UDC 681.3

ELEMENTS OF CONCRETE ALGORITMICS: COMPUTABILITY AND SOLVABILITY

O.I. Provotar, O.O. Provotar

Taras Shevchenko national university of Kyiv

Abstract. An approach to proving the fundamental results of the theory of recursive functions using specific algorithms is consider. For this, the basic constructions of the algorithm are describing exactly and Church's thesis for more narrow classes of algorithmically computational functions is specified (concretized). Using this approach, the belonging of functions to classes of algorithmically computable is argued by the construction of the corresponding algorithms.
Ключові слова: теза Чорча, розв'язність, універсальна функція.
Ключевые слова: тезис Черча, разрешимость, универсальная функция.
Key words: Church thesis, solvability, universal function.
Анотація. Розглядається підхід до доведення фундаментальних результатів теорії рекурсивних функцій за допомогою використання конкретних алгоритмів. Для цього точно описуються основні конструкції алгоритму і переформульовується (конкретизується) теза Чорча для більш вузьких класів алгоритмічно обчислюваних функцій. За допомогою такого підходу належність функцій до класів алгоритмічно обчислюваних аргументується побудовою відповідних алгоритмів.
Аннотация. Рассматривается подход к доказательству фундаментальных результатов теории рекурсивных функций с помощью использования конкретных алгоритмов. Для этого точно описываются основные конструкции алгоритма и уточняется (конкретизируется) тезис Черча для более узких классов алгоритмически вычислительных функций. С помощью такого подхода принадлежность функций к классам алгоритмически вычислимых аргументируется построением соответствующих алгоритмов.

Introduction

Everyone knows Church's thesis [1-3] that the class of algorithmically calculated functions coincides with the class of partially recursive functions. The class of partially recursive functions is determined mathematically accurately. Therefore, this thesis can be used both to prove the algorithmic computability of functions and to prove that the function is not algorithmically computable. To do this, we only need to show that such a function belongs to the class of partially recursive functions and in this case it is algorithmically calculated, or does not belong and, accordingly, is not algorithmically calculated.

It is pretty difficult to show the algorithmic computability of functions by its belonging to the class of partially recursive functions, except for the simplest functions. In addition, in each case it requires the construction of a mathematical model of the function in the form of a term from the calculated operations on the basic functions. Basic functions, as we know [1,2], are called the simplest functions

$$o(x) = 0,\ s(x) = x + 1 \text{ and selector functions } I_m^n(x_1, \dots, x_n) = x_m,$$

where $n \geq m \geq 1$. The main computational operations will be superposition operations $\boldsymbol{S}^{n+1}$, primitive recursion $\boldsymbol{R}$ and minimization $\boldsymbol{M}$.

For example, a superposition operation $\boldsymbol{S}^{n+1}$ allows from function $g(x_1, \dots, x_n)$ and functions $g_1(x_1, \dots, x_m)$, ... , $g_n(x_1, \dots, x_m)$, create a function

$$f(x_1, \dots, x_m) = g(g_1(x_1, \dots, x_m), \dots, g_n(x_1, \dots, x_m)),$$

which is denoted by the term $\boldsymbol{S}^{n+1}(g, g_1, \dots, g_n)$.

A natural question about the possibility of using concrete algorithms to prove the fundamental results of the theory of recursive functions without constructing appropriate computational terms arises. Or, in other words, to tell in a language understandable to programmers about the main results of the theory of algorithms (recursive functions), in particular about the problems of computability and solvability. Therefore, the purpose of the work is to offer approaches and means to solve the problem. To do this, it is proposed to accurately describe the basic constructions of the algorithm and reformulate (concretize) Chorch's thesis for narrower classes of algorithmically calculated functions.

Concretization of the concept of algorithm

Next, the algorithms will be defined as syntactically correct constructions in the PseudoPascal language, which is a simplified dialect of the Pascal language. The operators of this language will be the following:

< identifier > = = < word >
< operator > = = < identifier > = < arithmetic expression >
< operator > = = **if** < relation > **then** < operator > **else** < operator >
< operator > = = **while** < relation > **do** {< operator >, … , < operator >}
< operator > = = **for** < relation > **to** < relation > {< operator >, … , < operator >}

We concretize Church's thesis for classes of primitively recursive, recursive and partially recursive functions, respectively.

1. A class of functions calculated by the algorithms defined everywhere without using the **while … do** operator. This class is described as follows:

1) We assume that the simplest functions

$$o(x) = 0,$$
$$s(x) = x + 1 \text{ та функції-селектори}$$
$$I_m^n \quad (x_1, \dots, x_n) = x_m, \text{ де } n \geq m \geq 1$$

are calculated by the algorithms defined everywhere without using the operator **while … do**, and therefore belong to this class..

2) All functions that are calculated by the algorithms defined everywhere without using an operator **while … do** and in their calculation auxiliary functions use, which are calculated by the algorithms defined everywhere without using of the operator **while … do** also belong to this class. Thus, the following thesis is true.

Thesis 1. A class of functions which are calculated by algorithms defined everywhere without using of an operator **while … do** coincides with the class of primitive recursive functions.

2. A class of functions which are calculated by algorithms defined everywhere. This class is described as follows:

1) All primitively recursive functions belong to this class.

2) All functions which are calculated by algorithms defined everywhere and at their calculation the functions which are calculated by algorithms defined everywhere are used also belong to this class.. Thus, the following thesis is true.

Thesis 2. The class of functions which are calculated by algorithms defined everywhere coincides with the class of recursive functions.

3. A class of functions which are calculated by arbitrary algorithms. This class is described as follows:

1) All recursive functions belong to this class.

2) All functions which are calculated by arbitrary algorithms and at their calculation the functions which are calculated by arbitrary algorithms are used also belong to this class. Thus, the following thesis is true.

Thesis 3. The class of functions calculated by arbitrary algorithms coincides with the class of partially recursive functions.

Computability

We show how to prove the algorithmic computability of functions using the above theses. Suppose we need to prove that function

$$f(x,y) = \begin{cases} [x/y], & y \neq 0 \\ x, & y = 0 \end{cases}$$

is primitive recursive function.

Consider the sequence

$$1y \,\square\, x,\ 2y \,\square\, x,\ \dots,\ [x/y] \,\square\, x,\ \dots,\ xy \,\square\, x.$$

Since the fraction $[x/y]$ means how many times the number y is "placed" in the number x, then $[x/y]$ is equal to the number of zeros in this sequence. Really, if, for example, y is twice "placed" in the number x, then

$$1y \,\square\, x = 0,\ 2y \,\square\, x = 0, \text{ a } 3y \,\square\, x \neq 0.$$

Therefore, the algorithm for calculating the function is as follows:

```
function f(x, y)
    begin
      s = 0
      if y = 0 then f = x
      else {for i = 1 to x
            if iy □ x = 0 then s = s + 1
            f = s}
     end.
```

Hence the function $f(x, y)$ is primitive recursive function.

If all pairs of natural numbers are arranged in sequence

$$<0, 0>, <0, 1>, <1, 0>, <0, 2>, <1, 1>, <2, 0>, <0, 3>, \dots ,$$

that is, arrange all the pairs so that the pair $<x, y>$ goes earlier than a pair $<u, v>$ if

$$x + y < u + v,$$

or if

$$x + y = u + v \text{ and } x < u,$$

then the relation

$$<x, y> \leftrightarrow n,$$

where n is the pair number in this sequence, specifies the bijection between the set of pairs and the set of natural numbers,.

This bijection is called the Cantor numbering of pairs of numbers and is denoted $\boldsymbol{c}(x, y)$, that is $\boldsymbol{c}(x, y)$ is the number of pair $<x, y>$ in the Cantor sequence. The left and right elements of the pair $<x, y>$ with number n define the functions $\boldsymbol{l}(n)$ and $\boldsymbol{r}(n)$, which are called the left and right coordinate functions.

We show that the functions $\boldsymbol{c}(x, y)$, $\boldsymbol{l}(n)$, $\boldsymbol{r}(n)$ are primitive recursive function. Really, the function $\boldsymbol{c}(x, y)$ is calculated by the following algorithm:

```
function c(x, y)
    begin
        s = 0
      for i = 0 to (x + y)
          s = s + i
       for i = 0 to (x + y)
          {j = (x + y) □ i
           if  x = i ∧ y = j then k = i}
       c = s + k
    end.
```

Given that $\boldsymbol{l}(n) \le n$, $\boldsymbol{r}(n) \le n$ function $\boldsymbol{l}(n)$ is calculated by the algorithm:

```
function l(n)
    begin
       for i = 0 to n
       for j = 0 to n
            if c(i, j) = n  then l = i
    end,
```

and the function $\boldsymbol{r}(n)$ is calculated by an algorithm:

```
function r(n)
    begin
       for i = 0 to n
       for j = 0 to n
              if c(i, j) = n  then r = j
    end.
```

So, $\boldsymbol{c}(x, y)$, $\boldsymbol{l}(n)$ and $\boldsymbol{r}(n)$ are primitive recursive functions.

We show that the function

$$f(x) = [e \cdot x]$$

is primitive recursive functions.

Really, inequality holds

$$xS_n < ex < x(S_n + 1/n!n),$$

where

$$e = 1 + 1/1! + 1/2! + \ldots + 1/n! + \theta/n!\cdot n,\ 0 < \theta < 1,$$
$$S_n = 1 + 1/1! + 1/2! + \ldots + 1/n!.$$

Since

$$\lim_{n\to\infty}(x(S_n + 1/n!\cdot n) - x\cdot S_n) = 0\text{, then}$$
$$[x(S_n + 1/n!\cdot n)] - [x\cdot S_n)] = 0 \text{ for some } n\text{, that is}$$
$$[x(S_n + 1/n!\cdot n)] = [x\cdot S_n)] \text{ for some } n.$$

So, in order to find the value of $[e\cdot x]$, it is necessary to find the minimum n for which the previous equality is fulfilled and to put

$$[e\cdot x] = [x\cdot S_n].$$

The graph of the function $F(x_1, \ldots, x_n)$ [1-3] is the set $<x_1, \ldots, x_n, F(x_1, \ldots, x_n)>$. We show that if the graph G_f of the everywhere defined function $f(x_1, \ldots, x_n)$ is a recursively enumerable set [1-3], then the function f is recursive

Really, the graph G_f is a set of the form:

$$<f_1(t), \ldots, f_n(t), g(t)>,$$

where f_i, g are primitive recursive functions. Then the value of the function f at any point can be calculated using the following algorithm:

```
function f(x1, … , xn)
    begin
        i = 0
        while f1(i) ≠ x1 ∨ … ∨ fn(i) ≠ xn
            do i = i +1
        f = g(i)
    end,
```

hence, the function f is recursive.

Fundamental results of the theory of algorithms

Let $\mathfrak{I}$ be a system of partial functions of one argument. A partial function $F(x, y)$ of two variables is called universal for the family $\mathfrak{I}$ if the following conditions are satisfied:

1. For each fixed i the function $F(i, y)$ belongs to $\mathfrak{I}$;
2. For each function $f(y)$ from $\mathfrak{I}$ there exists a number i such that for all y equality $F(i, y) = f(y)$ is fulfilled.

It is known [1] that for the class of partially recursive functions of one argument there is a universal partially recursive function Kleene $\boldsymbol{K}(n, x)$.

If we mapp every natural number into the set of partially recursive functions $\boldsymbol{K}(n, x)$ then we get the numbering of the Kleene of partially recursive functions. The number n is called the Kleene number of the function $\boldsymbol{K}(n, x)$, which is also denoted by K_n.

Rice's theorem is true: the set A of Kleene numbers of functions belonging to a nonempty family of partially recursive functions of one argument which are differed from the set of all such functions, cannot be recursive. This theorem is used to substantiate the results on the algorithmic computability of functions.

For example, show that function

$$f(x) = \begin{cases} 1, \boldsymbol{K}(x,x)=1 \\ 0, \text{ in other cases} \end{cases}$$

is not algorithmically computational.

For this purpose the set $M = \{n_0, n_1, \dots\}$ of Kleene numbers of partially recursive functions $\boldsymbol{K}(n_0, x)$, $\boldsymbol{K}(n_1, x)$, … such that $\boldsymbol{K}(n_0, n_0) = 1$, $\boldsymbol{K}(n_1, n_1) = 1$, … is considered. By Rice's theorem, the set M is nonrecursive. Suppose that there exists an algorithm

```
function f(x)
    begin
      . . . . . .
        f = . . .
    end,
```

which calculates the function $f(x)$. Then the set M will be recursive. We receive a contradiction.

Algorithmic solvability. Consider problems for which there are no algorithms. Such problems will be called algorithmically unsolvable.

One of such algorithmically unsolvable problems is the stopping problem [1,3,5] which consists in the following: it is necessary to answer the question of whether there is an algorithm whose output for an arbitrary pair x, y of input natural numbers is 0, if $\boldsymbol{K}(x, y)$ is defined and 1 if $\boldsymbol{K}(x, y)$ is undefined. The existence of such an algorithm is equivalent to the existence of an algorithm whose output for an arbitrary pair x, y of input natural numbers is 0, if the algorithm for calculating partially recursive function f with Kleene number x stops at input y and 1, if the algorithm for calculating partially recursive function f with Kleene number x at input y works indefinitely. That is why this problem is called the stop problem.

Theorem 1. The stopping problem is algorithmically unsolvable.

Proof. It follows that the domain of the function $\boldsymbol{K}(x, y)$ is a nonrecursive recursively enumerable set.

Another algorithmically unsolvable problem is the belonging problem, which is as follows: it is necessary to answer the question about existence of an algorithm whose output for an arbitrary pair x, y of input natural numbers is 0 if $x \in \pi_y$ and 1 if $x \notin \pi_y$.

Theorem 2. The belonging problem is algorithmically unsolvable.

Proof. Really, the solvability of the affiliation problem means the recursiveness of the set of pairs $<x, y>$ for which the equation $\boldsymbol{K}(x, t) = y$ has a solution. But it is not so.

Universal numerical sets. The set U of pairs of numbers (i, x) such that $\boldsymbol{K}(i, x) = 1$ will be called the universal numerical set [4], that is

$$U = \{(i, x), \boldsymbol{K}(i, x) = 1\}.$$

Theorem 3. The problem of belonging to the universal numerical set is algorithmically unsolvable.

Proof. Really, the solvability of this problem means that the characteristic function

$$\chi_U(i, x) = \begin{cases} 1, & \boldsymbol{K}(i, x) = 1 \\ 0, & \boldsymbol{K}(i, x) \neq 1 \end{cases}$$

is recursive. Let's show that this is not so. Really, if this function is recursive, then the function $f(x)$, which is calculated by the algorithm

```
function f(x)
    begin
      i := 0
      while χU(x, x) = 1 do
            i := i + 1
      f := 1
    end,
```

is partially recursive functions, and, therefore, $f(x) = \boldsymbol{K}(n, x)$, for some n. Calculate this function for x equal to n.

If $\chi_U(n, n) = 1$, then it means, on the one hand, that $\boldsymbol{K}(n, n)$ is not defined ($f(n) = \boldsymbol{K}(n, n)$). On the other hand, since $\chi_U(n, n) = 1 \Leftrightarrow \boldsymbol{K}(n, n) = 1$, then $\boldsymbol{K}(n, n)$ is defined. If $\chi_U(n, n) \neq 1$, then it means, on the one hand, that $\boldsymbol{K}(n, n)$ is defined and $\boldsymbol{K}(n, n) = 1$. On the other hand $\boldsymbol{K}(n, n) \neq 1$.

Universal set.of words. Let $\Sigma = \{a_1, \dots, a_n\}$ be an alphabet. Each grammar above the alphabet Σ can be represented by a word in the alphabet $\Sigma \cup N'$, where $N' = \{ S, \rightarrow, ', * \}$. For example, grammar G with rules $\{S \rightarrow aABb|Bbb, Bb \rightarrow C, AC \rightarrow aac\}$ can be represented by a word

$$*A_1*A_2* \dots *A_4*,$$

where $A_1 = S \rightarrow a\,S'S''b$, $A_2 = S \rightarrow S''bb$, $A_3 = S''b \rightarrow S'''$, $A_4 = S'S''' \rightarrow aac$.

The set of all such words will be called the base.

Universal set.of words [4] we will be called the set V of words $*A_1*A_2* \dots *A_n*w$ such that the word w is infered in the grammar G with rules $A_1, A_2, \dots, A_n$, that is

$$V = \{*A_1*A_2* \dots *A_n*w, S \Rightarrow_G w\}.$$

It is clear that each grammar G generates recursively enumerable set (the set of all words that are infered in the grammar).

Therefore, a universal set.of words is a set of words $*A_1*A_2* \dots *A_n*w$ such that w belongs to the recursively enumerable set, which is generated by the grammar with rules $A_1, A_2, \dots, A_n$, that is

$$V = \{*A_1*A_2* \dots *A_n*w, w \in \text{recursively enumerable set, which is generated } G\}.$$

We can show that the theorem 4 holds.

Theorem 4. The problem of belonging to a universal set of words is algorithmically unsolvable.

The Post correspondence problem. Let's look at another example of an algorithmically unsolvable problem. It is called the Post correspondence problem [4].

Let

$$P = \{(v_1, w_1), (v_2, w_2), \dots, (v_n, w_n)\}$$

is a set of pairs of words in the alphabet Σ. It is said that a set of pairs of words have a solution if there exist such a sequence

$$(v_{i_1}, w_{i_1}),(v_{i_2}, w_{i_2}),\dots,(v_{i_k}, w_{i_k})$$

of pairs of words that

$$v_{i_1} v_{i_2} \dots v_{i_k} = w_{i_1} w_{i_2} \dots w_{i_k},$$

that is, the word formed by the left coordinates of the sequence of pairs coincides with the word formed by the right coordinates of the sequence of pairs.

Post correspondence problem is to answer the question of the existence of an algorithm whose output for an arbitrary input set P of word pairs is 0 if P has a solution and 1 if P has no solution.

Theorem 5. Post correspondence problem is algorithmically unsolvable.

Proof. By an arbitrary word $*A_1*A_2* \dots *A_n*w$ of universal set.of words we build Post correspondence problem according to the following rules:

– a pair $(FS\Rightarrow, F)$, where F is a special symbol, is added to the set of pairs of words;

– for each symbol a of the alphabet Σ we add pairs (a, a) to the set of pairs of words;

– for each nonterminal symbol S', S'', …, $S'^{\dots'}$ we add pairs (S', S'), (S'', S''), …, $(S'^{\dots'}, S'^{\dots'})$ to the set of pairs of words;

– for the word w we add a pair $(E, wE\Rightarrow)$ to the set of pairs of words, where E is a special symbol;

– for each rule $A_i = u \to v$ we add a pair (v, u) to the set of pairs of words;

– we add a pair $(\Rightarrow, \Rightarrow)$ to the set of pairs of words.

It can be shown that

$$*A_1*A_2* \dots *A_n*w \in V \Leftrightarrow \text{the set of pairs has a solution.}$$

Therefore, if the Post correspondences problem is algorithmically solvable, then the problem of belonging to a universal set.of words will also be algorithmically solvable. And this is not so.

Unsolvability in the class of context-free grammars (problem of ambiguity). This problem is to answer the question of the existence of an algorithm whose output for an arbitrary input context-free grammar G is 0 if G is ambiguous and 1 otherwise.

Let

$$Q = \{(v_1, w_1), (v_2, w_2), \dots, (v_n, w_n)\}$$

is a set of pairs of words in the alphabet Σ. For the set $A = \{v_1, v_2, \dots, v_n\}$ of the left components of the elements from Q we construct the context-free grammar G_A with one nonterminal A and a set of terminal symbols $\Sigma \cup I$, where $I = \{a_1, a_2, \dots, a_n\}$ and $\Sigma \cap I = \varnothing$. Alphabet I is called the alphabet of index symbols. The rules of grammar G_A have the following form:

$$A \to v_1Aa_1 \mid v_2Aa_2 \mid \ldots \mid v_nAa_n \mid v_1a_1 \mid v_2a_2 \mid \ldots \mid v_na_n.$$

This grammar generates language

$$L(G_A) = \{v_{i_1} \ldots v_{i_m} a_{i_m} \ldots a_{i_1},\ 1 \le i_1, \ldots, i_m \le n\}.$$

Similarly for the set $B = \{w_1, w_2, \ldots, w_n\}$ of right components from Q we build a context-free grammar G_B with one nonterminal B and a set of rules

$$B \to w_1Ba_1 \mid w_2Ba_2 \mid \ldots \mid w_nBa_n \mid w_1a_1 \mid w_2a_2 \mid \ldots \mid w_na_n.$$

This grammar generates language

$$L(G_B) = \{w_{i_1} \ldots w_{i_m} a_{i_m} \ldots a_{i_1},\ 1 \le i_1, \ldots, i_m \le n\}.$$

Let's form a grammar

$$G_{AB} = (\{S, A, B\}, \Sigma \cup \{a_1, a_2, \ldots, a_n\}, P(G_A) \cup P(G_B), S).$$

The following theorem holds.

Theorem 6. Grammar G_{AB} is ambiguous $\Leftrightarrow$ Q has a solution.

Proof. Let $i_1, \ldots, i_m$ is a solution of a set of pairs of words Q. Consider two inferences in grammar G_{AB}:

$$S \Rightarrow A \Rightarrow v_{i_1} Aa_{i_1} \Rightarrow v_{i_1} v_{i_2} Aa_{i_2} a_{i_1} \Rightarrow \ldots \Rightarrow v_{i_1} v_{i_2} \ldots v_{i_m} a_{i_m} \ldots a_{i_2} a_{i_1},$$
$$S \Rightarrow B \Rightarrow w_{i_1} Ba_{i_1} \Rightarrow w_{i_1} w_{i_2} Ba_{i_2} a_{i_1} \Rightarrow \ldots \Rightarrow w_{i_1} w_{i_2} \ldots w_{i_m} a_{i_m} \ldots a_{i_2} a_{i_1}.$$

Since $i_1, \ldots, i_m$ is a solution, then

$$v_{i_1} v_{i_2} \ldots v_{i_m} = w_{i_1} w_{i_2} \ldots w_{i_m},$$

that is, these two inferences generate the same word. Because these inferences are different, the grammar G_{AB} is ambiguous.

On the other hand, for a given terminal word, there is no more than one inference in the grammar G_A and no more than one inference in the grammar G_B. Therefore, a terminal word can have two different inference only if one of them begins with $S \Rightarrow A$ and continues with the inference in the grammar G_A, and the other begins with $S \Rightarrow B$ and continues with the inference in the grammar G_B.

This word ends with symbols $a_{i_m} \ldots a_{i_1}$ for some $m \ge 1$. Since

$$v_{i_1} v_{i_2} \ldots v_{i_m} = w_{i_1} w_{i_2} \ldots w_{i_m},$$

then $i_1, \ldots, i_m$ is a solution.

Theorem 7. The problem of ambiguity of context-free grammar is algorithmically unsolvable.

Proof. If the problem of ambiguity is solvable, then Post correspondence problem would be solvable. And this is not so.

Theorem 8. Problem

$$L(G_1) \cap L(G_2) = \varnothing$$

is algorithmically unsolvable.

Proof. Consider an arbitrary Post correspondence problem in the alphabet Σ and construct context-free grammars G_A and G_B. Since $L(G_A) \cap L(G_B)$ is a set of solutions of the Post correspondence problem, then

$$L(G_A) \cap L(G_B) = \varnothing \Leftrightarrow \text{Post correspondence problem has no solution.}$$

Therefore, if this problem is solvable for arbitrary G_1 and G_2, then Post correspondence problem will also be solvable. But this is not so.

Theorem 9. Problem

$$L(G_1) = L(G_2)$$

is algorithmically unsolvable.

Proof. Let's show that $\overline{L(G_A)}$ is a context-free grammar. Really, there exists an algorithm which for any input word in the alphabet $\Sigma \cup I$ gives an answer to the question of the word's belonging to the language $L(G_A)$. It is based on the fact that such a word must end by index symbols $a_{i_m} \dots a_{i_1}$, nd the prefix of this word must be a word $v_{i_1} \dots v_{i_m}$. If this is not so, then the input word does not belong to $L(G_A)$. Therefore, there exists an algorithm which for an arbitrary input word in the alphabet $\Sigma \cup I$ gives an answer to the question about belonging this word to the language $\overline{L(G_A)}$. Thus, this language is recursively enumerable set, that is, generated by context-free grammar.

Consider an arbitrary Post correspondence problem in the alphabet Σ and construct context-free grammars G_1 and G_2, which generate languages $\overline{L_A} \cup \overline{L_B}$ and $(\Sigma \cup I)^*$, respectively, where I is still the alphabet of index symbols. The equality

$$\overline{L_A} \cup \overline{L_B} = \overline{L_A \cap L_B}$$

is holds

Therefore, in $L(G_1)$ there are no words which are solutions of Post correspondence problem. Language $L(G_2)$ contains all words from $(\Sigma \cup I)^*$. So,

$L(G_1)$ and $L(G_2)$ coincide $\Leftrightarrow$ when the Post correspondence problem has no solution.

It follows that if the equivalence problem for context-free languages is algorithmically solvable, then Post correspondence problem will be solvable. And this is not so.

Theorem 10. Problem $L(G_1) \subseteq L(G_2)$ is algorithmically unsolvable.

Proof. Let G_1 be a context-free grammar, which generates language $(\Sigma \cup I)^*$ and G_2 be a context-free grammar, which generates language $\overline{L_A} \cup \overline{L_B}$, where I is an alphabet of index symbols. Then

$$L(G_1) \subseteq L(G_2) \Leftrightarrow \overline{L_A} \cup \overline{L_B} = (\Sigma \cup I)^*.$$

That is, $L(G_1) \subseteq L(G_2) \Leftrightarrow$ when the Post correspondence problem has no solution. It follows that if the inclusion problem for context-free languages is algorithmically solvable then the Post correspondence problem will be solvable. But this is not so.

Conclusions. Thus, an approach whith help of which the belonging of functions to classes of algorithmically calculated can be argued by constructing appropriate algorithms, unlike to algebraic terms, are proposed. It can be useful for programmers of different ages who want to learn the basics of computational theory and the theory of algorithms, as well as students of relevant specialties. It should also be noted that the proposed approach based on the concretization of Church's thesis allows to achieve a clearer formulation of various problems and processes for their solution. This is achieved primarily through the so-called PseudoPascal macro operators for the implementation of the simplest mechanical operations.

References

1. *A.I. Maltsev.* Algorithms and recursive functions. – Science: Moscow, 1965. – 390 p.
2. *V.A. Uspenskij, A.Л. Semenov.* Algorithm theory: basic diyscoveries and applications - Science: Moscow, 1987. – 288 p.
3. *O.I. Provotar.* Concret algoritmics. - Naukova Dumka: Kyiv. 2017. – 168 p.
4. *D. Hopcroft, R. Motwani, D. Ullman.* Introduction to the theory of automata, languages and calculations. - M: Williams, 2002. – 528 p.
5. *M.S. Nikitchenko, S.S Shkilniak*. Mathematical logic and theory of algorithms. - K: VPC "Kyiv University", 2008. – 528 p.

About the authors:

Provotar Oleksandr Ivanovych Doctor of Physical and Mathematical Sciences, Professor, Professor of Taras Shevchenko National University of Kyiv, number of publications in domestic publications – 120, in foreign – 30, h index 4, ORCID number 0000-0002-6556-3264, t. +38-050-444-17-35, email: aprowata1@bigmir.net.

Provotar Olga Oleksandrivna, junior researcher at the V.M. Glushkov Cybernetics Institute NAS of Ukraine, number of publications in domestic publications – 10, in foreign – 2, ORCID number 0000-0002-6591-3615, email: provotar@huspi.com.

Place of work of the authors:

Taras Shevchenko National University of Kyiv, 03187, Kyiv -187, Academician Glushkov Avenue, 2, b. 6. T.: (044) 259 0511. Fax: (044) 259 7044. E-mail: aprowata1@bigmir.net.

V.M. GlushkovI Cybernetics Institute NAS of Ukraine, 03680, МСП, Kyiv -187, Academician Glushkov Avenue, 40. T.: (044) 259 0511. Fax: (044) 259 7044. E-mail: provotar@huspi.com.

+38-050-444-17-35, email: aprowata1@bigmir.net.