Calculabilitate şi complexităţi
Subiectul 2

Functii recursive, calculabile cu programe standard, Turing calculabile

Ce tre să știi?

Nota 6:

- definitiile celor 3 tipuri de functii

- relatiile intre ele (enunturi)

Fiecare demonstratie, la alegere: 2p

Definiții

Funcții recursive

Sunt de forma $f : N^k \to N$.

- Funcțiile elementare
  - Funcțiile constante: $c_p(x1,\ x2,\ ...,\ xk) = p$
  - Proiecții: $\pi_r(x1,\ x2,\ ...,\ xk) = xr$
  - Succesor: $succ(x) = x + 1$
- Operații
  - Compunerea funcțională
    - $h : N^m \to N,\ gi : N^k \to N\ \forall\ 1 \le i \le m$
    - Putem forma funcția f prin compunerea funcțională a lui h cu g1, g2, ..., gm: $f(x1,\ x2,\ ...,\ xk) = h(g1(x1,\ x2,\ ...,\ xk),\ ...,\ gm(x1,\ x2,\ ...,\ xk))$.
  - Recurență primitivă

- $f : N^{k+1} \to N$ este definită prin recurență primitivă de funcțiile $g : N^k \to N$ și $h : N^{k+2} \to N$ dacă:
  - $f(x1,\ x2,\ ...,\ xk,\ 0)\ =\ g(x1,\ x2,\ ...,\ xk)$
  - $f(x1,\ x2,\ ...,\ xk,\ t\ +\ 1)\ = h(x1,\ x2,\ ...,\ xk,\ t,\ f(x1,\ x2,\ ...,\ xk,\ t))$

- Minimizarea mărginită
  - Funcția $f : N^k \to N$ se obține prin minimizare mărginită de funcția $g : N^{k+1} \to N$ dacă $f(x1,\ x2,\ ...,\ xk)\ = min_{y<=z}[g(x1,\ x2,\ ...,\ xk,\ y)\ =\ 0]$

Funcții Turing calculabile

O funcție $f : N^k \to N$ se numește Turing calculabilă dacă există o mașină Turing ce are ca input $(x1,\ x2,\ ...,\ xk)\ \in\ N^k$ și ca output $f(x1,\ x2,\ ...,\ xk)$.

Limbajul Standard

Programele Standard (scrise în limbajul Standard) calculează funcții de forma $f : N^k \to N$.

- Variabile
  - De intrare: $X1,\ X2,\ X3,\ ...$
  - Interne/ de lucru/ auxiliare: Z1, Z2, Z3, ...
  - De ieșire: Y.
- Instrucțiuni
  - Sunt de două tipuri:
    - Etichetate: A: instrucțiune neetichetată, etichetele sunt de forma A1, B1, C1, A2, B2, C2, ...
    - Neetichetate:
      - $V \leftarrow v$
      - $V \leftarrow V\ +\ 1$
      - $V \leftarrow V\ -\ 1$
      - $IF\ \ V\ \neq 0\ GOTO\ L$, L etichetă

Un program standard, se termină fie prin salt la eticheta **E**, fie prin salt la o etichetă care nu există (care nu are asociată nicio instrucțiune), fie la finalul ultimei instrucțiuni.

Funcția $f: N^k \to N$ pentru care există un program standard ce are ca input x1, x2, ..., xk și ca output f(x1, x2, ..., xk) se numește calculabilă cu **programe standard**.

Relații între aceste tipuri de funcții

- Orice funcție calculabilă cu programe Standard este Turing calculabilă
- Orice funcție recursivă este calculabilă cu programe standard
- Orice funcție Turing calculabilă este recursivă

Demonstrații

Orice funcție calculabilă cu programe Standard este Turing calculabilă

Fie $f: N^k \to N$ calculabilă cu programe standard.

Construim mașina M care are ca input x1, x2, ..., xk.

Deci, avem deja pe bandă valorile corespunzătoare variabilelor X1, X2, ..., Xk. Noi trebuie să fixăm pe bandă pozitiile corespunzătoare variabilei Y și variabilelor Z1, Z2, ... Zm.

Mai întâi, deplasăm tot conținutul benzii la dreapta cu două poziții. Prima poziție va fi un ō - (echivalent cu un 1) = Y + 1 și a doua un 0 - separator.

La dreapta input-ului, vom fixa câte un ō și câte un 0 pentru fiecare variabilă auxiliară.

O stare <Ij> a mașinii va fi o codificare a instrucțiunii Ij din programul Standard care calculează f.

Avem următoarele cazuri:

- $Ij = V \leftarrow V$ - Trece în codificarea lui Ij+1 dacă există instrucțiunea Ij+1. Altfel, trece în stare finală.
- $Ij = V \leftarrow V + 1$ - Identifică poziția asociată lui V pe banda mașinii Turing și crește cu o unitate numărul de ō la această poziție. Trece în codificarea lui Ij+1 dacă există instrucțiunea Ij+1. Altfel, trece în stare finală.
- $Ij = V \leftarrow V - 1$ - Identifică poziția asociată lui V pe banda mașinii Turing și scade cu o unitate numărul de ō la această poziție. Trece în codificarea lui Ij+1 dacă există instrucțiunea Ij+1. Altfel, trece în stare finală.
- $Ij = IF\ V \neq 0\ GOTO\ L$ - Identifică poziția asociată lui V pe banda mașinii Turing.
    - Dacă valoarea lui V este 0 (e un singur ō pe bandă la poziția asociată), atunci trece în codificarea lui Ij+1 dacă există instrucțiunea Ij+1, iar altfel, trece în stare finală.
    - Dacă valoarea lui V este diferită de 0, atunci mașina trece în starea <Ir> unde $r = min_p\{Ip \mid Ip\ are\ eticheta\ L\}$. Dacă nu există un astfel de r sau L = E, atunci mașina trece în stare finală.

Orice Turing calculabilă este recursivă

Preliminarii

- Funcția pereche: $<,>: N^2 \rightarrow N,\ <x, y> = 2^x(2y + 1) - 1$
    - Funcțiile $l, r : N \rightarrow N$:
        - $l(z) = x\ astfel\ \hat{i}nc\hat{a}t\ \exists\ y\ cu\ <x, y> = z$ (left)
        - $r(z) = y\ astfel\ \hat{i}nc\hat{a}t\ \exists\ x\ cu\ <x, y> = z$ (right)
- Al n-lea număr prim: $pn : N \rightarrow N$
- Gödelizare: $[a1, a2, ..., ak] = p1^{a1}\, p2^{a2} ... pk^{\,ak}$
    - $e_i(z) = ai$
    - $Lt(z) = k$

Toate funcțiile definite aici sunt primitiv recursive.

Demonstrație

Fie $f : N^k \to N$ Turing calculabilă.

Deci, există M = (Q, V, U, δ, q_0, B, F) mașină care calculează f

Renumerotăm stările: $Q = \{q0,\ q1,\ ...,\ qr\}\ \to\ Q = \{0,\ 1,\ ...,\ r\}$

Renumerotăm alfabetul:
$U = \{0,\ 1,\ B,\ ...,\}\ cu\ |U|\ =\ m\ \to U = \{s0,\ s1,\ ...,\ sm\}\ -\!\!>\ U = \{0,\ 1,\ ...,\ m\}$

Definim o configurație a mașinii M. O configurație este formată din:

- Starea q
- Poziția capului de citire-scriere - p
- Conținutul benzii - o secvența finită (până la B) si1, si2, ... sik.

Deci, vom folosi funcțiile definite mai sus și vom numi o configurație:

<#(q), <p, [s1, s2, ..., sk]> > = z.

Definim niște funcții:

- **h1(z)** = numărul atașat stării configurației imediat următoare configurației cu numărul atașat z, dacă z este o configurație validă și î altfel.
- **h2(z)** = poziția capului de citire al configurației imediat următoare configurației cu numărul atașat z, dacă z este o configurație validă și î altfel.
- **h3(z)** = numărul atașat conținutului benzii în configurația imediat următoare configurației cu numărul atașat z, dacă z este o configurație validă și î altfel.

Deci, h1 este corespondentul lui q din configurația următoare, h2 al lui p și h3 al lui si1, si2, ... sik.

Acum vom defini $C_M : N^{k+!} \to N$, unde $C_M(x,\ n) =$ numărul atașat configurației mașinii M, pe intrarea x, la pasul n (și x = (x_1, x_2, ..., x_k)).

Dacă $C_M(x,\ n) = C_M(x,\ n\ +\ 1)$, atunci $C_M(x,\ n) = C_M(x,\ n\ +\ t)\ \forall\ t \geq 0$.

Acum, vom scrie C_M în funcție de h1, h2 și h3.

$C_M(x,\ 0)\ =\ <0,\ <0,\ [p1,\ p2,\ ....,\ p(x1+1),\ p(x1+2),\ ....,\ p(x1+x2+2),\ ......]>>$ = starea 0, poziția capului la începutul benzii, iar conținutul benzii este chiar input-ul.

$C_M(x,\ n\ +\ 1)\ =\ <\ h1(C_M(x,\ n)),\ <h2(C_M(x,\ n)),\ h3(C_M(x,\ n))>\ \ >$

Dacă h1, h2 și h3 sunt recursive, atunci și C_M este recursivă.

Fie $a\ \in\ \{0,\ ....r\},\ b\in\ \{0,\ ....,\ m\}$

Acum, definim:

- g1(a, b) = numărul stării în care trece M din starea cu numărul a citind simbolul cu numărul b
- g2(a, b) = direcția în care se deplasează capul de citire-scriere atunci când M se află în starea cu numărul a și citește simbolul cu numărul b: 0 pentru stânga, 2 pentru dreapta.
- g3(a, b) = numărul atașat simbolului scris de M pe bandă atunci când se află în starea cu numărul a și citește simbolul cu numărul b.

Deoarece g1, g2 și g3 au domeniu finit ele sunt recursive.

Tot ce avem de făcut este să scriem h1, h2 și h3 în funcție de g1, g2 și g3:

- $h1(z)\ =\ g1(l(z),\ e_{\ l(r(z))}(r(r(z)))$
- $h2(z)\ = l(r(z))\ +\ g2(\ l(z),\ e_{\ l(r(z))}(r(r(z)))$ -1
- $h3(z)\ =\ \frac{r(r(z))}{p_{l(r(z))}^{e_{l(r(z))}(r(r(z))}}\, p_{l(r(z))}^{g3(l(z),\, e_{\ l(r(z))}(r(r(z)))}$

Decii, h1, h2, h3 sunt recursive => **C_M este recursivă**, q.e.d.