COLING 82, J. Horecký (ed.)
North-Holland Publishing Company

KNOWLEDGE REPRESENTATION METHOD BASED ON PREDICATE CALCULUS IN AN INTELLIGENT CAI SYSTEM

Barbara Begier

Regional Computer Center
Technical University of Poznań
Poland

The knowledge representation method is introduced to be applied in the ICAI system to teach programming language. Knowledge about syntax and semantics of that language is represented by a set of axioms written in the predicate calculus language. The directed graph of concepts is mentioned as a method to represent an instructional structure of the domain knowledge. The proof procedure to answer student's questions is described.

INTRODUCTION

The early 70s have brought the Intelligent CAI /ICAI/ systems [3,4]. In these systems all course material is represented independently of teaching procedures. The goal of ICAI research is to obtain an individualization of instruction by providing an ability of answering student's questions as well as generating remedial comments, problems and advices, accordingly to current student's responses and his abilities and preferences in general.

ICAI researchers have firstly focused their investigations on representation of the subject matter. Mostly semantic nets have been used as representation of static [3] and procedural [4,6] domain knowledge. Database of an instructional system includes also a representation of curriculum to organize an instruction of new knowledge [5,7]. It is helpful in selecting material to be presented to the student.

This paper presents the knowledge representation method based on predicate calculus in an intelligent CAI system, which is applied to teach the programming language [1]. The prototype of the presented approach was an application of predicate calculus to describe programs, written in ALGOL, to prove their correctness, introduced by Burstall [2]. The knowledge about syntax and semantics of the programming language has been represented in the form of a set of first order logic axioms.

There are given some notes how to construct the predicate calculus language and axioms describing the subject matter. The directed graph of concepts as a method to represent an instructional structure of the domain knowledge has been introduced along with the manner of generating instructional texts to the student. Then the procedure answering student's questions has been described.

PREDICATE CALCULUS APPLIED TO REPRESENT KNOWLEDGE ABOUT THE PROGRAMMING LANGUAGE

The following criteria may be referred to knowledge representation methods for ICAI systems [1,7]:

- ability to express a large set of concepts of the domain being taught,
- facility of coding these concepts and relations among them,
- easy way to transform the formal notation into the natural language form,
- efficiency of information retrieval during the process of answering user's query and proving the correctness of his answer,
- ability of automated deduction application in the question - answering process.

Let us consider a subset of an ALGOL-like programming language, containing simple arithmetic and logical expressions, instruction of substitution and conditional and goto instructions. We assume that each instruction has been written in a separate line of program.

The predicate calculus language developed to represent knowledge about the programming language contains:

- names of sets, called sorts of objects, representing elements of syntax and semantics of a programming language,
- functions, transforming objects,
- predicates, representing relations between objects.

Some sorts, functions and predicates are introduced to represent syntax of the programming language. Others represent its semantics.

N o t a t i o n. The ordinary predicate calculus notation has been used. Some modifications improve the readability of statements:

- unquantified variables are generally quantified,
- two-places predicates are written in an infix manner,
- binary arithmetic functions are written in an infix manner,
- parenthesis are used in the ordinary meaning,
- clauses are separated by dots.

P r o g r a m s y n t a x. The program syntax has been described by a set of clauses written in the predicate calculus language. Sorts of objects /examples/: identifier, number, expression, arithmetic expression, logical expression, label, instruction, program line.

Functions transform some expressions into terms, by example:

dod: wa × wa → wa	where: wa - arithmetic expression,
lt: wa × wa → wl	wl - logical expression,
pod: id × wy → in	id - identifier,
sko: et → in	wy - expression, et - label,
ifl: wl × wi → in	in - instruction,
	wi - program line.

First of these functions constructs an expression, which represents an operation of addition, the second one gives as a result an expression representing the "less than" relation and the others construct appropriately the substitution instruction, the goto instruction and the conditional instruction.

Some predicates have been introduced to represent the syntax relations between syntax objects, like following:

wins ⊆ wi × in

pet ⊆ et × wi

bnas ⊆ wi × wi

First of these predicates indicates the location of an instruction in a given program line, the second one assigns a label at the beginning of a program line and the third one determines the direct succession of two program lines in a sequence.

Example. The syntax of a program containing three following substitutions:

$$
\begin{array}{rl}
 & J = L \\
 & L = L + 2 \\
E: & K = J + L
\end{array}
$$

is described like that:

$$w1\ \underline{\text{wins}}\ \text{pod}(J,L).\quad w2\ \underline{\text{bnas}}\ w1.\quad w2\ \underline{\text{wins}}\ \text{pod}(L,\text{dod}(L,2)).$$
$$E\ \underline{\text{pet}}\ w3.\quad w3\ \underline{\text{bnas}}\ w2.\quad w3\ \underline{\text{wins}}\ \text{pod}(K,\text{dod}(J,L)).$$

P r o g r a m s e m a n t i c s. Semantics of an algorithmic language is represented by a set of axioms written in the predicate calculus language. New sorts of objects, functions and predicates are to be introduced.
Sorts of objects: value, state
Value set contains values of arithmetic and logical expressions, arrays, subroutines or procedures etc. The particular kind of value is a location in a program, represented by a program line. A state is assigned to a program line and it is determined also by the valuation of variables.

Functions evaluate expressions:
$\quad$ owar: $wy \times st \rightarrow wr$
and sequence of states:
$\quad$ nas: $st \rightarrow st$

where: wy - expression,
st - state,
wr - value.

Predicates assign states to locations in a program:
$\quad \underline{\text{stwe}} \subseteq st \times wi$
$\quad \underline{\text{stzm}} \subseteq st \times wi$

First of them associates a state to a program line before an execution of an instruction from this line. The second one indicates, that the control passes to an instruction written in a given program line in a given state.

Sorts of objects, functions and predicates are the basis of a grammar of the predicate calculus language, which expressions are used to represent knowledge about the programming language.

Axioms written in this language describe arithmetic rules, properties of expressions, semantics of instructions /principles of execution of instructions/, meaning of program segments or blocks etc.

Example. An axiom describing semantics of an instruction of substitution:

$$
\begin{array}{l}
w1\ \underline{\text{wins}}\ \text{pod}(J,X) \\
\wedge\ s\ \underline{\text{stwe}}\ w1 \\
\wedge\ w2\ \underline{\text{bnas}}\ w1 \\
\qquad\qquad \Longrightarrow \text{nas}(s)\ \underline{\text{stwe}}\ w2 \\
\qquad\qquad\quad \wedge\ \text{owar}(J,\text{nas}(s)) = \text{owar}(X,s) \\
\qquad\qquad\quad \wedge\ \forall Y\ [Y \neq J \Longrightarrow \text{owar}(Y,\text{nas}(s)) = \text{owar}(Y,s)]
\end{array}
$$

The premises of this axiom include: the location of the substitution J = X in the program line, named w1, the assigning of the state s before an execution of this instruction, and a fact, that the program line w2 directly follows w1. The conclusion says, that a state next of s is the state before an execution of an instruction written in w2 and a value of the variable J in the state next of s is a value of an expression X in the state s and a value of any variable $Y \neq J$ doesn't change during the transfer from the state s to the next of s. A large subset of FORTRAN has been described in this manner [1].

It turns out that formulas of predicate calculus are easy to transform into natural language expressions. Axioms are divided into simple sentences. Translation rules are applied to simple sentences. Each object has a name in natural language. Also an appropriate natural language expression is selected for each function. Each predicate corresponds with a verb phrase in natural language. The proper translation rules for functions and objects are applied with reference to arguments of a predicate. Translation rules for Polish language have been reported in [1] as well as their application to all axioms describing FORTRAN.

DIRECTED GRAPH AS A METHOD OF INSTRUCTIONAL STRUCTURE REPRESENTATION

An assumption is done that all knowledge to be taught can be divided to instructional units. Thus the first step to construct an instructional structure representation is to select such units /concepts/. Each of them has a name and at least one sentence can be told about it /unreal concepts are not allowed/. Some introductory concepts are assumed to be known to the student.

The next step is to specify all relations between concepts. These relations could be:

/a/ Concept X is a part of Y,
/b/ Concept X has a property, represented by Z,
/c/ Concept X is a reason or a justification of T,
/d/ Concept X belongs to the object class represented by K,
/e/ Concept X is an alternative of A,
/f/ Concept X is equivalent to W at least in some circumstances.

Each relation corresponds with a graph, which nodes represent concepts from an introduced set of concepts. The composition of all obtained graphs results in a final graph, which represents an instructional structure of the subject matter. Because of the different interpretation of the particular arches of this graph /which are described by various relationships/ the "superior-inferior" relation is introduced as the universal one which represents every relation between concepts. Thus the directed graph has been obtained, with arrows directed to the superior concepts.

A set of axioms is associated with each node of the concept graph. Also some other information may be associated with it.

ANSWERING STUDENT'S QUESTIONS

The following problems have to be solved:

- choice and specifying of classes of user's queries, which can be answered by the ICAI system,
- recognition of a main subject of the query,
- translation of the query from natural language to the predicate calculus language formula,
- application of the automated theorem-proving techniques to retrieve an answer,
- generating of an answer in natural language form.

Three classes of queries have been considered:

/1/ Decision queries of general form in natural language

⟨interrogative particle⟩ ⟨sentence⟩ ?

where: ⟨interrogative particle⟩ /existing in Polish/ determines that a question belongs to this class,
⟨sentence⟩ - indicative sentence,

which require an answer in the form "Yes" or "No".

/2/ Objective queries of general form in natural language

Which ⟨general name⟩ ⟨predicate⟩ ?

which require to retrieve an object posessing some given features as an answer.

The query of this class may be transformed into the form:

Which X satisfies a condition: $W(X) \Rightarrow A(X)$?

where: X - an object to be found,

$W(X)$ - distinctive predicate of a set, which is specified by ⟨general name⟩ in the query,

$A(X)$ - formula obtained from the translated query, which describes some properties of X.

/3/ Problem queries of general form in natural language

/a/ Why ⟨clause⟩ ?

which can be transformed into the form:

Which Z satisfies: $Z \Rightarrow$ ⟨clause⟩ ?

/b/ What is implied by ⟨clause⟩ ?

which can be transformed into the form:

Which Z satisfies: ⟨clause⟩ $\Rightarrow Z$?

In the above problem queries: Z - the clause to be found, ⟨clause⟩ - clause obtained from the translated query. Problem queries require an answer in the form of a sentence.

An analysis of a user's query should fix the main subject of it in the terms of a subset of concepts,represented on the concept graph.

A definition of acceptable language of user's queries involves the form of translation rules from natural language into the predicate calculus formula. It is worth noticing that:

- queries in the natural language form have the threefold nature, it means they can be counted into the three mentioned above classes,
- queries fragments in the form of indicative clauses are built from expressed in natural language predicates,introduced in the presented formalization,
- in respect of quantity of expressions the language of user's queries is comparable with the language obtained in the process of translation of predicate calculus axioms into natural language,
- language of user's queries and the predicate calculus language have a common base of basic concepts because the sorts of objects, functions and predicates introduced in the predicate calculus language correspond with some specified natural language expressions.

It has been submitted that question-answering problem may be solved with an application of automated theorem-proving techniques, namely on the base of the resolution principle [8].This method, based only on the syntax of clauses, doesn't require to control the proof procedure by the user.The resolution principle requires to convert all formulas into the Skolem conjunctive form. Thus each formula becomes a set of clauses, each of them being a disjunction of literals.

The question-answering procedure for decision queries tries to prove that a negation of formula obtained after translation of the query, is false. If it's so, an answer is "Yes". If a proof procedure applied to the formula in its affirmative form provides a success, an answer is "No".Some questions may be unsolvable in the lack of knowledge.

The proof procedure for objective queries examines a formula

$$\sim \exists X (W(X) \Rightarrow A(X))$$

which is supposed to be false. The proof procedure tries to retrieve a counter-example, if it exists, which will be substituted in the place of X.

It has been assumed that problem queries are in the implication form after the translation process. Question-answering procedure for this class of queries has been reduced to such one, which tries to retrieve an answer from one axiom. For the first subclass of problem queries a search is made for an axiom in the implication form, which conclusion embodies the conclusion of the formula obtained from the transformed query. The premises of this axiom are an answer. The proof procedure applied to answer a question of the second subclass tries to find an axiom, which premises are implied by premises of the formula obtained from the transformed query. The conclusion of this axiom is an answer.

The proper way to reduce a number of clauses taking a part in the resolution process is to construct an initial active set of clauses as a set containing only clauses of axioms concerning concepts recognized in the query and clauses of formula obtained from the query. The translation rules, applied to transform axioms from the predicate calculus language into the natural language expressions, can be used also to translate the retrieved answer to the natural language.

CONCLUSIONS

Described above knowledge representatiom method based on predicate calculus has been applied to the large subset of FORTRAN 1900 [1]. It has been shown that this method satisfies criteria required in the ICAI system. An application of the predicate calculus language to describe knowledge about the programming language provides the automated answering of student's questions, which is the main advantage of this method.

This method is applicable to those domains of knowledge, which can be represented by the set of first order logic axioms, with regard to their formalized nature.

REFERENCES

[1] Begier B., Method of representation of subject matter in the computer system to teach programming language, Ph.D. Thesis /in Polish/, Reg. Comp. Center, Tech. Univ. of Poznań /1980/.

[2] Burstall R.M., Formal description of program structure and semantics in first order logic, Mach. Intell. 5 /1969/ 79-98

[3] Carbonell J.R., AI in CAI: an artificial intelligence approach to computer-assisted instruction, IEEE Trans. on Man-Machine Systems 11 /1970/ 190-202

[4] Clancey W.J., Bennet J.S., Cohen J.R., Applications-oriented AI research: education, Rep. STAN-CS-79-749, Dep. of Comp.Sc., Stanford Univ. /July 1979/

[5] Computer-aided instruction system to teach the programming language FORTRAN, Rep. of Reg. Comp. Center, Tech.Univ.Poznań /1980/

[6] Goldstein I.P., The genetic graph: a representation for the evolution of procedural knowledge, Int.J.Man-Machine Studies 11 /1979/ 51-77

[7] Laubsch J.H., Some thought about representing knowledge in instructional systems, Fourth Int. Joint Conf. on Art. Intell., Tbilisi /1975/ 122-125

[8] Loveland D.W., Automated theorem proving: a logical basis /North Holland, Amsterdam, 1978/