AN OPERATIONAL ALGEBRAIC SEMANTICS OF PROLOG PROGRAMS

DERANSART Pierre

INRIA

Domaine de Voluceau - Rocquencourt

BP 105

78153 Le Chesnay

April 1983

Abstract : We shall show that the resolution strategy implemented in most of the PROLOG interpretors may be equivalently viewed as a particular equation solving in an associated algebraic specification. We suggest and illustrate possible applications of this approach to analysis of PROLOG programs.

Keywords : PROLOG - ALGEBRAIC ABSTRACT DATA TYPES - PROGRAMMING ENVIRONMENT.

PRESENTATION (Long abstract)

We shall show that the resolution strategy implemented in most of the PROLOG interpretors may be equivalently viewed as a particular equation solving in an associated algebraic specification. We suggest and illustrate possible applications of this approach to analysis of PROLOG programs.

The basic point of this work is a rigorous correspondence between a PROLOG program and his translation -if any- into an algebraic specification. Most of the studies about PROLOG semantics [Vako76] are devoted to the "pure PROLOG", i.e. PROLOG (restricted to first order logic programming) without "control" nor evaluable predicates. By "control" we mean two things : the famous "cut" operator and the strategy of choosing the clauses and the litterals to be solved. Our aim is to integrate the second element into the semantics in order to get a kind of operational semantics taking in account this aspect of the control.

In fact, the logical part of a PROLOG program get rise, in numerous programs, to a rapid understanding and easy verification of the program properties, analogous to partial correctness proof of programs [ClTa77]. But halting problems or invertibility aspects give unexpected and sometimes difficult to manage behaviours, even of simple programs. A programmer is not only interested to know if his goal is a logical consequence of the axioms, but essentially interested to know how his goal will be satisfied, if there is no infinitely nested loop or if he will obtain all the solutions (the completness in this sense has to be defined), in which order, etc... . Lot of these questions have an empirical answer, without any aid of known semantic models.

On the other side, algebraic specifications have been extensively studied with practical (operational) and semantical points of view [ADJ78, GH78]. Some specifications can be viewed as equational theories. In our approach, specifications are only viewed as a practical way to describe environments and programs in the same formalism and are limited to so-called "specification with constructors" similar to equational theories with constructors of [HH80] but with conditional axioms.

This work should have various applications. Behaviour studies of PROLOG programs or equivalent program transformations are part of them. Some examples of non trivial programs have been studied by this method, like permutations, eight-queens problem and Baxter example [Ba81]. Practical limitations of this approach come from the type of conditional axioms which can be easily studied.

Each time the specification is a canonical and complete TRS the situation is quite agreable : it is in fact possible to use directly properties of the specification in order to transform or modify the programs.

In the general case of equalitarian axioms, the main difficulties seem to come out from the few existing works on such axiomatisation and the equalitarian TRS that can be defined on. Some constraints can be given such that numerous interesting programs fall down in this class, but the practical study of derivations remains difficult. It seems to us that a usefull tool could be a PROLOG programming environment in which narrowing of transformed goals could be formally analyzed. Nevertheless, difficulties come from two levels :

1) Semantical level : in all the cases, the obtained specification is a partial algebra, because of the manipulation of partial functions.

2) Operational level : the generalized TRS did not have been enough studied until now [Re82, Ka83]. The corresponding notions of canonical and complete TRS remain to be better known.

It seems to us that these difficulties reflect well the situation we feel in PROLOG programming : difficulties to specify the error cases in a satisfactory manner (frequently only positive cases are specified), quasi-impossibilities to have a clear idea of the set of produced solutions, his completness, except by personal conviction of the programmer.

Finally, our study can be viewed from a dual point of view :

- Conversion of an abstract data type into a PROLOG program. So it is a way to get a direct and efficient implementation of the transitive

Various papers are dealing with correspondence between specifications or functional programming and PROLOG [VaMa81, BD81]. Generally the correspondence shows that PROLOG is a suitable specification approach. But the correspondence is not always very precisely stated.

We will define a strict correspondence by the following manner :

- To any predicate we associate a functional decomposition. A predicate of arity n is said 1-decomposable, iff there exists an equivalent function of arity n-1 with corresponding domains. This notion can be generalized into k-decomposability.

- To any PROLOG program that can have a functional decomposition, it is possible to associate a specification with constructors. If there is no functional decomposition, the transformation is trivial and of few interest. In all the cases the transformation is a one to one correspondence.

- We show that the resolution of a PROLOG goal, using the usual interpretor strategy, is exactly the same as to solve an equation (the transformed goal) using a strategy called ℓ-i-resolution. If the specification is an equational theory, this problem reduces to an unification problem solved by ℓ-i-resolution (this approach uses a relation called "narrowing").

- Finally we use this transformation in order to study the solutions of the goal equations, in particular the capacity of invertibility of a program.

This approach gives an operational characterization of PROLOG programs admitting such an analysis (functional decomposition plus specification with particular properties). The approach is completely symetric and the obtained class is not restrictive : it has the power of computable functions. So it is possible to have dual point of view : in one sense PROLOG realizes an operational implementation of conditional algebraic specifications, on the other the models of the specification can be models of the PROLOG program.

closure of the ℓ-i-narrowing. In this case we shall speak of "compilation of specifications into a PROLOG program".

- Conversion of a PROLOG program into an abstract data type. This is a way to verify the original program structure (by typing the elements, verifying completness...) and, eventually, to modify it using correct transformations.

BIBLIOGRAPHY

[ADJ78] GOGUEN J.A., THATCHER J.W., WAGNER E.G.
An Initial Algebra Approach to the Specification Correctness and Implementation of Abstract Data Types in "Current Trends in Programming Methodology".
Chap. IV (R. Yeh, ed) - pp. 80-149 - Prenctice Ha-1 1978.

[Ba81] BAXTER L.
The Versatility of PROLOG.
SIGPLAN Notices - York University.

[BD81] BERGMAN M., DERANSART P.
Abstract Data Type and Rewriting System : Application to the Programming of Algebraic Abstract Data Types in PROLOG.
CAAP 81 - Trees in Algebra and Programming - 6th Colloquium - March 81 - LNCS 112.

[ClTa77] CLARK K., TARNLUND S.A.
A First Order Theory of Data and Programs.
Proc. IFIP 77 - pp. 939-944.

[GH78] GUTTAG J.V., HORNING J.J.
The Algebraic Specification of Abstract Data Types.
Acta Informatica 10 (1978) - pp. 27-52.

[HH80] HUET G., HULLOT J.M.
Proofs by Induction in Equational Theories with constructors.
Rapport INRIA n° 28.

[HO80] HUET G., OPPEN D.
Equations and Rewrite Rules : a Survey.
"Formal Languages : Perspectives and Open Problems".
Ed. Book R. - Academic Press 1980 - Also TR-CSL-111.
SRI International - January 1980.

[Ka83] KAPLAN S.
An Abstract Data Type Specification Language.
(French) Thesis - University of Orsay - February 3, 1983.

[Re82] REMY J.L.
Conditional Rewrite Rules Systems and Applications to Algebraic Data Types.
(French) Doctorat Thesis - University of Nancy - July 13, 1982.

[VaKo76] VAN EMDEN M.H., KOWALSKI R.
The Semantics of Predicate Logic as Programming Language.
JACM 23 - 1976 - pp. 733-742.

[VaMa81] VAN EMDEN M.H., MAIBAUM T.S.E.
Equations compared with Clauses for Specification of Abstract Data Types.
Advance in Data Base Theory - V1 - Ed. Gallaire, Minker, Nicolas - Plenum Press. - 1981.