ЛЕКЦІЯ 11

Формалізація поняття алгоритму.

Універсальні моделі алгоритмів

Формалізація поняття алгоритму. Універсальні моделі алгоритмів.

Інтуїтивні визначення алгоритму:

Алгоритм — зрозумілі і точні інструкції виконавцеві зробити скінченне число елементарних кроків, спрямованих на розв'язування поставленої задачі.

«**Алгоритм** — це будь-яка система обчислень, які виконуються по строго визначених правилах, що після деякого числа елементарних кроків явно (завідомо) приводить до розв'язку поставленої задачі». (А. Колмогоров)

Недолік інтуїтивного визначення:

Неточне визначення властивості «Елементарність кроків»

Формальний підхід до визначення алгоритмів у теорії алгоритмів:

Вибирається

1. скінченний набір вхідних об'єктів, які оголошуються елементарними;
2. скінченний набір способів побудови з них нових об'єктів.

Вимога до такої формалізації алгоритмів – **універсальність.**

*Основні типи у*ніверсальних алгоритмічних моделей обчислень:

1. *Рекурсивні функції.*
2. *Машина Тьюринга.*
3. *Нормальні алгоритми Маркова*.

Рекурсивні функції, машина Тьюринга, нормальні алгоритми Маркова — універсальні алгоритмічні моделі обчислень.

1. Перший тип алгоритмічних моделей

– *рекурсивні функції* —

є історично першою формалізацією поняття алгоритму.

Використовується для позначення трьох класів функцій: *примітивно рекурсивних*, *частково рекурсивних і загальнорекурсивних*.

2. Другий тип алгоритмічних моделей

Основна модель цього типу — *машина Тьюринга*.

Ця модель представляє алгоритм як деяке детерміноване обладнання, що виконує примітивні операції, які однозначно визначають алгоритм як послідовність елементарних кроків.

3. Третій тип алгоритмічних моделей

– це перетворення слів у довільних алфавітах.

Переваги

1. Максимальна абстрактність
2. Можливості застосувати поняття алгоритму до об'єктів довільної природи.

Прикладами моделей цього типу є *канонічні системи Поста і нормальні алгоритми Маркова*.

Будь-який алгоритм, описаний засобами однієї моделі, може бути описаний засобами іншої.

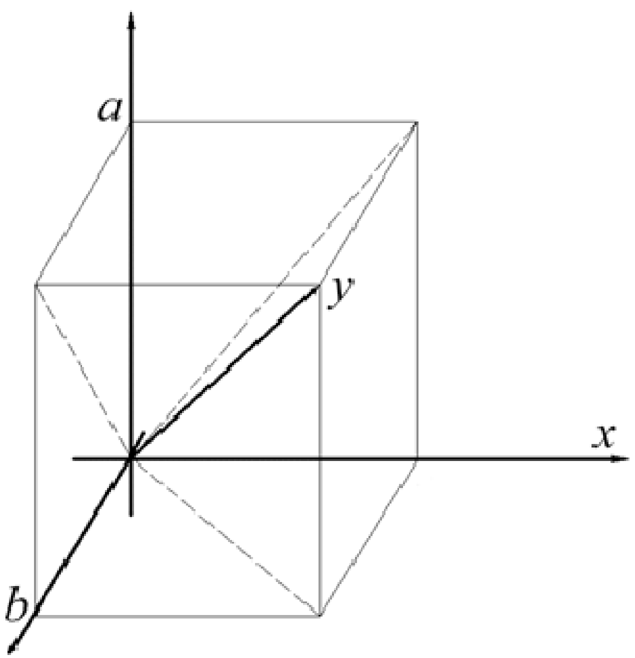
Завдяки взаємному зведенню моделей у загальній теорії алгоритмів вироблена система понять, що визначає властивості алгоритмів

Вона заснована на понятті обчислюваної функції, тобто функції, для обчислення якої існує алгоритм.

Обчислювана функція

Функція f називається **обчислюваною функцією**, якщо існує **алгоритм**, що перетворює будь-який об'єкт x, для якого визначена функція f, в об'єкт $f(x)$, і не застосовний до жодного x, для якого f не визначена.

Приклад.



Задана множина натуральних чисел N і деяка функція

$$y = f(x, a, b) = ax + b.$$

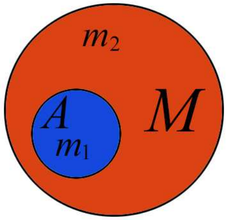
Функція $f : N^3 \to N$ обчислювана, тому що існує алгоритм $f : (a, b, x) \to y$, який по будь-якій трійці (a, b, x) обчислює y.

Розв’язність (можливість розв'язання)

Множина A ***розв'язна*** на M, тобто для $A \subseteq M$ існує алгоритм, який для кожного $m \in M$ визначає:

Чи належить m до A, тобто $m \in A$,

чи не належить m до A, тобто $m \notin A$.



Проблема **розв’язності** (*можливості розв'язання*) зводиться до обчислення відповідного предиката:

$$P(m) = \begin{cases} 1,\ true,\ m \in A, \\ 0,\ false,\ m \notin A. \end{cases}$$

Приклад .

$$\boxed{P(\{a,b,x,y\}) = \begin{cases} true,\ y = ax + b \\ false, y \neq ax + b \end{cases}}$$

Нехай A – множина розв'язків рівняння $y = ax + b$.

$$A = \{(x, y, a, b) | y = ax + b\}$$

Існує алгоритм Π перевірки для кожної четвірки (a, b, x, y) у вигляді підстановки цих чисел у рівняння і виконання зазначених у рівнянні дій, який ***обчислює*** предикат і в такий спосіб вирішує проблему ***розв’язності*** для заданого рівняння.

Приклад. **Розглянемо рівняння** $y = ax^2 + bx + c$

Обчислюваність.

Для визначення обчислюваності сформуємо алгоритм

$$F:(a,b,c,x,)\to y$$

```
for a in range(n):
 for b in range(m):
  for c in range(k):
   for x in range(t):y[a][b][c][x]=a*x*x+b*x+c
```

Розв'язність.

Для визначення розв'язності сформуємо алгоритм обчислення предиката:

$$P\left(\left\{a,b,c,x,y\right\}\right)=\begin{cases} true,\ y=ax^2+bx+c \\ false, y\neq ax^2+bx+c \end{cases}$$

Застосувавши вкладені цикли по y,a,b,c,x *виконаємо:*

```
if y==(a*x*x+b*x+c): P=True else P=False
```

Приклад. Теорема Ферма.

Чи можна для рівняння $x^n + y^n = z^n$, де $x, y, z, n \in N$, запропонувати алгоритм Π, що породжує всі цілочисельні розв'язки рівняння при заданому n?

Обчислюваність. Для визначення обчислюваності необхідно знайти алгоритм:

$$F : (x, y, n) \to z$$

Знайти $z = \sqrt[n]{x^n + y^n}$ при $x, y, z, n \in N$.

Розв’язність. Існує тривіальний алгоритм, що обчислює предикат. $P\left(\left\{a, b, c, x, y\right\}\right) = \begin{cases} true, \ z^n = x^n + y^n \\ false, z^n \neq x^n + y^n \end{cases}$

Наприклад,

$\Pi\left(1,1,2,1\right) \to true$, оскільки $1^1 + 1^1 = 2^1$

$\Pi\left(1,1,2,2\right) \to false$, оскільки $1^2 + 1^2 \neq 2^2$

Отже, множина розв'язків рівняння $x^n + y^n = z^n$ ***розв'язна.***

Хоча обчислюваність для теореми доведена, але алгоритм настільки складний, що поки не допускає практичного застосування.

$$z = f(x, y, n) \to f\,?$$

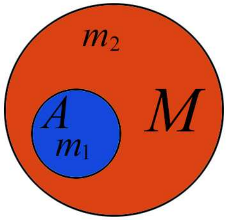
Приклад. Чи має рівняння $x^3 + y^3 + z^3 = 29$ цілий розв'язок?
1. Легко перевірити ***(обчислити предикат),*** що розв'язок існує – $(x, y, z) = (3, 1, 1)$.
2. Рівняння $x^3 + y^3 + z^3 = 30$ також має розв'язок, знайдений в 1990 році
(–283059965, –2218888517, 2220422932).

3. Для рівняння $x^3 + y^3 + z^3 = 33$ розв'язки невідомі.

Ці задачі пов'язані з 10 проблемою Гільберта і в 1970 році Ю. Матіясевич довів, що ***не існує алгоритму*** знаходження розв'язків таких задач!

Взаємозв'язок між обчислюваністю і розв’язністю (можливістю розв'язання):

1) якщо A ***обчислювана*** в M, то A ***розв'язна*** в M;



2) якщо A ***розв'язна*** в M, то із цього не випливає її ***обчислюваність***.

Проблеми **розв’язності і обчислюваності** множин вимагають більш точного визначення типу алгоритму.

Рекурсивні функції
(Перший тип алгоритмічних моделей)

Рекурсивна функція – це така функція, для якої значення аргументів визначаються раніше вже обчисленими значеннями функції або значеннями більш простих функцій.

Приклад. Прикладом рекурсивного визначення є числа Фіббоначі, що представляють собою послідовність чисел $f\left(n\right)$, що задовольняють умовам:

$$f\left(0\right)=1, \quad f\left(1\right)=1, f\left(2\right)=2, \ f\left(n+2\right)=f\left(n\right)+f\left(n+1\right),$$

тобто числа $1, 1, 2, 3, 5, 8, 13 \ldots$

Кожне наступне число є сумою двох попередніх чисел.

Визначимо **конкретний набір засобів**, за допомогою яких відбувається побудова обчислюваних функцій.

Примітивно-рекурсивні функції

Запропоновані Геделем усюди визначені числові функції назвемо найпростішими:

1) $0(x)=0$ – нуль-функція.

2) $S(x)=x+1$ – функція слідування.

3) $I_m^n\left(x_1,\ldots,x_m,\ldots,x_n\right) = x_m$ – функція проектування, де $m = 1,\ldots,n$.

Ці функції обчислювані інтуїтивному сенсі.

Приклад.

$0(1)=0$, $0(2)=0$, $0(3)=0$,……, $0(n)=0$.

$S(1)=1+1=2$, $S(2)=2+1=3$,….., $S(n)=n+1$.

$I_3^5\left(5,4,6,1,8\right) = 6$

Оператори

Застосовуючи їх до функцій, обчислюваних в інтуїтивному сенсі, одержуємо функції, також обчислювані в інтуїтивному сенсі.

Оператор суперпозиції

Суперпозиція — підстановка функцій у функції для одержання нових функцій із уже наявних.

Оператором суперпозиції P_m^n називається підстановка у функцію від m змінних m функцій від n тих самих змінних.

Дана m-місна функція: $h(x_1, x_2, \ldots, x_m)$ і m функцій від n змінних: $g_1(x_1, x_2, \ldots, x_n), \ldots, g_m(x_1, x_2, \ldots, x_n)$

Застосуємо до цих функцій оператор суперпозиції:

$$P_m^n(h, g_1, g_2, \ldots, g_m) =$$

$$= h(g_1(x_1, x_2, \ldots, x_n), \ldots, g_m(x_1, x_2, \ldots, x_n)) = f(x_1, x_2, \ldots, x_n)$$

Приклад. $y = h(x, y, z) = x^2 + 2y + z$, $a = x - y$, $b = x + y$, $c = -x^2 - y^2$,

$$P_3^2(h, a, b, c) = a^2 + 2b + c = x^2 - 2xy + y^2 + 2x + 2y - x^2 - y^2 = 2(x + y - xy)$$

У цьому випадку говорять, що *n-місна* функція $f\left(x_1, x_2, \ldots, x_n\right)$ отримана за допомогою оператора суперпозиції з *m-місної* функції $h\left(x_1, x_2, \ldots, x_m\right)$ і *m n-місних* функцій

$g_1\left(x_1, x_2, \ldots, x_n\right), \ldots, g_m\left(x_1, x_2, \ldots, x_n\right)$, якщо

$$f\left(x_1, x_2, \ldots, x_n\right) = h\left(g_1\left(x_1, x_2, \ldots, x_n\right), \ldots, g_m\left(x_1, x_2, \ldots, x_n\right)\right).$$

Приклад одержання нових функцій з базових функцій Геделя

Одержати константну функцію $f(x) = a, a \in N$,

за допомогою суперпозиції функцій $0(x) = 0$ і $s(x) = x + 1$.

$$s(x) = 0(x) + 1 = 1$$

$$s\left(s(x)\right) = \left(0(x) + 1\right) + 1 = 2$$

$$s\left(s\left(s(x)\right)\right) = \left(\left(0(x) + 1\right) + 1\right) + 1 = 3$$

$$\overbrace{s\left(s\left(\ldots s\right.\right.}^{a}\left(0(x)\right)\ldots\left.\left.\right)\right) = a$$

Оператор примітивної рекурсії

Оператор примітивної рекурсії R_n визначає $(n+1)$-місну функцію f через *n-місну* функцію g і $(n+2)$-місну функцію h у такий спосіб:

$$\left.\begin{aligned} &f(x_1,\ldots,x_n,0)=g(x_1,x_2,\ldots x_n);\\ &f(x_1,\ldots,x_n,y+1)=h(x_1,x_2,\ldots,x_n,y,f(x_1,x_2,\ldots,x_n,y)) \end{aligned}\right\}(1)$$

У випадку, коли $n=0$, тобто обумовлена функція f є одномісною, схема (1) набуває більш простого вигляду:

$$\begin{cases} f(0)=C;\\ f(y+1)=h(y,f(y)). \end{cases}$$ де C – константа. (2)

Схема для $n > 1$

$$f\left(x_1, \ldots, x_n, 0\right) = g\left(x_1, x_2, \ldots x_n\right)$$

$$f\left(x_1, \ldots, x_n, y+1\right) = R_n\left(g, h\right)$$

визначає $f\left(x_1, \ldots, x_n, y+1\right)$ рекурсивно через *g* і *h*,

та схема для $n = 0$

$$f\left(0\right) = C; \quad f\left(y+1\right) = h\left(y, f\left(y\right)\right)$$

Визначає $f\left(y+1\right)$ рекурсивно через y та $f\left(y\right)$.

Приклад – обчислення $n!$.

$$f\left(0\right) = 1$$
$$f\left(1\right) = f\left(0\right) \cdot 1 = 1$$
$$f\left(2\right) = f\left(1\right) \cdot 2 = 2$$
$$f\left(y+1\right) = f\left(y\right) \cdot \left(y+1\right)$$

Для обчислення $f\left(x_1, x_2, \ldots, x_n, k\right)$ знадобиться $k+1$ обчислень для $y = 0, 1, \ldots, k$.

Особливість оператора примітивної рекурсії:

Незалежно від числа змінних у функції f рекурсія ведеться тільки по одній змінній y, інші n змінних $x_1, x_2, \ldots, x_n$ на момент застосування схем (1) і (2) зафіксовані і відіграють роль параметрів.

Функція називається примітивно рекурсивною, якщо вона може бути отримана з нуль-функції $0\left(x\right)$, функції слідування $S\left(x\right)$ та функції проектування I_m^n за допомогою скінченного числа застосувань операторів суперпозиції і примітивної рекурсії.

Точне визначення примітивно-рекурсивної функції

1. Функції $0(x)$, $S(x)$ і $I_m^n(x)$ для всіх натуральних n, m, де $m \le n$, є примітивно рекурсивними.

2. Якщо $g_1(x_1, x_2, \ldots, x_n), \ldots, g_m(x_1, x_2, \ldots, x_n)$, примітивно рекурсивні, то оператор суперпозиції $P_m^n(h, g_1, \ldots, g_m)$ дає примітивно-рекурсивні функції для будь-яких натуральних n, m.

3. Якщо $g(x_1, \ldots, x_n)$ і $h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y))$ – примітивно рекурсивні функції, то оператор рекурсії $R_n(g, h)$ дає примітивно рекурсивні функції.

4. Інших примітивно рекурсивних функцій немає.

РОЗГЛЯНЕМО ДІЮ ОПЕРАТОРА ПРИМІТИВНОЇ РЕКУРСІЇ НА ПРИКЛАДАХ

Операція примітивної рекурсії в загальному вигляді (з параметрами)

Будемо говорити, що функція

$$f\left(x_1, x_2, ..., x_n, y+1\right)$$

отримана з функцій g і h у результаті застосування операції примітивної рекурсії, якщо:

1) $n \neq 0$

$$\begin{cases} f\left(x_1, ..., x_n, 0\right) = g\left(x_1, ... x_n\right) \\ f\left(x_1, ..., x_n, y+1\right) = h\left(x_1, ..., x_n, y, f\left(x_1, ... x_n, y\right)\right) \end{cases}$$

(схема примітивної рекурсії з параметрами)

Операція примітивної рекурсії (окремий випадок) (без параметрів)

Будемо говорити, що функція

$$f(y)$$

отримана з функцій g і h у результаті застосування операції примітивної рекурсії, якщо:

$$2)\ n = 0, \quad g \in N$$

$$\begin{cases} f(0) = g \\ f(y+1) = h\left(y, f(y)\right) \end{cases}$$

(схема примітивної рекурсії без параметрів)

Порядок застосування операції примітивної рекурсії

Крок 1. Початкова умова.

$$f\left(x_1,\dots,x_n,0\right)=g\left(x_1,\dots x_n\right)$$

Крок 2. Рекурсивний крок.

$$f\left(x_1,\dots,x_n,y+1\right)=h\left(x_1,\dots x_n,y,f\left(x_1,\dots,x_n,y\right)\right)$$

Способи позначення операції рекурсії:

1. $f=R_n\left(g,h\right)$
2. $f\left(x_1,\dots,x_n,y\right)=R_n\left(g\left(x_1,\dots,x_n\right),h\left(x_1,\dots,x_n,y,z\right)\right)$
3. $f\left(x_1,\dots,x_n,y\right)=\begin{cases}g\left(x_1,\dots,x_n\right)\\ h\left(x_1,\dots,x_n,y,z\right)\end{cases}$

   де $z=f\left(x_1,\dots,x_n,y-1\right)$

Приклад 1

Яку функцію отримуємо із g і h за допомогою схеми примітивної рекурсії?

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

Підказка

Використовуємо схему примітивної рекурсії без параметрів:

$$\begin{cases} f(0) = g \\ f(y+1) = h(y, f(y)) \end{cases}$$

Розв'язок. Використовуючи схему

$$\begin{cases} f(0) = g \\ f(y+1) = h(y, f(y)) \end{cases},$$

виконаємо рекурсивні кроки при $g = 0,\ h(x, y) = x$:

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

$$f(1) = h(y, f(y)) = h(0, f(0)) = h(0,0) = 0,$$

$$f(2) = h(y, f(y)) = h(1, f(1)) = h(1,0) = 1,$$

$$f(3) = h(y, f(y)) = h(2, f(2)) = h(2,1) = 2,$$

....

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

ВІДПОВІДЬ: $f(y+1) = y$

Приклад 1 (підсумки)

Яку функцію отримуємо із g і h за допомогою схеми примітивної рекурсії?

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

Відповідь. Функція $f(y+1) = y$

визначає усічене віднімання одиниці

$$f(y) = y \div 1$$

Цей вираз обчислюється так:

$$y \div 1 = \begin{cases} 0, & y = 0, \\ y - 1, & y > 0. \end{cases}$$

Чому? Тому що примітивно рекурсивні функції визначені на множині додатних цілих чисел.

Приклад 2

Довести, що $f_+(x,y) = x + y$ примітивно рекурсивна функція.

Підказка

1. Показати, що функцію $f_+(x,y) = x + y$ можна обчислити за допомогою схеми примітивної рекурсії:

$$\begin{cases} f_+(x,0) = g(x), \\ f_+(x,y+1) = h(x,y,f_+(x,y)) \end{cases}$$

2. Необхідно вказати обчислювані функції: одномісну $g(x)$ і тримісну $h(x,y,z)$, оскільки $f_+(x,y)$ – двомісна.

Розв'язок. Побудуємо схему $\begin{cases} f_+(x,0)=g(x) \\ f_+(x,y+1)=h\big(x,y,f_+(x,y)\big) \end{cases}$

$$f_+(x,0)=x+0=x=I_1^1(x)=g(x)$$

$$f_+(x,1)=x+1=f_+(x,0)+1=S\big(I_1^1(x)\big)=S\big(f_+(x,0)\big)$$

$$f_+(x,2)=x+2=(x+1)+1=S\Big(S\big(I_1^1(x)\big)\Big)=f_+(x,1)+1=S\big(f_+(x,1)\big)$$

....

$$f_+(x,y+1)=x+(y+1)=(x+y)+1=$$

$$=f_+(x,y)+1=S\big(f_+(x,y)\big)=h\big(f_+(x,y)\big)$$

Відповідь:

$$f_+(x,y)=R_3\Big(I_1^1,h\big(f_+(x,y)\big)\Big)\Rightarrow f_+(x,y)=R_3(g,h)$$

Функція $f_+(x,y)=x+y$ представлена у вигляді примітивної рекурсії

Приклад 3

Використовуючи властивість рекурсивності функції $f_+(x,y)=x+y$ обчислити рекурсивно значення $z=f_+(7,3)$.

В прикладі 2 одержали для функції $f_+(x,y)=x+y$:

$$f_+(x,y+1)=x+(y+1)=(x+y)+1=$$

$$=f_+(x,y)+1=S\left(f_+(x,y)\right)=h\left(f_+(x,y)\right)$$

Розв'язок

$$f_+(7,0)=g(x,0)=I_1^1(x)=x+0=7+0=7$$

$$f_+(7,1)=S\left(f_+(7,0)\right)=f_+(7,0)+1=8$$

$$f_+(7,2)=S\left(f_+(7,1)\right)=f_+(7,1)+1=9$$

$$f_+(7,3)=S\left(f_+(7,2)\right)=f_+(7,2)+1=10$$

Відповідь : $z=f_+(7,3)=10$

Приклад 4

Довести, що $f_{\times}(x,y)=x\cdot y$ – примітивно рекурсивна функція

Підказка

1. Показати, що функцію $f_{\times}(x,y)$ можна обчислити за допомогою схеми примітивної рекурсії
$$\begin{cases} f_{\times}(x,0)=g(x), \\ f_{\times}(x,y+1)=h\left(x,y,f_{\times}(x,y)\right) \end{cases}$$

2. Необхідно вказати обчислювані функції: одномісну $g(x)$ і тримісну $h(x,y,z)$, оскільки $f_{\times}(x,y)$ – двомісна.

Розв'язок. Побудуємо схему $\begin{cases} f_{\times}\left(x,0\right)=g\left(x\right) \\ f_{\times}\left(x,y+1\right)=h\left(x,y,f_{\times}\left(x,y\right)\right) \end{cases}$

$$f_{\times}\left(x,0\right)=x\cdot 0=0=0\left(x\right)=g\left(x\right)$$

$$f_{\times}\left(x,1\right)=x\cdot 1=f_{+}\left(x,f_{\times}\left(x,0\right)\right)=x+f_{\times}\left(x,0\right)=x$$

$$f_{\times}\left(x,2\right)=x\cdot 2=f_{+}\left(x,f_{\times}\left(x,1\right)\right)=x+f_{\times}\left(x,1\right)=x+x=2x$$

$$f_{\times}\left(x,3\right)=x\cdot 3=f_{+}\left(x,f_{\times}\left(x,2\right)\right)=x+f_{\times}\left(x,2\right)=x+2x=3x$$

$$f_{\times}\left(x,y+1\right)=x\cdot\left(y+1\right)=f_{+}\left(x,f_{\times}\left(x,y\right)\right)=x+xy=f_{\times}\left(x,y\right)+x=h\left(x,f_{\times}\left(x,y\right)\right)$$

Відповідь: $f_{\times}\left(x,y\right)=R_3\left(0\left(x\right),f_{+}\left(x,f_{\times}\left(x,y\right)\right)\right)$

Функція $f_{\times}\left(x,y\right)=x\cdot y$ представлена у вигляді примітивної рекурсії.

Приклад 5

Нехай дано функції g та h схеми примітивної рекурсії

$$g(x)=0$$

$$h(x,y,\mathrm{z})=x+z$$

Записати функції g і h, використовуючи базову систему функцій Геделя і функцію додавання $f_+(x,y)=x+y$:

Відповідь

$$f(x,y)=\begin{cases} g(x)=0(x) \\ h(x,y,\mathrm{z})=f_+\left(I_1^3(x,y,z),I_3^3(x,y,z)\right) \end{cases}$$

Піднесення до степеня $f_{\exp}\left(x, y\right) = x^y$ примітивно рекурсивне:

$f_{\exp}\left(x, 0\right) = 1;$

$f_{\exp}\left(x, y+1\right) = x^y \cdot x = f_{\times}\left(x, f_{\exp}\left(x, y\right)\right).$

У термінах оператора примітивної рекурсії одержимо:

$f_{\exp}\left(x, 0\right) = g\left(x\right) = 1$

Раніше доведено, що $f\left(x\right) = a -$ примітивно рекурсивна.

$h\left(x, y, z\right) = f_{\times}\left(x, z\right)$ — примітивно рекурсивна, що доведено в прикладі 2.

Отже, $f_{\exp}\left(x, y\right) = x^y$— примітивно рекурсивна.

Підсумок по примітивно рекурсивних функціях

Примітивно рекурсивні функції завжди можна одержати **з** **базових функцій Геделя** шляхом застосування скінченного числа операцій двох типів:

суперпозиції і рекурсії

Загальна властивість примітивно рекурсивних функцій — вони всюди визначені.

У цей клас, крім наведених вище функцій, входять багато функцій.

Частково рекурсивні функції

Далеко не всі функції, значення яких можуть бути обчислені, належать до класу примітивно рекурсивних функцій.

В операції R_n рекурсія проводиться по одній змінній.

Якщо побудувати схему **з рекурсією по двох змінних** (*подвійна рекурсія, рекурсія 2-го ступеня*), то за її застосування отримаємо функції, що не належать у загальному випадку до класу примітивно рекурсивних.

Такі функції належать до **класу частково рекурсивних функцій**. Для визначення їх обчислюваності необхідно до відомих нам операторів суперпозиції і примітивної рекурсії **додати оператор мінімізації**.

Оператор мінімізації

μ має один операнд, $f = \mu(g)$.

Значення $n-$місної функції f на заданому наборі аргументів $x_1, x_2, \ldots, x_n$ знаходять у такий спосіб.

1. Спочатку за допомогою $(n+1)$- місної функції g **формується рівняння** $g(x_1, x_2, \ldots, x_n, y) = 0$,
2. Потім **відшукується** його **розв'язок** відносно змінної y.

Якщо таких розв'язків декілька, то береться мінімальний з них (звідси і назва оператора); він і вважається значенням функції на даному наборі аргументів, $f(x_1, x_2, \ldots, x_n)$.

Іншими словами, операція мінімізації ставить у відповідність частково рекурсивній функції $g : N^{n+1} \to N$

Частково рекурсивну функцію $f : N^n \to N$

Операцію мінімізації зазвичай записують за допомогою оператора мінімізації

$$f\left(x_1, x_2, \ldots, x_n\right) = \mu\left[g\left(x_1, x_2, \ldots, x_n, y\right) = 0\right],$$

де $\mu\left[g\right] = \min\limits_{y \in N}\left\{g\left(x_1, \ldots x_n, y\right) = 0\right\}$.

У випадку, коли рівняння не має жодного розв'язку, вважається, що функція f не визначена на заданому наборі аргументів.

РОЗГЛЯНЕМО ДІЮ ОПЕРАТОРА МІНІМІЗАЦІЇ НА ПРИКЛАДАХ

Порядок застосування оператора мінімізації

1. Розглянемо обчислювану функцію

$$g\left(x_1,\ldots,x_n,y\right)$$

від $(n+1)$ змінної при $n \geq 0$ (тобто $(n+1)$– місну).

2. Зафіксуємо значення аргументів

$$\left(x_1,\ldots,x_n\right)$$

3. Визначимо мінімальне значення y при якому

$$g\left(x_1,\ldots,x_n,y\right)=0$$

Алгоритм знаходження $f(x_1,\ldots,x_n)$, якщо відомо, що $g(x_1,\ldots x_n,y)$ обчислювана:

1.Фіксуємо значення аргументів $x_1,\ldots,x_n$

2. Перевіряємо виконання рівності $g(x_1,\ldots,x_n,0)=0$. Якщо рівність вірна, то $f(x_1,\ldots,x_n)=0$, якщо ні, то переходимо до п.3.

3.Перевіряємо виконання рівності $g(x_1,\ldots,x_n,1)=0$. Якщо рівність вірна, то $f(x_1,\ldots,x_n)=1$, якщо ні, то переходимо до п.4.

4.Перевіряємо виконання рівності $g(x_1,\ldots,x_n,2)=0$. Якщо рівність вірна, то $f(x_1,\ldots,x_n)=2$, якщо ні, то переходимо до п.5.

5. І так далі….

Випадки, коли не можна знайти значення $f\left(x_1,\ldots,x_n\right)$

(I)

Взагалі **немає числа** y, для якого умова $g\left(x_1,\ldots,x_n,y\right)=0$ виконується

(II)

Існує таке y, що $g\left(x_1,\ldots,x_n,y\right)=0$, але при деякому z $\left(0\le z<y\right)$ значення **не визначене.**

Приклад 6

Знайти результат застосування операції мінімізації функції $g(x,y)=|x-2y|$

Підказка

1. Алгоритмом її знаходження є послідовне застосування оператора мінімізації $f(x)=\mu_y\left[g(x,y)=0\right]=\mu_y\left[|x-2y|=0\right]$, який визначає найменше y за умови, що $|x-2y|=0$

Розв'язок. $f(x)=\mu_y\left[|x-2y|=0\right]$

1. Фіксуємо $x=0$: $g(0,0)=|0-2\cdot 0|=0$, тобто $f(0)=0$.
2. Фіксуємо $x=1$:

   $g(1,0)=|1-2\cdot 0|=1\neq 0$

   $g(1,1)=|1-2\cdot 1|=1\neq 0$

   $g(1,2)=|1-2\cdot 2|=3\neq 0$. Функція $f(1)$ не визначена
3. Фіксуємо $x=2$:

   $g(2,0)=|2-2\cdot 0|=2\neq 0$

   $g(2,1)=|2-2\cdot 1|=0$. Функція $f(2)=1$
4. Фіксуємо $x=3$. Функція $f(3)$ не визначена
5. Фіксуємо $x=4$ Функція $f(4)=2$ …

Звідси: $f(x)=\mu y\left[|x-2y|=0\right]=\dfrac{x}{2}$

Приклад 2

За допомогою операції мінімізації обчислити

$$f(x,y)=x-y$$

Підказка

1. Підібрати обчислювану функцію $g(x,y,z)$
2. Визначити функцію $f(x,y)$ за допомогою такої операції мінімізації:

$$f(x,y)=\mu_z\left[g(x,y,z)=0\right].$$

3. Послідовно застосувати алгоритм мінімізації

Розв'язок. $f(x,y)=x-y$; $f(x,y)=\mu_z\left[g(x,y,z)=0\right]$

1. Підбираємо $g(x,y,z)$ з умови $f(x,y)=z$.

Тобто необхідно знайти таке мінімальне z, щоб $z=x-y$ або $y+z-x=0$.

$g(x,y,z)=\left|y+z-x\right|$

2. Виходячи з вимоги додатних значень примітивно рекурсивних функцій, запишемо
$$f(x,y)=\mu_z\left[x=y+z\right]=\mu_z\left[\left|y+z-x\right|=0\right]$$

3. Фіксуючи значення x та y, знаходимо значення z, при якому $f(x,y)=z$.

Приклад 3

Обчислити $f(4,1)$ якщо $f(x,y) = x - y$

Розв'язок

1. Фіксуємо $x = 4$ та $y = 1$
2. $g(4,1,0) = |1 + 0 - 4| = 3 \neq 0$
3. $g(4,1,1) = |1 + 1 - 4| = 2 \neq 0$
4. $g(4,1,2) = |1 + 2 - 4| = 1 \neq 0$
5. $g(4,1,3) = |1 + 3 - 4| = 0$

Звідси випливає $f(4,1) = 3$

Приклад 4

Обчислити $f(x,y)=\frac{x}{y}$, використовуючи операцію мінімізації.

Розв'язок

1. Побудуємо обчислювану функцію $g(x,y,z)$.

   Використовуємо принцип мінімізації: знайти таке найменше z, щоб $f(x,y)=z$. Використовуючи задану функцію $f(x,y)$, запишемо:

   $\frac{x}{y}=z$ або $x-yz=0$ $g(x,y,z)=|x-yz|$

2. Побудуємо оператор мінімізації:

   $$f(x,y)=\mu_z[x=yz]\Rightarrow f(x,y)=\mu_z[|x-yz|=0]$$

Приклад 5

Обчислити обернену функцію $f^{-1}(x)$ за допомогою операції мінімізації при $f(x)=2x$

Підказка

Обернена функція $f^{-1}(x)$ може бути отримана з функції $f(x)$ в результаті виконання операції мінімізації за формулою): Нехай $y=f(x)$

$$f^{-1}(x)=\mu_y\left[f(y)=x\right]=\mu_y\left[\left|f(y)-x\right|=0\right]$$

Розв'язок

$$f(x)=y=2x \quad f^{-1}(x)=x=2y$$

$$f^{-1}(x)=\mu_y\left[f(y)=x\right]=\mu_y\left[\left|f(y)-x\right|=0\right]=$$

$$\mu_y\left[\left|2y-x\right|=0\right]=\frac{x}{2}$$

Приклад 6

Обчислити обернену функцію $f^{-1}(x)$ за допомогою операції мінімізації при $f(x) = x^2$

Розв'язок

$$f^{-1}(x) = \mu_y\left[f(y) = x\right] = \mu_y\left[\left|f(y) - x\right| = 0\right] =$$

$$= \mu_y\left[\left|y^2 - x\right| = 0\right] = \sqrt{x}$$

Відповідь: $f^{-1}(x) = \mu_y\left[\left|y^2 - x\right| = 0\right] = \sqrt{x}$

Приклад 7

Обчислити $f^{-1}(4)$, якщо $f(x) = x^2$

Розв'язок

Розв'язок шукаємо у вигляді операції мінімізації функції $f^{-1}(x) = \mu_y\left[\left|y^2 - x\right| = 0\right]$

1. Фіксуємо $x = 4$
2. $g(4,0) = \left|0^2 - 4\right| = 4 \neq 0$
3. $g(4,1) = \left|1^2 - 4\right| = 3 \neq 0$
4. $g(4,2) = \left|2^2 - 4\right| = 0$

Відповідь: $f^{-1}(4) = 2$

Підсумок

Множина функцій, одержуваних застосуванням до базового набору скінченного числа операцій суперпозиції, примітивної рекурсії і мінімізації називається множиною **часткова рекурсивних функцій**.

Загальнорекурсивна функція

Загальнорекурсивна функція — частково рекурсивна функція, визначена для всіх значень аргументів.

Задача визначення того, чи є частково рекурсивна функція з даним описом загальнорекурсивною, <u>алгоритмічно нерозв’язна</u>.

Будь-яка **примітивно рекурсивна функція** є **частково рекурсивною**, тому що за визначенням *оператори для побудови частково рекурсивних* функцій *містять* у собі *оператори для побудови примітивно рекурсивних функцій*.

Також зрозуміло, що **примітивно рекурсивна** функція визначена всюди і тому є загальнорекурсивною функцією

(у примітивно рекурсивної функції немає причини «зависати», тому що при її побудові використовуються оператори, що визначають всюди визначені функції).

Досить складно довести існування і навести приклад загальнорекурсивної функції, що **не є** примітивно рекурсивною.

Одним з популярних прикладів є **функція Акермана**.

$$A(m,n)=\begin{cases} n+1, & m=0; \\ A(m-1,1), & m>0, n=0; \\ A\left(m-1, A\left(m,n-1\right)\right), & m>0, n>0. \end{cases}$$

```
def akker(m,n):
 if m==0:  return n+1
 elif (m>0) and (n==0): return akker(m-1,1)
 elif (m>0) and (n>0):
   return akker(m-1,akker(m,n-1))
```

```
int Akk(int m, int n)
{
     if (m=0)return (n+1);
     else if ((m>0) and (n=0))
     return Akk(m-1,1);
     return Akk(m-1,Akk(m,n-1));
}
```

Таблиця значень функції Акермана

m n	0	1	2	3	4
0	1	2	3	5	13
1	2	3	5	13	65533
2	3	4	7	29	265536-3
3	4	5	9	61	265536-3
4	5	6	11	125	$2^{2^{2^{65536}}}-3$

Примітивно рекурсивні функції в програмі

У термінах **імперативного програмування** — примітивно рекурсивні функції відповідають програмним блокам, у яких використовуються:

1) Арифметичні операції $\left(a+b, a\times b, a^{b}\right)$.

2) Умовний оператор.

3) Оператор детермінованого циклу.

Імперативне програмування — це парадигма програмування, яка описує процес обчислення у вигляді інструкцій, які змінюють стан програми.

Приклади імперативних мов: FORTRAN, BASIC, Java, PHP, C, C++, C#

Функціональне програмування — парадигма програмування, яка розглядає програму як обчислення математичних функцій та уникає станів та змінних даних. Функціональне програмування наголошує на застосуванні функцій, на відміну від імперативного програмування, яке наголошує на змінах в стані та виконанні послідовностей команд.

Частково рекурсивні функції в програмі

Програмні блоки, що містять оператор ітераційного циклу, переходять у клас частково рекурсивних функцій.

Число ітерацій в ітераційному циклі заздалегідь невідомо і може бути нескінченним.

Частково рекурсивні функції для деяких значень аргументу можуть бути не визначені через те, що оператор мінімізації аргументу не завжди коректно визначений, оскільки функція f може не дорівнювати нулю *при жодних значеннях аргументів.*
З позицій імперативного програмування, результатом частково рекурсивної функції може бути не тільки число, але і Exception або нескінченний цикл, що відповідає невизначеному значенню.