3 Funkcje częściowo rekurencyjne

Pojęcie obliczalności może być zdefiniowane na wiele równoważnych sposobów. Bieżący wykład poświęcony jest funkcjom częściowo rekurencyjnym (Kleene 1952) będącym czysto matematyczną formalizacją pojęcia obliczalności. Pokażemy równość zbiorów funkcji częściowo rekurencyjnych oraz funkcji obliczalnych przez maszyny licznikowe.

Definicja 3.1

Funkcjami prostymi nazywamy funkcje:

- zerową: $Z : \mathbb{N}^n \longrightarrow \mathbb{N}$, gdzie $Z(x_1, \ldots, x_n) = 0$
- następnika: $S : \mathbb{N} \longrightarrow \mathbb{N}$, gdzie $S(x) = x + 1$
- rzutowania: $p_i^n : \mathbb{N}^n \longrightarrow \mathbb{N}$, gdzie $p_i^n(x_1, \ldots, x_n) = x_i$

Definicja 3.2

Zbiór $\mathcal{PR}$ funkcji pierwotnie rekurencyjnych definiujemy jako najmniejszy zbiór funkcji zawierający funkcje proste (zerową, następnika i rzutowania), który jest zamknięty ze względu na operacje podstawiania oraz rekursji.

Przykład 3.1

Funkcja $f(x) = n$ $(n \in \mathbb{N})$ jest pierwotnie rekurencyjna, ponieważ jest zło-

żeniem funkcji prostych

$$f(x) = \underbrace{S(S(\ldots S(}_{n} Z(x)) \ldots)).$$

Funkcja $f(x) = x + n$ ($n \in \mathbb{N}$) jest pierwotnie rekurencyjna, ponieważ jest złożeniem funkcji prostych – rzutowania $p_1^1(x)$ oraz następnika $S(x)$:

$$f(x) = \underbrace{S(S(\ldots S(}_{n} p_1^1(x)) \ldots)).$$

Funkcja $f(x, y) = x + y$ jest pierwotnie rekurencyjna, ponieważ może być uzyskana z funkcji prostych za pomocą operacji rekursji:

$$\begin{aligned} f(x, 0) &= x, \\ f(x, y + 1) &= S\Big(f(x, y)\Big). \end{aligned}$$

Definicja 3.3

Zbiór $\mathcal{R}$ funkcji częściowo rekurencyjnych definiujemy jako najmniejszy zbiór funkcji zawierający funkcje proste (zerową, następnika i rzutowania), który jest zamknięty ze względu na operacje podstawiania, rekursji oraz minimalizacji. Funkcje częściowo rekurencyjne, które są totalne, nazywamy funkcjami (ogólnie) rekurencyjnymi.

Twierdzenie 3.1

Każda funkcja częściowo rekurencyjna jest ML-obliczalna.

Dowód

Zauważmy, że funkcje proste są w oczywisty sposób ML-obliczalne. Ponadto, zastosowanie operacji podstawienia, rekursji oraz minimalizacji nie wyprowadza poza klasę funkcji ML-obliczalnych. Zatem każda funkcja częściowo rekurencyjna jest ML-obliczalna. ■

Twierdzenie 3.2

Każda funkcja ML-obliczalna jest częściowo rekurencyjna.

Dowód (szkic)

Dla programu na maszynę licznikową skonstruujemy obliczaną przez niego funkcję częściowo rekurencyjną.

1. Zbiór konfiguracji

Zbiorem stanów (konfiguracji) maszyny licznikowej nazywamy zbiór funkcji

$$S \;=\; \Big\{ f : I\!N \cup \{L\} \to I\!N,\ \text{t.ż. } \exists_m \forall_{n>m}\, f(n) = 0 \Big\}.$$

Wartości funkcji f opisującej stan maszyny interpretujemy jako:

- $f(L)$ – zawartość licznika rozkazów
- $f(n)$ – zawartość rejestru o numerze n

2. Funkcja zmiany stanu

Funkcją zmiany stanu maszyny licznikowej nazywamy funkcję $\delta : \mathcal{I} \times S \to S$, gdzie $\mathcal{I}$ jest jej zbiorem instrukcji. Wartość funkcji f' dla stanu $\delta(i, f)$ określona jest następująco:

$$f'(L) \;=\; \begin{cases} q & \text{jeśli } i \approx I(x,y,q) \ \wedge\ R[x] = R[y] \\ f(L)+1 & \text{w przeciwnym przypadku} \end{cases}$$

$$i \approx Z(x) \implies f'(c) \;=\; \begin{cases} 0 & \text{jeśli } c = x \\ f(c) & \text{w przeciwnym przypadku} \end{cases}$$

$$i \approx S(x) \implies f'(c) \;=\; \begin{cases} f(c)+1 & \text{jeśli } c = x \\ f(c) & \text{w przeciwnym przypadku} \end{cases}$$

$$i \approx T(x,y) \implies f'(c) \;=\; \begin{cases} f(x) & \text{jeśli } c = y \\ f(c) & \text{w przeciwnym przypadku} \end{cases}$$

$$i \approx I(x,y,q) \implies \forall_{c \in I\!N}\, f'(c) = f(c)$$

3. Program

Niech $P = \{I_1, \ldots, I_k\}$ będzie programem na maszynę licznikową, zaś $\phi_P^{(m)}$ będzie m-argumentową funkcją obliczaną przez P. Element $\overline{x}$ należy do dziedziny funkcji $\phi_P^{(m)}$ wtedy i tylko wtedy, gdy istnieje ciąg stanów (konfiguracji) maszyny licznikowej f_0, f_1, …, f_t taki, że:

- f_0 jest konfiguracją początkową programu P
- $\forall_{j=0,\ldots,t-1}\ f_j(L) \leqslant k$
- $\forall_{j=0,\ldots,t-1}\ f_{j+1} = \delta(I_{f_j(L)}, f_j)$
- $f_t(L) \geqslant k+1$

Wówczas $\phi_P^{(m)}(\overline{x}) = f_t(0)$.

4. Kodowanie konfiguracji

Kodowanie K_I instrukcji ML-programu zdefiniowaliśmy w czasie jednego z poprzednich wykładów. Kodowanie K_f stanu (konfiguracji) maszyny licznikowej definiujemy jako $K_f = \tau\Big(f(L), f(0), \ldots, f(\rho(P))\Big)$.

5. Funkcja zmiany stanów określona na kodzie konfiguracji

Możemy teraz określić funkcję zmiany stanu $\delta' : K_I \times K_f \to K_f$ na parze liczb naturalnych (k_i,k_s), oznaczających odpowiednio kod instrukcji oraz kod stanu maszyny, wykorzystując kodowania bijektywne π, β oraz τ.

6. Kod konfiguracji po zadanej liczbie kroków obliczeń

Niech $\sigma : \mathbb{N}^{n+1} \to \mathbb{N}$ będzie funkcją zwracającą kod stanu maszyny licznikowej po j krokach obliczeń na $\overline{x} \in \mathbb{N}^n$. Wartość $\sigma(\overline{x}, j)$ określamy:

$$\sigma(\overline{x}, 0) = K_f(f_0)$$

$$\sigma(\overline{x}, j+1) \ = \ \begin{cases} \delta'\Big(K_I(I_j), \sigma(\overline{x}, j)\Big) & \iff \ j < t+1 \\ \sigma(\overline{x}, j) & \iff \ j \geqslant t+1 \end{cases}$$

7. Zawartość pamięci po zadanej liczbie kroków obliczeń

Określamy funkcje (częściowo rekurencyjne):

$$g_L\Big(\sigma(\overline{x}, j)\Big) \ = \ p_0\Big(\tau^{-1}(\sigma(\overline{x}, j))\Big)$$

$$g_W\Big(\sigma(\overline{x}, j)\Big) \ = \ p_1\Big(\tau^{-1}(\sigma(\overline{x}, j))\Big)$$

zwracające zawartość licznika rozkazów oraz rejestru $R[0]$ po j krokach obliczeń maszyny licznikowej.

8. Funkcja obliczana przez program

Wówczas funkcja $\phi_P^{(m)}$ określona jako:

$$\phi_P^{(m)}(\overline{x}) \;=\; g_W\Big(\sigma\Big(\overline{x}, \mu j\big(g_L(\overline{x}, j) \geqslant k+1\big)\Big)\Big)$$

jest częściowo rekrencyjna, co kończy dowód. ■

Ćwiczenie 3.1

Uzasadnij, że wszystkie wykorzystane w dowodzie twierdzenia 3.1 funkcje są częściowo rekurencyjne.

Funkcja Ackermanna (1928)

Przypomnijmy, że stosując operator podstawienia i rekursji do funkcji totalnej otrzymujemy również funkcję totalną. Operator minimalizacji nie ma tej własności. W celu uzasadnienia, że operator minimalizacji jest istotny, zaś zbiory funkcji pierwotnie rekurencyjnych i częściowo rekurencyjnych nie są równe (dokładnie $\mathcal{PR} \subsetneq \mathcal{R}$), pokażemy dwa przykłady funkcji częściowo rekurencyjnych (a więc obliczalnych), które nie są pierwotnie rekurencyjne.

Definicja 3.4

Funkcja Ackermanna $A : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$ określona jest następująco:

$$\begin{aligned} A(0, y) &= y+1 \\ A(x+1, 0) &= A(x, 1) \\ A(x+1, y+1) &= A\Big(x, A(x+1, y)\Big) \end{aligned}$$

Przykład 3.2

Przykładowe wartości funkcji Ackermanna:

$\mathbf{x} \backslash \mathbf{y}$	**0**	**1**	**2**	**3**	**4**	...	**n**
0	1	2	3	4	5	...	$n+1$
1	2	3	4	5	6	...	$n+2$
2	3	5	7	9	11	...	$2n+3$
3	5	13	29	61	125	...	$2^{n+3}-3$
4	$2^{2^2}-3$	$2^{2^{2^2}}-3$	$2^{2^{2^{2^2}}}-3$	$2^{2^{2^{2^{2^2}}}}-3$	$2^{2^{2^{2^{2^{2^2}}}}}-3$	...	$\underbrace{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}_{n+3}-3$

Uwaga

Złożoności obliczeniowej algorytmu liczącego wartość funkcji Ackermanna nie da się wyrazić za pomocą funkcji elementarnych. Trudność obliczeniowa wynika ze złożoności wywołań rekurencyjnych. W celu obliczenia jej wartości wykorzystywane są wyłącznie proste operacje arytmetyczne (dodawanie i odejmowanie) jednak ogromna liczba wywołań rekurencyjnych może doprowadzić do przepełnienia stosu (*stack overflow*).

Twierdzenie 3.3

Funkcja Ackermanna jest częściowo rekurencyjna.

Dowód

Dla każdego wywołania rekurencyjnego $A(x, y)$ zachodzi jeden z warunków:

- wartość x zmniejsza się
- wartość x nie zmienia się, natomiast wartość y zmniejsza się

Za każdym razem kiedy y osiągnie 0 wartość x maleje. Liczba wywołań rekurencyjnych jest więc ograniczona. Zauważmy jednak, że kiedy wartość x się zmniejsza, wartość y znacznie rośnie. Liczba wywołań rekurencyjnych, choć ograniczona, będzie przez to ogromna. ■

Twierdzenie 3.4

Funkcja Ackermanna nie jest pierwotnie rekurencyjna.

Idea dowodu

Funkcja Ackermanna nie może być pierwotnie rekurencyjna, ponieważ jej wartości rosną znacznie szybciej niż wartości dowolnej funkcji pierwotnie rekurencyjnej.

Własności funkcji Ackermanna

1. Dla ustalonej wartości $y \in \mathbb{N}$ funkcja $A_y(x) = A(x, y)$ jest pierwotnie rekurencyjna.
2. Dla ustalonej wartości $x \in \mathbb{N}$ funkcja $A_x(y) = A(x, y)$ jest pierwotnie rekurencyjna.

3. Dla dowolnej funkcji pierwotnie rekurencyjnej f istnieją takie stałe $x_f, y_f \in \mathbb{N}$, że
$$\forall_{x \geqslant x_f} \ \max\Big\{f(x_1,\ldots,x_n) | x_1,\ldots,x_n \leqslant x\Big\} \ < \ A_{y_f}(x),$$
czyli wartości funkcji Ackermanna rośnie znacznie szybciej niż wartości dowolnej funkcji pierwotnie rekurencyjnej (wskazanie wartości stałych x_f, y_f dla funkcji prostych oraz operatorów podstawiania i rekursji)
4. Funkcja Ackermanna jest rosnąca ze względu na pierwszą współrzędną: $A(x+1,y) > A(x,y)$ (indukcja)
5. Funkcja Ackermanna jest rosnąca ze względu na drugą współrzędną: $A(x,y+1) > A(x,y)$ (indukcja)

Ćwiczenie 3.2

Udowodnij powyższe własności funkcji Ackermanna.

Dowód twierdzenia 3.4 (przez sprzeczność)

Przypuśćmy, że funkcja Ackermanna $A(x,y)$ jest pierwotnie rekurencyjna. Zatem istnieją stałe $x_A, y_A \in \mathbb{N}$, takie że
$$\forall_{x \geqslant x_A} \ \max\Big\{A(x_1,x_2) | x_1, x_2 \leqslant x\Big\} \ < \ A_{y_A}(x) \ \Big(= \ A(x,y_A)\Big).$$
Przyjmijmy $x \geqslant \max\{x_A, y_A\}$ oraz $x_1 = x_2 = x$. Wówczas otrzymujemy nierówność $A(x,x) < A(x,y_A)$, która przeczy monotoniczności funkcji Ackermanna ze względu na drugą współrzędną. Zatem funkcja Ackermanna nie jest pierwotnie rekurencyjna. ■

Funkcja Sudana (1927)

Funkcja Sudana jest kolejnym przykładem funkcji częściowo rekurencyjnej, która nie jest pierwotnie rekurencyjna.

Definicja 3.5

Funkcją Sudana nazywamy funkcję $F_i : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$ określoną:

$$\begin{aligned} F_0(x,y) &= x+y, \\ F_n(x,0) &= x, \text{ dla } n \geqslant 0, \\ F_{n+1}(x,y+1) &= F_n\Big(F_{n+1}(x,y), F_{n+1}(x,y)+y+1\Big), \text{ dla } n \geqslant 0. \end{aligned}$$

Przykład 3.3

Przykładowe wartości funkcji Sudana $F_1(x, y)$:

x \ y	**0**	**1**	**2**	**3**	**4**	**5**
0	0	1	2	3	4	5
1	1	3	5	7	9	11
2	4	8	12	16	20	24
3	11	19	27	35	43	51
4	26	42	58	74	90	106
5	57	89	121	153	185	217
6	120	184	248	312	376	440

Przykładowe wartości funkcji Sudana $F_2(x, y)$:

x \ y	**0**	**1**	**2**	**3**	**4**	**5**
0	0	1	2	3	4	5
1	1	8	27	74	185	440
2	19	10228	$\approx 1.55 \cdot 10^{10}$	$\approx 5.74 \cdot 10^{24}$	$\approx 3.67 \cdot 10^{58}$	$\approx 5.02 \cdot 10^{135}$

Twierdzenie 3.5

Funckja Sudana jest funkcją częściowo rekurencyjną, ale nie jest pierwotnie rekurencyjna.

Ćwiczenie 3.3

Uzasadnij prawdziwość tezy twierdzenia 3.5.