S©MR ISSN 1813-3304

СИБИРСКИЕ ЭЛЕКТРОННЫЕ
МАТЕМАТИЧЕСКИЕ ИЗВЕСТИЯ

Siberian Electronic Mathematical Reports
http://semr.math.nsc.ru

Том 15, стр. 987–995 (2018) УДК 510.5
DOI 10.17377/semi.2018.15.083 MSC 03D15,68Q15

ON THE COMPLEXITY OF FORMULAS IN SEMANTIC PROGRAMMING

S. OSPICHEV, D. PONOMAREV

Abstract. We consider the complexity of Δ_0 formulas augmented with conditional terms. We show that for formulas having n bounded quantifiers, for a fixed n, deciding the truth in a list superstructure with polynomial computable basic operations is of polynomial complexity. When the quantifier prefix has n alternations of quantifiers, the truth problem is complete for the n-th level of the polynomial-time hierarchy. Under no restrictions on the quantifier prefix the truth problem is PSPACE-complete. Thus, the complexity results indicate the analogy between the truth problem for Δ_0 formulas with conditional terms and the truth problem for quantified boolean formulas.

Keywords: semantic programming, list structures, polynomial time/space complexity, Δ_0-formulas.

1. Introduction

In this paper, we study the algorithmic complexity of hereditarily finite list extensions of structures. The generalized computability theory based on Σ-definability, which has been developed by Yuri Ershov [1] and Jon Barwise [2], considers hereditarily finite extensions consisting of hereditarily finite sets. In the papers by Yuri Ershov, Sergei Goncharov, and Dmitry Sviridenko [3, 4, 5, 6, 7] a theory of hereditarily finite extensions has been developed, which rests on the concept of Semantic Programming. In the paradigm of Semantic Programming, a program is specified by a Σ-formula in a suitable superstructure of finite lists. Two different types of

Ospichev, S., Ponomarev, D., On the complexity of formulas in semantic programming.

The authors were supported by the Russian Science Foundation (Grant No. 17-11-01176).
Received June, 28, 2018 , published September, 10, 2018.

implementation of logic programs on the basis of Σ-definability have been considered [8]. The first one is based on deciding the truth of Σ-formulas corresponding to the program in the constructed model. The second approach is based on the axiomatic definition of the theory of the list superstructure. Both of these approaches raise the natural question of how fast one can compute a program represented by Σ-formulas. In the recent paper [8] Sergey Goncharov has put a hypothesis that in case the base model $\mathcal{M}$ is polynomially computable then deciding the truth of a given Δ_0-formula in a hereditarily finite list extension of $\mathcal{M}$ has polynomial complexity. In this paper, we confirm this hypothesis and consider the complexity of this problem for a number of natural restrictions on Δ_0-formulas.

2. Preliminaries

The reader is referred to [3] for basic concepts and notations on list structures and to [9, 10] for the fundamentals of the complexity theory.

2.1. **Complexity Classes.** For a finite alphabet Σ, we denote by Σ^* the set of all words over Σ. For $A \subseteq \Sigma^*$, a function $f : A \to \Sigma^*$ is said to be $\mathtt{P}$-computable/$\mathtt{NP}$-computable if there is a deterministic/nondeterministic Turing machine T, respectively, and a polynomial p such that for any $x \in A$ the value of $f(x)$ can be computed by T in no more than $p(|x|)$ steps, where $|x|$ is the length of the word x.

A function $f : A \to \Sigma^*$ is $\mathtt{PSPACE}$-computable if there is a Turing machine T and a polynomial p, such that for any $x \in A$ the value of $f(x)$ can be computed by T using no more than $p(|x|)$ cells of the tape of T.

A subset $A \subseteq \Sigma^*$ is said to be $\mathtt{P}$-/$\mathtt{NP}$-/$\mathtt{PSPACE}$-computable, respectively, if so is the characteristic function $\chi_A : \Sigma^* \to \{0, 1\}$.

We say that a structure $\mathcal{M}$ is $\mathtt{P}$-computable if so are the functions, predicates, and the domain of $\mathcal{M}$.

A set A is $\mathtt{P}$-reducible to a set B if A is m-reducible to B by some $\mathtt{P}$-computable function. The notion of a complete set in some class (with respect to $\mathtt{P}$-reducibility) is defined in a standard way.

The polynomial-time hierarchy is an analogue of the arithmetic hierarchy, in which $\mathtt{P}$-computable sets play the role of computable ones and $\mathtt{NP}$-computable sets play the role of computably enumerable sets. The classes of the polynomial-time hierarchy are defined as follows:

$$\begin{aligned}&\Delta_0^p = \Sigma_0^p = \Pi_0^p = \mathtt{P}\\&\Delta_{i+1}^p = \mathtt{P}^{\Sigma_i^p}\\&\Sigma_{i+1}^p = \mathtt{NP}^{\Sigma_i^p}\\&\Pi_{i+1}^p = \mathtt{coNP}^{\Sigma_i^p}\end{aligned}$$

where $\mathtt{P}$ is the class of $\mathtt{P}$-computable sets and $\mathtt{P}^A$ is the class of sets, whose characteristic functions are computable in a polynomial number of steps by a deterministic Turing machine with an oracle, a $\mathtt{P}$-complete set from the class A. The classes $\mathtt{NP}^A$ and $\mathtt{coNP}^A$ are defined similarly. Then the notion of a Σ_k^p- or Π_k^p-computable function (or a subset), for $k \geqslant 0$, is defined in a straightforward way.

For any $i \geqslant 0$, it holds that $\Sigma_i^p \cup \Pi_i^p \subseteq \Delta_{i+1}^p \subseteq \Sigma_{i+1}^p \cap \Pi_{i+1}^p$.

2.2. **List Structures and Δ_0^*-formulas.** Here we follow [3] and introduce a framework for working with lists.

Let $\mathcal{M}$ be a model of signature σ. A superstructure of finite lists $HW(\mathcal{M})$ for $\mathcal{M}$ is defined by extending σ with the following LISP-like functions and predicates over lists:

(1) *nil* – the constant which represents the empty list;
(2) *head* – the last element of a non-empty list and *nil*, otherwise;
(3) *tail* – the list without the last element, for a non-empty list, and *nil*, otherwise;
(4) *cons* – the list obtained from adding a new last element to a list;
(5) *conc* – concatenation of two lists;
(6) $\in$ – the predicate "to be an element of a list";
(7) $\sqsubseteq$ – the predicate"to be an initial segment of a list".

Δ_0-*formulas* are first-order formulas, in which quantification is of the following two types:

- a restriction onto the list elements $\forall x \in t$ and $\exists x \in t$;
- a restriction onto the initial segments of lists $\forall x \sqsubseteq t$ and $\exists x \sqsubseteq t$.

Note that for any list terms s, t, $s \sqsubseteq t$ is equivalent to $s \in t'$, where $t' = \langle t, tail(t), tail(tail(t)), \ldots \rangle$, and this transformation can be done in polynomial time in the size of t. Therefore, in this paper we consider bounded quantifiers only of the form $\exists x \in t$ and $\forall x \in t$. The equality of terms $s = t$ can be represented as $s \sqsubseteq t \wedge t \sqsubseteq s$ and hence, is expressible via the $\in$-predicate with no more than polynomial increase of the size of the expression.

In [8], the language of Δ_0-formulas has been extended with *conditional terms* thus giving so called Δ_0^*-*formulas*. Both concepts are defined inductively as follows:

(1) each standard term is a conditional term of rank 0;
(2) a Δ_0^*-formula is a Δ_0-formula, in which conditional terms can occur at the places of standard terms, the rank of a Δ_0^*-formula is the maximum of the ranks of the terms occurring in it;
(3) if $t_0, \ldots, t_{n+1}$ are conditional terms and $\theta_0, \ldots, \theta_n$ are Δ_0^*-formulas, where $n \geqslant 0$, then the term $t(\overline{v})$ of the form $Cond(t_0, \ldots, t_{n+1}, \theta_0, \ldots, \theta_n)$ is a conditional term with the following interpretation:

$$t(\overline{v}) = \begin{cases} t_0(\overline{v}) & \text{if } \theta_0(\overline{v}) \\ t_1(\overline{v}) & \text{if } \neg\theta_0(\overline{v}) \wedge \theta_1(\overline{v}) \\ \ldots & \\ t_i(\overline{v}) & \text{if } \theta_i(\overline{v}) \wedge \neg\theta_0(\overline{v}) \wedge \neg\theta_1(\overline{v}) \wedge \ldots \wedge \neg\theta_{i-1}(\overline{v}) \\ \ldots & \\ t_n(\overline{v}) & \text{if } \theta_n(\overline{v}) \wedge \neg\theta_0(\overline{v}) \wedge \neg\theta_1(\overline{v}) \wedge \ldots \wedge \neg\theta_{n-1}(\overline{v}) \\ t_{n+1}(\overline{v}) & \text{if } \neg\theta_0(\overline{v}) \wedge \neg\theta_1(\overline{v}) \wedge \ldots \wedge \neg\theta_n(\overline{v}) \end{cases}$$

The rank of $t(\overline{v})$ is the maximum rank of the terms occurring in $t_0, \ldots, t_{n+1}$ and $\theta_0, \ldots, \theta_n$ incremented by 1.

The formulas mentioned on the right-hand side of the definition of $t(\overline{v})$ are called *conditions* and the terms $t_0, \ldots, t_{n+1}$ are called (possible) *instances* of t. Note that a condition of rank 0 is a Δ_0-formula.

If t is a standard term and $\varphi(t^c)$ is a Δ_0^*-formula, where t^c is a conditional term occurring in φ, then $\varphi(t^c/t)$ denotes the formula obtained by substituting t^c with t. Let $\mathcal{M}$ be a structure, $\mathcal{C} \in \{\mathtt{P}, \mathtt{PSPACE}\}$, where $k \geqslant 0$, a complexity class. Let $\mathcal{S}$ be the maximal $\mathcal{C}$-computable set[1] of Δ_0-formulas true on $HW(\mathcal{M})$. A Δ_0^*-formula φ is called $\mathcal{C}$-*conditional* if for any condition $\psi(t_1^c, \dots, t_n^c)$ occurring in φ (where $n \geqslant 0$ and $t_1^c, \dots, t_n^c$ are the conditional terms in ψ) and for any standard terms $t_1, \dots, t_n$ of size bounded by the size of φ, it holds that $\psi(t_1^c/t_1, \dots, t_n^c/t_n) \in \mathcal{S}$. Note that by this definition any Δ_0-formula occurring as a condition in φ is contained in $\mathcal{S}$.

It follows from the results in [8] that for any Δ_0^*-formula φ one can compute an equivalent Δ_0-formula ψ, whose size is exponential in the size of φ. When an underlying structure $\mathcal{M}$ is fixed, testing the truth of a Δ_0^*-formula can be reduced to that for a Δ_0-formula constructed for φ with no exponential overhead, as the next lemma states.

Lemma 1. *Let $\mathcal{M}$ be a structure, $\mathcal{C} \in \{\mathtt{P}, \mathtt{PSPACE}\}$ a complexity class, and $\varphi(t_1^c, \dots, t_n^c)$ a $\mathcal{C}$-conditional Δ_0^*-formula, where $t_1^c, \dots, t_n^c$ are the conditional terms in φ, $n \geqslant 0$. There is a $\mathcal{C}$-computable function, which gives a Δ_0-formula $\psi = \varphi(t_1^c/t_1, \dots, t_n^c/t_n)$, where $t_1, \dots, t_n$ are standard terms of size bounded by the size of φ, such that $HW(\mathcal{M}) \models \varphi$ iff $HW(\mathcal{M}) \models \psi$.*

Proof. We use induction on the rank k of φ. The induction base $k = 0$ is trivial, since in this case φ is a Δ_0-formula. Let φ be of rank $k+1$, where $k \geqslant 0$, and let t^c be a conditional term of rank $k+1$ occurring in φ, which has the form $Cond(t'_0, \dots, t'_{m+1}, \theta_0, \dots, \theta_m)$, $m \geqslant 0$. Since each condition θ_i, for $0 \leqslant i \leqslant m$, is of rank $\leqslant k$, by the induction assumption it holds $HW(\mathcal{M}) \models \theta_i$ iff $HW(\mathcal{M}) \models \theta_i(t_1^c/t_1, \dots, t_n^c/t_n))$, where $t_1^c, \dots, t_n^c$ are the conditional terms in θ_i and $t_1, \dots, t_n$ are some standard terms of size bounded by the size of φ. As φ is $\mathcal{C}$-conditional, there is a $\mathcal{C}$-computable function, which gives a (unique) instance t'_i of the term t^c, for which θ_i is true.

Let ψ be the formula obtained from φ by replacing every conditional term t of rank $k+1$ with the corresponding instance. The total number of the conditions of the terms occurring in φ is bounded by the size of φ, therefore this transformation can be done by a $\mathcal{C}$-computable function giving a formula $\psi(t_1^c, \dots, t_m^c)$, where $t_1^c, \dots, t_m^c$ are the conditional terms in ψ (each of rank $\leqslant k$) and $m \leqslant n$. It holds $HW(\mathcal{M}) \models \varphi$ iff $HW(\mathcal{M}) \models \psi$ and the formula ψ is of rank k. By the induction assumption, there is a $\mathcal{C}$-computable function, which gives a Δ_0-formula $\psi' = \psi(t_1^c/t_1, \dots, t_n^c/t_n)$, where $t_1, \dots, t_n$ are standard terms of size bounded by the size of φ, such that $HW(\mathcal{M}) \models \psi$ iff $HW(\mathcal{M}) \models \psi'$, thus, ψ' is the required formula for φ and the lemma is proved. □

It is therefore important to describe the complexity of Δ_0^*-formulas in terms of Δ_0-formulas and identify P-computable classes. We address this problem in Section 4 and begin with a crucial observation on computability of list structures.

3. P-computable List Structures

Theorem 1. *For any* P*-computable structure $\mathcal{M}$ there exists a* P*-computable representation of its superstructure of finite lists $HW(\mathcal{M})$.*

[1]It will be clear from the results in Section 4 that this set is non-empty.

Proof. Let $\mathcal{M} = \langle \Gamma^*, M, \sigma \rangle$ be a $\mathtt{P}$-computable structure of words over an alphabet Γ. For convenience, we denote the elements $m \in M$ as numerals $\underline{m}$. We show that the domain of $HW(\mathcal{M})$ and all of its functions and predicates are $\mathtt{P}$-computable.

Consider the alphabet $\Sigma = \{\langle, \rangle, ,, nil, \#\} \bigcup \Gamma$. For simplicity, we use the shortcut $\langle_i$ for a word of the form $\langle\langle\langle \ldots \langle$, where $\langle$ occurs i times. Similarly, we use the shortcuts $\rangle_i$ and $,_i$. For a word w in the language Σ^*, let $depth(w)$ denote the length of the maximal initial subword consisting only of "$\langle$". If there is no such word, we let $depth(w) = 0$.

Let us define a representation of $HW(\mathcal{M})$ as a structure with the domain $A \subseteq \Sigma^*$ consisting of the words defined as follows:

(1) A contains $\#nil\#$ and every word $\#\underline{m}\#$, for $m \in M$.

(2) If $\gamma_1, \gamma_2, \ldots, \gamma_n$ are some words from A, then A contains a word of the form $\gamma = \#\langle_i \gamma_1 ,_i \gamma_2, \ldots, \gamma_n \rangle_i \#$, $i = j + 1$, where j is the maximal depth of the words γ_k, for $k = 1, \ldots, n$.

It is easy to see that deciding whether an element is contained in A reduces to bracket parsing (brackets with a greater index must not occur between brackets having a smaller index) and testing the containment of numerals (by the condition, the characteristic function of M is $\mathtt{P}$-computable).

We now show that all list functions are $\mathtt{P}$-computable. Let γ be a word from A and $i = depth(\gamma)$.

(1) $head(\gamma)$: Find the subword between"$,_i$" (or "$\langle_i$" if $i = 1$) and "$\rangle_i$" and output it as the result. If $i = 0$ then output nil.

(2) $tail(\gamma)$: delete $head(\gamma)$ from γ. In new γ search all subwords "$\langle_j$", where $j < i$ and define new $depth(\gamma)$ as maximum of $j + 1$. Again, if $i = 0$ then the result is nil.

(3) $cons(\gamma_1, \gamma_2)$: Add γ_2 as the new last element to γ_1; if $depth(\gamma_1) \leq depth(\gamma_2)$ then set $depth(\gamma)$ to $depth(\gamma_2) + 1$.

(4) $conc(\gamma_1, \gamma_2)$: merge lists and define the new depth as $max(depth(\gamma_1), depth(\gamma_2))$.

(5) $\gamma_1 \in \gamma_2$: if $depth(\gamma_2) = j$ then γ_1 is a subword of γ_2 between "$\langle_j$" and "$\rangle_j$" (if there are no "$,_j$") or between $\langle_j$ and $,_j$ or between "$,_j$" and "$,_j$", or between "$,_j$" and "$\rangle_j$".

(6) $\gamma_1 \sqsubseteq \gamma_2$: if $j = depth(\gamma_1)$ and $k = depth(\gamma_2)$ then replace all the subwords"$\langle_j$" and "$,_j$" in γ_1 with "$\langle_k$" and"$,_k$", respectively, erase the subword "$\rangle_j$"; the predicate is true if the word obtained from γ_1 is the initial subword in γ_2.

The theorem is proved. □

An immediate corollary is the following lemma:

Lemma 2. *There is* $\mathtt{P}$*-computable function* f *such that for any (standard) list term* t *and* $\gamma_1, \ldots \gamma_n \in A$ *it holds* $f(t, \overline{\gamma}) = t(\overline{\gamma})$.

Proof. The computation of a list term t can be represented as a tree, where:

(1) the computation is made level-wise, from leaf nodes to the root;

(2) the leaf nodes are nil, constants from $\mathcal{M}$, or γ_i, for $i \leq n$;

(3) every node has at most two child nodes (since all the list functions have at most two arguments);

(4) by Theorem 1, any node can be computed in polynomial time based on the computation results for the child nodes. The length of the result at a node is at most $2a+c$, where a is the maximum length of the results for the child nodes and c is a constant;

The root node can be computed by a Turing machine in at most $p(2^k * (|\gamma|))$ steps, where p is some polynomial (for convenience, p can be defined as the sum of all the polynomials required to compute the list functions, plus some constant), $|\gamma| = max|\gamma_i|$, for $i \leq n$, and k is the height of the tree.

Since $2^k \leq |t|$, the root node can be computed in at most $p(|t| * (|\gamma|))$ steps. The number of nodes is bounded by $|t|$, thus, the value of the term t is computed in $|t| * p(|t|, |\gamma|)$ steps. □

4. Deciding the Truth of Δ_0^*-formulas.

By using the reduction from Lemma 1 it suffices to prove the results in this section only for Δ_0-formulas. Some of these results are formulated using restrictions on the quantifier prefix of Δ_0^*-formulas. Note that by Lemma 1, the prefix is preserved under the reduction of Δ_0^*- to Δ_0-formulas.

Theorem 2. *For a given $n \geqslant 0$, the set of* P*-conditional Δ_0^*-formulas with at most n bounded quantifiers, which are true in a* P*-computable structure $HW(\mathcal{M})$, is* P*-computable.*

Proof. We assume that the formulas are given in the prenex normal form.

Consider a quantifier-free formula $\varphi(\overline{\gamma})$, where $\overline{\gamma} = \gamma_1, \ldots, \gamma_m$ – is a m-tuple of lists. Then deciding the truth of this formula can be reduced to at most $|\varphi|$ computations of formulas of the form $t(\overline{\gamma}) \in q(\overline{\gamma})$, where t, q are some list terms. By Theorem 1 and Lemma 2, this can be verified in polynomial time.

From now on we consider formulas without free variables. For a given quantifier-free Δ_0-formula φ, consider the formulas $\psi_1, \ldots, \psi_{n+1}$ defined as follows:

(1) $\psi_1 = Q_1(x_1, t_1)Q_2(x_2, t_2) \ldots Q_n(x_n, t_n)\ \varphi(x_1, x_2, \ldots, x_n)$,
(2) $\psi_2 = Q_2(x_2, t_2) \ldots Q_n(x_n, t_n)\ \varphi(\gamma_1, x_2, \ldots, x_n)$,
(3) $\psi_i = Q_i(x_i, t_i) \ldots Q_n(x_n, t_n)\ \varphi(\gamma_1, \gamma_2, \ldots, \gamma_{i-1}, x_i, x_{i+1}, \ldots, x_n)$,
(4) $\psi_{n+1} = \varphi(\gamma_1, \gamma_2, \ldots, \gamma_n)$,

where $Q_i(x_i, t_i)$ is $\exists x_i \in t_i$ or $\forall x_i \in t_i$.

The formula ψ_1 is equivalent to

$$\psi_2(head(t_1)) \wedge \psi_2(head(tail(t_1))) \wedge \ldots \wedge \psi_2(head(tail(\ldots(tail(t_1))\ldots)))$$

if $Q_i(x_i, t_i)$ is $\forall x_i \in t_i$ and as

$$\psi_2(head(t_1)) \vee \psi_2(head(tail(t_1))) \vee \ldots \vee \psi_2(head(tail(\ldots(tail(t_1))\ldots)))$$

if $Q_i(x_i, t_i)$ is $\exists x_i \in t_i$.

Therefore, the truth of ψ_1 can be decided by at most $|t_1|$ truth tests for ψ_2. Similarly, the truth of ψ_2 can be tested with at most $|t_1| * |t_2|$ computations of ψ_3 (we have the multiplier $|t_1|$, since ψ_2 depends on $head(tail(\ldots(tail(t_1))\ldots)))$. Finally, the truth of ψ_n can be verified with at most $|t_1| * |t_2| * \ldots * |t_n|$ computations of ψ_{n+1}, which is a quantifier-free formula.

Let t be the maximum of $|t_i|$. Then the truth of ψ_1 can be decided with at most t^{n^2} computations of the quantifier-free formula $\psi_{n+1} = \varphi(\gamma_1, \gamma_2, \ldots, \gamma_n)$. Since n is fixed, this is a polytime procedure. □

Theorem 3. *The set of Δ_0^*-formulas, which are true in a* P*-computable structure $HW(\mathcal{M})$, is* PSPACE*-complete.*

Proof. For the lower complexity bound, we show that the set of true quantified boolean formulas P-reduces to the set of true Δ_0-formulas.

For consider $\varphi = Q_1X_1...Q_kX_k\ \psi$, where ψ is a boolean formula over variables $X_1, \ldots, X_n$ and $Q_i \in \{\exists, \forall\}$. Let us define Δ_0-formula $\varphi' = Q_1{}'x_1Q_2{}'x_2..Q_n{}'x_n\psi'$, where $Q_i{}'x_i = Q_ix_i \in \langle nil, \langle nil\rangle\rangle$, and ψ' is obtained from ψ by replacing every positive literal X_i with $x_i = \langle nil\rangle$ and each negative literal $\neg X_i$ with $x_i = nil$, respectively. Then one can readily verify that φ is true iff φ' is true.

Let us now demonstrate that the set of true Δ_0-formulas is PSPACE-computable. We use induction on the structural complexity of φ. By the condition that $HW(\mathcal{M})$ is P-computable and by Lemma 2, the set of true atomic formulas is P-computable (and thus, it is PSPACE-computable), then so are their boolean combinations (we note that PSPACE is closed under complementation). If φ has the form $\exists x \in t\ \psi(x, \overline{y})$ where t is a finite list (e.g., obtained from a list term by Lemma 2) then we compute the formula $\psi(a, \overline{y})$ for every element $a \in t$ by reusing space. This gives a PSPACE procedure to compute φ: the formula is true iff there is $a \in t$ such that $\psi(a, \overline{y})$ is true and by the induction assumption, the set of all such formulas is PSPACE-computable. □

Theorem 4. *The set of* P*-conditional Δ_0^*-formulas with k alternations of quantifiers $\exists x \in t$ and $\forall x \in t$, which are true in a* P*-computable structure $HW(\mathcal{M})$, is complete for the k-th level of the polynomial-time hierarchy.*

Proof. The lower bound is proved identically to that in Theorem 3 for the case of k alternating quantifiers and follows from the fact that the set of true quantified boolean formulas with k quantifier alternations is complete for the k-th level of the polynomial-time hierarchy. For the upper complexity bound we use the following criterion [11].

For any $k \geq 1$ and any set A, it holds $A \in \Sigma_k^p$ iff there is polynomial p and a P-computable set A' such that $x \in A$ iff $\exists y_1 \forall y_2 \ldots Q_k y_k[\langle x, y_1, y_2, \ldots, y_k\rangle \in A']$.

Similarly, $A \in \Pi_k^p$ is equivalent to the condition that $x \in A$ iff $\forall y_1 \exists y_2..Q'_k y_k[\langle x, y_1, y_2, .., y_k\rangle \in A']$.

In the formulas above, the ranges of $y_1, y_2, \ldots, y_k$ are bounded by a polynomial $p(x)$ and the quantifiers alternate, i.e., Q_k is $\exists$, if k is odd and Q_k is $\forall$, if k is even. Similarly, $Q_k{}'$ is $\exists$, if k is even and it is $\forall$, if k is odd.

Let us denote by $\mathcal{S}_k$ the set of all P-conditional Δ_0^*-formulas with k alternations of quantifiers, which are true in $HW(\mathcal{M})$. We use induction on k.

Case $k = 0$. By Lemma 2, the set of all true quantifier-free formulas $\mathcal{S}_0$ is P-computable and hence, is in $\Delta_0^p = \Sigma_0^p = \Pi_0^p$.

Case $k{+}1$. By the induction assumption, $\mathcal{S}_k$ is at the k-th level of the polynomial-time hierarchy.

Consider a Δ_0-formula $\varphi = \exists x_1 \in t_1 \exists x_2 \in t_2 \ldots \exists x_n \in t_n \psi(x_1, x_2, \ldots, x_n)$, where ψ is from $\mathcal{S}_k$. Let $\varphi' = \psi(head(a), head(tail(a)), head(tail(tail(a))), \ldots) \wedge head(a) \in t_1 \wedge head(tail(a)) \in t_2(head(a)) \ldots$. The formula φ is true iff there exists an a such that $\varphi'(a)$ is true. By by the induction assumption and Lemma 2, the set of all true formulas φ' is Π_k^p-computable and by Theorem 1, the length of a is bounded by a polynomial $p(|\varphi|)$. Then the set $\mathcal{S}_{k+1}$ of all true formulas φ is Σ_{k+1}^p-computable. The case with $\forall x \in t$ as the last quantifier is proved similarly. $\square$

5. Conclusions

We have shown that deciding the truth of Δ_0^*-formulas in a list superstructure has the same complexity as deciding the truth of quantified boolean formulas, provided the basic operations of the underlying structure are polynomially computable. If this is the case, then there exists a polynomially computable representation of its superstructure of finite lists. Δ_0^*-formulas are obtained as an extension of Δ_0-formulas with conditional terms, which employ an analog of the "if .. then .. else" operator in their definition. As has been previously shown in the literature, the extension of Δ_0 formulas with conditional terms is conservative: for any Δ_0^*-formula φ there is an equivalent Δ_0-formula of size is exponential in the size of φ. We have demonstrated however that this fact has no consequences for the complexity of deciding the truth of a Δ_0^*-formula in a given structure, since computing the value of a conditional term in a structure reduces to deciding the truth of polynomially many Δ_0-formulas. The polynomial complexity of Δ_0-formulas in hereditarily finite list structures gives the possibility to implement the concept of semantic programming as a language for applied problems, e.g., described by locally simple models [12].

References

[1] *Ershov Yu. L.*, Definability and computability. Consultants Bureau, New York (1996). MR1393198

[2] *Barwise, J.*, Admissible sets and structures. Springer, Berlin (1975). MR54:12519

[3] *Goncharov S. S. and Sviridenko D. I.*, Σ-programming, Transl. II. Ser., Amer. Math. Soc., no. 142, 101–121 (1989). MR835905

[4] *Goncharov S. S. and Sviridenko D. I.*, Σ-programming and its Semantics, Vychisl. Systemy, no. 120, 24–51 (1987) (in Russian). MR0995247

[5] *Goncharov S. S. and Sviridenko D. I.*, Theoretical Aspects of Σ-programming, Lect. Notes Comp. Sci., vol. 215, 169–179 (1986). Zbl 0621.68021

[6] *Ershov Yu. L., Goncharov S. S., and Sviridenko D. I.*, Semantic Programming, in: Information processing 86: Proc. IFIP 10th World Comput. Congress. Vol. 10, Elsevier Sci., Dublin, 1093–1100 (1986). Zbl 0606.68011

[7] *Ershov Yu. L., Goncharov S. S., and Sviridenko D. I.*, Semantic Foundations of Programming, in: Fundamentals of Computation Theory: Proc. Intern. Conf. FCT 87, Kazan, 116–122 (Lect. Notes Comp. Sci., vol. 278) (1987). Zbl 0642.68029

[8] *Goncharov S. S.*, Conditional Terms in Semantic Programming, Siberian Mathematical Journal, vol. 58, no. 5, 794–800 (2017). MR3766340

[9] *S. Arora and B. Barak*, Computational Complexity: a Modern Approach, Cambridge University Press (2009). MR2500087

[10] *C.H. Papadimitriou*, Computational complexity, Addison-Wesley (1994). MR1251285

[11] *L.J. Stockmeyer and A.R. Meyer*, Word Problems Requiring Exponential Time, Proc. Fifth Annual ACM Symposium on Theory of Computing, Austin, Texas, USA (1973), 1–9. MR 0418518

[12] *A. A. Malykh, A. V. Mantsivoda*, Document models, Vestnik of the Irkutsk State University, Ser. Matematika, vol. 21, 89—107 (2017), in russian. Zbl 1390.68613

Sergey Ospichev
Sobolev Institute of Mathematics,
pr. Koptyuga, 4,
630090, Novosibirsk, Russia
Novosibirsk State University,
Pirogova, 2,
630090, Novosibirsk, Russia
E-mail address: `ospichev@math.nsc.ru`

Denis Ponomarev
Sobolev Institute of Mathematics,
pr. Koptyuga, 4,
630090, Novosibirsk, Russia
A.P. Ershov Institute of Informatics Systems,
pr. Lavrentyeva, 6,
630090, Novosibirsk, Russia
Novosibirsk State University,
Pirogova, 2,
630090, Novosibirsk, Russia
E-mail address: `ponom@iis.nsk.su`