



Article

# Polynomial Analogue of Gandy's Fixed Point Theorem

Sergey Goncharov <sup>\*,†</sup>  and Andrey Nechesov <sup>\*,†</sup> 

Sobolev Institute of Mathematics, Academician Koptyug Ave., 4, 630090 Novosibirsk, Russia

\* Correspondence: s.s.goncharov@math.nsc.ru (S.G.); nechesov@math.nsc.ru (A.N.)

† These authors contributed equally to this work.

**Abstract:** The paper suggests a general method for proving the fact whether a certain set is p-computable or not. The method is based on a polynomial analogue of the classical Gandy's fixed point theorem. Classical Gandy's theorem deals with the extension of a predicate through a special operator  $\Gamma_{\Phi(x)}^{\Omega^*}$  and states that the smallest fixed point of this operator is a  $\Sigma$ -set. Our work uses a new type of operator which extends predicates so that the smallest fixed point remains a p-computable set. Moreover, if in the classical Gandy's fixed point theorem, the special  $\Sigma$ -formula  $\Phi(\bar{x})$  is used in the construction of the operator, then a new operator uses special generating families of formulas instead of a single formula. This work opens up broad prospects for the application of the polynomial analogue of Gandy's theorem in the construction of new types of terms and formulas, in the construction of new data types and programs of polynomial computational complexity in Turing complete languages.

**Keywords:** polynomial computability; p-computability; Gandy's fixed point theorem; semantic programming; polynomial operators;  $\Delta_0^p$ -operators; computer science



**Citation:** Goncharov, S.; Nechesov, A. Polynomial Analogue of Gandy's Fixed Point Theorem. *Mathematics* **2021**, *9*, 2102. <https://doi.org/10.3390/math9172102>

Academic Editors: Francesco Aldo Costabile, Maria I. Gualtieri and Anna Napoli

Received: 20 July 2021

Accepted: 27 August 2021

Published: 31 August 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

In both mathematics and programming, we are increasingly confronted with inductively given constructs. These constructs can be, for example, new types of terms and formulas in logic or programs and new data types in high-level programming languages that are inductively defined using basic tools. All these inductively generated sets can be viewed as the smallest fixed points of a suitable operator. Classical Gandy's theorem [1,2] allows us to inductively define some abstract set through the special operator  $\Gamma_{\Phi(x)}^{\Omega^*}$  [1] where the smallest fixed point will be a  $\Sigma$ -set. The  $\Sigma$ -set is most often not a computable set and, moreover, not a p-computable set. Therefore, the question arises of how to modify Gandy's theorem so that the smallest fixed point be a computable or a p-computable set. In this paper, we just talk about the construction of a  $\Delta_0^p$ -operator with the smallest fixed point being a p-computable set, which allows us to consider many inductive formulas definable constructions as some polynomially computable set.

## 2. P-Computability

Let  $\Sigma$  be a finite alphabet and  $A, B \subseteq \Sigma^*$  where  $\Sigma^*$  is the set of finite words over the alphabet  $\Sigma$ . We say that a function  $f : A \rightarrow B$  is p-computable [3–5] if there exists a one-tape/multi-tapes deterministic Turing machine  $T$  over the alphabet  $\Sigma$  and numbers  $C, p \in \mathbb{N}$  such that for all  $a$  from  $A$  the value of the function  $f(a)$  is computed on  $T$  in at most  $C \cdot |a|^p$  steps, where  $|a| \geq 1$ . The set  $A$  is called p-computable if its characteristic function  $\chi_A : \Sigma^* \rightarrow \{0, 1\}$  is p-computable. The class  $P$  of problems which can be solved in polynomial time will often be denoted by  $\Delta_0^p$  (accepted notation for the polynomial hierarchy). Therefore, the notation  $\Delta_0^p$ -function for a p-computable function and  $\Delta_0^p$ -set for a p-computable set will also be used. A partial function  $f : A \rightarrow B$  is called a partial p-computable function, if there exists a set  $D \subseteq A$  such that  $f : D \rightarrow B$  is a p-computable function (the Turing machine which represents  $f$  computes  $f(a)$  and stops at the final state

$q_0$ ) and  $f(a)$  is undefined (notation  $f(a) \uparrow^p$  or simple  $\uparrow^p$ ) if  $a \in A \setminus D$ , while the Turing machine on the element  $a \in A \setminus D$  stops at the final state  $q_1$  and number of steps does not exceed  $C * |a|^p$  steps. As we can see, the partial  $p$ -computable function is a  $p$ -computable function, but sometimes it is convenient to assume that the value of a  $p$ -computable function is undefined. We will also denote partial  $p$ -computable functions as  $\Delta_0^p$ -functions.

### 3. Word Splitting

Now let  $\Sigma_0$  be some finite alphabet and  $\Sigma = \Sigma_0 \cup \{<, >\} \cup \{, \}$  is a new alphabet obtained by adding new symbols (brackets and comma) to  $\Sigma_0$ . Word splitting is the following partial function  $R : \Sigma^* \rightarrow (\Sigma \cup \{\#\})^*$  such that:

$$R(w) = \begin{cases} w_1\#\dots\#w_n, & \text{where } w = \langle w_1, \dots, w_n \rangle \text{ and every } w_i \in \Sigma^* \text{ satisfies (1) or (2)} \\ \uparrow, & \text{otherwise} \end{cases}$$

(1)  $w_i \in \Sigma_0^*$

(2)  $w_i$  starts with a left bracket and the number of left brackets in the word is equal to the number of right brackets, while for any initial subword  $\alpha_i$  such that  $w_i = \alpha_i\beta_i$  it is not implemented, where the word  $w_i$  can be represented as some concatenation of the words  $\alpha_i, \beta_i \in \Sigma^*$  and  $|\alpha_i| \geq 1$ .

**Proposition 1.** *The word splitting is unique.*

**Proof.** Prove by contradiction. Let there be two different splittings  $R(w) = w_1\#\dots\#w_n$  and  $R(w) = l_1\#\dots\#l_k$  such that  $w = \langle w_1, \dots, w_n \rangle$  and  $w = \langle l_1, \dots, l_k \rangle$ . Then, by definition, either  $w_1, l_1 \in \Sigma_0^*$ , or  $w_1$  and  $l_1$  start with a left bracket and the number of right and left brackets for each word is the same. In the first case,  $w_1$  and  $l_1$  are the same. In the second case,  $w_1$  is the subword of  $l_1$  or  $l_1$  is the subword of  $w_1$ . Then, by definition, no proper subword starting with a left bracket can have an equal number of right and left brackets. Equality of words was also obtained. Further, in a similar way, we show that the remaining  $w_i = l_i$  and at the same time  $n = k$ .  $\square$

**Proposition 2.**  *$R(w)$  is  $\Delta_0^p$ -function.*

**Proof.** Consider a Turing machine  $T$  with two semi-tapes (hereafter called tapes) over the alphabet  $\Sigma \cup \{1, B, \#\}$  where  $B$  is an empty symbol:

- (1) The 1st tape: we will store the word  $w$ .
- (2) The 2nd tape: we will store the difference between the number of left and right brackets of the word  $w$ .

Algorithm of the multi-tapes machine:

- (1) If the first symbol on the first tape is not a left bracket, then  $T$  stops the work in the final state  $q_1$ . Otherwise,  $T$  replaces it on  $B$  symbol and goes on to the next steps.
- (2) If the second symbol in the word  $w$  is from  $\Sigma_0$ , then  $T$  reads the word  $w$  until it meets a symbol, not from  $\Sigma_0$ . If it is not a comma or a right bracket then  $T$  stops the work in state  $q_1$ .
- (3) If the second symbol is not from  $\Sigma_0$  and is not a left bracket, then  $T$  stops the work in the state  $q_1$ .
- (4) When  $T$  reads the left bracket of the word  $w$ , then  $T$  adds 1 on the second tape and shifts the head of the second tape to the right and when  $T$  reads the right bracket of the word  $w$ , then  $T$  replaces symbol 1 with  $B$  of the second tape and shifts the head of the second tape to the left.
- (5) If there are no more symbols 1 on the second tape when  $T$  reads the right bracket from the first tape, then the machine replaces the right bracket with  $B$  on the first tape. If there are no other symbols from  $\Sigma$  after this right bracket, then the machine stops work in the final state  $q_0$ , otherwise, in the final state  $q_1$ .
- (6) If  $T$  meets a comma on the first tape and there are no symbols 1 on the second tape, then  $T$  replaces this comma with  $\#$  symbol.

Computational complexity  $R(w)$ :

- (1)  $T$  reads the word  $w$  on the first tape periodically replacing the comma or brackets with symbol  $\#$ . The number of such steps does not exceed  $|w|$ .
- (2) On the second tape  $T$  writes or erases symbols 1. The number of such additions and removals does not exceed  $|w|$ .
- (3) Steps from (1) and (2) taken simultaneously. It turns out that the computational complexity  $t(R(w)) \leq |w|$ .  $\square$

Inductively define the notion rank of element  $r(w)$  for  $w \in \Sigma^*$ :

$$r(w) = \begin{cases} 0, & \text{if } R(w) \uparrow^p \\ \sup\{r(w_1), \dots, r(w_k)\} + 1, & \text{if } R(w) = w_1\# \dots \#w_k \end{cases}$$

#### 4. Generating Formulas and Families. False Element

Let  $\mathfrak{M}$  be a model of signature  $\sigma = \{c_1, \dots, c_r, f_1^{(m_1)}, \dots, f_s^{(m_s)}, R_1^{(p_1)}, \dots, R_t^{(p_t)}, P_1^{(1)}, \dots, P_n^{(1)}\}$  with the basic set  $M \subseteq \Sigma_0^*$ , where  $c_l$  is constant symbols ( $l \in [1, \dots, r]$ ),  $f_i$  is functional symbols ( $i \in [1, \dots, s]$ ),  $R_j$  is predicate symbols ( $j \in [1, \dots, t]$ ),  $P_k$  is unary predicate symbols,  $k \in [1, \dots, n]$ .  $P(\Sigma^*)$  is the set of all subsets of the set  $\Sigma^*$ .  $F_{P_1^+}, \dots, F_{P_n^+}$  is families (generating families) positive quantifier-free formulas (hereafter called generating formulas) of signature  $\sigma$  which can include unary predicates  $P_1, \dots, P_n$  with inputs of the form  $P_j(x_i)$ . Moreover, we require that for any free variable  $x_i$  in the formula  $\varphi_m \in F_{P_k^+}$  there should be no occurrences of the form  $P_j(x_i)$  and  $P_h(x_i)$  for each  $x_i$ , where  $j \neq h$ . This property will be called predicate separability.

The idea is to generate new elements in the form of lists  $\langle a_1, \dots, a_{n_m} \rangle$  obtained from  $a_1, \dots, a_{n_m} \in M$  such that  $\mathfrak{M} \models \varphi_m(a_1, \dots, a_{n_m})$  and then add this set of elements  $Q_i$  to the main set of the model where:

$$Q_i = \cup_{\varphi_m(x_1, \dots, x_{n_m}) \in F_{P_i^+}} \{ \langle a_1, \dots, a_{n_m} \rangle \mid \mathfrak{M} \models \varphi_m(a_1, \dots, a_{n_m}) \}$$

If we are to extend the main set of elements  $M$  of the model  $\mathfrak{M}$  to this new set of elements  $Q_i$ , then we need to redefine the functions on these new elements and redefine the truth of the predicates. It is clear that the functions on new elements will not be defined, so we will expand the basic set of elements  $M$  of the  $\mathfrak{M}$  model of signature  $\sigma$  with a special *false*-element to  $M \cup \{false\}$ . Next, we define the semantic meaning of terms and formulas in the  $\mathfrak{M}_{false}$  model for all elements from  $\Sigma^* \cup \{false\}$  and not only for  $M \cup \{false\}$ .

Since everywhere below only positive quantifier-free formulas with a positive occurrence in the form of  $P_i(x_j)$  for some  $P_i$  and  $x_j$  appear, then for these formulas on the model  $\mathfrak{M}_{false}$  we inductively define the values of functions and the truth of predicates as well as the truth of positive quantifier-free formulas  $\varphi_i, i \in I$ :

- (1)  $\mathfrak{M} \models \varphi_i(a_1, \dots, a_k)$  if and only if  $\mathfrak{M}_{false} \models \varphi_i(a_1, \dots, a_k)$  where  $a_1, \dots, a_k \in M$ .
- (2) the function value  $f_j(a_1, \dots, a_{n_j})$  equal *false*, if at least one  $a_i \in \Sigma^* \cup \{false\} \setminus M$ ,  $j \in [1, \dots, s]$
- (3) the function value  $f_j(t_1(\bar{a}), \dots, t_n(\bar{a}))$  equal *false* if at least the value of one of the terms  $t_j(\bar{a})$  equals *false*.
- (4) the formulas of the form  $false = t(a_1, \dots, a_n)$  including  $false = false$  will be considered false.
- (5) the formulas of the form  $a = a$  will be considered true for  $a \in M$  and false otherwise.
- (6) the formulas of the form  $R_i(t_1(\bar{a}), \dots, t_{n_i}(\bar{a}))$  will be considered false if at least one of the terms  $t_j(\bar{a})$  has value *false*.
- (7)  $\mathfrak{M} \models P(a)$  if and only if  $\mathfrak{M}_{false} \models P(a)$  where  $a \in M$ .
- (8)  $\Phi \& \Psi, \Phi \vee \Psi$  retain their standard definitions of truth.

Let us denote enrichment of the model  $\mathfrak{M}_{false}$  by  $\langle \mathfrak{M}_{false}, Q \rangle$  such that:

- (1)  $M \cup \{false\} \cup Q$  is a new main set.
- (2) All predicates  $R_i(t_1, \dots, t_{n_i})$  remain unchanged if the values of the terms  $t_1, \dots, t_{n_i}$  are from  $M$  and are *false* otherwise.

- (3) All predicates  $P_j(a)$  remain unchanged if  $a \in M$  and  $P_j(a)$  are *false* otherwise.
- (4) All functions  $f_i(a_1, \dots, a_n)$  remain unchanged for  $a_1, \dots, a_n \in M$  and have a *false* value otherwise.

Denote the expression  $\langle \mathfrak{M}_{false}, Q, P_i \rangle$  it is  $\langle \mathfrak{M}_{false}, Q \rangle$  enrichment at which the truth set of the predicate  $P_i$  is extended to  $P_i^{\mathfrak{M}_{false}} \cup Q$ .

**5. Fixed Points of Monotone Locally Finite Operators**

Let  $\mathfrak{M}_{false}$  be a model of signature  $\sigma$  and  $Q = (Q_1, \dots, Q_n), Q_i \subseteq \Sigma^*, i \in [1, \dots, n]$ . Then we introduce the notation:  $\mathfrak{M}_{false}^{(Q_1, \dots, Q_n)} = \langle \dots \langle \mathfrak{M}_{false}, Q_1, P_1 \rangle \dots \rangle, Q_n, P_n \rangle$ . Construct an operator:

$$\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}} : P(\Sigma^*) \times \dots \times P(\Sigma^*) \rightarrow P(\Sigma^*) \times \dots \times P(\Sigma^*), \tag{1}$$

which transfers  $n$ -th sets  $(Q_1, \dots, Q_n)$  to  $n$ -th sets  $(Q'_1, \dots, Q'_n)$  according to the following rule:  $Q'_i = Q_i \cup \bigcup_{\varphi_m(x_1, \dots, x_{k_m}) \in F_{P_i^+}} \{ \langle a_1, \dots, a_{k_m} \rangle \mid \mathfrak{M}_{false}^{(i-1)} \models \varphi_m(a_1, \dots, a_{k_m}) \}$ .

where  $\varphi_m \in F_{P_i^+}, a_1, \dots, a_{k_m} \in M^{(i-1)}$  and  $\mathfrak{M}_{false}^{(i-1)}$  is built on the model  $\mathfrak{M}_{false}$  of signature  $\sigma$  in the following way:

$$\mathfrak{M}_{false}^{(0)} = \mathfrak{M}_{false}, \dots, \mathfrak{M}_{false}^{(i)} = \langle \mathfrak{M}_{false}^{(i-1)}, Q'_i, P_i \rangle, \text{ where } i \in [1, \dots, n].$$

We fix a partial order  $\leq_n: (Q_1, \dots, Q_n) \leq_n (R_1, \dots, R_n)$ , if  $Q_i \subseteq R_i$  for all  $i \in [1, \dots, n]$

**Remark 1.** Operator  $\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}$  is monotone with respect to the order  $\leq_n$ , i.e.,

$$(Q_1, \dots, Q_n) \leq_n (R_1, \dots, R_n) \Rightarrow \Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}(Q_1, \dots, Q_n) \leq_n \Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}(R_1, \dots, R_n).$$

**Remark 2.** Operator  $\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}$  possesses the property of a fixed point, i.e.:

$$(Q_1, \dots, Q_n) \leq_n \Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}(Q_1, \dots, Q_n).$$

Associate the operator  $\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}$  with the sequence:  $\Gamma_0, \Gamma_1, \dots, \Gamma_t, \dots$ :

$$\Gamma_0 = \{ \emptyset, \dots, \emptyset \} \leq_n \dots \leq_n \Gamma_{t+1} = \Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}(\Gamma_t) \leq_n \dots \leq_n \Gamma_w = \bigcup_{k < w} \Gamma_k. \tag{2}$$

Let us denote projection onto the  $j$ -th coordinate by  $I_j(\Gamma_k) = Q_j$ .

We will say that operator  $\Gamma : P(\Sigma^*) \times \dots \times P(\Sigma^*) \rightarrow P(\Sigma^*) \times \dots \times P(\Sigma^*)$  is locally finite if for any  $X_1, \dots, X_n \subseteq \Sigma^*$  and any  $j \in [1, \dots, n]$  is done:

$$I_j(\Gamma(X_1, \dots, X_n)) = \bigcup_{X'_1 \subseteq X_1} \dots \bigcup_{X'_n \subseteq X_n} I_j(\Gamma(X'_1, \dots, X'_n)), \tag{3}$$

where  $X'_1, \dots, X'_n$  are finite sets.

**Proposition 3.** Operator  $\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}$  is locally finite.

**Proof.** Let  $X_1, \dots, X_n \subseteq \Sigma^*$ , where  $X'_i$  are finite sets.

$\Leftarrow$  Inclusion in equality (3) for operator  $\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}$  in one way is fulfilled by construction of our operator  $\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}$ .

$\Rightarrow$  Let  $w$  be from  $I_j(\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}(X_1, \dots, X_n))$ . We get that  $w$  is a finite list made up of a finite number of elements from  $M \cup X_1 \cup \dots \cup X_n$ . Mark all the elements involved in constructing  $w$  from  $X_j$  as  $C_j$  for all  $j \in [1, \dots, n]$ . Note that all sets  $C_j$  are finite and  $C_j \subseteq X_j$ . Therefore, narrowing our sets  $X_i$  to  $C_i$  we get  $w \in I_j(\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}(C_1, \dots, C_n))$ .  $\square$

**Proposition 4.** *The smallest fixed point of the operator  $\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}$  is reached in  $w$  steps.*

**Proof.** Claim that the fixed point of the operator  $\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}$  is reached in  $w$  steps following automatically from the fact that the operator  $\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}$  is monotone, has the fixed point property and is locally finite.  $\square$

### 6. Formulas Families $F_{P_i^+}^*$

Further, we will consider generating families of formulas of the form  $F_{P_i^+} = \{\varphi_m(\underline{x}_1, \dots, \underline{x}_{n_m}) \mid m \in N\}$  where  $\underline{x}_i$  is encoding the variable  $x_i$  with a string of  $v$  symbols length  $i$ . Let  $\epsilon$  be a string of symbols above the alphabet  $\{0, 1\}$  length  $n_m$ . Then the formula  $\varphi_m^\epsilon(\underline{x}_1, \dots, \underline{x}_{n_m})$  is obtained from  $\varphi_m(\underline{x}_1, \dots, \underline{x}_{n_m})$  replacing all occurrences of the form  $P_j(x_i)$  on  $i$ -th symbol in word  $\epsilon$ . The number of free variables in this formula may be less, nevertheless we leave their number in the notation for  $\varphi_m^\epsilon$  as before.

$$F_{P_i^+}^* = \{\varphi_m^\epsilon(\underline{x}_1, \dots, \underline{x}_{n_m}) \mid \varphi_m(\underline{x}_1, \dots, \underline{x}_{n_m}) \in F_{P_i^+}, \epsilon \in \{0, 1\}^* \text{ and } |\epsilon| = n_m\}$$

The formula  $\varphi_m^\epsilon(l_1, \dots, l_{n_m})$  is obtained from  $\varphi_m^\epsilon(\underline{x}_1, \dots, \underline{x}_{n_m})$  substituting free variables  $\underline{x}_i$  by the corresponding values  $l_i$  for all  $i \in [1, \dots, n_m]$ . Due to the predicate separability of the formula  $\varphi_m$  the maximum number of such occurrences in  $\varphi_m^\epsilon$  may not be more than  $n_m$ .

Define  $\Omega = \Sigma \cup \sigma \cup \{0, 1\} \cup \{v\} \cup \{\#\} \cup \{\vee, \&\} \cup \{(\ , )\}$  as a set of symbols such that any formula of the form  $\varphi_m(\underline{x}_1, \dots, \underline{x}_{n_m})$ ,  $\varphi_m(l_1, \dots, l_{n_m})$ ,  $\varphi_m^\epsilon(\underline{x}_1, \dots, \underline{x}_{n_m})$ ,  $\varphi_m^\epsilon(l_1, \dots, l_{n_m}) \in \Omega^*$ , where  $l_1, \dots, l_{n_m} \in \Sigma^*$ ,  $\varphi_m \in F_{P_j^+}$  for some  $j \in [1, \dots, n]$ .

Define a potentially generating formula as a formula  $\varphi_m(\underline{x}_1, \dots, \underline{x}_k)$  potentially generating an element  $l \in \Sigma^*$  such that  $R(l) = l_1\# \dots \#l_k$  and the following holds:

$$\mathfrak{M}_{false} \models \varphi_m^\epsilon(l_1, \dots, l_k)$$

for some signification  $\epsilon$ . If for any  $l \in \Sigma^*$  there is only one potentially generating formula in the family, then we can define a partial function  $\gamma_i : \Sigma^* \rightarrow \Omega^*$  that constructs from an element  $l \in \Sigma^*$  its potentially generating formula  $\varphi_m(\underline{x}_1, \dots, \underline{x}_k)$  if such a formula exists and is undefined otherwise  $\gamma_i(l) \uparrow$ . In the next chapter we will require for functions  $\gamma_i$  to be  $p$ -computable.

### 7. $\Delta_0^p$ -Models and $\Delta_0^p$ -Operators

Model  $\mathfrak{M}$  of the finite signature  $\sigma$  will be called a  $p$ -computable model ( $\Delta_0^p$ -model) [4–7] if all functions are  $p$ -computable functions, all predicates and the main set are  $\Delta_0^p$ -sets. If we want to mark the degree of the polynomial  $n$  and the constant  $C$ , we will write  $C$ - $n$ - $\Delta_0^p$ -model instead of writing  $\Delta_0^p$ -model. Sometimes, there will be records of the form  $C$ - $p$ - $\Delta_0^p$ . In the first case,  $p$  is the degree of the polynomial and in the second,  $\Delta_0^p$  is the designation for the first level of the polynomial hierarchy. Designation of  $C$ - $p$ - $\Delta_0^p$ -function will be also applied for functions and  $C$ - $p$ - $\Delta_0^p$ -set will be also applied for sets. Note that  $\mathfrak{M}_{false}$  will be a  $C$ - $p$ - $\Delta_0^p$ -model if such is the model  $\mathfrak{M}$ .

Let us call  $\Delta_0^p$ -operator the operator  $\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}$  from (1) if for some  $C, p \in N$  the following four properties hold:

(1)  $p$ -computable model:  $\mathfrak{M}$  is a  $C$ - $p$ - $\Delta_0^p$ -model.

(2) predicate separability, quantifier-free and positivity: each family  $F_{P_1^+}, \dots, F_{P_n^+}$  is either a finite or countable family of formulas, all formulas  $\varphi_j \in F_{P_i^+}$  are positive, quantifier-free, predicate-separable.

(3) uniqueness of the generating formula: for any two formulas  $\varphi_1(x_1, \dots, x_k), \varphi_2(x_1, \dots, x_k) \in F_{P_i^+}$  with the same number of free variables and for any signification  $\mathfrak{E} : P_j(x_i) \rightarrow \{0, 1\}, i \in [1, \dots, k], j \in [1, \dots, n]$  it is not true that there exists such significations as  $\epsilon_1$  and  $\epsilon_2$  consistent with  $\mathfrak{E}$  such that:

$$\text{there exists } l_1, \dots, l_k \text{ from } M \text{ such that } \mathfrak{M}_{false} \models \varphi_1^{\epsilon_1}(l_1, \dots, l_k) \& \varphi_2^{\epsilon_2}(l_1, \dots, l_k)$$

(4)  $p$ -computability of element: we also require that all functions  $\gamma_i$  should be  $C$ -( $p$ -1)- $\Delta_0^p$ -functions and families  $F_{P_i^+}^*$  -  $C$ - $p$ - $\Delta_0^p$ -families (time for checking  $t(\mathfrak{M}_{false} \models \varphi_m^{\epsilon}(l_1, \dots, l_k)) \leq C \cdot |l|^p$ , for all  $\varphi_m \in F_{P_i^+}$  and  $l_i \in \Sigma^* \cup \{false\}, i \in [1, \dots, k]$ ).

Note that the  $\Delta_0^p$ -operator thus defined retains all the original properties: it is monotone, has a fixed point property and is locally finite, and therefore the smallest fixed point of the operator is reached in  $w$  steps.

We say that the smallest fixed point  $\Gamma_w = (P_1, \dots, P_n)$  will be  $\Delta_0^p$ -set if any  $P_i$  is a  $\Delta_0^p$ -set, where  $i \in [1, \dots, n]$ . Let  $\gamma_i$  be the  $C$ -( $p$ -1)- $\Delta_0^p$ -function for  $\Delta_0^p$ -operator  $\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}^{\mathfrak{M}}$  and  $\varphi_m(\underline{x}_1, \dots, \underline{x}_k) \in F_{P_i^+}$  - potential generating formula for  $l$ , where  $R(l) = l_1 \# \dots \# l_k$  and all  $l_1, \dots, l_k \in \Sigma^*$ . Then the following lemma is true for any signification  $\varphi_m^{\epsilon}(\underline{x}_1, \dots, \underline{x}_k)$ :

**Lemma 1.**  $\varphi_m^{\epsilon}(\underline{x}_1, \dots, \underline{x}_k)$  is built according to the formula  $\varphi_m(\underline{x}_1, \dots, \underline{x}_k)$  and by signification  $\epsilon$  for the time not exceeding  $12 \cdot C \cdot |l|^{p-1}$ .

**Proof.** Consider the Turing machine  $T$  over  $\Omega$  alphabet consisting of five semi-tapes (hereafter called tapes):

- (1) the 1st tape: the formula  $\varphi_m(\underline{x}_1, \dots, \underline{x}_k)$  is written out.
- (2) the 2nd tape: the word  $\epsilon$  of length  $k$  is written out.
- (3) the 3rd tape: for variables.
- (4) the 4th tape: remembers the last position of the first tape.
- (5) the 5th tape: builds a new formula  $\varphi_m^{\epsilon}(\underline{x}_1, \dots, \underline{x}_k)$ .

Let the formula  $\varphi_m(\underline{x}_1, \dots, \underline{x}_k)$  be written out on the first tape and the second tape should contain the word  $\epsilon$ . The machine  $T$  starts to work in the extreme left position and reads the formula from the first tape. As soon as  $T$  reaches the word of the form  $P_j(x_i)$ ,  $T$  begins to read this word and writes out in parallel symbol 1 on the fourth tape for each symbol of  $P_j(x_i)$  and symbol 1 for each symbol  $v$  of  $P_j(x_i)$  on the third tape, moving in parallel, the machine head on the second tape containing the word  $\epsilon$  with a single delay. When all the symbols  $v \dots v(x_i)$  are read, the head on the second tape will observe symbol  $\epsilon_{i_1}$  which must be substituted for the word  $P_j(x_i)$ . Since the head position of the first tape is recorded on the fourth tape,  $T$  starts to overwrite on the first tape the word  $P_j(x_i)$  on symbols # and reduce in parallel the number of symbols 1 on the fourth tape. One as soon as there are no one-symbols left on the fourth tape, then  $T$  write the symbol  $\epsilon_{i_1}$  to the first tape. Then  $T$  returns the heads of second, third and fourth tapes to the extreme left position and continue to sequentially find and replace the remaining occurrences of the form  $P_j(x_r)$  on the first tape and replace them with symbols # and  $\epsilon_r$ . After all the replacements  $T$  must return the head of the first tape to the extreme left position and starts copying the formula of the first tape to the fifth tape while skipping the symbols #.

Calculate the total operating time of such a machine  $T$ :

- (1) The machine  $T$  works with the formula  $\varphi_m(\underline{x}_1, \dots, \underline{x}_{n_m})$  on the first tape which includes words such as  $P_j(x_i)$ . The length of this formula does not exceed  $C \cdot |l|^{p-1}$ . In total, the number of steps does not exceed three lengths of  $\varphi_m(\underline{x}_1, \dots, \underline{x}_{n_m})$ .
- (2) On the second tape, the machine does not change the word  $\epsilon$ , simply reads it in parallel

with the symbols  $v$  from the first tape and periodically returns the head to the extreme left position. The total number of shifts to the right of the machine head of the second tape does not exceed the length of the word on the first tape. The same goes for the number of the machine head shifts to the left. Therefore, on this tape, there will be no more than  $2 \cdot C \cdot |l|^{p-1}$  steps.

(3) On the third tape, the last monitored variable is written out. The number of the machine head shifts to the right and to the left does not exceed  $2 \cdot C \cdot |l|^{p-1}$  on this tape.

(4) For the fourth tape it is also does not exceed  $2 \cdot C \cdot |l|^{p-1}$ .

(5) To copy the final word from the first tape to the fifth and taking into account the preliminary setting of the head of the first tape to the extreme left position, it will also take no more than  $2 \cdot C \cdot |l|^{p-1}$ .  $\square$

Let  $\varphi_m(x_1, \dots, x_k) \in F_{P_i^+}$  be potentially generative formula for an element  $l$ .

**Lemma 2.**  $\varphi_m^\varepsilon(l_1, \dots, l_k)$  is built by word  $l$  and the formula  $\varphi_m^\varepsilon(x_1, \dots, x_k)$  for the time not exceeding  $4 \cdot C \cdot |l|^p$ .

**Proof.** Consider a Turing machine  $T$  over alphabet  $\Omega$  that also consists of three semi-tapes (hereafter called tapes):

(1) the 1st tape: the formula  $\varphi_m^\varepsilon(x_1, \dots, x_k)$  is written out, where the length of the formula does not exceed  $C \cdot |l|^{p-1}$ .

(2) the second tape: the word  $w_2 = \#R(l) = \#l_1\#\dots\#l_k$  written out, where the length of the word does not exceeding  $|l|$ .

(3) the third tape: builds a new formula  $\varphi_m^\varepsilon(l_1, \dots, l_k)$ .

The machine starts to work with the formula of the first tape, if necessary simultaneously copying the result to the third tape. If the machine  $T$  on the first tape reads a symbol that is not  $v$ , then  $T$  copies it to the third tape. If  $T$  reads the symbol  $v$  on the first tape, then  $T$  starts the process of finding the corresponding  $l_i$  for replacement. When the machine  $T$  reads the  $i$ -th symbol  $v$  successively from the first tape,  $T$  transfers the machine head of second tape to the  $i$ -th symbol  $\#$  that comes before the corresponding  $l_i$ . When  $T$  reads all symbols  $v$  successively from first tape, then the machine will write the corresponding  $l_s$  from second tape to the third tape. By repeating this algorithm on the third tape the word  $\varphi_m^\varepsilon(l_1, \dots, l_k)$  will eventually be written.

Calculate the total operating time of such a machine  $T$ :

(1) the machine  $T$  reads a word from the first tape or just stands and waits for further reading. The number of movements to the right does not exceed  $C \cdot |l|^{p-1}$

(2) on the second tape, the machine head moves both to the right and to the left, but again only reading. Therefore, the number of steps does not exceed  $C \cdot |l|^{p-1} \times 2 \cdot |l| \leq 2 \cdot C \cdot |l|^p$ .

(3) the third tape: the number of steps does not exceed  $C \cdot |l|^{p-1}$ .  $\square$

### 8. A Polynomial Analogue of Gandy's Theorem

Let  $\Gamma_w$  from equality (2) be the smallest fixed point of  $\Delta_0^p$ -operator  $\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}$ , then the next theorem is true:

**Theorem 1 (polynomial analogue of Gandy's theorem).** *The smallest fixed point  $\Gamma_w$  of  $\Delta_0^p$ -operator  $\Gamma_{F_{P_1^+}, \dots, F_{P_n^+}}$  is a  $\Delta_0^p$ -set.*

**Proof.** The main idea of the proof is to show that the time for checking the truth of the formula  $P_i(l)$  on  $\mathfrak{M}_{false}^{\Gamma_w}$  does not exceed the time  $k \cdot C \cdot r(l) \cdot |l|^p$ , where  $k$  and  $C$  are fixed constants and  $r(l)$  is the rank of the element  $l$  and  $r(l) \geq 1, i \in [1, \dots, n]$ . Since the rank  $r(l) < |l|$ , we get that for any  $l$  the complexity does not exceed  $k \cdot C \cdot |l|^{p+1}$ .

Without loss of generality, we show this for  $P_1(l)$ , assuming in the induction step that this estimate is true for all  $P_i(l_j)$ , where  $r(l_j) < r(l)$  and  $i \in [1, \dots, n]$ .

Using the induction by complexity  $r(l)$  we show that  $t(P_1(l)) \leq 25 \cdot C \cdot r(l) \cdot |l|^p$ , where the

constant  $C$  is the maximum for all constants that participate in the splitting function  $R(l)$ , in functions  $\gamma_i$  and in the algorithm for checking the truth of the formula  $\varphi_m^\epsilon(l_1, \dots, l_{n_m})$ .

Induction base  $r(l) = 1$ :

Case 1:  $\gamma_i(l) \uparrow^p$ , then the formula  $P_1(l)$  is false.

Case2:  $\gamma_i(l) = \varphi_m(x_1, \dots, x_k)$ , then  $R(l) = l_1 \# \dots \# l_k$  and all elements of  $l_i$  (where  $i \in [1, \dots, k]$ ) are either elements of the base set  $M$  or elements from  $\Sigma^*$  for which  $R(l_i) \uparrow^p$ . Given all  $P_i(l_i)$  are false on  $\mathfrak{M}_{false}^{\Gamma_w}$ , we can create  $\varphi_m^\epsilon(x_1, \dots, x_k)$  from the potentially generating formula  $\varphi_m(x_1, \dots, x_k)$  for element  $l$ , where  $\epsilon = 0 \dots 0$  and  $|\epsilon| = k$ . We get:

$$\begin{aligned} \mathfrak{M}_{false}^{\Gamma_w} \models P_1(l) \text{ if and only if } \mathfrak{M}_{false}^{\Gamma_w} \models \varphi_m(l_1, \dots, l_k) \text{ if and only if } \mathfrak{M}_{false}^{\Gamma_w} \models \varphi_m^\epsilon(l_1, \dots, l_k) \\ \text{if and only if } \mathfrak{M}_{false}^\epsilon \models \varphi_m^\epsilon(l_1, \dots, l_k) \end{aligned}$$

The time required to construct a potentially generating formula  $\varphi_m(x_1, \dots, x_k)$  using  $l$  does not exceed  $C \cdot |l|^{p-1}$ . Next, we build  $\varphi_m^\epsilon(x_1, \dots, x_k)$  and  $\varphi_m^\epsilon(l_1, \dots, l_k)$ . The time required for this does not exceed  $12 \cdot C \cdot |l|^{p-1}$  and  $4 \cdot C \cdot |l|^p$  (Lemmas 1 and 2). Verifying the truth of the last formula for  $\mathfrak{M}_{false}^\epsilon$  does not exceed  $C \cdot |l|^p$ . Summing everything up, we get that the verification time does not exceed  $25 \cdot C \cdot r(l) \cdot |l|^p$ .

The induction step: let our assumption be true for  $r(l) = s$ . We will show this for  $s + 1$ :

Case 1:  $\gamma_i(l) \uparrow^p$ , then the formula  $P_1(l)$  is false. We get it in time:

$$t(P_1(l)) \leq C \cdot |l|^{p-1} \leq 25 \cdot C \cdot r(l) \cdot |l|^p$$

Case 2:  $\gamma_i(l) = \varphi_m(x_1, \dots, x_k)$

$$\begin{aligned} \mathfrak{M}_{false}^{\Gamma_w} \models P_1(l) \text{ if and only if } \mathfrak{M}_{false}^{\Gamma_w} \models \varphi_m(l_1, \dots, l_k) \text{ if and only if } \mathfrak{M}_{false}^{\Gamma_w} \models \varphi_m^\epsilon(l_1, \dots, l_k) \\ \text{if and only if } \mathfrak{M}_{false}^\epsilon \models \varphi_m^\epsilon(l_1, \dots, l_k) \end{aligned}$$

where  $\epsilon$  string of symbols  $\epsilon_i$  such that  $\epsilon_i = 1$  if formula  $P_j(l_i)$  is true on  $\mathfrak{M}_{false}^{\Gamma_w}$  and 0 otherwise.

Let's calculate the time spent on all transitions:

- (1) constructing a potentially generating formula  $\varphi_m(x_1, \dots, x_k)$  using  $l$  in time  $C \cdot |l|^{p-1}$
- (2) determining the truth of all predicates  $P_{i_1}(l_1), \dots, P_{i_k}(l_k)$  which are included in the formula. By the induction hypothesis, we obtain:

$$\sum_{j=1}^k t(P_{i_j}(l_j)) \leq \sum_{j=1}^k 25 \cdot C \cdot r(l_j) \cdot |l_j|^p \leq 25 \cdot C \cdot (r(l) - 1) \cdot |l|^p.$$

- (3) further, we fix the signification  $\epsilon : P_{i_j}(x_i) \rightarrow \{0, 1\}$  considering whether the predicate  $P_{i_j}(l_i)$  is true or false, if the formula does not include any of the predicates  $P_{j_i}$  for the variable  $x_i$ , then we determine the truth for  $P_1(x_i)$  by default.

(4) By the formula  $\varphi_m(x_1, \dots, x_k)$  and by the signification  $\epsilon$  we construct  $\varphi_m^\epsilon(x_1, \dots, x_k)$ . The time required for this does not exceed  $12 \cdot |l|^{p-1} \leq 12 \cdot |l|^p$ .

(5) By the formula  $\varphi_m^\epsilon(x_1, \dots, x_k)$  and  $l$  we construct  $\varphi_m^\epsilon(l_1, \dots, l_k)$ . The time required for this does not exceed  $4 \cdot C \cdot |l|^p$ .

If we sum up all the time of calculations, then we get the following:

$$\begin{aligned} t(P_1(l)) &\leq \sum_{i=1}^k (25 \cdot C \cdot r(l_i) \cdot |l_i|^p) + 25 \cdot C \cdot |l|^p \leq \\ &\leq 25 \cdot C \cdot (r(l) - 1) \cdot |l|^p + 25 \cdot C \cdot |l|^p \leq 25 \cdot C \cdot r(l) \cdot |l|^p \end{aligned}$$

We have shown that for any element  $l$  of rank  $r(l)$  in time  $25 \cdot C \cdot r(l) \cdot |l|^p$  we determine the fact whether it belongs to the predicate  $P_1$ . Since  $r(l)$  is always less than  $|l|$ , we can write the following:

$$t(P_1(l)) \leq 25 \cdot C \cdot r(l) \cdot |l|^p \leq 25 \cdot C \cdot |l|^{p+1}.$$

□



### 9. Corollaries and Applications

For the  $\Delta_0^p$ -model  $\mathfrak{M}$  as an application of the polynomial analogue of Gandy’s theorem, we present several corollaries. Some of these corollaries have already been proven earlier by other authors using other methods, some are presented for the first time.

Let the model  $\mathfrak{M}$  have a one-place predicate  $U$  that selects the elements of the main set  $M$  and a distinguished one-place predicate  $List = \emptyset$  (a predicate that will select list elements), then we will show how easy it is to prove the following statement on hereditarily finite lists  $HW(M)$  which was already proven earlier in [8] but using a different technique:

**Corollary 1.** *If  $\mathfrak{M}$  is a  $\Delta_0^p$ -model, then  $HW(M)$  is a  $\Delta_0^p$ -set.*

**Proof.** A countable generating family of formulas  $F_{List^+}$  is as follows:

$$\varphi_n : \&_{i=1}^n (U(x_n) \vee List(x_n)), n \in \mathbb{N}$$

This family of formulas is predicate-separable, all formulas are positive quantifier-free, and the predicate  $List$  is included in formulas positively. We can easily see that the operator  $\Gamma_{F_{List^+}}^{\mathfrak{M}}$  is a  $\Delta_0^p$ -operator.  $\square$

Let the signature  $\sigma$  have the form:  $\sigma = \{c_0, \dots, c_k, f_1^{(m_1)}, \dots, f_s^{(m_s)}, R_1^{(p_1)}, \dots, R_t^{(p_t)}\}$ . Consider the model  $\mathfrak{N}$  with the basic set of elements  $N$  and signatures  $\sigma = \{\underline{1}, s^{(1)}\}$ . The interpretation of the constant  $\underline{1}$  will be 1 and  $s$ -the standard successor function. Further, an entry of the form  $\underline{n+1}$  will mean a term of the form  $n$ -fold application of the function  $s$  to  $\underline{1}$ .

**Corollary 2.** *The set of quantifier-free formulas of signature  $\sigma$  is a  $\Delta_0^p$ -set.*

**Proof.** The process of constructing auxiliary  $\Delta_0^p$ -sets using generating families for the corresponding predicates in the  $\Delta_0^p$ -model  $\mathfrak{N}$  is as follows:

- (1) Constants:  $F_{Cons^+} : \varphi_i : (x_1 = \underline{1}) \& (x_2 = \underline{i}), i \in [1, \dots, k]$
- (2) Variables:  $F_{Var^+} : \varphi_i : (x_1 = \underline{2}) \& (x_2 = \underline{i}), i \in N$
- (3) Function symbols:  $F_{Func^+} : \varphi_i : (x_1 = \underline{3}) \& (x_2 = \underline{i}), i \in [1, \dots, s]$
- (4) Predicate symbols:  $F_{R^+} : \varphi_i : (x_1 = \underline{4}) \& (x_2 = \underline{i}), i \in [1, \dots, t]$ .
- (5) Terms that are not constants and variables:

$$F_{Term_1^+} : \varphi_i : (x_1 = \underline{5}) \& Func(x_2) \& \&_{i=3}^{n_i+2} (Term_1(x_i) \vee Cons(x_i) \vee Var(x_i))$$

- (6) The set of standard terms:  $F_{Term^+} : F_{Term_1^+} \cup F_{Cons^+} \cup F_{Var^+}$

Generating family for quantifier-free formulas  $F_{Free^+}$ :

- (1)  $\varphi_1(R_i) : (x_1 = \underline{8}) \& R(x_2) \& \&_{i=3}^{p_i+2} Term(x_i)$
- (2)  $\varphi_2(P_i) : (x_1 = \underline{9}) \& P(x_2) \& Term(x_i)$
- (3)  $\varphi_3(=) : (x_1 = \underline{10}) \& Term(x_2) \& Term(x_3)$
- (4)  $\varphi_4(\&) : (x_1 = \underline{11}) \& Free(x_2) \& Free(x_3)$
- (5)  $\varphi_5(\vee) : (x_1 = \underline{12}) \& Free(x_2) \& Free(x_3)$
- (6)  $\varphi_6(\rightarrow) : (x_1 = \underline{13}) \& Free(x_2) \& Free(x_3)$
- (7)  $\varphi_7(\neg) : (x_1 = \underline{14}) \& Free(x_2) \& Free(x_3) \quad \square$

Define the signature  $\sigma' = \sigma \cup \{Cons, Var, Func, Term, Free\} \cup \{\in^{(2)}, \subseteq^{(2)}\}$ .

**Corollary 3.** *The set of  $\Delta_0$ -formulas [9] signature  $\sigma'$  is a  $\Delta_0^p$ -set.*

**Proof.** Define a family of  $\Delta_0$ -formulas  $F_{D0^+}$ . Just as in Corollary 2, we write out generating formulas for terms and formulas, with the only difference that we also add formulas for the above predicates:

- (8)  $\varphi_8(\in) : (x_1 = \underline{15}) \& Term(x_2) \& Term(x_3)$
- (9)  $\varphi_9(\subseteq) : (x_1 = \underline{16}) \& Term(x_2) \& Term(x_3)$

We also write out generating formulas for  $(\exists x_k \in t)\varphi(\bar{x})$ ,  $(\forall x_m \in t)\varphi(\bar{x})$ ,  $(\exists x_t \subseteq t)\varphi(\bar{x})$ ,  $(\forall x_t \subseteq t)\varphi(\bar{x})$ :

(10)  $\varphi_{10}(\exists x_k \in t(\bar{x})) : (x_1 = \underline{17}) \& \text{Var}(x_2) \& \text{Term}(x_3) \& D0(x_4)$   
 (11)  $\varphi_{11}(\forall x_m \in t(\bar{x})) : (x_1 = \underline{18}) \& \text{Var}(x_2) \& \text{Term}(x_3) \& D0(x_4)$   
 (12)  $\varphi_{12}(\exists x_t \subseteq t(\bar{x})) : (x_1 = \underline{19}) \& \text{Var}(x_2) \& \text{Term}(x_3) \& D0(x_4)$   
 (13)  $\varphi_{13}(\forall x_t \subseteq t(\bar{x})) : (x_1 = \underline{20}) \& \text{Var}(x_2) \& \text{Term}(x_3) \& D0(x_4) \quad \square$

**Corollary 4.** *The set of conditional terms of signature  $\sigma'$  and  $\Delta_0^*$ -formulas is a  $\Delta_0^p$ -sets [9].*

**Proof.** This is where the approach gets more interesting. We need to simultaneously generate both conditional terms and formulas containing these conditional terms. Therefore, we construct two generating families:  $F_{T\text{Cond}^+}$ ,  $F_{F\text{Cond}^+}$ . In addition to generating formulas for standard terms in  $F_{T\text{Cond}^+}$ , we add countably many generating formulas for conditional terms:

(8)  $\varphi_{k+8} : (x_1 = \underline{21}) \&_{i=1}^k (T\text{Cond}(x_{2i}) \& F\text{Cond}(x_{2i+1})) \& T\text{Cond}(x_{2k+2})$ ,  $k \in \mathbb{N}$ .

The family  $F_{F\text{Cond}^+}$  is defined by the same generating formulas as the family  $F_{D0^+}$ , with the only difference that the predicates *Term* must be replaced with *TCond* everywhere.  $\square$

## 10. Conclusions

This work is a starting point for building a methodology for developing fast and reliable software. In this work, we study sufficient conditions for the  $\Delta_0^p$ -operator under which the smallest fixed point remains a  $\Delta_0^p$ -set. This allows us to create new elements and data types. Moreover, there are polynomial algorithms for checking the fact whether a certain element belongs to a given data type or not. The question of programming methodology is also of interest: which constructs can be used and which not for creating programs, if we want our programs to be polynomially computable. With the help of the main theorem of our paper and the theorems from the works [8–14] it is already possible to develop logical programming languages, with programs being of polynomial computational complexity.

**Author Contributions:** Conceptualization, S.G. and A.N.; methodology, S.G. and A.N.; formal analysis, S.G.; validation, S.G.; investigation, S.G. and A.N.; writing—original draft preparation, A.N.; writing—review and editing, A.N.; supervision, S.G.; project administration, S.G.; software, A.N. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Barwise, J. *Admissible Sets and Structures*; Springer: New York, NY, USA, 1975.
2. Ershov, Y.L. *Definability and Computability*; Springer: New York, NY, USA, 1996.
3. Lewis, H.; Papadimitriou, C. *Elements of the Theory of Computation*; Prentice-Hall: Upper Saddle River, NJ, USA, 1998.
4. Alaev, P.E. Structures Computable in Polynomial Time. *Algebra Log.* **2017**, *55*, 421–435. [[CrossRef](#)]
5. Alaev, P.E. Existence and Uniqueness of Structures Computable in Polynomial Time. *Algebra Log.* **2016**, *55*, 72–76. [[CrossRef](#)]
6. Alaev, P.E.; Selivanov, V.L. Polynomial computability of fields of algebraic numbers. *Dokl. Math.* **2018**, *98*, 341–343. [[CrossRef](#)]
7. Cenzer, D.; Remmel, J. Polynomial-time versus recursive models. *Ann. Pure Appl. Log.* **1991**, *54*, 17–58. [[CrossRef](#)]
8. Ospichev, S.S.; Ponomaryov, D.K. On the complexity of formulas in semantic programming. *Sib. Electron. Math. Rep.* **2018**, *15*, 987–995.
9. Goncharov, S.S. Conditional Terms in Semantic Programming. *Sib. Math. J.* **2017**, *58*, 794–800. [[CrossRef](#)]
10. Ershov, Y.L.; Goncharov, S.S.; Sviridenko, D.I. Semantic programming. In Proceedings of the Information Processing 86: IFIP 10th World Computer Congress, Dublin, Ireland, 1–5 September 1986; Volume 10, pp. 1113–1120

11. Goncharov, S.S.; Sviridenko, D.I. Logical language of description of polynomial computing. *Dokl. Math.* **2019**, *99*, 121–124. [[CrossRef](#)]
12. Goncharov, S.S.; Sviridenko, D.I. Recursive terms in semantic programming. *Sib. Math. J.* **2018**, *59*, 1014–1023. [[CrossRef](#)]
13. Goncharov, S.S.; Ospichev, S.S.; Ponomaryov, D.K.; Sviridenko, D.I. The expressiveness of looping terms in the semantic programming. *Sib. Electron. Math. Rep.* **2020**, *17*, 380–394. [[CrossRef](#)]
14. Goncharov, S.S.; Sviridenko, D.I.  $\Sigma$ -programming. *Transl. II. Ser. Am. Math. Soc.* **1989**, *142*, 101–121.