J.King Saud Univ., Vol.8, Computer & Information Sci., pp. 79 - 110 (A.H 1416/1996)

Experiment in Knowledge Based Programing

Pavol Navrat and Viera Rozinajova

Computer Science and Engineering Dept., Slovak Technical University
Ilkovicova 3, 812 19 Bratislava, Slovakia

(Received, 06 Sept. 1993; accepted for publication, 21 Sept. 1994)

Abstract. The paper investigates possibilities of writing programs with having the relevant knowledge on programing available in explicity form. in order to perform experiments, a knowledge base was built which codes some of the knowledge related to the problem of selecting a proper data type in the process of program formation. The base is presented in paper along with one experiment which also shows the system performance and user-system interaction. The experiment setting is such that the user makes a guess which data type is appropriate to use and this hypothesis is either confirmed or rejected by the system. Moreover, as a result of the system's deductive reasoning other acceptable data types are proposed by the system. The result shows that the system is able to provide an advice to a programer. This can be particularly useful in the process of learning programing.

Introduction

During the last dozen or so years, there has been reported a growing number of research work in the area of knowledge based programing. Knowledge base programing recognizes the importance of explicity represented knowledge on programing as well as on the particular problem domain for a successful accomplishment of the programing task. It seeks methods of using such knowledge in the program formation process.

A question might be raised, what is the relationship of such knowledge base to a proper formal theory of programing. Indeed, it is fully justified to expect from a theory of programing, that it would provide a method for specification of programs, a method for reasoning about specifications, a method of developing programs from their specifications, and a method of transforming programs to increase efficiency. There has been much work in developing such theory [6; 29; 26]. Despite the continuing effort, however, there is not such formal theory at hand which would provide the expected methods usable on such industrial scale which is typical to today's software production.

Knowledge based programing attempts to provide such methods, too. For normalization of the programing know-how, knowledge engineering (which belongs to the field of artificial intelligence) techniques are borrowed. Instead of axiomatisation directly in the first order predicate logic, or some its suitable variant, formulation of the programing expertise in some knowledge representation language is attempted. here, either the "standard" formalisms, like production rules, frames, semantic networks, or some especially tailored languages designed for the given purpose are employed. In either case, the formalisms provide for more or less convenient representation of various kinds of knowledge (e.g., properties, relations and actions) and their various properties and relations (e.g., property inheritance). While it should be clear, that such knowledge engineering approach cannot be considered a full substitute for a proper development of a theory, at the same time it reflects the fact that the very nature of the program formation has strong similarities to other construction or design-type processes [5].

There is also another reason for making use of artificial intelligence techniques in devising more systematic and efficient methods of development of efficient programs. Constructing a program from a given specification of a problem can be considered as a special case of problem solving process. how to solve problems is a central topic of study in artificial intelligence.

There are several open problems in knowledge based programming. Creating a sufficiently complete knowledge base, which was tackled as early as in 1978 by [9] remains the constant problem e.g., [27; 1]. An architecture of the system which uses the knowledge base must be devised, e.g. [11; 25; 24]. Specifically, its reasoning method and knowledge base structure must be defined e.g., [7; 22]. The underlying software development methodology is also important e.g., [3; 12]. We have participated also in several related works e.g., [15; 18]. In [16], an original

presentation of kinds of knowledge involved in program formation was presented. In [20], an overview of the development of the field is given.

As can be seen from the above list, the area is too broad to be tackled in a single research endeavor. We have been interested in building a base of programing knowledge which would serve as a source of knowledge for the beginning students of programing. The representation should be explicit, thus:

- facilitating the study of the respective pieces of knowledge in a human readable form, and.
- allowing for appropriate machine manipulation, including reasoning to infer solutio-
  ns to partial problems occurring during the programing process.

We have built an experimental knowledge base on programing. We have conducted several experiments, some of which are described elsewhere [21]. Here, we shall concentrate on a short description of our method and on still another experiment. The work is reported in [23].

Selecting a Data Type During Program Formation

Next, we shall briefly describe our approach and at least one of the experiments not presented elsewhere. First, the goal should be formulated. We want to investigate possibilities of writing know-how used by a programmer when developing a program. Our concern is the scope of knowledge necessary for rather introductory level of programing. We leave out from our immediate focus many important software development issues. Also, we do not attempt to write all kinds of knowledge used in program development. We have attempted to present an original taxonomy in [16]. We are especially aware of the importance of the domain knowledge. One of the reasons for our concentration on programing knowledge is our interest in a possible role that such externalization's of the fundamentals of the programing craft could play in learning programing. This aspect was extensively studied e.g., [2; 27].

One of the important topics on this level is certainly the problem of selecting appropriate data type for representing data in a program being developed. The selection is based on information derived from the problem specification. How to select is part of the programing know-how. In this work, we should like to make explicit precisely this knowledge. Our aim is to build an experimental knowledge base that would allow

experiments in various problem solving situations occurring during program formation phase when a decision on how to represent relevant data is made.

For creating knowledge bases, there are basically two approaches applicable. Either an especially devised knowledge representation language supported by especially tailored knowledge base development tool, or a general tool supporting some form of production rules, frames, semantic networks etc. is used. For the former approach, we can list among many results published also ours e.g., [17; 18, 19; 8]. In the present work, however, we have used a general commercially available tool.

The choice of a way of representing knowledge depends on properties of the particular kind of knowledge. An almost obvious choice in many cases of building knowledge bases is to use rules. Indeed, rules are very useful way of representing many kinds of knowledge. On the other hand, we should like to stress that pure rules impose a low level of structuring of knowledge. Next, there are important relations among the knowledge pieces which cannot be conveniently represented by pure rules. More concretely, in the domain of programing knowledge, an important role seems to be played by knowledge related to so called stereotypical situations. There are more or less "standard" situations in the program development process, such as e.g., programming input of a sequential field, processing a sequential file, programming a loop etc. here, the programing knowledge involved is partially declarative, partially procedural and hence requires more complex forms of representation than pure rules. According to [13], "program frames" are a suitable way of organizing the programing knowledge.

The basic outline for our experiments was to find our how far can explicitly recorded knowledge written in a suitably devised form support a programmer who is a noise. We were also interested how far can such knowledge base act as a model of the domain. A possible scenario is supporting a student in acquiring programing skills. To evaluate results of the experiments, we relied on our teaching experience in this area at both the Kuwait and Slovak Technical Universities. Of course, we can speak only about very intuitive evaluation in this regard. However, we have tried to devise experiments which require nontrivial decisions.

The relevant part of programing knowledge concerns various structured data types, both static and dynamic. Each data type has some properties which serve as clues for selecting them for certain class of applications. We have represented the system of knowledge pieces which mirrors the system of data types as system of classes and subclasses. Other classes of objects are problem specification or problem solving

methods. Specific knowledge pieces of programing know-how can conveniently be represented as rules. We use rules such as (in English paraphrase).

Rule 2 :

If 1. Use of dynamic data structure is recommended.
2. The problem requires explicity representation of binary relation.
3. The multiset of elements is set of elements.

Then Use of graph is recommended.

Our set of rules comprises a number of rules (several dozens in the current stage of development) which describe different conditions that suggest use of the respective data type. The conditions refer to properties of data as given in the problem specification, e.g.,

Rule 12:

If 1. Use of dynamic data structure is recommended
2. Ordering upon set of elements is defined
3. This ordering is referred to in problem specification
4. Multiset of elements is set of elements

Then Use of search tree is recommended.

or to properties of the expected processing of data, e.g.,

Rule 14 :

If 1. Use of dynamic data structure is recommended
2. Ordering upon set off elements is defined
3. Frequency of operations remove element and insert element is high
4. Multiset of elements is set of elements

Then Use of search tree is recommended

or to properties of the problem solving method, e.g.,

Rule 21 :

If 1. Use of structured data type is recommended
2. Problem solving method presupposes organizing multiset of elements as matrix
3. All elements are of the same type
4. Matrix is not a sparse matrix

Then Use of two-dimensional array is recommended

We have built an experimental base of programing knowledge. We have taken the option of using a general commercially available development tool. The choice was based partially on analysis of properties of tools offered on the market, but we have to admit that it was also partially constrained by the availability to us. We have

chosen Nexpert Object tool. It provides several way of representing knowledge, not only rules. The domain can be described also by classes and objects. Objects represent in fact declarative knowledge that rules refer to. There can be define hierarchical relations among objects.

The tool has so called event-driven architecture. A rule or a hypothesis can become relevant because some external event justifies its evaluation.

Objects are organized in classes. As a consequence, they are capable of inheriting properties of values. Moreover, there can be associated with any property so called demon which stands for an action to be triggered if the related property is about to be changed.
The set of rules from Appendix A is represented in NEXPERT in its working format as in Appendix B.

Experiments

There are two typical situations in the regard of the data type selection (meta-) problem, being a part of the problem solving process aiming to form a program. In the first kind of situations, the user does not know what data type he/she (from now on, she) should use. Therefore she just inputs the problem specification (more precisely, the relevant part of it). it is the task of the system to infer appropriate suggestions. in such case, the system starts a forward reasoning process. It may ask for additional information from the user when it cannot be inferred from the provided data. In the second kind of situations, the user knows something about how to select a data type and she is able to make an initial suggestion. She expects the system to either confirm or reject her hypothesis. In case it rejects her hypothesis in the subsequent backward reasoning process, the system is able to provide its suggestions(s).

Experiments related to the former kind of situations aimed to find out how such knowledge base can serve as a model for decisions made by the programer during the program formation process. The system has similar information available as would a programer have at this stage. The question is, whether the system can produce also similar solutions to those found by a programer. for the latter kind of situations, the experiments aimed to investigate to which extent can such knowledge base support a programmer who is able to propose a solution to a particular programing (meta-) problem i.e. how to implement representation of some structured data collection processed during the process of solving the given problem. The system receives the

proposed solution i.e. a hypothesis. It attempts either to confirm or to reject it. During this process it may make esquires to user asking her to give additional informations about the problem.

From among the former kind of experiments, we experimented with such problems as selecting data type for solving a system of linear equations or for the "rainfall problem" (cf. [27]). The experiments are described in [21]. from among the latter kind of experiments, we experimented for example with conforming/rejecting a hypothesis of selecting a search tree. We like to present this experiment here.

The user wishes to use for given problem data type search tree and requires from the system confirmation or refutation of her hypothesis.
Initially, following hypothesis was set (cf., Appendix C):
-> Use of search tree is recommended

System starts deductive reasoning; the first rule to be processed is rule 17 :
Q : "In input multiset of elements?"
A : YES
Q : "While processing multiset: will elements be processed more than once ?"
A : YES

Rule 17 is set to true:
-> * Use of structured data type is recommended.*

Ruel 11:
The first condition - the use of structured data type is recommended (from now on, SDT) is true.
Q : "Is the ordering upon the set of elements defined?"
A : YES
Q : "Will the frequency of operations remove and insert element be high ?"
A : YES
Q : "Is the multiset of elements set of elements?"
A : YES
Rule 11 is set to true.
-> * Use of search tree is recommended*

Rule 14:
The first condition - SDT is true.

The second condition is satisfied from previous question already, so proceeds to the third condition, where the system needs to ask:
Q : "Is ordering upon a set of elements referred to in program specification?"
A : YES
The fourth condition - multiset of elements is set of elements (i.e. the value of each element of set varies from all other elements of this set) is also satisfied, so:

Rule 14 is set to true, which is confirmation of previously announced partial result:
-> * Use of search tree is recommended*

Rule 1:
The first condition - SDT is true.
Q : "Can the cardinality of multiset be estimated in advance ?"
A : NO
Rule 1 is set to true:
-> * Use of dynamic data structure is recommended*

Rule 13:
The first condition - SDT is true.
The next 3 conditions are evaluated from previous evaluation, so again:
-> * Use of search tree is recommended*
The same result will be achieved from rule 12.

Rule 21:
The first condition - SDT is true.
Q : "Does the problem solving method presuppose organizing multiset of elements as matrix?"
A : NO
Ruel 21 is set to false.

Rule 22:
The first condition - SDT is true.
Q : "Does the specification presuppose organizing multiset of elements as matrix?"
A : NO
Rule 22 is set to false.

Rule 23:
The first condition-SDT is true.
Q : "Are all elements of the multiset of same type ?"

A : YES
Q : "Does the mapping from the pair of ordinals to the set of element exist?"
A : NO
Rule 23 is set to false.
Coming out from the previous three rules:
-> * Use of two dimensional array is not recommended*

Rule 18:
The first condition-SDT is true.
Q : "Does the accessing of element depend on the contents of the element?"
A : YES
Rule 18 is set to true:
-> * Use of table is recommended*

Rule 4 :
The first condition has just been satisfied by setting up previous rule to true (use of table).
Q : "Should access time be not dependent on the size of the set of elements?"
A : YES (That means the access time should be not dependent on the set size).
Rule 4 is set to true:
-> * Use of hash table is recommended

Rule 19:
The first condition has just been satisfied by setting up previous rule to true (use of table).
The second condition is also satisfied from previous reasoning;
Q : "Is the optimization of an operation of accessing the element required ?"
A : YES
Rule 19 is set to true:
-> * Use of table with ordered elements is recommended*

Rule 15:
the first condition-SDT is true.
Q : "Will LIFO strategy be employed during the processing of elements?"
A : NO
Rules 15 and 16 are set to false:
-> * Use of stack is not recommended

Rule 9 :
The first condition-SDT is true.
Q : "Will FIFO strategy be employed during the processing of elements?"
A : NO
Rules 9 and 10 are set to false:
-> * Use of queue is not recommended

Rule 8 :
The first condition-SDT is true.
Q : "Does the mapping from the set of ordinals to the set of elements exist?"
A : YES
The third condition is satisfied already.
Q : "Can the order of processing of elements be decided in advance?"
A : YES
Rule 8 is set to true:
-> * Use of one dimensional array is recommended*

Rule 7 :
As one of its conditions (the ordering upon the set of elements is defined) is explicitly not satisfied:
Rule 7 is set to false.

Rule 5 and 6:
As one of their conditions (the order of processing is sequential most of the time) is explicitly not satisfied:
Rule 5 and 6 are set to false:
-> * Use of linear list is not recommended*

Rule 2:
The first condition - SDT is true.
Q : "Does the problem require the explicity representation of binary relation?"
A : YES
The third condition is satisfied, so:
Rule is set to true:
-> * Use of graph is recommended*

Rule 3 :
The conditions are similar except of using dynamic data type instead of structured data type; all are satisfied, so:

Rule 3 is set to true, and the hypothesis is same as the one rule 2.

Rule 20:
The first condition is the use of graph - this is satisfied.
Q : "For the relation predecessor the inverse relation is one-one?"
A : YES
Q : "Is there only one element, for which the relation predecessor is not defined?"
A : YES
Rule 20 is set to true:
-> * Use of tree is recommended*

The above gives the user-system interaction commented with regard to advancement of the reasoning process. The trace of this process is recorded in Appendix D. List of all established and rejected hypotheses is in Appendix E.

The system has performed a deductive reasoning. it made several esquires concerning the problem that the user attempts to solve and intends to use search tree. Based on the information supplied from the user, it is able to conclude (from its knowledge base on data types) that the choice of search tree is appropriate. it makes, however, other recommendations, too. We see that it recommends also graph, table, has table, table with sorted elements and one dimensional array. This is in accord with the programming practice where it happens frequently that one can have multiple choice. The choices can either be equally appropriate for the given problem or the decision may require finer analysis of problem properties and more programing knowledge.

Results and discussion

The knowledge base presented is quite modest. At the same time, it allows nontrivial reasoning sequences to be modeled. The experiment demonstrates that the programing knowledge can successfully be represented even when using a tool which is not dedicated to this particular application. On the other hand, there are limits of such approach because especially tailored formalisms and tools would be desirable to allow more complicated modes of reasoning, involving different layers of knowledge cf., [22]. Still, the approach allows extensions to other areas of programming expertise besides the problem of selecting the proper data type, and even to other areas of problem solving. The availability of extending is obviously facilitated by the use of a general knowledge engineering tool. However, we find it even more important to make a careful design of the knowledge base. In the end, its contents will be decisive.

Important contributions to investigation of programing knowledge were published by Anderson et al. e.g., [1; 2] and Soloway et al. e.g., [10; 27].

Our point of view has been problem solving aspects rather than more programing language related aspects. We do not even mention explicitly a programing language, differing sharply in such way not only to syntactically oriented approaches, but differing in some sense also to more semantically based approaches such as Pascal Tutor of [4]. We do not, however, attempt to design a complete learning environment. There is also a point of view presented by e.g., [30], who finds building models (in form of knowledge bases) useful in clarifying and organising student's ideas about the subject matter. This presupposes that the student studies the particular topic prior to attempting to build any model. We feel it is worth considering the knowledge bases as sources from which certain matters can be studied. For example, such externalization's of knowledge can be extremely important in areas where no mature scientific formalized theory could have been developed so far.

The approach presented in this paper could be helpful from a very early stage of preparing a novice programmer. The specific problem domain chosen for experiments, however, is part a rather intermediate level. Structured data types are definitely not one of the introductory chapters in studying programing. Therefore, our recommendation would be to use it when dealing with topics related to problem solving with data structures.

Appendix A. Experimental knowledge base (set of rules) written in natural language

Rule 1:
IF 1. Use of structured data type is recommended
2. Cardinality of multiset cannot be estimated in advance
THEN Use of dynamic data structure is recommended

Rule 2:
IF 1. Use of dynamic data structure is recommended
2. The problem requires explicity representation of binary relation
3. The multiset of elements is set of elements
THEN Use of graph is recommended

Rule 3:
IF 1. Use of structured data type is recommended
2. The problem requires explicity representation of binary relation
3. The multiset of elements is set of elements
THEN Use of graph is recommended

Rule 4:
IF 1. Use of table is recommended
2.Complexity consideration tells us that access time should not be dependent on size of the set of elements
THEN Use of hash table is recommended

Rule 5:
IF 1. Use of structured data type is recommended
2. It can be assumed that the order of processing of elements will be sequential most of the time
THEN Use of linear list is recommended

Rule 6:
IF 1. Use of dynamic data structure is recommended
2. It can be assumed that the order of processing of elements will be sequential most of the time
THEN Use of linear list is recommended

Rule 7:
IF 1. Use of structured data type is recommended
2. There exists mapping from the set of ordinals to the set of elements
3. All elements are of same type
4. Order of processing of elements cannot be decided in advance
THEN Use of one dimensional array is recommended

Rule 8:
IF 1. Use of structured data type is recommended
2. Ordering upon multiset of elements is not defined
3. All elements are of same type
4. Frequency of operations over more than one lament is high
5. There exists mapping from the set of ordinals to the set of elements
THEN Use of one dimensional array is recommended

Rule 9:
IF 1. Use of structured data type is recommended
2. During processing multiset of elements first-in first-out strategy will be employed
THEN Use of queue is recommended

Rule 10:
IF 1. Use of dynamic data structure is recommended
2. During processing multiset of elements first-in first-out strategy will be employed
THEN Use of queue is recommended

Rule 11:
IF 1. Use of structured data type is recommended
2. Ordering upon set of elements is defined
3. This ordering is referred to in problem specification
4. Multiset of elements is set of elements
THEN Use of search tree is recommended

Rule 12:
IF 1. Use of dynamic data structure is recommended
2. Ordering upon set of elements is defined
3. This ordering is referred to in problem specification
4. Multiset of elements is set of elements

THEN Use of search tree is recommended

Rule 13:
IF 1. Use of structured data type is recommended
2. Ordering upon set of elements is defined
3. Frequency of operations remove element and insert element is high
4. Multiset of elements is set of elements
THEN Use of search tree is recommended

Rule 14:
IF 1. Use of dynamic data structure is recommended
2. Ordering upon set of elements is defined
3. Frequency of operations remove element and insert element is high
4. Multiset of elements is set of elements
THEN Use of search tree is recommended

Rule 15:
IF 1. Use of structured data type is recommended
2. During processing of elements has-in first-our strategy will be employed
THEN Use of stack is recommended

Rule 16:
IF 1. Use of dynamic data structure is recommended
2. During processing of elements las-in first-out strategy will be employed
THEN Use of stack is recommended

Rule 17:
IF 1. Input is multiset of elements
2. It cannot be ruled out that while processing the multiset, elements will be processed more than once
THEN Use of structured data type is recommended

Rule 18:
IF 1. Use of structured data type is recommended
2. Accessing an element depends on the contents of this element
THEN Use of table is recommended

Rule 19:
IF 1. Use of table is recommended
2. Ordering upon set of elements is defined
3. Optimization of the accessing of element is required
THEN Use of table with ordered elements is recommended

Rule 20:
IF 1. Use of graph is recommended
2. For the relation predecessor successor defined on the set the inverse relation in one-one
3. Only one element exists, for which the relation predecessor successor is not defined (root)
THEN Use of tree is recommended

Rule 21:
IF 1. Use of structured data type is recommended
2. Problem solving method presupposes organizing multiset of elements as matrix
3. All elements are of same type
4. Matrix is not sparse matrix
THEN Use of two dimensional array is recommended

Rule 22:
IF 1. Use of structured data type is recommended
2. Specification presupposes organising multiset of elements as matrix
3. All elements are of same type
4. Matrix is not sparse matrix
THEN Use of two dimensional array is recommended

Rule 23:
IF 1. Use of structured data type is recommended
2. All elements are of same type
3. There exists mapping from the pair of ordinals to the set of elements
4. This mapping will be used for accessing an element
THEN Use of two dimensional array is recommended

Appendix B : Experimental knowledge base (set of rules) as written in NEXPERT

Rule 1

If

there is evidence of | Structured_type|. Use_is_recommended
And there is no evidence of |Inputs|. Cardinal_not_estim
Then Dynamic_data_type. Use_is_recommended
is confirmed.

Rule 3

If

there is evidence of |Structured_data_type|. Use_is_recommended
And there is evidence of |Processing|. Repr_bin_rel_req
And there is evidence of |Inputs|. Multiset_is_set
Then Graph. Use_is_recommended
is confirmed.

Rule 2

If

there is evidence of |Dynamic_data_structure|. Use_is_recommended
And there is evidence of |Processing|. Repr_bin_rel_req
And there i evidence of |Inputs|. Multiset_is_set
Then Graph. Use_is_recommended
is confirmed.

Rule 4

If

there is evidence of |Table|. Use_is_recommended
And there is evidence of |Processing|. Acc_time_not_dep_set_size
Then Has_table. Use_is_recommended
is confirmed.

Rule 6

If

there is evidence of |Dynamic_data_structure|. Use_is_recommended
And there is evidence of |Processing|. Order_process_sequent _most
Then Linear_ist. Use_is_recommended
is confirmed.

Rule 5

If
there is evidence of |Structured_data_type|. Use_is_recommended
And there is evidence of |Processing|. Order_process_sequent_most
Then Linear_list. use_is_recommended
is confirmed.

Rule 8

If
there is evidence of |Structured_dat~_type|. Use_is_recommended
And there is no evidence of |Inputs|. Ordering_is_defined
And there is evidence of |Inputs|. All_else_same_type
And there is evidence of |Processing|. Freq_oper_more_ele_high
And there is evidence of |Inputs|. Mapp set ord set ele
Then One_dimensional_array. Use_-s_recommended
is confirmed.

Rule 7

If
there is evidence of |Structured_data_type|. Use_is_recommended
And there is evidence of |Inputs|. Mapp_set_ord_set_ele
And there is evidence of |Inputs|. All_ele_same_type
And there is no evidence of |Processing|. Order_process_decided_adv
Then One_dimensional_array. use_is_recommended
is confirmed.

Rule 10

If
there is evidence of |Dynamic_data_structure|. Use_is_recommended
And there is evidence of |Processing|. FIFO_strategy
Then Queue. Use_is_recommended
is confirmed.
And |Stack|. Use_is_recommended is set to FALSE

Rule 9

If
there is evidence of |Structured_dat_type|. Use_is_recommended
And there is evidence of |Processing|. FIFO_strategy
Then Queue. Use_is_recommended
is confirmed.
And |Stack|. Use_is_recommended is set to FALSE

Rule 14

If
there is evidence of |Structured_data_type|. Use_is_recommended
And there is evidence of |Inputs|. Ordering_is_defined
And there is evidence of |Processing|. Freq_oper_ins_remove_high
And there is evidence of |Inputs|. Multiset_is_set
Then Search_tree. Use_is_recommended
is confirmed.

Rule 13

If
there is evidence of |Dynamic_data_structure|. Use_is_recommended
And there is evidence of |Inputs|. Ordering_is_defined
And there is evidence of |Processing|. Freq_oper_ins_remove_high
And there is evidence of |Inputs|. Multiset_is_set
Then Search_tree. Use_is_recommended
if confirmed.

Rule 12

If
there is evidence of |Dynamic_dat_structure|. Use_is_recommended
And there is evidence of |Inputs|. Ordering_is_defined
And there is evidence of |Inputs|. Ordering _in_specif
And there is evidence of |Inputs|. Multiset_is_set
Then Search_tree. Use_is_recommended
is confirmed.

Rule 11

If
there is evidence of |Structured_data_type|. Use_is_recommended
And there is evidence of |Inputs|. Ordering_is_defined
And there is evidence of |Inputs|. Ordering_is_specif
And there is evidence of |Inputs|. Multiset_is_set
Then Search_tree. Use_is_recommended
is confirmed.

Rule 16

If
there is evidence of |Structured_data_type|. Use_is_recommended
And there is evidence of |Processing|. LIFO_strategy
Then Stack. Use_is_recommended
is confirmed
And |Queen|. UYse_is_recommended is set to FALSE.

Rule 15

If there is evidence of |Dynamic_data_structure|. Use_is_recommended
And there is evidence of |Processing|. LIFO_strategy
Then Stack. Use_is_recommendd
is confirmed.
And |Queue|. Use_is_recommended is set to FALSE

Rule 17

If there is evidence of |Inputs|. Is_multiset_of_elements
And there is evidence of |Processing|. Proc_ele_mset_more_once
Then Structured_data_type. Use_is _recommended
is confirmed.

Rule 18

If
there is evidence of |Structured_data_type|. Use_is_recommended
And there is evidence of |Procesing|. Acc_ele_depends_contents
Then Table. Use_is_recommended
is confirmed.

Rule 19

If
there is evidence of |Table|. Use_is_recommended
And there is evidence of |Inputs|. Ordering_is_defined
And there is evidence of |Processing|. Acc_ele_optimiz_req
Then Table_with_ordered_elements. use_is recommended
is confirmed

Rule 20

If
there is evidence of |Graph|. Use_is_recommended
And there is evidence of |Processing|. Rel_pel_pred_succ_inv_one_one
And there is evidence of |Processing|. Rel_pred_succ_not_ex_one
Then Tree. Use_is_recommended
is confirmed.

Rule 21

If

there is evidence of |Structured_data-type|. Use_is_recommended
And there is evidence of |Problem_solving_method|. Org_ multiset_ as_matrix
And there is evidence of |Inputs|. All_ele_same_type
And there is no evidence of |Inputs|. Matrix_is_sparse
Then Two-dimensional_array. Use_is_recommended
is confirmed.
And |One_dimensional-array|. Use_is_recommended is set to FALSE

Rule 22

If

there is evidence of |Structured data type|. Use_is_recommended
And there is evidence of |Inputs|. Specif_org_multiset_matrix
And there is evidence of |Inputs|. All_ele_same_type
And there is no evidence of |Inputs|. Matrix_is_sparse
Then Two_dimensional_array. Use_ is_recommended
is confirmed.
And |One_dimensional_array|. Use_is_recommended is set to FALSE

Rule 23

If

there is evidence of |Structured_data_type|. Use_is_recommended
And there is evidence of |Inputs|. All_ele_same_type
And there is evidence of |Inputs|. Mapp_par_ord_set_ele
And there is evidence of |Inputs|. Mapp_used_access_ele
Then Two_dimensional_array. Use_is_recommended
is confirmed.
And |One_dimensional_array|. Use_is_recommended is set to FALSE.

Appendix C : Problem Specification for the Experiment

Specification is given as written in NEXPERT

Suggesting Search_tree.. Use_is_recommended

Appendix D : Trace of Reasoning

The sequence of rules' conditions and conclusions aas they wre evaluated.

Suggesting Search_tree. Use_is_recommended
& Inputs. Is_multiset_of_elements is set to True
# Condition there is evidence of |Inputs|. Is_multiset_of_elements in rule 17. (True).
# Processing. proc_ele_mset_more_once is set to True
# Condition there is evidenc of
|Processing|. Proc_ele_mset_more_once in rule 17. (True).
Rule 17 is set to ture
Structured_data_type. Use_is_recommended is set to True
Condition there is evidence of
|Structured_dat_type|. Use_is_recommended in rule 11. (True).
Inputs. Ordering_is_defined is set to True
Condition there is evidence of |Inputs|. Ordering_is_defined in rule 11. (True).
Processing. Freq_oper_ins_remove_high is set to True
Condition there is evidence of
|Processing|. Freq-oper_ins_remove_high in rule 11. (True).
Inputs. multiset_is_set is set to True
Condition there is evidence of |Inputs|. Multiset_is_set in rule 11. (True).
Rule 11 is set to true

Search_tree. Use_is_is_recommended is set to True
Condition there is evidence of
|Structured_data_type|. Use_is_recommended in rule 14. (True).
Condition there is evidence of |Inputs|. Ordering_is_defined in rule 14. (True).
Inputs. Ordering_in_specif is set to True
Condition there is evidenc eof |Inputs|.Ordering_in_specif in rule 14 (True).
Condition there is evidence of |Inputs|.Multiset_is_set in rule 14. (True).
Rule 14 is set to true
Condtion there is evidence of
|Structured_data_type|. Use_is_recommended in rule 1. (True).
Inputs. Cardinal not estim is set to False
Condition there is no evidence of |Inputs|. Cardinal_not_cstim in rule 1. (True).
Rule 1 is set to true
Dynamic_data_type. use_is_recommended is set to True
Condition there is evidence of
|Dynamic_data_type|. Use_is_recommended in rule 13. (True).
Condition there is evidence of |Inputs|. Ordering_is_defined in rule 13, (True).

Condition there is evidence of |Inputs|. Ordering_in_specif in rule 13. (True).
Condition there is evidence of |Inputs|. Multiset_is_set in rule 13. (True).
3 Rule 13 is set to true
Condition there is evidence of
|Dynamic_data_type|. Use_is_recommended in rule 12. (True).
Condition there is evidence of |Inputs|. Ordering_is_defined in rule 12. (True).
Condition there is evidence of |Inpauts|. Multiset_is_set in rule 12. (True).
Rule 12 is set to true
Condition there is evidence of
|Structured_data_type|.Use_is_recommended in rule 21. (True).
Problem_solving_method.Org-multiset_as_matrix is set to False
Condition there is evidence of
|Problem_solving_method|.Org_multiset_as_matrix in rule 21. (False).
Rule 21 is set to false
Condition there is evidence of
|Structured_data_type|.Use_is_recommended in rule 22. (True).
Inputs.Specif_org_multiset_matrix is set to False
Condition there is evidence of
|Inputs|. Specif_org_multiset_matrix in rule 22. (False).
Rule 22 is set to false
Condition there is evidence of
|Structured_data_type|.Use_is_recommended in rule 23. (True).
Inputs.All_ele_same_type is set to True
Condition there is evidence of |Inputs|.All_ele_same_type in rule 23. (True).
Inputs.Mapp_pair_ord_set_ele is set to False
Condition there is evidence of |Inputs|.Mapp_pari_ord_set_ele in rule 23. (False).
Rule 23 is set to false
Two dimensional array.Use_is_recommended is set to False
Condition there is evidence of
|Structured_data_type|.Use_is_recommended in rule 18. (True).
Processing.Acc_ele_depends_contents is set to True
Condition there is evidence of
|Procesing|.Acc_ele_depends_contents in rule 18. (True).
Rule 18 is set to true
Table.Use_is_recommended is set to True
Condition there is evidence of |Table|.Use_is_recommended in rule 4. (True).
Processing.Acc_time_not_dep_set_size is set to True
Condition there is evidence of
|Processing|.Acc_time_not_dep_set_size in rule 4. (True).
Rule 4 is set to true
Hash_table.Use_is_recommended is set to True
Condition there is evidenceof |Table|.Use_is_recommended in rule 19. (True).

```
# Condition there is evidence of |Inputs|.Ordering_is_defined in rule 19. (True).
# Processing.Acc_ele_optimiz_req is set to True
# Condition there is evidence of |Processing|.Acc_ele_optimiz_req in rule 19. (True).
# Rule 19 is set to true
# Table_with_ordered_elements.Use_is_recommended is set to True
# Condition there is evidence of
|Structured_data_type|.Use_is_recommended in rule 15. (True).
# Procesing. LIFO_strategy is set to False
# Condition there is evidence of |Processing|. LIFO-strategy in rule 15. (False).
# Rule 15 is set to false
# Rule 16 is set to false
# Stack. Use_is_recommended is set to False
# Condition there is evidence of
|Structured_data_type|.Use_is_recommended in rule 10. (True).
# Processing. FIFO_strategy is set to False
# Condition there is evidence of |Processing|. FIFO_strategy in rule 10. (False).
# Rule 10 is set to false
# Rule 9 is set to false
# Queue.Use_is_recommended is set to False
# Condition there is evidence of
|Structured_data_type|.Use_is_recommended in rule 8. (True).
# Inputs.Mapp_set_ord_set_ele is set to True
# Condition there is evidence of |Inputs|.Mapp_set_ord_set_ele in rule 8. (True).
# Condition there is evidence of |Inputs|.All_ele_same_type in rule 8. (True).
# Processing.Order_process_decided_adv is set to False
# Condition there is no evidence of
|Processing|.Order_process_decided_adv in rule 8. (True).
# Rule 8 is set to true
# One-dimensional-array.Use_is_recommended is set to True
# Rule 7 is set to false
# Condition there is evidence of
|Structured_data_type|.Use_is_recommended in rule 6. (True).
# Processing.Order_process_sequent_most is set to False
# Condition there is evidence of
|Processing|.Order_process_sequent_most in rule 6. (False).
# Rule 6 is set to false
# Rule 5 is set to false
# Linear_list.Use_is_recommended is set to False
# Condition there is evidence of
|Structured_data_type|.Use_is_recommended in rule 2. (True).
# Processing.Repr_bin_rel_req is set to True
```

Condition there is evidence of |Processing|.Repr_bin_rel_req in rule 2. (True).
Condition there is evidence of |Inputs|.Multset_is_set in rule 2. (True).
Rule 2 is set to true
Graph.Use_is_recommended is set to True
Condition there is evidence of
|Dynamic_data_type|.Use_is_recommended in rule 3. (True).
Condition there is evidence of |Processing|.Repr_bin_rel_req in rule 3 (True).
Condition there is evidence of |Inputs|.Multiset_is_set in rule 3. (True).
Rule 3 is set to true
Condition there is evidence of |Graph|.Use_is_recommended in rule 20. (True).
Processing.Rel_pred_succ_inv_one_one is set to True
Condition there is evidence of
|Processing|.Rel_pred_succ_inv_one_one in rule 20. (True).
Processing.Rel_pred_succ_not_ex)_one is set to True
Condition there is evidence of
|Processing|.Rel_pred_succ_not_ex_one in rule 20. (True).
Rule 20 is set to true
Tree.Use_is_recommended is set to True

Appendix E : Results of the Reasoning Process

All the hypotheses that were established (with their respective suggestive evidence) and rejected (with their respective counter arguments) are listed.

Hypothesis Dynamic_data_type.Use_is_recommended was established
Suggestive Evidence:
Rule Number 1
|Structured_data_type|.Use_is_recommended is TRUE
|Inputs|.Cardinal_not_estim is FALSE

Hypothesis Graph.Use_is_recommended was established
Suggestive Evidence:
Rule Number 3
|Dynamic_data_type|.Use_is_recommended is TRUE
|Processing|.Repr_bin_rel_req is TRUE
|Inputs|.Multiset_is_set is TRUE
Rule Number 2
|Structured_data_type|.Use_is_recommended is TRUE
|Procesing|.Repr_bin_rel is TRUE
|Inputs|.Multiset_is_set is TRUE

Hypothesis Has_table.use_is_recommnded was establihsed
Suggestive Evidence:
Rule Number 4
|Table|.Use_is_recommended is TRUE
|Processing|.Acc_time_not_dept_set_size is TRUE

Hypothesis Linear_list.Use_is_recommended was rejected
Counter Arguments:
Rule Number 5
|Processing|.Order_process_sequent_most is FALSE
Rule Number 6
|Processing|. Order_process_sequent_most is FALSE
Hypothesis One_dimensional_array.Use_is_recommended was esta-blished
Suggestive Evidence:
Rule Number 8
|Structured_data_type|. Use_is_recommended is TRUE
|Inputs|.Mapp_set_ord_set_ele is TRUE
|Inputs|.All_ele_same_type is TRUE
|Processing|.Order_process_decided_adv is FALSE
Counter Arguments:
Rule Number 7
|Inputs|.Ordering_is_defined is TRUE

Hypothesis Queue.Use_is_recommended was rejected
Counter Arguments:
Rule Number 9
|Processing|.FIFO_strategy is FALSE
Rule Number 10
|Procesing|. FIFO_strategy is FALSE

Hypothesis Search_tree.Use_is_recommended was established
Suggestive Evidence:
Rule Number 12
|Dynamic_data_type|.Use_is_recommended is TRUE
|Inputs|.Orderin_is_defined is TRUE
|Processing|.Freq_oper_ins_remove_igh is TRUE
|Inputs|.Multiset_is_set is TRUE
Rule Number 13
|Dynamic_data_type|.Use_is_recommended is TRUE
|Inputs|.Ordering_is_defined is TRUE

|Inputs|.Ordering_in-specif is TRUE
|Inputs|.Multiset_is_set is TRUE
Rule Number 14
|Structured_data_type|.Use_is_recommended is TRUE
|Inputs|.Ordering_is_defined is TRUE
|Inputs|.Ordering_in_specif is TRUE
|Inputs|.Multiset_is_set is TRUE
Rule Number 11
|Structured_data_type|.Use_is_recommended is TRUE
|Inputs|.Ordering_is_defined is TRUE
|Processing|.Freq_oper_ins_remove_high is TRUE
|Inputs|.Multiset_is_set is TRUE

Hypothesis Stack.Use_is_recommended was rejected
Counter Arguments:
Rule Number 16
|Processing|.LIFO_strategy is FALSE
Rule Number 15
|Processing|.LIFO_strategy is FALSE

Hypotheis Structured_data_type.Use_is_recommended was established
Suggestive Evidence:
Rule Number 17
|Inputs|.Is_multiset_of_elements is TRUE
|Processing|.Proc-ele-mset_more_once is TRUE

Hypothesis Table.Use_is_recommended was established
Suggestive Evidence:
Rule Number 18
|Structured_data_type|.Use_is_recommended is TRUE
|Processing|.Acc_ele_depends_contents is TRUE

Hypothesis Table with ordered elements.Use is recommended was established
Suggestive Evidence:
Rule Number 19
|Table|.Use_is_recommended is TRUE
|Inputs|.Ordering_is_defined is TRUE
|Processing|.Acc_ele_optimiz_req is TRUE

Hypothesis Tree.Use_is_recommended was established
Suggestive Evidence:
Rule Number 20
|Graph|.Rel_pred_succ_inv_one_one is TRUE
|Processing|.Rel_pred_succ_not_ex_one is TRUE

Hypothesis Two_dimensional_array.Use_is_recommended was rejected
Counter Arguments:
Rule Number 23
|Inputs|.Mapp_pair_ord_set_ele is FALSE
Rule Number 22
|Inputs|.Specif_org_multiset_matrix is FALSE
Rule Number 21
|Problem_solving_method|.Org_multiset_as_matrix is FALSE

References

[1] Anderson, J.R.; Farrell, R., and Sauers, R. "Learning to Program in LISP." *Cognitive Science*, 8, (1984), 87-129.

[2] Anderson, J.R. and Skwarecki, E. "The Automated Tutoring of Introductory Computer Programming." *Communications of the ACM*, 29, No.9 (1986), 842-849.

[3] Aslett, M.J. "*A Knowledge Based Approach to Software Development.*" Amsterdam, The Netherlands: North Holland Publishing Co., (1991), 249

[4] Boyle, T. and Margetts, S. "The CORE Guided Discovery Approach to Acquiring Programming Skills." *Computers and Education*, 18, No.1-3 (1992), 127-133.

[5] Dasgupta, S. "The Structure of Design Processes." In : *Advances in Computers*, Vol. 28, New York : Academic Press, (1989), 1-67.

[6] Dijkstra, E.W. "*A Discipline of Programming.*" Englewood Cliffs: Prentice hall, 1976.

[7] Fickas, S. and Helm, B.R. "Knowledge Representation and Reasoning in the Design of Composite Systems." *IEEE Transactions on software Engineering*, 18, No.6 (1992), 470-487.

[8] Gasparovic, L. and Navrat, P. "Extalk - Smalltalk Based Expert Systems Development Tool." In: A Mrazik (Ed.) *Proc. EastDurOOPe'91, Short Papers*, Bratislava 1991, 65-73.

[9] Green, C. and Barstow, D.R. "On Program Synthesis Knowledge." *Artificial Intelligence*, 10, (1978), 241-279.

[10] Johnwon, W.L. and Soloway, E.M. "PROUST: An Automatic Debugger for Pascal Programs." *Byte*, 10, No.4 (1985), 179-190.

[11] Keiser, G.E. and Feiler, P.H. "An Architecture for Intelligent Assistance in Software Development." In: *Proc. ACM* , (1987), 180-188.

[12] Krieg-Bruckner, B. "The PROSPECTRA Methodology of Program Development." In: J. Zalewski, W. Ehrenberger (Ed.): *IFIP* 1989, Amsterdam : Elsevier, (1989), 257-271.

[13] Molnar, L "A Knowledge Based Program Creation." *DrSc. Dissertation. Slovak Technical University*, Bratislava, (1989).

[14] Navrat, P. "Towards a Master Programmer: A Paradigm for Automated Tutoring of Programming." In: I. Plander (Ed.) *Proc. Artificial Intelligence and Information Control Systems of Robots* 87, Amsterdam : North-Holland, (1987), 375-379.

[15] Navrat, P.; Molnar, L., and Vijtek, V."Using Automatic Program Synthesizer to Generate Solutions from Diverse Problem Environments." *Computers and Aritficial Intelligence*, 7, No.2 (1988), 139-146.

[16] Navrat, P. and Mlada, I. "What Knowledge is The Knowledge Based Programming Based On?: An Inquiry into Knowledge Sources." In: I. Plander (Ed.) *Proc. Artificial Intelligence and Information-Control Systems of Robots* 89, Amsterdam: North-Holland, 1989, pp. 187-190.

[17] Navrat, P. - Fric, P. - Adamy, M. - Mlada, I. : KEX: Computer Aided Knowledge Engineering System. In: Proc. Computers'89 Conference, Blahova,(1989), 156-162.

[18] Navrat, P. ; Molnar, L. and Vojtek, V. "Using Automatic Program Synthesizer as a Problem Solver: Some Interesting Experiments." In: J. Davenport (Ed.) *Proc. EUROCAL'87, LNCS 378*, Berlin: Springer, , (1989), 412-423.

[19] Navrat, P. and Fric, P. "A Tool for Knowledge-Based Systems Development." In: V. Marik (Ed.) *Proc. Artificial Intelligence Applications Conference AI'91*, Prague, (1991), 351-360.

[20] Navrat, P. "Intelligent Support for Software Construction, and Higher Education in Informatics at The Slovak TU: The DEC Connection." In: *Proc. DECSYM92 Latest Trends in Computing*, , Side - Antalya: (1992), 253-263.

[21] Navrat, P. and Rozinajova, V."Making Programming Knowledge Explicit." *Computers and Education*, 21, No.4 (1993), 281-299.

[22] Rich, C. "Feldman, Y.A. : Seven Layers of Knowledge Representation and Reasoning in Support of Software Development." *IEEE Transactions on Software Engineering*, 18, No.6 (1992), 451-469.

[23] Rozinajova, V. and Navrat, P. "Explicit Knowledge Representation in Support of Learning Programming." In: P. Brna; S. Ohlsson, H. Pain, (Eds.), *Proc. AI-ED 93 World Conference on Artificial Intelligence in Education*, Charlottesville: Association for the Advancement of Computing in Education, (1993), 584.

[24] Setliff, D. and Rutenbar, R.A."Knowledge Representation and Reasoningin a Software Synthesis." *IEEE Transactions on Software Engineering*, 18, No.6 (1992), 523-533.

[25] Smith, D.R. "KIDS: A Semiautomatic Program Development System." *IEEE Transactions on Software Engineering*, 16, No.9 (1990), 1024-1043.

[26] Smith, D.R. and Lowry, M.R."Algorithm Theories and Design Tactics." *Science of Computer Programming*, 14, (1990), 305-321.

[27] Soloway, E., Ehrlich, K."Empirical Studies of Programming Knowledge." *IEEE Transactions on Software Engineering*, 10, No.5 (1984), 595-609.

[28] Office of Naval Research "*Symposium on Automatic Programming for Digital Computers.*" Dept. of Navy, Washington D.C. , 1954.

[29] Turski, W.M. and Maibaum, T.S.E."T*he Specification of Computer Programs.*" Wokingham: Addison Wesley, (1987), 287 .

[30] Webb, M. "Learning of Building Rule-Based Models." *Computers and Education*, 18, No.1-3 (1992),89-100.

تجربة في البرمجة المبنية على المعرفة

بافول نافرات و فييرا روزيناجوفا

قسم علوم هندسةالحاسب، جامعة سلوفاكيا التقنية
الكوفيكوا، براتشلوفا، سلوفاكيا

(قدم للنشر في ٠٦/ ٠٩/ ١٩٩٣م، وقبل للنشر في ٢١/ ٠٩/ ١٩٩٤م)

ملخص البحث . تبحث هذه الورقة إمكانية كتابة البرامج الحاسوبية من غير أن تكون هناك معرفة تامة بطريقة البرمجة . لكي تجرى التجارب قمنا بإعداد قاعدة معرفة تشتمل على معلومات عن كيفية اختيار نوع البيانات المناسب عند كتابة برنامج . بالإضافة إلى هذه القاعدة، نقدم تجربة مع واجهة للتعامل مع المستخدم ونتائج تحليل الآداء . من خلال التجربة يقوم المستخدم باختيار نوع بيانات مقترح ويرد النظام على الاختيار إما بالموافقة أو الرفض وكنتيجة لإمكانات النظام للاستدلال المنطقي . يقوم النظام باختيار نوع بيانات بديل . أوضحت النتائج أن النظام بمكن أن يكون مفيداً في توجيه المبرمج وقديكون مهماً في عملية تعليم البرمجة .