REPRINTED FROM:

ADVANCED RESEARCH ON COMPUTERS IN EDUCATION

Proceedings of the IFIP TC3 International Conference on
Advanced Research on Computers in Education
Tokyo, Japan, 18–20 July, 1990

Edited by

ROBERT LEWIS
Department of Psychology
University of Lancaster
Lancaster, U.K.

SETSUKO OTSUKI
Department of Artificial Intelligence
Kyushu Institute of Technology
Iizuka, Japan

1991

NORTH-HOLLAND
AMSTERDAM · NEW YORK · OXFORD · TOKYO
ISBN 0444887881

Advanced Research on Computers in Education
R. Lewis and S. Otsuki (Editors)
Elsevier Science Publishers B.V. (North-Holland)

The Relevance of Computational Models of Knowledge Acquisition for the Design of Helps in the Problem Solving Monitor ABSYNT*

Claus Möbus**

Department of Computational Sciences
University of Oldenburg
D-2900 Oldenburg, F.R. Germany
Eunet: moebus@uniol.uucp

Computational models of knowledge acquisition are indispensable for the design of intelligent tutoring systems. They give advice how to design *instructions, helps and explanations.* We want to show how two kinds of models (*external* and *internal*) are useful for the design of problem solving monitors (PSMs). Especial the quality of helps is crucial for the acceptance of a PSM. To put it short: *"When are helps useful and when are they confusing or pose new problems to the learner?"*

1. Introduction

This paper offers a contribution to ICAI in the framework of the problem solving monitor ABSYNT. Our system - a special variant of an Intelligent Tutoring System (ITS) - is designed with respect to a sequence of 37 programming tasks which are to be solved by students in the visual functional computer language ABSYNT (ABstract SYNtax Trees). Besides providing the learner with a friendly problem solving environment including a *help component.* it serves us as a testbed for research in the domain of *intention-based diagnostics, plan-parsing* and *design of helps* for problems solvers.

Research in these domains cannot be done without studying the *knowledge acquisition process* of the student. Learning processes are modelled by *computational learning models* (e.g.[1], [2]). In the domain of PSMs we distinguish external and internal computational models. An *internal* model is an integrated part of the PSM and is usually termed *"student model"* [3]. Its main purpose is the user-tailored generation of instructions, helps and explanations. An *external* model is not a functional component of a PSM but is developed in parallel using a broad data basis to gain a more complete insight into the learning process of the subject. At the present state of art these models will represent the knowledge acquisition process and the knowledge state of the student at different grain sizes and ranges. One of the reasons for this discrepancy is the fact that PSMs are at the present moment unable to analyse the full range of problem solving behavior which includes verbal data [4]. Internal models are based on data which the PSM can gather online, whereas our external models are based on videotaped problem solving sessions of dyades, which contain verbal episodes.

---

* ABSYNT was made reality by K.D.FRANK, G.JANKE, K.KOHNERT, O.SCHRÖDER and H.J.THOLE

** This research was sponsored by the Deutsche Forschungsgemeinschaft (DFG) in the SPP Psychology of Knowledge under grant no. MO 292/3-3

We think that the development of PSMs or ITSs should include *both* questions: First, how should the learning process be modelled with an external model to develop *hypotheses for the design* of optimal helps and second how should the student model acquire knowledge to *generate* actual user-tailored helps. Modeling the knowledge acquisition process of students with external models has led us to the conclusion that learning processes in our domain can be adequately described by a combination of an *impasse-driven* (IDL) [5] and *success-driven* (SDL) [6]-[7] learning theory (IDL-SDL-Theory) [8]-[10].

IDL-SDL makes predictions *when* the student will *accept* information as help, *when* s/he even actively will *search* for new information and *what* content of information will *suit* the students needs. This has practical consequences for the construction of PSMs. The *design of interactive and adaptive helps* requires the successful solution of a *synchronization problem* between the knowledge state of the learner and the diagnosis of the PSM concerning this state: the student model. So in our PSM the *update* of the internal student model and the *provision* of help information follows IDL-SDL-Theory developed with external models.

2. The Problem-Solving Monitor ABSYNT

PSMs provide the learner with a problem-solving environment including a diagnosis but no curricular component. ABSYNT is used to communicate knowledge about a visual, purely functional, tree-like visual programming language based on ideas published in german school [11] and university text books [12]. Further motivation for the design of ABSYNT is given in [13]. Basic research dealing with the design of the system from a *psychological* point of view is described in [14] - [17].

ABSYNT provides an *iconic environment* and is aimed at supporting the acquisition of functional programming concepts up to *recursive systems*. A program consists of a head and a body tree. Also there is a start tree from which programs can be called. The nodes of the trees are constants, parameters, primitive and self-defined operators. The connections between the nodes are the "pipelines" for control and data flow. Programs are edited by taking nodes with the mouse from a menu bar and connecting them.

On demand there is also a visual trace which was implemented according to the runnable specification of the interpreter [15]. Additionally the user can test hypotheses about the correctness of her/his implementations. Figures 1 and 2 depict snapshots of the interface when a student has programmed a wrong solution of the problem "even" and tries to propose some hypotheses about the usefulness of parts of his program. The answers to her/his hypotheses are generated by rules defining a goals-means-relation (GMR; more details below). This feedback can be viewed as *helps* from the system *on the language level.*

3. Rule-based Help for the Acquisition of Semantic and Planning Knowledge

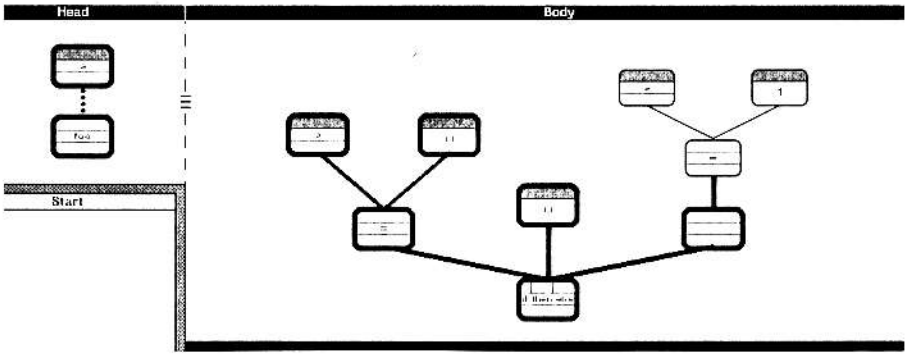
It is standard theory in cognitive science to assume that programming requires the *activation* and *application* of at least four knowledge sources:

1. *mathematical* and *algorithmical* preknowledge
2. knowledge about the *syntax* of the language
3. knowledge about the *semantics* of the language
4. *planning* knowledge about the pragmatical use of the language

It is quite natural to design helps accordingly. In our research we confine ourselves to the two last knowledge sources.

We designed:

ad 3: *2-D-rules* describing the operational *semantics* of the ABSYNT-language [16],[18]
ad 4: *Planning rules* which describe *programming knowledge* in ABSYNT [19]



No: Your hypothesis cannot be completed to a solution known by the system.

Figure 1: A snapshot of the ABSYNT-interface showing an incorrect program with a user hypothesis (**bold**) and the systems feedback.

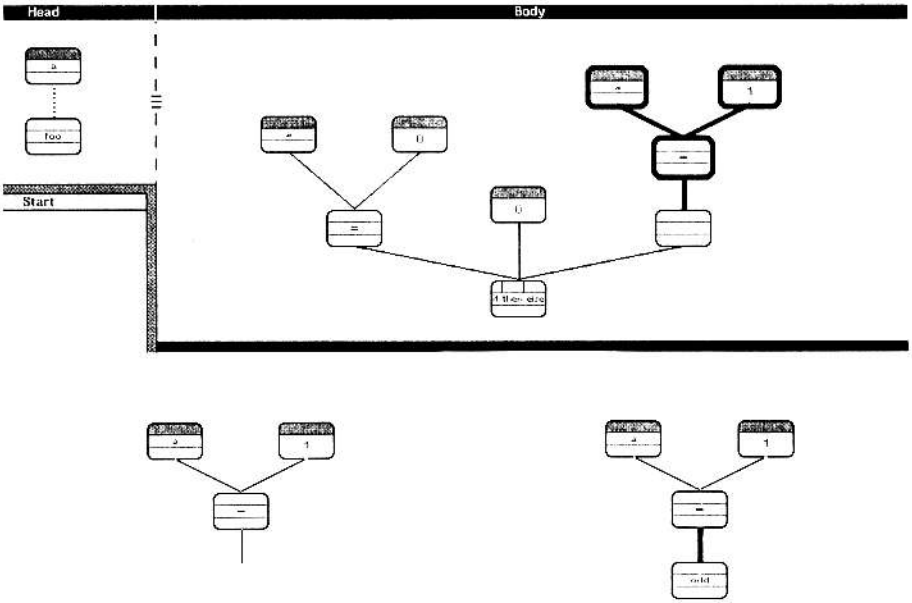
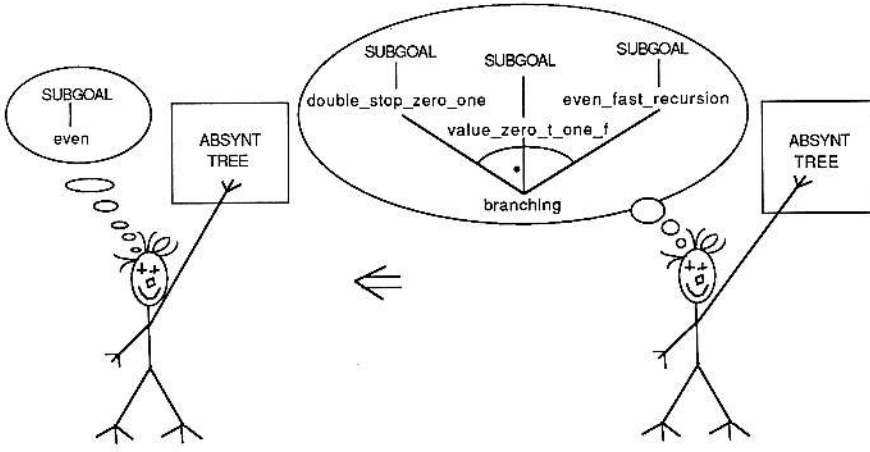


Figure 2: The ABSYNT-interface showing another user hypothesis. The system returns the hypothesis (lower half on the *left*) to indicate its correctness. On demand (**bold line**) the system shows the next node of a complete solution (lower half on the *right*).

The behavior of the ABSYNT-interpreter can be predicted by the knowledge of 18 "state-centered" *semantic rules*. They were represented as two-dimensional visual rules which serve as help material for ABSYNT-users.The complete set can be found in [18].

Planning knowledge for 37 tasks is represented in 462 rules which define the GMR. The GMR can be looked at as a *rule-based inference system*, a *grammar* or an *AND/OR-Graph* with parametrized nodes. The rules are similar but more powerful than those found in [1],[2],[20]. The GMR is able to *analyse* and *synthesize* several millions of solutions even if the height of ABSYNT-trees is restricted to five nodes. Because nodes of the AND/OR graph can be parametrized for subgoals, the relation enables analysis and synthesis of even *partial* solutions which enables the testing of user hypotheses (Figures 1, 2). An example for the graphical and natural language compilation of one planning rule is given in Figure 3.

Rule: "Planning a Recursion on the Goal level"



IF the **main goal** is to program the ***even predicate which can be applied to a subgoal***

THEN the solution of this goal comprises the following step:

- *leave space* in the worksheet of the ABSYNT environment for the yet to be programmed ABSYNT tree

AND

IF your next planning step creates the more differentiated AND-goal tree ***branching(...)***

THEN the solution of this **new goal** is a ABSYNT tree which can be inserted in the solution of the **main goal**

Figure 3

The GMR is defined by the *planning rules* and represents the core of the help system at the *language level*, which is designed according to some postulates. It should:

- *offer the environment to check various hypotheses* about the usefulness of several parts of the program proposed by the student
- *embody expert knowledge* to generate helps or solution proposals
- *diagnose* goals, intentions and the knowledge state of the problem solver
- *communicate new knowledge* (helps) only in *sensitive* time periods, where the problem solver is willing to accept such information [5]
- *gather data* from the hyothesis-testing process *online* to adapt the internal student model continuously
- *deliver only minimal information* so that the student is able to leave the impasse situation by his own thus improving his problem solving skills

This *interactive hypothesis driven approach* is rather different from other systems known from literature [1], [21]-[25] and is a direct consequence of IDL-SDL-Theory (see below).

4. External and Internal Computational Models of Knowledge Acquisition

4.1 An External Model for the Acquisition of Rule Knowledge on the Basis of Visual Helps

A necessary prerequisite of programming is some knowledge about the syntax and semantics of the language. We studied the acquisition of *semantic knowledge*. The semantics of programming languages can be defined in three ways: (a) the *operational* approach, (b) the *denotational* approach and (c) the *axiomatic* approach. We chose the operational approach because it seemed to us more suitable for novices than the others. The behavior of the ABSYNT-interpreter is represented by *two-dimensional (2-D) visual rules* which were supplied as help material in case of difficulties or impasses. We asked subjects to predict the actions of the ABSYNT-interpreter [8]-[10]. The results dealing with knowledge acquisition can be described by an *iterative two-stage simulation model* which is capable of predicting 60% of continuous portions of encoded protocols [10]:

1. *Knowledge acquisition* by impasse-driven learning:
   Difficulties [26], [27] or impasses [5], [28] lead to problem solving by the application of *weak heuristics*. With the help of the visual rules *new* knowledge about the semantics of ABSYNT is stored in memory.

2. *Knowledge optimization* by success-driven learning:
   Due to practice, the knowledge is reorganized so that it can be used more efficiently. In the similation model this is done by the *composition of rules* to compound rules [29],[30] and macro-operators like rule nets [31].

The data show that the subjects predicted the behavior of the interpreter and the computation of programs on the basis of *mental rules* or *mental macro-operators*. They used help information *only in nonoptimization* stages of the process. The question is, whether to adapt the help material accordingly. That is to offer visual rules and visual macro-rules *synchronized* to the mental operators the students use.

4.2 An Internal Model for the Acquisition of Rule Knowledge on the Basis of Checking Users Hypotheses

Our domain makes it absolutely necessary to *constrain* the overwhelming large *feedback space* by an internal model (student model) which is not implemented yet. It is not unusual that a user hypothesis can be completed by a hundred solutions. Even when only the *next* node is shown there are too many possibilities to choose from. IDL-SDL-Theory tells us to propose only helps which the user can assimilate according to his knowledge state. Generation of appropriate helps must be done by the student model which has to be learned automatically.

A *knowledge state* is viewed as a set of rules, malrules, and their composites. The student model consists of that set of rules which can generate the implementations and which can be derived from the student's proposals of hypotheses.

The *acquisition of rules* and their composites is easy. Those rules which were used for successful parsing make a chain of partially instantiated planning rules. These rules can be composed and generalized according to [29, 30] to *higher planning schemes*. Composing n successive rules results in an n-th order scheme. The highest scheme is a rule which relates a programming task to a complete solution: an example.

The *acquisition of malrules* is a bit trickier. Programs are trees. Selecting a subtree for a hypothesis is equivalent to *cut* the tree (Figure 4 left). With our GMR it is possible to reconstruct various goals depending on which tree is used for the *reconstruction of the goals*. We can reconstruct the root goal of the *whole* tree, the root goal $Z_{A'}$ of the *selected* tree, the root goal Z of the *not selected* tree and the context of $Z_{A'}$ in the not selected but *automatically completed* tree which is $<Z_A,Z_B,Z_C>$. These Z's are goals of the subtrees within the right side of the *same* planning rule. If $Z_{A'}$ is not equivalent to Z_A then the selected tree implements a wrong goal. From this information (goal conflicts) we can generate malrules, their composites, and error explanations.

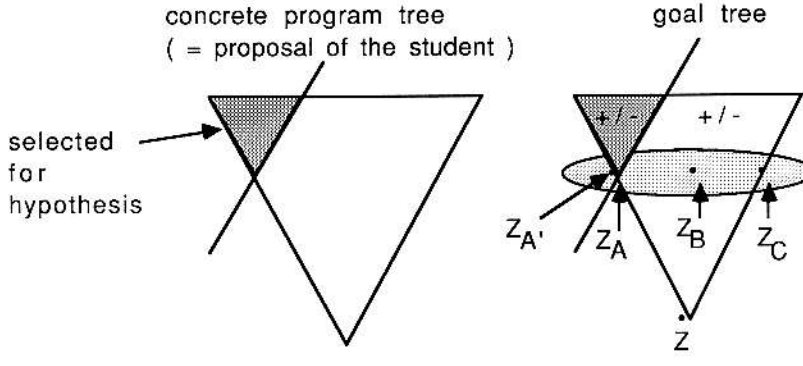


Figure 4: A schematic ABSYNT-program and the corresponding goal trees

5. Conclusions for the Design of Helps in PSMs

In the near future we will convert the rules of the GMR into visual planning helps (see Figure 3 as an example) *on the goal level*. But additional design features are necessary, which are recommended by our external and internal model:

Our *external* model allowed the further development of IDL-SDL-Theory. That theory gives two main advices to PSM-builders:

- The *content* of helps has to be syncronized to the knowledge state consisting of rules and macrooperators.
- Helps should only be offered at impasse *time*. Best is to let the user ask for help.

Our *internal* model (yet to be implemented) will
- automatically learn rules and malrules at impasse time (hypothesis test time)
- constrain the feedback space
- explain errors on the basis of goal conflicts or goal collisions: implementations vs. task requirements

Both models give valuable information for the design of PSMs: The *external* model for *general design decisions*, the *internal* model for *concrete help generation.*

REFERENCES

[1] ANDERSON, J.R.: Production Systems, Learning, and Tutoring. In: KLAHR, D., et al. , Production System Models of Learning and Development, 437-458
[2] ANDERSON, J.R., A Theory of the Origins of Human Knowledge, Artificial Intelligence, 40, 1989,313-351
[3] KASS, R., Student Modeling in Intelligent Tutoring Systems - Implications for User Modeling, in: KOBSA, A. & WAHLSTER, W. (eds): User Models in Dialog Systems, Berlin-Heidelberg-New York: Springer, 1989
[4] VanLEHN, K. & GARLICK, S., Cirrus: An Automated Protocol Analysis Tool, in: P. LANGLEY (ed.), Proceedings of the 4th International Workshop on Machine Learning, Los Altos, CA: Kaufman, 1987, 205-217
[5] VanLEHN, K. Towards a Theory of Impasse-Driven Learning, in: H.MANDL & A.LESGOLD (eds), Learning Issues for Intelligent Tutoring Systems, Berlin-Heidelberg-New York: Springer, 1988, 19-41
[6] NEVES, D.M., ANDERSON, J.R., Knowledge Compilation: Mechanisms for the Automatization of Cognitive Skills, in: ANDERSON, J.R. (ed): Cognitive Skills and their Acquisition, Hillsdale: Erlbaum, 1981, 57-84
[7] WOLFF,J.G., Cognitive Development as Optimization, in: L.BOLC (ed), Computational Models of Learning, Berlin-Heidelberg-New York: Springer, 1987, 161-205
[8] SCHRÖDER, O., FRANK, K.D., KOHNERT, K., MÖBUS, C., RAUTERBERG, M., Instruction-Based Knowledge Acquisition and Modification: The Operational Knowledge for a Functional, Visual Programming Language, Computers in Human Behavior, Special Issue: German Experimental Research in Learning and Instruction with Computers, Vol. 6, No. 1, 1990, 31-49
[9] SCHRÖDER, O., KOHNERT, K., Toward a Model of Instruction-Based Knowledge Acquisition: The Operational Knowledge for a Functional Visual Programming Language, Journal of Artificial Intelligence in Education, Vol. 1, No. 2, 1989/90, 105-128
[10] SCHRÖDER, O., A Model of the Acquisition of Rule Knowledge with Visual Helps: The Operational Knowledge for a Functional, Visual Programming Language, Proceedings of the 3rd International Conference on Computer-Assisted Learning (ICCAL '90), Berlin-Heidelberg-New York: Springer (in press)

[11] SCHMITT, H. & WOHLFARTH, P., Mathematikbuch 5N, München: Bayerischer Schulbuchverlag, 1978
[12] BAUER, F.L. & GOOS, G., Informatik: Eine einführende Übersicht, Erster Teil, Berlin: Springer 1982
[13] DOSCH, W., New Prospects of Teaching Programming Languages, in: F.B. LOVIS & E.D. TAGG (eds), Informatics Education for All Students, Elsevier Science Publishers B.V. (North-Holland), IFIP 1984, 153-169
[14] MÖBUS, C., Die Entwicklung zum Programmierexperten durch das Problemlösen mit Automaten, in: H.MANDL & P.M.FISCHER (ed), Lernen im Dialog mit dem Computer, München: Urban & Schwarzenberg, 1985, 140-154
[15] MÖBUS, C. & THOLE, H.-J. Tutors, Instructions and Helps. In: CHRISTALLER, Th. (ed): Künstliche Intelligenz KIFS 1987, Informatik-Fachberichte 202, Heidelberg, Springer 1989, 336-385
[16] MÖBUS, C. & SCHRÖDER, O., Knowledge Specification and Instructions for a Visual Computer Language. In: KLIX, F. et al. (eds): Man-Computer Interaction Research Macinter II, Proceedings of the second Network Seminar of Macinter held in Berlin /GDR, 21. - 25. März 1988, Amsterdam: North Holland, 1989, 535-565
[17] MÖBUS, C., Toward the Design of Adaptive Instructions and Helps for Knowledge Communication with the Problem Solving Monitor ABSYNT, in: STEPANKOVA, O. & MARIK, V.(eds), Proceedings of the CEPES UNESCO Workshop, "The Advent of AI in Higher Education", Prague, October, 23-25, 1989, Heidelberg-Berlin-New York: Springer (in press)
[18] MÖBUS, C. & SCHRÖDER, O., Representing Semantic Knowledge with 2-Dimensional Rules in the Domain of Functional Programming, in: TAUBER, M., GORNY, P. (eds), Visualization in Human-Computer Interaction. Heidelberg, Springer: Lecture Notes in Computer Science, Berlin (in press)
[19] MÖBUS, C. & THOLE, H.J., Interactive Support for Planning Visual Programs in the Problems Solving Monitor ABSYNT: Giving Feedback to User Hypothesis on the Language Level, Proceedings of the 3rd International Conference on Computer-Assisted Learning (ICCAL '90), Berlin-Heidelberg-New York: Springer (in press)
[20] VanLEHN, K., Learning One Subprocedure per Lesson, Artif. Intelligence, 1987,31,1-40
[21] HOUGHTON, R.C., Online Help Systems: A Conspectus, Communications of the ACM, 27, 2, 1984, 126-133
[22] KEARSLEY, G., Online Help Systems, Design and Implementation. Norwood: Ablex, 1988
[23] JOHNSON, W.L., Intention-Based Diagnosis of Novice Programming Errors, Research Notes in Artificial Intelligence, London: Pitman, 1986
[24] MURRAY, W.R., Automated Program Debugging for Intelligent Tutoring Systems, Research Notes in Artificial Intelligence, London: Pitman, 1988
[25] GREER, J.E., MARK, M.A. & McCALLA, G.I.: Incorporating Granularity-Based Recognition into SCENT. In: BIERMANN, D., BREUKER, J., SANDBERG, J. (eds): Artificial Intelligence and Education, Amsterdam: IOS, 1989, 107-115
[26] LAIRD, J.E., ROSENBLOOM, P.S. & NEWELL, A., Chunking in SOAR: The Anatomy of a General Learning Meachnism, Machine Learning, 1986, 1, 11-46
[27] LAIRD, J.E. NEWELL, A. & ROSENBLOOM, P.S., SOAR: An Architecture for General Intelligence, Artificial Intelligence, 33, 1987, 1-64
[28] BROWN, J.S. & VanLEHN, K., Repair Theory: A Generative Theory of Bugs in Procedural Skills, Cognitive Science, 1980, 4, 379-426
[29] VERE, S., Relational Production Systems, Artificial Intelligence, 1977, 8, 47-68
[30] LEWIS, C., Composition of Productions, in KLAHR, LANGLEY & NECHES (eds), Production System Models of Learning and Development, Cambridge, Mass.: MIT Press, 1987, 329-358
[31] CHENG, P. & CARBONELL, J.G., The FERMI System: Inducing Iterative Macro-operators from Experience, Proceedings of the 5th National Conference on Artificial Intelligence, Cambridge: Morgan Kaufman, 1986, 490-495