2 Struktura zbioru funkcji ML-obliczalnych

W czasie poprzedniego wykładu zdefiniowany został model maszyny licznikowej oraz pojęcie ML-obliczalności. Bieżący wykład poświęcony będzie głębszemu zbadaniu zbioru funkcji obliczalnych przez maszyny licznikowe. Omówione zostaną sposoby konstrukcji funkcji ML-obliczalnych takie jak rekursja czy minimalizacja.

Zaczniemy od zdefiniowania pojęcia programu w postaci standardowej.

Definicja 2.1

Program P na maszynę licznikową nazywamy programem w postaci standardowej, jeśli dla każdej instrukcji $I(m,n,q)$ zachodzi warunek $q \leqslant k+1$, gdzie k jest liczbą instrukcji programu P.

Ćwiczenie 2.1

Udowodnij, że dla dowolnego ML-programu P istnieje program P' w postaci standardowej obliczający tę samą funkcję.

Uwaga

Od tej pory będziemy rozważać wyłącznie programy w postaci standardowej.

Oznaczenie

Dla ML-programu P przez $\rho(P)$ oznaczamy funkcję zwracającą najmniejszy adres rejestru nieużywanego przez program P.

Składanie programów

Rozpoczniemy od zdefiniowania składania programów polegającego na bezpośrednim przejściu od ostatniej instrukcji jednego programu do pierwszej instrukcji drugiego programu.

Niech $P = \{I_0, I_1, \ldots, I_{k_1}\}$ oraz $R = \{J_0, J_1, \ldots, J_{k_2}\}$ będą ML-programami w postaci standardowej. Złożeniem P i R nazywamy program

$$PR = \{I_0, \ldots, I_{k_1}, J'_0, \ldots, J'_{k_2}\},$$

gdzie $J'_i = J_i$ jeśli J_i jest instrukcją postaci $Z(k)$, $S(k)$ lub $T(m,n)$ oraz $J'_i = I(m, n, q + k_1 + 1)$ jeśli $J_i = I(m, n, q)$.

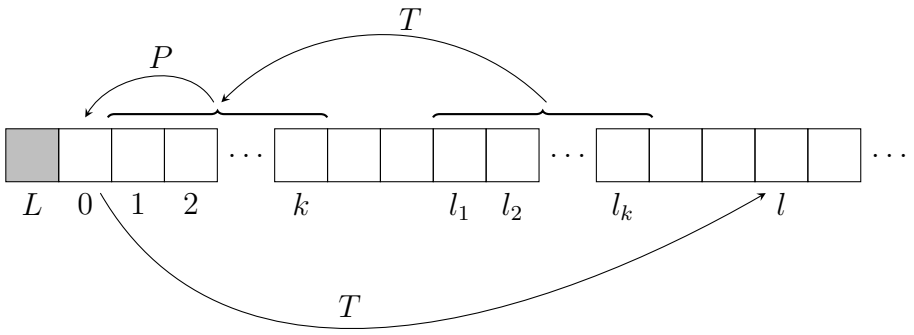
Podprogramy

Dowolny program na maszynę licznikową może być wywołany jako podprogram wewnątrz innego ML-programu. Wywołanie podprogramu wymaga odpowiedniego przygotowania jego argumentów oraz obsłużenia zwróconej przez niego wartości.

Niech $P = \{I_1, I_2, \ldots, I_n\}$ będzie ML-programem w postaci standardowej liczącym funkcję $\phi_P^{(k)}$, oraz $l_1, \ldots, l_k \notin \{1, 2, \ldots, k\}$. Zapis $P[l_1, \ldots, l_k \to l]$ oznacza wywołanie programu P na argumentach znajdujących się w rejestrach l_1, $\ldots$, l_k oraz umieszczenie zwróconej przez niego wartości w rejestrze l.

Przypomnijmy, że argumenty ML-programu powinny znajdować się w rejestrach 1, $\ldots$, k, natomiast zwracana przez niego wartośc zapisywana jest w rejestrze 0. Przed uruchomieniem programu P, należy więc skopiować jego argumenty z rejestrów l_1, $\ldots$, l_k do rejestrów 1, $\ldots$, k oraz wyzerować wszystkie pozostałe używane przez niego rejestry. Po zakończeniu działania programu P należy skopiować zwróconą przez niego wartość z rejestru 0 do rejestru l.

Przykładowy fragment kodu programu R wywołującego program P jako podprogram jest przedstawiony poniżej:



Rysunek 2.1: Schemat wywołania podprogramu

$$
\begin{array}{ll}
\vdots & \\
T(l_1, 1) & // \text{ kopiowanie argumentów} \\
T(l_2, 2) & \\
\vdots & \\
T(l_k, k) & \\
Z(0) & // \text{ przygotowanie miejsca} \\
Z(k+1) & \\
Z(k+2) & \\
\vdots & \\
Z(\rho(P)) & \\
P & // \text{ uruchomienie programu } P \\
T(0, l) & // \text{ kopiowanie zwróconego wyniku} \\
\vdots &
\end{array}
$$

Uwaga

W schemacie opisanym powyżej zakładamy, że program R nie używa rejestrów o adresach $0, \ldots, \rho(P)$ (poza wykorzystaniem rejestru 0 do zwrócenia wyniku). Możemy opuścić to założenie wymagając wykonania kopii pamięci programu R w rejestrach o adresach wyższych niż $\max(\rho(R), \rho(P))$ przed wykonaniem podprogramu i przywrócenia jej po jego wykonaniu (pomijając oczywiście rejestr zawierający wynik działania podprogramu).

Ćwiczenie 2.2

Napisz program P na maszynę licznikową liczący funkcję $f(x_1, x_2) = x_1 \cdot x_2$.

1. Użyj programu liczącego funkcję $f(x_1, x_2) = x_1 + x_2$ jako podprogramu.
2. Napisz program P bez wywoływania podprogramu.

Podstawianie

Operacja podstawiania jest uogólnieniem operacji złożenia funkcji. Na przykład, jeśli zdefiniowane są funkcje $f : \mathbb{N}^3 \to \mathbb{N}$ oraz $g : \mathbb{N}^2 \to \mathbb{N}$, to za pomocą operatora podstawiania można zdefiniowac funkcję $h : \mathbb{N}^3 \to \mathbb{N}$ taką, że $h(x, y, z) = f(x, g(x, y), z)$.

Udowodnimy teraz, że składanie funkcji obliczalnych daje w wyniku również funkcję obliczalną.

Twierdzenie 2.1

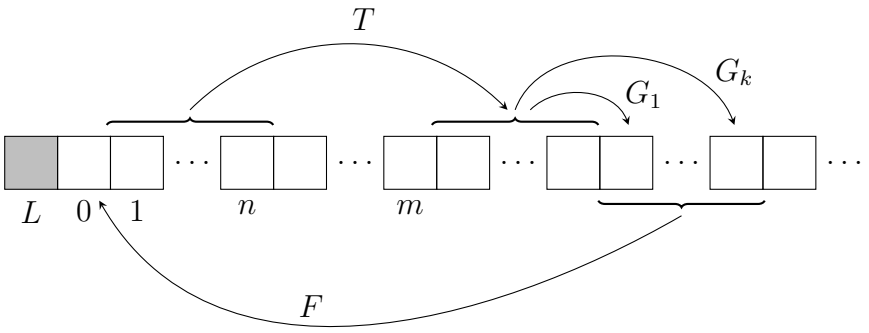
Niech $f : \mathbb{N}^k \to \mathbb{N}$ oraz $g_1, g_2, \ldots, g_k : \mathbb{N}^n \to \mathbb{N}$ będą funkcjami obliczalnymi ($f \in \mathbb{C}_k$, $g_1, \ldots, g_k \in \mathbb{C}_n$). Wówczas funkcja $h : \mathbb{N}^n \to \mathbb{N}$ określona jako $h(\overline{x}) = f\Big(g_1(\overline{x}), \ldots, g_k(\overline{x})\Big)$ jest obliczalna ($h \in \mathbb{C}_n$).

Dowód

Niech f będzie obliczana przez program F, zaś $g_1, \ldots, g_k$ odpowiednio przez programy $G_1, \ldots, G_k$. Ponadto, niech $m = \max\{n, k, \rho(F), \rho(G_1), \ldots, \rho(G_k)\}$ będzie najmniejszym numerem komórki nie używanej przez żaden z programów F, G_1, $\ldots$, G_n.

Działanie programu H obliczającego funkcję h będzie polegało na skopiowaniu wartości argumentów (rejestry $1, \ldots, n$) w miejsce nieużywane przez żaden z rozważanych programów oraz kolejnym wywołaniu G_1, $\ldots$, G_k jako podprogramów. Następnie wywołany zostanie (jako podprogram) program F na wynikach zwróconych przez programy G_1, $\ldots$, G_k, a wynik jego działania zostanie umieszczony w rejestrze 0.

Kod programu H obliczającego funkcję h zdefiniowaną przez podstawienie jest przedstawiony poniżej:



Rysunek 2.2: Schemat programu liczącego funkcję h.

$T(1, m+1)$ // kopiowanie argumentów
$T(2, m+2)$
$\vdots$
$T(n, m+n)$
$G_1[m+1, \ldots, m+n \to m+n+1]$ // przygotowanie argumentów
$G_2[m+1, \ldots, m+n \to m+n+2]$ // dla programu F
$\vdots$
$G_k[m+1, \ldots, m+n \to m+n+k]$
$F[m+n+1, \ldots, m+n+k \to 0]$ // wykonanie programu F

■

Wniosek

Operacja podstawienia pozwala utożsamiać i wprowadzać nieistotne argumenty. Przykładowo, jeśli funkcja $f(x, y)$ jest obliczalna, funkcje $f(x, x)$, $f(y, x)$, $g(x, y, z) = f(x, y)$, itp., również są obliczalne.

Przykład 2.1

Rozważmy funkcję $f(x, y) = x + y$. W oczywisty sposób $f \in \mathbb{C}_2$ oraz $f \notin \mathbb{C}_1$. Ponadto, rozważaną funkcję f możemy określić jako $f(x, y) = g(x, y, z)$ (ignorujemy trzeci argument). A zatem $f \in \mathbb{C}_3$.

Ćwiczenie 2.3

Uzasadnij, że jeśli $f \in \mathbb{C}_n$, to również $f \in \mathbb{C}_{n+1}$.

Rekursja

Rekursja jest sposobem określenia kolejnej wartości funkcji na podstawie wartości wcześniej obliczonych. Odpowiada iteracji spotykanej w językach programowania.

Twierdzenie 2.1(O operatorze rekursji)

Niech $f : \mathbb{N}^n \to \mathbb{N}$ oraz $g : \mathbb{N}^{n+2} \to \mathbb{N}$ będą funkcjami obliczalnymi ($f \in \mathbb{C}_n$, $g \in \mathbb{C}_{n+2}$). Wówczas funkcja $h : \mathbb{N}^{n+1} \to \mathbb{N}$ określona rekurencyjnie

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

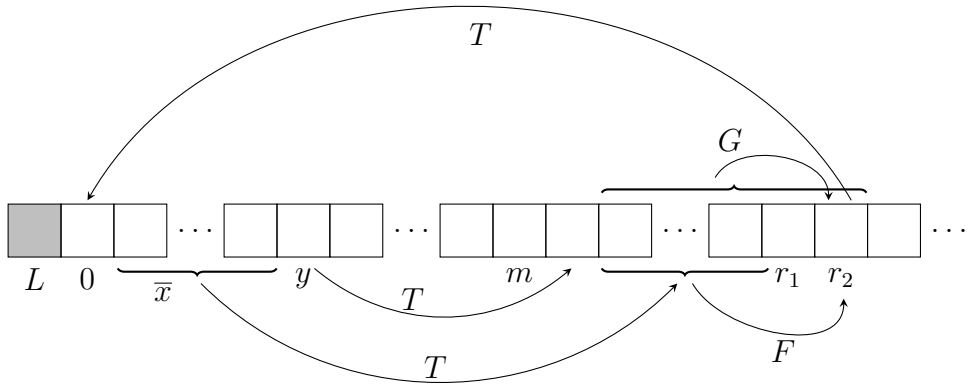
również jest obliczalna ($h \in \mathbb{C}^{n+1}$).

Dowód

Niech $f \in \mathbb{C}_n$ oraz $g \in \mathbb{C}_{n+2}$ będą obliczane odpowiednio przez programy F oraz G. Ponadto, niech $m = \max\{n+2, \rho(F), \rho(G)\}$ będzie najmniejszym adresem rejestru nie używanego przez żaden z programów F oraz G.

Działanie programu H liczącego funkcję h rozpoczyna się od skopiowania argumentów funkcji do nieużywanego obszaru pamięci i wyznaczenia wartości elementu zerowego za pomocą wywołania F jako podprogramu. Następnie, sekwencyjnie wywołujemy G jako podprogramy dla kolejnych wartości parametru y (rejestr r_1), umieszczając wyniki w rejestrze r_2. Po zakończeniu działania programu, wynik kopiowany jest do rejestru 0.

Kod program H obliczającego funkcję h zdefiniowaną za pomocą operatora rekursji jest przedstawiony poniżej:



Rysunek 2.3: Schemat programu liczącego funkcję h

	$T(n+1, m+1)$	// kopiowanie argumentu y
	$T(1, m+2)$	// kopiowanie argumentu $\overline{x}$
	$T(2, m+3)$	
	$\vdots$	
	$T(n, m+n+1)$	
	$F[m+2, \ldots, m+n+1 \to r_2]$	// wartość elementu zerowego
L:	$I(m+1, r_1, K)$	// warunek końca pętli $R[r_1] = y$
	$G[m+2, \ldots, r_1, r_2 \to r_2]$	// kolejna iteracja G
	$S(r_1)$	// zwiększenie wartości licznika (r_1)
	$I(1, 1, L)$	// GOTO L
K:	$T(r_2, 0)$	// kopiowanie wyniku

■

Minimalizacja

Niech $f : \mathbb{N}^{n+1} \to \mathbb{N}$ będzie funkcją ML-obliczalną. Minimalizacją f nazywamy funkcję $g : \mathbb{N}^n \to \mathbb{N}$, określoną jako najmniejsza wartość $y \in \mathbb{N}$ taka, że dla każdego $z < y$ funkcja $f(\overline{x}, z)$ jest określona oraz $f(\overline{x}, y) = 0$. Jeśli taka wartość $y \in \mathbb{N}$ nie istnieje, wartość funkcji g jest nieokreślona. Operator minimalizacji oznaczany jest przez $g(\overline{x}) = \mu y\Big(f(\overline{x}, y) = 0\Big)$.

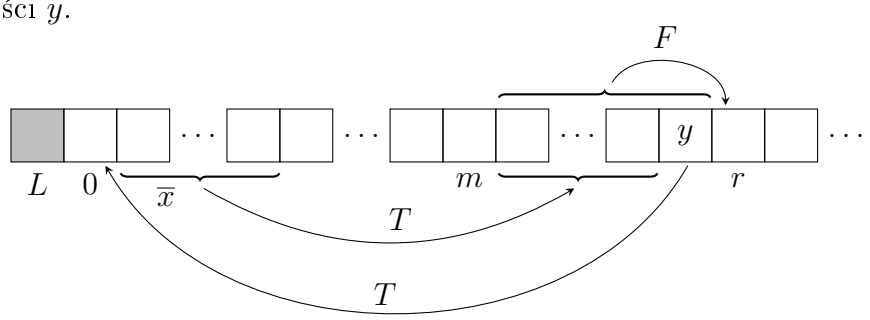
Twierdzenie 2.1(O operatorze minimalizacji)

Niech $f : \mathbb{N}^{n+1} \to \mathbb{N}$ będzie ML-obliczalna. Wówczas funkcja $g : \mathbb{N}^n \to \mathbb{N}$ określona jako $g(\overline{x}) = \mu y\Big(f(\overline{x}, y) = 0\Big)$ również jest ML-obliczalna.

Dowód

Niech funkcja $f \in \mathbb{C}_{n+1}$ będzie obliczana przez program F. Ponadto, niech $m = \max\{n+1, \rho(F)\}$ będzie najmniejszym adresem rejestru nie używanego przez program F.

Działanie programu G obliczającego funkcję g polega na obliczeniu wartości funkcji f dla kolejnych wartości parametru y. Program kończy działanie po otrzymaniu wartości 0. Zauważmy, że program G nie zatrzyma się jeśli funkcja f nigdy nie przyjmuje wartości 0 lub jeśli nie jest określona dla pewnych wartości y.



Rysunek 2.4: Schemat programu liczącego operator minimalizacji

Kod programu G obliczającego minimalizację funkcji f jest przedstawiony poniżej:

	$T(1, m+1)$	// kopiowanie argumentu $\overline{x}$
	$T(2, m+2)$	
	$\vdots$	
	$T(n, m+n)$	
L:	$F[m+1, \ldots, m+n+1 \to r]$	// wartość funkcji $f(\overline{x}, y)$
	$I(r, m+n+3, K)$	// warunek końca pętli $R[r] = 0$
	$S(m+n+1)$	// zwiększenie wartości argumentu y
	$I(1, 1, L)$	// GOTO L
K:	$T(m+n+1, 0)$	// kopiowanie wynikowej wartości y

■

Uwaga

Operator minimalizacji zastosowany do funkcji totalnej nie musi dać w wyniku funkcji totalnej.

Ćwiczenie 2.4

Uzasadnij, że dla ML-programu P funkcja $\rho(P)$ jest obliczalna.

Ćwiczenie 2.5

Niech $f : \mathbb{N}^{n+1} \longrightarrow \mathbb{N}$ będzie ML-obliczalna. Udowodnij, że następujące modyfikacje operatora minimalizacji są obliczalne:

- $g_1(\overline{x}) = \mu y\Big(f(\overline{x}, y) = k\Big)$, $k \in \mathbb{N}$
- $g_2(\overline{x}) = \mu y\Big(f(\overline{x}, y) < k\Big)$, $k \in \mathbb{N}$
- $g_3(\overline{x}) = \mu y\Big(f(\overline{x}, y) > k\Big)$, $k \in \mathbb{N}$
- $g_4(\overline{x}) = \mu y\Big(f(\overline{x}, y) \leqslant k\Big)$, $k \in \mathbb{N}$
- $g_5(\overline{x}) = \mu y\Big(f(\overline{x}, y) \geqslant k\Big)$, $k \in \mathbb{N}$