PROCEEDINGS OF THE YEREVAN STATE UNIVERSITY

Physical and Mathematical Sciences 2011, № 3, p. 40–46

Informatics

ON *SLDNF*-RESOLUTION IN LOGIC PROGRAMMING WITH NEGATION

L. A. SARGSYAN*

Chair of Programming and Information Technologies YSU, Armenia

The paper is devoted to the logic programming with negation and with built-in predicates. General logic programs (logic programs with negation) and general goals (goals with negation) are considered. Modification of *SLDNF*-resolution for built-in predicates is introduced. The soundness of modified *SLDNF*-resolution is proved. *SLDNF*-resolution used in real systems (practical *SLDNF*-resolution) is considered and the soundness of practical *SLDNF*-resolution is proved.

Keywords: logic programming, negation, built-in predicates, *SLDNF*-resolution, soundness, practical *SLDNF*-resolution.

Introduction. The foundations of Horn programming and *SLD*-resolution with built-in predicates are investigated in [1]. Logic programs described in [1] lack sufficient expressiveness for many situations. The problem is that often a negative literal is needed in the body of program clause. It is important to extend the definition of programs to include negative literals in the clause bodies. Logic programming systems also require a mechanism for handling negative subgoals. The method usually chosen is *SLDNF*-resolution, which is the augment of *SLD*-resolution with the negation as failure rule [2]. In [2] *SLDNF*-resolution is investigated in pure logic programming. However, real logic programming systems use both negation and built-in predicates. In this paper the *SLDNF*-resolution in logic programming with negation and with built-in predicates is introduced. In order to justify the use of negation as failure rule the completion of program is modified. The soundness of *SLDNF*-resolution is shown for this case. *SLDNF*-resolution rules used in real systems, i.e. practical *SLDNF*-resolution, is considered and the soundness of practical *SLDNF*-resolution is proved.

Notation and Background. Consider three nonintersecting sets Φ, Π and X. Φ is a set of functional symbols each possessing an arity. For any $n \geq 0$, Φ contains a countable number of symbols of arity n. X is a countable set of variables. Terms are composed of elements of sets Φ and X. By $Var(t)$ we denote the set of all variables involving in the term t. The set of all terms with no variables is denoted by M.

We assume also that $\Pi=\Pi_1\cup\Pi_2$, where Π_1 is the set of predicate symbols, Π_2 is the set of built-in predicate symbols, each k-ary ($k > 0$) built-in predicate is

* E-mail: lusine.sargsyan@ysu.am

being a calculable mapping $M^k \to \{true, false\}$. The atoms are defined as usual [2]. A formula of the first-order predicate logic with equality over logical operations $\neg$, &, $\vee$, $\supset$, ~ and quantifiers $\exists$ and $\forall$ is defined conventionally [2]. A predicate term is an atom, which uses predicate symbol from Π_1. A literal is a predicate term or the negation of a predicate term. A ground literal is a literal with no variables. A condition is an atom or the negation of an atom, which uses predicate symbol from Π_2.

A substitution σ is a set $\{t_1/x_1,...,t_n/x_n\}$, where t_i is a term, x_i is a variable, $t_i \neq x_i$, $i \neq j \Rightarrow x_i \neq x_j$, $i,j=1,...,n$, $n \geq 0$. The following denotations are introduced: $\mathrm{Arg}(\sigma) = \{x_1,...,x_n\}$, $Var(\sigma) = Var(t_1) \cup \ldots \cup Var(t_n)$. The composition of substitutions is defined traditionally [2].

Atoms (or terms) A_1 and A_2 are said to be unified, if there exists a substitution δ such that $A_1\delta = A_2\delta$. The substitution δ is called the unifier of A_1 and A_2. A unifier σ is called the most general unifier of A_1 and A_2 ($\sigma = mgu(A_1, A_2)$), if for any unifier δ there exists a substitution γ such that $\sigma\gamma = \delta$.

In this work the set of Herbrand interpretations [1], denoted by H, are considered.

The value of a closed formula F on the interpretation I is defined in the natural way and denoted by $Val_I(F)$. The formula F is termed identically true, if F takes the value *true* on any interpretation from H. If F and F' are closed formulae and the formula $F \supset F'$ is identically true, we will say that F' is a logical consequence of F and denote this fact by $F \models F'$.

A general logic program (or simply program) P is a sequence $S_1,\ldots,S_n$ of clauses, $n > 0$. A clause $S \in \{S_1,\ldots,S_n\}$ has the form A:-$K_1,\ldots,K_m$, where A is a predicate term, each $K \in \{K_1,\ldots,K_m\}$ is a literal or a condition, $m \geq 0$. Atom A is called the head of the clause S, the sequence $K_1,\ldots,K_m$ is called the body of S. If $m = 0$, S is termed a fact, else S is termed a rule. With the program P we associate the formula $comp(P)$:

$$F(p_1) \& \ \ldots \ \& F(p_u),$$

where $p_1,\ldots,p_u$ are the predicate symbols from program P, $p_i \in \Pi_1$, $i = 1,\ldots,u$, $u \geq 1$, and every $F(p)$, where $p \in \{p_1,\ldots,p_u\}$, is defined in the following way:

If p is a 0-ary predicate symbol and p is a fact of program P, then $F(p)$ is p, else if p does not appear in the head of any clause of P, then $F(p)$ is $\neg p$, else if the definition of p is p:-$B_1,\ldots,p$:-B_v, where B_i is the body of the clause p:-B_i, $i=1,\ldots,v$, $v \geq 1$, then $F(p)$ is $p \sim E_1 \vee \ldots \vee E_v$, where E_i is $\exists y_1 \ldots \exists y_d(K_1 \& \ \ldots \ \& K_m)$, $y_1,\ldots,y_d$ ($d \geq 0$) are the variables of the rule p:-B_i, and B_i is $K_1,\ldots,K_m$, $m \geq 1$, $i = 1,\ldots,v$.

If p is an n-ary ($n > 0$) predicate symbol and p does not appear in the head of any P program clause, then $F(p)$ is $\forall x_1 \ldots \forall x_n \neg p(x_1,\ldots,x_n)$, else if the definition of p is A_1:-$B_1,\ldots,A_v$:-B_v, where B_i is the body of the clause A_i:-B_i, $i = 1,\ldots,v$, $v \geq 1$, then $F(p)$ is $\forall x_1 \ldots \forall x_n(p(x_1,\ldots,x_n) \sim E_1 \vee \ldots \vee E_v)$, where $x_1,\ldots,x_n$ are variables not appearing in the clauses A_1:-$B_1,\ldots$, A_v:-B_v, each E_i has a form $\exists y_1 \ldots \exists y_d(K_1 \& \ldots \& K_m)$, $y_1,\ldots,y_d$ ($d \geq 0$) are the variables of the rule A_i:-B_i, and B_i is $K_1,\ldots,K_m$, $m \geq 0$, $i = 1,\ldots,v$.

A general goal (or simply goal) Q has the form ?-$L_1,\ldots,L_n$, where L_i is a literal or a condition, $i = 1,\ldots,n$, $n \geq 0$; number n is called the length of the goal Q.

The goal Q is identified with the formula $\exists y_1 \ldots \exists y_s(L_1 \& \ldots \& L_n)$, where $y_1,\ldots,y_s$ are the variables involved in the $L_1,\ldots,L_n$, $n \geq 1$, $s \geq 0$. We denote the set $\{y_1,\ldots,y_s\}$ by $Var(Q)$. If $n = 0$, Q is called an empty goal.

Let us describe the set of answers to a nonempty goal Q to a program P in accordance with logical semantics and denote it by $Log(P,Q)$.

If $comp(P)| \neq \neg Q$ and $comp(P)| \neq Q$, then $Log(P,Q) = \varnothing$;

if $comp(P)| = \neg Q$, then $Log(P,Q) = \{no\}$;

if $comp(P)| = Q$ and $Var(Q) = \varnothing$, then $Log(P,Q) = \{yes\}$;

if $comp(P)| = Q$ and $Var(Q) = \{y_1,\ldots,y_r\}$, $r>0$, then $Log(P,Q)$ consists of all such collections of terms $<t_1,\ldots,t_r> \in M^r$, for which $P| = Q\theta$, where $\theta = \{t_1/y_1, \ldots, t_r/y_r\}$.

***SLDNF*-Resolution in Logic Programming with Negation for Built-in Predicates. Procedural Semantics.** Now we describe the set of *SLDNF*-resolution rules (computation rules). Each computation rule ρ is defined via functions Sel_ρ and Sub_ρ. Let Q be a goal $?\text{-}L_1,\ldots,L_n$, $n \geq 1$, $Sel_\rho(Q) \in \{1,\ldots,n\}$. Let $Sel_\rho(Q) = j$ $(1 \leq j \leq n)$. If L_j is a literal, then $Sub_\rho(Q)$ is undefined. Let L_j be a condition. So, $Sub_\rho(Q)$ is a set of substitutions and for any $\sigma \in Sub_\rho(Q)$ the following conditions are satisfied:

1. $Arg(\sigma) \subset Var(L_j)$,
2. $Var(\sigma) \cap (Var(Q) \backslash Var(L_j)) = \varnothing$,
3. $Val(L_j\sigma\gamma) = true$ for any substitution γ such that $Var(L_j\sigma\gamma) = \varnothing$, and for any substitution δ, where $Var(L_j\delta) = \varnothing$ and $Val(L_j\delta) = true$, there exists $\sigma \in Sub_\rho(Q)$ and γ, so that $L_j\delta = L_j\sigma\gamma$.

We specify the class of safe computation rules and denote it by R. A computation rule ρ is safe, if having selected a ground negative literal $\neg A$ in some goal, ρ attempts to finish the construction of a finitely failed *SLDNF*-tree with root $?\text{-}A$ before continuing with the remainder of the computation. If ρ selects a nonground negative literal, then ρ rejects the goal.

Let P be a program, Q be a nonempty goal and $\rho \in R$. Then the *SLDNF*-tree for (P,Q) via ρ is defined as follows:

1. Each node of the tree is a goal.
2. The root node is Q.

Let $?\text{-}L_1,\ldots,L_n$ $(n \geq 1)$ be a node of the tree and $Sel_\rho(?\text{-}L_1,\ldots,L_n) = j$ $(1 \leq j \leq n)$.

3. Let L_j be a positive literal, then this node has a descendant for each clause $A\text{:-}K_1,\ldots,K_m$ $(m \geq 0)$ from program P, such that L_j and A are unifiable. The descendant is $?\text{-}L_1\sigma,\ldots,L_{j-1}\sigma,K_1\sigma,\ldots,K_m\sigma,L_{j+1}\sigma,\ldots,L_n\sigma$, where $\sigma = mgu(L_j,A)$.

4. Let L_j be a ground negative literal. If the subgoal L_j is successful, the single descendant of the node is $?\text{-}L_1,\ldots,L_{j-1},L_{j+1},\ldots,L_n$. If the subgoal L_j fails, the node has no descendants.

5. Let L_j be a condition. If $Sub_\rho(Q_i) \neq \varnothing$, then this node has a descendant for each $\delta \in Sub_\rho(Q_i)$. The descendant is $?\text{-}L_1\delta,\ldots,L_{j-1}\delta,L_{j+1}\delta,\ldots,L_n\delta$. If $Sub_\rho(Q_i) = \varnothing$, then this node has no descendants.

6. Nodes which are the empty goal have no descendants.

Let P be a program, Q be a nonempty goal and $\rho \in R$. If *SLDNF*-tree for (P,Q) via ρ is finite, contains no branches, which end in the empty goal, and contains at least one derivation of rejected goal, then the goal Q is also

rejected by ρ. A finitely failed *SLDNF*-tree for (P,Q) via ρ is one which is finite, contains no branches, which end in the empty goal and contains no rejected nodes.

Let P be a program, Q be a nonempty goal and $\rho \in R$. An *SLDNF*-derivation $Q_1,Q_2,\ldots$ of (P,Q) via ρ, where $Q_1 = Q$ is defined as follows:

Suppose Q_i $(i > 0)$ is $?\text{-}L_1,\ldots,L_n$ $(n \geq 1)$ and $Sel_\rho(?\text{-}L_1,\ldots,L_n) = j$ $(1 \leq j \leq n)$.

If L_j is a positive literal, $A\text{:-}K_1,\ldots,K_m$ $(m \geq 0)$ is a clause from P and $\sigma = mgu(L_j, A)$, then the derived goal Q_{i+1} is $?\text{-}L_1\sigma,\ldots,L_{j-1}\sigma,K_1\sigma,\ldots,K_m\sigma,L_{j+1}\sigma,\ldots,L_n\sigma$.

If L_j is a ground negative literal $\neg A$, an attempt is made to construct an *SLDNF*-tree with $?\text{-}A$ at the root. If the goal $?\text{-}A$ succeeds, the subgoal $\neg A$ fails and so the goal Q_i also fails. If A fails finitely, the subgoal $\neg A$ succeeds and the derived goal Q_{i+1} is $?\text{-}L_1,\ldots,L_{j-1}, L_{j+1},\ldots,L_n$.

If L_j is a condition, $Sub_\rho(Q_i) \neq \varnothing$, $\delta \in Sub_\rho(Q_i)$, then the derived goal Q_{i+1} is $?\text{-}L_1\delta,\ldots,L_{j-1}\delta,L_{j+1}\delta,\ldots,L_n\delta$.

Now we describe the set of answers to a nonempty goal Q to a program P corresponding to the procedural semantics based on a computation rule $\rho \in R$ and denote this set by $Proc_\rho(P,Q)$.

If the goal Q is rejected by the rule ρ, then $Proc_\rho(P,Q) = \varnothing$;

if (P,Q) has a finitely failed *SLDNF*-tree for (P,Q) via ρ, then $Proc_\rho(P,Q) = \{no\}$;

if $(P,Q)\vdash_\rho ?\text{-}$ and $Var(Q) = \varnothing$, then $Proc_\rho(P,Q) = \{yes\}$;

if $(P,Q)\vdash_\rho ?\text{-}$ and $Var(Q) = \{y_1,\ldots,y_r\}$, $r > 0$, then $Proc_\rho(P,Q)$ consists of all such collections of terms $\langle t_1, \ldots, t_r\rangle \in M^r$ that there exist an *SLDNF*-derivation of (P,Q) via ρ, which end in the empty goal and such substitution δ that $\{t_1/y_1,\ldots,t_r/y_r\} \subset \sigma_1\ldots\sigma_{k-1}\delta$, where σ_i is the substitution corresponding to the application of the rule ρ that results the goal Q_{i+1}, $i = 1,\ldots,k–1$, $k > 1$.

Soundness of *SLDNF*-Resolution. Let us state three Lemmas omitting their proofs.

Lemma 1. Let P be a program, Q be a nonempty goal $?\text{-}L_1,\ldots,L_n$, $\rho \in R$, $Sel_\rho(Q) = i$ $(1 \leq i \leq n)$ and L_i be a predicate term or a condition. If there are no derived goals of (P,Q) via one application of the rule ρ, then $comp(P)\models\neg Q$.

Lemma 2. Let P be a program, Q be a nonempty goal $?\text{-}L_1,\ldots,L_n$, $\rho \in R$, $Sel_\rho(Q) = i$ $(1 \leq i \leq n)$, L_i be a predicate term and $\{Q_1,\ldots,Q_r\}$ $(r > 0)$ be the set of all derived goals of (P,Q) via one application of the rule ρ, $Q_j \neq ?\text{-}$ $(1 \leq j \leq r)$. Then $comp(P)| = Q \sim Q_1 \vee \ldots \vee Q_r$.

Lemma 3. Let P be a program, Q be a nonempty goal $?\text{-}L_1,\ldots,L_n$, $\rho \in R$, $Sel_\rho(Q) = i$ $(1 \leq i \leq n)$, L_i be a condition and D is the nonempty set of nonempty derived goals of (P,Q) via one application of the rule ρ. Then for any interpretation I

$Val_I(Q) = true \Leftrightarrow$ there exists $Q' \in D$ such that $Val_I(Q') = true$.

Theorem 1. Let P be a program, Q be a nonempty goal and $\rho \in R$. Then, if (P,Q) has a finitely failed *SLDNF*-tree via ρ, then $comp(P) \models \neg Q$.

Proof. We use the mathematical induction on the number m of negative subgoals selected during the construction of the finitely failed *SLDNF*-tree (including construction of subsidiary trees). Suppose first that $m = 0$. Then the

result follows by a straightforward induction on the depth of the tree, using Lemmas 1, 2 and 3.

Assume the hypothesis holds for $k < m$ ($m > 0$). Consider a finitely failed *SLDNF*-tree for (P,Q), construction of which requires the selection of m negative literals. There is at least one goal, selected literal of which is negative, appearing on the main tree. Consider a goal G of the least depth h in this tree, selected literal of which is negative. Let G be $?\text{-}L_1,...,L_n$, $Sel_\rho(G) = i$ ($1 \le i \le n$, $n > 0$) and L_i is $\neg A$, where A is ground predicate term. There are two cases to consider.

a) Goal $?\text{-}A$ fails.

Thus, subgoal $\neg A$ is deleted from G. The derived goal G' has the form $?\text{-}L_1,...,L_{i-1},L_{i+1},\ldots,L_n$. By the induction hypothesis, $comp(P)| = \neg G'$. $\neg G'$ has the following form: $\neg\exists(L_1\&...\&L_{i-1}\&L_{i+1}\&...\&L_n) \approx \forall(\neg L_1\vee\ldots\vee\neg L_{i-1}\vee\neg L_{i+1}\vee\ldots\vee\neg L_n)$, consequently $comp(P)| = \forall(\neg L_1\vee\ldots\vee\neg L_{i-1}\vee\neg(\neg A)\vee\neg L_{i+1}\vee\ldots\vee\neg L_n)$. It is obvious, that the formula $\forall(\neg L_1\vee\ldots\vee\neg L_{i-1}\vee\neg(\neg A)\vee\neg L_{i+1}\vee\ldots\vee\neg L_n)$ is $\neg G$, therefore, $comp(P)| = \neg G$.

b) $(P,?\text{-}A)$ has an *SLDNF*-derivation, which ends in the empty goal.

If for certain nonempty goal Q', we have $(P,Q')|\!\!-\!\!\!-_\rho ?\text{-}$ and the number of negative subgoals selected during the derivation is less than m, then by induction on the derivation length and using induction hypothesis, Lemmas 2, 3, we obtain that $comp(P)| = Q'$, therefore, $comp(P)| = A$. Since $\neg G'$ is the formula $\forall(\neg L_1\vee\ldots\vee\neg L_{i-1}\vee A\vee\neg L_{i+1}\vee\ldots\vee\neg L_n)$ and $comp(P)| = A$, then it is evident that $comp(P)| = \forall(\neg L_1\vee\ldots\vee\neg L_{i-1}\vee A\vee\neg L_{i+1}\vee\ldots\vee\neg L_n)$, i.e. $comp(P)| = \neg G'$.

Let $\{Q_1,\ldots,Q_r\}$ be the set of all goals of depth h, $r > 0$. We have $comp(P)| = \neg Q_i$ for any $i = 1,\ldots,r$. Using the fact, that ρ selects no negative literals to the depth h and Lemmas 1, 2, 3, we obtain that $comp(P)| = Q \sim Q_1\vee\ldots\vee Q_r$. Consequently, $comp(P)| = \neg Q$. □

Theorem 2. Let P be a program, Q be a goal $?\text{-}L_1,\ldots,L_n$ ($n \ge 1$) and $\rho \in R$. Then, if (P,Q) has an *SLDNF*-derivation, which ends in the empty goal, $\sigma_1,\ldots,\sigma_s$ ($s > 0$) is the sequence of substitutions using in this derivation, then $comp(P)| = \forall y_1\ldots\forall y_m((L_1\&\ldots\&L_n)\sigma_1\ldots\sigma_s)$, where $y_1,\ldots,y_m$ ($m\ge 0$) are the variables appearing in $(L_1\&\ldots\&L_n)\sigma_1\ldots\sigma_s$.

Proof. We prove by induction on the length s of *SLDNF*-derivation.

Suppose first that $s = 2$. This means that Q has the form $?\text{-}L_1$. We consider 3 cases.

a) L_1 is a predicate term. Thus, P has a fact A and $L_1\sigma_1 = A\sigma_1$, where $\sigma_1 = mgu(L_1,A)$. Since $comp(P)| = \forall(A\sigma_1)$, then $comp(P)| = \forall(L_1\sigma_1)$.

b) L_1 is a ground negative literal of the form $\neg A$, and (P, A) has a finitely failed *SLDNF*-tree via ρ. Theorem 1 shows that $comp(P)| = \neg A$, i.e. $comp(P)| = L_1$ and σ_1 is an empty substitution.

c) L_1 is a condition. Then $Sub_\rho(Q) \ne \varnothing$ and $\sigma_1 \in Sub_\rho(Q)$. It follows from definition of the set $Sub_\rho(Q)$, that $Val_I(\forall(L_1\sigma_1)) = true$ for any substitution I and $comp(P)| = \forall(L_1\sigma_1)$ can be argued.

Next suppose that $s > 2$. Assume the hypothesis holds for derivations with the length less than s. Consider the *SLDNF*-derivation $Q_1,Q_2,\ldots,Q_s$, where $Q_1 = Q$, $Q_s = ?\text{-}$, $\sigma_1,\sigma_2,\ldots,\sigma_{s-1}$ is the sequence of mgu's used in this derivation,

Q has the form ?- $L_1,\ldots,L_n$, where $n > 0$. Let $Sel_\rho(Q) = i$, $i \in \{1,..,n\}$. We consider three possible cases.

a) L_i is a predicate term. Then P has a clause of the form A:- $K_1,\ldots,K_m$ ($m \geq 0$), such that L_i and A are unifiable and $\sigma_1 = mgu(L_i,A)$. The goal Q_2 is ?-$L_1\sigma_1,\ldots,L_{i-1}\sigma_1,K_1\sigma_1,\ldots,K_m\sigma_1,L_{i+1}\sigma_1,\ldots,L_n\sigma_1$. By the induction hypothesis $comp(P)| = \forall((L_1\sigma_1\&\ldots\&L_{i-1}\sigma_1\&K_1\sigma_1\&\ldots\&K_m\sigma_1\&L_{i+1}\sigma_1\&\ldots\&L_n\sigma_1)\sigma_2\ldots\sigma_{s-1})$. Thus, if $m > 0$, $comp(P)| = \forall((K_1\&\ldots\&K_m)\sigma_1\sigma_2\ldots\sigma_{s-1})$. Consequently, $comp(P)| = = \forall(A\sigma_1\sigma_2\ldots\sigma_{s-1})$ and $comp(P)| = \forall((L_1\sigma_1\&\ldots\&L_{i-1}\sigma_1\&A\sigma_1\&L_{i+1}\sigma_1\&\ldots\&L_n\sigma_1)\sigma_2\ldots\sigma_{s-1})$, since $A\sigma_1 = L_i\sigma_1$, then $comp(P)| = \forall((L_1\sigma_1\&\ldots\&L_{i-1}\sigma_1\&L_i\sigma_1\&L_{i+1}\sigma_1\&\ldots\&L_n\sigma_1)\sigma_2\ldots\sigma_{s-1})$, i.e. $comp(P)| = \forall((L_1\&\ldots\&L_{i-1}\&L_i\&L_{i+1}\&\ldots\&L_n)\sigma_1\sigma_2\ldots\sigma_{s-1})$.

b) L_1 is a ground negative literal. It follows from Theorem 1 that $comp(P)| = L_i$ and σ_1 is an empty substitution. The goal Q_2 is ?-$L_1,\ldots,L_{i-1},L_{i+1},\ldots,L_n$. By the induction hypothesis $comp(P)| = \forall((L_1\&\ldots\&L_{i-1}\&L_{i+1}\&\ldots\&L_n)\sigma_2\ldots\sigma_{s-1})$. Since σ_1 is an empty substitution, $comp(P)| = L_i$ and L_i contains no variables, then $comp(P)| = \forall((L_1\&\ldots\&L_n)\sigma_1\ldots\sigma_{s-1})$.

c) L_1 is a condition. Then $Sub_\rho(Q) \neq \varnothing$. The goal Q_2 is ?-$L_1\sigma_1,\ldots,L_{i-1}\sigma_1$, $L_{i+1}\sigma_1,\ldots,L_n\sigma_1$, where $\sigma_1 \in Sub_\rho(Q)$. By the induction hypothesis

$comp(P)| = \forall((L_1\sigma_1\&\ldots\&L_{i-1}\sigma_1\&L_{i+1}\sigma_1\&\ldots\&L_n\sigma_1)\sigma_2\ldots\sigma_{s-1})$, i.e.

$comp(P)| = \forall((L_1\&\ldots\&L_{i-1}\&L_{i+1}\&\ldots\&L_n)\sigma_1\ldots\sigma_{s-1})$.

It follows from definition of the set $Sub_\rho(Q)$ that $Val_I(\forall(L_i\sigma_1)) = true$ for any interpretation I and $comp(P)| = \forall(L_i\sigma_1)$. So we obtain that $comp(P)| = \forall(L_i\sigma_1\ldots\sigma_{s-1})$ and, consequently, $comp(P)| = \forall((L_1\&\ldots\&L_n)\sigma_1\ldots\sigma_{s-1})$. □

Corollary to Theorems 1 and 2. Let P be a program, Q be nonempty goal and $\rho \in R$. Then $Proc_\rho(P,Q) \subset Log(P,Q)$.

Practical *SLDNF*-Resolution. In practice we use computation rules with definition domain less than the definition domain of rules of the set R. Let $\rho \in R$. Define the set of practical computation rules R_ρ. Each rule $\rho' \in R_\rho$, as well as the rule ρ, corresponds to two functions $Sel_{\rho'}$ and $Sub_{\rho'}$, such that $Sel_{\rho'} = Sel_\rho$, the definition domain of $Sub_{\rho'}$ is a subset of the definition domain of the function Sub_ρ, and if $Sub_{\rho'}(Q)$ is defined for nonempty goal Q, then $Sub_{\rho'}(Q) = Sub_\rho(Q)$.

Let P be a program, Q be a goal ?-$L_1,\ldots,L_n$, $\rho' \in R_\rho$ be a practical computation rule and $Sel_{\rho'}(Q) = j$, $1 \leq j \leq n$, $n > 0$. If L_j is a condition and $Sub_{\rho'}(Q)$ is undefined or L_j is a negative literal, which is not ground, then ρ' rejects the goal Q. If *SLDNF*-tree for (P,Q) via ρ' is finite, contains no branches ending in the empty goal and contains at least one derivation of rejected goal, then the goal Q is rejected by ρ'.

An *SLDNF*-derivation of (P,Q), *SLDNF*-tree for (P,Q) and finitely failed *SLDNF*-tree for (P,Q) via ρ' are defined in the same manner as for rules of the set R. If there exists an *SLDNF*-derivation of (P,Q) via ρ' for the goal G, we denote this by $(P,Q)\vdash_{\rho'} G$.

The set of all goals, which are rejected by ρ', we denote by $Reject(\rho')$. Introduce a partial order in the set R_ρ. Let $\rho', \rho'' \in R_\rho$. Then $\rho' < \rho''$, if $Reject(\rho'') \subset Reject(\rho')$. R_ρ is a complete lattice, whose greatest element is the rule ρ and, whose least element is the rule that rejects all goals, for which $Sub_\rho(Q)$ is defined.

The set of answers to a nonempty goal Q to a program P corresponding to the procedural semantics based on a practical computation rule $\rho' \in R_\rho$ is defined in the same manner as for rules of the set R. We denote this set by $Proc_{\rho'}(P,Q)$.

It is easy to see that the following theorem holds.

Theorem 3. Let P be a program, Q be a nonempty goal, $\rho \in R$ and $\rho', \rho'' \in R_\rho$. Then

a) $\rho' < \rho'' \Rightarrow Proc_{\rho'}(P,Q) \subset Proc_{\rho''}(P,Q)$,

b) $Proc_{\rho'}(P,Q) \subset Proc_{\rho}(P,Q)$.

Theorem 4. Let P be a program, Q be a nonempty goal, $\rho \in R$ and $\rho' \in R_\rho$. Then $Proc_{\rho'}(P,Q) \subset Log(P,Q)$.

The proof follows from the Theorem 3 and the corollary to Theorems 1 and 2.

Received 15.07.2011

REFERENCES

1. **Nigiyan S.A.** Horn Programming with Built-in Predicates. Programming and Computer Software, 1996, v. 22, № 1, p. 19–25.
2. **Lloyd J.W.** Foundations of Logic Programming. Berlin: Springer–Verlag, 1984.