ГОДИШНИК НА СОФИЙСКИЯ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“

ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА
Книга 1 — Математика
Том 84, 1990

ANNUAIRE DE L'UNIVERSITE DE SOFIA „ST. KLIMENT OHRIDSKI“

FACULTE DE MATHEMATIQUES ET INFORMATIQUE
Livre 1 — Mathématiques
Tome 84, 1990

HORN CLAUSE PROGRAMS ON ABSTRACT STRUCTURES WITH PARAMETERS*

EXTENDED ABSTRACT

IVAN N. SOSKOV

Иван Н. Сосков. ПРОГРАММЫ С ХОРНОВЫМИ КЛАУЗАМИ НА АБСТРАКТНЫЕ СТРУКТУРЫ С ПАРАМЕТРАМИ

В статье вводится понятие абстрактных структур с параметрами. Параметрами являются подмножества области определения структур, которые трактуются как эффективно нумеруемые, а не как полувычислимые множества. С помощью этих структур определяется семантика программ с Хорновыми клаузами. Таким образом получаемый язык программирования показывается, что замкнут относительно рекурсии и имеет максимальную выразительную силу по отношению ко всем языкам программирования, обладающими некоторыми теоретико-модельными условиями.

Ivan N. Soskov. HORN CLAUSE PROGRAMS ON ABSTRACT STRUCTURES WITH PARAMETERS

In the paper abstract structures with parameters are introduced. The parameters are subsets of the domain of a structure which are treated as effectively enumerable rather than semicomputable sets. Semantics of Horn clause programs on such structures is defined. The obtained this way programming language is shown to be closed under recursion and possessing maximal expressive power with respect to all programming languages satisfying cetrain model-theoretic conditions.

0. INTRODUCTION

In this paper we present a generalized version of the declarative semantics of the Horn clause programs and use it to study the more complicated Horn clause

* Research partially supported by the Ministry of Science and Higher Education, contract No 933.

programs with parameters.

The usual minimal model semantics of van Emden and Kowalski [1] is not appropriate to study the respective notion of Horn clause computability. To compare Horn clause programs with other commonly studied programming languages (classes of program schemes) we need a concept of Horn clause computability, that applies to any first order structure. In [2] and [3] two concepts of Horn clause computability on first order structures have been defined, which seem to be equivalent. The first one is based on the usual declarative semantics, while the second is a generalization of the usual procedural semantics. In both approaches the underlined functions and predicates of a first order structure are considered as built-in functions and predicates, and the objects computable by means of Horn clause programs are subsets of the domain of the structure. In both papers the obtained concepts of Horn clause computability have been compared with some other concepts of computability and are proved to be stronger or at least equal to them.

Furthermore, it is shown in [4] that each programming language, satisfying certain natural conditions, is uniformly translatable into the language of the Horn clause programs.

One may suppose that having semantics of the Horn clause programs on arbitrary first order structures, we automatically obtain the semantics of Horn clause programs with parameters (Horn clause modules) on such structures. Indeed, it seems promising when given a structure $\mathfrak{A}$ and a subset A of the domain of $\mathfrak{A}$, to define the semantics of a Horn clause program P using A as a parameter, to be equal to the semantics of P on the extended structure $\langle\mathfrak{A}, A\rangle$, where A is a new built one in predicate. Unfortunately, this approach is not satisfactory because the obtained notion of computability is not transitive, unless the equality relation is among the underlined predicates of $\mathfrak{A}$. Namely, one can construct a structure $\mathfrak{A}$ and a Horn clause program P, such that if A denotes the set computable by means of P on $\mathfrak{A}$, then the Horn clause computable sets on $\langle\mathfrak{A}, A\rangle$ do not coincide with the Horn clause computable sets on $\mathfrak{A}$.

To explain this, let us recall that there exist two kinds of effectively computable sets — the semi-computable sets and the effectively enumerable ones. Intuitively, a set W is semi-computable if there exists a computational process, which stops and gives a positive answer iff its argument belongs to W. This is the way in which we treat the built in predicates. A set W is effectively enumerable if there exists a computational process, which generates the elements of W in the course of the computation. A typical example of this kind of sets are the sets computable by means of Horn clause programs.

In the classical case of computability on the natural numbers both classes of sets coincide with the recursively enumerable sets. In general, the semi-computable sets are a proper subclass of the effectively enumerable ones. In fact the effectively enumerable sets on a structure $\mathfrak{A}$ coincide with the semi-computable on $\mathfrak{A}$ sets if and only if the equality relation is semi-computable.

So, we obtain that the discussed above approach to the semantics of the Horn clause programs with parameters is reasonable only for structures, on which the effectively enumerable sets coincide with the semi-computable ones, i. e. for structures on which the equality is semi-computable.

Among all attempts to obtain semantics of Horn clause programs with parameters the most satisfactory one belongs to Fitting [5]. His approach is a particular case of the one described above, applied to the structure $\mathfrak{A} = (\mathbf{N};\ G_S)$, where $\mathbf{N}$ is the set of all natural numbers and G_S is the graph of the successor function. In this case the equality relation is semi-computable, because for any natural x and y, $x = y$ iff there exists a z, such that $(x, z) \in G_S$ and $(y, z) \in G_S$.

To obtain an appropriate semantics of the Horn clause programs with parameters on arbitrary first order structures, we introduce here first order structures with parameters, where the parameters are subsets of the domain of the structure, which are treated as effectively enumerable sets rather than semi-computable ones. After that we define semantics of the Horn clause programs on such structures. The obtained in this way programming language has some nice properties. First of all, it has a greater expressive power compared to all programming languages, satisfying certain natural conditions. This fact allows us to establish that it is closed with respect to least fixed points and that the respective notion of computability is transitive.

1. HORN CLAUSE PROGRAMS ON ABSTRACT STRUCTURES

Let $\mathcal{L} = (c_1, \ldots, c_r, f_1, \ldots, f_n, T_1, \ldots, T_k)$ be a fixed first order language, where $c_1, \ldots, c_r$ are constant symbols, $f_1, \ldots, f_n$ are functional symbols and $T_1, \ldots, T_k$ — predicate symbols. Let us suppose that each f_i is a_i-ary and each T_j is b_j-ary. Here each of r, n, k may be equal to 0.

A first order structure of the language $\mathcal{L}$ is a $r+n+k+1$-tuple $\mathfrak{A} = (A, p_1, \ldots, p_r, \theta_1, \ldots, \theta_n, \Sigma_1, \ldots, \Sigma_k)$, where A — the domain of $\mathfrak{A}$ — is an arbitrary non-empty set of objects, $p_1, \ldots, p_r$ are elements of A, each θ_i is an a_i-ary function on A and each Σ_j is a subset of A^{b_j}.

In what follows we shall consider only structures of the language $\mathcal{L}$ with denumerable (finite or countable) domains. Given a structure $\mathfrak{A}$, we shall denote the domain of $\mathfrak{A}$ by $|\mathfrak{A}|$.

We assume that the reader is familiar with the basic syntactic notions as terms, *atomic formulae (atoms), etc.* As usual, ground terms and ground atoms are called respectively terms and atoms without variables.

A *Horn clause program* is an ordered pair $\langle P,\ H\rangle$, where H — the goal relation — is a predicate symbol, not belonging to $\{T_1, \ldots, T_k\}$, and P is a finite conjunction $F_1 \& \ldots \& F_l$ of universal closures of Horn clauses, i. e. each F_i is in the form $\forall X_1 \ldots \forall X_q\ (\Pi \vee \neg\Pi_1 \vee \ldots \vee \neg\Pi_m)$, where $m \geqq 0$ and all Π, Π_1, $\ldots$, Π_m are atoms with variables among X_1, $\ldots$, X_q.

The constants, the functional and the predicate symbols, which occur in P, constitute the first order language $\mathcal{L}_P$ of P. We shall always assume that $\mathcal{L}_P$ is consistent with $\mathcal{L}$, i. e. if $\mathcal{L}_P$ and $\mathcal{L}$ have common symbols, then these symbols are of the same arity and play the same role in both languages.

To define the semantics of a Horn clause program $\langle P, H\rangle$ on the structure $\mathfrak{A} = (A, p_1, \ldots, p_r, \theta_1, \ldots, \theta_n, \Sigma_1, \ldots, \Sigma_k)$, we need a first order theory $\partial(\mathfrak{A})$, known as the diagram of $\mathfrak{A}$. By means of $\partial(\mathfrak{A})$ we describe the underlined functions and predicates of $\mathfrak{A}$.

A new formal constant k_s is introduced for each element s of A. The constants k_s, $s \in A$, are called names for the elements of A. Let $K = \{k_s : s \in A\}$. We shall assume that none of the elements of K occurs in P. Let $\mathcal{L}_K$ be the extension of $\mathcal{L}$ with the constants of K.

Now define the diagram $\partial(\mathfrak{A})$ of $\mathfrak{A}$, based on K, to be the set of all ground atoms of $\mathcal{L}_K$ which are true on $\mathfrak{A}$.

Let $\mathcal{T}_K$ denote the set of all ground terms of the language $\mathcal{L}_K$. If $\tau \in \mathcal{T}_K$, then by $\tau_{\mathfrak{A}}$ we shall denote the value of τ on $\mathfrak{A}$.

Let H be a-ary. Define the subset W of A^a by the equivalence

$$(s_1, \ldots, s_a) \in W \iff \exists\tau^1 \ldots \exists\tau^a\Big(\tau^1 \in \mathcal{T}_K \,\&\, \ldots \,\&\, \tau^a \in \mathcal{T}_K \,\&\, \tau^1_{\mathfrak{A}} = s_1 \,\&\, \ldots$$
$$\&\, \tau^a_{\mathfrak{A}} = s_a \,\&\, \partial(\mathfrak{A}) \cup \{P\} \vdash H\left(\tau^1, \ldots, \tau^a\right)\Big).$$

Here the sign "$\vdash$" means derivability in the sense of the first order predicate calculus.

Since no one of the elements of K occurs in P, the set W does not depend on the choice of K.

Define the semantics $\mathcal{P}(\langle P, H\rangle, \mathfrak{A})$ of $\langle P, H\rangle$ on $\mathfrak{A}$ to be equal to W.

Notice that if we know the diagram $\partial(\mathfrak{A})$, then we have an effective way to generate all elements of W. So we see that W is effectively enumerable on $\mathfrak{A}$. On the other hand, since the equality is not assumed to be computable on $\mathfrak{A}$, we can not decide effectively whether an a-tuple $(s_1, \ldots, s_a)$ belongs to W.

Now we shall show that our semantics agrees with the minimal model semantics of van Emden and Kowalski.

Let P be a finite conjunction of universal closures of Horn clauses. Let $\mathcal{L}_P = (c_1, \ldots, c_r, f_1, \ldots, f_n, H_1, \ldots, H_k)$. Let A_P be the Herbrand universe of $\mathcal{L}_P$ and let M_P be the least Herbrand model of P. In other words, M_P is the set of all ground atoms Π of the language $\mathcal{L}_P$, such that $P \vdash \Pi$.

Consider the first order structure $\mathfrak{A}_P = (A_P, c_1, \ldots, c_r, \varphi_1, \ldots, \varphi_n)$ of the language $(c_1, \ldots, c_r, f_1, \ldots, f_n)$, where each φ_i is an a_i-ary function on A_P defined by $\varphi_i(\tau_1, \ldots, \tau_{a_i}) = f_i(\tau_1, \ldots, \tau_{a_i})$.

Proposition 1. *Let $1 \leqq j \leqq k$. Then*

$$\left(\tau_1, \ldots, \tau_{b_j}\right) \in \mathcal{P}(\langle P, H_j\rangle, \mathfrak{A}_P) \iff H_j\left(\tau_1, \ldots, \tau_{b_j}\right) \in M_P.$$

2. HORN CLAUSE COMPUTABILITY

In this section we shall examine Horn clause computability from the viewpoint of a general theory of computability on abstract structures.

Let $\mathcal{A}$ be a class of denumerable abstract structures. We want to treat $\mathcal{A}$ as an abstract data type and, hence, we have to suppose some additional properties of $\mathcal{A}$. The obvious requirement is that $\mathcal{A}$ should be closed under isomorphisms. Since isomorphisms preserve the equality, we shall suppose something more. Namely, we shall suppose that $\mathcal{A}$ is closed under strong homomorphisms. Strong homomorphisms are isomorphisms, which need not to be injective. More precisely, if $\mathfrak{A} = (A, p_1, \ldots, p_r, \theta_1, \ldots, \theta_n, \Sigma_1, \ldots, \Sigma_k)$ and $\mathfrak{B} = (B, q_1, \ldots, q_r, \varphi_1, \ldots, \varphi_n, \sigma_1, \ldots, \sigma_k)$ are structures, then the surjective mapping x of A onto B is a *strong homomorphism* from $\mathfrak{A}$ to $\mathfrak{B}$ if the following conditions hold:

(i) $$\varkappa(p_i) = q_i, \quad i = 1, \ldots, r;$$

(ii) $$\varkappa(\theta_i(s_1, \ldots, s_{a_i})) = \varphi_i(\varkappa(s_1), \ldots, \varkappa(s_{a_i}))$$

for all $s_1, \ldots, s_{a_i}$ of A, $i = 1, \ldots, n$;

(iii) $$(s_1, \ldots, s_{b_j}) \in \Sigma_j \iff (\varkappa(s_1), \ldots, \varkappa(s_{b_j})) \in \sigma_j$$

for all $s_1, \ldots, s_{b_j}$ of A, $j = 1, \ldots, k$.

The class $\mathcal{A}$ is *closed under strong homomorphisms* if whenever $\mathfrak{B}$ belongs to $\mathcal{A}$, $\mathfrak{A}$ is a structure and there exists a strong homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$, then $\mathfrak{A} \in \mathcal{A}$.

Each closed under strong homomorphisms class of denumerable structures will be called for short *abstract data type* (ADT).

There are many natural examples of ADT. Consider, for example, the class of all structures or the class of all denumerable models of T, where T is a first order theory of the language $\mathcal{L}$ without equality.

Let us fix an ADT $\mathcal{A}$. A *programming language* on $\mathcal{A}$ is an ordered triple $L = \langle \mathcal{D}, \rho, \mathcal{S} \rangle$, where $\mathcal{D}$ is a denumerable set of objects — the syntactic descriptions of the programs of L, ρ — the arity function — is a mapping of $\mathcal{D}$ into $\mathbf{N} \setminus \{0\}$ and $\mathcal{S}$ — the semantics of the programs in L — is a mapping of $\mathcal{D} \times \mathcal{A}$, such that if $d \in \mathcal{D}$ and $\mathfrak{A} \in \mathcal{A}$, then $\mathcal{S}(d, \mathfrak{A})$ is equal to the object computable by means of the program d on the structure $\mathfrak{A}$. This object is typically a partial function or a set. Here we shall suppose that $\mathcal{S}(d, \mathfrak{A})$ is a subset of $|\mathfrak{A}|^{\rho(d)}$.

There are at least two natural conditions which should satisfy each programming language L on $\mathcal{A}$. First of all it should be in some sense effective.

The programming language $L = \langle \mathcal{D}, \rho, \mathcal{S} \rangle$ is said to be *effective* on $\mathcal{A}$ if for each $d \in \mathcal{D}$, the semantics of d is uniformly effective on all structures $\mathcal{B}$ of $\mathcal{A}$, such that $|\mathcal{B}| = \mathbf{N}$. In other words, L is effective iff for each d of $\mathcal{D}$, there exists an enumeration operator Γ such that whenever $\mathcal{B} \in \mathcal{A}$ and $\mathcal{B} = (\mathbf{N}, q_1, \ldots, q_r, \varphi_1, \ldots, \varphi_n, \sigma_1, \ldots, \sigma_k)$, then

$$(s_1, \ldots, s_{\rho(d)}) \in \mathcal{S}(d, \mathcal{A}) \iff \langle s_1, \ldots, s_{\rho(d)} \rangle \in \Gamma(\varphi_1, \ldots, \varphi_n, \sigma_1, \ldots, \sigma_k).$$

For the definition of the enumeration operators the reader may consult [6].

The second condition is related to the implementation independence property of the language L. Let $\mathfrak{A}$ and $\mathfrak{B}$ be elements of $\mathcal{A}$ and suppose that $\varkappa$ is a strong homomorphism from $\mathfrak{A}$ to $\mathcal{B}$. Let $d \in \mathcal{D}$ and let $W_{\mathfrak{A}} = \mathcal{S}(d, \mathfrak{A})$ and $W_{\mathfrak{B}} = \mathcal{S}(d, \mathfrak{B})$. Suppose for simplicity that $\rho(d) = 1$. Now the implementation independence property of L can be described by either of the following two conditions:

(1) For all $s \in |\mathfrak{A}|$, $s \in W_{\mathfrak{A}} \iff \varkappa(s) \in W_{\mathfrak{B}}$,
(2) $W_{\mathfrak{B}} = \{\varkappa(s) : s \in W_{\mathfrak{A}}\}$.

Clearly, (2) follows from (1). On the other hand, since $\varkappa$ need not be injective, we can not argue that (1) follows from (2).

The condition (1) is appropriate if the sets $W_{\mathfrak{A}}$ and $W_{\mathfrak{B}}$ are supposed to be semi-computable. If they are effectively enumerable, then we may expect only the weaker condition (2).

The programming language L is *e-invariant* on $\mathcal{A}$ if whenever $d \in \mathcal{D}$, $\mathfrak{A}$ and $\mathfrak{B}$ are elements of $\mathcal{A}$ and $\varkappa$ is a strong homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$, then

$$\mathcal{S}(d,\mathfrak{B}) = \left\{\left(\varkappa(s_1),\ldots,\varkappa\left(s_{\rho(d)}\right)\right) : \left(s_1,\ldots,s_{\rho(d)}\right) \in \mathcal{S}(d,\mathfrak{A})\right\}.$$

Let $LP = \langle \mathcal{HC}, \rho_0, \mathcal{P}\rangle$, where $\mathcal{HC}$ consists of all Horn clause programs, $\rho_0(\langle P, H\rangle)$ is equal to the arity of H and $\mathcal{P}$ is the semantics of the Horn clause programs defined in the previous section.

Proposition 2. *The programming language LP is effective and e-invariant on each* ADT.

The following theorem proved in [4] describes the main property of the Horn clause computability.

Let $\mathcal{A}$ be an ADT and let $L_1 = \langle \mathcal{D}_1, \rho_1, \mathcal{S}_1\rangle$ and $L_2 = \langle \mathcal{D}_2, \rho_2, \mathcal{S}_2\rangle$ be programming languages on $\mathcal{A}$. Then L_1 is translatable into L_2 on $\mathcal{A}$ (cf. [7]), in symbols $L_1 \leqq {}_{\mathcal{A}}L_2$, iff for each $d_1 \in \mathcal{D}_1$ there exists a $d_2 \in \mathcal{D}_2$, such that $\rho_1(d_1) = \rho_2(d_2)$, and for all $\mathfrak{A} \in \mathcal{A}$, $\mathcal{S}_1(d_1,\mathfrak{A}) = \mathcal{S}_2(d_2,\mathfrak{A})$.

Theorem 1. *Let $\mathcal{A}$ be an abstract data type. Let L be an effective and e-invariant programming language on $\mathcal{A}$. Then $L \leqq {}_{\mathcal{A}}LP$.*

This result should be compared with the generalized Church thesis, formulated in [3].

Other results, concerning universal programming languages satisfying certain natural model-theoretic conditions, can be found in [8] and [4].

3. ABSTRACT STRUCTURES WITH PARAMETERS

In this main section of the paper we shall introduce the abstract structures with parameters and define semantics of the Horn clause programs on such structures.

Let $S_1, \ldots, S_l, \ldots$ be a sequence of new distinct predicate symbols intended to denote parameters. For the sake of simplicity we shall consider here only unary parameters. So we shall suppose that $S_1, \ldots, S_l, \ldots$ are unary. However, all definitions and results can be easily generalized for parameters of arbitrary finite arity.

An *abstract structure with parameters* is an ordered tuple $\mathfrak{A}^* = (A_1,\ldots,A_l)$, $l \geqq 0$, where $\mathfrak{A}$ is a structure of the language $\mathcal{L}$ and $A_1, \ldots, A_l$ are subsets of $|\mathfrak{A}|$.

For every structure with parameters $\mathfrak{A}^*$ let $\nu(\mathfrak{A}^*)$ be the number of the parameters of $\mathfrak{A}^*$. Two abstract structures $\mathfrak{A}^*$ and $\mathfrak{B}^*$ are of the same similarity type if $\nu(\mathfrak{A}^*) = \nu(\mathfrak{B}^*)$.

If $\mathfrak{A}^* = (\mathfrak{A}, A_1,\ldots,A_l)$ is a structure with parameters and W is a subset of $|\mathfrak{A}|$, then by $(\mathfrak{A}^*, W)$ we shall denote the structure $(\mathfrak{A}, A_1,\ldots,A_l, W)$.

The parameters and the underlined predicates play different roles in the definition of the strong homomorphisms between structures with parameters.

Let $\mathfrak{A}^* = (\mathfrak{A}, A_1, \ldots, A_l)$ and $\mathfrak{B}^* = (\mathfrak{B}, B_1, \ldots, B_l)$ be two structures with parameters of the same similarity type. The surjective mapping $\varkappa$ of $|\mathfrak{A}|$ onto $|\mathfrak{B}|$ is called *strong homomorphism* from $\mathfrak{A}^*$ to $\mathfrak{B}^*$ iff $\varkappa$ is a strong homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$ and for $i = 1, \ldots, l$, $B_i = \{\varkappa(s) : s \in A_i\}$.

We generalize the notion of ADT, defining an abstract data type to be a closed under strong homomorphisms class of denumerable structures with parameters of the same similarity type.

Further on the notions of effectiveness and e-invariance of a programming language on an ADT are generalized in an obvious way.

From now on we shall consider only Horn clause programs $\langle P, H\rangle$, such that $H \notin \{T_1, \ldots, T_k, S_1, \ldots, S_l, \ldots\}$ and if for some l the symbol S_l occurs in P, then it is used as an unary predicate.

Now we shall define the semantics of a Horn clause program $\langle P, H\rangle$ on a structure with parameters $\mathfrak{A}^* = (\mathfrak{A}, A_1, \ldots, A_l)$ in a way providing that the respective programming language is effective and e-invariant on each ADT.

For each element s of $|\mathfrak{A}|$ we introduce $l+1$ distinct names k_s^0, k_s^1, $\ldots$, k_s^l. Let $K_0 = \{k_s^0 : s \in |\mathfrak{A}|\}$, $\ldots$, $K_l = \{k_s^l : s \in |\mathfrak{A}|\}$ and let $K = K_0 \cup K_1 \cup \ldots \cup K_l$. We shall suppose that the sets of names are chosen so that no one of the elements of K occurs in P. Let $\mathcal{L}_K$ be the extension of $\mathcal{L}$ with the constants of K and let $\mathcal{T}_K$ be the set of all ground terms of $\mathcal{L}_K$. Let $\partial(\mathfrak{A})$ be the set of all ground atoms of $\mathcal{L}_K$ which are true on $\mathfrak{A}$.

For i, $1 \leqq i \leqq l$, let $\partial(A_i) = \{S_i(k) : k \in K_i$ and k is the name of an element of $A_i\}$.

Let $\partial(\mathfrak{A}^*) = \partial(\mathfrak{A}) \cup \partial(A_1) \cup \ldots \cup \partial(A_l)$.

Notice that in the definition of $\partial(\mathfrak{A}^*)$ the underlined predicates and the parameters are not treated in an equal manner. For example, suppose that the underlined function θ_i applied to some s gives t. Suppose that $t \in \Sigma_j$ and $t \in A_m$, where Σ_j is an underlined predicate and A_m is a parameter. Then both $T_j(k_t^m)$ and $T_j(f_i(k_s^m))$ are elements of $\partial(\mathfrak{A}^*)$. On the other hand, $S_m(k_t^m) \in \partial(\mathfrak{A}^*)$ but $S_m(f_i(k_s^m))$ does not belong to $\partial(\mathfrak{A}^*)$. The picture changes if the equality relation is among the underlined predicates. In such a case $f_i(k_s^m) = k_t^m \in \partial(\mathfrak{A}^*)$ and, hence, $\partial(\mathfrak{A}^*) \vdash S_m(f_i(k_s^m))$.

Suppose that the predicate symbol H is a-ary. Now the semantics $\mathcal{P}^*(\langle P, H\rangle, \mathfrak{A}^*)$ of $\langle P, H\rangle$ on $\mathfrak{A}^*$ is the subset W of $|\mathfrak{A}|^a$ defined by the equivalence:

$$(s_1, \ldots, s_a) \in W \iff \exists\tau^1 \ldots \exists\tau^a\Big(\tau^1 \in \mathcal{T}_K \;\&\; \ldots \;\&\; \tau^a \in \mathcal{T}_K \;\&\; \tau^1_{\mathfrak{A}} = s_1 \;\&\; \ldots$$
$$\& \; \tau^a_{\mathfrak{A}} = s_a \;\&\; \partial(\mathfrak{A}^*) \cup \{P\} \vdash H(\tau^1, \ldots, \tau^a)\Big).$$

It follows immediately from the definition that for structures $\mathfrak{A}$ without parameters $\mathcal{P}(\langle P, H\rangle, \mathfrak{A}) = \mathcal{P}^*(\langle P, H\rangle, \mathfrak{A})$.

Let LPP be the programming language $\langle \mathcal{HC}, \rho_l, \mathcal{P}^*\rangle$, where $\mathcal{HC}$ is the set of all Horn clause programs and ρ_0 is defined as in the previous section.

Proposition 3. *The programming language LPP is effective and e-invariant on each* ADT.

The following theorem generalizes Theorem 1 and and shows that our definition is in some sense the best possible one.

Theorem 2. *Let $\mathcal{A}$ be an* ADT *and let L be an effective and e-invariant programming language on $\mathcal{A}$. Then $L \leqq_{\mathcal{A}} LPP$.*

The proof of this theorem is long and technical and is omitted here.

As a first application of Theorem 2, we shall show that the respective Horn clause computability is transitive.

Let $l \geqq 0$ and let $\mathcal{A}$ be the class of all structures with parameters $\mathfrak{A}^*$ such that $\nu(\mathfrak{A}^*) = l$. Clearly $\mathcal{A}$ is an ADT.

Let $\langle P_0, H_0\rangle$ be a Horn clause program, where the predicate symbol H_0 is unary. For each $\mathfrak{A}^* \in \mathcal{A}$, set $W_{\mathfrak{A}} = \mathcal{P}^*(\langle P_0, H_0\rangle, \mathfrak{A}^*)$.

Theorem 3. *For each Horn clause program $\langle P, H\rangle$ there exists a Horn clause program $\langle Q, R\rangle$ such that for all $\mathfrak{A}^* \in \mathcal{A}$,*

$$\mathcal{P}^*(\langle P, H\rangle, (\mathfrak{A}^*, W_{\mathfrak{A}})) = \mathcal{P}^*(\langle Q, R\rangle, \mathfrak{A}^*).$$

P r o o f. Consider the programming language $L = \langle \mathcal{HC}, \rho_l, \mathcal{S}\rangle$ on $\mathcal{A}$, where for $\langle P, H\rangle \in \mathcal{HC}$ and $\mathfrak{A}^* \in \mathcal{A}$, $\mathcal{S}(\langle P, H\rangle, \mathfrak{A}^*) = \mathcal{P}^*(\langle P, H\rangle, (\mathfrak{A}^*, W_{\mathfrak{A}}))$.

Now we shall show that L is effective and e-invariant on $\mathcal{A}$. The effectiveness of L follows from the effectiveness of LPP and from the well known fact that the enumeration operators are closed under composition.

Let $\langle P, H\rangle$ be a Horn clause program and $\mathfrak{A}^*$ and $\mathfrak{B}^*$ be elements of $\mathcal{A}$. Suppose that $\varkappa$ is a strong homomorphism from $\mathfrak{A}^*$ to $\mathfrak{B}^*$. From the e-invariance of LPP follows that $\varkappa(W_{\mathfrak{A}}) = W_{\mathfrak{B}}$. Hence $\varkappa$ is a strong homomorphism from $(\mathfrak{A}^*, W_{\mathfrak{A}})$ to $(\mathfrak{B}^*, W_{\mathfrak{B}})$. From here, using once more the e-invariance of LPP, we obtain that

$$\varkappa(\mathcal{P}^*(\langle P, H\rangle, (\mathfrak{A}^*, W_{\mathfrak{A}}))) = \mathcal{P}^*(\langle P, H\rangle, (\mathfrak{B}^*, W_{\mathfrak{B}}))$$

and, hence, that

$$\varkappa(\mathcal{S}(\langle P, H\rangle, \mathfrak{A}^*)) = \mathcal{S}(\langle P, H\rangle, \mathfrak{B}^*).$$

Now applying Theorem 2, we obtain that $L \leqq_{\mathcal{A}} LPP$. ∎

The last theorem shows that we can eliminate the computable by means of Horn clause programs parameters in a uniform way.

4. HORN CLAUSE OPERATORS

Let $l \geqq 0$ and let $\mathcal{A}$ be the class of all structures with parameters $\mathfrak{A}^*$ such that $\nu(\mathfrak{A}^*) = l$.

Let $\mathfrak{A}^* \in \mathcal{A}$ and let $\langle P, H\rangle$ be a Horn clause program, where H is unary. Define the mapping $\Gamma_{P,H}$ of the subsets of $|\mathfrak{A}^*|$ into the subsets of $|\mathfrak{A}^*|$ by the equality $\Gamma_{P,H}(W) = \mathcal{P}^*(\langle P, H\rangle, (\mathfrak{A}^*, W))$. It follows easily from the definition of $\mathcal{P}^*$ that the operator $\Gamma_{P,H}$ is compact, i. e.

$$s \in \Gamma_{P,H}(W) \iff \exists D\,(D \subseteq W \;\&\; D \text{ is finite} \;\&\; s \in \Gamma_{P,H}(D)).$$

From here, applying the Knaster–Tarski theorem, we obtain that $\Gamma_{P,H}$ has a least fixed point W_0 and $W_0 = \bigcup_{k=0}^{\infty} \Gamma^k_{P,H}(\varnothing)$. We denote this least fixed point by $\mu W.\mathcal{P}^*(\langle P, H\rangle, (\mathfrak{A}^*, W))$.

Now we shall show that the least fixed point of each Horn clause operator is computable by means of Horn clause programs. In fact we have something more:

Theorem 4 (First Recursion Theorem for Horn clause operators). *For each Horn clause program $\langle P, H\rangle$ there exists a Horn clause program $\langle P^*, H^*\rangle$ such that for all $\mathfrak{A}^* \in \mathcal{A}$,*

$$\mu W.\mathcal{P}^*(\langle P, H\rangle, (\mathfrak{A}^*, W)) = \mathcal{P}^*(\langle P^*, H^*\rangle, \mathfrak{A}^*).$$

P r o o f. Let L be the programming language $\langle \mathcal{D}, \rho, \mathcal{S}\rangle$ on $\mathcal{A}$, where $\mathcal{D}$ consists of all Horn clause programs $\langle P, H\rangle$ such that H is unary, $\rho(d) = 1$ for $d \in \mathcal{D}$ and $\mathcal{S}(\langle P, H\rangle, \mathfrak{A}^*) = \mu W.\mathcal{P}^*(\langle P, H\rangle, (\mathfrak{A}^*, W))$. To prove the theorem it is sufficient to show that $L \leqq_{\mathcal{A}} LPP$.

We shall prove that L is effective and e-invariant on $\mathcal{A}$. Indeed, the effectiveness of L follows from the uniform version of the First Recursion Theorem for the enumeration operators.

To prove the e-invariance of L, suppose that $\langle P, H\rangle \in \mathcal{D}$, let $\mathfrak{A}^*$ and $\mathfrak{B}^*$ be elements of $\mathcal{A}$ and let $\varkappa$ be a strong homomorphism from $\mathfrak{A}^*$ to $\mathfrak{B}^*$. Let us define the sequences $W^n_{\mathfrak{A}}$ and $W^n_{\mathfrak{B}}$ of sets in the following way:

$W^0_{\mathfrak{A}} = W^0_{\mathfrak{B}} = \varnothing$;

$W^{n+1}_{\mathfrak{A}} = \mathcal{P}^*(\langle P, H\rangle, (\mathfrak{A}^*, W^n_{\mathfrak{A}}))$ and $W^{n+1}_{\mathfrak{B}} = \mathcal{P}^*(\langle P, H\rangle, (\mathfrak{B}^*, W^n_{\mathfrak{B}}))$.

Now using the e-invariance of LPP, we obtain by induction on n that $\varkappa(W^n_{\mathfrak{A}}) = W^n_{\mathfrak{B}}$, $n = 0, 1, \ldots$ Hence

$$\varkappa(\mu W.\mathcal{P}^*(\langle P, H\rangle, (\mathfrak{A}^*, W))) = \mu W.\mathcal{P}^*(\langle P, H\rangle, (\mathfrak{B}^*, W)).$$

By this the e-invariance of L is proved. From here, by Theorem 2, it follows $L \leqq_{\mathcal{A}} LPP$. ∎

REFERENCES

1. V a n E m d e n, M. H., R. A. K o w a l s k i. The semantics of the predicate logic as a programming language. — J. ACM, 23(4), 1976, 733–742.
2. S o s k o v, I. N. On the computational power of the logic programs. In: Heyting'88: Mathematical Logic (ed. P. P. Petkov). Plenum Press, New York and London, 1989, 117–137.
3. T u c k e r, J. V., J. I. Z u c k e r. Horn programs and semicomputable relations on abstract structures. — Proceedings of the 16th International Colloquium on Automata, Languages and Programming, July 1989, Stresa. Springer Lecture Notes in Computer Science, Berlin, 1989.
4. S o s k o v, I. N. Maximal concepts of computability and maximal programming languages (submitted).
5. F i t t i n g, M. Enumeration operators and modular logic programming. — J. Logic Programming, 4, 1987, 11–21.
6. R o g e r s, H., J r. Theory of recursive functions and effective computability. McGraw-Hill Book Company, 1967.

7. Paterson, M., C. Hewitt. Comparative Schematology. MIT AI Lab Memo No 201, Nov., 1970.
8. Trakhtenbrot, B. A. *Recursive program schemes and computable functionals. In:* MFCS'76 (ed. A. Mazurkiewicz). — Lect. Notes in Comp. Sci., 45, Springer-Verlag, 1976, 137–151.

Received 4.04.1991