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 $\Phi$, $\Pi$ and $X$. $\Phi$ is a set of functional symbols each possessing an arity. For any $n \geq 0$, $\Phi$ contains a countable number of symbols of arity $n$. $X$ is a countable set of variables. Terms are composed of elements of sets $\Phi$ and $X$. By $\text{Var}(t)$ 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 $\Pi_1$ is the set of predicate symbols, $\Pi_2$ 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 $M^a \rightarrow \{true, false\}$. The atoms are defined as usual \[2\]. A formula of the first-order predicate logic with equality over logical operations $\neg, \&, \lor, \supset, \sim$ and quantifiers $\exists$ and $\forall$ is defined conventionally \[2\]. A predicate term is an atom, which uses predicate symbol from $\Pi_1$. A literal or a condition, each substitution is defined traditionally \[2\]. A substitution $\sigma$ is a set $\{t_1/x_1, \ldots, t_n/x_n\}$, where $t_i$ is a term, $x_i$ is a variable, $i \neq j \Rightarrow x_i \neq x_j$, $i, j = 1, \ldots, n$, $n \geq 0$. The following denotations are introduced: $\text{Arg}(\sigma) = \{x_1, \ldots, x_n\}$, $\text{Var}(\sigma) = \text{Var}(t_1) \cup \ldots \cup \text{Var}(t_n)$. 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 $\delta$ such that $A_1\delta = A_2\delta$. The substitution $\delta$ is called the unifier of $A_1$ and $A_2$. A unifier $\sigma$ is the most general unifier of $A_1$ and $A_2$ ($\sigma = \text{mgu}(A_1, A_2)$), if for any unifier $\delta$ there exists a substitution $\gamma$ 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 $\text{Val}(F)$. 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 $A\vdash K_1, \ldots, K_m$, 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 $\text{comp}(P)$:

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

where $p_1, \ldots, p_u$ are the predicate symbols from program $P$, $p_i \in \Pi_i$, $i = 1, \ldots, u$, $u \geq 1$, 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 $p:-B_1, \ldots, p:-B_n$, where $B_i$ is the body of the clause $p:-B_i$, $i = 1, \ldots, v$, $v \geq 1$, then $F(p)$ is $p:-E_1 \lor \ldots \lor E_v$, 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 $A_1:-B_1, \ldots, A_i:-B_i$, where $B_i$ is the body of the clause $A_i:-B_i$, $i = 1, \ldots, v$, $v \geq 1$, then $F(p)$ is $\forall x_1, \ldots, \forall x_n p(x_1, \ldots, x_n) \sim E_1 \lor \ldots \lor E_v$, where $x_1, \ldots, x_n$ are variables not appearing in the clauses $A_i:-B_1, \ldots, A_i:-B_i$, 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 $\exists L_1, \ldots, L_m$, 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 $Q$.\[1\]
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 \( \text{Var}(Q) \). 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 \( \text{Log}(P, Q) \).

- If \( \text{comp}(P) \neq -Q \) and \( \text{comp}(P) \neq Q \), then \( \text{Log}(P, Q) = \emptyset \);
- if \( \text{comp}(P) = -Q \), then \( \text{Log}(P, Q) = \{ \text{no} \} \);
- if \( \text{comp}(P) = Q \) and \( \text{Var}(Q) = \emptyset \), then \( \text{Log}(P, Q) = \{ \text{yes} \} \);
- if \( \text{comp}(P) = Q \) and \( \text{Var}(Q) = \{ y_1, \ldots, y_r \} \), \( r > 0 \), then \( \text{Log}(P, Q) \) consists of all such collections of terms \(< t_1, \ldots, t_r > \in M^f \), for which \( P|_\theta = Q \theta \), 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 \( \rho \) is defined via functions \( \text{Sel}_\rho \) and \( \text{Sub}_\rho \). Let \( Q \) be a goal \( ?-L_1, \ldots, L_n \), \( n \geq 1 \), \( \text{Sel}_\rho(Q) \in \{ 1, \ldots, n \} \). Let \( \text{Sel}_\rho(Q) = j (1 \leq j \leq n) \). If \( L_j \) is a literal, then \( \text{Sub}_\rho(Q) \) is undefined. Let \( L_j \) be a condition. So, \( \text{Sub}_\rho(Q) \) is a set of substitutions and for any \( \sigma \in \text{Sub}_\rho(Q) \) the following conditions are satisfied:

1. \( \text{Arg}(\sigma) \subset \text{Var}(L_j) \),
2. \( \text{Var}(\sigma) \cap (\text{Var}(Q) \setminus \text{Var}(L_j)) = \emptyset \),
3. \( \text{Val}(L_j \sigma \gamma) = \text{true} \) for any substitution \( \gamma \) such that \( \text{Var}(L_j \sigma \gamma) = \emptyset \), and for any substitution \( \delta \), where \( \text{Var}(L_j \delta) = \emptyset \) and \( \text{Val}(L_j \delta) = \text{true} \), there exists \( \sigma \in \text{Sub}_\rho(Q) \) and \( \gamma \), so that \( L_j \delta = L_j \sigma \gamma \).

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

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

1. Each node of the tree is a goal.
2. The root node is \( Q \).
3. Let \( ?-L_1, \ldots, L_n \) (\( n \geq 1 \)) be a node of the tree and \( \text{Sel}_\rho(?-L_1, \ldots, L_n) = j (1 \leq j \leq n) \).
4. Let \( L_j \) be a positive literal, then this node has a descendant for each clause \( A \& K_1, \ldots, K_m \) (\( m \geq 0 \)) from program \( P \), such that \( L_j \) and \( A \) are unifiable. The descendant is \( ?-L_{j1} \sigma_1, \ldots, L_{jn} \sigma_n, K_1 \sigma, \ldots, K_m \sigma, L_1 \sigma, \ldots, L_n \sigma \), where \( \sigma = \text{mgu}(L_j, A) \).
5. Let \( L_j \) be a ground negative literal. If the subgoal \( L_j \) is successful, the single descendant of the node is \( ?-L_1, \ldots, L_{j1}, L_{j1+1}, \ldots, L_n \). If the subgoal \( L_j \) fails, the node has no descendants.
6. Let \( L_j \) be a condition. If \( \text{Sub}_\rho(Q) \neq \emptyset \), then this node has a descendant for each \( \delta \in \text{Sub}_\rho(Q) \). The descendant is \( ?-L_1 \delta, \ldots, L_{j1} \delta, L_{j1+1} \delta, \ldots, L_n \delta \). If \( \text{Sub}_\rho(Q) = \emptyset \), then this node has no descendants.
7. Nodes which are the empty goal have no descendants.

Let \( P \) be a program, \( Q \) be a nonempty goal and \( \rho \in \mathcal{R} \). If SLDNF-tree for \( (P, Q) \) via \( \rho \) 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
retracted by \(\rho\). A finitely failed SLDNF-tree for \((P,Q)\) via \(\rho\) 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 \(\rho\), where \(Q_1 = Q\) is defined as follows:

Suppose \(Q_i\) (\(i > 0\)) is \(-L_1, \ldots, L_n\) \((n \geq 1\) and \(\text{Sel}_i(P, Q_i = l_i \ldots, L_n) = j\) \((1 \leq j \leq n)\).

If \(L_j\) is a positive literal, \(A \cdot K_{i_1}, \ldots, K_{i_m} (m \geq 0)\) is a clause from \(P\) and \(\sigma = \text{mgu}(L_j, A)\), then the derived goal \(Q_{i+1}\) is \(-L_1 \sigma, \ldots, L_j \sigma, K_{i_1} \sigma, \ldots, K_{i_m} \sigma, L_{j+1}\sigma, \ldots, L_n\sigma\).

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

If \(L_j\) is a condition, \(\text{Sub}_j(Q_i) \neq \emptyset\), \(\delta \in \text{Sub}_j(Q_i)\), then the derived goal \(Q_{i+1}\) is \(\sim L_1 \delta, \ldots, L_j \delta, L_{j+1} \delta, \ldots, L_n \delta\).

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 \(\text{Proc}_\rho(P, Q)\).

If the goal \(Q\) is rejected by the rule \(\rho\), then \(\text{Proc}_\rho(P, Q) = \emptyset\);

if \((P, Q)\) has a finitely failed SLDNF-tree for \((P, Q)\) via \(\rho\), then \(\text{Proc}_\rho(P, Q) = \{\text{yes}\}\);

if \((P, Q)\) fails and \(\text{Var}(Q) = \emptyset\), then \(\text{Proc}_\rho(P, Q) = \{\text{no}\}\);

if \((P, Q)\) fails and \(\text{Var}(Q) = \{y_1, \ldots, y_r\}\), \(r > 0\), then \(\text{Proc}_\rho(P, Q)\) consists of all such collections of terms \(\langle t_1, \ldots, t_r \rangle \in \mathcal{M}'\) that there exist an SLDNF-derivation of \((P, Q)\) via \(\rho\), which end in the empty goal and such substitution \(\delta\) that \(\{t_1/y_1, \ldots, t_r/y_r\} \subset \sigma_1, \sigma_2, \ldots, \delta\), where \(\sigma_i\) is the substitution corresponding to the application of the rule \(\rho\) 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 \(-L_1, \ldots, L_n\), \(\rho \in R\), \(\text{Sel}_i(Q) = i\) \((1 \leq i \leq n)\) 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 \(\rho\), then \(\text{comp}(P)|\sim Q\).

**Lemma 2.** Let \(P\) be a program, \(Q\) be a nonempty goal \(-L_1, \ldots, L_n\), \(\rho \in R\), \(\text{Sel}_i(Q) = i\) \((1 \leq i \leq n)\), \(L_i\) 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 \(\rho\), \(Q_j \neq \sim -L_j (1 \leq j \leq r)\). Then \(\text{comp}(P)|\sim Q_{\sim \ldots \sim Q}\).

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

\(\text{Val}(Q) = \text{true} \iff \text{there exists } Q \in D \text{ such that } \text{Val}(Q') = \text{true}\).

**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 \(\rho\), then \(\text{comp}(P)|\sim -Q\).

**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 \(?-L_1, \ldots, L_n\), \( Sel_j(G) = i \) \( (1 \leq i \leq n, n > 0) \) and \( L_i \) is \( \neg A \), where \( A \) is ground predicate term. There are two cases to consider.

a) Goal \(?-A\) fails.

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

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

If for certain nonempty goal \( Q' \), we have \((P, Q')\) \( \vdash \neg ?-\) 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 \( \text{comp}(P) = Q' \), therefore, \( \text{comp}(P) = A \). Since \( \neg G' \) is the formula \( \forall (\neg L_1 \lor \cdots \lor \neg L_{i-1} \lor \neg L_{i+1} \lor \cdots \lor \neg L_n) \) and \( \text{comp}(P) = A \), then it is evident that \( \text{comp}(P) = \forall (\neg L_1 \lor \cdots \lor \neg L_{i-1} \lor \neg L_{i+1} \lor \cdots \lor \neg L_n) \), i.e. \( \text{comp}(P) = \neg G' \).

Let \( \{Q_1, \ldots, Q_r\} \) be the set of all goals of depth \( h \), \( r > 0 \). We have \( \text{comp}(P) = \neg Q \) for any \( i = 1, \ldots, r \). Using the fact, that \( \rho \) selects no negative literals to the depth \( h \) and Lemmas 1, 2, 3, we obtain that \( \text{comp}(P) = Q \equiv Q_1 \lor \cdots \lor Q_r \). Consequently, \( \text{comp}(P) = \neg Q \). \( \square \)

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

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

Suppose first that \( s = 2 \). This means that \( Q \) has the form \(?-L_1 \). 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 \( \sigma_1 = \text{mgu} (L_1, A) \). Since \( \text{comp}(P) = \forall (A \sigma_1) \), then \( \text{comp}(P) = \forall (L_1 \sigma_1) \).

b) \( L_1 \) is a ground negative literal of the form \( \neg A \), and \( (P, A) \) has a finitely failed SLDNF-tree via \( \rho \). Theorem 1 shows that \( \text{comp}(P) = \neg A \), i.e. \( \text{comp}(P) = L_1 \) and \( \sigma_1 \) is an empty substitution.

c) \( L_1 \) is a condition. Then \( \text{Sub}_A(Q) \neq \emptyset \) and \( \sigma_1 \in \text{Sub}_A(Q) \). It follows from definition of the set \( \text{Sub}_A(Q) \), that \( \text{Val} (\forall (L_1 \sigma_1)) = true \) for any substitution \( I \) and \( \text{comp}(P) = \forall (L_1 \sigma_1) \) 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 \( Q_1 = Q, Q_i = ?-, \sigma_1, \sigma_2, \ldots, \sigma_{s-1} \) is the sequence of mgu's used in this derivation,
Q has the form ?:- L₁,...,Lₙ, where n > 0. Let Sel₁(Q) = i, i ∈ {1,...,n}. We consider three possible cases.

a) Lᵢ is a predicate term. Then P has a clause of the form A:- K₁,...,Kₘ (m ≥ 0), such that Lᵢ and A are unifiable and σ₁ = mgu(Lᵢ,A). The goal Q₂ is ?:-L₁σ₁,...,Lᵢσᵢ,K₁σ₁,...,Kₘσ₁,Lᵢ₊₁σᵢ,...,Lᵢ₋₁σᵢ. By the induction hypothesis \( \text{comp}(P) = \forall((L₁σ₁&...&Lᵢ₋₁σ₁&K₁σ₁&...&Kₘσ₁&Lᵢ₊₁σᵢ&...&Lᵢ₋₁σ₁)σᵢ₂...σᵢ₋₁) \). Thus, if \( m > 0 \), \( \text{comp}(P) = \forall((K₁&...&Kₘσ₂...σ₋₁) \). Consequently, \( \text{comp}(P) = \forall(Aσ₂...σ₋₁) \) and \( \text{comp}(P) = \forall((L₁σ₁&...&Lᵢ₋₁σ₁&Aσ₂&Lᵢ₊₁σ₁&...&Lᵢ₋₁σ₁)σ₂...σ₋₁) \), since \( Aσ₁ = Lᵢσᵢ \), then \( \text{comp}(P) = \forall((L₁σ₁&...&Lᵢ₊₁σ₁&Lᵢ₊₁σ₁&...&Lᵢ₋₁σ₁)σ₂...σ₋₁) \), i.e. \( \text{comp}(P) = \forall((L₁&...&Lᵢ₋₁&Lᵢ&Lᵢ₊₁&...&Lᵢ₋₁)σ₂...σ₋₁) \).

b) L₁ is a ground negative literal. It follows from Theorem 1 that \( \text{comp}(P) = L₁ \) and σ₁ is an empty substitution. The goal Q₂ is ?:-L₁,...,Lᵢ₋₁,Lᵢ₊₁,...,Lₙ. By the induction hypothesis \( \text{comp}(P) = \forall((L₁&...&Lᵢ₋₁&Lᵢ₊₁&...&Lₙ)σ₂...σ₋₁) \). Since σ₁ is an empty substitution, \( \text{comp}(P) = L₁ \) and \( L₁ \) contains no variables, then \( \text{comp}(P) = \forall((L₁&...&Lₙ)σ₁...σ₋₁) \).

c) L₁ is a condition. Then Sub₂(A) ≠ ∅. The goal Q₂ is ?:-L₁σ₁,...,Lᵢ₋₁σ₁, Lᵢ₊₁σ₁,...,Lₙσ₁, where σ₁ ∈ Sub₂(A). By the induction hypothesis \( \text{comp}(P) = \forall((L₁σ₁&...&Lᵢ₋₁σ₁&Lᵢ₊₁σ₁&...&Lₙσ₁)σ₂...σ₋₁) \), i.e. \( \text{comp}(P) = \forall((L₁&...&Lᵢ₋₁&Lᵢ₊₁&...&Lₙ)σ₁...σ₋₁) \).

It follows from definition of the set Sub₂(Q) that \( \text{Val}(\forall(L₁σ₁)) \) = true for any interpretation \( I \) and \( \text{comp}(P) = \forall(L₁σ₁) \). So we obtain that \( \text{comp}(P) = \forall(L₁σ₁...σ₋₁) \) and, consequently, \( \text{comp}(P) = \forall((L₁&...&Lₙ)σ₁...σ₋₁) \).

**Corollary to Theorems 1 and 2.** Let P be a program, Q be nonempty goal and \( ρ \in R \). Then \( \text{Proc}(P,Q) \subseteq \text{Log}(P,Q) \).

**Practical SLDNF-Resolution.** In practice we use computation rules with definition domain less than the definition domain of rules of the set R. Let \( ρ \in R \).

Define the set of practical computation rules \( R_ρ \). Each rule \( ρ' \in R_ρ \) as well as the rule \( ρ \), corresponds to two functions \( \text{Sel}_ρ \) and \( \text{Sub}_ρ \), such that \( \text{Sel}_ρ = \text{Sel}_ρ \) the definition domain of \( \text{Sub}_ρ \) is a subset of the definition domain of the function \( \text{Sub}_ρ \) and if \( \text{Sub}_ρ(Q) \) is defined for nonempty goal Q, then \( \text{Sub}_ρ(Q) = \text{Sub}_ρ(Q) \).

Let P be a program, Q be a goal ?:-L₁,...,Lₙ, \( ρ' \in R_ρ \) be a practical computation rule and \( \text{Sel}_ρ(Q) \) = j, 1 ≤ j ≤ n, n > 0. If \( L_j \) is a condition and \( \text{Sub}_ρ(Q) \) 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 \( (P,Q)\rightarrow ρ' G \).

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

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) $\rho' < \rho'' \Rightarrow \text{Proc}_\rho(P, Q) \subseteq \text{Proc}_{\rho''}(P, Q)$,

b) $\text{Proc}_\rho(P, Q) \subset \text{Proc}_{\rho'}(P, Q)$.

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

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

Received 15.07.2011

REFERENCES


I. U. Sargsyan. Ob SLDNF-резолюции в логическом программировании с отрицанием
стр. 40–46

Работа посвящена логическому программированию с отрицанием и встроенными предикатами. Рассматриваются обобщенные логические программы (логические программы, использующие отрицание) и обобщенные запросы (запросы, использующие отрицание). Предложена модификация SLDNF-резолюции для встроенных предикатов. Доказана логическая непротиворечивость модифицированной SLDNF-резолюции. Рассмотрена SLDNF-резолюция, используемая в реальных системах логического программирования (практическая SLDNF-резолюция) и доказана ее непротиворечивость.