УДК 681.3 https://doi.org/10.15407/pp2020.02-03.198

# ЕЛЕМЕНТИ КОНКРЕТНОЇ АЛГОРИТМІКИ: ОБЧИСЛЮВАНІСТЬ І РОЗВ'ЯЗНІСТЬ

*О.І. Провотар, О.О. Провотар*

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.

Ключові слова: теза Чорча, розв'язність, універсальна функція.

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 Чорча для более узких классов алгоритмически вычислительных функций. С помощью такого подхода принадлежность функций к классам алгоритмически вычислимых аргументируется построением соответствующих алгоритмов.

Ключевые слова: тезис Чорча, разрешимость, универсальная функция.

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.

Key words: Church thesis, solvability, universal function.

## Вступ

Всім відома теза Чорча [1–3] про те, що клас алгоритмічно обчислюваних функцій співпадає з класом частково рекурсивних функцій. Клас частково рекурсивних функцій визначається математично точно. Тому, тезу можна використовувати як для доведення алгоритмічної обчислюваності функцій, так і для доведення того, що функція не є алгоритмічно обчислюваною. Для цього треба лише показати, що така функція належить до класу частково рекурсивних функцій і в цьому випадку вона є алгоритмічно обчислюваною, або не належить і, відповідно, не є алгоритмічно обчислюваною.

Показати алгоритмічну обчислюваність функцій шляхом її належності до класу частково рекурсивних функцій досить складно, окрім найпростіших функцій. Крім того, в кожному конкретному випадку це потребує побудови математичної моделі функції у вигляді терма із обчислюваних операцій над базовими функціями. Базовими функціями, як відомо [1, 2] називаються найпростіші функції $o(x) = 0$, $s(x) = x + 1$ та функції-селектори $I_m^n(x_1, \dots, x_n) = x_m$, де $n \geq m \geq 1$. Основними обчислюваними операціями будуть операції *суперпозиції* $\boldsymbol{S}^{n+1}$, *примітивної рекурсії* $\boldsymbol{R}$ та *мінімізації* $\boldsymbol{M}$.

Наприклад, операція суперпозиції $\boldsymbol{S}^{n+1}$ дозволяє із $n$-арної функції $g(x_1, \dots, x_n)$ та $n$ функцій $g_1(x_1, \dots, x_m)$, ... , $g_n(x_1, \dots, x_m)$, однакової арності утворити функцію

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

яку позначають термом $\boldsymbol{S}^{n+1}(g, g_1, \dots, g_n)$.

Виникає цілком закономірне питання про можливість використання конкретних алгоритмів для доведення фундаментальних результатів теорії рекурсивних функцій без побудови відповідних обчислювальних термів. Або, іншими словами, розповісти мовою, зрозумілою для програмістів, про основні результати теорії алгоритмів (рекурсивних функцій), зокрема про проблеми обчислюваності та розв'язності. Отже, мета роботи полягає у тому, щоб запропонувати підходи і засоби для вирішення поставленої проблеми. Для цього пропонується точно описати основні конструкції алгоритму і переформулювати (конкретизувати) тезу Чорча для більш вузьких класів алгоритмічно обчислюваних функцій.

## Конкретизація поняття алгоритму

Далі алгоритми будемо визначати як синтаксично правильні конструкції в мові ПсевдоPascal, яка є спрощеним діалектом мови Pascal. Операторами цієї мови будуть наступні:

<ідентифікатор> = = <слово>,

<оператор> = = <ідентифікатор> = <арифметичний вираз>

<оператор> = = **if** <відношення> **then** <оператор> **else** <оператор>

<оператор> = = **while** < відношення > **do** {<оператор>, … , <оператор>}

<оператор> = = **for** < відношення > **to** < відношення > {<оператор>, … , <оператор>}

Конкретизуємо тезу Чорча для класів ПРФ, РФ та ЧРФ відповідно.

1. Клас функцій, що обчислюються всюди визначеними алгоритмами без використання оператора **while … do**. Цей клас описується наступним чином:

- вважатимемо, що найпростіші функції

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

обчислюються всюди визначеними алгоритмами без використання оператора **while … do**, а, отже належать до цього класу;

- всі функції, які обчислюються всюди визначеними алгоритмами без використання оператора **while … do** і при їх обчисленні використовуються допоміжні функції, які обчислюються всюди визначеними алгоритмами без використання оператора **while … do** теж відносимо до цього класу. Таким чином, справедлива наступна.

**Теза 1.** Клас функцій, що обчислюються всюди визначеними алгоритмами без використання оператора **while … do** співпадає з класом *примітивно рекурсивних функцій*.

2. Клас функцій, що обчислюються всюди визначеними алгоритмами. Цей клас описується наступним чином:

- всі ПРФ належать до цього класу;
- всі функції, які обчислюються всюди визначеними алгоритмами і при їх обчисленні використовуються функції, які обчислюються всюди визначеними алгоритмами теж відносимо до цього класу. Таким чином, справедлива наступна.

**Теза 2.** Клас функцій, що обчислюються всюди визначеними алгоритмами співпадає з класом *рекурсивних функцій*.

3. Клас функцій, що обчислюються довільними алгоритмами. Цей клас описується наступним чином:

- всі РФ належать до цього класу;
- всі функції, які обчислюються довільними алгоритмами і при їх обчисленні використовуються функції, які обчислюються довільними алгоритмами теж відносимо до цього класу. Таким чином, справедлива наступна.

**Теза 3.** Клас функцій, що обчислюються довільними алгоритмами співпадає з класом *частково рекурсивних функцій*.

## Обчислюваність

Покажемо, як доводити алгоритмічну обчислюваність функцій, використовуючи запропоновані вище тези. Нехай, треба довести, що функція

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

ПР функція.

Розглянемо послідовність

$$1y \dot{-} x,\ 2y \dot{-} x,\ \dots,\ [x/y] \dot{-} x,\ \dots,\ xy \dot{-} x.$$

Оскільки частка $[x/y]$ означає скільки разів число $y$ "поміщається" в числі $x$, то $[x/y]$ дорівнює числу нулів в цій послідовності. Дійсно, якщо, наприклад, $y$ два рази "поміщається" в числі $x$, то

$$1y \dot{-} x = 0,\ 2y \dot{-} x = 0, \text{ а } 3y \dot{-} x \neq 0.$$

Тому алгоритм обчислення функції наступний:

```
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.
```

Отже, функція $f(x, y)$ є ПРФ.

Якщо всі пари натуральних чисел розташувати в послідовність

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

тобто, впорядкувати всі пари так, що пара $<x, y>$ йде раніше за пару $<u, v>$ якщо

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

або якщо

$$x + y = u + v \text{ і } x < u,$$

то співвідношенням

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

де $n$ – номер пари в цій послідовності, задається бієкція між множиною пар та множиною натуральних чисел.

Така бієкція називається нумерацією Кантора пар чисел і позначається $\boldsymbol{c}(x, y)$, тобто $\boldsymbol{c}(x, y)$ – це номер пари $<x, y>$ в послідовності Кантора.

Лівий та правий елементи пари $<x, y>$ з номером $n$ визначають функції $\boldsymbol{l}(n)$ і $\boldsymbol{r}(n)$, які називаються лівою та правою координатними функціями.

Покажемо, що функції $\boldsymbol{c}(x, y)$, $\boldsymbol{l}(n)$, $\boldsymbol{r}(n)$ – ПР функції. Дійсно, функція $\boldsymbol{c}(x, y)$ обчислюється наступним алгоритмом:

```
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.
```

Враховуючи, що $\boldsymbol{l}(n) \le n$, $\boldsymbol{r}(n) \le n$ функція $\boldsymbol{l}(n)$ обчислюється алгоритмом:

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

а функція $\boldsymbol{r}(n)$ обчислюється алгоритмом:

```
function r(n)
    begin
      for i = 0 to n
      for j = 0 to n
```

if $\boldsymbol{c}(i, j) = n$ then $\boldsymbol{r} = j$

end.

Таким чином, $\boldsymbol{c}(x, y)$, $\boldsymbol{l}(n)$ та $\boldsymbol{r}(n)$ – ПР функції.

Покажемо, що функція

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

є ПРФ.

Дійсно, справедлива нерівність

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

де

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

Оскільки

$$\lim_{n\to\infty}(x(S_n + 1/n! \cdot n) - x \cdot S_n) = 0\text{, то}$$
$$[x(S_n + 1/n! \cdot n)] - [x \cdot S_n)] = 0 \text{ для деякого } n\text{, тобто}$$
$$[x(S_n + 1/n! \cdot n)] = [x \cdot S_n)] \text{ для деякого } n.$$

Отже, для того, щоб знайти значення $[e \cdot x]$, треба знайти мінімальне $n$, для якого виконується попередня рівність і покласти

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

Графіком функції $F(x_1, \ldots, x_n)$ [1–3] називається сукупність $(n + 1)$-ок виду $<x_1, \ldots, x_n, F(x_1, \ldots, x_n)>$. Покажемо, що якщо графік $G_f$ всюди визначеної функції $f(x_1, \ldots, x_n)$ є рекурсивно перелічимою множиною [1–3], то функція $f$ рекурсивна.

Дійсно, графік $G_f$ – це сукупність $(n + 1)$-ок виду:

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

де $f_i$, $g$ – ПРФ. Тоді значення функції $f$ в довільній точці можна обчислити за допомогою наступного алгоритму:

function $f(x_1, \ldots, x_n)$
  begin
    $i = 0$
    while $f_1(i) \neq x_1 \vee \ldots \vee f_n(i) \neq x_n$
      do $i = i + 1$
    $f = g(i)$
  end,

отже, функція $f$ рекурсивна.

## Фундаментальні результати теорії алгоритмів

Нехай $\mathfrak{I}$ – система часткових одномісних функцій. Часткова функція $F(x, y)$ від двох змінних називається універсальною для сімейства $\mathfrak{I}$, якщо виконуються наступні умови:

- для кожного фіксованого $i$ функція $F(i, y)$ належить $\mathfrak{I}$;
- для кожної функції $f(y)$ із $\mathfrak{I}$ існує таке число $i$, що для всіх $y$ $F(i, y) = f(y)$.

Відомо [ ], що для класу одномісних ЧРФ існує універсальна ЧРФ Кліні $\boldsymbol{K}(n, x)$. Відображення, яке кожному натуральному числу $n$ ставить у відповідність одномісну ЧРФ $\boldsymbol{K}(n, x)$ називати нумерацією Кліні одномісних ЧРФ. Число $n$ називається клінівським номером функції $\boldsymbol{K}(n, x)$, яка також позначається $K_n$.

Справдлива також теорема Райса про те, що множина $A$ клінівських номерів функцій, що належать до непустого сімейство одномісних ЧРФ, відмінних від сукупності всіх таких функцій, не може бути рекурсивною. Ця теорема використовується для обгрунтування результатів про алгоритмічну обчислюваність функцій.

Наприклад, покажемо, що функція

$$f(x) = \begin{cases} 1, & \boldsymbol{K}(x, x) = 1 \\ 0, & \text{в інших випадках} \end{cases}$$

не є алгоритмічно обчислювальною.

Для цього розглядається множина $M = \{n_0, n_1, \dots\}$ клінівських номерів ЧРФ $\boldsymbol{K}(n_0, x)$, $\boldsymbol{K}(n_1, x)$, … таких, що $\boldsymbol{K}(n_0, n_0) = 1$, $\boldsymbol{K}(n_1, n_1) = 1$, … . За теоремою Райса множина $M$ є нерекурсивною. Припустимо, що існує алгоритм

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

який обчислює функцію $f(x)$. Тоді множина $M$ буде рекурсивною. А це не так.

**Алгоритмічна розв'язність.** Розглянемо проблеми, для розв'язання яких не існує алгоритмів. Такі проблеми будемо називати алгоритмічно нерозв'язними.

Однією з таких алгоритмічно нерозв'язних проблем є *проблема зупинки* [1, 3, 5] яка полягає у наступному: треба дати відповідь на питання про те, чи існує алгоритм, виходом якого для довільної пари $x, y$ вхідних натуральних чисел є 0, якщо $\boldsymbol{K}(x, y)$ визначена і 1, якщо $\boldsymbol{K}(x, y)$ невизначена. Існування такого алгоритму еквівалентне існуванню алгоритму виходом якого для довільної пари $x, y$ вхідних натуральних чисел є 0, якщо алгоритм обчислення ЧРФ $f$ з клінівським номером $x$ зупиняється на вході $y$ і 1, якщо алгоритм обчислення ЧРФ $f$ з клінівським номером $x$ на вході $y$ працює нескінченно довго. Саме тому ця проблема називається проблемою зупинки.

**Теорема 1.** Проблема зупинки алгоритмічно нерозв'язна.

*Доведення*. Випливає з того, що область визначення функції $\boldsymbol{K}(x, y)$ є нерекурсивною РПМ.

Іншою алгоритмічно нерозв'язною проблемою є *проблема належності* яка полягає в наступному: треба дати відповідь на питання про те, чи існує алгоритм, виходом якого для довільної пари $x, y$ вхідних натуральних чисел є 0, якщо $x \in \pi_y$ і 1, якщо $x \notin \pi_y$.

**Теорема 2.** Проблема належності є алгоритмічно нерозв'язною.

*Доведення*. Дійсно, розв'язність проблеми належності означає рекурсивність множини пар $\langle x, y \rangle$ для яких рівняння $\boldsymbol{K}(x, t) = y$ має розв'язок. А це не так.

**Універсальні числові множини.** *Універсальною числовою множиною* [4] будемо називати множину $U$ пар чисел $(i, x)$ таких, що $\boldsymbol{K}(i, x) = 1$, тобто

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

**Теорема 3.** Проблема належності до універсальної числової множини є алгоритмічно нерозв'язною.

*Доведення*. Дійсно, розв'язність цієї проблеми означає, що характеристична функція

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

є рекурсивною. Покажемо, що це не так. Дійсно, якби ця функція була рекурсивною, то функція $f(x)$, яка обчислюється алгоритмом

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

буде ЧРФ, а, отже, $f(x) = \boldsymbol{K}(n, x)$, для деякого $n$. Обчислимо цю функцію для $x = n$.

Якщо $\chi_U(n, n) = 1$, то це означає, з одного боку, що $\boldsymbol{K}(n, n)$ не визначена ($f(n) = \boldsymbol{K}(n, n)$). З іншого боку, оскільки $\chi_U(n, n) = 1 \Leftrightarrow \boldsymbol{K}(n, n) = 1$, то $\boldsymbol{K}(n, n)$ – визначена.

Якщо $\chi_U(n, n) \neq 1$, то це означає, з одного боку, що $\boldsymbol{K}(n, n)$ визначена і $\boldsymbol{K}(n, n) = 1$. З іншого боку $\boldsymbol{K}(n, n) \neq 1$.

**Універсальні словарні множини**. Нехай $\Sigma = \{a_1, \ldots, a_n\}$ – алфавіт. Кожну граматику над алфавітом $\Sigma$ можна подати словом в алфавіті $\Sigma \cup N'$, де $N' = \{ S, \rightarrow, ', * \}$. Наприклад, граматику $G$ з продукціями $\{S \rightarrow aABb|Bbb, Bb \rightarrow C, AC \rightarrow aac\}$ можна подати словом

$$*A_1*A_2* \ldots *A_4*,$$

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

Множину всих таких слів будемо називати базою.

*Універсальною словарною множиною* [4] будемо називати множину $V$ слів $*A_1*A_2* \ldots *A_n*w$ таких, що слово $w$ виводиться в граматиці $G$ з продукціями $A_1, A_2, \ldots, A_n$, тобто

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

Зрозуміло, що кожна граматика $G$ породжує РПМ – множину всих слів, які виводяться граматиці. Тому, універсальна словарна множина – це множина слів $*A_1*A_2* \ldots *A_n*w$ таких, що $w$ належить РПМ, яка породжується граматикою з продукціями $A_1, A_2, \ldots, A_n$, тобто

$$V = \{*A_1*A_2* \ldots *A_n*w, w \in \text{РПМ, яка породжується } G\}.$$

Занумеруємо символи алфавіту $\Sigma \cup N'$, числами 1, 2, …, $p = n+4$. Номером пустого слова $\varepsilon$ є число 0. Номером довільного слова $b_{i_m} \ldots b_{i_1} b_{i_0}$ в алфавіті $\Sigma \cup N'$, називається число

$$\wp(w) = i_0 + i_1 p + \ldots + i_m p^m.$$

Символом $\alpha(n)$ будемо позначати слово з номером $n$. Так як при фіксованому $p$ кожне додатнє число $n$ можна подати одним і лише одним способом у вигляді

$$n = i_0 + i_1 p + \ldots + i_m p^m \quad (1 \leq i_k \leq m),$$

то кожне число є номером одного і тільки одного слова множини $(\Sigma \cup N')^*$.

Довільну часткову функцію $F:(\Sigma \cup N')^* \rightarrow (\Sigma \cup N')^*$ будемо називати частковою словарною функцією в алфавіті $\Sigma \cup N'$. Часткова числова функція $f:N \rightarrow N$ називається функцією, що представляє словарну функцію $F$ в нумерації $\alpha$, якщо

$$F(\alpha(x)) = \alpha(f(x)).$$

Зрозуміло, що для кожної часткової словарної функції існує єдина часткова числова функція, яка представляє словарну функцію і кожна часткова числова функція представляє єдину словарну функцію.

Часткова словарна функція $F$ називається примітивно рекурсивною, рекурсивною або частково рекурсивною, якщо такою є часткова числова функція $f$, яка її представляє. Те ж саме стосується і ПРМ, РМ, РПМ.

Так як множина $V$ є РПМ, то існує алгоритм, який по довільному вхідному слову $*A_1*A_2* \ldots *A_n*w \in V$ видає 1, якщо слово $w$ породжується граматикою з продукціями з бази $*A_1*A_2* \ldots *A_n*$. Тоді буде існувати

алгоритм, який по номеру $\wp(*A_1*A_2* \dots *A_n*w)$ видає 1, якщо слово $w$ породжується граматикою з продукціями з бази $*A_1*A_2* \dots *A_n*$. Цей алгоритм обчислює ЧРФ $g(x)$. Припустимо, що ця функція рекурсивна, тобто

$$g(x) = \begin{cases} 1, \ \alpha(x) \in V \\ 0, \ \alpha(x) \notin V \end{cases}.$$

Оскільки множина слів $V$ є РПМ, то існує граматика з продукціями $B_1$, $B_2$, … , $B_k$, яка породжує цю множину. Утворимо слово

$$*B_1*B_2* \dots *B_k**A_1*A_2* \dots *A_n*w$$

і знайдемо значення функції $g(n)$, де $n = \wp(*B_1*B_2* \dots *B_k**A_1*A_2* \dots *A_n*w)$.

Якщо $g(n) = 1$, то $\alpha(n) \in V$. Але це слово відмінне від кожного слова з $V$, а, отже, $\alpha(n) \notin V$.

Якщо $g(n) = 0$, то $\alpha(n) \notin V$. Але слово $*A_1*A_2*\dots*A_n*w$ породжується граматикою з продукціями $B_1$, $B_2$, …, $B_k$, а, отже, $\alpha(n) \in V$.

В обох випадках отримуємо суперечність, а тому функція $g(x)$ не може бути рекурсивною.

Тому, справедлива теорема.

**Теорема 4.** Проблема належності до універсальної словарної множини є алгоритмічно нерозв'язною.

*Доведення*. Якби ця проблема була алгоритмічно розв'язною, то функція $g(x)$ була б рекурсивною, а це не так.

**Проблема відповідностей Поста**. Розглянемо ще один приклад алгоритмічно нерозв'язної проблеми на словах. Вона називається *проблемою відповідностей Поста* (ПВП) [4].

Нехай

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

множина пар слів в алфавіті $\Sigma$. Говорять, що множина пар слів має розв'язок, якщо існує така послідовність

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

пар слів, що

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

тобто слово, утворене лівими координатами послідовності пар співпадає зі словом, утвореним правими координатами послідовності пар.

ПВП полягає в тому, щоб дати відповідь на питання про існування алгоритму, виходом якого для довільної вхідної множини $P$ пар слів є 0, якщо $P$ має розвязок і 1, якщо $P$ не має розв'язку.

**Теорема 5.** Проблема відповідностей Поста алгоритмічно нерозв'язна.

*Доведення.* За довільною словом $*A_1*A_2* \dots *A_n*w$ універсальної словарної множини будуємо ПВП згідно наступних правил:

- пару $(FS\Rightarrow, F)$, де $F$ – спеціальний символ, додаємо до множини пар слів;
- для кожного символа $a$ алфавіту $\Sigma$ до множини пар слів додаємо пари $(a, a)$;
- для кожного не термінального символа $S'$, $S''$, …, $S'^{\cdots\prime}$ до множини пар слів додаємо пари $(S', S')$, $(S'', S'')$, …, $(S'^{\cdots\prime}, S'^{\cdots\prime})$;
- для слова $w$ до множини пар слів додаємо пару $(E, wE\Rightarrow)$, де $E$ – спеціальний символ;
- для кожної продукції $A_i = u \to v$ до множини пар слів додаємо пару $(v, u)$;
- до множини пар слів додаємо пару $(\Rightarrow, \Rightarrow)$.

Можна показати, що справедливе співвідношення:

$$*A_1*A_2* \dots *A_n*w \in V \Leftrightarrow \text{множина пар має розв'язок.}$$

Тому, якщо проблема відповідностей Поста алгоритмічно розв'язна, то алгоритмічно розв'язною буде і проблема належності до універсальної словарної множини. А це не так.

**Нерозв'язність в класі КВ-граматик (проблема неоднозначності).** Ця проблема полягає у тому, щоб дати відповідь на питання про існування алгоритму, виходом якого для довільної вхідної КВ-граматики $G$ є 0, якщо $G$ неоднозначна і 1 в іншому випадку.

Нехай

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

множина пар слів в алфавіті $\Sigma$. Для множини $A = \{v_1, v_2, \dots, v_n\}$ лівих компонент елементів з $Q$ будуємо КВ-граматику $G_A$ з одним не терміналом $A$ та множиною термінальних символів $\Sigma \cup I$, де $I = \{a_1, a_2, \dots, a_n\}$, причому $\Sigma \cap I = \varnothing$. Алфавіт $I$ називається алфавітом індексних символів. Продукції граматики $G_A$ мають наступний вигляд:

$$A \to v_1Aa_1 | \; v_2Aa_2 | \; \dots \; | \; v_nAa_n | \; v_1a_1 | \; v_2a_2 | \; \dots \; | \; v_na_n.$$

Ця граматика породжує мову

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

Аналогічно для множини $B = \{w_1, w_2, \dots, w_n\}$ правих компонент з $Q$ будуємо КВ-граматику $G_B$ з одним не терміналом $B$ та множиною продукцій

$$B \to w_1Ba_1 | \; w_2Ba_2 | \; \dots \; | \; w_nBa_n | \; w_1a_1 | \; w_2a_2 | \; \dots \; | \; w_na_n.$$

Ця граматика породжує мову

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

Утворимо граматику

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

Справедлива наступна

**Теорема 6.** Граматика $G_{AB}$ неоднозначна $\Leftrightarrow$ $Q$ має розв'язок.

*Доведення* (достатність). Нехай $i_1, \dots, i_m$ – розв'язок множини пар слів $Q$. Розглянемо два виведення в граматиці $G_{AB}$:

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

Так як $i_1, \dots, i_m$ – розв'язок, то

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

тобто ці два виведення породжують одне і теж слово. Оскільки ці виведення різні, то граматика $G_{AB}$ неоднозначна.

(Необхідність). Для даного термінального слова існує не більше одного виведення в граматиці $G_A$ і не більше одного виведення в граматиці $G_B$. Тому, термінальне слово може мати два різні виведення тільки в тому випадку, якщо одне з них починається з $S \Rightarrow A$ і продовжується виведенням в граматиці $G_A$, а інше починається з $S \Rightarrow B$ і продовжується виведенням в граматиці $G_B$.

Таке слово має закінчення і символів $a_{i_m} \dots a_{i_1}$ для деякого $m \ge 1$. Так як

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

то $i_1, \dots, i_m$ – розв'язок.

**Теорема 7.** Проблема неоднозначності КВ-граматик алгоритмічно нерозв'язна.

*Доведення*. Якби проблема неоднозначності була б розв'язною, то була б розв'язною і ПВП. А це не так.

**Теорема 8.** Проблема

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

алгоритмічно нерозв'язна.

*Доведення*. Розглянемо довільну ПВП в афавіті $\Sigma$ і побудуємо КВ-граматики $G_A$ та $G_B$. Так як $L(G_A) \cap L(G_B)$ – множина розв'язків ПВП, то

$$L(G_A) \cap L(G_B) = \varnothing \Leftrightarrow \text{ПВП не має розв'язку.}$$

Отже, якби ця проблема була розв'язною для довільних $G_1$ та $G_2$, то була б розв'язною і ПВП. А це не так.

**Теорема 9.** Проблема

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

алгоритмічно нерозв'язна.

*Доведення*. Покажемо, що $\overline{L(G_A)}$ – КВ-мова. Дійсно, існує алгоритм, який за довільним вхідним словом в алфавіті $\Sigma \cup I$ дає відповідь на питання про належність цього слова до мови $L(G_A)$. Він грунтується на тому, що таке слово повинно завершувать множиною індексних символів $a_{i_m} \ldots a_{i_1}$, причому префікс цього слова повинен бути словом $\nu_{i_1} \ldots \nu_{i_m}$. Якщо це не так, то вхідне слово не належить до $L(G_A)$. Тому існує алгоритм, який за довільним вхідним словом в алфавіті $\Sigma \cup I$ дає відповідь на питання про належність цього слова до мови $\overline{L(G_A)}$. Отже, ця мова є РПМ, тобто, породжується КВ-граматикою.

Розглянемо довільну ПВП в афавіті $\Sigma$ і побудуємо КВ-граматики $G_1$ та $G_2$, які породжують мови $\overline{L_A} \cup \overline{L_B}$ та $(\Sigma \cup I)^*$ відповідно, де $I$, як і раніше – алфавіт індексних символів. Але справедливе співвідношення

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

Тому в $L(G_1)$ відсутні слова, які є розв'язками ПВП. Мова $L(G_2)$ містить всі слова із $(\Sigma \cup I)^*$. Тому,

$$L(G_1) \text{ і } L(G_2) \text{ співпадають} \Leftrightarrow \text{коли ПВП не має розв'язку.}$$

Звідси випливає, що якби проблема еквівалентності для КВ-мов була б алгоритмічно розв'язною, то була б розв'язною і ПВП. А це не так.

**Теорема 10.** Проблема $L(G_1) \subseteq L(G_2)$ – алгоритмічно нерозв'язна.

*Доведення.* Нехай $G_1$ – КВ-граматика, яка породжує мову $(\Sigma \cup I)^*$ і $G_2$ – КВ-граматика, яка породжує мову $\overline{L_A} \cup \overline{L_B}$, де $I$ – алфавіт індексних символів. Тоді справедливе співвідношення

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

Тобто, $L(G_1) \subseteq L(G_2) \Leftrightarrow$ коли ПВП не має розв'язку. Звідси випливає, що якби проблема включення для КВ-мов була б алгоритмічно розв'язною, то була б алгоритмічно розв'язною і ПВП. А це не так.

## Висновки

Таким чином, в статті пропонується підхід, за допомогою якого належність функцій до класів алгоритмічно обчислюваних можна аргументувати побудовою відповідних алгоритмів, на відміну від алгебраїчних термів. Він може бути корисним для програмістів різних вікових категорій, які хочуть познайомитися з основами теорії обчислюваності та теорії алгоритмів, а також студентам відповідних спеціальностей.

Слід також зазначити, що запропонований підхід на основі конкретизації тези Чорча дозволяє досягти більш зрозумілого формулювання різних задач та процесів їх розв'язання. Це досягається в першу чергу за рахунок так званих макрооператорів мови ПсевдоPascal для реалізації найпростіших механічних операцій.

## Література


1. Мальцев А.И. Алгоритмы и рекурсивные функции. М.: Наука. 1965. 390 с.
2. Успенский В.А., Семенов А.Л. Теория алгоритмов: основные открытия и приложения. М.: Наука. 1987. 288 с.
3. Провотар О.І. Конкретна алгоритміка. Київ. Наукова думка. 2017. 168 с.
4. Хопкрофт Д., Мотвани Р., Ульман Д. Введение в теорию автоматов, языков и вычислений. М: "Вильямс", 2002. 528 с.
5. Нікітченко М.С., Шкільняк С.С. Математична логіка та теорія алгоритмів. К.: ВПС "Київський університет", 2008. 528 с.


## References


1. Maltsev A.I. Algorithms and recursive functions. M.: Science. 1965.390 p.
2. Uspensky V.A., Semenov A.L. Algorithm theory: basic discoveries and applications. M.: Science. 1987. 288 p.
3. Provotar O.I. The algorithm is specific. Kiev. Naukova dumka. 2017.168 p.
4. Hopcroft D., Motvani R., Ullman D. Introduction to the theory of automata, languages and computations. M: "Williams", 2002. 528 p.
5. Nikitchenko M.S., Shkilnyak S.S. Mathematical logic and theory of algorithms. K.: VPS "Kiev University", 2008. 528 p.


Одержано 10.03.2020


***Про авторів***:

*Провотар Олександр Іванович,*
доктор фізико-математичних наук, професор, професор.
Кількість наукових публікацій в українських виданнях – 100.
Кількість наукових публікацій в зарубіжних виданнях – 30.
Індекс Гірша – 4.
http://orcid.org/0000-0002-6556-3264, м.т. +38-050-444-17-35, email: aprowata1@bigmir.net.

*Провотар Ольга Олександрівна*,
кандидат фізико-математичних наук,
молодший науковий співробітник.
Кількість наукових публікацій в українських виданнях – 10.
Кількість наукових публікацій в зарубіжних виданнях – 1.
http://orcid.org/0000-0002-6591-3615, email: provotar@huspi.com.

***Місце роботи авторів***:

Київський національний університет імені Тараса Шевченка,
03187, Київ-187, Проспект Академіка Глушкова, 2, к. 6.
Тел.: (044) 259 0511.
Факс: (044) 259 7044.
E-mail: aprowata1@bigmir.net.

Інститут кібернетики імені В.М. Глушкова НАН України,
03680, МСП, Київ-187, проспект Академіка Глушкова, 40.
Тел.: (044) 259 0511.
Факс: (044) 259 7044.
E-mail: provotar@huspi.com.