Toward the Design of Adaptive Instructions and Helps for Knowledge Communication with the Problem Solving Monitor[1] ABSYNT[2]

Claus Möbus

University of Oldenburg
Dept. of Computational Sciences
D-2900 Oldenburg
W. Germany
Eunet: moebus@uniol.uucp

0. Abstract

For approximately ten years computer aided knowledge communication disappeared from the research scene. Today, it has been reestablished under the abbreviations of ICAI (Intelligent Computer Aided Instruction) and ITS (Intelligent Tutoring Systems) with regular conferences, research journals and textbooks [1,2,3,4,5].

This paper offers contributions to CAI and ICAI in the framework of the problem solving monitor (PSM) ABSYNT. Our system - a special variant of an ITS - is designed with respect to a sequence of programming tasks in the visual functional computer language ABSYNT (ABstract SYNtax Trees). It provides the learner with a friendly environment including a help but no curricular component.

First, we show that conventional instructions and helps can be improved by using existing AI methodology, visualization of information and cognitive modelling to make them adaptive to the knowledge state of the user. Second, we demonstrate the improvement of ICAI by an interactive help system which supports planning tasks of the user. It checks hypotheses postulated by the user, and gives feedback concerning imcomplete proposals.

1. Introduction

The "Advent of AI in Higher Education" is a positive event, but we should not overlook that [6]:

"ICAI is an emerging field that is ill-defined at present. The distinction between intelligent CAI systems and computer-based instruction programs cannot be sharply drawn. ICAI programs use AI programming techniques ... Developers of ICAI systems focus on problems of knowledge representation, student misconceptions, and inferencing. By and large, the have ignored instructional theory and past research findings in computer-based instruction."

In our project we try to avoid these omissions. To a great extent the psychological efficiency of a PSM depends on the quality of instructions and helps built into the system. To put it short: "When are helps useful and when do they confuse or inhibit the problem solver?" The answer certainly depends on the knowledge state of the problem solver and is a research problem of cognitive psychology. The implementation of helps requires AI methodology, and thus the design of helps is a paradigmatical research topic of AI, Cognitive Science and Cognitive Psychology.

---

[1] This research was sponsored by the Deutsche Forschungsgemeinschaft (DFG) in the SPP Knowledge Psychology under contract no MO 292/3

[2] ABSYNT was transformed from an idea to reality by K.D.FRANK, G.JANKE, K.KOHNERT, O.SCHRÖDER & H.J.THOLE

2. The Problem-Solving Monitor ABSYNT

PSMs provide the learner with a problem-solving environment including an error diagnosis and a help module but no curricular component. ABSYNT is used to communicate knowledge about a visual, functional, tree-like programming language based on ideas published in german school [7] and university text books [8]. Further motivation for the design of ABSYNT is given in [9] and [10]. Basic research dealing with the design of the system from a psychological point of view is described in [11] - [14]. Figure 1 shows the interface of the programming environment after a student programmed a wrong solution to the "even" predicate. The program does not terminated for odd arguments.

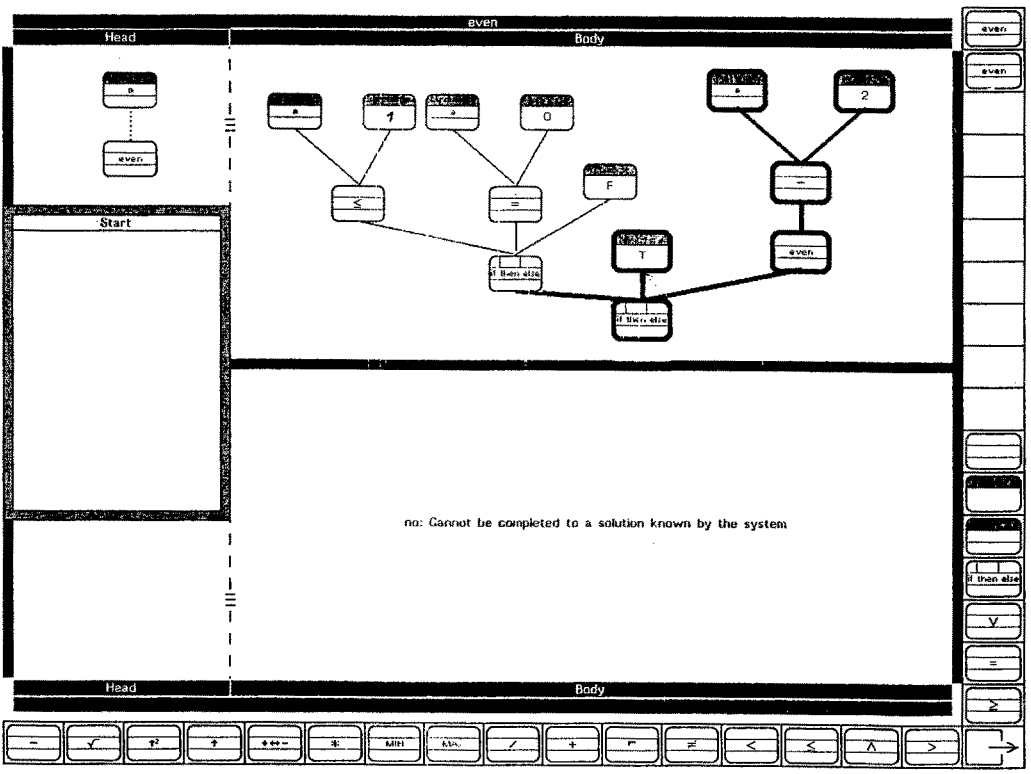

Figure 1: wrong implementation of the "even" predicate in ABSYNT with a nonembedable hypothesis (bold lines)

3. Programming Knowledge and Corresponding Helps

Programming requires several knowledge sources, see e.g. [15] and [16]:

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 would be quite natural to design helps accordingly. In our system we confined ourselves to knowledge sources 3 and 4. We designed:

ad 3: *2-D-Rules* [14] explaining the operational semantics of the ABSYNT-language (the behavior of the ABSYNT-Interpreter)
ad 4: a *Hypothesis-Driven Help System* to support planning steps in programming [17]

By and large, the design of helps is a twofold synchronization problem. We have to choose both content and application time of the given information very carefully so that it is accepted as help [18] - [24].

3.1 The Design of Helps when Acquiring Knowledge about the Semantics of ABSYNT

The behavior of the ABSYNT-Interpreter computing ABSYNT programs is represented by two sets of 2-D-Rules which serve as instruction and help material for ABSYNT-users [14]. In a study of the help-based knowledge acquisition process [25] we found that the acquisition of semantic knowledge can be described by a two-stage process:

1. *Knowledge enlargement* through impasse-driven learning (IDL) [26]
2. *Knowledge optimization* through success-driven learning (SDL) [27]-[31]

The inability of the student to predict the behavior of the ABSYNT interpreter leads to an impasse which in turn leads to problem solving by applying weak heuristics: new knowledge about the ABSYNT semantics is generated by looking into the 2-D-Helps. Otherwise the student optimizes his/her knowledge similar to [27],[28],[31]. A specification of a computational model for this two-stage process is given in [32].

The data show that in the course of time the subjects predict the behavior of the interpreter on the basis of mental rules which first were representations of the visual rules. Then, during the process of skill acquisition the mental rules are composed. The question arises whether to compose the visual 2-D-rules in the help material accordingly. The evaluation ABSYNT programs requires the firing of the rule chains. Composition of these chains yields the new help rules, which have lost some - for a more skilled person unimportant - details.

The idea of adaptive helps seems to be ingenious. But, will the problem solver assimilate the new information quickly even if we are able to synchronize the generation of new help information with the mental proceduralization process? It is a problem with adaptive helps that the students get unexpected new information the equivalence of which with old information has to be checked. The solution of this problem will certainly improve the acceptance of ICAI considerably.

3.2 The Design of Planning Helps when Programming in ABSYNT

Even more important is a help system for the programming novice which embodies planning knowledge. Here too, we face the twofold synchronization problem. The status of the help system implemented so far derives from the following postulates.

The help system should:

1. *diagnose* goals, intentions, and the knowledge state of the problem solver
2. *communicate new knowledge* (helps) only in sensitive time periods, where the problem solver is willing to accept such information
3. *gather user data* online to adapt the user model continuously
4. *embody expert knowledge* to check user proposals and generate helps
5. *give only minimal feedback* so that the student is able to leave the impasse situation by improving his/her problem solving skills
6. *check various hypothesis* about the usefulness of program fragments

The last postulate is rather important. In contrast to some authors (e.g.[33]) we think that semantic errors often cannot be localized in a line of code (or here in a subtree). Most often the whole proposal of the user is inconsistent with the problem. Repairs depend on those parts of the program which the user wants to retain as correct. So we developed our help system, which is driven by hypotheses of the student about the correctness or usefulness of program fragments. This interactive hypothesis-driven approach is rather different from other systems known from literature [16] and [33] - [37].

Our help system is based on a goals-means-relation (GMR). This relation can be viewed as a rule-based-inference system [38], a grammar [39], or an AND/OR-Graph [40] with parametrized nodes. As nodes of the AND/OR graph can be parametrized by subgoals, the relation enables analysis and synthesis of incomplete as well as complete program proposals and even recursive systems of programs. At present the relation is defined for 21 programming tasks. The system is able to generate and parse over several million solutions if the height of the ABSYNT-trees is less than 6 nodes. This can be achieved with approximately only 330 rules.

Due to its flexibility the help system promises some positive consequences for the motivation of the problem solver. When the student programs a proposal which is diagnosed by the ITS as wrong, s/he is trapped in an impasse. According to IDL-theory s/he is now sensitive to help information. This could be assimilated by problem solving actions. The student may ask the system to check a hypothesis about the usefulness of her/his program or program fragments. There is hope that the feedback of the system to this hypothesis is sufficient help information to overcome the impasse without further help.

Errors in functional programs are often difficult to localize. This is true for most nonsyntactic bugs. Often the only possibility is to argue that the goals various parts of the program compute are inconsistent with the main goal. In figure 1 (upper half) we have the impasse situation of subject 8. It is an inconsistent implementation of the "even" goal. There are several possibilities to localize bugs and to repair the program. The programmer is supposed to put forward positive (or negative) hypotheses like: "I presume that the **bold** marked subtree of the program can be (cannot be) embedded in a correct solution!"

Then the student has to mark this hypothesis with the mouse (bold lines in upper half of figures 1 and 2). The feedback of the system is given in the lower window of the ABSYNT environment. In figure 1 the

student gets a "nonembedable" response. In figure 2 the hypothesis is more restricted. Now, the hypothesis is embedable. For further study it is copied into the feedback window. The student can now select one link (**bold** in lower feedback window in figure 3) to uncover one node of the proposed solution and to program the correct solution (figure 4).

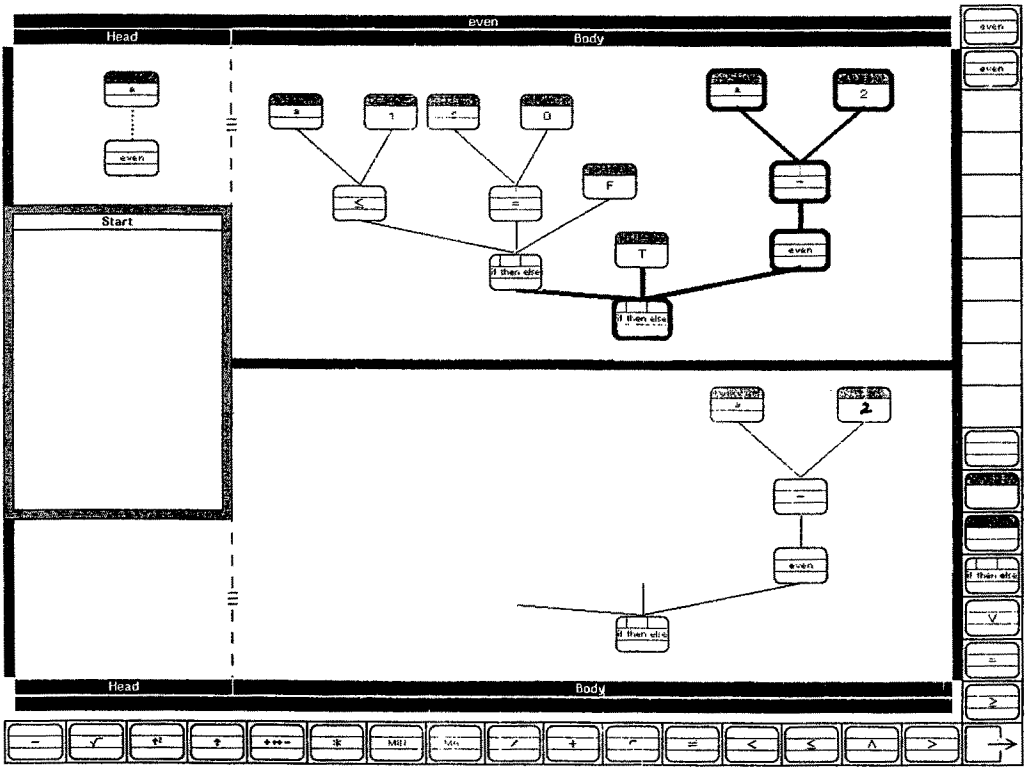

Figure 2: wrong implementation of the "even" predicate in ABSYNT with an embedable hypothesis (bold lines) and a copy of the hypothesis in the feedback window (lower half)

4. Further Research Topics

The rules of the GMR are highly standardized, so that it will be easy to use the learning mechanism of automatic rule composition [28] to speed up the system and diagnose typical problem solving schemes in only a few steps.

To further restrict the number of alternative solution proposals we need a user model which filters the proposals, so that the feedback information is helpful and does not generate new subproblems. This can be achieved by storing the hypotheses, their results, and the corresponding actions and repairs of the problem solver.

As our helps are at present helps on the low operator level we work on explanations and helps on the higher goal level. One possibility is the study of goal conflicts between task and student proposal and how this goal conflict can be reduced.

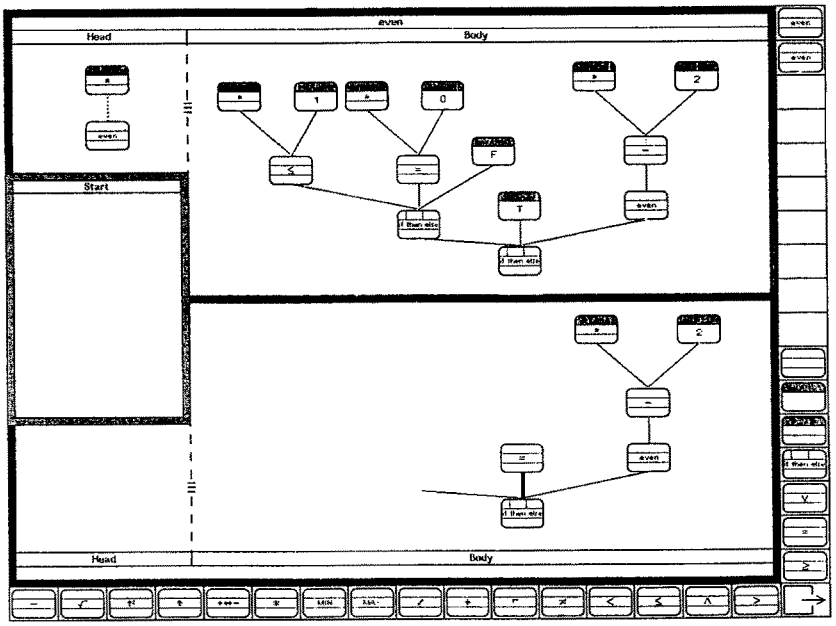

Figure 3: feedback of the help system: first the student selects an open link of the hypothesis (bold line in lower feedback window), then the machine uncovers one node of a solution

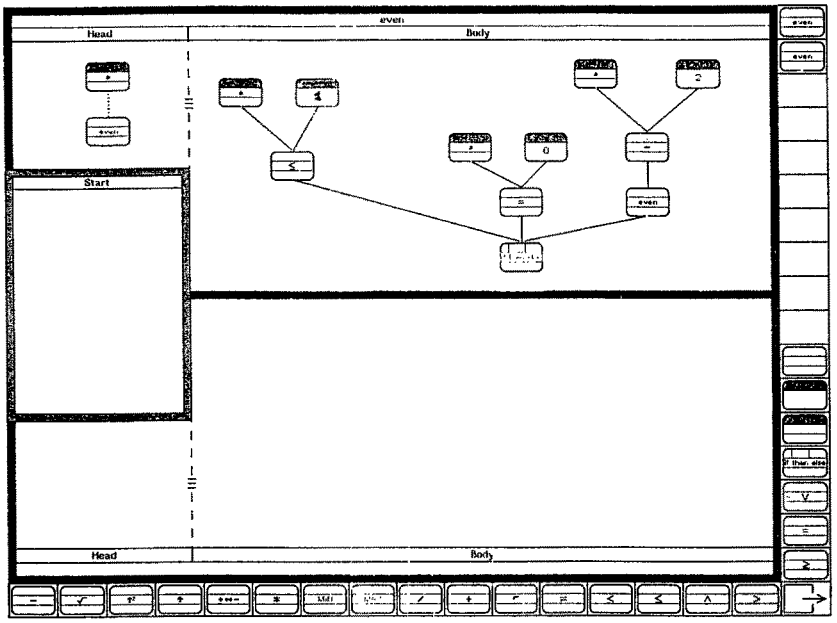

Figure 4: correct implementation of "even" predicate

5. Summary

From a theoretical point of view the optimal selection of the moment when help should be given does not seem to be a hard problem: IDL- and SDL-theory and our results provide strong evidence that problem solvers prefer to accept help information during impasses, whereas during the knowledge optimization phase new information is usually ignored.

Still an active research question in the design of ITSs and PSMs, though, is the content synchronization of help information and knowledge state of the problem solver. We tried to solve this problem by offering a mixed-initiative dialogue in case of an impasse: the student selects information, proposes a hypothesis and its outcome which is checked by the PSM.

References

[1] WENGER, E., Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge, Los Altos California: Morgan Kaufman Publishers, Inc., 1987

[2] POLSON, M.C. & RICHARDSON, J. (ed), Foundations of Intelligent Tutoring Systems, Hillsdale, N.J.: Lawrence Erlbaum Press, 1988

[3] PSOTKA, J., MASSEY, L.D. & MUTTER, S.A. (eds), Intelligent Tutoring Systems: Lessons Learned, Hillsdale, N.J.: Lawrence Erlbaum Press, 1988

[4] BIERMAN, D., BREUKER, J. & SANDBERG, J. (eds), Artificial Intelligence and Education, Amsterdam: IOS, 1989, ISBN 9051990146

[5] MAURER, H. (ed), Computer Assisted Learning, Lecture Notes in Computer Science, Berlin-Heidelberg-New York: Springer 1989

[6] KEARSLEY, G.P. (ed), Artificial Intelligence & Instruction, Reading, Mass.: Addison-Wesley 1987

[7] SCHMITT, H. & WOHLFARTH, P., Mathematikbuch 5N, München: Bayerischer Schulbuchverlag, 1978

[8] BAUER, F.L. & GOOS, G., Informatik: Eine einführende Übersicht, Erster Teil, Berlin: Springer 1982

[9] 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

[10] DOSCH, W., Principles of Teaching Programming Languages, in: E. SCERRI (ed), Proceedings of the 2nd Biennal Meeting of the Community of Mediterranean Universities, Malta, 17-21, October 1988 (in press)

[11] MÖBUS, C. & THOLE, H.-J. Tutors, Instructions and Helps. In: CHRISTALLER, Th. (ed): Künstliche Intelligenz KIFS 1987, Informatik-Fachberichte 202, Heidelberg, Springer 1989, S. 336-385

[12] MÖBUS, C. & SCHRÖDER, O., Knowledge Specification and Instructions for a Visual Computer Language. In: KLIX, F., STREITZ, N.A., WAERN, Y. & WANDKE, N. (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, S. 535-565

[13] JANKE, G., & KOHNERT, K., Interface Design of a Visual Programming Language: Evaluating Runnable Specifications. In: KLIX, F., STREITZ, N.A., WAERN, Y. & WANDKE, N. (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, S. 567-581

[14] 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)

[15] SHNEIDERMAN, B., Empirical Studies of Programmers: The Territory, Paths, and Destinations, 1-12 , in: E. SOLOWAY & S. IYENGAR (eds), Empirical Studies of Programmers, Norwood,N.J.: Ablex, 1986

[16] WEBER,G.,WALOSZEK,G. & WENDER, K.F., The Role of Episodic Memory in an Intelligent Tutoring System, in: J.SELF(ed), Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instruction, London: Chapman & Hall, 1989
[17] JANKE, G., MÖBUS, C., & THOLE, H.J., Empirische Pilotstudie zur Konstruktion eines problemlösezentrierten Hilfesystems für einen Problemlösemonitor, 44-55, in: F.STETTER & W.BRAUER (eds), Informatik und Schule 1989: Zukunfts-perspektiven der Informatik für Schule und Ausbildung, Informatik-Fachberichte Nr.220, Berlin-Heidelberg-New York: Springer 1989
[18] HOUGHTON, R.C., Online Help Systems: A Conspectus, Communications of the ACM, 1984,27,126-133
[19] SHNEIDERMAN, B., Designing the User Interface, Reading Mass., 1987
[20] McKENDREE, J., Feedback Content During Complex Skill Acquisition, in: G.SALVENDY, S.L.SAUTER & J.J.HURRELL (eds), Social, Ergonomic and Stress Aspects of Work with Computers, 181-188, Amsterdam: Elsevier Science Publ., 1987
[21] HARTLEY, J.R. & PILKINGTON, R., Software Tools for Supporting Learning in Intelligent On-Line Help Systems, in: P.ERCOLI & R.LEWIS (eds), Artificial Intelligence Tools in Education, 39-65, Amsterdam: North-Holland, 1988
[22] HARTLEY, J.R. & SMITH, M.J., Question Answering and Explanation Giving in Online Help Systems, in: J.SELF (ed), Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instruction, 338-360, London, 1988
[23] KEARSLEY, G., Online Help Systems: Design and Implementation, Norwood, N.J., 1988
[24] MOLL, Th. & FISCHBACHER, K., Über die Verbesserung der Benutzerunterstützung durch ein Online-Tutorial, in: S.MAASS & H.OBERQUELLE (Hrsgb), Software-Ergonomie '89, 223-232, Stuttgart 1989
[25] 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 (in press)
[26] van LEHN, K., Towards a Theory of Impasse-Driven Learning. In: MANDL, H., LESGOLD, A. (eds), Learning Issues for Intelligent Tutoring Systems, Springer, New York, 1988, S. 19-41
[27] 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, S. 57-84
[28] LEWIS, C., Composition of Productions, in KLAHR, LANGLEY & NECHES (eds), Production System Models of Learning and Development, 329-358, Cambridge, Mass.: MIT Press, 1987
[29] LAIRD, J.E., ROSENBLOOM, P.S. & NEWELL, A., Chunking in SOAR: The Anatomy of a General Learning Meachnism, Machine Learning, 1986, 1, 11-46
[30] WOLFF, J.G., Cognitive Development as Optimisation, in: L.BOLC (ed), Computational Models of Learning, 161-205, Berlin: Springer, 1987
[31] ANDERSON, J.R., A Theory of the Origins of Human Knowledge, Artificial Intelligence, 1989, 40, 313-351
[32] 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 (in press)
[33] KATZ, I.R. & ANDERSON, J.R., Debugging: An Analysis of Bug-Location Strategies, Human-Computer Interaction, 1987-1988,3,351-399
[34] JOHNSON, W.L., Intention-Based Diagnosis of Novice Programming Errors, Research Notes in Artificial Intelligence, London: Pitman, 1986
[35] ANDERSON, J.R.: Production Systems, Learning, and Tutoring, in: KLAHR, D., LANGLEY, P., NECHES, R., Production System Models of Learning and Development, 437-458, Cambridge,Mass.: 1987
[36] MURRAY, W.R., Automated Program Debugging for Intelligent Tutoring Systems, Research Notes in Artificial Intelligence, London: Pitman, 1988
[37] 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, S. 107-115
[38] NILSSON, N.J., Principles of Artificial Intelligence, Palo Alto, CA; Tioga Publishing Co., 1980
[39] ABRAMSON, H. & DAHL, V., Logic Grammars, New York: Springer 1989
[40] LEVI, G. & SIROVICH, F., Generalized And/Or-Graphs, Artificial Intelligence, 1976, 7, 243-259