STUDIA UNIV. BABEŞ-BOLYAI, MATHEMATICA, XXXIX, 3, 1994

LOGICAL GRAMMARS AND UNFOLD TRANSFORMATION OF LOGIC PROGRAMS

Doina TATAR[*]

Dedicated to Professor Sever Groze on his 65th anniversary

Received: June 28, 1994
AMS subject classification: 68Q50, 68Q55, 68T27

REZUMAT. - Gramatici logice şi transformări "unfold" ale programelor logice. În articol este utilizată noţiunea anterior definită în [12] de "gramatică logică" pentru a demonstra faptul că transformările "unfold" ale programelor logice conduc la programe care sunt echivalente cu cele originale.

Abstract. The previously defined notion of "logical grammar" [12] is utilized to demonstrate that unfold transformations of logic programs produce programs which are equivalent to the original one.

1. Introduction. The two principal papers which we are based in this note are:" Logical grammars as a tool for studying logic programming" [12] (where we propose a new semantic for operational behavior of logic programs, which is based on the framework of formal languages), and "Unfold /fold transformations of logic programs" by J.C.Shepherdson [10]. In a way, this paper is the first certificate (besides those from [12]) that this new approach for semantic of logic programs is better matching for some connected discussions.
We start by looking at the unfold transformations : the origin of those goes back to Burstall and Darlington,who introduced them in the context of recursive programs. In [10] the definition of unfolding for a logic program is:

[*] *"Babeş-Bolyai" University, Faculty of Mathematics and Computer Science, 3400 Cluj-Napoca, Romania*

Definition 1.1 Let P be a logic program and

$$C : A \leftarrow p(t), S$$

be a clause of program P (where t stands for a tuple of terms) and

$$D_1 : p(t_1) \leftarrow S_1$$

$$D_r : p(t_r) \leftarrow S_r$$

be all the clauses of P whose heads $p(t_1),\ldots,p(t_r)$ unify with p(t) with mgu $\theta_1,\ldots,\theta_r$. Then the result of unfolding C w.r.t. p(t) is the program P' obtained by replacing C by the r clauses

$$C_1 : A\theta_1 \leftarrow S_1\theta_1, S\theta_1$$

$$C_r : A\theta_r \leftarrow S_r\theta_r, S\theta_r$$

Here A is an atom, p is a symbol of predicate and S is the rest of a clause.

In [8] is obtained the result:

Theorem If P' is the logic program obtained from P by unfolding, then a goal G success with the answer substitution θ from P' iff the goal G success with the answer substitution θ from P.

In the next section we will present the logical grammars [12] and will prove that the unfold transformation preserve the meaning introduced by those.

2. Logical grammars The language considered here is essentially that of first-order predicate logic without function symbols. Let:

- P be a set of predicates.
- C be a set of constants.
- V be a set of variables.

An atom over C ∪ V is of the form

$$p(u_1,\ldots,u_n),\ n \geq 0$$

where $p \in P$ with arity n, and each u_j is an element of $ C ∪ V.

If the arguments u_j are not interesting in a particular context, then we will denote an atom simply by p. Let A be the set of atoms over C ∪ V, and, if

$P' \subseteq P$, let $A_{P'}$ be the set of atoms with predicate symbols from P'. In some recent papers [13], the set of predicates is considered as divided into two disjoint sets:the set EDB of extensional predicates (or extensional databases predicate) which represent basic "facts," and the set IDB(of intensional databases predicates) representing facts deduced from the basic facts via the logic program. Particularly ,the set EDB can be the empty set (as,for simplicity,in most of the following demonstrations).

Definition 2.1 A logic program P is a sequence of Horn clauses, that is, clauses of the form:

$$p \leftarrow q_1,\dots,q_n$$

where p and $q_1,\dots,q_n$ are atomic formulas in first-order logic, the comma is the logic operation "and", and the sign $\leftarrow$ is "if" or reverse of the logical implication.

We refer to the left (p) and right-hand side ($q_1,\dots,q_n$) of a clause as its head and body. A clause is logically interpreted as the universal closure of the implication $q_1 \wedge \dots \wedge q_n \rightarrow p$. If a clause has no right-hand side we will call it a fact or a unit clause. Let us observe that this definition considers only the class of positive logic programs (all atoms in all clauses are positive).

From the properties of IDB and EDB predicates it follows that a predicate from the set EDB cannot occur in the head of clauses, but a predicate from the set IDB can occur in the set of facts.

Definition 2.2 A goal G consists of a conjunction of atoms, such that a successfully terminating computation corresponds to a demonstration of this goal by refutation (SLD-refutation), and is denoted by:

$$\leftarrow r_1,\dots,r_t$$

Over the course of time, we may want to "apply" the same IDB for many quite different EDB's. In this context, and because IDB is the "core" of logic program, the properties of the IDB merit careful study. Recent papers have addressed the problems of studying and optimizing logic programs from various points of view. ([2], [3], [4]). In [12] our tool is the logical grammars:

Definition 2.3 The logical grammar GL associated with a logic program P and the goal G is the system:

$$GL = (I_N, I_T, X_0, F)$$

where:

- $I_N = A_{IDB} \cup \{X_0\}$ is the set of nonterminals.
- $I_T = A_{EDB} \cup \{\lambda\} \cup \{False, True\}$ is the set of terminals.
- X_0 = is the goal G.
- F is a finite set of production rule, of the form:

$$a)\ p \rightarrow q_1 \ldots q_m,\ m \geq 1$$

where $p \in IDB$ and $p \leftarrow q_1, \ldots, q_m$ is a clause in the program P.

or

$$b)\ p \rightarrow \lambda$$

where p is a unit clause in the program P

We assume in the following that substitutions, composition of substitutions, and the most general unifier $\sigma = mgu(g,h)$ of atoms g and h are defined as in logic programming. ([1], [2], [3]).

For a logic grammar GL we define the rewriting relation " $\Rightarrow$ " as follows:

Definition 2.4 If $R \in A^+$ and $Q \in A^*$, then:

$$R \Rightarrow^{\sigma}_{GL} Q$$

if exist an atom $h \in I_N$, and a production rule in F:

$$g \rightarrow h_1 \ldots h_m$$

such that:

$$R = R_1 h R_2, \sigma = mgu(h,g)$$

and

$$Q = \sigma(R_1)\sigma(h_1)\ldots\sigma(h_m)\sigma(R_2)$$

(here the variables of the production rule are renamed to new variables, so that all the variables in the rule do not appear in R).

Let $\Rightarrow *$ denote the reflexive and transitive closure of the relation $\Rightarrow$. If θ is the composition of all substitutions in every direct derivation, let denote it by $\Rightarrow^{\theta} *$.

Definition 2.5 For a logical grammar $GL = (I_N, I_T, X_0, F)$, the generated language is L(GL), where:

$L(GL) = \{(R,\theta) | X_0 \Rightarrow^{\theta} *R, R \in A^*_{IT}, \theta = \theta_1 ... \theta_k$, k is the length of derivation for R, and θ_i is the substitution in the step i$\} \cup \{\Omega\}$.

We have some possibilities for the pair (R, θ):

- if X_0 (or the goal G) is a ground formula, (not containing the variables) then the substitution θ is the empty substitution, and R is True or False, depending on the fact that G is a formula deducible or not from the set of clauses P (by refutation).

- if X_0 contains variables, and the computation is succesfully terminating, in the pairs (R,θ) we have $R \in I^*_T$, and the number of pairs represents the number of solutions. If EDB $= \phi$,then $R = \lambda$. Let denote the last situation
by R= [] ,the empty clause, like usually in logic, and let θ be the answer substitution.

- if the program P is not terminating for the goal G, then $L(GL) = \{\Omega\}$, where $\Omega \notin$ IDB $\cup$ EDB.

For the purpose of simpler manipulation, let us denote the logical grammar GL with the symbol initial X_0 by GL_{X0} and the pairs in the logical language
$L(GL_{X0})$ by the triplets (P,θ,X_0). Let us, furthermore suppose EDB $= \phi$. Then the definition (2.5) becomes:

$$L(GL_{X0}) = \{([],\theta,X_0) | X_0 \Rightarrow^{\theta} * []\} \cup \{\Omega\}$$

In [5] the "succes set " for a logic program P is defined as:

$succ(P) = \{G(t_1,...,t_n | t_1,...,t_n$ are terms in a standard Herbrand interpretation, then they are grounded terms, and the goal $G(t_1,/cdots,t_n)$ is deductible from P$\}$ or, equivalentely,

$succ(P) = \{G(t_1,...,t_n | t_1,...,t_n$ are terms in a standard Herbrand interpretation and there exists a SLD-refutation of $G(t_1,...,t_n)$ from P$\}$

The set succ(P) is not completely adequate as operational semantics since it hides one of the fundamental aspects of logic programming: the ability to compute substitutions. A more adequate definition is [12] the following:

succ'(P) = $\{(G(t_1,\dots,t_n),\theta) \mid t_1,\dots,t_n\}$ are non-ground terms, and $G(t_1,\dots,t_n)$ has a SLD-refutation with computed answer $\theta\}$.

In [12] we proved the connection between the set succ'(P) and the logical grammars.

Definition 2.6 Let U'_H be a nonstandard Herbrand interpretation (which admits non-ground terms). We denote by L_p and we will call them the total la.guage of a logical program P, the following language:

$$L_p = \bigcup_{X_0 \in U'_H} L(GL_{X_0})$$

Conformally with the previously definitions,

$L_p = \{([],\theta,G) \mid G$ is an arbitrary goal, $G \Rightarrow^{\theta *} []\} \cup \{(\Omega,G) \mid$ P is cycling for the goal G$\}$ $\cup \{(False,G) \mid G$ has not a SLD-refutation from P$\}$. In [12] we demonstrated that this language L_p represents a sound and complete semantic for P:

Lema If $G \Rightarrow^{\sigma} G'$ and $(G',\theta) \in$ succ'(P) then $(G,\sigma\theta) \in$ succ'(P).

Theorem (of soundness) Let P be a logic program and G a goal. If $([],\theta,G) \in L_p$ then $(G,\theta) \in$ succ'(P).

Theorem (of completeness) Let P be a logic program and G a goal. If $(G,\theta) \in$ succ'(P), then $([],\theta,G) \in L_p$.

For a logic program P let denote by GL_p the associated logic grammar like in definition (2.5).

We can define two kinds of equivalence of the logic programs:

Definition Two logic programs P_1 and P_2 are strong equivalent if $\forall X_0$ we have $L(GL_{P1}) = L(GL_{P2})$ (or, in the formal languages terminology, if GL_{P1} and GL_{P2} are equivalent for every goal X_0).

The equivalence of two programs in the following is the strong equivalence, therefore the equivalence for same goal.

Definition. Two logic programs P_1 and P_2 are equivalent if $L_{P1} = L_{P2}$ where L_P is defined like in definition 2.6.

The consequence of the introduced notion is the possibility of the definition for some transformations about logic programs, such that the obtained programs are equivalent with the initial programs.

The main result of this section is the following theorem;

Theorem If P' is a logic program obtained from P by unfolding, then P and P' are strong equivalent.

Proof It is enough to prove that, if:

$$Q,R \in A^{*}{}_{l_T \cup l_N}\ ,\ Q \Rightarrow_{GL_p}{}^{*} R$$

then

$$Q \Rightarrow_{GL_{p'}}{}^{*} R$$

where GL_p and $GL_{p'}$ are the logic grammars associated with the programs P and P'. Let observe that ,in accordance with the definition 1.1 and 2.3, in GL_p there exist the production rules:

1. $A \rightarrow p(t)S$
2. $p(t_1) \rightarrow S_1$

$p(t_r) \rightarrow S_r$

and in $GL_{p'}$ these became

3. $A\theta_1 \rightarrow S_1\theta_1 S\theta_1$

$A\theta_r \rightarrow S_r\theta_r S\theta_r$

Also, conformaly with definition 2.4, if $Q \Rightarrow_{GL_p}{}^{*} R$ where a rule 1 followed by 2 is utilized, then $Q \Rightarrow_{GL_{p'}} R$ by a single rule 3. The grammar GL_p has the same generative power like $GL_{p'}$, thus P and P' are strong equivalent.

REFERENCES

1. K.R. Apt, M.H.van Emden: "Contribution to the theory of logic programming", J. of ACM, vol.29, 1982, pg.841-862.
2. K.R. Apt, D. Pedreschi: Studies in pure Prolog:termination, CWI Report CS-R9048, September, 1990.
3. K.R. Apt, D. Pedreschi: Proving termination of general Prolog programs, CWI Report CS-R9111, February, 1991.
4. S.Debray, P.Mishra:"Denotational and operational semantics for PROLOG", The Journal of Logic Programming, vol.5, nr.1, 1988, pp.33-61.
5. M.Falaschi, G.Levi, M.Martelli, G.Palamidessi:"Declarative modelling of the operational behaviour of logic languages", Raport Univ.di Pisa,TR-10/1980.I.Guessarian: Some fixpoint techniques in algebraic structures
6. P.A.Gardner, J.C.Shepherdson:"Unfold/fold transformation of LP", Festschrift in Honour of Alan Robinson, Oxford University Press,London, 1990.
7. C.J. Hogger: "Derivation of Logic Programs", Journal of ACM, April, 1981, pp. 372-393.
8. T.Przymusinski:"On the declarative and procedural semantics of logic programs" J.of Automated Reasoning, vol5, 1989, pp.167-205.
9. I. Shioya: "Logic hypergraph grammars and context-free hypergraph grammars". Systems and Computers, vol.22, nr.7, 1991.
10. J.C.Shepherdson:"Unfold/fold transformations of LP ", Mathem. Struct.in Computer Science(1992), vol.2, pp.143-157.
11. D.Tatar:" Utilizarea gramaticii sintactice asociata unei scheme de recursive in studiul programelor", Studii şi cercetari matematice, nr.4, 1988, pp.337-347.
12. D.Tatar:"Logical grammars as a tool for studying logic programming", Studia Univ."Babes-Bolyai", Mathematica, 1993(to appear).
13. A. van Gelder, K.A.Ross, J.S.Schlipf:"The Well-Founded Semantics for General Logic Programs", Journal of ACM, July, 1991, pp. 620-651.
14. M.H.van Emden, R.A.Kowalski:"The semantics of predicate logic", J.of ACM, oct.1976, pp.733-742.