PROCEEDINGS OF THE YEREVAN STATE UNIVERSITY

Physical and Mathematical Sciences 2014, № 1, p. 40–47

Informatics

ON OPTIMIZATION OF MONADIC LOGIC PROGRAMS

S. A. KHACHATRYAN *

Chair of Programming and Information Technologies YSU, Armenia

The article is devoted to the optimization of monadic logic programs and goals (programs and goals, which do not use functional symbols of arity > 1 and use only predicate symbols of arity 1). A program P is terminating with respect to a goal G, if an SLD-tree of P and G is finite. In general, monadic programs are not terminating. Program and goal transformations are introduced, by which a monadic program P and a variable-free monadic goal G are transformed into P' and G', such that P' is terminating with respect to G' and $P \models G$, if and only if $P' \models G'$. Note that the transformed program P' is the same for all goals.

MSC2010: 68N17.

***Keywords*:** monadic logic programs, optimization, termination, transformation.

1. Introduction. The present article deals with the study of termination of logic programs. We study termination of monadic programs with respect to variable-free monadic goals. A program (a goal) is called monadic, if it does not use functional symbols of arity > 1 and uses only predicate symbols of arity 1.

It is known that there is an algorithm, which accepts a monadic program and a monadic goal as an input and decides whether or not the goal is a logical consequence of the program [1].

The technical tool we shall use is called *level mapping* by Cavedon [2], who studied various classes of logic programs with negation. A level mapping is a function assigning natural numbers to variable-free atoms. In [3] it is shown, that if a logic program is *recurrent,* i.e. if the heads of variable-free instances of program clauses have higher levels than the atoms occurring in the body of the same instance, then it is terminating with respect to bounded goals, i.e. goals, whose instances are below some fixed level. We present program and goal transformations, by which a monadic program P and a monadic variable-free goal G are transformed into P' and G' such that P' is terminating with respect to G' and $P \models G$, if and only if $P' \models G'$. Note

* E-mail: suren1525@gmail.com

that the transformed program P' is the same for all goals, in other words, program transformation is independent of goals.

2. Notation and Background. Consider three non-intersecting 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 set of symbols of arity n. Π is a set of predicate symbols each possessing an arity. For any $n \geq 0$, Π contains a countable set of symbols of arity n. X is a countable set of variables. Terms are composed from the elements of sets Φ and X. The atoms are defined as usual [4, 5]. A formula of the first-order predicate logic over logical operations $\neg, \vee, \wedge, \rightarrow$ and quantifiers $\exists$ and $\forall$ is defined conventionally.

A *ground term* is a term containing no variables. A *ground atom* is an atom term containing no variables. The *Herbrand universe U* is the set of all ground terms and the *Herbrand base B* is the set of all ground atoms. A *Herbrand interpretation I* is a subset of the Herbrand base. The value of a closed formula F in interpretation I is defined as usual. An interpretation I is said to be a model of a non-empty set of closed formulas W ($I \models W$,) if I is a model of formulas of W. For closed formulas F and G, the relation $F \models G$ means that every Herbrand model of F is a model of G too. If W is a non-empty set of closed formulas and F is a closed formula, then the relation $W \models F$ means that every Herbrand model of the formulas of W is a model of F. A non-empty set of closed formulas is called satisfiable, if it has a Herbrand model, otherwise it is called unsatisfiable.

A direct consequence of these definitions is the following proposition.

Proposition 2.1. Let W be a non-empty set of closed formulas and F be a closed formula. Then $W \models F$, if and only if $W \cup \{\neg F\}$ is unsatisfiable.

A *program clause* is a construct of the form $A \leftarrow B_1, \ldots, B_m$, where $m \geq 0$, $B_1, \ldots, B_m$ and A are atoms. The atom A is called the head, and the sequence of atoms $B_1, \ldots, B_m$ is the body of the clause. In case if $m = 0$, the clause is called a fact and is denoted as A. The formula $\forall(B_1 \wedge \cdots \wedge B_m \rightarrow A)$ is associated to the clause $A \leftarrow B_1, \ldots, B_m$.

A *logic program* is a non-empty finite set of program clauses.

A *goal* is a construct of the form $\leftarrow C_1, \ldots, C_k$, where $C_1, \ldots, C_k$ are atoms. A goal, in which the sequence $C_1, \ldots, C_k$ is empty, is called an *empty goal*. The formula $\exists(C_1 \wedge \cdots \wedge C_k)$ is associated to the non-empty goal $\leftarrow C_1, \ldots, C_k$.

A *substitution* $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ is a finite set of pairs of a variable and a term, where t_i is distinct from x_i and the variables $x_1, \ldots, x_n$ are also distinct. A *simple expression* is either a term or an atom. If E is a simple expression, then the *instance* of E by θ, denoted by $E\theta$, is the simple expression obtained from E by simultaneous replacement of each occurrence of the variable x_i in E by the term t_i, $i = 1, \ldots, n$. If $E\theta$ is ground, then $E\theta$ is called a *ground instance* of E. The composition of substitutions is defined in a usual way [4]. If $L = \{E_1, \ldots, E_m\}$ is a finite set of simple expressions and θ is a substitution, then $L\theta$ denotes the set of $E_1\theta, \ldots, E_m\theta$. Let L be a finite set of simple expressions. A substitution θ is called a *unifier* for L, if $L\theta$ is a singleton. A unifier θ is called a *most general unifier (mgu)* for L, if for each unifier σ of L there exists a substitution γ such that $\sigma = \theta\gamma$.

If S is a program clause of the form $A \leftarrow B_1,\ldots,B_m$, $m \geq 0$, then $S\theta$ denotes the clause $A\theta \leftarrow B_1\theta,\ldots,B_m\theta$. If $P = \{S_1,\ldots,S_n\}$, $n \geq 1$, is a program, then $P\theta$ denotes the program consisting of $S_1\theta,\ldots,S_n\theta$. If G is a goal of the form $\leftarrow C_1,\ldots,C_k$, $k > 0$, then $G\theta$ denotes the goal $\leftarrow C_1\theta,\ldots,C_k\theta$.

We will denote the set of variables occurring in a simple expression E by $Var(E)$. The set of variables of a program clause S of the form $A \leftarrow B_1,\ldots,B_m$ we will denote by $Var(S)$ and define as $Var(S) = Var(A) \cup Var(B_1) \cup \cdots \cup Var(B_m)$.

The set of variables of a goal G of the form $\leftarrow C_1,\ldots,C_k$ we will denote by $Var(G)$ and define as $Var(G) = Var(C_1) \cup \cdots \cup Var(C_k)$.

Let G be a non-empty goal $\leftarrow C_1,\ldots,C_k$ and S be a clause $A \leftarrow B_1,\ldots,B_m$. Then the goal Q is obtained by *SLD-resolution* from G and S, if:

- $Var(G) \cap Var(S) = \emptyset$.
- C_i and A are unifiable, where C_i is an atom (called the *selected atom*) in G, $1 \leq i \leq k$.
- Q is the goal $\leftarrow C_1\theta,\ldots,C_{i-1}\theta,B_1\theta,\ldots,B_m\theta,C_{i+1}\theta,\ldots,C_k\theta$, where $\theta = mgu(C_i,A)$.

The choice of the selected atom is performed by what is called a selection function. Q is called an *SLD-resolvent* of G and S.

Let P be a program and G be a goal. An *SLD-derivation* of $P \cup \{G\}$ is a (finite or infinite) sequence $G_0, G_1, \ldots$ of goals such that $G_0 = G$ and each G_{i+1} is obtained by SLD-resolution from the goal G_i and a clause of P ($i \geq 0$).

Let P be a logic program and G be a goal. An *SLD-tree* of P and G is a tree satisfying the following conditions:

- Each node of the tree is a (possibly empty) goal.
- The root node is G.
- Let $\leftarrow C_1,\ldots,C_k (k \geq 1)$ be a node in the tree and suppose, that $C_i (1 \leq i \leq k)$ is the selected atom. Then for each clause $A \leftarrow B_1,\ldots,B_m, m \geq 0$, in P such that C_i and A are unifiable with $mgu\theta$, the node has a child $\leftarrow C_1\theta,\ldots,C_{i-1}\theta, B_1\theta,\ldots,B_m\theta, C_{i+1}\theta,\ldots,C_k\theta$.
- Nodes which are empty goals have no children.

To every branch of an SLD-tree there corresponds an SLD-derivation.

A program P is terminating with respect to a goal G, if an SLD-tree of P and G is finite.

A *depth* is a mapping from ground terms to natural numbers $depth : U \rightarrow N$, defined as follows:

$$depth(f(t_1,\ldots,t_n)) = \begin{cases} 1 + \max\{depth(t_i) | i = 1,\ldots,n\}, \text{ if } n > 0; \\ 0, \text{ if } n = 0. \end{cases}$$

For example, $depth(s(s(0))) = 1 + depth(s(0)) = 1 + 1 + depth(0) = 1 + 1 + 0 = 2$.

We denote the number of elements in a finite set X as $\overline{\overline{X}}$.

3. Monadic Logic Programs. In this section we study monadic programs and goals. Each monadic program P has a corresponding permitted set of goals, which is denoted by $\Delta(P)$. For a monadic program P, as permitted set of goals, we regard

monadic goals, which use only predicate symbols appearing in P and do not contain variables. In this section only variable-free goals are considered.

For monadic programs and goals we make a number of assumptions:

1. 0 is the only constant in programs and goals.
2. Rules do not contain ground terms.
3. Every rule contains one variable. For convenience, we also assume that variables in different rules are identical, i.e. there is just one variable in a program, denoted by x.
4. Terms in rules are of depth at most 1, i.e. x or $f(x)$ for some functional symbol f.
5. A term in every fact is the constant 0 or the variable x.

Any monadic program P_1 and monadic goal $G_1 \in \Delta(P_1)$ can be transformed into monadic program P_2 and monadic goal $G_2 \in \Delta(P_2)$ satisfying the mentioned assumptions such that $P_1 \models G_1$, if and only if $P_2 \models G_2$ [6]. Note that only assumption 1 is for goals. In this section we will consider only monadic programs and goals satisfying these assumptions.

The set of predicate symbols used in the set W of formulas will be denoted by Π_W, and the set of functional symbols used in W will denoted by Φ_W. For a monadic program P and a variable-free monadic goal G, $constraint(P,G)$ will be defined in the following way: $constraint\,(P,G) = \overline{\overline{P}} * \sum_{i=0}^{d-1} \left(\overline{\overline{\Phi_{P\cup\{G\}}}}\right)^i$, where $d = l + 2^{\overline{\overline{\Pi_P}}}$ and l is the maximum depth of the terms in G.

Definition 3.1. *(Program Transformation).* Let P be a monadic program. For the program P we construct a program P' as follows: for every clause S of the form $p(t) \leftarrow p_1(t_1), \ldots, p_m(t_m) \in P, m \geq 0$, we add the clause

$$p^T\,(t, s\,(z)) \leftarrow p_1^T\,(t_1, z)\,, \ldots, p_m^T(t_m, z)$$

to P', where the variable $z \notin Var\,(S)$ and the functional symbol $s \notin \Phi_P$.

As a result of this transformation, for each clause of program P a new clause is defined, which has the same structure as the original one, but with the new predicate symbols of arity 2. The second parameter of the atoms is used for determining levels of the atoms. These parameters are added as constraints.

Definition 3.2. *(Goal Transformation).* For a monadic program P and a variable-free goal G of the form $\leftarrow p_1\,(\tau_1)\,, \ldots, p_k(\tau_k)$, we define the goal G' as follows:

$$\leftarrow p_1^T\left(\tau_1, s^h\,(0)\right), \ldots, p_k^T\,(\tau_k, s^h\,(0)),$$

where $h = constraint\,(P,G)$.

Theorem 3.1. Let P be a monadic program and $G \in \Delta(P)$. P' is the program obtained from P by program transformation, and G' is the goal obtained from G by goal transformation. Then $P \models G$, if and only if $P' \models G'$.

To prove Theorem 3.1, we will introduce some notations and prove auxiliary lemmas.

Let W be a set of formulas. We will denote by U_W the set of ground terms by using functional symbols from Φ_W. For W we define $I_W = \{p(t) | \, p \in \Pi_W, t \in U_W\}$. For an interpretation $I \subset I_W$ we define $I[t] = \{p | \, p(t) \in I\}$ and $\Psi_W(I) = \{I[t] | \, t \in U_W\}$. It is easy to see that for any interpretation $I \subset I_W$ the following holds: $\overline{\overline{\Psi_W(I)}} \leq 2^{\overline{\Pi_W}}$.

We denote by $U^{(1)}$ the set of ground terms, which do not use functional symbols of arity > 1. Next we denote by $B^{(1)}$ the set of ground atoms, which do not use functional symbols of arity > 1 and use only predicate symbols of arity 1.

Lemma 3.1. Let P be a monadic program, $p(\tau) \in B^{(1)}$, terms $t_1, t_2 \in U_{P \cup \{p(\tau)\}}$ and interpretations $I, J \subset I_{P \cup \{p(\tau)\}}$. If: $I[0] = J[0]$; $I[\tau] = J[\tau]$; $I[t_1] = J[t_2]$; $I[f(t_1)] = J[f(t_2)]$, for every functional symbol $f \in \Phi_P$, than:

$$I \models P\{x/t_1\} \cup \{\neg p(\tau)\}, \text{ if and only if } J \models P\{x/t_2\} \cup \{\neg p(\tau)\}.$$

Proof. For any atom $q(\tau_1)$ used in formulas of $P\{x/t_1\} \cup \{\neg p(\tau)\}$, τ_1 is 0 or τ or t_1 or $f(t_1)$ for some functional symbol $f \in \Phi_P$. For the corresponding atom $q(\tau_2)$ used in formulas of $P\{x/t_2\} \cup \{\neg p(\tau)\}$, τ_2 is 0 or τ or t_2 or $f(t_2)$ accordingly.

Therefore, in view of the assumptions of this Lemma $q(\tau_1) \in I$, if and only if $q(\tau_2) \in J$. □

For a term $t_0 \in U^{(1)}$ we define the set of terms $st(t_0)$ in the following way:

- $st(t_0) = \{t_0\}$, if t_0 is a constant;
- $st(t_0) = st(t_1) \cup \{f(t) | t \in st(t_1)\}$, if $t_0 = f(t_1)$.

Note that all subterms of the term t_0 are in $st(t_0)$.

Lemma 3.2. Let P be a monadic program and an atom $p(\tau) \in B^{(1)}$. Then $P \models p(\tau)$, if and only if $\bigcup_{depth(t)<d} P\{x/t\} \models p(\tau)$ for $d = depth(\tau) + 2^{\overline{\overline{\Pi_P}}}$, where $t \in U_{P \cup \{p(\tau)\}}$.

Proof. Suppose $\bigcup_{depth(t)<d} P\{x/t\} \models p(\tau)$. It is straightforward that $P \models p(\tau)$. Now, suppose $P \models p(\tau)$. By Proposition 2.1, it follows that $W = P \cup \{\neg p(\tau)\}$ is unsatisfiable. To prove that $\bigcup_{depth(t)<d} P\{x/t\} \models p(\tau)$, we will show that $W_d = \bigcup_{depth(t)<d} P\{x/t\} \cup \{\neg p(\tau)\}$ is unsatisfiable (Proposition 2.1). Assume in the contrary that W_d is satisfiable. If W_d is satisfiable then it has a finite model $I \subset I_W$. Given I, we define a model $J \subset I_W$ for W as follows.

A term $t_0 \in U_W$ is called *initial,* if:

$$depth(t_0) \leq depth(\tau) \text{ or } (\forall t \in st(t_0), depth(t) \geq depth(\tau) \text{ and } t \neq t_0 \Rightarrow J[t] \neq J[t_0]).$$

We define J inductively by induction on k, the depth of terms. If $k = 0$, we take $J[0] = I[0]$. Now, assume $J[t]$ has been defined for all terms $t \in U_W$ of depth smaller than k. Take a term $t_0 = f(t_1) \in U_W$ of depth k.

If t_1 is initial, put $J[t_0] = I[t_0]$. If t_1 is not initial, for t_1 we can uniquely choose (by some algorithm) an initial term $t_2 \in st(t_1)$ such that $depth(t_2) \geq depth(\tau)$ and $J[t_1] = J[t_2]$. We put $J[t_0] = J[f(t_2)]$. We show, that if term $t \in U_W$ is initial, then $depth(t) < d$. From the definition of J, it can be shown by an easy induction, that $\forall t_1, t_2 \in U_W$, *if* t_2 *is not initial and* t_2 *is a subterm of* t_1, *then* t_1 *is not initial.*

Immediately it follows that $\forall t_1, t_2 \in U_W$, if t_1 is initial and t_2 is a subterm of t_1, then t_2 is initial. For any initial term $t \in U_W$, from the above, it follows that:

$$\forall t_1, t_2 \in U_W, \text{ if } t_1 \text{ and } t_2 \text{ are subterms of } t \text{ such that } depth(t_1) \geq depth(\tau), \\ depth(t_2) \geq depth(\tau) \text{ and } t_1 \neq t_2 \Rightarrow J[t_1] \neq J[t_2].$$

Therefore, we conclude, that if t is initial, then

$$depth(t) < depth(\tau) + \overline{\overline{\Psi_W(J)}} \leq depth(\tau) + 2^{\overline{\overline{\Pi_W}}} = depth(\tau) + 2^{\overline{\overline{\Pi_P}}} = d.$$

Finally, we show that J is a model of W. As terms 0 and τ are initial, it follows that:

- $J[0] = I[0]$;
- $J[\tau] = I[\tau]$.

Define ground instance of W as follows: $W\{x/t\} = P\{x/t\} \cup \{\neg p(\tau)\}$, $t \in U_W$. Take the ground instance $W\{x/t_0\}$. If t_0 is initial, then by the definition of J it follows that:

- $J[t_0] = I[t_0]$;
- $J[f(t_0)] = I[f(t_0)]$ for any functional symbol $f \in \Phi_W$.

As $depth(t_0) < d$, using Lemma 3.1 we can conclude the validity of $W\{x/t_0\}$ in J from the validity of $W\{x/t_0\}$ in I. If t_0 is not initial, we take the initial term $t_1 \in st(t_0)$, which is used in definition of $J[f(t_0)]$, $f \in \Phi_W$. Consequently, $J[t_0] = J[t_1]$. Hence, $J[t_0] = J[t_1] = I[t_1]$. Moreover, by the definition of J we have $J[f(t_0)] = J[f(t_1)] = I[f(t_1)]$ for any functional symbol $f \in \Phi_W$. As $depth(t_1) < d$, using Lemma 3.1 we can conclude the validity of $W\{x/t_0\}$ in J from the validity of $W\{x/t_1\}$ in I. Hence, J is a model of W. Thus, W is satisfiable. This is a contradiction. □

Recall the mapping $T_P : 2^B \to 2^B$ (see, e.g. [4]), which is defined as follows: $T_P(I) = \{A\theta \mid \theta \text{ is grounding substitution for } A \leftarrow B_1, \ldots, B_m \in P \text{ and } B_1\theta, \ldots, B_m\theta \in I, m \geq 0\}$. Define T_P^k, $k \geq 0$, as follows:

- $T_P^0 = \emptyset$;
- $T_P^k = T_P\left(T_P^{k-1}\right), k \geq 1$.

The function T_P has a least fixpoint, denoted by $T_P^{fix} = \sup\left\{T_P^k \mid k \geq 0\right\}$.

Lemma 3.3. Let P be a monadic program. Then $P \models p(\tau)$, if and only if $T_P^h \models p(\tau)$, where $\tau \in U^{(1)}$ and $h = constraint(P, p(\tau))$.

Proof. Suppose $P \models p(\tau)$. By Lemma 3.2, $P_0 = \bigcup_{depth(t)<d} P\{x/t\} \models p(\tau)$ for $d = depth(\tau) + 2^{\overline{\overline{\Pi_P}}}$, $t \in U_{P \cup \{p(\tau)\}}$. $T_{P_0}^k \subset T_P^k$, as each clause of P_0 is a ground instance of a clause of P, $k \geq 0$. As $P_0 \models p(\tau)$ and P_0 is a variable-free program, it follows that $T_{P_0}^{c_0} \models p(\tau)$, where $c_0 = \overline{\overline{P_0}}$ [7]. The number of ground terms in $U_{P \cup \{p(\tau)\}}$ of depth i will be $\left(\overline{\overline{\Phi_{P \cup \{p(\tau)\}}}}\right)^i$. Hence, the number of ground terms in $U_{P \cup \{p(\tau)\}}$ with depth of $< d$ will be $\sum_{i=0}^{d-1} \left(\overline{\overline{\phi_{P \cup \{p(\tau)\}}}}\right)^i$. Whence, $c_0 \leq c * \sum_{i=0}^{d-1} \left(\overline{\overline{\Phi_{P \cup \{p(\tau)\}}}}\right)^i = constraint(P, p(\tau)) = h$, where $c = \overline{\overline{P}}$. Therefore, $T_{P_0}^h \models p(\tau)$, as $T_{P_0}^{c_0} \subset T_{P_0}^h$. Thus, $T_P^h \models p(\tau)$, as $T_{P_0}^h \subset T_P^h$. Now, to show the converse, suppose $T_P^h \models p(\tau)$.

As $T_P^h \subset T_P^{fix}$, it follows that $T_P^{fix} \models p(\tau)$. Hence, as T_P^{fix} coincides with the least Herbrand model of P, it follows that $P \models p(\tau)$ [4]. □

Lemma 3.4. Let P be a monadic program and P' by the program obtained from P by program transformation. Then $P' \models p^T(\tau, s^k(0))$, if and only if $T^k_{P'} \models p^T\left(\tau, s^k(0)\right)$, where $\tau \in U^{(1)}$, $p^T \in \Pi_{P'}$,$k \geq 1$.

Proof.

- Suppose $P' \models p^T(\tau, s^k(0))$. For any $k \geq 1$ and any $p^T(\tau, \tau_0) \in T^k_{P'} \backslash T^{k-1}_{P'}$, from the structure of P', it follows that $depth(\tau_0) \geq k$. Hence, as $depth\left(s^k(0)\right) = k$ and $P' \models p^T(\tau, s^k(0))$, it follows that $p^T(\tau, s^k(0)) \in T^k_{P'}$.
- Suppose $T^k_{P'} \models p^T\left(\tau, s^k(0)\right)$. As $T^k_{P'} \subset T^{fix}_{P'}$, it follows that $T^{fix}_{P'} \models p^T\left(\tau, s^k(0)\right)$. Whence, as $T^{fix}_{P'}$ coincides with the least Herbrand model of P', it follows that $P' \models p^T\left(\tau, s^k(0)\right)$ [4]. □

Lemma 3.5. Let P be a monadic program and P' be the program obtained from P by program transformation. Then $T^k_P \models p(\tau)$, if and only if $T^k_{P'} \models p^T\left(\tau, s^k(0)\right)$, where $\tau \in U^{(1)}$, $p \in \Pi_P$, $k \geq 0$.

Proof. Let prove only one side of implication, the other side can be proven similarly. We prove by induction. For $k = 0$, $T^k_P = \emptyset$ and the result obviously holds. Thus consider $k \geq 1$. Suppose $T^k_P \models p(\tau)$. Then a grounding substitution θ exists for a clause S of the form $p(t) \leftarrow p_1(t_1), \ldots, p_m(t_m) \in P$ such that $p(t)\theta = p(\tau)$ and $p_1(t_1)\theta, \ldots, p_m(t_m)\theta \in T^{k-1}_P$. By induction it follows that for every $i = 1, \ldots, m$, $T^{k-1}_{P'} \models p^T_i(t_i, s^{k-1}(0))\theta$.

As corresponding clause $S' \in P'$ is $p^T(t, s(z)) \leftarrow p^T_1(t_1, z), \ldots, p^T_m(t_m, z)$, and $\sigma = \theta \cup \{z/s^{k-1}(0)\}$ is a grounding substitution for S', it follows that $T_{P'}(T^{k-1}_{P'}) \models p^T(t, s(z))\sigma$. Hence, $T^k_{P'} \models p^T\left(\tau, s^k(0)\right)$. □

Lemma 3.6. Let P be a monadic program and P' be the program obtained from P by program transformation. Then $P \models p(\tau)$, if and only if $P' \models p^T(\tau, s^h(0))$, where $\tau \in U^{(1)}$, $p \in \Pi_P$ and $h = constraint(P, p(\tau))$.

Proof.

- Suppose $P \models p(\tau)$. By Lemma 3.3, $T^h_P \models p(\tau)$. Consequently, $T^h_{P'} \models p^T\left(\tau, s^h(0)\right)$ (Lemma 3.5). Whence, by Lemma 3.4, $P' \models p^T(\tau, s^h(0))$.
- Suppose $P' \models p^T(\tau, s^h(0))$. By Lemma 3.4, $T^h_{P'} \models p^T\left(\tau, s^h(0)\right)$. Then the Lemma 3.5 will imply $T^h_P \models p(\tau)$. Hence, by Lemma 3.3, $P \models p(\tau)$. □

Proof of the Theorem 3.1. Immediate from Lemma 3.6. □

Theorem 3.2.*(Termination).* Let P be a monadic program and $G \in \Delta(P)$. Let also P' be the program obtained from P by program transformation, and G' be the goal obtained from G by goal transformation. Then an SLD-tree of P' and G' is finite.

To prove Theorem 3.2, we will use the following notions and results.

Definition 3.1. *(Level Mapping).* A *level mapping* is a function $||: B \rightarrow \rightarrow N$ from the Herbrand base to the set of natural numbers N. For an atom $A \in B$, $|A|$ denotes the level of A.

Definition 3.2. *(Recurrency).* A clause $A \leftarrow B_1, \ldots, B_m$ is recurrent (wrt $||$) if for every grounding substitution θ, $|A\theta| > |B_i\theta|$ for all $i = 1, \ldots, m$. A program P is recurrent (wrt $||$) if every clause in P is recurrent (wrt $||$).

For a variable-free goal G of the form $\leftarrow C_1,\ldots,C_k$, $|G|$ denotes the (finite) multiset consisting of the natural numbers $|C_1|,\ldots,|C_k|$. Let N be the set of natural numbers. A multiset ordering over N is an ordering of finite multisets of natural numbers such that a multiset X is smaller than a multiset Y, if X can be obtained from Y by replacing one or more elements in Y by any (finite) number of natural numbers, each of which is smaller than one of the replaced elements.

The next lemma follows from [3].

Lemma 3.7. Let P be a logic program which is recurrent with respect to a level mapping $||$ and G be a variable-free goal. Then, if Q is an SLD-resolvent of G and a clause from P, then the multiset $|Q|$ is smaller than $|G|$ in the multiset ordering.

Corollary 3.1. Every SLD-derivation of a recurrent program and a variable-free goal is finite.

Proof. Immediate, as the multiset ordering over N is well-founded [8,9]. □

Proof of the Theorem 3.2. Since SLD-trees are finitely branching, by Kenig's Lemma, "SLD-tree of P' and G' is finite" is equivalent to stating that every SLD-derivation for P' and G' is finite. As G' is variable-free, it follows from Corollary 3.1, that for proving the finiteness of any SLD-derivation of P' and G', it is enough to define level mapping function $||$ and to show that P' is recurrent wrt that function. For an atom $p^T(\tau_1,\tau_2)\in B$, let us define the level mapping function $||$ as $\left|p^T(\tau_1,\tau_2)\right| = depth(\tau_2)$.

Let us prove the recurrency of each clause $S'\in P'$ wrt level mapping defined above. S' has a form $p^T(t,s(z))\leftarrow p_1^T(t_1,z),\ldots p_m^T(t_m,z)$. Let $i\in\{1,\ldots,m\}$ and θ be a grounding substitution for S'. Then

$$|p^T(t,s(z))\theta| = depth(s(z)\,\theta) = depth(z\theta)+1 > depth(z\theta) = \left|p_i^T(t_i,z)\,\theta\right|.$$

It follows that all clauses of P' are recurrent and, hence, P' is recurrent. □

Received 24.12.2013

REFERENCES

1. **Gurevich Yu.Sh.** The Decision Problem for the Logic of Predicates and of Operations. // Algebra and Logic, 1969, v. 8, № 3, p. 160–174.
2. **Cavedon L.** Acyclic Logic Programs and the Completeness of SLDNF-Resolution. // Theoretical Comput., 1991, v. 86, p. 81–92.
3. **Bezem M.A.** Strong Termination of Logic Programs. // Journal of Logic Programming, 1993, v. 15, № (1 & 2), p. 79–97.
4. **Lloyd J.W.** Foundations of Logic Programming. Springer-Verlag, 1984.
5. **Nilsson U., Maluszski J.** Logic, Programming and PROLOG (2-nd ed.). John Wiley & Sons Inc., 1995.
6. **Matos A.B.** Monadic Logic Programs and Functional Complexity. // Theoretical Computer Science, 1997, v. 176, p. 175–204.
7. **Khachatryan S.** On the Optimization of Variable-Free Logic Programs. In Proceedings of CSIT, 2011, p. 50-51.
8. **Dershowitz N.** Termination of Rewriting. // J. Symbolic Comput., 1987, v. 3, p. 69–116.
9. **Vereshchagin N., Shen A.** Basic Set Theory. AMS, 2002.