О.І. Провотар

КОНКРЕТНА АЛГОРИТМІКА

Київ 2017

Зміст

Передмова

В основу цієї книги покладено курс лекцій з теорії алгоритмів, який автор читає в Київському національному університеті імені Тараса Шевченка починаючи з 2003 року.

Ідея написання такої книги виникла з кількох причин. Однією з них є несприйняття матеріалу більшою частиною студентів при поданні його за допомогою тих чи інших математичних формалізмів, зокрема в рамках теорії рекурсивних функцій. Другою причиною була орієнтація матеріалу на студентів спеціальності «Програмна інженерія». Ця категорія студентів потребувала специфічного підходу до викладення матеріалу, який повинен, по можливості, базуватись на програмістських конструкціях.

Тому виникла необхідність у такому поданні матеріалу, аби задовольнити таку специфічну слухацьку аудиторію. Як цього вдалося досягти ?

Всім відома теза Чьорча 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 так званими базовими (найпростішими) функціями.

Виникає цілком закономірне питання про можливість використання конкретних алгоритмів для доведення, в тому числі, фундаментальних результатів теорії рекурсивних функцій. Для цього потрібно по можливості точно описати основні конструкції алгоритму і переформулювати (конкретизувати) тезу Чорча для більш вузьких класів алгоритмічно обчислюваних функцій.

За допомогою такого підхщду належність функцій до класів алгоритмічно обчислюваних можна аргументувати побудовою відповідних алгоритмів, на відміну від алгебраїчних термів теорії рекурсивних функцій. Цей підхід 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 викладення теорії алгоритмів на основі конкретизації тези Черча дозволяє досягти не тільки кращого сприйняття його студентами, а й робить більш зрозумілими формулювання різних задач та процес їх розв'язання.

Це досягається в першу чергу за рахунок так званих макрооператорів мови ПсевдоPackal для реалізації найпростіших механічних операцій.

Від читача ніяких попередніх знань не вимагається. Більшість доведень важливих результатів теорії алгоритмів приводиться повністю з урахуванням специфіки запропонованого підходу – доведення полягає у побудові алгоритмів того чи іншого класу.

Логічна символіка використовується дуже обмежено в зрозумілій для читача інтерпретації.

О.І. Провотар

Зауваження про основні позначення

Позначення	*Назва*	*Сторінка*
Σ^*	Множина всих слів в алфавіті Σ	7
N	Множина натуральних чисел	8
1^x	Слово з 1 довжиною x	10
$x := y$	Змінній з іменем x надати значення y	10
$f(x) = [x]$	Ціла частина x	20
$\exists x\ (\ldots)$	Існує x таке, що ...	43
$\Rightarrow$	Якщо, то	38
$\Leftrightarrow$	Тоді і тільки тоді	41
Σ	Множиина термінальних символів	92
$\Rightarrow$	Виведення в граматиці	93
$\forall\ x\ (\ldots)$	Для будь-якого x виконується ...	104
R	Множина раціональних чисел	107
$\rightarrow$	Якщо, то	116

Поняття алгоритму

Алгоритм Евкліда (визначення НСД).

Вхід. p і q, додатні цілі числа.
Вихід. g, НСД чрg. 7исел p і q.
Метод. 1. Знайти r, остачу від ділення p на q.
2. Якщо $r = 0$, покласти $g = q$ і зупинитись.
Інакше, покласти $p = q$, $q = r$ і перейти на l.

Алгоритм – всюди визначений, якщо він зупиняється на всіх входах, тобто на всіх значеннях вхідних даних.

Приклад 1.1.
Вхід. p і q, додатні цілі числа.
Вихід. g, сума чисел p і q.
Метод. Покласти $g = p + q$ і зупинитись.

Алгоритм – частковий, якщо він не зупиняється на деяких входах.

Приклад 1.2.
Вхід. p – довільне ціле число.
Вихід. p.
Метод. 1. Якщо $p = 0$, то перейти на 1. Інакше – зупинитись.

При $p = 0$ алгоритм не зупиняється.

1.1. Загальні риси алгоритмів

1. *Дискретність.* В процесі виконання алгоритму із початкової скінченної системи величин та величин, знайдених в попередні моменти часу послідовно (в дискретному часі) будується система величин по деякому закону (програмі).

2. *Елементарність.* Закон повинен бути простим (конструктивним).

3. *Детермінованість.* Система величин, які одержуються в якийсь момент часу однозначно визначається системою величин, одержаних в попередні моменти часу.

4. *Результативність.* Якщо спосіб одержання величини не дає результату, то повинно бути вказано, що вважати результатом алгоритму.

5. *Масовість.* Початкова система величин (вхід алгоритму) може вибиратись із потенційно нескінченної множини.

Алгоритм, визначений загальними рисами 1-5 називається *інтуїтивним* поняттям алгоритму. Такому інтуїтивному поняттю алгоритму задовольняє наступне: алгоритм – це скінченна множина правил (програма), яка дозволяє кожному елементу x із деякої нескінченної області (області визначення алгоритму) ставити у відповідність скінченну послідовність елементів $<x_0, x_1, \ldots, x_k>$ (процес обчислень) так, що кожний наступний елемент x_{i+1} будується за попереднім елементом x_i застосуванням до нього деякого елементарного правила (крок алгоритму), а застосування правила до елемента x_k уже неможливе. В цьому випадку елемент x_k вважається результатом застосування алгоритму до елемента x.

Отже, алгоритм – це скінченна послідовність правил "конструктивного" перетворення вхідних даних у вихідні.

Якщо алгоритм зупиняється через скінченну кількість кроків (скінченна кількість застосувань правил перетворення), то говорять, що алгоритм визначений на вхідних даних, в іншому випадку – коли алгоритм працює нескінченно довго – говорять, що алгоритм невизначений на вхідних даних.

Приклад 1.3. Необхідно знайти алгоритм, який дозволяє для кожної четвірки цілих чисел a, b, c, d вияснити, існують чи ні цілі числа x, y, які задовольняють рівнянню

$$ax^2 + bxy + cy^2 = d.$$

Такий алгоритм був побудований.

Приклад 1.4 (проблема відповідностей Поста). Нехай $(x, y) \subseteq \Sigma^* \times \Sigma^*$ – скінченна множина пар непустих слів в алфавіті Σ.

Проблема. Чи існує скінченна послідовність пар $(x_1, y_1), \ldots, (x_n, y_n)$ (не обов'язково різних) таких, що

$$x_1 x_2 \ldots x_n = y_1 y_2 \ldots y_n.$$

Така послідовність називається розв'язуючою.

Приклад 1.5. Нехай $\{(abbb, b), (a, aab), (ba, b)\}$ – скінченна множина пар в алфавіті $\Sigma = \{a, b\}$.

Тоді послідовність $(a, aab), (a, aab), (ba, b), (abbb, b)$ – розв'язуюча так, як

$$(a)(a)(ba)(abbb) = (aab)(aab)(b)(b).$$

Скінченна множина пар $\{(ab, aba), (aba, baa), (baa, aa)\}$ пар в цьому ж алфавіті розв'язуючих послідовностей не має.

Побудувати алгоритм, який розв'язує цю проблему неможливо. Але для цього потрібно довести неіснування алгоритму. Як це зробити? Для цього

треба точно знати, що таке алгоритм, тобто мати якесь математичне поняття, еквівалентне поняттю алгоритму.

Розв'язання задачі точного визначення алгоритму було одержано в роботах Гільберта, Геделя, Чьорча, Кліні, Поста, Т'юрінга. Для цього розглядалися функції $f{:}N \to N$ і клас функцій, які обчислюються алгоритмами ототожнювався з класом спеціально побудованих функцій – рекурсивних (таке ототожнення або гіпотеза відома під назвою тези Чьорча).

Алгоритмічна обчислюваність функцій означає існування алгоритму, який знаходить їх значення (у випадку визначеності функцій) і працює нескінченно довго в іншому.

Тези Чьорча достатньо, щоб доводити нерозв'язність алгоритмічних проблем. Тому, що питання про алгоритмічну обчислюваність функцій еквівалентне питанню про її рекурсивність.

Поняття рекурсивної функції – математично точне. Тому можна безпосередньо доводити, що функція, яка розв'язує задачу не може бути рекурсивною.

1.2. Поняття примітивно рекурсивної, частково рекурсивної та рекурсивної функцій.

Розглянемо спосіб опису алгоритмічно обчислюваних функцій, що грунтується на породженні таких функцій за допомогою певних обчислюваних операцій із так званих базових функцій.

Надалі під функцією будемо розуміти функцію натуральних аргументів і значень.

Базовими функціями називаються найпростіші функції

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

Всі базові функції всюди визначені.

Основними обчислювальними операціями будуть операції *суперпозиції* $\boldsymbol{S}^{n+1}$, *примітивної рекурсії* $\boldsymbol{R}$ та *мінімізації* $\boldsymbol{M}$.

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

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

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

Операція примітивної рекурсії $\boldsymbol{R}$ дозволяє із n-арної функції g та $n + 2$-арної функції h утворити $n + 1$-арну функцію f за допомогою наступних співвідношень:

$$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n)$$
$$f(x_1, \dots, x_n, y + 1) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)).$$

Таку функцію позначають $\boldsymbol{R}(g, h)$.

Операція мінімізації $\boldsymbol{M}$ дозволяє із n-арної функції g утворити n-арну функцію f, що задається співвідношенням:

$$f(x_1, \dots, x_n) = \mu_y(g(x_1, \dots, x_{n-1}, y) = x_n).$$

Тобто, значення функції $f(x_1, \dots, x_n)$ дорівнює найменшому y для якого $g(x_1, \dots, x_{n-1}, y) = x_n$.

Значення функції $f(x_1, \dots, x_n)$ вважається невизначеним у випадках:

1) для всіх y значення $g(x_1, \dots, x_{n-1}, y) \neq x_n$;

2) для всіх $y < t$ значення $g(x_1, \dots, x_{n-1}, y)$ визначене $i \neq x_n$, а значення $g(x_1, \dots, x_{n-1}, t)$ невизначене;

3) значення $g(x_1, \dots, x_{n-1}, 0)$ невизначене.

Таку функцію позначають $\boldsymbol{M}(g)$.

Функцію, яку можна одержати з базових функцій за допомогою скінченної кількості застосувань операцій суперпозиції та примітивної рекурсії, називають *примітивно рекурсивною функцією* (скорочено ПРФ).

Функцію, яку можна одержати з базових функцій за допомогою скінченної кількості застосувань операцій суперпозиції, примітивної рекурсії та мінімізації, називають *частково рекурсивною функцією* (скорочено ЧРФ).

Всюди визначену ЧРФ називають *рекурсивною функцією* (скорочено РФ).

Із визначень ПРФ, ЧРФ, РФ маємо такі співвідношення між класами функцій:

$$\text{ПРФ} \subseteq \text{РФ} \subseteq \text{ЧРФ}.$$

Приклад 1.6. За визначенням, функції

$$o(x) = 0,\ s(x) = x + 1,\ I_m^n(x_1, \dots, x_n) = x_m,$$

де $n \geq m \geq 1$ – базові.

Для n-місної функції $o^n(x_1, \dots, x_n) = 0$ маємо

$$o^n(x_1, \dots, x_n) = \boldsymbol{S}^2(o, I_1^n(x_1, \dots, x_n))$$

і тому функція o^n – примітивно рекурсивна.

Зрозуміло, що існують алгоритми для обчислення значень базових функцій

$$o(x) = 0,\ s(x) = x + 1,\ I_m^n(x_1, \dots, x_n) = x_m.$$

При цьому конструктивне перетворення вхідних даних функції $f(x_1, \dots, x_n)$ в результати означає, що для будь-яких натуральних чисел $x_1, \dots, x_n$ вхідне слово 1^{x_1} # … # 1^{x_n} перетворюється в слово $1^{f(x_1, \dots, x_n)}$, якщо $f(x_1, \dots, x_n)$ визначене і працює нескінченно довго, якщо $f(x_1, \dots, x_n)$ не визначене ($1^{x_i} - x_i$ одиниць). Так, обчислення функції $o(x)$ зводиться до витирання вихідного слова. Обчислення функції $s(x)$ до дописування одиниці до вхідного слова. Обчислення $I_m^n(x_1, \dots, x_n)$ до стирання всих вхідних компонент, крім m-ї.

Оператори суперпозиції, примітивної рекурсії та мінімізації теж породжують алгоритмічно обчислювані функції із алгоритмічно обчислюваних функцій.

Дійсно, у випадку суперпозиції, якщо ми можемо обчислювати значення функцій $g, g_1, \dots, g_n$, то значення їх суперпозиції $f = \boldsymbol{S}^{n+1}(g, g_1, \dots, g_n)$ в точці $(a_1, \dots, a_m)$ можна обчислити за допомогою наступного алгоритму

function $f(x_1, \dots, x_m)$
  begin
    $b_1 := g_1(x_1, \dots, x_m)$
    ……
    $b_n := g_n(x_1, \dots, x_m)$
    $b := g(b_1, \dots, b_n)$
  end.

Число b буде значенням функції f в точці $(a_1, \dots, a_m)$.

У випадку примітивної рекурсії, якщо ми можемо обчислювати значення функцій g, h, то значення функції $f = \boldsymbol{R}(g, h)$ в точці $(a_1, \dots, a_n, m + 1)$ може бути обчислене за допомогою наступного алгоритму

function $f(x_1, \dots, x_n, y)$
  begin
    if $y = 0$ then $f := g(x_1, \dots, x_n)$
    else $f := h(x_1, \dots, x_n, y - 1, f(x_1, \dots, x_n, y - 1))$
  end.

В результаті рекурсивних викликів, одержимо наступну послідовність чисел:

$b_0 = g(a_1, \dots, a_n)$,
$b_1 = h(a_1, \dots, a_n, 0, b_0)$,
$b_2 = h(a_1, \dots, a_n, 1, b_1)$,
……..,
$b_{m+1} = h(a_1, \dots, a_n, m, b_m)$.

Число b_{m+1} буде значенням функції f в точці $(a_1, \dots, a_n, m + 1)$.

У випадку операції мінімізації, якщо ми можемо обчислювати значення функції g, то значення функції $f = \boldsymbol{M}(g)$ в точці $(a_1, \dots, a_n)$ може бути обчислене за допомогою наступного алгоритму

function $f(x_1, \dots, x_n)$
begin
$i := 0$
while $g(x_1, \dots, x_{n-1}, i) \neq x_n$
do $i := i + 1$
$f := i$
end.

Іншими словами, ми послідовно обчислюємо значення

$g(a_1, \dots, a_n, 0)$,
$g(a_1, \dots, a_n, 1)$,
$\dots$.

Найменше значення a, для якого

$$g(a_1, \dots, a_{n-1}, a) = a_n$$

і є значенням функції f в точці $(a_1, \dots, a_n)$.

Характеристичною функцією χ_A множини натуральних чисел A називається одномісна функція, рівна 0 в точках множини A і рівна 1 в точках, які не належать A.

Частковою характеристичною функцією множини натуральних чисел A називається одномісна функція, рівна 0 в точках множини A і не визначена в точках, які не належать A.

Множина натуральних чисел A називається *примітивно рекурсивною*, якщо її характеристична функція примітивно рекурсивна.

Множина натуральних чисел A називається *рекурсивною*, якщо її характеристична функція рекурсивна.

Зрозуміло, що кожна примітивно рекурсивна функція є рекурсивною (за визначенням). Звідси випливає, що кожна примітивно рекурсивна множина є рекурсивною.

Теза Черча. *Клас алгоритмічно обчислюваних числових функцій співпадає з класом всіх частково рекурсивних функцій.*

Далі алгоритми будемо записувати в мові ПсевдоPascal, яка є спрощеним діалектом мови Pascal. Операторами цієї мови будуть наступні:

<ідентифікатор> := <вираз>,
if <предикат> then <оператор> | {<оператор>, … , <оператор>}

else <оператор> | {<оператор>, … , <оператор>},
while < предикат> do <оператор> | {<оператор>, … , <оператор>},
for <вираз> to <вираз> <оператор> | {<оператор>, … , <оператор>}.

Нехай $\chi(x)$ – характеристична функція множини натуральних чисел A. Тоді функція $\chi_ч(x) = 0 - \chi(x)$ – часткова характеристична функція множини A.

Теорема 1.1. Нехай $f(x)$ – примітивно рекурсивна функція, A – примітивно рекурсивна множина. Тоді часткова функція $f_ч(x) = f(x)$, якщо $x \in A$ і невизначена, якщо $x \notin A$ є частково рекурсивною.

Доведення. Існує алгоритм, який обчислює її значення в точках, де вона визначена і працює нескінченно довго в точках, де вона невизначена.

```
function f_ч(x)
    begin
      i = 0
      while χ_A(x) ≠ 0
          do i = i + 1
      f_ч = f(x)
    end.
```

Всюди визначені частково рекурсивні функції називаються *загальнорекурсивними*.

2

Властивості ПР функцій

Крім приведеної вище схеми рекурсії, можна розглянути більш прості схеми. Наприклад, функція f може задаватись рекурсивною схемою виду:

$$f(0) = a$$
$$f(x + 1) = h(f(x)).$$

В цьому випадку вона може бути обчислена рекурсивним і ітераційним алгоритмами виду:

```
function f(x)
    begin
      if x = 0 then f := a
      else f := h(f(x – 1))
    end
```

```
function f(x)
    begin
      s := a
      if x = 0 then f := s
      else for i = 1 to x
            s := h(s)
      f := s
    end.
```

Приклад 2.1. Функція $f(x) = x \dot{-} 1$ може бути обчислена наступними алгоритмами:

```
function f (x)
    begin
      if x = 0 ∨ x = 1 then f := 0
      else f := f(x – 1) + 1
    end.
```

```
function f (x)
    begin
      s := 0
      if x = 0 ∨ x = 1 then f := 0
      else for i = 2 to x
            s := s + 1
      f := s
    end.
```

Ітераційний алгоритм обчислення функції $f(x)$ використовує функцію $s(x) = x + 1$. Зрозуміло, що ця функція обчислюється всюди визначеним алгоритмом без використання оператора while … do. Тому і функція $f(x)$ обчис-

люється всюди визначеним алгоритмом без використання оператора while … do.

Якщо ж рекурсивна схема має вигляд:

$$f(0) = a$$
$$f(x + 1) = h(x, f(x)),$$

то рекурсивний та ітераційний алгоритми обчислення функції f будуть наступними:

```
function f(x)
    begin
      s := a
      if x = 0 then f := s
      else f := h(x – 1, f(x – 1))
    end
```

```
function f(x)
    begin
      s := a
      if x = 0 then f := s
      else for i = 1 to x
          begin
            j := i ∸1
            s := h(j, s)
          end
      f := s
    end.
```

Нарешті, у випадку загальної схеми рекурсії рекурсивний та ітераційний алгоритми обчислення функції f мають вигляд:

```
function f(x, y)
    begin
      s := g(x)
      if y = 0 then f := s
      else f := h(x, y – 1, f(x, y – 1))
    end.
```

```
function f(x, y)
    begin
      s := g(x)
      if y = 0 then f := s
      else for i = 1 to y
          begin
            j := i ∸1
            s := h(x, j, s)
          end
      f := s
    end.
```

Отже, рекурсивні алгоритми задають спосіб обчислення рекурсивної функції, а саме: значення функції в довільній точці обчислюється через значення цієї функції для менших значень аргументу. Тому символ "–" в рекурсивних алгоритмах означає не операцію віднімання, а описує організацію процесу обчислення функції. Ітераційні алгоритми, в свою чергу, описують іншу організацію процесу обчислення.

2.1. Конкретизація тези Черча

Далі, конкретизуючи тезу Черча для класів ПРФ, РФ та ЧРФ і з урахуванням вище сказаного, можна одержати інші визначення цих класів, а саме:

функцію, яку можна обчислити всюди визначеним алгоритмом будемо називати рекурсивною;

функцію, яку можна обчислити всюди визначеним алгоритмом без використання оператора while … do будемо називати примітивно рекурсивною;

функцію, яку можна обчислити довільним алгоритмом будемо називати частково рекурсивною.

Зауваження. Можна запропонувати більш строге визначення ПРФ, РФ та ЧРФ. Воно полягає у визначенні таких класів функцій:

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

1) Вважатимемо, що найпростіші функції

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

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

Крім того, всюди визначеними алгоритмами без використання оператора while … do обчислюються найпростіші функції-предикати $x=y$, $x\neq y$, $x>y$, $x<y$, $x\leq y$, $x\geq y$, а, отже теж належать до цього класу..

2) Всі функції, які обчислюються всюди визначеними алгоритмами без використання оператора while … do і при їх обчисленні використовуються 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:

1) Всі ПРФ належать до цього класу.

2) 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:

1) Всі РФ належать до цього класу.

2) 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.

Приклад 2.2. Функція $f(x, y) = x + y$ є ПРФ.

Дійсно, алгоритм обчислення цієї функції наступний:

```
function f(x, y)
    begin
        if y = 0 then f := x
        else f := f(x, y – 1) + 1
    end.
```

Приклад 2.3. Функція $f(x, y) = x^y$ є ПРФ.

Дійсно, алгоритм обчислення цієї функції наступний:

```
function f(x, y)
    begin
        if y = 0 then f := 1
        else f := f(x, y – 1) × x
    end.
```

Приклад 2.4. Функція $f(x, y) = x \dot{-} y$ є ПРФ.

Дійсно, алгоритм обчислення цієї функції наступний:

```
function f(x, y)
    begin
        if x < y ∨ x = y then f := 0
        else f := f(x – 1, y) + 1
    end.
```

2.2. Теореми про сумування і множення

Теорема 2.1 (про сумування)**.** Нехай функція g примітивно рекурсивна. Тоді функція f, яка визначається рівністю

$$f(x)=\sum_{i=0}^{x} g(i)$$

теж примітивно рекурсивна.

Доведення.

```
function  f(x)
    begin
       if x = 0 then f := g(0)
       else f := f(x – 1) + g(x)
    end
```

```
2. function f(x)
        begin
           s := 0
           for i = 0 to x
                s := s + g(i)
           f := s
        end
```

Наслідок 2.1. Якщо функція g примітивно рекурсивна, то 2-місна функція f, яка визначається схемою

$$f(x, y) = \begin{cases} \sum_{i=x}^{y} g(i), & x \le y \\ 0, & x > y \end{cases}$$

також ПР функція.

Доведення.

```
function f(x,y)
    begin
       if x > y then f := 0
       else
          begin
             s := 0
             for i = x to y
                 s := s + g(i)
             f := s
          end
    end.
```

Наслідок 2.2. Якщо g, h, k – ПР функції, то функція f^*, що визначається співвідношенням

$$f^*(x)=\begin{cases} \sum_{h(x)}^{k(x)} g(i), \text{ якщо } h(x) \le k(x) \\ 0, \text{якщо } h(x) > k(x) \end{cases}$$

також ПР функція.

Ця функція є суперпозицією функції f з наслідку 2.1 та функцій h, k ($f^*(x) = f(h(x), k(x))$) і обчислюється наступним алгоритмом:

```
function  f*(x)
    begin
        if   h(x) > k(x) then f*:= 0
        else f* := f(h(x), k(x))
    end
```

де f – функція з наслідку 1.

Приклад 2.5. Функція $f(x) = x\cdot(x + 1)/2$ є ПРФ.

Нехай $g(x) = x$. Тоді

$$f(x) = \sum_{i=0}^{x} g(i).$$

Отже, за теоремою про сумування фунція $f(x)$ є ПРФ, оскільки функція $g(x)$ є ПРФ.

Теорема 2.2 (про множення)**.** Нехай функція g примітивно рекурсивна. Тоді функція f, яка визначається рівністю

$$f(x) = \prod_{i=0}^{x} g(i)$$

теж примітивно рекурсивна.

Доведення.

```
function  f(x)
    begin
        if   x = 0 then f := g(0)
        else    f := f(x – 1) g(x)
    end.
```

Приклад 2.6. Функція $f(x) = x!$ є ПРФ.

Нехай $g(x) = x$. Тоді

$$f(x) = \prod_{i=0}^{x} g(i).$$

Отже, за теоремою про сумування фунція $f(x)$ є ПРФ, оскільки функція $g(x)$ є ПРФ.

Теорема 2.3. Нехай функції $f_1, f_2, \dots, f_{k+1}, \alpha_1, \alpha_2, \dots, \alpha_k$ примітивно рекурсивні, причому при будь-яких значеннях змінних ніякі дві з функцій $\alpha_1, \alpha_2, \dots, \alpha_k$ не дорівнюють 0. Тоді функція, яка визначається кусковою схемою

$$f(x)=\begin{cases} f_1(x), \text{якщо } \alpha_1(x)=0, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ f_k(x), \text{якщо } \alpha_k(x)=0, \\ f_{k+1}(x) \text{ в інших випадках,} \end{cases}$$

буде примітивно рекурсивною.

Доведення.

```
function  f(x)
      begin
         if   α1(x) = 0 then f := f1(x)
         if   α2(x) = 0 then f := f2(x)
         ……….
         if   αk(x) = 0 then f := fk(x)
         else f = fk+1(x)
      end.
```

В теоремі 2.3 розглянуто типовий випадок, коли умова має вигляд $\alpha_i = 0$. Так як умови вигляду

$$\alpha_i = \beta_i,\ \alpha_i \le \beta_i,\ \alpha_i < \beta_i$$

рівносильні, відповідно, умовам

$$|\alpha_i - \beta_i| = 0,\ \ \alpha_i \dot{-} \beta_i = 0,\ \overline{sg}\ (\beta_i \dot{-} \alpha_i) = 0,$$

то теорема 2.3 залишається справедливою і в тому разі, коли в кусковій схемі рівності $\alpha_i = 0$ замінюються іншими умовами, де α_i, β_i – ПР функції.

Приклад 2.7. Функція $\max(x, y)$ є ПРФ.

Дійсно,

$$\max(x,y)=\begin{cases} x, & x \ge y \Leftrightarrow y \dot{-} x = 0 \\ y, & x < y \Leftrightarrow \overline{sg}(y \dot{-} x) = 0 \end{cases}.$$

Отже, за теоремою 2.3 фунція $\max(x, y)$ є ПР.

2.3. Мажоруючі неявні функції

Розглянемо рівняння

$$g(x, y) = 0,$$

ліва частина якого є всюди визначена функція. Припустимо, що для кожного значення x це рівняння має єдиний розв'язок y. Тоді цей розв'язок буде однозначною всюди визначеною функцією від x. Чи буде ця функція примітивно рекурсивною, якщо функція g є ПР функцією від x, y? В загальному випадку відповідь на це питання негативна. Але справедлива наступна теорема.

Теорема 2.4. Нехай $g(x, y)$, $\alpha(x)$ такі примітивно рекурсивні функції, що рівняння

$$g(x, y) = 0$$

для кожного x має хоча б один розв'язок і

$$\mu_y(g(x, y) = 0) \le \alpha(x)$$

для будь-якого x. Тоді функція

$$f(x) = \mu_y(g(x, y) = 0)$$

теж примітивно рекурсивна.

Доведення.

```
function  f(x)
    begin
        s := 0
        for i = 0 to α(x)
        if h(x, i) ≠ 0 then s := s + 1
        f := s
    end,
```

де $h(x, i) = g(x, 0) \cdot \ldots \cdot g(x, i)$.

Приклад 2.8. Функція $f(x) = [x\sqrt{2}\,]$ є ПРФ.

Дійсно, маємо

$$[x\sqrt{2}\,] \le x\sqrt{2} < [x\sqrt{2}\,] + 1.$$

Тому,

$$[x\sqrt{2}\,]^2 \le 2x^2 < ([x\sqrt{2}\,] + 1)^2.$$

Покладемо

$$g(x, z) = \overline{sg}\,((z + 1)^2 \dot{-} 2x^2),\ \alpha(x) = 2x.$$

Покажемо, що

$$\mu_z((\overline{sg}\,((z + 1)^2 \dot{-} 2x^2) = 0) = [x\sqrt{2}\,].$$

Дійсно,

$$([x\sqrt{2}\,] + 1)^2 \dot{-} 2x^2 > 0,$$

Тому $[x\sqrt{2}]$ є розв'язком рівняння

$$\overline{sg}((z+1)^2 \dot{-} 2x^2) = 0.$$

Крім того, це найменший розв'язок цього рівняння. Дійсно, якщо $z_0 < [x\sqrt{2}]$ є розв'язком рівняння, то

$$(z_0+1)^2 \le [x\sqrt{2}] \le 2x^2.$$

Тому, $(z_0+1)^2 \dot{-} 2x^2 = 0$, а, отже $\overline{sg}((z_0+1)^2 \dot{-} 2x^2) = 1$.

Отже, за теоремою 2.4 фунція $f(x) = [x\sqrt{2}]$ є ПРФ.

3

Примітивна рекурсивність деяких арифметичних функцій

Нехай

$$f(x, y) = [x/y] –$$

частка від ділення x на y, а

$$g(x, y) = rest(x, y) –$$

остача від ділення x на y.

Для того, щоб введені функції були всюди визначені, покладемо

$$[x/0] = x,\ rest(x, 0) = x$$

для всіх x. Зрозуміло, що так визначені функції зв’язані тотожністю

$$rest(x, y) = x \dot{-} (y \times [x/y]).$$

Тому, із того, що функція $[x/y]$ – ПР функція, буде випливати, що $rest(x, y)$ – ПР функція

Теорема 3.1. Функція

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

ПР функція.

Доведення. Розглянемо послідовність

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

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
              begin
                  for i = 1 to x
                   if iy ∸ x = 0 then s := s + 1
                   f := s
              end
end.
```

Нехай $div(x, y) = 1$, якщо $rest(x, y) = 0$, $div(x, y) = 0$, якщо $rest(x, y) \neq 0$.

Теорема 3.2. Функція $div(x, y)$ ПР функція.
Доведення. Функція $div(x, y)$ обчислюється наступним алгоритмом:

```
function  div(x, y)
      begin
          if rest(x, y) = 0 then div := 1
          else div := 0
      end.
```

Нехай

$$nd(x) = \sum_{i=0}^{x} div(x, i).$$

При $x \neq 0$ число $nd(x)$ співпадає з числом різних дільників числа x. Крім того, функція $nd(x)$ – ПР функція.

Теорема 3.3. Функція $nd(x)$ ПР функція.
Доведення. Функція $nd(x)$ обчислюється наступним алгоритмом:

```
function nd(x)
      begin
          s := 0
          for i = 0 to x
               s := s + div(x, i)
          nd := s
      end.
```

Позначимо через $\chi_p(x)$ функцію таку, що $\chi_p(x) = 0$ для x простого і $\chi_p(x) = 1$ для x непростого.

Теорема 3.4. Функція $\chi_p(x)$ ПР функція.
Доведення. Функція $\chi_p(x)$ обчислюється наступним алгоритмом:

```
function  χp(x)
    begin
      if  nd(x) = 2 then χp := 0
      else χp := 1
    end.
```

Нехай

$$\pi(x) = \sum_{i=0}^{x} \overline{sg}\,\chi_p(i),$$

тобто $\pi(x)$ дорівнює числу простих чисел не більших x.

Теорема 3.5. Функція $\pi(x)$ ПР функція.

Доведення. Випливає з теореми про сумування.

Для простих чисел 2, 3, 5, 7, ... введемо функцію $p(n)$, значенням якої є $(n+1)$-е просте число в натуральному ряді чисел.

Теорема 3.6. Функція $p(n) = p_n$ є ПРФ.

Дійсно, алгоритм обчислення цієї функції наступний:

```
function  p(n)
    begin
      s := 0
      k := 2
      for   i = 2 to 2^(2^n)
       if χp(i) = 0 ∧ s ≤ n then
            begin
              s := s + 1
              k := i
            end
       p := k
    end.
```

Теорема 3.7. Функція $f(x) = [\sqrt{x}\,]$ є ПРФ.

Дійсно, алгоритм обчислення цієї функції наступний:

```
function f(x)
    begin
      if x = 0 then f := 0
      else
        for i = 1 to x
        if (i^2 ∸ x) ≤ 0 ∧ ((i+1)^2 ∸ x) > 0
        then f := i + 1
    end.
```

3.1. Рекурсія з поверненням

Нехай $\alpha_1(x), \dots, \alpha_s(x)$ – всюди визначені функції, які для всіх значень x задовольняють умовам

$$\alpha_i(x + 1) \le x \quad (i = 1, \dots, s).$$

Говорять, що функція $f(x)$ одержується із функцій $h(y, z_1, \dots, z_s)$ та допоміжних функцій $\alpha_1, \dots, \alpha_s$ рекурсією з поверненням, якщо для всіх y

$$f(0) = a,$$
$$f(y + 1) = h(y, f(\alpha_1(y + 1)), \dots, f(\alpha_s(y + 1))).$$

В більш спрощеному вигляді ця схема рекурсії має вигляд:

$$f(0) = a,$$
$$f(x + 1) = h(x, f(\alpha(x + 1))).$$

Теорема 3.8. Якщо функції h та α_i ПРФ, то функція $f(x)$ є ПРФ.

Доведемо теорему для спрощеної схеми рекурсії. Введемо функцію

$$F(x) = \prod_{i=0}^{x} p(i)^{f(i)}.$$

Значення цієї функції в точках 0, 1, 2, ... обчислюється наступним чином:

$$F(0) = p(0)^{f(0)}, F(1) = p(0)^{f(0)}p(1)^{f(1)}, F(2) = p(0)^{f(0)}p(1)^{f(1)}p(2)^{f(2)}, \dots.$$

Тому, значення функції f точці x може бути обчислене як $f(x) = exp(p(x), F(x))$, а для $u < x$ значення цієї функції обчислюється як $f(u) = exp(p(u), F(x))$.

Покажемо, що функція $F(x)$ задається рекурсивною схемою. Дійсно,

$$F(0) = 2^a,$$
$$F(x + 1) = F(x)p(x + 1)^{f(x+1)} = F(x)p(x + 1)^{h(x, f(\alpha(x+1)))} =$$
$$F(x)p(x + 1)^{h(x, exp(p(\alpha(x+1)), F(x))}.$$

Таким чином, функція $f(x)$ є ПРФ.

Інше доведення полягає в побудові алгоритму для обчислення функції f. Значення функції f в точці a може бути обчислене за допомогою наступного алгоритму:

function $f(x)$

begin
  if $x = 0$ then $f := b$
  else $f := h(x, f(\alpha(x)))$
end.

В результаті, наприклад, 2-х рекурсивних викликів, при умові, що $\alpha(\alpha(2)) = 0$ та $x = 2$, одержимо наступну послідовність чисел:

$$b_0 = b,$$
$$b_1 = h(\alpha(\alpha(2)), b),$$
$$b_2 = h(\alpha(2), b_1).$$

Число b_2 буде значенням функції f в точці 2.

Приклад (Послідовність Фібоначчі). Нехай функція $G(x)$ задається рівностями

$$G(0) = 0,$$
$$G(x + 1) = G(x) + G(x \dot{-} 1) + \overline{sg}\, x.$$

Очевидно, що $G(x)$ одержується рекурсією 2-го рівня з функції

$$h(y, z_1, z_2) = \overline{sg}\, y + z_1 + z_2$$

та допоміжних функцій

$$\alpha_1(y) = y \dot{-} 1,\ \alpha_2(y) = y \dot{-} 2.$$

($G(x + 1) = h(x, G(\alpha_1(x + 1)), G(\alpha_2(x + 1))$).

Отже, $F(x)$ – ПР функція.

4

Нумерації пар і n-ок чисел

Одну із бієкцій (нумерацій) між N та $N \times N$ можна задати наступним чином. Всі пари натуральних чисел розташуємо в послідовність

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

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

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

або якщо

$$x + y = u + v \text{ і } x < u.$$

Бієкцію задаємо співвідношенням

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

де n – номер пари в цій послідовності.

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.

Теорема 4.1. Функції $\boldsymbol{c}(x, y)$, $\boldsymbol{l}(n)$, $\boldsymbol{r}(n)$ – ПР функції.

Доведення. 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)
            begin
                j := (x + y) ∸ i
                if  x = i ∧ y = j then k := i
            end
        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 c(i, j) = n  then r := j
end.
```

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

Зазначимо також, що із визначення функцій $\boldsymbol{c}(x, y)$, $\boldsymbol{l}(n)$, $\boldsymbol{r}(n)$ випливають наступні співвідношення

$$\boldsymbol{c}(\boldsymbol{l}(n), \boldsymbol{r}(n)) = n,\ \boldsymbol{l}(\boldsymbol{c}(x, y)) = x,\ \boldsymbol{r}(\boldsymbol{c}(x, y)) = y.$$

З допомогою нумерації пар чисел можна одержати нумерацію трійок, четвірок і т. д. натуральних чисел. Для цього вводяться наступні функції

$$\boldsymbol{c}^3(x_1, x_2, x_3) = \boldsymbol{c}(\boldsymbol{c}(x_1, x_2), x_3)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$\boldsymbol{c}^{n+1}(x_1, x_2, \ldots, x_{n+1}) = \boldsymbol{c}^n(\boldsymbol{c}(x_1, x_2), x_3, \ldots, x_{n+1}).$$

За визначенням, число $\boldsymbol{c}^n(x_1, x_2, \ldots, x_n)$ називається канторовим номером n-ки $\langle x_1, x_2, \ldots, x_n \rangle$.

Якщо $\boldsymbol{c}^n(x_1, x_2, \ldots, x_n) = x$, то із тотожностей для $\boldsymbol{c}(x, y)$, $\boldsymbol{l}(x)$, $\boldsymbol{r}(x)$, одержимо:

$$x_n = \boldsymbol{r}(x),\ x_{n-1} = \boldsymbol{rl}(x),\ \ldots,\ x_2 = \boldsymbol{rl}\ \ldots\ \boldsymbol{l}(x),\ x_1 = \boldsymbol{ll}\ \ldots\ \boldsymbol{l}(x).$$

Ввівши позначення $\boldsymbol{c}_{nn}(x), \ldots, \boldsymbol{c}_{n1}(x)$ для правих частин вище приведених рівностей, одержимо:

$$\boldsymbol{c}^n(\boldsymbol{c}_{n1}(x), \ldots, \boldsymbol{c}_{nn}(x)) = x,$$
$$\boldsymbol{c}_{ni}(\boldsymbol{c}^n(x_1, x_2, \ldots, x_n)) = x_i\ (i = 1, \ldots, n).$$

Це аналоги канторової нумерації для n-ок чисел.

4.1. Функція Геделя

Введемо функцію

$$\boldsymbol{\Gamma}(x, y) = rest(\boldsymbol{l}(x), 1 + (y + 1)\cdot\boldsymbol{r}(x)).$$

Функція Геделя $\boldsymbol{\Gamma}(x, y)$ дозволяє генерувати довільні скінченні послідовності натуральних чисел згідно наступної властивості цієї функції: для будь-яких чисел $a_0, \dots, a_n \in N$ існує x таке, що $\boldsymbol{\Gamma}(x, i) = a_i$, $i = 0, \dots, n$.

Лема. Для будь-яких взаємно простих чисел q, p має місце наступне співвідношення:

$$\{rest(p\cdot i, q) \mid i = 0, \dots, q - 1\} = \{0, \dots, q - 1\}.$$

Доведення. Для будь-якого числа x маємо

$$0 \le rest(x, q) \le q - 1.$$

Тому достатньо показати, що для будь-яких $0 \le i < j \le q - 1$ виконується нерівність $rest(pi, q) \ne rest(p\cdot j, q)$. Допустимо противне. Тоді $p\cdot j - p\cdot i$ ділиться на q. Отже, $j - i$ ділиться на q, а це не так ($j - i \le q - 1$).

Теорема 4.2 (китайська теорема про остачі). Для будь-яких попарно простих чисел $p_1, \dots, p_s$ і натуральних чисел $a_1, \dots, a_s$ таких, що $a_i < p_i$, $i = 1, \dots, s$ існує натуральне число $M < p_1\times\dots\times p_s$ таке, що rest $(M, p_i) = a_i$ для всіх $i = 1, \dots, s$.

Доведення (індукцією по s).

1. Нехай $s = 2$. За попередньою лемою існують числа $n < p_1$, $m < p_2$ такі, що

$$rest(p_1\cdot m, p_2) = a_2,\ rest(p_2\cdot n, p_1) = a_1.$$

Нехай $M' = p_1\cdot m + p_2\cdot n$ і $M_0 = rest(M', p_1\cdot p_2)$. Тоді M_0 – шукане бо, $M_0 < p_1\cdot p_2$ і $rest(M_0, p_1) = a_1$, $rest(M_0, p_2) = a_2$ ($M' = k\cdot p_1\cdot p_2 + M_0$ для деякого k, а отже $rest(M_0, p_1) = rest(M' - k\cdot p_1\cdot p_2, p_1) = rest(M', p_1) = rest(p_2\cdot n, p_1) = a_1$).

2. Нехай твердження леми вірне для $s \le m$. Покажемо, що воно вірне і для $s = m + 1$.

Маємо натуральні $a_1, \dots, a_{s+1}$ та попарно прості $p_1, \dots, p_{s+1}$ такі, що $a_i < p_i$, $i = 1, \dots, s + 1$.

Покладемо, $p'_1 = p_1\cdot p_2$, $p'_i = p_{i+1}$, $a'_1 = M_0$, $a'_i = a_{i+1}$, $i = 2, \dots, s$. Тоді $p'_1, \dots, p'_s$ попарно прості та $a'_i < p'_i$ для всіх $i = 1, \dots, s$. Тому за припущенням індукції знайдеться натуральне M таке, що $M < p'_1 \dots p'_s$ та rest$(M, p'_i) = a'_i$ для всіх $i = 1, \dots, s$. Звідси маємо $M < p_1 \dots p_{s+1}$, $rest(M, p_i) = a_i$ для всіх $i = 3, \dots, s + 1$ та $rest(M, p_1\cdot p_2) = M_0$. Остання рівність означає, що $M = \beta\cdot p_1\cdot p_2 + M_0$ для деякого β. Звідси маємо

$$rest(M, p_1) = rest(M_0, p_1) = a_1,\ rest(M, p_2) = rest(M_0, p_2) = a_2.$$

Теорема 4.3. Для будь-якої послідовності $a_0, \dots, a_n$ натуральних чисел існує число $x \in N$ таке, що

$$\boldsymbol{\Gamma}(x, i) = a_i,\ i = 0, 1, \dots, n.$$

Доведення. Нехай R таке число, що числа $1 + (i + 1)\cdot R$, $i = 0, \dots, n$ попарно прості і $a_i < 1 + (i + 1)\cdot R$ для всіх $i = 0, \dots, n$. Тоді за попередньою теоремою існує число L таке, що $rest(L, 1 + (i + 1)\cdot R) = a_i$ для всіх $i = 0, \dots, n$. Покладемо $x = \boldsymbol{c}(L, R)$. Тоді

$$\begin{aligned}&\boldsymbol{\Gamma}(x, i) = rest(\boldsymbol{l}(x), 1 + (i + 1)\cdot \boldsymbol{r}(x)) = \\ &rest((\boldsymbol{l}(\boldsymbol{c}(L, R)), 1 + (i + 1)\cdot \boldsymbol{r}(\boldsymbol{c}(L, R))) = \\ &rest(L, 1 + (i + 1)\cdot R) = a_i\end{aligned}$$

для всіх $i = 0, \dots, n$ і теорема доведена.

Залишилось знайти число R. Таким числом є число $R = (1 + n + a_0 + \dots + a_n)!$. Дійсно, $a_i < 1 + (i + 1)\cdot R$ для всіх $i = 0, \dots, n$. Крім того числа $1 + (i + 1)\cdot R$ будуть попарно простими. Справді, припустимо супротивне: існує просте $q > 1$ таке, що для деяких i, j таких, що $j > i$, $1 + (i + 1)\cdot R$ та $1 + (j + 1)\cdot R$ діляться на q. Але $j - i \le n$, тому $j - i$ входить як множник в число R. Тому R ділиться на q, а отже $R = \gamma\cdot q$. Звідси

$$1 + (i + 1)\cdot R = 1 + (i + 1)\cdot \gamma\cdot q \text{ та } 1 + (j + 1)\cdot R = 1 + (j + 1)\cdot \gamma\cdot q$$

не діляться на q, а це суперечить припущенню. Теорему доведено.

Функція Геделя разом з деякими іншими функціями утворює розширення множини базових функцій, яке дозволяє промоделювати операцію примітивної рекурсії.

Приклад 4.1. Розв’язати систему рівнянь

$$\begin{aligned}&\boldsymbol{\Gamma}(x, 0) = 1,\\ &\boldsymbol{\Gamma}(x, 1) = 2,\\ &\boldsymbol{\Gamma}(x, 2) = 3.\end{aligned}$$

За попередньою теоремою знаходимо число R таке, що числа

$$1 + R,\ 1 + 2\cdot R \text{ та } 1 + 3\cdot R$$

попарно прості і

$$1 + R > 1,\ 1 + 2\cdot R > 2,\ 1 + 3\cdot R > 3.$$

Таким числом є число 2. Далі знаходимо число L таке, що

$$rest(L, 3) = 1,\ rest(L, 5) = 2,\ rest(L, 7) = 3.$$

Таким числом є число 52. Тоді розв'язок системи рівнянь шукаємо як канторовий номер пари (52, 2), тобто

$$x = \boldsymbol{c}(52, 2) = 1537.$$

4.2. Моделювання примітивної рекурсії

Теорема 4.4 (про моделювання примітивної рекурсії). Якщо функція виникає за схемою примітивної рекурсії із функцій g, h, то її значення в будь-якій точці може бути обчислене всюди визначеним алгоритмом через значення функцій g, h, $\boldsymbol{\Gamma}$.

Доведення. Доводимо для випадку, коли f – функція двох змінних (в загальному випадку доведення аналогічне). За схемою примітивної рекурсії маємо:

$$\begin{array}{l} f(x, 0) = g(x) \\ f(x, 1) = h(x, 0, f(x, 0)) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ f(x, y) = h(x, y-1, f(x, y-1)). \end{array} \qquad (1)$$

За основною властивістю функції Геделя існує таке t, що

$$\begin{array}{l} \boldsymbol{\Gamma}(t, 0) = f(x, 0) \\ \boldsymbol{\Gamma}(t, 1) = f(x, 1) \\ \ldots\ldots\ldots\ldots\ldots \\ \boldsymbol{\Gamma}(t, y) = f(x, y). \end{array} \qquad (2)$$

Перепишемо (2) у вигляді

$$\begin{array}{l} \boldsymbol{\Gamma}(t, 0) = g(x) \\ \boldsymbol{\Gamma}(t, 1) = h(x, 0, \boldsymbol{\Gamma}(t, 0)) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \boldsymbol{\Gamma}(t, y) = h(x, y-1, \boldsymbol{\Gamma}(t, y-1)). \end{array} \qquad (3)$$

Тоді функція $f(x, y)$ може бути обчислена алгоритмом:

```
function f(x, y)
    begin
```

```
i := 0
s := 0
if y = 0
then
    begin
      while |Γ(i, 0) – g(x)| ≠ 0
            do i = i + 1
    end
else
     begin
       for j = 1 to y
          if Γ(i, j) ≠ h(x, j – 1, Γ(i, j – 1)) then s := 1
       while |Γ(i, 0) – g(x)| + s ≠ 0
          do
             begin
               s := 0
               i := i + 1
               for j = 1 to y
                  if Γ(i, j) ≠ h(x, j – 1, Γ(i, j – 1)) then s := 1
             end
     end
  f := Γ(i, y)
end.
```

Теорему доведено.

Приклад 4.2. Обчислимо за допомогою приведеного алгоритму значення функції $f(x, y) = x + y$ в точці (0, 1). Ця функція задається рекурсивною схемою

$$f(x, y) = x$$
$$f(x, y +1) = f(x, y) + 1.$$

В загальному випадку ця рекурсивна схема має вигляд:

$$f(x, y) = g(x)$$
$$f(x, y + 1) = h(x, y, f(x, y)),$$

де $g(x) = x$, $h(x, y, z) = z + 1$.

Для $i = 0$, $j = 1$ обчислюємо:

$$\Gamma(0, 1) = rest\ (l(0), 1 + (1 + 1)\cdot r(0)) = 0,$$
$$\Gamma(0, 0) = rest\ (l(0), 1 + (0 + 1)\cdot r(0)) = 0.$$

Для $i = 1, j = 1$ обчислюємо:

$$\boldsymbol{\Gamma}(1, 1) = rest\ (\boldsymbol{l}(1), 1 + (1 + 1)\cdot\boldsymbol{r}(1)) = 0,$$
$$\boldsymbol{\Gamma}(1, 0) = rest\ (\boldsymbol{l}(1), 1 + (0 + 1)\cdot\boldsymbol{r}(1)) = 0.$$

Для $i = 2, j = 1$ обчислюємо:

$$\boldsymbol{\Gamma}(2, 1) = rest\ (\boldsymbol{l}(2), 1 + (1 + 1)\cdot\boldsymbol{r}(2)) = 0,$$
$$\boldsymbol{\Gamma}(2, 0) = rest\ (\boldsymbol{l}(2), 1 + (0 + 1)\cdot\boldsymbol{r}(2)) = 0.$$

Для $i = 3, j = 1$ обчислюємо:

$$\boldsymbol{\Gamma}(3, 1) = rest\ (\boldsymbol{l}(3), 1 + (1 + 1)\cdot\boldsymbol{r}(3)) = 0,$$
$$\boldsymbol{\Gamma}(3, 0) = rest\ (\boldsymbol{l}(3), 1 + (0 + 1)\cdot\boldsymbol{r}(3)) = 0.$$

Для $i = 4, j = 1$ обчислюємо:

$$\boldsymbol{\Gamma}(4, 1) = rest\ (\boldsymbol{l}(4), 1 + (1 + 1)\cdot\boldsymbol{r}(4)) = 1,$$
$$\boldsymbol{\Gamma}(4, 0) = rest\ (\boldsymbol{l}(4), 1 + (0 + 1)\cdot\boldsymbol{r}(4)) = 1.$$

Отже, $f(0, 1) = \boldsymbol{\Gamma}(4, 1) = 1$.

5

Рекурсивні та примітивно рекурсивні множини

Тут будуть розглянуті найпростіші властивості примітивно рекурсивних, рекурсивних та рекурсивно перелічимих множин.

Підмножина A множини натуральних чисел N називається *рекурсивною* (*примітивно рекурсивною*), якщо характеристична функція множини A рекурсивна (примітивно рекурсивна).

Так, як всі ПР функції рекурсивні, то кожна ПР множина є рекурсивною множиною.

Наслідок. Підмножина A множини натуральних чисел N *рекурсивна* або *примітивно рекурсивна* тоді і тільки, коли існує алгоритм, який для довільного $n \in N$ дає відповідь на питання $n \in A$ чи $n \notin A$.

Причому, у випадку ПР множини такий алгоритм будується з алгоритмів типу

$$\boldsymbol{S}^{n+1}(g, g_1, \dots, g_n), \boldsymbol{R}(g, h),$$

а у випадку P множини – з алгоритмів типу

$$\boldsymbol{S}^{n+1}(g, g_1, \dots, g_n), \boldsymbol{R}(g, h), \boldsymbol{M}^l(g),$$

де M^l – оператор слабкої мінімізації.

5.1. Властивості Р та ПР множин

Характеристичні функції порожньої множини $\varnothing$ та множини N – це одномісні функції 1 та 0. Ці фукції – ПР функції. Дійсно, вони обчислюються алгоритмами:

1. function $\chi_{\varnothing}(x)$
   begin
   $\chi_{\varnothing} := 1$
   end.

2. function $\chi_N(x)$
   begin
   $\chi_N := 0$
   end.

Тому множини $\varnothing$ та N – ПР множини.

Характеристичною функцією для скінченної множини чисел $A = \{a_1, \dots, a_n\}$ буде ПР функція $\chi_A(x) = \mathrm{sg}(|x - a_1||x - a_2| \dots |x - a_n|)$. Вона обчислюється алгоритмом:

```
function χ_A(x)
    begin
      if |x – a_1||x – a_2|…|x – a_n| = 0 then χ_A := 0
      else χ_A := 1
    end.
```

Тому, кожна скінченна множина натуральних чисел є ПР множиною.

ПР множинами будуть, наприклад, множина всіх парних чисел та множина всіх простих чисел. Характеристичними функціями цих множин будуть функції

$$d(x) = sg\ rest(x, 2) \text{ та}$$
$$\chi_p(x) = sg\ |nd(x) - 2|$$

відповідно. Вони обчислюються алгоритмами:

```
1. function d(x)
      begin
        if rest(x, 2) = 0 then d := 0
      else d := 1
    end.

2. function χ_p(x)
      begin
        if nd(x) = 2 then χ_p := 0
        else χ_p := 1
      end.
```

Аналогічно можна показати, що ПР множинами будуть і інші множини чисел.

Теорема 5.1. Доповнення Р (ПР) множини, а також об'єднання і перетин будь-якої скінченної системи Р (ПР) множин є Р (ПР) множиною.

Доведення. Нехай $f_1(x)$, …, $f_n(x)$ – характеристичні функції множин $A_1, \dots, A_n$. Тоді функції

$$f(x) = \overline{sg}\, f_1(x),$$
$$g(x) = f_1(x) \cdot \dots \cdot f_n(x),$$
$$h(x) = sg(f_1(x) + \dots + f_n(x))$$

будуть характеристичними для доповнення множини A_1, об'єднання та перетину множин A_1, … , A_n. Якщо $f_1(x)$, … , $f_n(x)$ рекурсивні або ПР функції, то такими ж будуть і функції $f(x)$, $g(x)$, $h(x)$.

Інше доведення полягає в побудові алгоритмів обчислення характеристичних функцій, а саме:

Нехай $f_1(x)$, …, $f_n(x)$ – алгоритми для обчислення характеристичних функцій множин A_1,…, A_n. Тоді алгоритм

a)
```
function f_i-bar (x)
    begin
      if f_i(x) = 0 then f_i-bar :=1
      else f_i-bar := 0
    end
```

обчислює характеристичну для $\overline{A_i}$.

b) Алгоритм

```
function f(x)
    begin
      if f_i(x) · ... · f_n(x) = 0 then f := 0
      else f := 1
    end
```

обчислює характеристичну функцію об'єднання множин.

c) Алгоритм

```
function f(x)
    begin
      if (f_1(x) + ... + f_n(x)) = 0 then f := 0
      else f := 1
    end
```

обчислює характеристичну функцію перетину множин.

Приклад 5.1. Нехай, як і раніше, функція $\chi_p(x)$ така, що $\chi_p(x) = 0$ для x простого і $\chi_p(x) = 1$ для x непростого. Ця функція є характеристичною для множини простих чисел і обчислюється наступним алгоритмом:

```
function χ_p(x)
    begin
      if nd(x) = 2 then χ_p := 1
      else χ_p := 0
    end.
```

Тоді доповнення до множини простих чисел є ПР множиною, оскільки характеристична функція доповнення обчислюється наступним алгоритмом:

```
function  χ̄_p(x)
    begin
      if  nd(x) = 2 then χ̄_p := 0
      else χ̄_p := 1
    end.
```

Теорема 5.2. Якщо всюди визначена функція $f(x)$ рекурсивна (ПР), то множина A розв'язків рівняння

$$f(x) = 0$$

рекурсивна (ПР).

Доведення. Характеристична функція множини розв'язків обчислюється наступним алгоритмом:

```
function χ(x)
    begin
      if f(x) = 0 then χ := 0
      else χ := 1
    end.
```

Приклад 5.2. Нехай $div_2(x) = 1$, якщо $rest(x, 2) = 0$, $div_2(x, 2) = 0$, якщо $rest(x, 2) \neq 0$. Функція $div_2(x)$ ПР функція. Вона обчислюється наступним алгоритмом:

```
function  div2(x)
    begin
      if rest(x, 2) = 0 then div2 := 1
      else div2 := 0
    end.
```

Тоді, характеристична функція множини розв'язків рівняння

$$div_2(x) = 0$$

є ПР множиною, оскільки обчислюється наступним алгоритмом:

```
function χ(x)
    begin
      if div2(x) = 0 then χ := 0
      else χ := 1
```

end.

5.2. ПР (Р) образу ПР (Р) функції

Множина значень ПР функції, взагалі кажучи, не буде ПР множиною, ні навіть Р множиною. Але існують умови того, щоб множина значень ПР (Р) функції була ПР (Р) множиною.

Теорема 5.3. Якщо ПР функція $f(x)$ задовольняє умові $f(x) \geq x$ ($x = 0, 1, 2, \ldots$), зокрема, якщо $f(x)$ монотонно зростає, то множина M всіх значень цієї функції є ПР множиною.

Доведення. $M = \{f(0), f(1), \ldots\}$. Алгоритм обчислення характеристичної функції цієї множини наступний:

```
function χ_M (x)
    begin
      s := 0
      for i = 0 to x
        if f(i) = x then s := 1
      if s = 1then χ_M := 0
      else χ_M := 1
    end.
```

Якщо функція монотонно зростає, тобто правдива імплікація

$$x < y \Rightarrow f(x) < f(y),$$

то $f(x) \geq x$ для будь-якого x. Дійсно, за методом математичної індукції маємо:

1. $f(0) \geq 0$;
2. Нехай $f(x) \geq x$;
3. Тоді $f(x + 1) > f(x) \geq x$. Отже, $f(x + 1) \geq x + 1$.

У випадку правдивості імплікації

$$x < y \Rightarrow f(x) \leq f(y),$$

множина M всіх значень цієї функції буде ПР множиною. Дійсно, в цьому випадку слід розглянути функцію $g(x) = f(x) + x$. Ця функція примітивно рекурсивна і задовольняє співвідношенню

$$x < y \Rightarrow g(x) < g(y),$$

оскільки з того, що x < y випливає, що $g(x) = f(x) + x < f(y) + y = g(y)$. Тому множина всіх значень цієї функції є ПР множиною. Якщо через χ позначити характеристичну функцію цієї множини, то характеристична функція χ_M множини M може бути обчислена алгоритмом

```
function χ_M (x)
    begin
      s := 0
      for i = 0 to x
          if g(i) ÷ i = x then s := 1
      if s = 1 then χ_M := 0
      else χ_M := 1
    end.
```

Приклад 5.3. Розглянемо функцію $f(x) = x^2$. Множина значень цієї M функції є ПР множиною. Характеристична функція цієї множини обчислюється наступним алгоритмом:

```
function χ_M (x)
    begin
      s := 0
      for i = 0 to x
          if i^2 = x then s := 1
      if s = 1 then χ_M := 0
      else χ_M := 1
    end.
```

6

Рекурсивно перелічимі множини

Множина чисел A називається рекурсивно перелічимою (РПМ), якщо існує двомісна ПР функція $f(a, x)$ така, що рівняння $f(a, x) = 0$ має розв'язок x тоді і тільки тоді, коли $a \in A$.

Наслідок. Множина чисел A рекурсивно перелічима тоді і тільки тоді, коли існує алгоритм, який для довільного $n \in N$ дає відповідь на питання $n \in A$, якщо n дійсно належить A, або працює нескінченно довго, якщо $n \notin A$.

6.1. Властивості ПР (Р) множин

Теорема 6.1. Кожна ПР множина є рекурсивно перелічимою.

Доведення. Існує алгоритм

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

який обчислює часткову характеристичну функцію g ПР множини, де f – характеристична функція множини.

Теорема 6.2. Нехай $F(a, x)$ – ПР функція від змінних a, x. Множина M тих значень параметра a, для яких рівняння

$$F(a, x) = 0$$

має хоча б один розв'язок x є РП множиною.

Доведення. Часткова характеристична функція множини M обчислюється наступним алгоритмом:

```
function χ_M (x)
    begin
      i := 0
      while F(x, i) ≠ 0
            do i := i + 1
      χ_M := 0
  end.
```

Теорема 6.3. Непуста множина A РП множина $\Leftrightarrow$ коли вона співпадає з множиною значень деякої ПРФ.

Необхідність (множина A значень ПРФ $f(x)$ є РПМ). Дійсно, часткова характеристична функція множини A може бути обчислена алгоритмом:

```
function χ_A(x)
    begin
       i := 0
       while f(i) ≠ x
            do i := i + 1
       χ_A := 0
    end.
```

Достатність (якщо множина A – РПМ, то співпадає з множиною значень деякої ПРФ). Розглянемо функцію, яка обчислюється алгоритмом

```
function f(n)
    begin
       if F(l(n), r(n)) = 0 then f := l(n)
       else f := b
    end,
```

де $F(a, x)$ ПРФ така, що рівняння $F(a, x) = 0$ має розв'язок $\Leftrightarrow a \in A, b \in A$.

Ця функція ПРФ за побудовою. Крім того:

a) Значення цієї функції належать до A;

b) Якщо m – довільний елемент множини A, то рівняння $F(m, x) = 0$ має розв'язок i. Покладемо $n = \boldsymbol{c}(m, i)$. Тоді значення функції f в точці n дорівнює m.

Теорема 6.4. Сума та перетин скінченної кількості РПМ є РПМ.

Доведення. Нехай A_i – РПМ, а f_i – функції такі, що рівняння $f_i(a, x) = 0$ має розв'язок $\Leftrightarrow a \in A_i$. Тоді часткові характеристичні функції суми та перетину обчислюються алгоритмами:

a) function $\chi_{\cup}(x)$

begin
$i := 0$
while $f_1(x, i) \cdot \ldots \cdot f_n(x, i) \neq 0$
do $i := i + 1$
$\chi_{\cup} := 0$
end.

b) function $\chi_{\cap}(x)$
begin
$i := 0$
while $f_1(x, i) + \ldots + f_n(x, i) \neq 0$
do $i := i + 1$
$\chi_{\cap} := 0$
end.

6.2. Теорема Поста

Теорема 6.5 (Поста). Якщо множина A і її доповнення A' рекурсивно перелічимі, то A і A' рекурсивні.

Доведення. Розглянемо алгоритм обчислення функції $h(n)$:

function $h(n)$
begin
$i := 0$
while $|f(i) - n| \cdot |f'(i) - n| \neq 0$
do $i := i + 1$
$h := i$
end.

Тоді характеристичні функції множин A і A' обчислюються алгоритмами:

a) function $\chi_A(n)$
begin
if $|f(h(n)) - n| = 0$ then $\chi_A := 0$
else $\chi_A := 1$
end.

b) function $\chi_{A'}(n)$
begin
if $|f'(h(n)) - n| = 0$ then $\chi_{A'} := 0$
else $\chi_{A'} := 1$
end,

де f, f' – ПРФ з множинами значень A і A' відповідно.

Таким чином, доповнення РП множини, яка не є рекурсивною, не може бути РП множиною.

Теорема 6.6. Сукупність значень M ПР функції $F(x_1, \dots, x_n)$ є РП множиною.

Доведення. Часткова характеристична функція множини M може бути обчислена алгоритмом:

```
function χ_M (x)
    begin
        i := 0
        while F(c_n1(i), … , c_nn (i)) ≠ x
              do i := i + 1
        χ_M := 0
    end.
```

Приклад 6.1. Покажемо, що множина значень M функції $f(x, y) = x + y$ є РП множиною. Дійсно, часткова характеристична функція множини M може бути обчислена алгоритмом

```
function χ_M (x)
    begin
        i := 0
        while f(l(i), r(i)) ≠ x
              do i := i + 1
        χ_M := 0
    end.
```

Приклад 6.2. Предикат $P(x_1, \dots, x_n)$ будемо називати ПР предикатом, якщо він є ПР функцією. Покажемо, що предикат

$$P(x) = \exists n\,(x = 1 + 2 + \dots + n)$$

є ПР предикатом.

Дійсно, він обчислюється алгоритмом

```
function P(x)
    begin
        if x = 0 then P := 1
        else
            if x = 1 then P := 0
              else
```

```
                    begin
                      s := 1
                      for i = 2 to x
                          if s ≠ x then
                              begin
                            s := s + i
                            P := 1
                          end
                    else P := 0
                end
            end.
```

Отже, $P(x)$ – ПР предикат.

Приклад 6.3. В мові запису алгоритмів зустрічаються різні предикати. Оскільки предикати є функціями, то важливо, щоб ці функції-предикати були ПР, Р або ЧР функціями в залежності від того, алгоритм обчислення якої функції ми будуємо. Покажемо, що предикати

$$P(x) = \text{“}x = n\text{”},\ P(x, y) = \text{“}x = y\text{”},\ Q(x, y) = \text{“}x \le y\text{”},\ R(x, y) = \text{“}x < y\text{”}$$

є примітивно рекурсивними. Дійсно, ці предикати обчислюються алгоритмами:

```
function P(x)
    begin
        if |x – n| = 0 then P := 0
        else P := 1
    end,
```

```
function P(x, y)
    begin
        if |x – y| = 0 then P := 0
        else P := 1
    end,
```

```
function Q(x, y)
    begin
        if x ∸ y = 0 then P := 0
        else P := 1
    end,
```

```
function R(x, y)
    begin
        if sg̅(y ∸ x) = 0 then P := 0
        else P := 1
    end.
```

Приклад 6.4. Покажемо, що повний прообраз рекурсивної множини A відносно рекурсивної функції f є рекурсивною множиною. Дійсно, характеристична функція χ прообразу рекурсивної множини A обчислюється наступним алгоритмом

```
function χ(x)
    begin
```

if $\chi_A(f(x)) = 0$ then $\chi := 0$
else $\chi := 1$
end.

7

Множини *n*-ок натуральних чисел

Множина A n-ок натуральних чисел називається ПР, Р або РП, якщо такою є множина $\boldsymbol{c}(A)$ номерів всіх n-ок із A.

7.1. Властивості множин *n*–ок натуральних чисел

Теорема 7.1. Якщо функція $f(x_1, \dots, x_n)$ рекурсивна (примітивно рекурсивна), то множина M розв'язків рівняння

$$f(x_1, \dots, x_n) = 0$$

є рекурсивною (примітивно рекурсивною) множиною.

Доведення. Алгоритм для обчислення характеристичної функції множини M наступний:

```
function χ_M (x_1, … , x_n)
    begin
        if f(x_1, … , x_n) = 0 then χ_M := 0
        else χ_M := 1
    end.
```

Приклад 7.1. У вище наведеному алгоритмі обчислюється характеристична функція множини M n-ок чисел. Зрозуміло, що тоді функція

$$\chi_{\boldsymbol{c}(M)}(x) = \chi_M(\boldsymbol{c}_{n1}(x), \dots, \boldsymbol{c}_{nn}(x))$$

буде характеристичною для множини $\boldsymbol{c}(M)$ канторових номерів цих n-ок чисел. Навпаки, якщо функція $\chi(x)$ є характеристичною для множини чисел M, то функція $\chi^n(x_1, \dots, x_n) = \chi(\boldsymbol{c}(x_1, \dots, x_n))$ буде характеристичною для прообразу цієї множини відносно канторової нумераційної функції.

Звідси, зокрема, випливає, що

а) множина n-ок чисел M примітивно рекурсивна тоді і тільки тоді, коли множина $\boldsymbol{c}(M)$ примітивно рекурсивна;

в) множина n-ок чисел M рекурсивна тоді і тільки тоді, коли множина $\boldsymbol{c}(M)$ рекурсивна;

с) множина n-ок чисел M рекурсивно перелічима тоді і тільки тоді, коли множина $\boldsymbol{c}(M)$ рекурсивно перелічима.

Теорема 7.2. Для того, щоб непуста сукупність n-ок була РП необхідно і достатньо, щоб вона була сукупністю всіх n-ок виду

$$< f_1(x), \dots , f_n(x)>, x = 0, 1, \dots .$$

де $f_1(x), \dots , f_n(x)$ – деякі ПРФ.

Достатність (множина всіх n-ок виду $<f_1(x), \dots , f_n(x)>$, $x = 0, 1, \dots$, де f_i – ПРФ є РПМ множиною). Часткова характеристична функція χ_M множини M канторових номерів n-ок виду $<f_1(x), \dots , f_n(x)>$, $x = 0, 1, \dots$, може бути обчислена алгоритмом:

```
function χ_M (x)
    begin
      i := 0
      while c(f_1(i), … , f_n (i)) ≠ x
            do i := i + 1
       χ_M := 0
    end.
```

Необхідність (множина n-ок M – РПМ $\Rightarrow$ співпадає з множиною всіх n-ок виду $< f_1(x), \dots , f_n (x)>$, $x = 0,1, \dots$, де f_i – ПРФ). Множина A канторових номерів n-ок з M співпадає з множиною значень функції

```
function f(x)
    begin
       if F(l(x), r(x)) = 0 then f := l(x)
       else f := b
    end,
```

де $F(a, x)$ така, що рівняння $F(a, x) = 0$ має розв'язок $\Leftrightarrow a \in A$, а $b \in A$. Тому, множина M співпадає з множиною

$$<\boldsymbol{c}_{n1}(f(x)), \dots , \boldsymbol{c}_{nn}(f(x))>, x = 0, 1, \dots .$$

Приклад 7.2. Покажемо, що кожна нескінченна РП множина M містить нескінченну рекурсивну підмножину. Дійсно, нехай множина M співпадає з множиною значень ПР функції $f(x)$. Розглянемо множину, характеристична функція якої обчислюється алгоритмом

```
function χ_M (x)
    begin
      s := 0
      for i = 0 to x
        if g(i) = x then s := 1
      if s = 1then χ_M := 0
      else χ_M := 1
    end,
```

де функція $g(x)$ обчислюється алгоритмом.

```
function g(x)
    begin
      s := 1
      i := 0
      j := 0
      if x = 0 then g := f(0)
      else
        begin
          while s ≠ x do
            begin
              while f(i) ≥ f(j) do
                j := j + 1
            end
          i := j
          s := s + 1
        end
      g := f(j)
    end.
```

Ця функція буде характеристичною для нескінченної рекурсивної підмножини множини M.

Зазначимо, що тут було використане твердження про те, що множина М рекурсивна тоді і тільки тоді, коли вона співпадає з множиною значень строго зростаючої рекурсивної функції.

Графіком функції $F(x_1, \dots, x_n)$ називається сукупність $(n + 1)$-ок виду $<x_1, \dots, x_n, F(x_1, \dots, x_n)>$.

Теорема 7.3. Якщо графік G_f всюди визначеної функції $f(x_1, \dots, x_n)$ є РПМ, то функція f рекурсивна.

Доведення. Графік G_f – це сукупність $(n + 1)$-ок виду:

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

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

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

отже, функція f – рекурсивна.

Приклад 7.3. Покажемо, що графік ПР функції $f(x)$ є ПР множиною. Дійсно, характеристична функція графіка обчислюється алгоритмом

function $\chi(x, y)$
  begin
    if $y = f(x)$ then $\chi := 0$
    else $\chi := 1$
  end.

Теорема 7.4. Нехай функція $f(x_1, \dots, x_n, y)$ обчислюється алгоритмом:

function $f(x_1, \dots, x_n, y)$
  begin
    if $y = 0$ then $f := g(x_1, \dots, x_n)$
    else $f := h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y - 1))$
  end.

Тоді функцію $f(x_1, \dots, x_n, y)$ можна обчислити алгоритмом:

function $f(x_1, \dots, x_n, y)$
  begin
    $f := \boldsymbol{c}_{n+2n+2}(\varphi(G(\boldsymbol{c}^n(x_1, \dots, x_n)), y))$
  end,

де функція $\varphi(x, y)$ обчислюється алгоритмом:

function $\varphi(x, y)$
  begin
    if $y = 0$ then $\varphi := x$
    else $\varphi := \Phi(\varphi(x, y - 1))$

end,

а

$$G(x) = \boldsymbol{c}^{n+2}(\boldsymbol{c}_{n1}(x), \dots, \boldsymbol{c}_{nn}(x), 0, g(\boldsymbol{c}_{n1}(x), \dots, \boldsymbol{c}_{nn}(x))),$$
$$\Phi(u) = \boldsymbol{c}^{n+2}(\boldsymbol{c}_{n+21}(u), \dots, \boldsymbol{c}_{n+2n+1}(u) + 1, h(\boldsymbol{c}_{n+21}(u), \dots, \boldsymbol{c}_{n+2n+2}(u))).$$

Доведення. Введемо функцію

$$\psi(x, y) = \boldsymbol{c}^{n+2}(\boldsymbol{c}_{n1}(x), \dots, \boldsymbol{c}_{nn}(x), y, f(\boldsymbol{c}_{n1}(x), \dots, \boldsymbol{c}_{nn}(x), y)).$$

Тому функцію $f(x_1, \dots, x_n, y)$ можна обчислити алгоритмом

function $f(x_1, \dots, x_n, y)$
begin
$f := \boldsymbol{c}_{n+2n+2}(\ \psi(\boldsymbol{c}^n(x_1, \dots, x_n), y))$
end,

де функція $\psi(x, y)$ обчислюється алгоритмом

function $\psi(x, y)$
begin
if $y := 0$ then $\psi = G(x)$
else $\psi := \Phi(\psi(x, y - 1))$
end,

а

$$G(x) = \boldsymbol{c}^{n+2}(\boldsymbol{c}_{n1}(x), \dots, \boldsymbol{c}_{nn}(x), 0, g(\boldsymbol{c}_{n1}(x), \dots, \boldsymbol{c}_{nn}(x))),$$

$$\Phi(u) = \boldsymbol{c}^{n+2}(\boldsymbol{c}_{n+21}(u), \dots, \boldsymbol{c}_{n+2n+1}(u) + 1, h(\boldsymbol{c}_{n+21}(u), \dots, \boldsymbol{c}_{n+2n+2}(u))).$$

Якщо ввести функцію $\varphi(x, y)$, яка обчислюється алгоритмом

function $\varphi(x, y)$
begin
if $y = 0$ then $\varphi := x$
else $\varphi := \Phi(\varphi(x, y - 1))$
end,

то алгоритм обчислення функції $f(x_1, \dots, x_n, y)$ буде наступним:

function $f(x_1, \dots, x_n, y)$
begin
$f := \boldsymbol{c}_{n+2n+2}(\ \varphi(G(\boldsymbol{c}^n(x_1, \dots, x_n)), y))$
end.

7.2. Допустимі функції

Клас ПРФ визначається наступним чином. Задається клас найпростіших ПРФ $o(x) = 0$, $s(x) = x + 1$, $I_m^n(x_1, \dots, x_n) = x_m$, де $n \geq m \geq 1$. Будь-яка функція, яка обчислюється алгоритмом

a) function $f(x_1, \dots, x_n)$
begin
$f := h(g_1(x_1, \dots, x_m), \dots, g_n(x_1, \dots, x_m))$
end

або алгоритмом

b) function $f(x_1, \dots, x_n, y)$
begin
if $y := 0$ then $f = g(x_1, \dots, x_n)$
else $f := h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y - 1))$
end,

де g_i, h – ПРФ, називається ПРФ.

Виявляється, що клас ПРФ можна визначити іншим способом. А саме, функції $s(x) = x + 1$, $q(x) = x \dot{-} [\sqrt{x}]$ будемо називати найпростішими допустимими функціями. Будь-яка одномісна функція, яка обчислюється алгоритмом

c) function $f(x)$
begin
$f := h(x) + g(x)$
end,

або алгоритмом

d) function $f(x)$
begin
$f := h(g(x))$
end,

або алгоритмом

e) function $f(x)$
begin
if $x = 0$ then $f := 0$
else $f := h(f(x - 1))$
end,

де g, h – допустимі функції, називається допустимою функцією (ДФ).

Будь-яка n-місна функція $f(x_1, \dots, x_n)$ – ДФ, якщо одномісна функція $f(\alpha_1(t), \dots, \alpha_n(t))$ може бути обчислена алгоритмами c) – e), де $\alpha_i(t)$ довільні ДФ.

Теорема 7.5. Клас ПРФ співпадає з класом ДФ.

Доведення. Покажемо, що кожна ДФ є ПРФ. Дійсно, функції $s(x)$ і $q(x)$ є ПРФ. Якщо $h(x)$ і $g(x)$ – ПРФ, то функції, які обчислюються алгоритмами c), d) – ПРФ. Функція, яка обчислюється алгоритмом e) може бути обчислена таким алгоритмом

```
a)  function f(x)
        begin
            if x = 0 then f := o(x)
            else f := I_2^2 (x, h(f(x – 1)))
        end,
```

отже, є ПРФ.

ДФ $f(x_1, \dots, x_n)$ може бути обчислена алгоритмом

```
function f(x_1, … , x_n)
    begin
        f := g(c(x_1, … , x_n))
    end,
```

де функція g обчислюється алгоритмом

```
function g(y)
    begin
        g := f(c_n1(y), … , c_nn(y))
    end
```

(g – одномісна ДФ, тому g – ПРФ). Отже, $f(x_1, \dots, x_n)$ – ПРФ.

Покажемо, що кожна ПРФ є ДФ. Для цього треба показати, що

1. Функції $o(x)$, $s(x)$, $I_m^n(x_1, \dots, x_n)$ є ДФ.

2. Функції, які обчислюються алгоритмами a) – b) можуть бути обчислені алгоритмами c) – e).

Нехай функції $h, g_1, \dots, g_n$ – ДФ, а функція f обчислюється алгоритмом a). Треба показати, що одномісна функція $f(\alpha_1(t), \dots, \alpha_m(t))$ може бути обчислена алгоритмами c) – e), де $\alpha_i(t)$ довільні ДФ. Дійсно, функції $\varphi_i(t) = g_i(\alpha_1(t), \dots, \alpha_m(t))$ обчислюються алгоритмами c) – e). Тому $f(\alpha_1(t), \dots, \alpha_m(t)) = h(\varphi_1(t), \dots, \varphi_n(t))$ обчислюється алгоритмами c) – e).

Нехай функція f задається рекурсивною схемою b) і функції h і g – ДФ. Треба показати, що функція f – ДФ, тобто функція $f(\alpha_1(t), \dots, \alpha_m(t), \alpha(t))$ може бути обчислена алгоритмами c) – e), де $\alpha_i(t)$, $\alpha(t)$ – довільні ДФ.

Дійсно, функцію $f(x_1, \dots, x_n, y)$ можна обчислити алгоритмом:

function $f(x_1, \dots, x_n, y)$
begin
$f := \boldsymbol{c}_{n+2n+2}(\varphi(G(\boldsymbol{c}^n(x_1, \dots, x_n)), y))$
end,

де функція $\varphi(x, y)$ обчислюється алгоритмом:

function $\varphi(x, y)$
begin
if $y = 0$ then $\varphi := x$
else $\varphi := \Phi(\varphi(x, y - 1))$
end.

Функцію $\varphi(x, y)$ можна обчислити алгоритмом:

function $\varphi(x, y)$
begin
$\varphi := \theta(w(y, x))$
end,

де функція $\theta(x)$ обчислюється алгоритмом:

function $\theta(x)$
begin
if $x = 0$ then $\theta := 0$
else $\theta := \Theta(x - 1, \theta(x - 1))$
end,

а $w(x, y) = ((x + y)^2 + y^2) + x$.

В свою чергу, функцію $\theta(x)$ можна обчислити алгоритмом:

function $\theta(x)$
begin
$\theta := \boldsymbol{r}(\psi(x))$
end,

де функція $\psi(x)$ обчислюється алгоритмом:

function $\psi(x)$
begin
if $x = 0$ then $\psi := 0$

else $\psi := \Phi(\psi(x))$
end,

а $\Phi(u) = \boldsymbol{c}(\boldsymbol{l}(u)) + 1$, $\Theta(\boldsymbol{l}(u), \boldsymbol{r}(u)))$.

Наслідок (Торема Робінсон)**.** Клас одномісних допустимих функцій співпадає з класом одномісних ПРФ.

8

Частково рекурсивні функції

8.1. Теорема про графік ЧРФ

Теорема 8.1 (про графік частково рекурсивної функції). Для того, щоб часткова функція f була частково рекурсивною, необхідно і достатньо, щоб графік f був рекурсивно перелічимим.

Достатність (графік G_f функції f є РПМ $\Rightarrow$ f є ЧРФ). За попередньою теоремою множина G_f співпадає з множиною пар виду $< f_1(x),\ f_2(x) >$, $x = 0,1, \ldots$.

Тоді для обчислення функції f існує алгоритм

```
function f (x)
    begin
        i := 0
        while f1(i) ≠ x
              do i := i + 1
         f := f2 (i)
    end.
```

Цей алгоритм або обчислює значення $f(x)$, або працює нескінченно довго у випадку, якщо $f(x)$ невизначена.

Необхідність (f є ЧРФ $\Rightarrow$ графік G_f функції f є РПМ). Часткова характеристична функція множини G_f може бути обчислена алгоритмом:

```
function χ_{G_f} (x, y)
    begin
        i := 0
        while i ≠ x
            do i := i + 1
         while f(x) ≠ y
            do i := i + 1
         χ_{G_f} := 0
    end.
```

Наслідок 8.1. Область визначення кожної ЧРФ є РПМ.

Доведення. Графік ЧРФ можна подати у вигляді

$$<u_1(t), \dots, u_{n+1}(t)>.$$

Тоді алгоритм обчислення часткової характеристичної функції області визначення є таким:

```
function χ(x_1, … , x_n)
    begin
      i := 0
      while x_1 ≠ u_1(i) ∨ … ∨ x_n ≠ u_n (i)
            do i := i + 1
      χ := 0
    end.
```

Наслідок 8.2. Область значень ЧРФ є РПМ.

Доведення. Графік ЧРФ можна подати у вигляді:

$$<u_1(t), \dots, u_{n+1}(t)>.$$

Тоді алгоритм обчислення часткової характеристичної функції множини значень є таким:

```
function χ(x)
    begin
      i := 0
      while x ≠ u_{n+1}(i)
          do i := i + 1
      χ := 0
    end.
```

Наслідок 8.3. Множина A n-ок чисел РП $\Leftrightarrow$ коли часткова характеристична функція множини A є ЧРФ.

Доведення ($\Rightarrow$). Якщо A – РПМ, то $A = <u_1(t), \dots, u_n(t)>$, $t = 0, 1, 2, \dots$ Тоді алгоритм обчислення часткової характеристичної функції χ_A є таким:

```
function χ_A (x_1, ... , x_n)
    begin
      i := 0
      while x_1 ≠ u_1(i) ∨ … ∨ x_n  ≠ u_n(i)
            do i := i + 1
      χ_A := 0
    end.
```

(⇐). Якщо χ_A є ЧРФ (χ_A – часткова характеристична функція множини A), то A співпадає з областю визначення χ_A, а отже, є РПМ (наслідок 1).

Наслідок 8.4. Якщо $F(x,y)$ – ЧРФ, то сукупність A тих x, для яких рівняння $F(x, y) = 0$ має розв'язок y є РПМ.

Доведення. Графік функції $F(x, y)$ можна подати у вигляді

$$<\alpha_1(t), \alpha_2(t), \alpha(t)>, \ t = 0, 1, 2, \ldots$$

Тоді часткова характеристична функція множини A обчислюється алгоритмом:

```
function χA(x)
    begin
        i := 0
        while α(i) ≠ 0 ∨ α1(i) ≠ x
              do i := i + 1
        χA := 0
    end.
```

Наслідок 8.5. Сукупність A розв'язків рівняння

$$f(x_1, \ldots, x_n) = 0,$$

де f – ЧРФ, є РПМ.

Доведення. Графік функції f можна подати у вигляді

$$<\alpha_1(t), \ldots, \alpha_n(t), \alpha(t)>, t = 0, 1, 2, \ldots$$

Тоді часткова характеристична функція множини A обчислюється алгоритмом

```
function χA(x1, ... , xn)
    begin
        i := 0
        while α1(i) ≠ x1 ∨ …∨ αn(i) ≠ xn ∨ α(i) ≠ 0
              do i := i + 1
        χA := 0
    end.
```

8.2. Кускове задання ЧРФ

Теорема 8.2. Якщо функції $f_i(x)$ і $g_i(x)$ ($i = 1, 2, \ldots, n$) ЧРФ, то функція

$$h(x)=\begin{cases} f_1(x), \text{якщо } g_1(x)=0 \\ f_2(x), \text{якщо } g_2(x)=0 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ f_n(x), \text{якщо } g_n(x)=0 \\ \infty, \text{в інших випадках} \end{cases}$$

є ЧРФ.

Доведення. Алгоритм обчислення функції *h* має вигляд:

function $h(x)$
begin
$i := 0$
while $(\beta_1^1(i) \neq x \vee \beta_2^1(i) \neq 0) \wedge$
$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$
$\wedge(\beta_1^n(i) \neq x \vee \beta_2^n(i) \neq 0)$
do $i := i+1$
if $(\beta_1^1(i) = x \wedge \beta_2^1(i) = 0)$ then
$j := 0$
while $\alpha_1^1(j) \neq x$
do $j := j + 1$
$h := \alpha_2^1(j)$
. . . . . .
if $(\beta_1^n(i) = x \wedge \beta_2^n(i) = 0)$ then
$j := 0$
while $\alpha_1^n(j) \neq x$
do $j := j + 1$
$h := \alpha_2^n(j)$
end,

де $<\alpha_1^i(t), \alpha_2^i(t)>$ – графік $f_i(x)$, $<\beta_1^i(t), \beta_2^i(t)>$ – графік $g_i(x)$.

Приклад 8.1. Покажемо, що образ РП множини *M* відносно ЧР функції $f(x)$ є РП множиною. Дійсно, нехай $G_f = <f_1(x), f_2(x)>$ – графік ЧР функції $f(x)$ і множина *M* співпадає з множиною значень функції $\alpha(x)$. Тоді часткова характеристична функція образу множини *M* відносно функції $f(x)$ обчислюється наступним алгоритмом:

function $\chi(x)$
begin

```
        i := 0
        j := 0
        while f2(i) ≠ x do
              i := i + 1
        while α(j) ≠ f1(i) do
              j := j + 1
    χ := 0
end.
```

Приклад 8.2. Покажемо, що множина

$$M = \{<x_1, \dots, x_n>, \exists y\, f(x_1, \dots, x_n, y) = 0\}$$

є РП множиною, де f – ЧР функція. Дійсно, часткова характеристична функція множини M обчислюється наступним алгоритмом:

```
function χM(x1, ... , xn)
    begin
      i := 0
      while f1(i) ≠ x1 ∨ f2(i) ≠ x2 ∨ ... ∨ fn(i) ≠ xn ∨ fn+2(i) ≠ 0
            do i := i + 1
      χM := 0
    end,
```

де $G_f = <f_1(x), f_2(x), \dots, f_{n+2}(x)>$ – графік ЧР функції $f(x_1, \dots, x_n, y)$.

Приклад 8.3. Покажемо, що повний прообраз РП множини M відносно ЧР функції є РП множиною. Дійсно, нехай $G_f = <f_1(x), f_2(x)>$ – графік ЧР функції $f(x)$ і множина M співпадає з множиною значень функції $\alpha(x)$. Тоді часткова характеристична функція прообразу множини M відносно функції $f(x)$ обчислюється наступним алгоритмом:

```
function χ(x)
    begin
      i := 0
      j := 0
      while f1(i) ≠ x do
            i := i + 1
      while α(j) ≠ f2(i) do
            j := j + 1
      χ := 0
    end.
```

Приклад 8.4. Покажемо, що множина A всіх натуральних n для яких існує розв'язок рівняння

$$x^n + y^n = z^n$$

в натуральних числах, відмінних від нуля, є РП множиною.

Дійсно, часткова характеристична функція χ_A множини A обчислюється наступним алгоритмом:

```
function χ(x)
    begin
      i := 1
      while c31(i) + c32(i) ≠ c33(i)  do
              i := i + 1
      χ := 0
    end.
```

9

Рекурсії 2-го рівня

Як уже зазначалося, функція f називається рекурсивною, якщо вона може бути обчислена алгоритмом

function $f(x_1, \dots, x_n, y)$
  begin
    if $y = 0$ then $f := g(x_1, \dots, x_n)$
    else $f := h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y - 1))$
  end,

тобто, значення такої функції обчислюються через значення цієїж функції для менших значень аргументу. Якщо відоме значення функції для $y = 0$, то за допомогою приведеного алгоритму можна обчислити значення функції для будь-якого y. Крім того, при умові, що g, h є ПРФ, функція f буде ПРФ.

Функція f виникає із функції h рекурсією з поверненням, якщо вона може бути обчислена алгоритмом

function $f(x)$
  begin
    if $x = 0$ then $f := b$
    else $f := h(x, f(\alpha(x)))$
  end.

При умові, що h є ПРФ, функція f буде ПРФ, тобто може бути обчислена алгоритмом

function $f(x)$
  begin
    if $x = 0$ then $f := b$
    else $f := G(x, f(x - 1))$
  end.

Значення такої функції обчислюються через значення цієїж функції для менших значень аргументу ($\alpha(x + 1) \le x$).

У кожному з вищеприведених алгоритмів функція обчислюється рекурсією по одному аргументу. Для того, щоб задати функцію рекурсією по двох

аргументах (обчислювати її значення через її ж значення, але для менших значень аргумента), треба упорядкувати множину пар натуральних чисел, наприклад в такий спосіб

$$(0,0) < (0,1) < \ldots < (1,0) < (1,1) < \ldots < (2,0) < (2,1) < \ldots ,$$

а потім задати алгоритм визначення значення функції через значення цієї ж фунції для менших значень аргументів. Наприклад, обчислення значення функції $F(x, y)$ рекурсією по двох аргументах полягає у заданні значеня $F(0, 0)$ та у визначенні $F(x, y)$ через значення цієї функції для менших значень аргументу. Можливі випадки:

a) Для $F(0, x)$ значення з меншим аргументом має вигляд $F(0, y)$, де $y < x$.

b) Для $F(n + 1, 0)$ значення з меншим аргументом має вигляд $F(m, x)$, де $m \le n$, а x довільне.

c) Для $F(n + 1, x + 1)$ значення з меншим аргументом мають вигляд $F(n + 1, x)$, $F(n, g(x))$, $F(n, F(n + 1, x))$, … .

Приклад. Нехай функція $B(n, x)$ визначається схемою

$$B(n + 1, x + 1) = B(n, B(n + 1, x)),$$
$$B(n + 1, 0) = B(n, 1),$$
$$B(0, x) = 1 + x.$$

Ця функція задається рекурсією 2-го рівня, оскільки значення функції $B(n, x)$ обчислюється через значення цієї ж фунції для менших значень аргументів. Зрозуміло, що така функція обчислюється алгоритмом

```
function B(n, x)
    begin
        if n ≥ 1 ∧ x = 0 then B := B(n – 1, 1)
        if n = 0 then B := 1 + x
        if x ≥ 1 ∧ n ≥ 1 then B := B(n – 1, B(n, x – 1))
    end.
```

Якщо всі функції, які зустрічаються в цьому алгоритмі є ПРФ, то функція $B(n, x)$ буде по крайній мірі рекурсивною функцією.

9.1. Функція Акермана

Визначимо послідовність всюди визначених функцій α_0, α_1, … одного аргументу. Покладемо,

$$\alpha_0(x) = x + 1,\ \alpha_i(x) = \alpha_{i-1}^{[x+2]}(x),$$

де позначення $f^{[n]}(x)$ означає $f(f(\ \dots\ f(x)\ \dots\ ))$, причому функція f використовується n разів. За індукцією можна довести наступні властивості функцій α_i:

1) $\alpha_i(x) > x$ для всіх i та x;
2) $\alpha_i(x)$ зростає із зростанням x;
3) $\alpha_i(x)$ зростає із зростанням i (для кожного фіксованого x);
4) $\alpha_i(x) \geq \alpha_{i-1}(\alpha_{i-1}(x))$.

Теорема 9.1. Нехай $f(x_1, x_2, \dots, x_n)$ – ПР функція. Тоді існує k таке, що

$$f(x_1, x_2, \dots, x_n) \leq \alpha_k(\max(x_1, x_2, \dots, x_n))$$

для всіх $x_1, x_2, \dots, x_n$.

Доведення. Для базисних функцій твердження очевидне. Розглянемо операцію суперпозиції. Нехай

$$f(x_1, \dots, x_n) = g(g_1(x_1, \dots, x_n), \dots, g_k(x_1, \dots, x_n))$$

і

$$g_i(x_1, x_2, \dots, x_n) \leq \alpha_N(\max(x_1, x_2, \dots, x_n))$$

для всіх i та x,

$$g(y_1, y_2, \dots, y_k) \leq \alpha_N(\max(y_1, y_2, \dots, y_k)).$$

Тоді

$$g(g_1(x_1, \dots, x_n), \dots, g_k(x_1, \dots, x_n)) \leq \alpha_N(\max(g_1(x_1, \dots, x_n), \dots, g_k(x_1, \dots, x_n))) \leq$$
$$\alpha_N(\alpha_N(x_1, \dots, x_n)) \leq \alpha_{N+1}(x_1, \dots, x_n).$$

Якщо функція $f(x_1, x_2, \dots, x_n)$ визначається за допомогою наступних співвідношень:

$$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n)$$
$$f(x_1, \dots, x_n, y + 1) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$$

і

$$g(x_1, \dots, x_n) \leq \alpha_N(\max(x_1, x_2, \dots, x_n)),$$
$$h(x_1, \dots, x_n, y, z) \leq \alpha_N(\max(x_1, \dots, x_n, y, z)),$$

то

$$f(x_1, \dots, x_n, 1) = h(x_1, \dots, x_n, 0, f(x_1, \dots, x_n, 0)) \leq$$

$$\leq \alpha_N(\max(x_1, \dots, x_n, 0, f(x_1, \dots, x_n, 0)) \leq$$
$$\leq \alpha_N(\max(x_1, \dots, x_n, 0, \alpha_N(\max(x_1, x_2, \dots, x_n)))) \leq \alpha_N(\alpha_N(\max(x_1, \dots, x_n))).$$

Аналогічно,

$$f(x_1, \dots, x_n, 2) = h(x_1, \dots, x_n, 1, f(x_1, \dots, x_n, 1)) \leq$$
$$\leq \alpha_N(\max(x_1, \dots, x_n, 1, f(x_1, \dots, x_n, 1)) \leq$$
$$\leq \alpha_N(\max(x_1, \dots, x_n, 1, \alpha_N(\alpha_N(\max(x_1, x_2, \dots, x_n))))) \leq$$
$$\alpha_N(\alpha_N(\alpha_N(\max(x_1, \dots, x_n)))).$$

І взагалі

$$f(x_1, \dots, x_n, i) \leq \alpha_N^{[i+1]}(\max(x_1, x_2, \dots, x_n)) \leq \alpha_{N+1}(\max(i, \max(x_1, x_2, \dots, x_n))).$$

Наслідок. Функція Акермана $A(n) = \alpha_n(n)$ росте швидше ніж будь-яка ПР функція.

10

Універсальні функції

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:

1. Для кожного фіксованого i функція $F(i, y)$ належить $\mathfrak{I}$;

2. Для кожної функції $f(y)$ із $\mathfrak{I}$ існує таке число i, що для всіх y $F(i, y) = f(y)$.

10.1. Універсальна рекурсивна функція

Теорема 10.1. Система всіх одномісних ПР функцій має універсальну рекурсивну функцію.

Така функція позначається через $D(n, x)$ і має наступні властивості:

1. Для кожного фіксованого одномісна функція $D(n, x)$ є ПР функцією.

2. Для кожної одномісної ПР функції $f(x)$ існує число n таке, що $D(n, x) = f(x)$.

Доведення. На основі теореми Робінсон всі одномісні ПРФ можна одержати із функцій $s(x) = x + 1$ і $q(x) = x \dot{-} [\sqrt{x}]$ за допомогою операцій додавання, суперпозиції та ітерування. Тому, всі одномісні ПРФ можна записати наступним чином:

$$s(x),\ q(x),$$
$$s(x) + q(x),\ s(q(x)),\ q(s(x)),\ s(x)^*,\ q(x)^*,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

Тобто, до першого рівня допустимих одномісних функцій належать найпростіші допустимі функції. До другого рівня – одномісні функції, які можна одержати із функцій першого рівня за допомогою операцій додавання, суперпозиції та ітерування. До третього рівня належать всі одномісні функції, які можна одержати із функцій перших двох рівнів за допомогою операцій додавання, суперпозиції та ітерування. І так далі.

Кожна одномісна ПРФ буде в цьому переліку допустимих функцій. Якщо покласти,

$$N(s(x)) = 0,\ N(q(x)) = 1,\ N(s(x) + q(x)) = 3,\ \ldots\ ,$$

то, таким чином, можна занумерувати всі функції цієї послідовності, а, отже, всі ПРФ.

Визначимо функцію

$$F(n, x) = f_n(x),$$

де $f_n(x)$ – ПРФ з номером n ($N(f_n(x)) = n$). Ця функція буде універсальною для класу одномісних ПРФ.

Отже, така нумерація всіх ПРФ дозволяє стверджувати, що універсальна функція для класу одномісних ПРФ існує. Разом з тим, нічого не можна сказати про алгоритмічні властивості цієї функції (примітивна рекурсивність, рекурсивність, часткова рекурсивність).

Розглянемо іншу нумерацію одномісних ПРФ. Покладемо за визначенням

$$N(s(x)) = 1,\ N(q(x)) = 3.$$

Далі номери функцій визначаємо індуктивно. А саме, якщо $N(\alpha(x)) = a$, $N(\beta(x)) = b$, то

$$N(\alpha(x) + \beta(x)) = 2\cdot 3^a\cdot 5^b,$$
$$N(\alpha(x) \cdot \beta(x)) = 4\cdot 3^a\cdot 5^b,$$
$$N(\alpha(x)^*) = 8\cdot 3^a.$$

Визначимо функцію

$$F(n, x) = f_n(x),$$

де $f_n(x)$ – ПРФ з номером n ($N(f_n(x)) = n$). В цьому випадку не всі натуральні числа є номерами ПРФ. Це означає, що функція $F(n, x)$ визначена не для всіх значень n, а, отже, є частковою. Але для тих значень n, для яких вона визначена справедливі наступні співвідношення:

$$F(n,x) = \begin{cases} f_a(x) + f_b(x), & \text{якщо } n = 2\cdot 3^a\cdot 5^b; \\ f_a(f_b(x)), & \text{якщо } n = 4\cdot 3^a\cdot 5^b; \\ f_a(f_n(x-1)), & \text{якщо } n = 8\cdot 3^a,\ x > 0; \\ 0, & \text{якщо } n = 8\cdot 3^a,\ x = 0; \\ q(x), & \text{якщо } n = 3; \\ s(x), & \text{якщо } n = 1. \end{cases}$$

Ці співвідношення можна переписати у вигляді (враховуючи визначення функції $F(n, x)$)

$$F(n,x)=\begin{cases} F(ex_1 n,x)+F(ex_2 n,x), & \text{якщо } ex_0 n=1;\\ F(ex_1 n,\ F(ex_2 n,x)), & \text{якщо } ex_0 n=2;\\ F(ex_1 n,\ F(n,x-1)), & \text{якщо } ex_0 n=3,\ x>0;\\ 0, & \text{якщо } ex_0 n=3,\ x=0;\\ Q(n,x), & \text{в інших випадках,} \end{cases}$$

де

$$Q(n,x)=s(x)\overline{\text{sg}}|n-1|+q(x)\overline{\text{sg}}|n-3|.$$

Введемо функцію

$$D(n+1,x+1)=\begin{cases} D(ex_1(n+1),x+1)+D(ex_2(n+1),x+1), & \text{якщо } ex_0(n+1)=1;\\ D(ex_1(n+1),\ D(ex_2(n+1),x+1)), & \text{якщо } ex_0 n=2;\\ D(ex_1(n+1),\ D(n+1,x)), & \text{якщо } ex_0(n+1)=3;\\ Q(n+1,x+1), & \text{в інших випадках,} \end{cases}$$

$$D(n+1,0)=\begin{cases} D(ex_1(n+1),0)+D(ex_2(n+1),0), & \text{якщо } ex_0(n+1)=1;\\ D(ex_1(n+1),\ D(ex_2(n+1),0)), & \text{якщо } ex_0 n=2;\\ 0, & \text{в інших випадках,} \end{cases}$$

$D(0, x) = 0$.

Ця функція всюди визначена і її значення дорівнюють значенням функції $F(n, x)$ в точках, де вона визначена. Крім того, ця функція є рекурсивною (визначається схемою рекурсії 2-го рівня) і обчислюється алгоритмом

function $D(x, y)$
begin
if $n\ge 1\wedge x\ge 1\wedge ex_0(n+1)=1$ then $D:=D(ex_1(n+1),x+1)+D(ex_2(n+1),x+1)$
else if $n\ge 1\wedge x\ge 1\wedge ex_0 n=2$ then $D:=D(ex_1(n+1),D(ex_2(n+1),x+1))$
else if $n\ge 1\wedge x\ge 1\wedge ex_0(n+1)=3$ then $D:=D(ex_1(n+1),D(n+1,x))$
else if $n\ge 1\wedge x\ge 1$ then $Q(n+1,x+1)$
if $n\ge 1\wedge x=0\wedge ex_0(n+1)=1$ then $D:=D(ex_1(n+1),0)+D(ex_2(n+1),0)$
else if $n\ge 1\wedge x=0\wedge ex_0 n=2$ then $D:=D(ex_1(n+1),D(ex_2(n+1),0))$
else if $n\ge 1\wedge x=0$ then $D:=0$
if $n\ge 0$ then $D:=0$
end.

Наслідок. Функція $D^{n+1}(x_0, x_1, \dots, x_n) = D(x_0, \boldsymbol{c}(x_1, \dots, x_n))$ є рекурсивною функцією універсальною для класу n-місних ПРФ .

Зауваження. Зрозуміло, що якщо функція $D(n, x)$ є універсальною для класу одномісних ПР функцій, то функція $d(n) = D(n, n) + 1$ буде РФ функцією, відмінною від усіх ПР функцій.

10.2. Універсальна ЧРФ

Теорема 10.2. Кожна ЧРФ $f(x_1, \dots, x_n)$ може бути обчислена алгоритмом:

```
function f (x1, ... , xn)
    begin
        i := 0
        while F(x1, ... , xn, i) ≠ 0 do
            i := i +1
        f := G(i)
    end,
```

де F, G деякі (залежні від f) ПРФ.

Доведення. Якщо f – ЧРФ, то її графік G_f є РПМ. Тому $G_f = \langle f_1(x), \dots, f_n(x), f_{n+1}(x)\rangle$, $x = 0, 1, 2, \dots$, а f_i – ПРФ. Покладемо

$$F(x_1, \dots, x_n, i) = |f_1(i) - x_1| + \dots + |f_n(i) - x_n)|, \text{ а}$$
$$G(x) = f_{n+1}(x).$$

Якщо f в точці $\langle x_1, \dots, x_n\rangle$ визначена, то ця точуа належить G_f, а, отже, існує i таке, що $F(x_1, \dots, x_n, i) = 0$. Значення функції f в цьому випадку дорівнює $f_{n+1}(i)$. Якщо f в точці $\langle x_1, \dots, x_n\rangle$ не визначена, то алгоритм працює нескінченно довго.

Зауваження. Теорему 1 можна сформулювати і так: кожна ЧРФ $f(x_1, \dots, x_n\rangle$ може бути представлена у формі

$$f(x_1, \dots, x_n) = G(\mu_i(F(x_1, \dots, x_n, i) = 0)).$$

Теорема 10.3. Кожна ЧРФ $f(x)$ може бути обчислена алгоритмом

```
function f(x)
    begin
        i := 0
        while F(x, i) ≠ 0 do
```

```
            i := i + 1
        f := l(i)
    end,
```

де F – деяка (залежна від f) ПРФ.

Доведення. Існує ПРФ $g(a, b, z)$ така, що рівняння $g(a, b, z) = 0$ має розв'язок $\Leftrightarrow \langle a, b \rangle \in G_f$. Графік G_f функції f є РПМ. Тому часткова характеристична функція для G_f обчислюється наступним алгоритмом:

```
function χ_G(x, y)
    begin
        i := 0
        while g(x, y, i) ≠ 0 do
                i := i + 1
        χ_G := 0
    end,
```

Покладемо $F(x, i) = g(x, \boldsymbol{l}(i), \boldsymbol{r}(i))$. Тоді функція f може бути обчислена наступним алгоритмом:

```
function f(x)
    begin
        i := 0
        while F(x, i) ≠ 0 do
                i := i + 1
        f := l(i)
    end.
```

Дійсно, нехай функція f визначена в точці x_0. Тоді пара $\langle x_0, f(x_0) \rangle \in G_f$. Тому рівняння

$$g(x_0, f(x_0), z) = 0$$

має розвязок z_0. Нехай $\boldsymbol{c}(f(x_0), z_0) = i$. Тоді $f(x_0) = \boldsymbol{l}(i)$, $z_0 = \boldsymbol{r}(i)$.

Якою б не була пара $\langle f(x_0), z_0 \rangle$ завжди існує i – номер цієї пари ($\boldsymbol{c}$ – бієкція).

Теорема 10.4. Кожна ЧРФ $f(x)$ може бути обчислена алгоритмом

```
function f(x)
    begin
        i := 0
        while D(n, x, i) ≠ 0 do
                i := i + 1
```

$f := \boldsymbol{l}(i)$
end,

де n – деяке (залежне від f) натуральне число.

Доведення. Випливає із теореми 10.1 (D – універсальна для класу двохмісних ПРФ).

Зауваження. Можна розглянути алгоритм:

```
function f(x)
    begin
        f := T(n, x)
    end
```

```
function T(n, x)
    begin
        i := 0
        while D(n, x, i) ≠ 0 do
                i := i + 1
        T := l(i)
    end.
```

Тоді кожна ЧРФ може бути об числена через функцію $T(n, x)$ для деякого n. Отже, $T(n, x)$ – універсальна.

11

Довизначення функцій

Часткова функція $g(x_1, \ldots, x_s)$ називається розширенням часткової функції $f(x_1, \ldots, x_s)$, якщо в кожній точці, в якій функція f визначена, функція g також визначена і її значення в цій точці дорівнює значенню функції f.

Всюди визначене розширення часткової функції f називається довизначенням f. Зрозуміло, що кожна часткова функція має довизначення. Ситуація змінюється, коли шукають не довільні довизначення, а рекурсивні.

11.1. Нерекурсивні функції

Теорема 11.1. Існує одномісна ЧРФ, яка приймає значення 0,1 і не має рекурсивних довизначень

Доведення. Розглянемо функцію

$$V(x) = \begin{cases} 0, U(x.x) > 0 \\ 1, U(x,x) = 0 \\ \infty, \text{ в інших випадках} \end{cases}$$

де $U(x, y)$ – універсальна для всіх одномісних ЧРФ. Функція $V(x)$ – ЧРФ. Покажемо, що вона не може мати рекурсивних довизначень.

Розглянемо функцію

$$w(x) = \begin{cases} 0, U(x,x) > 0 \\ 1, \text{ в інших випадках} \end{cases}$$

Ця функція є довизначенням $V(x)$. Покажемо, що не існує алгоритму, який її обчислює. Припустимо, що такий алгоритм існує, тобто функція $w(x)$ – рекурсивна. Це означає, що

$$w(x) = U(m, x)$$

для деякого m. Тобто, цю функцію можна обчислити в довільній точці алгоритмом А1:

```
A1:  function w(x)                    function U(n, x)
         begin                            begin
             w := U(m, x)                     i := 0
         end                                  while D(n, x, i) ≠ 0 do
                                                      i := i + 1
                                              U := I(i)
                                              end.
```

Обчислимо значення функції w в точці m. Якщо $w(m) = 0$, то $U(m, m) = 0$ (оскільки $w(m) = U(m, m)$). З іншого боку, якщо $w(m) = 0$, то $U(m, m) > 0$ (визначення функції w). Якщо $w(m) = 1$, то $U(m, m) = 1$ (оскільки $w(m) = U(m, m)$). З іншого боку, якщо $w(m) = 1$, то $U(m, m) = 0$ або $U(m, m)$ не визначена (визначення функції w). В обох випадках одержали суперечність. Отже, алгоритм А1 функцію w не обчислює.

Теорема 11.2. Ніяка ЧРФ $U(x,y)$ універсальна для сукупності всіх одномісних ЧРФ не може мати рекурсивних довизначень.

Доведення. Припустимо, що функція $U(x,y)$ має рекурсивне довизначення $P(x, y)$. Тоді функція

$$w(x) = \begin{cases} 0, & p(x,x) > 0 \\ 1, & \text{в іншому випадку} \end{cases}$$

буде рекурсивним довизначенням функції $V(x)$ з теореми 10.1. Дійсно, якщо $U(x, x) > 0$, то $P(x, x) > 0$, а, отже, $w(x) = V(x) = 0$. Якщо $U(x, x) = 0$, то $P(x, x) = 0$, а, отже, $w(x) = V(x) = 1$. Якщо $U(x, x)$ – не визначена, то $w(x) = 1$ (при $P(x, x) = 0$) і $w(x) = 0$ (при $P(x, x) > 0$). Крім того, $w(x)$ – рекурсивна. Дійсно, цю функцію можна обчислити алгоритмом:

```
function w(x)
    begin
        if P(x, x) > 0 then w := 0
        else w := 1
    end.
```

Теорема 11.3. Існують не рекурсивні РПМ.

Доведення. Нехай

$$V(x) = \begin{cases} 0, U(x,x) > 0 \\ 1, U(x,x) = 0 \\ \infty, \text{в інших випадках.} \end{cases}$$

Ця функція є ЧРФ, приймає значення 0,1 і не має рекурсивних довизначень. Множина розв'язків рівняння $V(x) = 0$ за наслідком 8.5 є РПМ. Припустимо, що ця множина рекурсивна і нехай χ – характеристична функція цієї множини. Тоді існує алгоритм, який обчислює значення цієї функції:

```
function χ(x)                    function U(n, x)
    begin                            begin
        χ := U(m, x)                     i := 0
    end                                  while D(n, x, i) ≠ 0 do
                                             i := i + 1
                                         U := I(i)
                                     end.
```

Оскільки

$$\chi(x) = 0 \Leftrightarrow V(x) = 0 \Leftrightarrow U(x, x) > 0 \Leftrightarrow w(x) = 0,$$

а

$$\chi(x) = 1 \Leftrightarrow V(x) \neq 0 \Leftrightarrow w(x) = 1,$$

то існує алгоритм, який обчислює функцію $w(x)$:

```
function w(x)
    begin
      if χ(x) = 0 then w := 0
      else w := 1
    end,
```

тобто функція w рекурсивна. Але це не так (теорема 10.1).

Зауваження. Множина розв'язків рівняння $V(x) = 0$ не є порожньою. Дійсно, $x + 1 = g(x) = U(n, x)$. Тоді $U(n, n) = n + 1 > 0$. Тому $V(n) = 0$.

Теорема 11.4. Якщо область визначення M ЧРФ $f(x)$ рекурсивна, то $f(x)$ має рекурсивне довизначення.

Доведення. Функція

$$g(x) = \begin{cases} f(x), \text{ якщо } \chi(x) = 0 \\ 0, \text{ якщо } \chi(x) = 1 \end{cases}$$

де $\chi(x)$ – характеристична для M, буде довизначенням функції $f(x)$. Крім того, ця функція алгоритмічно обчислювана, наприклад, таким алгоритмом:

```
function g(x)
    begin
```

```
i := 0
j := 0
while β1(i) ≠ x do  i := i + 1
if β2(i) = 0 then
        begin
            while α1(j) ≠ x do j := j + 1
             g := α2(j)
         end
else g := 0
end,
```

де $<\alpha_1(t), \alpha_2(t)>$ – графік $f(x)$, $<\beta_1(t), \beta_2(t)>$ – графік $\chi(x)$.

Зауваження. Алгоритм не циклить, бо якщо $\beta_2(i) = 0$, то $\beta_1(i) = x \in M$ – область визначення функції f.

Теорема 11.5. Якщо ЧРФ $U(x, y)$ є універсальною для всіх одномісних ЧРФ, то область визначення M цієї функції є нерекурсивною РПМ.

Доведення. Множина M – РПМ, як область визначення ЧРФ. Якби ця множина була рекурсивною то за теоремою 11.4 функція $U(x, y)$ мала б рекурсивне довизначення. На підставі теореми 11.2 такого довизначення функція $U(x, y)$ мати не може.

Зауваження. Питання про те чи існує алгоритм, який для довільної ЧРФ f і довільного x дає відповідь на питання про те, чи f в точці x визначена зводиться до питання – чи є рекурсивною область визначення ЧРФ f. Такого алгоритму не існує. Якби такий алгоритм існував, то існував би і алгоритм, який давав би відповідь на питання про те, чи визначена в довільній точці $<n, x>$ функція $T(n, x)$, де T – універсальна для одномісних ПРФ. Але область визначення універсальної функції нерекурсивна.

12

Універсальні функції Кліні

При вивченні властивостей ЧРФ іноді зручніше користуватися не універсальними функціями $T(x_0, x_1, \ldots, x_n)$, а особливими функціями Кліні.

Введемо функції:

$$K^2(x_0, x_1) = T^2(\boldsymbol{l}(x_0), \boldsymbol{c}(\boldsymbol{r}(x_0), x_1))$$
$$K^3(x_0, x_1, x_2) = K^2([(x_0, x_1], x_2)),$$

де $[x_0, x_1] = \boldsymbol{c}(\boldsymbol{l}(x_0), \boldsymbol{c}(\boldsymbol{r}(x_0), x_1))$ – нумерація пар натуральних чисел.

Лема. Функції $K^2(x_0, x_1)$ і $K^3(x_0, x_1, x_2)$ зв'язані з універсальними функціями $T^3(x_0, x_1, x_2)$ та $T^4(x_0, x_1, x_2, x_3)$ тотожностями:

1) $K^2(\boldsymbol{c}(x_0, x_1), x_2) = T^3(x_0, x_1, x_2)$.
2) $K^3(\boldsymbol{c}(x_0, x_1), x_2, x_3) = T^4(x_0, x_1, x_2, x_3)$.

Доведення. 1) Функція $T^2(x_0, x_1)$ зв'язана з функцією $T^3(x_0, x_1, x_2)$ тотожністю:

$$T^2(x_0, \boldsymbol{c}(x_1, x_2)) = T^3(x_0, x_1, x_2).$$

Дійсно,

$$\begin{aligned} T^2(x_0, \boldsymbol{c}(x_1, x_2)) &= \boldsymbol{l}(\mu_t(D^3(x_0, \boldsymbol{c}(x_1, x_2), t) = 0)) = \\ &= \boldsymbol{l}(\mu_t(D^2(x_0, \boldsymbol{c}(\boldsymbol{c}(x_1, x_2), t)) = 0)) = \\ &= \boldsymbol{l}(\mu_t(D^2(x_0, \boldsymbol{c}(x_1, x_2, t)) = 0)) = \\ &= \boldsymbol{l}(\mu_t(D^4(x_0, x_1, x_2, t)) = 0)) = \\ &= T^3(x_0, x_1, x_2). \end{aligned}$$

Тому

$$\begin{aligned} K^2(\boldsymbol{c}(x_0, x_1), x_2) &= T^2(\boldsymbol{l}(\boldsymbol{c}(x_0, x_1), \boldsymbol{c}(\boldsymbol{r}(\boldsymbol{c}\ (x_0, x_1), x_2)) = \\ &= T^2((x_0, \boldsymbol{c}(x_1, x_2)) = T^3(x_0, x_1, x_2). \end{aligned}$$

2) Функція $T^3(x_0, x_1, x_2)$ зв'язана з функцією $T^4(x_0, x_1, x_2, x_3)$ тотожністю:

$$T^3(x_0, \boldsymbol{c}(x_1, x_2), x_3) = T^4(x_0, x_1, x_2, x_3).$$

Дійсно,

$$\begin{aligned} &T^3(x_0, \boldsymbol{c}(x_1, x_2), x_3) = \boldsymbol{l}(\mu_t(D^4(x_0, \boldsymbol{c}(x_1, x_2), x_3), t) = 0)) = \\ &= \boldsymbol{l}(\mu_t(D^2(x_0, \boldsymbol{c}(\boldsymbol{c}(x_1, x_2), x_3, t) = 0)) = \\ &= \boldsymbol{l}(\mu_t(D^2(x_0, \boldsymbol{c}(x_1, x_2, x_3, t) = 0)) = \\ &= T^4(x_0, x_1, x_2, x_3). \end{aligned}$$

Тому,

$$\begin{aligned} &K^3(\boldsymbol{c}(x_0, x_1), x_2, x_3) = K^2([\boldsymbol{c}(x_0, x_1), x_2], x_3) = \\ &= K^2(\boldsymbol{c}(\boldsymbol{l}(\boldsymbol{c}(x_0, x_1), \boldsymbol{c}(\boldsymbol{r}(\boldsymbol{c}(x_0, x_1), x_2)), x_3) = \\ &= K^2(\boldsymbol{c}(x_0, \boldsymbol{c}(x_1, x_2)), x_3) = T^3(x_0, \boldsymbol{c}(x_1, x_2), x_3) = \\ &= T^4(x_0, x_1, x_2, x_3). \end{aligned}$$

Теорема 12.1. Функція Кліні $K^2(x_0, x_1)$ є ЧРФ, універсальною для всіх одномісних ЧРФ.

Доведення. Якщо $f(x)$ – довільна ЧРФ, то для функції $g(y, x) = f(x) + 0 \cdot y$ справедливе співвідношення

$$g(y, x) = T^3(a, y, x) = K^2(\boldsymbol{c}(a, y), x).$$

Поклавши $y = 0$ і позначивши $\boldsymbol{c}(a, 0) = b$, одержимо $f(x) = K^2(b, x)$.

Теорема 12.2. Функція Кліні $K^3(x_0, x_1, x_2)$ є ЧРФ, універсальною для всіх двомісних ЧРФ.

Доведення. Якщо $f(x, y)$ – довільна ЧРФ, то для функції $g(z, x, y) = 0 \cdot z + f(x, y)$ справедливе співвідношення

$$g(z, x, y) = T^4(a, z, x, y) = K^3(\boldsymbol{c}(a, \mathrm{z}), x, y).$$

Поклавши $z = 0$ і позначивши $\boldsymbol{c}(a, z) = b$ одержимо

$$f(x, y) = g(x, y, z) = K^3(b, x, y).$$

Відображення, яке кожному натуральному числу n ставить у відповідність одномісну ЧРФ $K(n, x)$ називати нумерацією Кліні одномісних ЧРФ. Число n називається клінівським номером функції $K(n, x)$, яка також позначається через K_n.

Теорема 12.3. Для кожної ЧРФ $f(x)$ існує число a таке, що всі числа послідовності

$$[a, 0], [a, 1], \ldots, [a, m], \ldots$$

будуть клінівськими номерами функції $f(x)$.

Доведення. За умовою

$$f(x) = K(n, x) = 0 \cdot y + f(x) = K(a, y, x) = K([a, y], x).$$

Тому числа послідовності є номерами функції $f(x)$.

Теорема 12.4. Для будь-якої ЧРФ $f(x, y)$ існує рекурсивна функція $s(x)$ така, що

$$f(x, y) = K(s(x), y).$$

Доведення. Випливає із співвідношень:
$f(x, y) = K^3(n, x, y) = K^2([n, x], y)$. Залишилось покласти $s(x) = [n, x]$.
Нумерація $[x_0, x_1, x_2]$ трійок натуральних чисел задається як

$$[x_0, x_1, x_2] = [[x_0, x_1], x_2].$$

Теорема 12.5 (Кліні про нерухому точку). Для кожної ЧРФ $h(x)$ існує число a таке, що

$$K(h(a), x) = K(a, x).$$

Доведення. Розглянемо функцію $K(h([y, y]), t)$. Тоді

$$K(h([y, y]), t) = K(b, y, t) = K([b, y], t).$$

Покладемо $y = b$. Тоді

$$K(h([b, b], t) = K([b, b], t)$$

Якщо позначити $[b, b] = a$, то одержимо

$$K(h(a), t) = K(a, t).$$

Теорема 12.6 (Райса). Якщо $\mathfrak{I}$ – непусте сімейство одномісних ЧРФ, відмінне від сукупності всіх таких функцій, то множина A клінівських номерів функцій, що належить $\mathfrak{I}$, не може бути рекурсивною.

Доведення. Нехай A – рекурсивна. Тоді рекурсивною буде і множина A'. Оскільки $\mathfrak{I}$ і $\overline{\mathfrak{I}}$ – не пусті, то A і A' – не пусті. Виберемо в A і A' по одному числу a, b і визначимо функцію

$$g(x) = \begin{cases} a, & x \in A' \\ b, & x \in A \end{cases}.$$

Ця функція рекурсивна ($g(x) = a\ \overline{sg}\,\chi(x) + b\,sg\,\chi(x)$, де $\chi(x)$ – характеристична для A'). За теоремою про нерухому точку існує число n таке, що $K(g(n), x) = K(n, x)$. Але тоді якщо $n \in A$, то $K(n, x) \in \Im$ і тому $K(g(n), x) \in \Im$. З іншого боку, якщо $n \in A$, то $g(n) = b$, тому $K(g(n), x) = K(b, x) \in \overline{\Im}$.
Одержали суперечність. Аналогічно n не може належати A'.

Зауваження. З теореми Райса випливає, що не існує алгоритму, який по номеру Кліні ЧРФ дасть відповідь на питання про належність цієї функції до класу, наприклад, ПРФ.

12.1. Алгоритми і універсальні функції Кліні

Раніше було показано, що кожна ЧРФ може бути обчислена алгоритмом:

```
function f(x)                    function T(n, x)
    begin                            begin
        f := T(n, x)                     i := 0
    end                                  while D(n, x, i) ≠ 0 do
                                                 i := i + 1
                                         T := l(i)
                                     end.
```

Це означає, що функція $T(n, x)$ є функцією універсальною для класу одномісних ЧРФ.

Якщо у співвідношенні

$$K^2(x_0, x_1) = T^2(\boldsymbol{l}(x_0), \boldsymbol{c}(\boldsymbol{r}(x_0), x_1))$$

покласти $n = \boldsymbol{l}(x_0)$, $x = \boldsymbol{c}(\boldsymbol{r}(x_0), x_1)$, то з другої рівності одержимо $x_1 = \boldsymbol{r}(x)$, $\boldsymbol{r}(x_0) = \boldsymbol{l}(x)$, а із співвідношення $\boldsymbol{c}(\boldsymbol{l}(x_0), \boldsymbol{r}(x_0)) = x_0$ – рівність $x_0 = \boldsymbol{c}(n, \boldsymbol{l}(x))$. Звідси,

$$T^2(n, x) = K^2(\boldsymbol{c}(n, \boldsymbol{l}(x)), \boldsymbol{r}(x)).$$

Тому будь-яка ЧРФ $f(x)$може бути обчислена алгоритмом

```
function f(x)                    function K(m, y)
    begin                            begin
        f := K(l(n), c(r(n), x))         i := 0
    end                                  while D(c(m, l(y)), r(y), i) ≠ 0 do
                                                 i := i + 1
                                         K := l(i)
                                     end,
```

де n – деяке (залежне від f) натуральне число, а точніше, n – число таке, що $f(x) = T(n, x)$.

Розглянемо функцію $g(z, x) = K(\boldsymbol{l}(z), \boldsymbol{c}(\boldsymbol{r}(z), x))$. Оскільки це ЧРФ, то

$$g(z, x) = T^3(a, z, x) = K(\boldsymbol{c}(a, z), x)).$$

Це означає, що функцію f можна обчислити алгоритмом

```
function f(x)
    begin
      f := K(c(a, n), x)
    end
```

```
function K(m, y)
    begin
      i := 0
      while D(c(m, l(y)), r(y), i) ≠ 0 do
            i := i + 1
      K := l(i)
    end,
```

тобто, кожна ЧРФ може бути обчислена через функцію $K(m, y)$ для деякого m. Отже, $K(m, y)$ – універсальна.

Приклад 1. Побудувати ПРФ, яка за номерами Кліні функцій $f(x)$ і $g(x)$ обчислює номер Кліні функції $f(g(x))$.

За умовою задачі

$$f(y) = K(m, y),\ g(x) = K(n, x),$$

і, відповідно, $f(g(x)) = K(m, K(n, x))$.

Розглянемо функцію $K(u, K(v, x))$. Для неї справедливі рівності:

$$K(u, K(v, x)) = K(a, u, v, x) = K([a, u, v], x).$$

Якщо покласти $u = m$, $v = n$, то зліва одержимо суперпозицію $f(g(x))$, а справа – її подання через універсальну функцію Кліні. Отже, достатньо покласти

$$\varphi(u, v) = [a, u, v].$$

Приклад 2. Нехай задані клінівські номери функцій $f(x)$ і $g(x)$. Треба знайти клінівський номер їх суми. Розглянемо функцію $K(u, x) + K(v, x)$ і відповідні її перетворення:

$$K(u, x) + K(v, x) = K(a, u, v, x) = K^3([a, u], v, x) =$$
$$K^2([[a, u], v], x) = K^2([a, u, v], x).$$

Тому, якщо m і n – номери f і g, то $[a, m, n]$ номер Кліні функції $f(x) + g(x)$.

13

Звідність та *m*-еквівалентність множин

Числова множина α називається m-звідною до числової множини β (позначається $\alpha \leq_m \beta$), якщо існує рекурсивна функція $f(x)$ така, що

$$f(\alpha) \subseteq \beta,$$
$$f(N \setminus \alpha) \subseteq N \setminus \beta.$$

При цьому говорять, що функція $f(x)$ m-зводить α до β.

Теорема 13.1. Кожна рекурсивна множина α m-зводиться до будь-якої непустої множини β, що має непусте доповнення. Якщо яка-небудь множина γ m-зводиться до рекурсивної (РП) множини δ, то γ – рекурсивна (РП) множина.

Доведення. Оскільки $\beta \neq \varnothing$ і $\overline{\beta} \neq \varnothing$, то існують числа $a \in \beta$ і $b \in \overline{\beta}$. Розглянемо функцію

$$f(x) = \begin{cases} a, & x \in \alpha \\ b, & x \in N \setminus \alpha \end{cases}.$$

Ця функція рекурсивна і m-зводить α до β. Дійсно, існує алгоритм обчислення характеристичної функції цієї множини:

```
function f(x)
    begin
        if χ_α(x) = 0 then f := a
        else f := b
    end.
```

Крім того, виконуються співвідношення:

$$f(\alpha) = \{a\} \subseteq \beta,$$
$$f(N \setminus \alpha) = \{b\} \subseteq N \setminus \beta.$$

Для доведення другої частини теореми припустимо, що функція $f(x)$ m-зводить γ до РМ δ. Тоді характеристична функція множини γ буде обчислюватися алгоритмом

```
function χγ(x)
    begin
      if χδ(f(x)) = 0 then χγ := 0
      else χγ := 1
    end,
```

де χ_δ – характеристична функція множини δ.

Якщо множина δ – РПМ, то вона співпадає з множиною значень деякої ПРФ функції $g(x)$. Тому, алгоритм обчислення часткової характеристичної функції χ_γ множини γ буде наступним:

```
function χγ(x)
    begin
      i := 0
      while g(i) ≠ x do
          i := i + 1
      χγ := 0
    end,
```

Твердження. Відношення m-звідності $\leq_m$ транзитивне.

Доведення. Дійсно, якщо $\alpha \leq_m \beta$ і $\beta \leq_m \gamma$, то існує рекурсивна функція h, яка m-зводить α до γ. Ця функція визначається співвідношенням $h(x) = g(f(x))$, де функція f m-зводить α до β, функція g m-зводить β до γ, оскільки

$$g(f(\alpha)) \subseteq \gamma \qquad (f(\alpha) \subseteq \beta),$$
$$g(f(N \setminus \alpha)) = N \setminus \gamma.$$

Множина β називається m-універсальною, якщо вона є РП-множиною і кожна РП-множина m-зводиться до β.

Із визначення m-звідності і транзитивності відношення $\leq_m$ випливає, що якщо m-універсальна множина m-зводиться до РПМ α, то α теж є m-універсальною множиною.

Наступна теорема дає приклад m-універсальної множини.

Теорема 13.2. Множина M канторових номерів тих пар $<a, b>$, для яких рівняння

$$K(a, x) = b$$

має розв’язок, є m-універсальною множиною.

Доведення. Покажемо, що множина M є РПМ. Дійсно, нехай

$$G = <\alpha_1(t), \alpha_2(t), \alpha(t)>, t = 0, 1, \ldots$$

є графіком функції $K(n, x)$. Тоді часткова характеристична функція $\chi_M(x, y)$ множини пар $<a, b>$, для яких рівняння $K(a, x) = b$ має розв'язок обчислюється алгоритмом:

```
function χ_M(x, y)
    begin
      i := 0
      while α_1(i) ≠ x ∨ α(i) ≠ y do
              i := i + 1
      χ_M := 0
    end.
```

Отже, множина M є РПМ.

Покажемо, що будь-яка РПМ m-зводиться до множини $\boldsymbol{c}(M)$ канторових номерів пар з M. Дійсно, нехай α – РПМ. Це означає, що α – множина значень ПРФ $f(x) = K(n_0, x)$ для деякого $n = n_0$. Розглянемо функцію

$$g(z) = \boldsymbol{c}(n_0, z).$$

Ця функція m-зводить множину α до множини $\boldsymbol{c}(M)$. Дійсно, якщо $z \in \alpha$, то $z = f(x_0)$, для деякого $x = x_0$. Тому рівняння $K(n_0, y) = z$ має розв'язок $y = x_0$, а, отже, пара $<n_0, z> \in M$. Звідси, $\boldsymbol{c}(n_0, z) \in \boldsymbol{c}(M)$. Отже, $g(\alpha) \subseteq \boldsymbol{c}(M)$.

Нехай $z \in N \setminus \alpha$. Тоді рівняння

$$K(n_0, x) = z$$

не має розв'язку, оскільки всі значення функції $K(n_0, x)$ належать до множини α. Звідси випливає, що пара $<n_0, z> \notin M$ і, тому, $\boldsymbol{c}(n_0, z) \notin \boldsymbol{c}(M)$. Отже, $g(N \setminus \alpha) \subseteq N \setminus \boldsymbol{c}(M)$.

14

Продуктивні множини

Вище (теорема 10.5) було показано, що існують нерекурсивні РПМ. Це означає, що доповнення таких множин не може бути РПМ. Звідси випливає, що існують також множини, які не є РПМ.

Далі символом π_n або πn будемо позначати сукупність значень функції $K(n, x)$ і називати множиною з постівським номером n.

Не РПМ α називається продуктивною, якщо існує рекурсивна функція $f(x)$ (продуктивна функція) така, що для кожного n

$$f(n) \in (\alpha \setminus \pi_n) \cup (\pi_n \setminus \alpha).$$

Теорема 14.1. Існує ПРФ $u(x, y)$ така, що для будь-яких чисел m, n

$$\pi_m \cup \pi_n = \pi_{u(m,n)}.$$

Доведення. Введемо функцію

$$G(m,n,x) = \begin{cases} K(m, \frac{x}{2}), & \text{якщо } x \text{ парне,} \\ K(n, \frac{x-1}{2}), & \text{якщо } x \text{ непарне.} \end{cases}$$

Ця функція частково рекурсивна. Дійсно, вона обчислюється наступним алгоритмом:

```
function G(m, n, x)
    begin
        i := 0
        if rest(x,2) = 0 then
            while α1(i) ≠ m ∨ α2(i) ≠ [x/2] do
                i := i + 1
        G := α(i)
        else
            while α1(i) ≠ n ∨ α2(i) ≠ [(x – 1)/2] do
                i := i + 1
        G := α(i)
    end,
```

де $<\alpha_1(i), \alpha_2(i), \alpha(i)>$ – графік функції $K(x, y)$. Множина її значень при $x = 1, 2, \ldots$ і фіксованих $m, n \in \boldsymbol{\pi}_m \cup \boldsymbol{\pi}_n$. Крім того

$$G(m, n, x) = K(a, m, n, x) = K([a, m, n], x).$$

Множина значень функції $K([a, m, n], x) \in \boldsymbol{\pi}_{[a,m,n]}$. Тому,

$$\boldsymbol{\pi}_m \cup \boldsymbol{\pi}_n = \boldsymbol{\pi}_{[a,m,n]}$$

і, отже, треба покласти

$$u(m, n) = [a, m, n].$$

Теорема 14.2. Будь-яка продуктивна множина містить нескінченну РПМ.

Доведення. Нехай α – продуктивна множина. Будуємо множину B наступним чином. Постівський номер порожньої множини позначимо через a_0. Через a_1 позначимо постівський номер множини $\{f(a_0)\}$. Через a_2 позначимо постівський номер множини $\{f(a_0), f(a_1)\}$. Через a_3 позначимо постівський номер множини $\{f(a_0), f(a_1), f(a_2)\}$. Продовжуючи цей процес, одержимо нескінченну послідовність

$$f(a_0), f(a_1), f(a_2), \ldots, f(a_n), \ldots .$$

Всі числа цієї послідовності різні і належать до α. Дійсно,

$$f(a_0) \in (\alpha \setminus \boldsymbol{\pi}_{a_0}) \cup (\boldsymbol{\pi}_{a_0} \setminus \alpha) = (\alpha \setminus \varnothing) \cup (\varnothing \setminus \alpha) = \alpha,$$
$$f(a_1) \in (\alpha \setminus \boldsymbol{\pi}_{a_1}) \cup (\boldsymbol{\pi}_{a_1} \setminus \alpha) = (\alpha \setminus \{f(a_0)\}) \cup (\{f(a_0)\} \setminus \alpha) = (\alpha \setminus \{f(a_0)\}),$$
$$f(a_2) \in (\alpha \setminus \boldsymbol{\pi}_{a_2}) \cup (\boldsymbol{\pi}_{a_2} \setminus \alpha) = (\alpha \setminus \{f(a_0), f(a_1)\}) \cup (\{f(a_0), f(a_1)\} \setminus \alpha) =$$
$$= (\alpha \setminus \{f(a_0), f(a_1)\}), \text{ і. т. д.}$$

Покажемо, що множина

$$B = \{f(a_0), f(a_1), f(a_2), \ldots\}$$

є РПМ. Із побудови множини B випливає, що

$$\boldsymbol{\pi}_{a_1} = \boldsymbol{\pi}_{a_0} \cup \{f(a_0)\}$$
$$\boldsymbol{\pi}_{a_2} = \boldsymbol{\pi}_{a_1} \cup \{f(a_1)\}$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$

Постівські номери одноелементних множин $\{f(a_0)\}$, $\{f(a_1)\}$, $\{f(a_2)\}$, ... знаходимо із співвідношень:

$$h(z) = z + 0 \cdot y = K(n_0, z, y) = K([n_0, z], y).$$

Звідси, $g(x) = h(f(x)) = f(x) + 0 \cdot y = K(n_0, f(x), y) = K([n_0, f(x)], y)$. Тому, постівські номери вище згаданих множин є, відповідно, $[n_0, f(a_0)]$, $[n_0, f(a_1)]$, $[n_0, f(a_2)]$ і. т. д. Звідси випливає, що

$$a_1 = u(a_0, g(a_0))$$
$$a_2 = u(a_1, g(a_1))$$
$$\ldots\ldots\ldots\ldots\ldots$$

Тому часткова характеристична функція $\chi_B(x)$ множини B обчислюється алгоритмом:

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

function a(x)
    begin
      if x = 0 then a := a_0
      else a := u(a(x – 1), g(x – 1))
    end,
```

де $u(m, n)$ ПРФ з попередньої теореми.

Теорема 14.3. Якщо продуктивна множина α m-зводиться до множини β, то β – продуктивна множина.

Доведення. Множина β не може бути РПМ, оскільки в цьому випадку множина α була б РПМ. А це не так.

За умовою теореми існують рекурсивні функції $f(x)$ та $g(x)$ такі, що

$$f(\alpha) \subseteq \beta,$$
$$f(N \setminus \alpha) \subseteq N \setminus \beta,$$

і для будь-якого n

$$g(n) \in (\alpha \setminus \pi_n) \cup (\pi_n \setminus \alpha).$$

Треба показати, що існує рекурсивна функція $h(x)$ така, що для будь-якого n

$$h(n) \in (\beta \setminus \pi_n) \cup (\pi_n \setminus \beta).$$

Розглянемо два випадки. Нехай $\pi_n \subseteq \beta$. Із властивостей функції f випливає, що тоді $f^{-1}(\pi_n) \subseteq \alpha$. Крім того, $f^{-1}(\pi_n)$ є РПМ. Дійсно, алгоритм обчислення часткової характеристичної функції цієї множини наступний:

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

де $<\varphi(t), \psi(t)>$, $t = 0, 1, 2, \ldots$ – графік одномісної ЧРФ $K(n, x)$.

Так як $f^{-1}(\pi_n)$ є РПМ, то вона співпадає з множиною значень деякої ПРФ $q(x)$. Із співвідношень

$$q(x) = K(m, x) = 0 \cdot y + q(x) = p(y, x) = K(s(y), x)$$

при $y = n$ одержимо

$$f^{-1}(\pi_n) = \pi_{s(n)},$$

де $s(x)$ – ПРФ (теорема 11.4).

Покладемо $h(x) = f(g(s(x)))$. Тоді для будь-якого n

$$h(n) \in (\beta \setminus \pi_n) \cup (\pi_n \setminus \beta).$$

Дійсно, $g(s(n)) \in (\alpha \setminus \pi_{s(n)}) \cup (\pi_{s(n)} \setminus \alpha) = \alpha \setminus \pi_{s(n)}$ $(\pi_{s(n)} \setminus \alpha = \varnothing)$. Отже, $h(n) = f(g(s(x))) \in \beta$. Крім того, $g(s(n)) \notin \pi_{s(n)}$, а, отже, $h(n) = f(g(s(x))) \notin \pi_n$. Таким чином, $h(n) \in \beta \setminus \pi_n$.

Якщо $\pi_n \not\subset \beta$, то $f^{-1}(\pi_n) = \pi_{s(n)} \not\subset \alpha$. Тоді для будь-якого n

$$g(s(n)) \in (\alpha \setminus \pi_{s(n)}) \cup (\pi_{s(n)} \setminus \alpha).$$

Отже, $h(n) \in (\beta \setminus \pi_n) \cup (\pi_n \setminus \beta)$.

Приклад 1. Довести, що існує ПРФ g така, що $\{x\} = \pi_{g(x)}$.

Для розв'язання задачі теба показати, що існує ПРФ g така, що $\{0\} = \pi_{g(0)}$, $\{1\} = \pi_{g(1)}$, … . Відомо, що для будь-якої ЧРФ $f(x, y)$ існує ПРФ $g(x)$ така, що

$$f(x, y) = K(g(x), y).$$

Покладемо $f(x, y) = x + 0 \cdot y$. Тоді будемо мати рівності:

$$\{0\} = f(0, y) = K(g(0), y),$$
$$\{1\} = f(1, y) = K(g(1), y),$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$

Отже, $\{x\} = \pi_{g(x)}$.

Приклад 2. Довести, що існує число n таке, що $\pi_n = \{n\}$.

За попередньою задачею існує ПРФ g така, що $\{x\} = \pi_{g(x)}$. За теоремою про нерухому точку для функції g існує число n таке, що

$$K(g(n), y) = K(n, y).$$

Отже, $\pi_{g(n)} = \pi_n = \{n\}$.

Приклад 3. Довести, що існує число n таке, що $\pi_n = \{n^2\}$.

Покладемо $f(x, y) = x^2 + 0 \cdot y$. Тоді будемо мати рівності:

$$\{0\} = f(0, y) = K(g(0), y),$$
$$\{1\} = f(1, y) = K(g(1), y),$$
$$\{4\} = f(2, y) = K(g(2), y),$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$

Отже, $\{x^2\} = \pi_{g(x)}$.

За теоремою про нерухому точку для функції g існує число n таке, що

$$K(g(n), y) = K(n, y).$$

Отже, $\pi_{g(n)} = \pi_n = \{n^2\}$.

Приклад 4. Довести, що кожна m-універсальна множина не рекурсивна.

Відомо, що існують не рекурсивні РПМ. Наприклад, область визначення функції $K(n, x)$. Нехай β – РПМ, яка не є рекурсивною множиною. Якщо припустити, що m-універсальна множина α є рекурсивною, то з того, що $\beta \leq_m \alpha$ випливає, що β буде РМ. Дійсно, характеристична фунція множини β може бути в цьому випадку обчислена алгоритмом:

```
function χβ(x)
    begin
        i := 0
        if  χβ(f(x)) = 0 then χβ := 0
```

else χ_{β} := 1
end,

де $f(x)$ – функція, яка m-зводить $\beta\alpha$ до α.

А це не так.

15

Креативні множини

РПМ множина α, доповнення якої $\overline{\alpha} = N \setminus \alpha$ продуктивне, називається креативною множиною.

Із теореми 14.3 випливають два наслідки.

Наслідок 15.1. Якщо креативна множина α m-зводиться до РПМ множини β, то множина β креативна.

Доведення. Якщо $\alpha \le_m \beta$, то $\overline{\alpha} \le_m \overline{\beta}$. Але $\overline{\alpha}$ – продуктивна множина. За попередньою теоремою множина $\overline{\beta}$ теж продуктивна. Отже, β – креативна множина.

Наслідок 15.2. Кожна m-універсальна множина є креативною.

Доведення. Дійсно, до m-універсальної множини m-зводиться будь-яка РПМ, в тому числі і креативна множина. За попереднім наслідком m-універсальна множина є креативною.

Предикат $P(x_1, x_2, \ldots, x_n)$ називається РПП, якщо множина n-ок $<x_1, x_2, \ldots, x_n>$, для яких предикат істинний, є РПМ.

Теорема 15.1 (про подання предикатів)**.** Для кожного РПМ предиката $P(x, y)$ існує ПРФ $h(y)$ така, що

$$P(x, y) \Leftrightarrow x \in \pi_{h(y)}.$$

Доведення. Графік предиката можна подати у вигляді множини $<v(t), w(t)>$, $t = 0, 1, \ldots$, де v, w – деякі ПРФ. Розглянемо функцію, яка обчислюється алгоритмом:

```
function f(y)
    begin
        i := 0
        while w(i) ≠ y do
            i := i + 1
        f := v(i)
    end.
```

Ця функція є ЧРФ і $P(f(y), y)$. Тому із співвідношень

$$f(y) = f(y) + 0 \cdot z = K(a, y, z) = K([a,y], z)$$

випливають наступні імплікації:

$$P(x, y) \Rightarrow x = f(y) \Rightarrow f(y) \in \pi_{h(y)},$$
$$x \in \pi_{h(y)} \Rightarrow x = f(y) \Rightarrow P(x, y),$$

де $h(y) = [a, y]$. Отже,

$$P(x, y) \Leftrightarrow x \in \pi_{h(y)}.$$

Позначимо через $Q(x_1, y)$ предикат $P(x_1, [y]_{n1}, \ldots, [y]_{nn})$. Тоді існує ПРФ $g(y)$ така, що

$$Q(x_1, y) \Leftrightarrow x_1 \in \pi_{g(y)}.$$

Підставивши сюди замість y вираз $[x_2, \ldots, x_{n+1}]$, одержимо загальний вигляд твердження про подання предикатів: для кожного РПМ предиката $P(x_1, x_2, \ldots, x_{n+1})$ існує ПРФ $h(x_2, \ldots, x_{n+1})$ така, що

$$P(x_1, x_2, \ldots, x_{n+1}) \Leftrightarrow x_1 \in \pi h(x_2, \ldots, x_{n+1}).$$

Теорема 15.2 (Майхіла про нерухому точку)**.** Для кожного РПП $P(x_1, x_2, \ldots, x_{n+1})$ існує ПРФ $g(x_2, \ldots, x_n)$ така, що

$$P(x_1, x_2, \ldots, x_n, g(x_2, \ldots, x_n)) \Leftrightarrow x_1 \in \pi g(x_2, \ldots, x_n)$$

для всіх $x_1, x_2, \ldots, x_n$.

Доведення. За попередньою теоремою існує ПРФ $h(x_2, \ldots, x_{n+1})$ така, що

$$P(x_1, x_2, \ldots, x_{n+1}) \Leftrightarrow x_1 \in \pi h(x_2, \ldots, x_{n+1}).$$

За теоремою Кліні про нерухому точку існує ПРФ $g(x_2, \ldots, x_n)$ така, що

$$K(h(x_2, \ldots, x_{n+1}, g(x_2, \ldots, x_n)), t) = K(g(x_2, \ldots, x_n), t).$$

Тому

$$\pi h(x_2, \ldots, x_{n+1}, g(x_2, \ldots, x_n)) = \pi g(x_2, \ldots, x_n).$$

Підставляючи замість x_{n+1} вираз $g(x_2, \ldots, x_n)$ у рівносильність одержимо:

$$P(x_1, x_2, \ldots, g(x_2, \ldots, x_n)) \Leftrightarrow x_1 \in \pi h(x_2, \ldots, g(x_2, \ldots, x_n)) = \pi g(x_2, \ldots, x_n).$$

Теорема 15.3 (Майхіла)**.** Кожна креативна множина є m-універсальною.

Доведення. Нехай α – задана креативна множина, а $f(x)$ – продуктивна функція для $\overline{\alpha}$. Треба показати, що довільна РПМ β m-зводиться до α.

Розглянемо предикат

$$P(x, y, z) = y \in \beta \wedge x = f(z).$$

Цей предикат є РПМ. Дійсно, часткова характеристична функція предиката обчислюється алгоритмом

```
function χ(x, y, z)
    begin
        i := 0
        while y ≠ w(i) ∨ x ≠ f(z) do
                i := i + 1
        χ := 0
    end,
```

де $\beta = \{w(t), t = 1, 2, \ldots\}$. За попередньою теоремою існує ПРФ $g(y)$ така, що

$$y \in \beta \wedge x = f(g(y)) \Leftrightarrow x \in \pi_{g(y)}.$$

Покажемо, що функція $f(g(x))$ m-зводить β до α. Дісно, правдива імплікація:

$$n \in \overline{\beta} \Rightarrow \pi_{g(n)} = \varnothing,$$

оскільки з попередньої рівносильності випливає, що жодне натуральне x не належить до $\pi_{g(n)}$. Крім того, правдиві наступні імплікації:

$$\pi_{g(n)} \subseteq \overline{\alpha} \Rightarrow f(g(n)) \in (\overline{\alpha} \setminus \pi_{g(n)}) \cup (\pi_{g(n)} \setminus \overline{\alpha}) \Rightarrow f(g(n)) \in \overline{\alpha}.$$

З іншого боку, правдива імплікація

$$n \in \beta \Rightarrow \pi_{g(n)} = \{f(g(n))\},$$

тобто, множина $\pi_{g(n)}$ містить тільки один елемент. Дійсно, якщо $n \in \beta$ то з попередньої рівносильності випливає, що $f(g(n)) \in \pi_{g(n)}$. Якщо ж $z \in \pi_{g(n)}$, то $z = f(g(y))$.

Отже,

$$n \in \beta \Rightarrow f(g(n)) \in \alpha.$$

Дійсно, якщо $f(g(n)) \in \overline{\alpha}$, то правдива імплікація

$$\pi_{g(n)} = \{f(g(n))\} \subseteq \overline{\alpha} \Rightarrow f(g(n)) \in (\overline{\alpha} \setminus \pi_{g(n)}) \cup (\pi_{g(n)} \setminus \overline{\alpha}) \Rightarrow$$
$$f(g(n)) \in (\overline{\alpha} \setminus \pi_{g(n)}).$$

Але $f(g(n))$ не може належати $\overline{\alpha} \setminus \pi_{g(n)}$. Таким чином, $f(g(n)) \in \alpha$.

Із наслідку 15.2 та теореми 15.3 випливає, що клас m-універсальних множин співпадає з класом креативних множин.

16

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*, 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 $K(x, y)$ визначена і 1, якщо $K(x, y)$ 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 16.1. Проблема зупинки алгоритмічно нерозв'язна.

Доведення. Випливає з того, що область визначення функції $K(x, y)$ є нерекурсивною РПМ (Теорема 11.5).

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 16.2. Проблема належності є алгоритмічно нерозв'язною.

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

16.1. Універсальні числові множини

Універсальною числовою множиною будемо називати множину U пар чисел (i, x) таких, що $K(i, x) = 1$, тобто

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

Теорема 16.3. Проблема належності до універсальної числової множини є алгоритмічно нерозв'язною.

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

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

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, $f(x) = K(n, x)$, для деякого n. Обчислимо цю функцію для $x = n$.

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

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

16.2. Словарні множини та функції

Нехай $\Sigma = \{a_1, \ldots, a_n\}$ – алфавіт. Через Σ^* позначимо множину всих слів в алфавіті Σ. Довільну підмножину L множини Σ^* будемо називати словарною множиною або мовою в алфавіті Σ.

Граматикою називається четвірка

$$G = (N, \sum, P, S), \text{ де}$$

N – множина нетермінальних символів;

$\sum$ – множина термінальних символів ($N \cap \sum = \varnothing$);

P – підмножина множини $(N \cup \sum)^* N (N \cup \sum)^* \times (N \cup \sum)^*$

(елемент (α, β) множини P називається продукцією і записується у вигляді $\alpha \to \beta$);

S – символ із N, який називається початковим символом граматики.

Граматика G породжує мову $L(G)$ – множину терминальних слів, які виводяться в цій граматиці. Наприклад, якщо

$$G = (N, \sum, P, S), \text{ то}$$

(1) S – вивідне слово;

(2) якщо $\alpha\beta\gamma$ – вивідне слово і продукція $\beta \to \delta$ знаходиться в P, то $\alpha\delta\gamma$ – теж вивідне слово.

Слово, яке не містить нетермінальних символів, називається термінальним.

Нехай $G = (N, \sum, P, S)$ – граматика. Вона називається

(1) праволінійною, якщо кожне правило із P має вигляд

$$A \to xB \text{ или } A \to x, \text{ де } A, B \in N, x \in \sum^*;$$

(2) контекстно-вільною, якщо кожне правило із P має вигляд

$$A \to \alpha, \text{ де } A \in N, \alpha \in (N \cup \sum)^*;$$

(3) контекстно-залежною, якщо кожне правило із P має вигляд

$$\alpha \to \beta, \text{ де } |\alpha| \le |\beta| \text{ і } \alpha, \beta \in (N \cup \sum)^*.$$

(4) загального вигляду, якщо не задовольняє жодному із вказаних обмежень.

16.3. Універсальні словарні множини

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

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

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

Множину всих таких слів будемо називати базою.

Універсальною словарною множиною 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 $*A_1*A_2* \dots *A_n*w$ таких, що w належить РПМ, яка породжується граматикою з продукціями $A_1, A_2, \dots, A_n$, тобто

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

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

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

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

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

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

Довільну часткову функцію $F{:}(\Sigma \cup N')^* \to (\Sigma \cup N')^*$ будемо називати частковою словарною функцією в алфавіті $\Sigma \cup N'$. Часткова числова функція $f{:}N \to N$ називається функцією, що представляє словарну функцію F в нумерації α, 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 $*A_1*A_2* \dots *A_n*w \in V$ видає 1, якщо слово w породжується граматикою з продукціями з бази $*A_1*A_2* \dots *A_n*$. 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 одержуємо суперечність, а тому функція $g(x)$ не може бути рекурсивною.

Тому, справедлива теорема

Теорема 16.4.

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.

16.4. Проблема відповідностей Поста

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

Нехай

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

множина пар слів в алфавіті Σ. 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 16.5. Проблема відповідностей Поста алгоритмічно нерозв'язна.

Доведення. За довільною словом $*A_1*A_2* \ldots *A_n*w$ 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 Σ до множини пар слів додаємо пари (a, a);

– для кожного не термінального символа S', S'', …, $S'^{\cdots'}$ до множини пар слів додаємо пари (S', S'), (S'', S''), …, $(S'^{\cdots'}, S'^{\cdots'})$;

– 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:

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

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

Приклад. Розглянемо граматику

$$G = (\{a, b, c\}, \{S, A, B, C\}, \{S \to aABb, S \to Bbb, Bb \to C, AC \to aac\})$$

і слово $w = aaac$. Цій граматиці ставимо у відповідність слово

$$*S \to aS'S''b*S \to S''bb*S''b \to S'''*S'S''' \to aac*aaac.$$

За правилами, наведеними вище, утворюємо наступну множину пар:

$$\{(FS\Rightarrow, F), (a, a), (b, b), (c, c), (S', S'), (S'', S''), (S''', S'''),$$
$$(E, \Rightarrow aaacE), (aS'S''b, S), (S''bb, S), (S''', S''b), (aac, S'S'''), (\Rightarrow, \Rightarrow)\}.$$

Утворимо послідовність пар ціє множини:

$$(FS\Rightarrow, F), (aS'S''b, S), (\Rightarrow, \Rightarrow), (a, a), (S', S'), (S''', S''b),$$
$$(\Rightarrow, \Rightarrow), (a, a), (aac, S'S'''), (E, \Rightarrow aaacE).$$

Ця послідовність пар є розв'язком ПВП.

17

Алгоритмічна розв'язність в класі КВ-мов

17.1. Проблема неоднозначності КВ-граматик

Ця проблема полягає в 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 Σ. 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 17.1. 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}:

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

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

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

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 $m \geq 1$. Так як

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

то $i_1, \dots, i_m$ – розв'язок.

Теорема 17.2. Проблема неоднозначності КВ-граматик алгоритмічно нерозв'язна.

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

Теорема 17.3. Проблема

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

алгоритмічно нерозв'язна.

Доведення. Розглянемо довільну ПВП в афавіті Σ 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 17.4. 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 Σ 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,

$L(G_1)$ і $L(G_2)$ співпадають $\Leftrightarrow$ 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 17.5. 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.

18

Алгоритмічна розв'язність в класі формальних теорій

Нехай зафіксовано алфавіт Σ, а також скінченна множина пар слів

$$M = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

в цьому алфавіті. Два слова v і w в алфавіті Σ назвемо еквівалентними ($v \equiv w$), якщо одне з них можна перетворити в інше роблячі заміни

$$(x_i \leftrightarrow y_i).$$

Проблема полягає в тому, щоб дати відповідь на питання про існування алгоритму, виходом якого для довільної множини M пар слів та слів v, $w \in 0$, якщо слова еквівалентні і 1 в іншому випадку. Ця проблема називається *проблемою еквівалентності слів для асоціативного числення* (ПЕАЧ).

Теорема 18.1. Проблема еквівалентності слів для асоціативного числення алгоритмічно нерозв'язна.

Доведення. За довільним словом $*S \to \alpha_1* \dots *S \to \alpha_k*A_1*A_2* \dots *A_n*w$ універсальної словарної множини будуємо ПЕАЧ згідно наступного правила:

– продукції з початковим нетерміналом граматики не використовуються для побудови множини M пар слів;

– для кожної продукції $A_i = v \to u$ до множини пар слів додаємо пару (v, u).

Можна показати, що справедливе співвідношення:

$$*S \to \alpha_1* \dots *S \to \alpha_k*A_1*A_2* \dots *A_n*w \in V \Leftrightarrow \alpha_1 \equiv w \vee \dots \vee \alpha_k \equiv w,$$

де S – початковий нетермінал граматики.

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

18.1. Теорії першого порядку

Символи теорії першого порядку (ТПП) визначаються наступним чином:

– логічні зв'язки $\neg$, $\rightarrow$;

– дужки (,);

– змінні x, y, z, u, … з індексами або без них;

– предикати P, Q, R, … з індексами або без них;

– функціональні символи f, g, h, … з індексами або без них;

– константи a, b, c … з індексами або без них.

Визначимо поняття *терма*.

Термом називається послідовність змінних, ком, дужок і функціональних символів, яку можна побудувати за наступними правилами:

1. Змінна є терм;

2. Константа є терм;

3. Якщо $t_1, \dots, t_k$ – терми, а f – функціональний символ розмірності $k > 0$, то $f(t_1, \dots, t_k)$ є термом.

Якщо A – предикатний символ розмірності k, а $t_1, \dots, t_k$ – терми, то формула $A(t_1, \dots, t_k)$ є *елементарною формулою*.

Формули теорії першого порядку визначаються за такими правилами:

1. Елементарна формула – формула.

2. Якщо α – формула, то $\neg\alpha$ – формула.

3. Якщо α і β – формули, то вирази $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$ – формули.

4. Якщо α є формулою, а ξ – змінна, то вирази $\forall\xi\alpha$ і $\exists\xi\alpha$ – формули, де $\exists\xi\alpha$ є скороченим записом формули $\neg(\forall\xi\neg\alpha)$).

Отже, поняття формули повністю визначене. Такі формули іноді називають *формулами першого порядку* або *формулами мови першого порядку*.

Формулу називаємо замкнутою, якщо вона не містить параметрів (вільних змінних). Істиносне значення такої формули залежить тільки від інтерпретації.

Аксіоми теорії першого порядку діляться на два класи: логічні і власні. Логічними аксіомами теорії першого порядку є наступні:

1. $\alpha \rightarrow (\beta \rightarrow \alpha)$;
2. $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$;
3. $(\alpha \wedge \beta) \rightarrow \alpha$;
4. $(\alpha \wedge \beta) \rightarrow \beta$;
5. $\alpha \rightarrow (\beta \rightarrow (\alpha \wedge \beta))$;
6. $\alpha \rightarrow (\alpha \vee \beta)$;
7. $\beta \rightarrow (\alpha \vee \beta)$;
8. $(\alpha \rightarrow \gamma) \rightarrow ((\beta \rightarrow \gamma) \rightarrow (\alpha \vee \beta \rightarrow \gamma))$;
9. $\neg\alpha \rightarrow (\alpha \rightarrow \beta)$;
10. $(\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow \neg\beta) \rightarrow \neg\alpha)$;
11. $\alpha \vee \neg\alpha$;

12. $\forall x \alpha(x) \to \alpha(t)$, де x – вільна змінна в α, t – терм, який не містить змінних, які були б зв'язаними при підстановці t замість x в α;
13. $\forall x(\alpha \to \beta) \to (\alpha \to \forall x\beta)$, де формула α не містить вільного входження змінної x.

Правила виведення в теорії першого порядку є:

1. $\alpha, \alpha \to \beta \Rightarrow \beta$;
2. $\alpha \Rightarrow \forall x \alpha$.

Інтерпретація, в якій істинні всі аксіоми теорії називається *моделлю* цієї теорії.

Теорія першого порядку, яка не містить власних аксіом називається *численням предикатів першого порядку*.

Приклад 18.1. Нехай сигнатура ТПП містить одну предикатну букву A^2 і не містить функціональних символів і констант. Замість $A^2(x, y)$ і $A^2(x, y)$ будемо писати

$$x < y \text{ і } \neg(x < y)$$

відповідно. Крім логічних аксіом, нехай ТПП містить дві власні аксіоми:

1. $\forall x \neg(x < y)$;
2. $\forall x \forall y \forall z\ (x < y \wedge y < z \to x < z)$.

Всяка модель цієї теорії називається *частково впорядкованою структурою*.

Приклад 18.2. Нехай сигнатура ТПП містить один предикат рівності, який будемо записувати у вигляді

$$x = y.$$

Крім логічних аксіом, нехай ТПП містить власні аксіоми рівності:

1. $\forall x\ (x = x)$;
2. $\forall x \forall y\ ((x = y) \to (y = x))$;
3. $\forall x \forall y \forall z\ (((x = y) \wedge (y = z)) \to (x = z))$.

Всяка модель цієї теорії називається *теорією рівності*.

Для довільної ТПП ми хочемо дати відповідь на питання: чи буде довільна формула теорії вивідною в цій теорії? Іншими словами, ми хочемо дати відповідь на питання про існування алгоритму, який за довільною формулою

теорії видає 0, якщо формула є вивідною і 1 в іншому випадку. У випадку існування такого алгоритму, як відомо, теорія називається розв'язною. Ця проблема називається *проблемою вивідності* для теорії першого порядку.

Розглянемо теорію, сигнатура якої містить один предикат рівності та один двохмісний функціональний символ, який називається множенням.

Крім логічних аксіом, ця теорія містить власні аксіоми рівності:

1. $\forall x\,(x = x)$;
2. $\forall x \forall y\,((x = y) \to (y = x))$;
3. $\forall x \forall y \forall z\,(((x = y) \wedge (y = z)) \to (x = z))$,

і аксіоми коректності множення та асоціативності

4. $\forall x \forall y \forall v \forall w\,((x = y) \wedge (v = w) \to (xv = yw))$;
5. $\forall x \forall y \forall z\,((xy)z = x(yz))$

відповідно.

Всяка модель цієї теорії називається *вільною напівгрупою*.

Прикладом конкретної моделі цієї теорії може бути множина слів в довільному алфавіті разом з операцією конкатенації.

Теорема 18.2. Проблема вивідності слів для теорії першого порядку алгоритмічно нерозв'язна.

Доведення. Покажемо, що проблему еквівалентності слів для асоціативного числення можна звести до проблеми вивідності слів в теорії напівгруп.

Дійсно, нехай зафіксовано алфавіт $\Sigma = \{a_1, \ldots, a_m\}$, а також скінченна множина пар слів

$$M = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

в цьому алфавіті. Треба вияснити, чи будуть слова v і w в алфавіті Σ еквівалентними.

Слово

$$\forall a_1 \forall a_2 \ldots \forall a_m\,(x_1 = y_1 \wedge\ x_2 = y_{2\,\wedge} \ldots \wedge x_n = y_n) \to v = w$$

є формулою теорії вільної напівгрупи. Крім того, можна показати, що справедливе співвідношення:

Формула $\forall a_1 \forall a_2 \ldots \forall a_m\,(x_1 = y_1 \wedge\ x_2 = y_{2\,\wedge} \ldots \wedge x_n = y_n) \to v = w$ вивідна $\Leftrightarrow$ $v \equiv w$ в асоціативному численні M.

Таким чином, якщо проблема вивідності формул в теорії вільної напівгрупи алгоритмічно розв’язна, то алгоритмічно розв’язною буде і проблема еквівалентності слів в асоціативному численні. А це не так.

19

Елементи рекурсивного аналізу

19.1. Конструктивні дійсні числа

Конструктивне (обчислюване) *дійсне число* α – це число, яке:
– є границею послідовності раціональних чисел

$$r(0), r(1), \ldots, r(n), \ldots$$

кожен елемент якої є значенням алгоритмічно обчислюваної функції r: $N \to R$, тобто раціональне число $r(n)$ є n-м членом цієї послідовності;

– існує алгоритмічно обчислювальна функція h: $R \to N$ (регулятор збіжності) така, що для довільного раціонального числа ε всі елементи послідовності з номерами $n \geq h(\varepsilon)$ будуть знаходитись на відстані від α меншій ε, тобто

$$|r(n) - \alpha| < \varepsilon.$$

Далі будемо називати такі числа рекурсивними.

Прикладами рекурсивних чисел є, наприклад, числа e, π, $\sqrt{2}$. Дійсно, щоб визначити ці числа, можна скористатися розкладом в ряди:

$$e = 2 + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!} + \cdots,$$

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots + (-1)^{n-1}\frac{1}{2n-1} + \cdots\right),$$

$$\sqrt{2} = 1 + \frac{1}{2} - \frac{1}{8} + \frac{1}{16} + \cdots + (-1)^{n-1}\frac{(2n-3)!!}{2n!!} + \cdots,$$

де

$$(2n)!! = 2\cdot4\cdot6\cdot \ldots\cdot2n,\ (2n-3)!! = 1\cdot3\cdot5\cdot \ldots\cdot(2n-3).$$

Утворимо послідовності $\{a(n)\}, \{b(n)\}, \{c(n)\}$ часткових сум цих рядів:

$$a(1) = 2, a(2) = 2 + \frac{1}{2!}, a(3) = 2 + \frac{1}{2!} + \frac{1}{3!}, \ldots,$$

$$a(n) = 2 + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!}, \ldots;$$
$$b(1) = 4, b(2) = 4\left(1 - \frac{1}{3}\right), b(3) = 4\left(1 - \frac{1}{3} + \frac{1}{5}\right), \ldots,$$
$$b(n) = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots + (-1)^{n-1}\frac{1}{2n-1}\right), \ldots;$$
$$c(1) = 1, c(2) = 1 + \frac{1}{2}, c(3) = 1 + \frac{1}{2} - \frac{1}{8}, \ldots,$$
$$c(n) = 1 + \frac{1}{2} - \frac{1}{8} + \frac{1}{16} + \cdots + (-1)^{n-1}\frac{(2n-3)!!}{2n!!}, \ldots.$$

Ці послідовності є послідовностями раціональних чисел, які збігаються до e, π та $\sqrt{2}$ відповідно. Крім того, кожен елемент цих послідовностей обчислюється алгоритмічно (є раціональним числом).

Два рекурсивних числа α та β будемо вважати рівними і записувати $\alpha = \beta$, якщо для послідовностей раціональних чисел $\{r(n)\}$ та $\{s(n)\}$, які збігаються до α та β відповідно, послідовність

$$\{|r(n) - s(n)|\}$$

збігається до 0.

Теорема 19.1. Проблема рівності двох рекурсивних чисел алгоритмічно нерозв'язна.

Доведення. Нехай $f(n)$ – довільна ПРФ і $\{r(n)\}$ – послідовность раціональних чисел, яка збігається до 0.

Будуємо послідовність $\{s(n)\}$ раціональних чисел наступним чином:

$$s(0) = f(0),\ s(n + 1) = s(n) + f(n + 1).$$

Ця послідовність збігається до 0 тоді і тільки тоді, коли функція $f(n)$ тотожно рівна 0, тобто

$$\lim_{n\to\infty} s(n) = 0 \Leftrightarrow f(n) \equiv 0.$$

Якби проблема рівності двох рекурсивних чисел була б алгоритмічно розв'язною, то алгоритмічно розв'язною була б і проблема тотожної рівності 0 довільної ПРФ. А це не так за теоремою Райса.

Рекурсивне число α будемо називати раціональним (ірраціональним) рекурсивним числом, якщо α є числом раціональним (ірраціональним).

Проблема розпізнавання рекурсивних чисел полягає у можливості побудови алгоритму, який би за довільним рекурсивним числом визначав: є воно раціональним чи ірраціональним.

Теорема 19.2. Проблема розпізнавання рекурсивних чисел алгоритмічно нерозв'язна.

Доведення. Визначимо послідовність $\{s(n)\}$ раціональних чисел:

$$s(0) = 2, s(n+1) = s(n) + \frac{1}{(n+2)!}.$$

Ця послідовність збігається до ірраціонального числа e, тобто

$$\lim_{n\to\infty} s(n) = e.$$

За довільною ПРФ $f(n)$ будуємо послідовність раціональних чисел $\{r(n)\}$, яка визначається наступним чином:

$$r(n) = \begin{cases} s(n), \text{якщо } f(k) = 0 \text{ для всіх } k \le n, \\ s(m), \text{якщо } m - \text{перше значення } k \le n, \text{для якого } f(m) > 0 \end{cases}.$$

Ця послідовність збігається до ірраціонального числа e тоді і тільки тоді, коли коли функція $f(n)$ тотожно рівна 0, тобто

$$\lim_{n\to\infty} r(n) = e \Leftrightarrow f(n) \equiv 0.$$

Якби проблема розпізнавання рекурсивних чисел була б алгоритмічно розв'язною, то алгоритмічно розв'язною була б і проблема тотожної рівності 0 довільної ПРФ. А це не так за теоремою Райса.

Теорема 19.3. Число α є конструктивним тоді і тільки тоді, коли множина

$$A = \{x \in Q, x < \alpha\}$$

алгоритмічно розв'язною.

Доведення. Нехай α раціональне число. Тоді довільне раціональне число x можна порівняти з раціональним числом α (раціональні числа алгоритмічно порівнювані), а, отже, визначити його належність (у випадку $x < \alpha$) чи неналежність (у випадку $x > \alpha$) до множини A.

Нехай α не є раціональним числом. Це означає, що існує послідовність раціональних чисел $\{s(n)\}$, яка збігається до α і кожен елемент цієї послідовності є алгоритмічно обчислюваною функцією від n. Розглянемо алгоритм:

function $\chi(x)$
begin

```
i := 1
χ := 2
while i > 0 & χ = 2 do
    begin
        if x < s(h(1/i)) − 1/i then χ := 0
        if x > s(h(1/i)) + 1/i then χ := 1
        i := i + 1
    end
end,
```

де $h(x)$ – регулятор збіжності послідовності $\{s(n)\}$.

Даний алгоритм обчислює характеристичну функцію множини раціональних чисел A, отже, ця множина є алгоритмічно розв'язною.

Нехай множина

$$A = \{x \in Q, x < \alpha\}$$

є алгоритмічно розв'язною. Покажемо, що в цьому випадку α є рекурсивним дійсним числом. Дійсно, існує послідовність раціональних чисел, яка збігається до α. Кожен член цієї послідовності обчислюється наступним алгоритмом:

```
function s(n)
    begin
        i := 1
        while i > 0 do
            begin
                if χ(r(i) − 1/n) = 0 ∧ χ(r(i) + 1/n) = 1
                then
                    begin
                        k := r(i)
                        i := 0
                    end
                else i := i + 1
            end
        s := k
end,
```

де $\{r(n)\}$ – послідовність всіх раціональних чисел, $\chi(x)$ – характеристична функція множини A.

Крім того, регулятор збіжності $h(x)$ обчислюється алгоритмом

```
function h(ε)
    begin
        i := 1
        while i > 0 do
            begin
                if χ(s(i) – ε) < 0 & χ(s(i) + ε) > 0
                then
                    begin
                        n := i
                        i := 0
                    end
                else i := i + 1
            end
        h := n
    end,
```

де $\chi(x)$ – характеристична функція множини A.

Теорема 19.4. Якщо числа α, β – рекурсивні дійсні числа, то числа

$$\alpha + \beta,\ \alpha - \beta,\ \alpha\beta,\ \frac{\alpha}{\beta}$$

є рекурсивними дійсними числами.

Доведення. Нехай числа α і β є границями послідовностей раціональних чисел $\{s(n)\}$ та $\{p(n)\}$ відповідно. Тоді число $\alpha + \beta$ буде границею послідовності $\{s(n) + p(n)\}$ раціональних чисел. Нехай h_1 і h_2 – регулятори збіжності цих послідовностей. Тоді для довільного раціонального числа ε знаходимо $h_1(\frac{\varepsilon}{2})$ та $h_2(\frac{\varepsilon}{2})$. Отже, члени послідовності $\{s(n)\}$ з номерами $n \geq h_1(\frac{\varepsilon}{2})$ будуть знаходитись на відстані від α меншій $\frac{\varepsilon}{2}$, тобто

$$|s(n) - \alpha| < \frac{\varepsilon}{2},$$

а члени послідовності $\{p(n)\}$ з номерами $m \geq h_1(\frac{\varepsilon}{2})$ будуть знаходитись на відстані від β меншій $\frac{\varepsilon}{2}$, тобто

$$|p(n) - \beta| < \frac{\varepsilon}{2}.$$

Тому,

$$\left|\left(s(k)+p(k)\right)-(\alpha+\beta)\right| = |s(k)-\alpha+p(k)-\beta)| \le$$
$$|s(k)-\alpha| + |p(k)-\beta)| < \frac{\varepsilon}{2}+\frac{\varepsilon}{2} = \varepsilon,$$

де $k = \max(n, m)$.

Отже, достатньо покласти

$$h(\varepsilon) = \max\left(h\left(\frac{\varepsilon}{2}\right), h\left(\frac{\varepsilon}{2}\right)\right).$$

Аналогічно доводиться для інших операцій.

19.2. Конструктивні функції

Конструктивна функція конструктивної дійсної змінної – це алгоритмічно обчислювана функція f: $R \to R$, яка рекурсивне дійсне число α перетворює у рекурсивне дійсне число β, якщо правдива наступна імплікація:

$$\lim_{n\to\infty} r(n) = \alpha \to \lim_{n\to\infty} f\left(r(n)\right) = \beta,$$

тобто, існує алгоритмічно обчислювальна функція δ: $R^+ \to R^+$ така, що для довільного раціонального числа $\varepsilon > 0$ правдива імплікація: якщо всі члени послідовності $\{r(n)\}$ з номерами $n \ge h(\delta(\varepsilon)))$ задовольняють співвідношенню

$$|r(n) - \alpha| < \delta(\varepsilon),$$

то всі члени послідовності $\{f(r(n))\}$ з номерами $n \ge H(\varepsilon)$ задовольняють співвідношенню

$$|f(r(n)) - \beta| < \varepsilon.$$

де послідовність раціональних чисел $\{r(n)\}$ збігається до рекурсивного дійсного числа α, h:$R \to N$ – регулятор збіжності послідовності $\{r(n)\}$, H:$R \to N$ – регулятор збіжності послідовності $\{f(r(n))\}$.

Важливим результатом теорії конструктивних функцій є твердження про те, що будь-яка коструктивна функція є неперервною.

Покажемо, наприклад, що функція Хевісайда

$$\theta(x) = \begin{cases} 0, x \le 0 \\ 1, x > 0 \end{cases}$$

не є конструктивною.

Дійсно, розглянемо послідовність $(-1)^n/n$, яка збігається до 0. Тоді відповідна послідовність значень функції буде розбіжною.

Крім того, справедлива наступна

Теорема 19.5. Якщо послідовність раціональних чисел $\{r(n)\}$ збігається до рекурсивного дійсного числа α і

$$r(n) = \sum_{m \le n} a_m r^{-m}$$

де $\{a_n\}$ – рекурсивна послідовність така, що

$$0 \le a_{n+1} \le r - 1,\ r \ge 2,$$

то

$$\alpha = \sum a_m r^{-m}$$

тобто, α рекурсивно розкладається в ряд.

Ця теорема лежить в основі різних методів обчислення конструктивних дійсних чисел та конструктивних функцій.

20

Доведення правильності алгоритмів

Більшість з розглянутих раніше результатів була одержана шляхом побудови алгоритмів на мові ПсевдоPackal. Зрозуміло, що питання правильності цих результатів залежить від правильності запропонованих алгоритмів. Отже, виникає необхідність в розробці методів доведення їх правильності.

Будемо розглядати інструкції мови ПсевдоPackal як функцію зміни стану обчислень. Наприклад, якщо в стані s із значенням змінної x рівним 8, виконати інструкцію

$$x := 1 + 2x,$$

то одержимо стан s', в якому значення змінної x рівне 17.

Якщо програма містить n змінних (x_1, ... , x_n), то стан s визначається n-кою (a_1, … , a_n), де a_i є значення змінної x_i.

Нехай U є множиною всіх n-ок, утворених з елементів деякої множини M і $U' \subseteq U$.

Характеристичний предикат $P_{U'}(x_1,...,x_n)$ множини U' визначається як

$$P_{U'}(x_1,...,x_n) = \begin{cases} 0, (x_1,...,x_n) \in U' \\ 1, (x_1,...,x_n) \notin U' \end{cases}.$$

Приклад. Нехай U є множиною всіх пар цілих чисел, а U' визначається як множина пар виду

...
… (-2,-3), (-2,-2), (-2,-1), (-2,0), (-2,1), (-2,2), (-2,3)
… (-1,-3), (-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2), (-1,3)
… (0,-3), (0,-2), (0,-1), (0,0), (0,1), (0,2), (0,3)
… (1,-3), (1,-2), (1,-1), (1,0), (1,1), (1,2), (1,3)
… (2,-3), (2,-2), (2,-1), (2,0), (2,1), (2,2), (2,3)
...

Характеристичним предикатом множини U' є

$$P_{U'}(x_1, x_2) = (x_1 = x_1) \wedge (x_2 \le x_3).$$

Приклад. Інструкція

$$x := 2y + 1$$

перетворює стан (x, y) із множини станів $\{(x, y), y \le 3\}$ в стан ($2y + 1$, y) із множини станів $\{(x, y) \mid (x \le 7) \wedge (y \le 3)\}$ оскільки правдива імплікація

$$(y \le 3) \to (2y + 1 \le 7).$$

Твердженням будемо називати трійку

$$\{p\}\boldsymbol{S}\{q\},$$

де $\boldsymbol{S}$ є алгоритмом, а p і q є предикатами, які називаються початковою і кінцевою умовами відповідно.

Твердження називається також трійкою Хоара.

Твердження називається правдивим в тому і тільки в тому випадку, коли виконується наступна умова: якщо алгоритм $\boldsymbol{S}$ починає виконуватись в стані, який задовольняє p і обчислення завершуються, то вони завершуються в стані, який задовольняє q.

Правдиве твердження записується у вигляді

$$\vdash \{p\}\ \boldsymbol{S}\ \{q\}.$$

При цьому алгоритм $\boldsymbol{S}$ називається частково правильним відносно p і q.

Приклад. $\vdash \{y \le 3\}\ x := 2y + 1\ \{(x \le 7) \wedge (y \le 3)\}$.

Приклад. Для кожного алгоритму $\boldsymbol{S}$ і будь-якого q правдие твердження

$$\{false\}\ \boldsymbol{S}\ \{q\},$$

оскільки не існує стану, який задовольняє *false*.

Аналогічно для будь-яких p і $\boldsymbol{S}$ правдивим буде твердження

$$\{p\}\ \boldsymbol{S}\ \{true\}.$$

Формальне доведення правильності алгоритму на мові ПсевдоPackal полягає у визначенні для кожної інструкції мови і кінцевої умови найслабшої початкової умови, виконання якої буде гарантувати виконання кінцевої умови після виконання інструкції.

Приклад. Нехай правдиве твердження має вигляд

$$\vdash \{y \le 3\}\ x := 2y + 1\ \{(x \le 7) \wedge (y \le 3)\}.$$

Умова $y \le 3$ не є єдиною умовою, яка гарантує виконання кінцевої умови. Іншою такою умовою може бути умова $y = 1 \vee y = 3$.

Зрозуміло, що умова $y = 1 \vee y = 3$ не описує всіх станів, з яких можна перейти після виконання інструкції до стану, що задовольняє кінцевій умові. Ми хочемо знайти таку початкову умову, яка б охоплювала максимально можливу кількість станів.

Говорять, що формула A слабкіша, ніж формула B, якщо правдива імплікація

$$B \to A.$$

Для заданої множини формул $\{A_1, A_2, \dots\}$ формула A є найслабшою, якщо правдива імплікація

$$A_j \to A$$

для кожного j.

Приклад. Формула $y \le 3$ є слабшою ніж формула $y = 1 \vee y = 3$, оскільки правдива імплікація $(y = 1 \vee y = 3) \to (y \le 3)$. Аналогічно, формула $y = 1 \vee y = 3$ є слабшою ніж формула $y = 1$.

Отже, чим слабший предикат, тим більше станів його задовольняє. Зрозуміло, також, що завжди можна посилити антецедент і послабити консеквент, тобто, якщо правдива імплікація

$$p \to q,$$

то правдивими будуть і імплікації

$$(p \wedge r) \to q \text{ та } p \to (q \vee r).$$

Для алгоритму $\boldsymbol{S}$ і формули q найслабшою початковою умовою називається найслабша формула p, для якої

$$\vdash \{p\}\boldsymbol{S}\{q\}.$$

Найслабша початкова умова позначається через $nww(\boldsymbol{S}, q)$. З визначення найслабшої початкової умови випливає

Твердження. $\vdash \{p\}\boldsymbol{S}\{q\} \Leftrightarrow \vdash p \to nww(\boldsymbol{S}, q)$.

Приклад. $nww(x := 2y + 1, (x \le 7) \wedge (y \le 3)) = y \le 3$.

Найслабша початкова умова залежить як від алгоритму $\boldsymbol{S}$, так і від кінцевої умови q. Якщо в останньому прикладі змінити кінцеву умову на $x \le 9$, то найслабшою початковою умовою буде предикат $y \le 4$. Аналогічно, якщо змінити інструкцію на $x := y + 2$, то найслабшою початковою умовою буде предикат $y \le 5$.

Наступні визначення задають правила для знаходження найслабших початкових умов для інструкцій мови ПсевдоPascal і кінцевих умов:

1. $nww(x := t, p(x)) = p(x)\{x \leftarrow t\}$;
2. $nww(\mathbf{S1}; \mathbf{S2}, q) = nww(\mathbf{S1}, nww(\mathbf{S2}, q))$;
3. $nww(\text{if } B \text{ then } \mathbf{S1} \text{ else } \mathbf{S2}, q) = (B \to nww(\mathbf{S1}, q)) \wedge \neg B \to nww(\mathbf{S2}, q))$;
4. $nww(\text{while } B \text{ do } \mathbf{S}, q) = (\neg B \wedge q) \vee (B \wedge nww(\mathbf{S}; \text{ while } B \text{ do } \mathbf{S}, q))$.
5. $nww(\text{for } i := t_1 \text{ to } t_2 \{\mathbf{S}\}, q) = (\neg(t_1 \le t_2) \wedge q) \vee ((t_1 \le t_2) \wedge nww(\mathbf{S}\{i \leftarrow t_1\}, nww(\text{for } i := t_1 + 1 \text{ to } t_2 \{\mathbf{S}\}, q)))$.

Приклади 1) $nww(y := y - 1, y \ge 0) = (y - 1 \ge 0)$;

2) $nww(x := x + 1; y := y + 2, x < y) =$
$= nww(x := x + 1, nww(y := y + 2, x < y)) =$
$= nww(x := x + 1, x < y + 2) = x + 1 < y + 2 = x < y + 1$;

3) $nww(\text{if } y = 0 \text{ then } x := 0 \text{ else } x := y + 1, x = y) =$
$= (y = 0 \to nww(x := 0, x = y)) \wedge (y \ne 0 \to nww(x := y + 1, x = y)) =$
$= ((y = 0) \to (y = 0)) \wedge ((y \ne 0) \to (y + 1 = y)) =$
$= true \wedge ((y \ne 0) \to false) =$
$= \neg(y \ne 0) = y = 0$;

4) $nww(\text{while } x > 0 \text{ do } x := x - 1, x = 0) =$
$= (\neg(x > 0) \wedge (x = 0)) \vee ((x > 0) \wedge nww(x := x - 1; \text{ while } x > 0 \text{ do } x := x - 1, x = 0)) =$
$= (x = 0) \vee ((x > 0) \wedge nww(x := x - 1, nww(\text{while } x > 0 \text{ do } x := x - 1, x = 0)))$
$= (x = 0) \vee ((x > 0) \wedge nww(\text{while } x > 0 \text{ do } x := x - 1, x = 0)\{x \leftarrow x - 1\}) =$
$= (x = 0) \vee (x = 1) \vee ((x > 1) \wedge nww(\text{while } x > 0 \text{ do } x := x - 1, x = 0)\{x \leftarrow x - 1\}\{x \leftarrow x - 1\}) =$
$= (x = 0) \vee \{x = 1) \vee (x = 2) \vee ... = (x \ge 0)$;

5) $nww(\text{for } i := 1 \text{ to } 2 \{s := s + 1\}, s = 3) =$
$= (\neg(1 \le 2) \wedge s = 3)) \vee ((1 \le 2) \wedge (nww(s := s + 1, nww(\text{for } i := 2 \text{ to } 2 \{s := s + 1\}, s = 3)) =$
$= (nww(s := s + 1; s := s + 1, (\neg(2 \le 2) \wedge s = 3)) \vee ((2 \le 2) \wedge (nww(\text{for } i := 3 \text{ to } 2 \{s := s + 1\}, s = 3)) =$
$= (nww(s := s + 1; s := s + 1, (nww(\text{for } i := 3 \text{ to } 2 \{s := s + 1\}, s = 3)) =$
$= (nww(s := s + 1; s := s + 1, (\neg(3 \le 2) \wedge s = 3)) \vee ((3 \le 2) \wedge (nww(\text{for } i := 4 \text{ to } 2 \{s := s + 1\}, s = 3)) =$
$= nww(s := s + 1; s := s + 1, s = 3)) =$
$= s = 1$.

Розглянемо кілька прикладів доведення правильності алгоритмів на мові ПсевдоPascal за допомогою знаходження найслабшої початкової умови для заданої кінцевої умови.

Нехай треба побудувати алгоритм, який міняє місцями вартості змінних x та y. Якщо ввести додаткову змінну z, то такий алгоритм може бути наступним:

```
function f(x, y)
    begin
        z := x
        x := y
        y := z
    end.
```

Зрозуміло, що доводити правильність такого алгоритму немає потреби.

Розглянемо інший алгоритм:

```
function f(x, y)
    begin
        x := x + y
        y := x − y
        x := x − y
    end.
```

Цей алгоритм теж міняє місцями вартості змінних x та y. Але, на відміну від попереднього алгоритму, виникає необхідність доведення його правильності. Для цього задаємо кінцеву умову $x = 2 \wedge y = 3$ і знаходимо найслабшу початкову умову, тобто маємо:

$$
\begin{aligned}
&nww(x := x + y;\ y := x - y;\ x := x - y,\ x = 2 \wedge y = 3) = \\
&= nww(x := x + y;\ y := x - y,\ nww(x := x - y,\ x = 2 \wedge y = 3)) = \\
&= nww(x := x + y;\ y := x - y,\ x - y = 2 \wedge y = 3) = \\
&= nww(x := x + y,\ nww(y := x - y,\ x - y = 2 \wedge y = 3)) = \\
&= nww(x := x + y,\ y = 2 \wedge x - y = 3) = \\
&= y = 2 \wedge x = 3.
\end{aligned}
$$

Отже, алгоритм міняє місцями вартості змінних x та y, тобто, працює правильно.

Для обчислення суми натуральних чисел від 1 до x ($x \geq 1$) може бути побудований алгоритм

```
function f(x)
    begin
        s := 0
        for i := 1 to x
            s := s + i
        f := s
```

end.

Для доведення його правильності задамо кінцеву умову, наприклад, $s = 3$ і знайдемо найслабшу початкову умову. Маємо:

nww(for $i := 1$ to 2 $\{s := s + i\}$, $s = 3$) =
$= (\neg(1 \le 2) \wedge s = 3) \vee ((1 \le 2) \wedge (nww(s := s + 1, nww(\text{for } i := 2 \text{ to } 2 \{s := s + i\}, s = 3))) =$
$= nww(s := s + 1, nww(\text{for } i := 2 \text{ to } 2 \{s := s + i\}, s = 3) =$
$= (nww(s := s + 1; s := s + 2, (\neg(2 \le 2) \wedge s = 3)) \vee ((2 \le 2) \wedge (nww(\text{for } i := 3 \text{ to } 2 \{s := s + i\}, s = 3))) =$
$= nww(s := s + 1; s := s + 2, (nww(\text{for } i := 3 \text{ to } 2 \{s := s + i\}, s = 3)) =$
$= nww(s := s + 1; s := s + 2, (\neg(3 \le 2) \wedge s = 3)) \vee ((3 \le 2) \wedge (nww(\text{for } i := 4 \text{ to } 2 \{s := s + i\}, s = 3))) =$
$= nww(s := s + 1; s := s + 2, s = 3) =$
$= s = 0.$

Можна тепер зробити висновок про те, що сума перших двох натуральних чисел ($x = 2$) дорівнює 3, а, отже, алгоритм працює правильно. В загальному випадку можна знайти найслабшу початкову умови виду для кінцевої умови $s = 1 + 2 + \ldots + x$. Будемо мати:

nww(for $i := 1$ to x $\{s := s + i\}$, $s = 1 + 2 + \ldots + x$) =
$= (\neg(1 \le x) \wedge s = 1 + 2 + \ldots + x) \vee ((1 \le x) \wedge (nww(s := s + 1, nww(\text{for } i := 2 \text{ to } 2 \{s := s + i\}, s = 1 + 2 + \ldots + x))) =$
$= (\neg(1 \le x) \wedge s = 1 + 2 + \ldots + x) \vee ((1 \le x) \wedge (nww(s := s + 1, (\neg(2 \le x) \wedge s = 1 + 2 + \ldots + x) \vee ((2 \le x) \wedge (nww(s := s + 2, nww(\text{for } i := 3 \text{ to } 2 \{s := s + i\}, s = 1 + 2 + \ldots + x)))) = \ldots =$
$= (\neg(1 \le x) \wedge s = 1 + 2 + \ldots + x) \vee ((1 \le x) \wedge (nww(s := s + 1, (\neg(2 \le x) \wedge s = 1 + 2 + \ldots + x) \vee ((2 \le x) \wedge (nww(s := s + 2, nww(\text{for } i := 3 \text{ to } 2 \{s := s + i\}, s = 1 + 2 + \ldots + x)))) \wedge ... \wedge (nww(s := s + x, nww(\text{for } i := x + 2 \text{ to } x \{s := s + i\}, s = 1 + 2 + \ldots + x) ...) =$
$= nww(s := s + 1; s := s + 2, \ldots, s := s + x, s = 1 + 2 + \ldots + x) =$
$= s = 0.$

Таким чином, сума перших x натуральних чисел приведеним алгоритмом обчислюється правильно.

20.1. Інваріанти циклу

Розглянемо алгоритм:

function $f(m, n)$

```
begin
    q := 0
    r := m
    while r ≥ n do
        begin
            q := q + 1
            r := r – n
        end
end.
```

Це алгоритм ділення цілого числа $m \geq 0$ на ціле число $n > 0$. Тобто, результатами роботи алгоритму будуть цілі q та r такі, що $q \cdot n + r = m$ і $0 \leq r < n$.

Ми хочемо дати відповіді на два питання:

1) Чи алгоритм зупиняється?

2) Чи алгоритм правильно обчислює частку і остачу при діленні m на n (при умові, що він зупиняється)?

На початку роботи алгоритму $r = m \geq 0$. Кожне виконання циклу зменшує r на n. Отже, маємо послідовність m, $m - n$, $m - 2 \cdot n$, … значень змінної r. Зрозуміло, що в цій послідовності буде існувати елемент $m - k \cdot n$ такий, що $m - k \cdot n \geq n$, а $m - (k + 1) \cdot n < n$. Це означає, що ми вийдемо з циклу і алгоритм завершить роботу.

Предикат p будемо називати інваріантом циклу

$$\text{while } B \text{ do } \boldsymbol{S},$$

якщо виконується наступна умова:

якщо предикати p і B (умова циклу) правдиві, то після виконання алгоритму $\boldsymbol{S}$ предикат p залишається правдивим.

З цього визначення безпосередньо випливає, що якщо інваріант циклу правдивий при входженні в цикл, то він залишається правдивим після кожної ітерації циклу і, при умові завершення циклу, інваріант циклу буде правдивим, а умова циклу фальшива.

Покажемо, що інваріантом циклу в алгоритмі ділення, приведеному вище, є предикат

$$q \cdot n + r = m \wedge r \geq 0.$$

Дійсно, предикат правдивий при входженні в цикл. Нехай предикат правдивий і виконується умова $r \geq n$. В цьому випадку ми входимо в цикл і одержуємо нові вартості r' і q', причому

$$q' \cdot n + r' = (q + 1) \cdot n + (r - n) = q \cdot n + r = m,$$
$$r' = r - n \geq 0.$$

Отже, предикат правдивий при виконанню цикла. Крім того, алгоритм завершує роботу коли предикат $q \cdot n + r = m \wedge r \geq 0$ правдивий, а умова циклу $r \geq n$ фальшива, тобто $r < n$.

Таким чином, це саме ті умови, яким повинні задовольняти результати роботи алгоритму ділення. Виконання цих умов можна дослідити шляхом побудови інваріанта циклу, а, отже, за його допомогою довести правильність алгоритму.

Для обчислення суми натуральних чисел від 1 до x ($x \geq 1$) може бути побудований алгоритм, який використовує цикл, наприклад:

```
function f(x)
    begin
        s := 0
        i := 0
        while i < x do
            begin
                i := i + 1
                s := s + i
            end
        f := s
    end.
```

Покажемо, що інваріантом циклу в цьому алгоритмі є предикат

$$i \leq x \wedge s = 1 + 2 + \ldots + i.$$

Дійсно, предикат правдивий при входженні в цикл. Нехай предикат правдивий і виконується умова $i < x$. В цьому випадку ми входимо в цикл і одержуємо нові вартості i' і s', причому

$$i' = i + 1 \leq x,$$
$$s' = 1 + 2 + \ldots + i + (i + 1).$$

Отже, предикат правдивий при виконанню цикла. Крім того, алгоритм завершує роботу коли предикат $i \leq x \wedge s = 1 + 2 + \ldots + i$ правдивий, а умова циклу $i < x$ фальшива, тобто $i \geq x$. Тобто, цей алгоритм обчислює суму натуральних чисел від 1 до x ($x \geq 1$).

Список літератури

1. А.И. Мальцев. Алгоритмы и рекурсивные функции. – Москва. – Наука, 1965. – 392 с.

2. В.А. Успенский, А.Л. Семенов. Теория алгоритмов: основные открытия и приложения. – Москва. – Наука, 1987. – 288 с.

3. Н.К. Верещагин, А.Шень. Лекции по математической логике и теории алгоритмов. Часть 2. Языки и исчисления. – М: МЦНМО, 2000. – 288 с.

4. Н.К. Верещагин, А.Шень. Лекции по математической логике и теории алгоритмов. Часть 3. Вычислимые функции. – М: МЦНМО, 1999. – 176 с.

5. Дж. Булос, Р. Джефри. Вычислимость и логика. – М: Мир, 1994. – 396 с.

6. Н. Катленд. Вычислимость. Ведение в теорию рекурсивных функций. – М: Мир, 1983. – 256 с.

7. С.К Клини. Введение в метаматематику. – М: Издательство иностренной литературы, 1957. – 526 с.

8. Х. Роджерс. Теория рекурсивных функций и эффективная вычислимость. – М: Мир, 1972. – 624 с.

9. Справочная книга по математической логике в четырех частях под редакцией Дж. Барвайса. Часть 3. Теория рекурсии. – М: Наука, 1982. – 360 с.

10. П. Мартин-Леф. Очерки по конструктивной математике. – М: Мир, 1975. – 136 с.

11. Р.Л. Гудстейн. Рекурсивный математический анализ. – М: Наука, 1970. – 472 с.

12. Б.А. Кушнер. Лекции по конструктивному математическому анализу. – М: Наука, 1973. – 446 с.

13. M. Ben-Ari. Logika matematyczna w informatyce. – Warszawa: Wydawnictwo Naukowo-Techniczne, 2005. – 344 s.

14. Д. Хопкрофт, Р. Мотвани, Д. Ульман. Введение в теорию автоматов, языков и вычислений. – М: “Вильямс”, 2002. – 528 с.

15. Л.П. Лисовик. Теория трансдьюсеров. Том 3. Книга 5. – К: Феникс, 2006. – 318 с.

16. Л.П. Лисовик. Теория трансдьюсеров. Том 1. Книга 1. – К: Феникс, 2005. – 272 с.

17. С.Л. Кривий. Вступ до методів створення програмних продуктів. – Київ-Чернівці: Букрек, 2012. – 424 с.

18. О.І. Провотар, О.С. Шкільняк. Приклади й задачі з теорії алгоритмів та

математичної логіки. . – К: ВПС “Київський університет”, 2012. – 168 с.

19. М.С. Нікітченко, С.С. Шкільняк. Математична логіка та теорія алгоритмів. – К: ВПС “Київський університет”, 2008. – 528 с.