stephen J., Squibb Helen R. (1990): *Algebra Mal-Rules and Cognitive Accounts of Error*, Cognitive Science 14, 1990, pp. 445-481

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

Sleeman Derek, Kelly A.E., Martinak R. (1989): *Studies of Diagnosis and Remediation with High School Algebra Students*, R.D. Ward, J.L. Moore - Cognitive Science 13, 1989, pp. 551-568

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, Brown John Seely (1980): *Planning Nets: A Representation for Formalizing Analogies and Semantic Models of Procedural Skills*, in R.E. Snow, P.A. Frebenco, W.E. Montague eds: Aptitude, Learning ans instruction - LEA, 1980, pp. 95-137

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