Interactive Support for Planning Visual Programs in the Problem Solving Monitor ABSYNT[1] : Giving Feedback to User Hypotheses on the Language Level

Claus Möbus & Heinz-J. Thole[2]

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

1. Introduction

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), *PSM* (Problem Solving Monitors) or *ITS* (Intelligent Tutoring Systems) with regular conferences, research journals and textbooks [1,2,3,4,5]. The difference between ICAI/ITS and CAI/TS was pointed out by [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 and are implemented in languages as LISP and PROLOG. Developers of ICAI systems focus on problems of *knowledge representation*, *student misconceptions*, and *inferencing*. By and large, they have ignored instructional theory and past research findings in computer-based instruction."

This paper offers a contribution to ICAI. We try to demonstrate the improvement of ICAI by the development of an *interactive help system*, which checks hypotheses postulated by the user during the

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

[2] We want to thank Klaus D. Frank for mapping verbal problem solving episodes to the goals in the goals-means-relation (GMR) and Gabi Janke for teaching students ABSYNT. Polishing the interface was done by Gabi Janke and Klaus Kohnert. Klaus succeeded in interfacing IF-Prolog to Interlisp/Loops, so that the GMR could be used by the students.

problem solving process. The system is capable to recognize even incomplete proposals and contains the knowledge to generate complete solutions of the programming tasks. Thus the interactive help system adaptively supports the planning activities of the user. This is done by a goals-means-relation (GMR) which contains the domain-knowledge to *analyse* and *synthesize* ABSYNT-programs. At the present moment this knowledge is worked out for 37 tasks in our curriculum and is condensed into 462 rules. The complexity of the solution space is rather astonishing. The system is capable to recognize and generate several millions of solutions even if the height of ABSYNT-trees is restricted to five nodes.

It has been shown by our empirical research that especially novices develop rather unusal solutions if they use local repairs or patches in order to debug their programs. Our psychological philosophy is to stimulate *explorative learning* but guided by our help system. That is the PSM should first encourage the problem solver to program a solution of the problem even when the program is suboptimal. In a second step this first solution will be criticized and modified in a tutorial dialog according to efficiency and stylistic standards. So, replanning is stimulated later.

The GMR will be improved in the near future. Under development is a *rule-composition learning mechanism* [7] which will be useful in improving the speed of the help system and in identifying preferred solution schemes of users. Furthermore we plan to integrate a *student model* which acquires knowledge about the results of the hypothesis testing process. This knowledge will be used to select those programming proposals from solution space which contain problem solving schemes the user has used successful in the past.

Our results can easily be generalized to domains where means or actions can be represented by tree-like terms: *recursive action sequences* [8].

In the last part of the paper we will demonstrate that the GMR can be used to describe problem solving behavior of novice programmers. It is shown that it is possible to map problem solving means like verbalizations and programming actions into goals of the GMR. The *semantics* of these structured nodes and their *psychological " reality"* will be a research topic in the future because this could ease the development of explanation and dialog components of the PSM.

2. The Problem-Solving Monitor ABSYNT

A special variant of ITS are PSMs that are designed with respect to certain tasks the user should learn to solve. They provide the learner with a problem-solving environment including helps but no curricular component. ABSYNT belongs to this category. Its task domain is *functional programming* comparable to pure LISP without the list-data structure. ABSYNT is a *visual tree-like* programming language (ABstract SYNtax Trees) based on ideas published in german school [9] and university text books [10]. Further motivation for the design of ABSYNT is given in [11], [12].

Basic research dealing with the design of the system from a *psychological* point of view is described in [13] - [17]. Figure 5 shows the interface of the programming environment when a student has programmed a wrong "solution" of the problem *even*.

3. The Design of Helps in ABSYNT as a Twofold Synchronization Problem

The psychological efficiency of a PSM depends to a great extent on the quality of instructions and helps built into the system [13]. To put it short: "When are helps useful and when are they distracting or inhibiting." The answer certainly dependends on the knowledge state of the problem solver and the state of the problem solving process. Both *content* and *application time* of the information have to be chosen carefully. Thus the design of helps is a paradigmatical research topic of cognitive and computer science [18] - [24].

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

A necessary prerequisite of programming is some knowledge about the syntax and semantics of the language. In the first period of our project we concentrated on the acquisition of semantic knowledge. The semantics of programming languages can be defined in three ways [25]: (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 other approaches. The behavior of the ABSYNT interpreter computing ABSYNT programs was represented by two-dimensional visual rules which served as instruction and help material for ABSYNT users [16].

In a study of the instruction-based knowledge acquisition process [26] we found that the acquisition of semantic knowledge could be described by a two-stage process:

1) *Knowledge enlargement* through impasse-driven learning (IDL) [27]

2) *Knowledge optimization* through success-driven learning(SDL) [28] - [31].

According to IDL- and SDL-theory and our results we have strong evidence that problem solvers prefer to accept help information in problem solving situations where an *impasse* occurs. During the knowledge optimization phase new information is usually ignored.

3.2 The Design of Helps to Acquire Planning Knowledge when Programming in ABSYNT

Even more important for the programming novice is a help system which embodies *planning* knowledge. Here too, we face the twofold synchronization problem : *content and application time* of feedback information <---> *knowledge state* of the student.

The present status of the help system implemented so far is a consequence of some postulates. The help system should:

- *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 [27]
- *gather* user *data* online to adapt the user model continuously
- embody expert knowledge to *check* user proposals and *generate* helps or solutions
- *deliver* only *minimal information* so that the student is able to leave the impasse situation by improving his problem solving skills
- *offer the environment* to check various hypothesis about the usefulness of several parts of the program

The last point is rather important. Contrary to some authors (e.g.[32]) we think that semantic errors often can *not* be localized to a line. Most times the proposal of the user as a whole is inconsistent with the problem due to its goal structure. Repairs should depend on those parts of the program which the user wants to retain. 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 [32] - [37].

Our answer to the above described postulates is a help system based on the GMR. This relation can be looked at as a *rule-based inference system* [38], a *grammar* [39], [40] or an *AND/OR-Graph* [41] with structured nodes.

A small excerpt of the AND/OR graph for the goal *even(Subgoal)* is shown in figure 1. The square nodes contain goals which correspond to ABSYNT operators or operands. The round shaped nodes are parametrized goals or schemas which have to be further elaborated in the programming process. Because nodes of the AND/OR graph can be parametrized for subgoals, the relation enables *analysis* and *synthesis* of partial and total solutions.

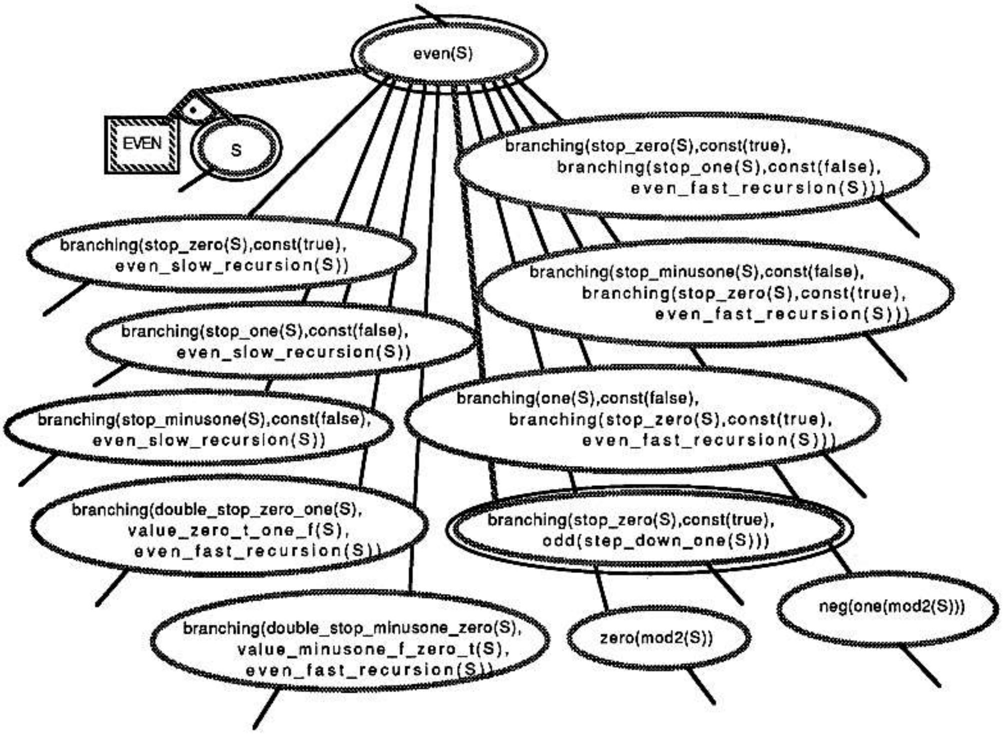


Figure 1

It is possible to derive rules from the graph. A rule consists of a source node (upper node in the graph), sink node(s) and the connecting link(s). For demonstration purposes we marked two subgraphs in figure 1. The corresponding rules are shown in "animation style" in figures 2a and 3.

Rule: "Planning an Abstraction on the Language Level"

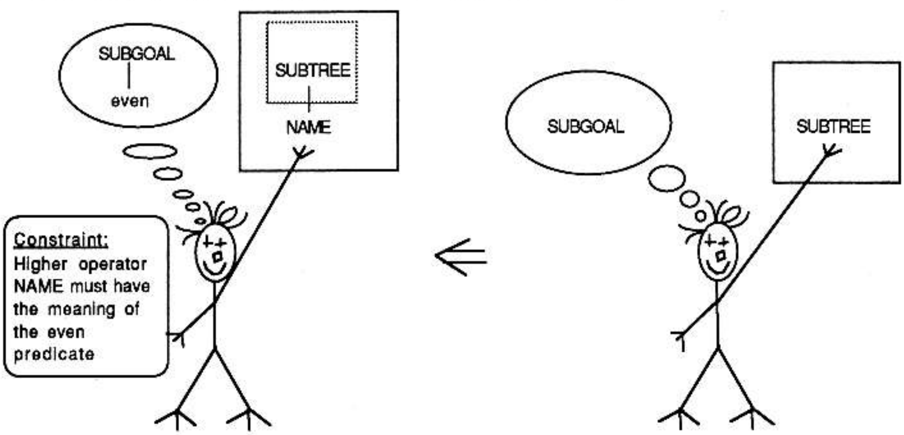


Figure 2a

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 steps:

- *select* for the root of the ABSYNT tree a higher operator with an optional NAME (e.g. foo) which must possess the meaning (semantics) of the ***even predicate***, either as an already programmed function or as an yet to be programmed function
- *leave space* in the worksheet of the ABSYNT environment for the yet to be programmed subtree

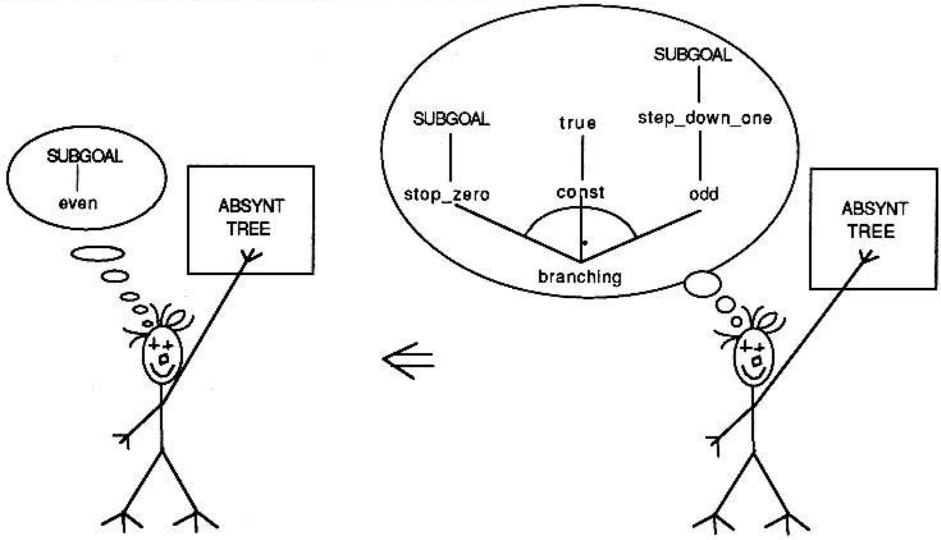
AND

IF your next planning step is to program ***the subgoal***

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

Figure 2b

Rule: "Planning an Abstraction 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

This approach is different from other systems with similar aims [32] - [37]. Due to its *flexibility* it seems to promise some positive consequences for the motivation of the problem solver and as will be described below, for the acquisition of problem solving skills.

Our work rests on the *impasse-driven learning theory* (IDLT) [27], [30], [42]. When the student programs a proposal, which is diagnosed by the ITS as wrong, s/he is trapped in an *impasse*. According to IDLT s/he is now *sensitive* to acquire help information. This should be assimilated in an active act of problem solving. So s/he has to *propose an hypothesis* about the usefulness of parts of her/his program. The *feedback* of the system to this hypothesis can then be regarded as *help* information which can be used by the student to circumvent his/her impasse.

Errors in functional programs are often difficult to localize. This is true for most nonsyntactic bugs. Often the only possible diagnosis is : The goals various parts of the program compute are inconsistent with the main goal. In figure 4a we have the impasse situation of a student. It is an inconsistent implementation of the "even" goal. There are several possibilities to localize bugs and to repair this program. The programmer's *knowledge and beliefs* can be used by our help system.

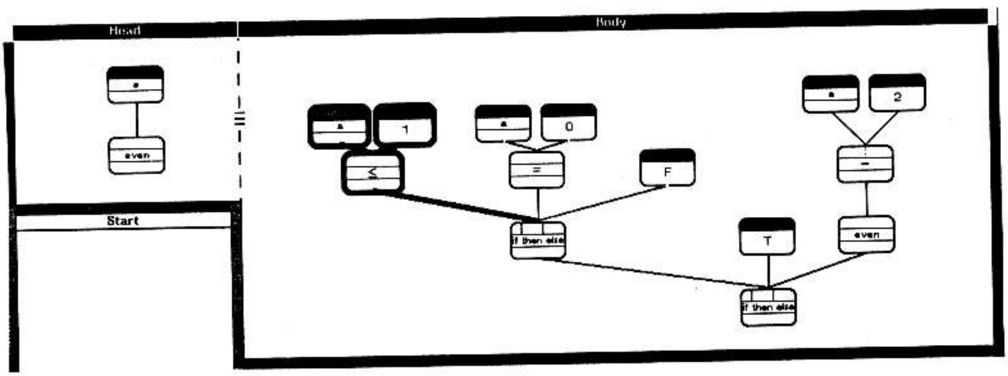


Figure 4a

The programmer has to put forward positive or negative hypotheses like: "I presume that this marked subtree of the program can be embedded in a correct solution!" or "I suspect that this marked subtree can *not* be embedded in a correct solution!"

S/he then has to mark this hypothesis with the mouse (**bold lines** in figure 4a). This corresponds to the hypothesis: "Is it possible to embed this marked part of my proposal in a correct solution?"

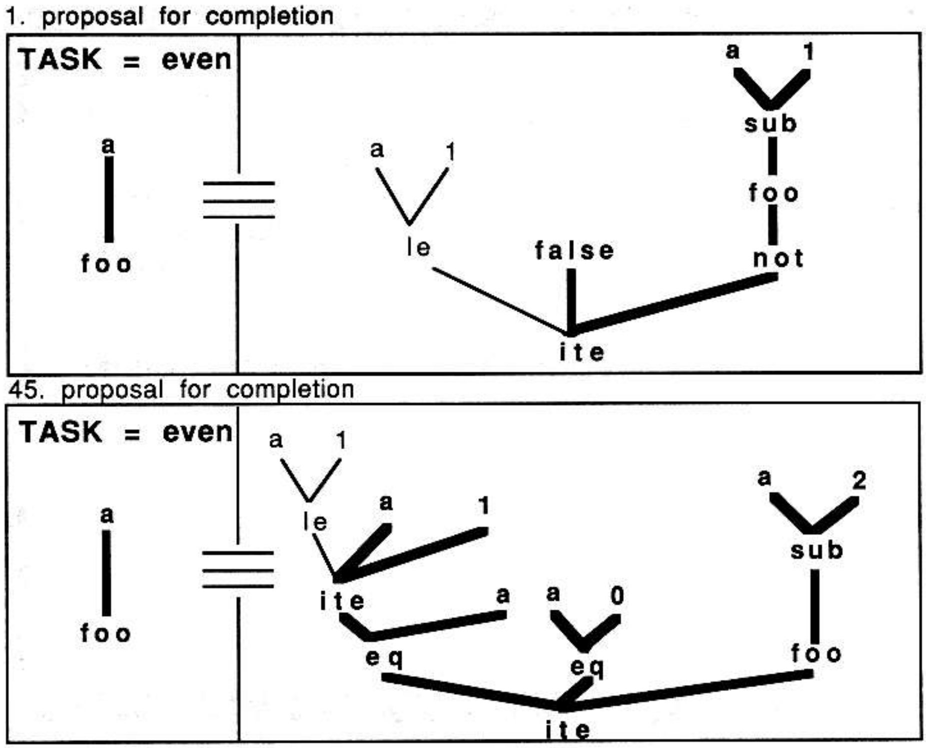


Figure 4b

The system is able to generate complete solutions *constrained* to this hypothesis. In figure 4b we see the 1st and the 45th synthesized solution of the problem. To avoid passiveness of the problem solver and to restrict the number of proposals the programmer actually sees only small parts of the complete feedback.

The first answer to the hypothesis is "Yes/No". If this does not resolve the difficulty, the student is given more information on demand. S/he is asked to choose one of the nodes of the solution which has a link with an embeddable hypothesis.

4. A Session with the Hypothesis-Driven Help System

To demonstrate the system we choose the impasse situation of figure 4a. Local patches and repairs led to this program. The student knows from earlier steps of the hypothesis testing sequence that the *complete* proposal is incorrect. The student suspects that the predicate is in error. He believes that the THEN and ELSE-branch are correct. S/He marks her/his hypothesis according to figure 5. S/He receives the message : "No: Your hypothesis cannot be completed to a solution known by the system".

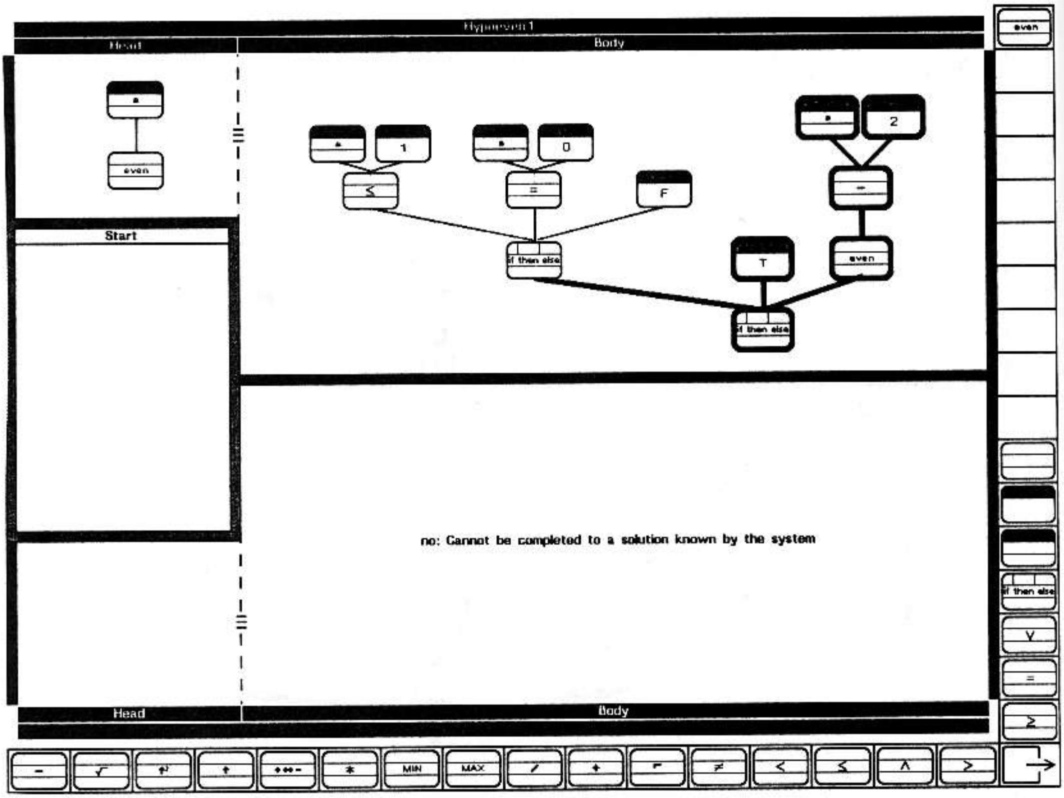


Figure 5

Because the student is quite certain that the recursion step in the ELSE-branch is correct s/he restricts her/his hypothesis even further (Figure 6, upper window). Now, s/he sees the copy of the hypothesis in the feedback window as a positive response (Figure 6, lower window).

The student knows that *dependent* on this hypothesis the error/errors are either in the predicate or in the THEN-branch of the IF-THEN-ELSE operator. S/He can start to repair the program in the workbench window (upper window). If s/he is still in an impasse, s/he can ask for further information. The hypothesis contains two *open* links. The student can mark one of these links. S/he chooses the THEN-branch. The feedback of the system is: the main operator of the THEN-branch could be the EQUAL-Operator (Figure 7).

The student hopes that this was the only error. To affirm this belief it is up to her/him to propose further hypotheses (e.g. Figure 4a). Now the student has some information to replan the program.

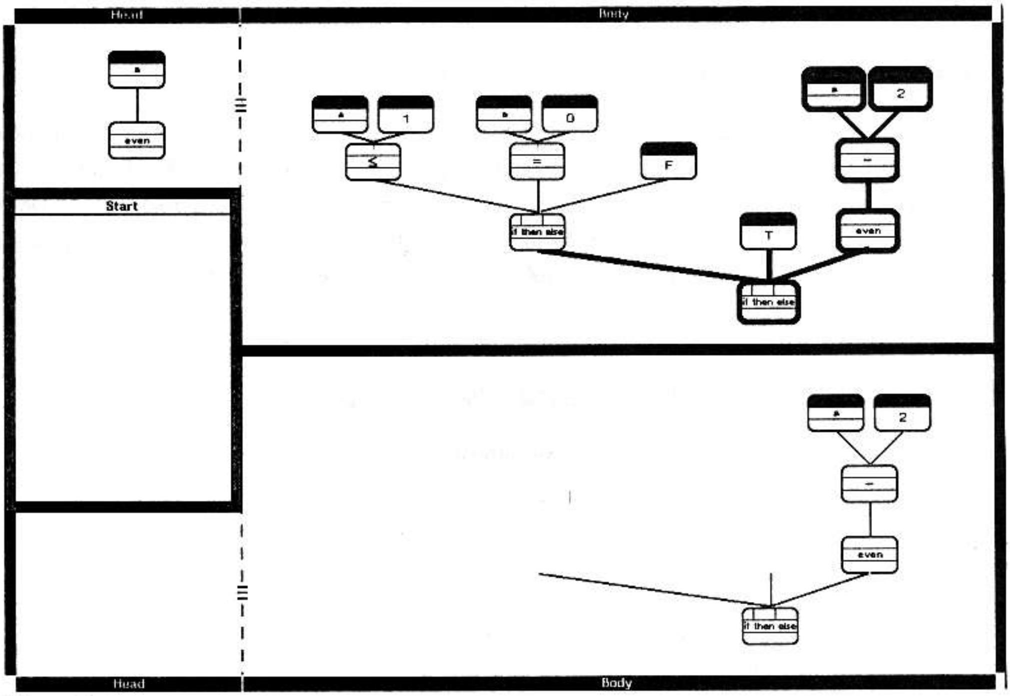


Figure 6

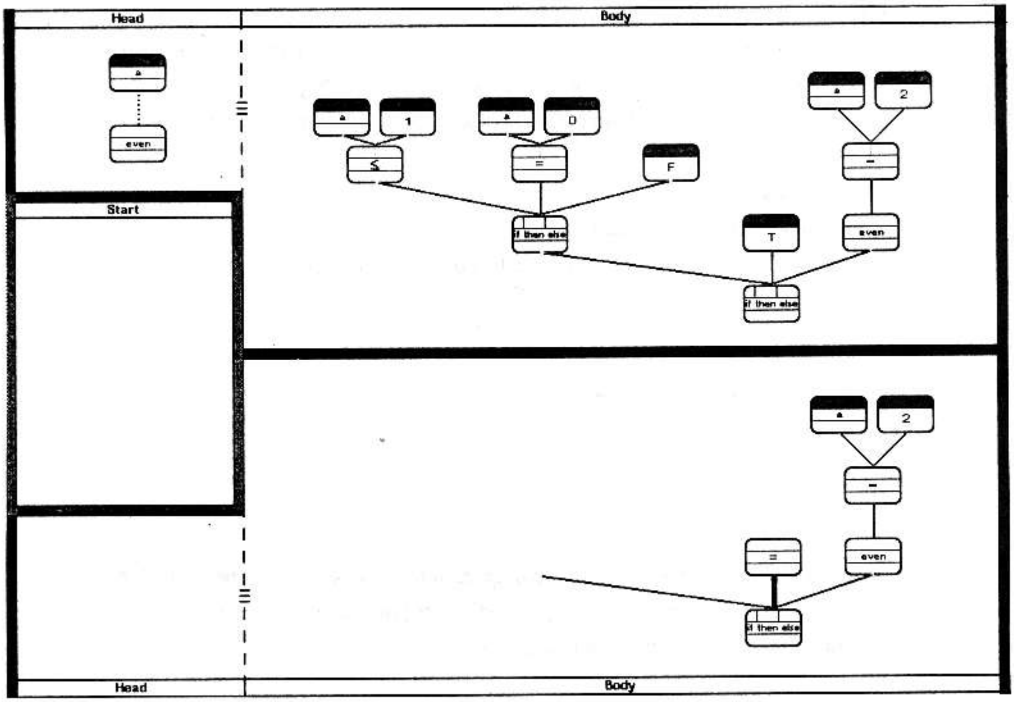


Figure 7

5. Further Research Topics

5.1 The Psychological "Reality" of the Goals in the GMR

We are interested in the question whether the goals in the GMR are *pure symbols* or bear some *psychological reality* [43]. In the latter case it would be possible to interpret the AND/OR graph of the GMR as a problem space [44]. The problem solving process can be viewed as a path through the AND/OR graph. This assumption is not unreasonable [45] but has to be further investigated. In figure 8 we have a small excerpt of a problem solving protocol of a subject programming the "absdiff" problem.

<u>Beginning part of a verbal protocol for the "absdiff" program:</u>

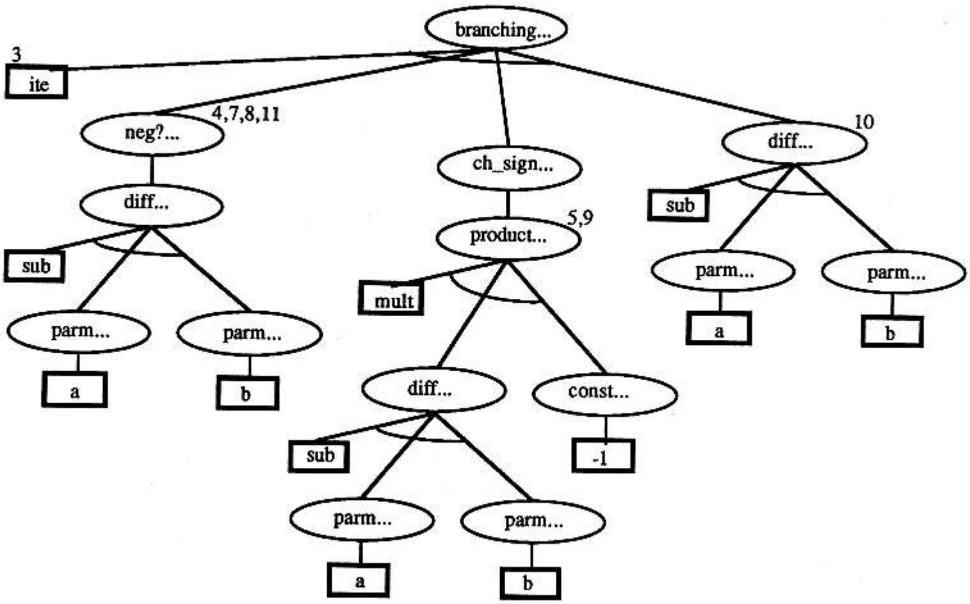
"Construct an ABSYNT program which returns the difference between two numbers as a positive value"

Segment Number	Protocol
1	Ok, first we put the parameters, with the - values
2	then a minus-operator node --
3	and now -- ah, now we take the if-then-else node.
4	Then, if the result is negative,
5	then you can multiply by minus 1, - I am not sure if we can do that with this node, but
6	and when the result is positive, then - you just say minus -
7	hm - now we only have to decide, how does he decide between minus and plus --
8	if it is negative,
9	then we have times minus 1,
10	and else - nothing, just take the result
11	- how can we reach a decision here - how can he do that, with minus --

The screen remains empty during this sequence.

Figure 8

The protocol is partitioned into episodes which are mapped into the AND/OR graph of the GMR (figure 9). If this mapping could be done automatically as proposed by [46] the quality of helps could be improved due to information about the intentions of the problem solver.



Screen is empty.

Figure 9

5.2 Rule Composition and Solution Schemas

As can be seen from figures 2 and 3 the planning rules are highly standardized. It is easy to use the *learning mechanism* of automatic rule composition [7] to speed up the system and diagnose typical problem solving schemes in only a few steps.

5.3 Individualized Helps and Student Models

To restrict further 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 hypothesis, their results and the corresponding actions and repairs of the problem solver.

5.4 Further development of the Planning Helps

The present implementation of the help system shows feedback only on the *language level*. In the near future these "low level" helps will be accompanied by "high level" helps on the *goal level*.

6. References

[1] WENGER, E., Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge, Los Altos Calfornia: 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, Berlin: Springer, 1989

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

[7] LEWIS, C., Composition of Productions, in: D.KLAHR, P.LANGLEY & R.NECHES (eds), Production System Models of Learning and Development, 329-358, Cambridge, Mass.: MIT Press, 1987

[8] WYSOTZKI, F., Representation and Induction of Infinite Concepts and Recursive Action Sequences, Proceedings of the 8th International Joint Conference on Artificial Intelligence, 1983, Karlsruhe, Palo Alto,Ca.: Morgan Kaufman Publisher

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

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

[11] 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

[12] 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)

[13] 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

[14] 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

[15] 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

[16] 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)

[17] JANKE, G, MÖBUS, C & THOLE, H-J, Empirische Pilotstudie zur Konstruktion eines problemlösezentrierten Hilfesystems für einen Problemlösemonitor, in: STETTER, F. (ed), Informatik und Schule, Informatik-Fachberichte, Berlin: Springer (in press)

[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, Ergonmic 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] PAGAN, F.G., Formal Specification of Programming Languages, Englewood Cliffs,N.J.: Prentice Hall, 1981

[26] 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)

[27] 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;

[28] 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;

[29] LEWIS, C., Composition of Productions, in KLAHR, LANGLEY & NECHES (eds), Production System Models of Learning and Development, 329-358, Cambridge, Mass.: MIT Press, 1987

[30] LAIRD, J.E., ROSENBLOOM, P.S. & NEWELL, A., Chunking in SOAR: The Anatomy of a General Learning Meachnism, Machine Learning, 1986, 1,11-46;

[31] WOLFF, J.G., Cognitive Development as Optimisation, in: L.BOLC (ed), Computational Models of Learning, 161-205, Berlin: Springer, 1987

[32] KATZ, I.R. & ANDERSON, J.R., Debugging: An Analysis of Bug-Location Strategies, Human-Computer Interaction, 1987-1988,3,351-399

[33] JOHNSON, W.L., Intention-Based Diagnosis of Novice Programming Errors, Research Notes in Artificial Intelligence, London: Pitman, 1986

[34] 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

[35] MURRAY, W.R., Automated Program Debugging for Intelligent Tutoring Systems, Research Notes in Artificial Intelligence, London: Pitman, 1988

[36] 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

[37] WEBER, G., WALOSZEK, G. & WENDER, K.F., The Role of Episodic Memory in an Intelligent Tutoring System, in: SELF, J. (ed), Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instruction, London: Chapman & Hall, 1989

[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] SABATIER, P., Quantifier Hierarchy in a Semantic Representation of Natural Language Sentences, in: V.DAHL & P.SAINT-DIZIER (eds), Natural Language Understanding and Logic Programming, North Holland, 1985 (op.cit. in [39])

[41] LEVI, G. & SIROVICH, F., Generalized And/Or-Graphs, Artificial Intelligence, 1976, 7, 243-259

[42] LAIRD, J.E. NEWELL, A. & ROSENBLOOM, P.S., SOAR: An Architecture for General Intelligence, Artificial Intelligence, 33, 1987, 1-64

[43] HARNAD, St., Minds, Machines and Searle, Journal of Experimental and Theoretical Artificial Intelligence, 1989,1, 5 - 25

[44] NEWELL, A. & SIMON, H., Human Problem Solving, Englewood Cliffs, N.J.: Prentice Hall 1972

[45] ERICSSON, K.A. & SIMON, H.A., Protocol Analysis: Verbal Reports as Data, Cambridge, Mass.: MIT Press, 1984

[46] WATERMAN, D.A. & NEWELL, A., Protocol Analysis as a Task for Artificial Intelligence, Artificial Intelligence, 1971, 2, 285-318

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

438

D.H. Norrie H.-W. Six (Eds.)

Computer Assisted Learning

3rd International Conference, ICCAL '90
Hagen, FRG, June 11–13, 1990
Proceedings

Springer-Verlag

New York Berlin Heidelberg London Paris Tokyo Hong Kong

Editors

Douglas H. Norrie
The University of Calgary
2500 University Drive N.W.
Calgary, Alberta T2N 1N4, Canada

Hans-Werner Six
Fachbereich Mathematik und Informatik
FernUniversität Hagen
Feithstraße 140, D-5800 Hagen, FRG

CR Subject Classification (1987): K.3.1, I.2

ISBN 3-540-52699-4 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-52699-4 Springer-Verlag New York Berlin Heidelberg

Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
2145/3140-543210 – Printed on acid-free paper