Siberian Mathematical Journal, vol. 59, *no.* 6, *pp.* 1014–1023, 2018

RECURSIVE TERMS IN SEMANTIC PROGRAMMING

S. S. Goncharov and D. I. Sviridenko UDC 510.2:510.3:510.6:51.8:004.8

Abstract: For constructing an enrichment of a language with restricted quantifiers, we extend the notion of term by the construction of the terms conditional and recursive with respect to lists. We show that the so-obtained extension of the language of formulas with restricted quantifiers over structures with hereditary finite lists is a conservative enrichment and the new terms are Δ-definable in the basic language.

DOI: 10.1134/S0037446618060058

Keywords: formula, term, restricted quantifier, Δ_0-formula, Δ-formula, Σ-formula, semantic programming, computability, computability over abstract structures, conditional term, recursive term, model, hereditary finite list, hereditary finite set

In [1], the notion of conditional term was defined and studied, which substantially increases the expressiveness of the language of semantic programming (see [2–7]) and, in particular, makes it possible to construct termal logical programs without increase of computational complexity with respect to the basic operations of the original computable model. This paper develops the concept of semantic programming so as to include, in particular, application to the problem of constructing systems of control of complex objects in which the control conditions depend on the type of the input data with the possibility of their representation as the formal constructions on the basis of logical structures.

One of the conceptual foundations of semantic programming is the Σ-definability of computability. Many works are devoted to this important notion. For example, Ershov's articles [8–10] contain a detailed study of Σ-definability for the case of superstructures over a model which consist of hereditary finite sets. Investigations of computability, based on Σ-definability on various abstract structures, are the contents of [11–15]. In [4, 5, 13], an attempt was made to construct computability theory via Σ-definability for the case of superstructures from hereditary finite lists. Such an approach is more natural from the practical point of view since it opens ways of specifying execution algorithms for logical programs by an efficiently controlled exhaustion of the elements of some set represented as a list. Conditional terms were introduced in [1] for increasing the efficiency of this algorithm in [1]. In the present article, we go on with the research in this direction and propose a termal construction significantly expanding the expressiveness of the language of semantic programming and increasing the efficiency of the execution algorithms for semantic programs. It is based on recursive terms defined by using only Δ_0-formulas. In the main methods and definitions, we follow [1, 4, 5, 10, 14].

Consider a model $\mathfrak{M}$ of signature σ and, in line with [4], define the superstructure of hereditary finite lists $HW(\mathfrak{M})$ for $\mathfrak{M}$ in which extend the signature σ to σ^* by adding the list functions of the language LISP: $\mathrm{head}(x)$, $\mathrm{cons}(x, y)$, $\mathrm{tail}(x, y)$, the constant nil, and the unary predicate $\mathscr{U}$ distinguishing the basic set of the basic structure $\mathfrak{M}$, the relations $x \in t$ "x is an element of a list t" and $x \sqsubseteq t$ "x is an initial sublist of a list t."

For constructing Δ_0- and Σ-formulas, semantic programming usually involves the two types of restrictions: those with respect to the elements of the lists $\forall x \in t$, $\exists x \in t$ and those with respect to the initial elements of the lists $\forall x \sqsubseteq t$ and $\exists x \sqsubseteq t$. Add the two more types of restricted quantifiers $\forall x = t$ and $\exists x = t$, which are in the same paradigm of restricted quantifiers and in which the usual equality of elements is used as partial order.

The authors were supported by the Russian Science Foundation (Grant 17–11–01176).

Novosibirsk. Translated from *Sibirskii Matematicheskii Zhurnal*, vol. 59, no. 6, pp. 1279–1290, November–December, 2018; DOI: 10.17377/smzh.2018.59.605. Original article submitted March 17, 2018.

These quantifiers correspond to the software construction of attribution. Note that here the main properties of Δ_0- and Σ-formulas do not change because the corresponding terms can be just inserted directly into a Δ_0- or Σ-formula at which a quantifier with equality acts and then this quantifier can be eliminated from consideration. However, for extending the notion of quantifier, we will need to use these quantifiers on applying the extended version of Δ_0- and Σ-formulas already with recursive terms for construction. Note that the added restricted quantifiers are expressible via the main restricted quantifiers and so we obtain a conservative extension for Δ_0- and Σ-formulas.

For describing the properties of the enrichment $HW(\mathfrak{M})$ of a model $\mathfrak{M}$ consider the class of terms $\mathrm{Term}(\sigma^*, V)$ with variables in V and the class of Δ_0- and Σ-formulas [3].

Extend the notion of term in signature σ^* by adding the class Term(Rec) of recursive terms of signature σ^* with variables in V and defining the classes of Δ_0(Rec)-formulas and Σ(Rec)-formulas of signature σ^* with variables in V by joint induction together with the new terms.

The induction base. Each term of signature σ^* with variables in V is a recursive term, Δ_0-formulas are Δ_0(Rec)-formulas, while Σ-formulas are Σ(Rec)-formulas.

The induction step. Let $t_0, t_1, \ldots, t_{n+1}$ be recursive terms and let $\varphi_0, \varphi_1, \ldots, \varphi_n$ be Δ_0(Rec)-formulas. Consider an expression t of the form

$$\mathrm{Cond}((t_0, t_1, \ldots, t_{n+1}), (\varphi_0, \varphi_1, \ldots, \varphi_n)).$$

It defines the term t by the scheme of the conditional definition and also the semantics of t in the model of hereditary finite lists in the form

$$t(\bar{v}) = \begin{cases} t_0(\bar{v}) & \text{if } \mathfrak{M} \models \varphi_0(\bar{v}), \\ t_1(\bar{v}) & \text{if } \mathfrak{M} \models \varphi_1(\bar{v}) \,\&\, \neg\varphi_0(\bar{v}), \\ \ldots, & \\ t_i(\bar{v}) & \text{if } \mathfrak{M} \models \varphi_i(\bar{v}) \,\&\, \neg\varphi_0(\bar{v}) \,\&\, \cdots \,\&\, \neg\varphi_{i-1}(\bar{v}), \\ \ldots, & \\ t_n(\bar{v}) & \text{if } \mathfrak{M} \models \varphi_n(\bar{v}) \,\&\, \neg\varphi_0(\bar{v}) \,\&\, \cdots \,\&\, \neg\varphi_{n-1}(\bar{v}), \\ t_{n+1}(\bar{v}) & \text{otherwise.} \end{cases}$$

The term t is recursive.

Consider another construction that extends the possibilities of defining new terms with the preservation of the possibility of determining their computational complexity and using logically defined constructions in constructing programs of logical type and also the ability for verification with securing boundedly many exhaustions in a predefined restricted domain.

Give the definition of term recursive with respect to lists. Let $g(\bar{v})$ and $h(\bar{v}, y, z, w)$ be recursive terms. Define the new recursive term t by recursion $\mathrm{Rec}(g, h)$ with respect to lists of recursive terms by defining the semantics of t as follows:

• $t(\bar{v}, \mathrm{nil}) = g(\bar{v})$ and $t(\bar{v}, \mathrm{cons}(y, z)) = h(\bar{v}, y, z, t(\bar{v}, y))$,

• for the elements of the basic model, i.e. the elements that are not lists, extend the definition by setting: if $\neg U(y)$ then $t(\bar{v}, y) = y$.

Moreover, the natural requirement is fulfilled that there are no other ways of constructing recursive terms.

Consider the construction of the extension of Δ_0- and Σ-formulas to Δ_0(Rec)- and Σ(Rec)-formulas respectively by adding to their definitions not only standard terms but also recursive terms.

If t and q are recursive terms then $t = q$ is a Δ_0(Rec)-formula. If $t_1, \ldots, t_n$ are recursive terms and P is an n-ary predicate symbol of the signature of our superstructure then $P(t_1, \ldots, t_n)$ is a Δ_0(Rec)-formula.

If φ and ψ are Δ_0(Rec)-formulas and t is a conditional term then the expressions $(\varphi\&\psi)$, $(\varphi \vee \psi)$, $(\varphi \to \psi)$, $\neg\varphi$, $(\forall x \in t)\varphi$, $(\forall x \sqsubseteq t)\varphi$, $(\forall x = t)\varphi$, $(\exists x \in t)\varphi$, $(\exists x \sqsubseteq t)\varphi$, $(\exists x = t)\varphi$ are Δ_0(Rec)-formulas.

Clearly, Δ_0-formulas are $\Delta_0(\mathrm{Rec})$-formulas. Call them *standard*, whereas if they contain a conditional operator or a recursive operator then refer to them as *nonstandard.*

Similarly, extend the notions of $\Sigma(\mathrm{Rec})$-formulas by induction on construction.

The induction base. If φ is a $\Delta_0(\mathrm{Rec})$-formula then φ is a $\Sigma(\mathrm{Rec})$-formula.

The induction step. If φ and ψ is a $\Sigma(\mathrm{Rec})$-formula and t is a recursive term then the expressions $(\varphi\&\psi)$, $(\varphi\vee\psi)$, $(\forall x\in t)\varphi$, $(\forall x\sqsubseteq t)\varphi$, $(\forall x=t)\varphi$, $(\exists x\in t)\varphi$, $(\exists x\sqsubseteq t)\varphi$, and $(\exists x=t)\varphi$ and also the expressions $(\exists x)\varphi$ are $\Sigma(\mathrm{Rec})$-formulas.

Note that Σ-formulas are $\Sigma(\mathrm{Rec})$-formulas.

By induction define the complexity of recursive terms and $\Delta_0(\mathrm{Rec})$-formulas. In what follows, the usual terms in the sense of the definitions of [9] will be called *standard*; and if, their construction include the construction of the definition of recursive or conditional term for this term or a subterm of this term; then such terms will be called *nonstandard*.

The induction base. If t is a standard term of signature σ then the complexity $r(t)$ of t is equal to 0. If φ is a Δ_0-formula then the complexity of φ is equal to 0.

The induction step. If a term t contains nonstandard recursive subterms then consider all possible cases:

Case 1. If $t=F(t_1,\ldots,t_n)$ and among the terms $t_1,\ldots,t_n$ there is a recursive nonstandard term then the complexity $r(t)$ of t is equal to

$$(\max\{r(t_1),\ldots,r(t_n)\}+1)(n+1).$$

Case 2. Suppose that t is obtained by the scheme of a conditional term; i.e.,

$$t=\mathrm{Cond}((t_0,t_1,\ldots,t_{n+1}),(\varphi_0,\varphi_1,\ldots,\varphi_n)),$$

where $t_0,\ldots,t_{n+1}$ are recursive terms and $\varphi_0,\ldots,\varphi_n$ are Δ_0^{**}-formulas. In this case, put

$$r(t)=(r(t_0)+r(t_1)+\cdots+r(t_{n+1})+(n+2))^2(\max\{r(\varphi_i)\mid 1\le i\le n+1\}+(n+2))^2.$$

Case 3. If t is obtained by the scheme of recursion with respect to lists from recursive terms g and h then the complexity $r(t)$ of t is equal to $r(g)+r(h)+2$.

Case 4. If φ is a $\Delta_0(\mathrm{Rec})$-formula but not a Δ_0-formula, then consider all cases of the definition of φ; namely, $(\varphi_1\,\&\,\varphi_2)$, $(\varphi_1\vee\varphi_2)$, $(\varphi_1\to\varphi_2)$, $\neg\varphi_1$, $(\forall v\in t)\varphi_1$, $(\exists v\in t)\varphi_2$, $(\forall v\sqsubseteq t)\varphi_1$, $(\exists v\subseteq t)\varphi_1$, and define the value $r(\varphi)$:

$$r((\varphi_1\,\&\,\varphi_2))=(r(\varphi_1)+r(\varphi_2))+1,\quad r((\varphi_1\vee\varphi_2))=(r(\varphi_1)+r(\varphi_2))+1,$$
$$r((\varphi_1\to\varphi_2))=(r(\varphi_1)+r(\varphi_2))+1,\quad r(\neg\varphi_1)=r(\varphi_1)+1,$$
$$r((\forall v=t)\varphi_1)=((r(\varphi_1)+1)\times(r(t)+1))+1,$$
$$r((\exists v=t)\varphi_1)=((r(\varphi_1)+1)\times(r(t)+1))+1,$$
$$r((\forall v\in t)\varphi_1)=((r(\varphi_1)+1)\times(r(t)+1))+1,$$
$$r((\exists v\in t)\varphi_1)=((r(\varphi_1)+1)\times(r(t)+1))+1,$$
$$r((\forall v\sqsubseteq t)\varphi_1)=((r(\varphi_1)+1)\times(r(t)+1))+1,$$
$$r((\exists v\sqsubseteq t)\varphi_1)=((r(\varphi_1)+1)\times(r(t)+1))+1.$$

In [1], there was shown that adding conditional terms to the standard terms in defining Δ_0- and Σ-formulas does not expand the expressiveness of Δ_0- and Σ-formulas in using only standard terms. However, the situation is completely different in the case of the construction of recursive terms. Note that the expressiveness of Σ does not change as compared to the usual Σ-formulas after adding recursive terms to their definition in this case either. In the case of Δ_0-formulas, adding recursive terms to their definition increases expressiveness but still they remain at the level of Δ_0-definable relations, i.e., such formulas and their negations are equivalent to Σ-formulas.

Theorem. (1) *There is an algorithm constructing from each* $\Delta_0(\mathrm{Rec})$*-formula* φ *some* Σ*-formulas* ψ_1 *and* ψ_2 *such that*

$$HW(\mathfrak{M}) \models (\forall \bar{v})(\varphi(\bar{v}) \Leftrightarrow \psi_1(\bar{v})), \quad HW(\mathfrak{M}) \models (\forall \bar{v})(\neg\varphi(\bar{v}) \Leftrightarrow \psi_2(\bar{v})),$$

i.e., each $\Delta_0(\mathrm{Rec})$*-formula defines a* Δ*-definable relation.*

(2) *There is an algorithm constructing from each* $\Sigma(\mathrm{Rec})$*-formula* φ *some* Σ*-formula* ψ *such that*

$$HW(\mathfrak{M}) \models (\forall \bar{v})(\varphi(\bar{v}) \Leftrightarrow \psi(\bar{v})).$$

Proof. Prove item (1).

The induction base. If $r(\varphi) = 0$ then φ does not contain nonstandard terms and we can take as ψ the Δ_0-formula φ itself and for $\neg\varphi$ we can take the Δ_0-formula $\neg\varphi$ itself.

The induction step: $n = r(\varphi) > 0$.

Let us first consider the case of atomic formulas.

Case 1. If an atomic formula does not contain nonstandard recursive terms then it and its negation are already Δ_0-formulas, and everything holds. Consider the cases of atomic formulas that contain at least one nonstandard recursive term.

Subcase 1.1. Suppose that φ has the form $t = q$, where at least one of the terms t and q is recursive and nonstandard.

Without loss of generality, by the symmetry of equality, we may assume that q is recursive and nonstandard. In this case, there are three possibilities open for q, i.e., $q = F(q_1, \ldots, q_k)$, or q is obtained by the recursive scheme, or by the scheme of list recursion.

Subcase 1.2. If $q = F(q_1, \ldots, q_k)$ then $t = q$ is equivalent to

$$(\exists z_1)(\exists z_2)\ldots(\exists z_k)\Big(\mathop{\&}_{1\le i\le k} z_i = q_i(\bar{v})\,\&\,t = F(z_1, z_2, \ldots, z_k)\Big).$$

Each formula $z_i = q_i(\bar{v})$ for $1 \le i \le k$ has smaller rank; and, by the induction hypothesis, it is equivalent to some Σ-formula ϕ_i for $1 \le i \le k$. Then, by equivalence, $t = q$ is equivalent to the Σ-formula

$$(\exists z_1)(\exists z_2)\ldots(\exists z_k)\Big(\mathop{\&}_{1\le i\le k} \phi_i\,\&\,t = F(z_1, z_2, \ldots, z_k)\Big).$$

The condition of the theorem is obtained for $t = q$.

Consider the negation $\neg t = q$. Obviously, the negation is equivalent to the formula

$$(\exists z_1)(\exists z_2)\ldots(\exists z_k)\Big(\mathop{\&}_{1\le i\le k} z_i = q_i(\bar{v})\,\&\,\neg t = F(z_1, z_2, \ldots, z_k)\Big).$$

Like in the case of equality, we infer that the latter is equivalent to the Σ-formula

$$(\exists z_1)(\exists z_2)\ldots(\exists z_k)\Big(\mathop{\&}_{1\le i\le k} \phi_i\&\neg t = F(z_1, z_2, \ldots, z_k)\Big).$$

Subcase 1.3. Consider the possibility of equality of terms in the case that $q(\bar{v})$ is obtained by the scheme of a recursive term; i.e.,

$$g = \mathrm{Cond}((g_0, t_1, \ldots, g_{n+1}), (\varphi_0, \varphi_1, \ldots, \varphi_n)).$$

In this case, the semantics of $q(\bar{v})$ is defined as follows:

$$q(\bar{v}) = \begin{cases} q_0(\bar{v}) & \text{if } HW(\mathfrak{M}) \models \varphi_0(\bar{v}), \\ q_1(\bar{v}) & \text{if } HW(\mathfrak{M}) \models \varphi_1(\bar{v})\,\&\,\neg\varphi_0(\bar{v}), \\ \ldots, & \\ q_n(\bar{v}) & \text{if } HW(\mathfrak{M}) \models \varphi_n(\bar{v})\,\&\,\neg\varphi_0(\bar{v})\,\&\cdots\&\,\neg\varphi_{n-1}(\bar{v}), \\ q_{n+1}(\bar{v}) & \text{if } HW(\mathfrak{M}) \models \neg\varphi_0(\bar{v})\,\&\,\neg\varphi_1(\bar{v})\,\&\cdots\&\,\neg\varphi_n(\bar{v}). \end{cases}$$

The formula $t = q$ is equivalent to

$$(t(\bar{v}) = q_0(\bar{v}) \,\&\, \varphi_0(\bar{v})) \vee (t(\bar{v}) = q_1(\bar{v}) \,\&\, \varphi_1(\bar{v}) \,\&\, \neg\varphi_0(\bar{v})) \vee \cdots \vee (t(\bar{v})$$
$$= q_i(\bar{v}) \,\&\, \varphi_i(\bar{v}) \,\&\, \neg\varphi_0(\bar{v}) \,\&\, \cdots \,\&\, \neg\varphi_{i-1}(\bar{v})) \vee \cdots \vee (t(\bar{v})$$
$$= q_n(\bar{v}) \,\&\, \varphi_n(\bar{v}) \,\&\, \neg\varphi_0(\bar{v}) \,\&\, \cdots \,\&\, \neg\varphi_{n-1}(\bar{v})) \vee (t(\bar{v})$$
$$= q_{n+1}(\bar{v}) \,\&\, \neg\varphi_0(\bar{v}) \,\&\, \neg\varphi_1(\bar{v}) \,\&\, \cdots \,\&\, \neg\varphi_n(\bar{v})).$$

Each conjunctive term in each disjunctive term has complexity less than the complexity of the formula $t = q$, and so, by the induction hypothesis, they and all their negations are equivalent to the standard Δ_0-formulas. Thus, all disjunctive terms are equivalent to the Σ-formulas ψ_0, $\psi_1, \ldots, \psi_n, \psi_{n+1}$; therefore, $t = q$ is equivalent to the standard Σ-formula $(\psi_0 \vee \psi_1 \vee \psi_2 \vee \cdots \vee \psi_{n+1})$.

Consider the case of the negation of our formula $\neg t = q$. By the definition of recursive scheme, this formula is equivalent to

$$(\neg t(\bar{v}) = q_0(\bar{v}) \,\&\, \varphi_0(\bar{v})) \vee (\neg t(\bar{v}) = q_1(\bar{v}) \,\&\, \varphi_1(\bar{v}) \,\&\, \neg\varphi_0(\bar{v})) \,\&\, \cdots \,\&\, (\neg t(\bar{v})$$
$$= q_i(\bar{v}) \,\&\, \varphi_i(\bar{v}) \,\&\, \neg\varphi_0(\bar{v}) \,\&\, \cdots \,\&\, \neg\varphi_{i-1}(\bar{v})) \,\&\, \cdots \,\&\, (\neg t(\bar{v})$$
$$= q_n(\bar{v}) \,\&\, \varphi_n(\bar{v}) \,\&\, \neg\varphi_0(\bar{v}) \,\&\, \cdots \,\&\, \neg\varphi_{n-1}(\bar{v})) \,\&\, (\neg t(\bar{v})$$
$$= q_{n+1}(\bar{v}) \,\&\, \neg\varphi_0(\bar{v}) \,\&\, \neg\varphi_1(\bar{v}) \,\&\, \cdots \,\&\, \neg\varphi_n(\bar{v})).$$

Each conjunctive term in each conjunctive term has complexity less than the complexity of $t = q$. By the induction hypothesis, they all and their negations are equivalent to standard Σ-formulas. Thus, all conjunctive terms are equivalent to Σ-formulas ψ_0, $\psi_1, \ldots, \psi_n, \psi_{n+1}$ respectively; therefore, $t = q$ is equivalent to the standard Σ-formula $(\psi_0 \& \psi_1 \& \psi_2 \& \cdots \& \psi_{n+1})$.

Subcase 1.4. Consider the possibility of equality of terms when $q(\bar{v})$ is obtained by the scheme of the list recursive definition, i.e., $g = \mathrm{Rec}(g_0, h)$, the semantics is defined as follows: $g(\bar{v}, \mathrm{nil}) = g_0(\bar{v})$ and $g(\bar{v}, \mathrm{cons}(y, z)) = h(\bar{v}, y, z, g(\bar{v}, y))$, and for the elements of the basic model, i.e., for the elements that are not lists, it is extended by the rule: if $\neg U(y)$ is fulfilled then $t(\bar{v}, y) = y$.

For a list y, $\neg U(y)$ is fulfilled in the model then, for y equal to nil, we take the one-element list α_y equal to $((\mathrm{nil}, g_0(\bar{v}))$, i.e., consisting of the pair of elements nil and $g_0(\bar{v})$. If, for a list $y = (x_1, \ldots, x_n)$, the list $\alpha_{(x_1,\ldots,x_n)}$ is already defined then, for the value z and the list $\mathrm{cons}((x_1, \ldots, x_n), z) = (x_1, \ldots, x_n, z)$ defining the list $(x_1, \ldots, x_n, z)$, define the list $\alpha_{\mathrm{cons}(y,z)}$ composed of pairs of elements $\mathrm{cons}(\alpha_{(x_1,\ldots,x_n)}), (\mathrm{cons}(y, z)$, $h(\bar{v}, y, z, g(\bar{v}, y)))$, i.e., the list α_w is defined from w by induction on the length of w. As a result of the definition, we conclude that the list $\alpha_{\mathrm{cons}(y,z)}$ for a list y and an element z is equal to

$$\langle\langle \mathrm{nil}, g(\bar{v}, \mathrm{nil})\rangle, \langle\langle x_1\rangle, g(\bar{v}, \langle x_1\rangle)\rangle, \ldots, \langle\langle x_1, \ldots, x_n\rangle,$$
$$g(\bar{v}, \langle x_1, \ldots, x_n, \rangle)\rangle, \langle\langle x_1, \ldots, x_n, z\rangle, h(\bar{v}, \langle x_1, \ldots, x_n\rangle, z, g(\bar{v}, \langle x_1, \ldots, x_n\rangle))\rangle\rangle,$$

where $y = (x_1, \ldots, x_n)$.

Note that, for each list y, given the fixed recursive terms g_0, h and the recursive term $g = \mathrm{Rec}(g_0, h)$, the list α_y is defined by

$$\psi^{g_0,h}(\bar{v}, \alpha, y) = \neg U(y) \,\&\, \neg U(\alpha) \,\&\, (\forall v \in \alpha)(\neg(v = \mathrm{nil})$$
$$\&\, \neg(\mathrm{tail}(v) = \mathrm{nil}) \,\&\, \mathrm{tail}(\mathrm{tail}(v)) = \mathrm{nil} \,\&\, (y = \mathrm{nil} \to (\mathrm{tail}(\alpha) = \mathrm{nil}$$
$$\&\, \mathrm{head}(\mathrm{head}(\alpha) = g_0(\bar{v}) \,\&\, \mathrm{head}(\mathrm{tail}(\mathrm{head}(\alpha))) = \mathrm{nil}))$$
$$\&\, ((\forall u \sqsubseteq \alpha)(((\neg u = \mathrm{nil} \,\&\, \mathrm{tail}(u) = \mathrm{nil})$$
$$\to u = \mathrm{cons}(\mathrm{nil}, \mathrm{cons}(\mathrm{cons}(\mathrm{nil}, \mathrm{nil}), g_0(\bar{v})))) \,\&\, ((\neg(u = \mathrm{nil} \,\&\, \neg(\mathrm{tail}(u) = \mathrm{nil}))$$
$$\to ((\mathrm{head}(\mathrm{head}(u)) = h(\bar{v}, \mathrm{head}(\mathrm{head}(\mathrm{tail}(u))),$$
$$\mathrm{head}(\mathrm{head}(\mathrm{head}(\mathrm{tail}(u)))), \mathrm{head}(\mathrm{head}(u))) \,\&\, \mathrm{head}(\mathrm{tail}(u)) \sqsubseteq y)$$
$$\&\, \mathrm{cons}(\mathrm{head}(\mathrm{tail}(\mathrm{head}(\mathrm{tail}(u)))), \mathrm{head}(\mathrm{tail}((\mathrm{head}(\mathrm{tail}(\mathrm{head}(u)))))))$$
$$= \mathrm{head}(\mathrm{tail}(\mathrm{head}(u)))) \,\&\, \mathrm{head}(\mathrm{tail}(\mathrm{head}(\alpha))) = y))).$$

Transform

$$(y = \text{nil} \rightarrow (\text{tail}(\alpha) = \text{nil} \;\&\; \text{head}(\text{head}(\alpha) = g_0(\bar{v}) \;\&\; \text{head}(\text{tail}(\alpha)) = \text{nil}))$$

to the equivalent Σ-formula ρ, equal to

$$\begin{gathered}(\exists w)(y = \text{nil} \rightarrow (\text{tail}(\alpha) = \text{nil} \;\&\; \text{head}(\text{head}(\alpha) = w \\ \&\; w = g_0(\bar{v}) \;\&\; \text{head}(\text{tail}(\text{head}(\alpha))) = \text{nil})),\end{gathered}$$

and for its negation, to the formula ρ^*, equal to

$$(\exists w)(y = \text{nil} \rightarrow (\text{tail}(\alpha) = \text{nil} \;\&\; \text{head}(\text{head}(\alpha) = w \& \neg w = g(\bar{v}) \;\&\; \text{head}(\text{tail}(\alpha)) = \text{nil})).$$

Also, transform

$$\begin{gathered}(\forall u \sqsubseteq \alpha)(((\neg u = \text{nil} \;\&\; \text{tail}(u) = \text{nil}) \rightarrow u = \text{cons}(\text{nil}, \text{cons}(\text{cons}(\text{nil}, \text{nil}), g_0(\bar{v}))) \\ \&\; (\neg(u = \text{nil} \;\&\; \neg(\text{tail}(u) = \text{nil})) \rightarrow ((\text{head}(\text{head}(u)) \\ = h(\bar{v}, \text{head}(\text{head}(\text{tail}(u))), \text{head}(\text{head}(\text{head}(\text{tail}(u)))), \text{head}(\text{head}(u))) \\ \&\; \text{head}(\text{tail}(u)) \sqsubseteq y) \;\&\; \text{cons}(\text{head}(\text{tail}(\text{head}(\text{tail}(u)))), \text{head}(\text{tail} \\ ((\text{head}(\text{tail}(\text{head}(u)))))))) = \text{head}(\text{tail}(\text{head}(u)))) \;\&\; \text{head}(\text{head}(\alpha)) = \text{cons}(y, z))))\end{gathered}$$

to the equivalent formula

$$\begin{gathered}\pi = (\forall u \sqsubseteq \alpha)(\exists w_1)(\exists w_2)(\exists w_3)(\exists w_4)(\exists w_2)(((\neg u = \text{nil} \;\&\; \text{tail}(u) = \text{nil}) \\ \rightarrow u = \text{cons}(\text{nil}, \text{cons}(\text{cons}(\text{nil}, \text{nil}), g_0(\bar{v}))) \;\&\; (\neg(u = \text{nil} \;\&\; \neg(\text{tail}(u) = \text{nil})) \\ \rightarrow ((\text{head}(\text{head}(u)) = w_1 \;\&\; w_1 = h(\bar{v}, w_3, w_4, w_5) \;\&\; w_3 = \text{head}(\text{head}(\text{tail}(u))) \\ \&\; w_4 = \text{head}(\text{head}(\text{head}(\text{tail}(u)))) \;\&\; w_5 = \text{head}(\text{head}(u))) \;\&\; \text{head}(\text{tail}(u)) \sqsubseteq y) \\ \&\; \text{cons}(\text{head}(\text{tail}(\text{head}(\text{tail}(u)))), \text{head}(\text{tail}((\text{head}(\text{tail}(\text{head}(u)))))))) \\ = \text{head}(\text{tail}(\text{head}(u)))) \;\&\; \text{head}(\text{tail}(\text{head}(\alpha))) = y).\end{gathered}$$

In this case, $t = g$ is equivalent to

$$(U(y) \rightarrow t = y) \;\&\; (\neg U(y) \rightarrow (\exists \alpha)(\psi^{g_0,h}(\bar{v}, \alpha, y) \;\&\; t = \text{head}(\text{head}(\alpha))),$$

the negation of the formula $\neg t = g$ is equivalent in our model of hereditary finite lists to

$$(U(y) \rightarrow \neg t = y) \;\&\; (\neg U(y) \rightarrow (\exists \alpha)(\psi^{g_0,h}(\bar{v}, \alpha, y) \;\&\; \neg t = \text{head}(\text{head}(\alpha))).$$

In this case, $t = g$ is equivalent to

$$(U(y) \rightarrow t = y) \;\&\; (\neg U(y) \rightarrow (\exists w)(\rho(\bar{v}, \alpha, y) \;\&\; \pi(\bar{v}, \alpha, y) \& \; \text{head}(\text{head}(\alpha)) = w \;\&\; t = w)),$$

and $\neg t = g$ is equivalent to

$$(\exists w)(\rho(\bar{v}, \alpha, y) \;\&\; \pi(\bar{v}, \alpha, y) \;\&\; \text{head}(\text{head}(\alpha)) = w \;\&\; t = w).$$

For the negation $\neg t = g$ of the formula, using the same considerations, define the equivalent formula

$$(U(y) \rightarrow \neg t = y) \& (\neg U(y) \rightarrow (\exists w)(\rho(\bar{v}, \alpha, y) \;\&\; \pi(\bar{v}, \alpha, y) \;\&\; \text{head}(\text{head}(\alpha)) = w \;\&\; \neg t = w).$$

SUBCASE 1.5. Consider the remaining case when ψ has the form $P(t_1(\bar{v}), \ldots, t_m(\bar{v}))$, including the formulas $t \in q$ where at least one term of $t_1(\bar{v}), \ldots, t_n(\bar{v})$ is a nonstandard recursive term. $P(t_1(\bar{v}), \ldots, t_m(\bar{v}))$ is equivalent to

$$(\exists z_1)(\exists z_2) \ldots (\exists z_m)\Big(\mathop{\&}_{1 \leq i \leq m} z_i = t_i(\bar{v}) \;\&\; P(z_1, \ldots, z_{i-1}, z_i, \ldots, z_m)\Big),$$

$\neg P(t_1(\bar v),\dots,t_m(\bar v))$ is equivalent to

$$(\exists z_1)(\exists z_2)\dots(\exists z_m)\Big(\mathop{\&}_{1\le i\le m} z_i = t_i(\bar v)\,\&\,\neg P(z_1,\dots,z_{i-1},z_i,z_{i+1},\dots z_m)\Big).$$

Here the conjunctive terms $z_i = q_i$ are equivalent, as was observed in the previous cases, to some standard Σ-formulas, and so both formulas are equivalent to standard Σ-formulas.

Consider all possibilities for defining φ.

Case 2. If the Δ_0(Rec)-formula φ looks as $(\varphi_1\,\&\,\varphi_2)$, $(\varphi_1\vee\varphi_2)$, $(\varphi_1\to\varphi_2)$, or $\neg\varphi_1$, then the complexity of the Δ_0(Rec)-formulas φ_1 and φ_2 is less than the complexity of φ and, by the induction hypothesis, for them there are equivalent Σ-formulas φ_1^* and φ_2^* respectively. The formulas $\neg\varphi_1$ and $\neg\varphi_2$ are equivalent to Σ-formulas $\varphi_{1,0}^*$ and $\varphi_{2,0}^*$ respectively, but then the conjunction $(\varphi_1\,\&\,\varphi_2)$ is equivalent to $(\varphi_1^*\,\&\,\varphi_2^*)$, and $\neg\varphi$ is equivalent to $(\varphi_{1,0}^*\vee\varphi_{2,0}^*)$, the disjunction $(\varphi_1\vee\varphi_2)$ is equivalent to $(\varphi_1^*\vee\varphi_2^*)$ and $\neg\varphi$ is equivalent to $(\varphi_{1,0}^*\,\&\,\varphi_{2,0}^*)$, the implication $(\varphi_1\to\varphi_2)$ is equivalent to $(\varphi_{1,0}^*\vee\varphi_2^*)$, and $\neg\varphi$ is equivalent to $(\varphi_1^*\,\&\,\varphi_{2,0}^*)$, and for the negation we have $\neg\varphi_1\equiv\varphi_{1,0}^*$ and $\neg\varphi\equiv\varphi_1^*$. In all cases, we obtain the desired Σ-formula equivalent to the formula and its negation.

Case 3. Consider all possibilities of the formulas with a restricted quantifier. Look at $(\exists v\in t)$. Let $\varphi=(\exists v\in t)\theta$. In this case, $r(\theta)<r(\varphi)$ and for θ there is an equivalent Σ-formula θ^* and for $\neg\theta$ there is an equivalent Σ-formula θ_0^*. The three subcases are possible:

Subcase 3.1: $(\exists v\in t)$. If t is a standard term then $\varphi=(\exists v\in t)\theta\equiv(\exists v\in t)\theta^*$, and we obtain a desired Σ-formula. Construct an equivalent formula for the negation of $(\exists v\in t)\theta$. In this case, $\neg(\exists v\in t)\theta$ is equivalent to the Σ-formula $(\forall v\in t)\neg\theta$ which is equivalent to the Σ-formula $(\forall v\in t)\theta_0^*$, and the condition is fulfilled.

Subcase 3.2: $(\exists v\in t)$. Let t be a recursive nonstandard term. For $\varphi=(\exists v\in t)\theta$, consider the equivalent formula $(\exists w)(w=t\,\&\,\theta)$. For the first conjunctive term $w=t$, by the above, there is an equivalent formula.

Note that the variable v does not occur freely in the definition of t and so v does not occur in the definitions of $t_1,\dots,t_n$ but $r(\theta)<r(\varphi)$. In this case, by the induction hypothesis, for θ there is an equivalent Σ-formula θ^* already without recursive standard terms, and for $\neg\theta$ there is an equivalent Σ-formula θ_0^* already without nonstandard terms.

In this case, $(\exists v\in F(t_1(\bar v),\dots,t_n(\bar v)))\theta$ is equivalent to the Δ_0(Rec)-formula $(\exists z_1=t_1)\dots(\exists z_n=t_n)$ $(\exists v\in F(z_1,\dots,z_n))\theta$, which is in turn equivalent to $(\exists z_1=t_1)\dots(\exists z_n=t_n)(\exists v\in F(z_1,\dots,z_n))\theta^*$, where θ^* is a Σ-formula.

Consider the formulas $z_i=t_i$ for $1\le i\le n$, which, obviously, have rank less than $r(\varphi)$. By the induction hypothesis, they are equivalent to suitable Σ-formulas ψ_i for $1\le i\le n$. Then

$$(\exists z_1=t_1)\dots(\exists z_n=t_n)(\exists v\in F(z_1,\dots,z_n))\theta^*$$

is equivalent to the Σ-formula

$$(\exists z_1)\dots(\exists z_n)\Big((\exists v\in F(z_1,\dots,z_n))\theta^*\,\&\,\Big(\mathop{\&}_{1\le i\le n}\psi_i\Big)\Big)$$

which is a desired formula for φ.

Consider the negation $\neg(\exists v\in F(t_1(\bar v),\dots,t_n(\bar v)))\theta$. In this case, $\neg(\exists v\in F(t_1(\bar v),\dots,t_n(\bar v)))\theta$ is equivalent to the Δ_0^{**}-formula

$$(\exists z_1=t_1)\dots(\exists z_n=t_n)(\exists v\in F(z_1,\dots,z_n))\neg\theta,$$

which is in turn equivalent to

$$(\exists z_1=t_1)\dots(\exists z_n=t_n)(\exists v\in F(z_1,\dots,z_n))\theta_0^*,$$

where θ_0^* is a Σ-formula.

Once again consider the formulas $z_i = t_i$ for $1 \le i \le n$, which, obviously, have rank less than the rank of $r(\varphi)$. Like in the first case, they are equivalent by the induction hypothesis to some suitable Σ-formulas ψ_i for $1 \le i \le n$. In this case, the formula

$$(\exists z_1 = t_1) \dots (\exists z_n = t_n)(\exists v \in F(z_1, \dots, z_n))\theta_0^*$$

is equivalent to the Σ-formula

$$(\exists z_1) \dots (\exists z_n)\Big((\exists v \in F(z_1, \dots, z_n))\theta_0^* \,\&\, \Big(\mathop{\&}_{1 \le i \le n} \psi_i\Big)\Big)$$

which is a desired formula for $\neg\varphi$.

Subcase 3.3: $(\exists v \in t)$. Consider the subcase that the term t in a formula φ of the form $(\exists v \in t)\theta$ is obtained by the scheme of a conditional term. Like in the previous case, the variable v does not occur freely in the definition of t and so v does not occur freely in the definitions of $t_1, \dots, t_n$ but $r(\theta) < r(\varphi)$. In this case, by the induction hypothesis, for θ and $\neg\theta$ there are equivalent Σ-formulas θ^* and θ_0^* already without recursive standard terms.

Thus, consider the case when $t(\bar{v})$ is obtained by the scheme of a conditional term, i.e.,

$$t(\bar{v}) = \begin{cases} t_0(\bar{v}) & \text{if } \varphi_0(\bar{v}), \\ t_1(\bar{v}) & \text{if } \varphi_1(\bar{v}) \,\&\, \neg\varphi_0(\bar{v}), \\ \dots, & \\ t_n(\bar{v}) & \text{if } \varphi_n(\bar{v}) \,\&\, \neg\varphi_0(\bar{v}) \,\&\, \dots \,\&\, \neg\varphi_{n-1}(\bar{v}), \\ t_{n+1}(\bar{v}) & \text{if } \neg\varphi_0(\bar{v}) \,\&\, \neg\varphi_1(\bar{v}) \,\&\, \dots \,\&\, \neg\varphi_n(\bar{v}). \end{cases}$$

Note that, in defining recursive terms, we have only used the Δ_0(Rec)-formulas $\varphi_0, \varphi_1, \dots, \varphi_n$ but the terms $t_0(\bar{v}), t_1(\bar{v}), \dots, t_{n+1}(\bar{v})$ are recursive of complexity less than the complexity of $t(\bar{v})$. Again by the induction hypothesis, for all formulas $\varphi_0, \varphi_i, \dots, \varphi_n$ of rank less than the entire formula, there are Σ-formulas $\varphi_0^*, \varphi_1^*, \dots, \varphi_n^*$ and $\varphi_{0,0}^*, \varphi_{i,0}^*, \dots, \varphi_{n,0}^*$ such that for any $0 \le i \le n$ the formula φ_1 is equivalent to the Σ-formula φ_i^* and $\neg\varphi_i$ is equivalent to the Σ-formula $\varphi_{i,0}^*$.

This implies that for all $0 \le i \le n$ the formulas of the form

$$(\exists v \in t_i(\bar{v}))(\theta \,\&\, \varphi_i \,\&\, \neg\varphi_0 \,\&\, \dots \,\&\, \neg\varphi_{i-1})$$

are equivalent to

$$(\exists v \in t_i(\bar{v}))(\theta^* \,\&\, \varphi_i^* \,\&\, \neg\varphi_{0,0}^* \,\&\, \dots \,\&\, \neg\varphi_{i-1,0}^*),$$

and so, by the induction hypothesis, are equivalent to the Σ-formula λ_i. Similarly, the Δ_0(Rec)-formula

$$(\exists v \in t_{n+1}(\bar{v}))(\theta \,\&\, \neg\varphi_0 \,\&\, \dots \,\&\, \neg\varphi_n)$$

is equivalent to

$$(\exists v \in t_{n+1}(\bar{v}))(\theta^* \,\&\, \varphi_{0,0}^* \,\&\, \dots \,\&\, \neg\varphi_{n,0}^*),$$

which already has complexity less than the complexity of $(\exists v \in t)\theta$ and so is equivalent to the Σ-formula λ_{n+1} by the induction hypothesis.

It is obvious from the definition of recursive term t that $(\exists v \in t)\theta$ is in this case equivalent to the disjunction of the two formulas

$$\bigvee_{0 \le i \le n} (\exists v \in t_i(\bar{v}))(\theta \,\&\, \neg\varphi_0 \,\&\, \dots \,\&\, \neg\varphi_{i-1} \;\varphi_i) \vee (\exists v \in t_{n+1}(\bar{v}))(\theta \,\&\, \neg\varphi_0 \,\&\, \dots \,\&\, \neg\varphi_n)$$

which is equivalent to the disjunction of Σ-formulas $\bigvee_{0 \le i \le n+1} \lambda_i$, which is a Σ-formula. The condition is proved for $(\exists v \in t)\theta$.

Prove for the negation $\neg(\exists v \in t)\theta$ of $(\exists v \in t)\theta$ that there exists an equivalent Σ-formula. Since $(\exists v \in t)\theta$ is equivalent to the disjunction

$$\bigvee_{0\le i\le n} (\exists v \in t_i(\bar v))(\theta \,\&\, \neg\varphi_0 \,\&\cdots\&\, \neg\varphi_{i-1}\ \varphi_i) \vee (\exists v \in t_{n+1}(\bar v))(\theta \,\&\, \neg\varphi_0 \,\&\cdots\&\, \neg\varphi_n),$$

$\neg(\exists v \in t)\theta$ is equivalent to

$$\&\!\!\!\bigvee_{0\le i\le n} \neg(\exists v \in t_i(\bar v))(\theta \,\&\, \neg\varphi_0 \,\&\cdots\&\, \neg\varphi_{i-1}\ \varphi_i)\&\, \neg(\exists v \in t_{n+1}(\bar v))(\theta \,\&\, \neg\varphi_0 \,\&\cdots\&\, \neg\varphi_n).$$

This formula is equivalent to

$$\&\!\!\!\bigvee_{0\le i\le n} (\forall v \in t_i(\bar v))(\neg\theta \vee \varphi_0 \vee\cdots\vee \varphi_{i-1} \vee \neg\varphi_i)\,\&\,(\forall v \in t_{n+1}(\bar v))(\neg\theta \vee \varphi_0 \vee\cdots\vee \varphi_n).$$

Again use the fact that all conjunctive terms in the last formula are Δ_0(Rec)-formulas of rank less than the main formula and find for them the equivalent Σ-formulas λ_i for $0 \le i \le n+1$. Then $\neg(\exists v \in t)\theta$ is equivalent to the Σ-formula $\&_{0\le i\le n+1}\lambda_i$.

Subcase 3.4: $(\exists v \in t)$. Consider the subcase that t in a formula φ like $(\exists v \in t)\theta$ is obtained by the scheme of list recursion from recursive terms g and h. In the previous case, the variable v does not occur freely in the definition of t and so v does not occur freely in the definitions of g and h and $r(\theta) < r(\varphi)$. In this case, by the induction hypothesis, for θ and $\neg\theta$ there are equivalent Σ-formulas θ^* and θ_0^* already without recursive nonstandard terms.

The cases of the formulas beginning with restricted quantifiers $(\forall v \in t_i(\bar v))$, $(\exists v \sqsubseteq t_i(\bar v))$, $(\forall v \sqsubseteq t_i(\bar v))$, $(\exists v = t_i(\bar v))$, and $(\forall v = t_i(\bar v))$ are examined by complete analogy to the case of a restricted quantifier $(\exists \in t_i(\bar v))$.

The theorem is proved.

It is not hard to observe that, in the recursive definition of term, in defining the conditions, we can use Δ_0(Rec)-relations, and the corollary remains valid.

Let $\mathfrak{M}$ be a model with binary relations $\le_1, \dots, \le_n$.

A model $\mathfrak{M}$ is Σ-restricted with respect to $\le_1, \dots, \le_n$ if $\mathfrak{M}$ satisfies the conditions:

1. Each relation $\le_i$ is transitive and directed with respect to the relation $\le_1$, i.e., for all elements a, b, and c of $\mathfrak{M}$, the conditions $a \le_i b$ and $b \le_i c$ imply that $a \le_i c$, and for all elements a and b there exists an element e such that for every element k if $k \le_i a$ and $k \le_i b$ then $k \le_1 e$.

2. Each Σ-formula $\psi(x_1, \dots, x_n, y, z)$ satisfies the condition of restrictedness for a given spectrum of binary relations

$$(\forall y \le_i w)(\exists z)\psi(x_1, \dots, x_n, y, z) \to (\exists u)(\forall y \le_i w)(\exists z \le_1 u)\psi(x_1, \dots, x_n, y, z).$$

We obtain a generalization of the Reflection Theorem of [10] applicable in the case of a list superstructure with several restrictions on the quantifiers.

Reflection Theorem. *If $\mathfrak{M}$ is Σ-restricted with respect to $\le_1, \dots, \le_n$ and $\varphi(x_1, \dots, x_n)$ of a Σ-formula then there exists a Δ_0-formula $\psi(x_1, \dots, x_n, y)$ such that $\varphi(x_1, \dots, x_n) \equiv (\exists y)\psi(x_1, \dots, x_n, y)$ in every model $\mathfrak{M}$ Σ-restricted with respect to $\le_1, \dots, \le_n$. The formula $\psi(x_1, \dots, x_n, y)$ is found efficiently from $\varphi(x_1, \dots, x_n)$.*

Corollary. *Each Σ(Rec)-relation on $HW(\mathfrak{M})$ is a Σ-relation on $HW(\mathfrak{M})$.*

For constructing a good computability theory, we need a generalization of the Reflection Theorem to all binary relations: $\in$, $\sqsubseteq$, and $=$.

If the basic operations and relations of our model are computable then all recursive terms will be computable. Moreover, if the basic terms and relations of the superstructure are of polynomially computable complexity then all conditional terms will be of polynomial complexity that can be efficiently

estimated starting from the definition of the conditional term and the formulas occurring in its definition whereas for the terms with recursion this cannot be proved. But if we choose from these terms those that are computable in polynomial time and use only such terms in defining Δ_0-formulas then for each constructed $\Delta_0(\mathrm{Rec})$-formula, the check of its truth on the elements of the model elements will have polynomial complexity. Owing to this fact, we can add functions defined by recursive terms to the basic operations of the model, and this will not change the expressiveness of our language while preserving all definable properties in our structure.

For all new recursive terms, the theorem on the polynomiality of the algorithm for checking the truth of $\Delta_0(\mathrm{Rec})$-formulas fails but, following the proof of the theorem of polynomiality in [16], we can however obtain a theorem on the complexity of checking truth and calculating terms in this case, which is uniform for all recursive terms and $\Delta_0(\mathrm{Rec})$-formulas, by replacing polynomiality with primitive recursivity.

Theorem. (1) *If a basic model $\mathfrak{M}$ admits a primitively recursive representation then the model $HW(\mathfrak{M})$ also admits a primitively recursive representation.*

(2) *If $HW(\mathfrak{M})$ admits a primitively recursive representation then there exists a primitive recursive algorithm computing from any recursive term $t(v_1, \ldots, v_k)$ and the indices $n_1, \ldots, n_k$ of elements $a_1, \ldots, a_k$ of $HW(\mathfrak{M})$ the index of the element equal to the value of the term $t(a_1, \ldots, a_k)$ and the truth check of each $\Delta_0(\mathrm{Rec})$-formula $\varphi(v_1, \ldots, v_k)$ at the elements $a_1, \ldots, a_k$ with indices $n_1, \ldots, n_k$ in $HW(\mathfrak{M})$.*

References

1. Goncharov S. S., "Conditional terms in semantic programming," Sib. Math. J., vol. 58, no. 5, 794–800 (2017).
2. Goncharov S. S. and Sviridenko D. I., "Σ-Programming," Trans. Amer. Math. Soc., no. 142, 101–121 (1989).
3. Goncharov S. S. and Sviridenko D. I., "Σ-Programming and their semantics," Vychisl. Systemy, no. 120, 24–51 (1987).
4. Goncharov S. S. and Sviridenko D. I., "Theoretical aspects of Σ-programming," Lect. Notes Comput. Sci., vol. 215, 169–179 (1986).
5. Ershov Yu. L., Goncharov S. S., and Sviridenko D. I., "Semantic programming," in: *Information Processing 86. Proc. IFIP 10th World Comput. Congress*, Dublin, 1986, vol. 31, 1113–1120.
6. Goncharov S. S. and Sviridenko D. I., "Mathematical bases of semantic programming," Soviet Math. Dokl., vol. 31, no. 6, 608–610 (1986).
7. Ershov Yu. L., Goncharov S. S., and Sviridenko D. I., "Semantic foundations of programming," Lect. Notes Comput. Sci., vol. 278, 116–122 (1987).
8. Ershov Yu. L., "The principle of Σ-enumeration," Soviet Math. Dokl., vol. 27, no. 4, 670–672 (1983).
9. Ershov Yu. L., "Dynamic logic over admissible sets," Soviet Math. Dokl., vol. 28, no. 5, 739–742 (1983).
10. Ershov Yu. L., *Definability and Computability*, Consultants Bureau, New York and London (1996).
11. Goncharov S. S., "The theory of lists, and its models," Vychisl. Systemy, no. 114, 84–95 (1986).
12. Goncharov S. S., "Data models and languages for their description," Trans. Amer. Math. Soc., no. 143, 139–152 (1989).
13. Ershov Yu. L., Goncharov S. S., Nerode A., and Remmel J., *Introduction to the Handbook of Recursive Mathematics.* Vol. 138, Elsevier, Amsterdam (1998).
14. Ershov Yu. L., Puzarenko V. G., and Stukachev A. I., "HF-computability," in: *Computability in Context*, Imperial College Press, London, 2011, 169–242.
15. Morozov A. S. and Puzarenko V. G., "Σ-subsets of natural numbers," Algebra and Logic, vol. 43, no. 3, 162–178 (2004).
16. Ospichev S. and Ponomarev D., "A note on the complexity of formulas in semantic programming," Sib. Èlektron. Mat. Izv., 2018, vol. 15, 987–995.

S. S. Goncharov; D. I. Sviridenko
Sobolev Institute of Mathematics, Novosibirsk, Russia
E-mail address: s.s.goncharov@math.nsc.ru; dsviridenko47@gmail.com