

SUBIECTUL 1

1. Mașini Turing ca dispozitive de acceptare și de calcul

- DEFINITIONI

Def. Se numește mașină Turing un 7-uplu de forma:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F), \text{ unde:}$$

Q = multimea stărilor, $Q \neq \emptyset$, Q finită

Σ = alfabetul de intrare, $\Sigma \neq \emptyset$, Σ finită

Γ = alfabet interum, $\Gamma \neq \emptyset$, Γ finită (alf. bazei)

F = multimea de stări finale ~~finală~~

δ = funcția de tranziție, $\delta: (Q \setminus F) \times \Gamma \rightarrow Q \times \Gamma \times \{\leftarrow, \circ, \rightarrow\}$

$\square \in \Gamma$, \square = blană

$$\delta(q, a) = (q', b, \Delta), \Delta \in \{\leftarrow, \circ, \rightarrow\}$$

q_0 = stare initială, $q_0 \in Q$

$Q \cap F = \emptyset$

$$\Sigma \subseteq \Gamma \setminus \{\square\}$$

Moduri de reprezentare:

1. $\Sigma \quad \boxed{a_1} \boxed{a_2} \dots \boxed{a_m} \quad$ - hard $w = a_1 a_2 \dots a_m \in \Sigma^*$


Soft: δ

2. formal graf

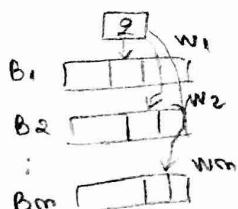
q_0 - stare initială

$q \rightarrow (q')$ - stare finală, $q \in F$

$$\delta(q, a) = (q', b, \Delta) \quad \begin{matrix} q \\ \xrightarrow{(a,b,\Delta)} \\ q' \end{matrix}$$

Def. TM cu mai multe baze (m -TM): $\rightarrow m$ -TM

$$\delta: (Q \setminus F)^m \times \Gamma^m \rightarrow Q \times \Gamma^m \times \{\leftarrow, \circ, \rightarrow\}^m, M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$$



$$I \in \Sigma^* \quad \boxed{w_1} \# \boxed{w_2} \# \dots \boxed{w_m}$$

$$\delta(q_i, \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix}) = (q_j, \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}, \begin{pmatrix} \Delta_1 \\ \Delta_2 \\ \vdots \\ \Delta_m \end{pmatrix})$$

Def. (NTM) = Mașină Turing mediterimimistă:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$$

$$\delta: (Q \setminus F) \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{\leftarrow, \circ, \rightarrow\}}$$

$$f(g, a) = \{(p_1, b_1, s_1), \dots, (p_l, b_l, s_l)\}$$

$$b = \max_w \text{card}(\{f(g, a) \mid g \in \Sigma^*, a \in \Gamma\})$$

a₁, ..., a_m - banda de intrare

r_m - banda de lucru pe care se simulează M

w - banda pe care se scriu adresele nodurilor din arborele de calcul al lui M

$$L(M) = \{w \mid w \in \Sigma^*, \exists g \in \Sigma^*, \exists u, v \quad (g \circ w \vdash_M^* ug v)\}$$

• TM ca acceptor (masina se poate opri sau nu să accepte cerința w)

$$L(M) = \{w \mid w \in \Sigma^*, \exists g \in F, \exists u \in \Gamma^*, g \circ w \vdash_M^* g \circ v\}$$

• TM ca traductor (calculator al valoarelor unei funcții)

$$f_M: \Sigma^* \rightarrow \Gamma^*, g \circ w \vdash_M^* g \circ f(w)$$

ex: L = {a^mb^mc^m | m ≥ 1}

succ: N → N, succ(m) = m + 1.

Dacă masina se oprește pe intrarea (x₁, ..., x_k), atunci acela este rezultatul și f_M. Dacă nu, f_M nu e definită în (x₁, ..., x_k)

ENUNȚURI:

1. Orice NTM cu 3 benzi poate fi simulațiat de o mașină Turing cu 3 benzi. ($L_{NTM} \subseteq L_{3-TM}$).

2. Orice m-TM poate fi simulațiat de o TM cu 3 benzi. În plus, dacă M este deterministă și M' este neterministă. ($L_{m-TM} \subseteq L_{NTM}$).

3. Orice NTM cu 2 benzi poate fi simulațiat de o mașină Turing deterministă cu 2 benzi. ($L_{NTM} \subseteq L_{2D-TM}$).

4. Orice mașină Turing deterministă cu 2 benzi poate fi simulațiată de o mașină Turing cu 2 benzi. ($L_{2D-TM} \subseteq L_{TM}$).

• $L_{NTM} \subseteq L_{3-TM}$

• $L_{NTM} \subseteq L_{2D-TM}$

• $L_{m-TM} \subseteq L_{TM}$

• $L_{2D-TM} \subseteq L_{TM}$.

Demonstratii:

1. $L_{NTM} \subseteq L_{3-TM}$

Consider M e NTM. Notez cu w multimea configurațiilor posibile ale lui M: $w = f(g, a, s, b, d) \mid g = \text{stare initială}, s = \text{stare}, a, b = \text{simb} \neq b, d = \text{direcția}$

(g, a, s, b, d) - dim starea g, citind a, trec din starea s, scriu b și merg în direcția d.

w - multime finită și poate fi ordonată lexicografic.

Ce va face M':

- Pe B₁: configurația initială
- Pe B₂: copiază conținutul lui B₁.
- Pe B₂: scrie primul element din w

• Verifică simbolul curent de pe B_2

• Dacă dacă se află în starea g^1 (dacă nu, efectuează pasul de return)

• Verifică dacă simbolul citit de pe B_1 este a .

• Serie b peste a

5. Schimbă starea în g^1

6. Deplasează capul de pe B_1 în direcția d

7. Deplasează capul de citire de pe B_2 la dreapta.

• Se va relua de la pasul 1 dacă există un simbol pe B_2 . Dacă nu, se efectuează pasul R (de întoarcere)

R: Copiază w de pe B_3 pe B_2

• Serie pe B_2 succesorul elem. aflat pe bandă.

• Intră în g_0 .

2. $L_{m-TM} \subseteq L_{TM}$

Fie M mașină Turing cu m benzi.

M' mașină Turing cu $\leq m$ benzi.

Se construiește mașina M' cu $\leq m$ benzi "foarte lată". Fiecare celula a acestei benzi poate fi reprezentată ca un vector, așezat pe verticală. Acesta are $2+m$ componente, număr de benzi ale lui M . Elementul de index par din vector reprezintă simbolul de pe banda aferentă, iar elementul de pe poz. impară sunt o sau și reprezintă poziția capului de citire (marcat cu 1).

Există un set de simboluri ale lui M' .

• În fiecare mișcare a lui M , M' face următoarele lucru:

1. (a) Poziționăm capul de citire pe prima celulă.

(b) Memorăm starea și apoi simbolurile de pe pistele de ordin par, care sunt deasupra lui 1 pe pistele de ordin impar (simboluri curente ale lui M).

2. Aplicând funcția de tranziție, M' va avea multe mișcări și nu va modifica banda astfel încât devine identică cu a lui M .

Variatii:

• $n\Delta-TM$: mașină Turing deterministă cu m benzi: are cel mult $\leq m$ intrări pentru fiecare combinație simbol-stare

• $nB-NTM$: mașină Turing ne deterministă cu m benzi: pot fi mai multe intrări pentru cel puțin $\leq m$ combinație simbol-stare.

• Bandă infinită se consideră membru unic la ambele capete. Putem avea și nMT cu banda nășig. la un capăt, care e de fapt formal și al puterii de calcul, echivalentă cu $\leq nMT$ cu 2 capături infinite.

• MT fără opțiunea de a sta pe loc:

$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{ \leftarrow, \rightarrow \}$.

SUBIECTUL 2

Functie recursive, functie calculabile cu programe standard, functie Turing - calculabile

Definitii:

Def. Functie unitiale de baza:

- functie succesor $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}$, $\text{succ}(x) = x + 1$
- fct. identice nulla: $m: \mathbb{N} \rightarrow \mathbb{N}$, $m(x) = 0$
- fct. proiectie $\text{pr}_i(m): \mathbb{N}^m \rightarrow \mathbb{N}$, $\text{pr}_i(m)(\overbrace{x_1, \dots, x_m}) = x_i$

Operatii unitiale:

(i) compunere functionala

$f: \mathbb{N}^m \rightarrow \mathbb{N}$ se obtine prin compunere functionala din $g_1, g_2, \dots, g_k: \mathbb{N}^m \rightarrow \mathbb{N}$ si $f(\vec{x}) = f(g_1(\vec{x}), \dots, g_k(\vec{x}))$

(ii) recurrenta primitiva

$f: \mathbb{N}^{m+1} \rightarrow \mathbb{N}$ se obtine prin recurrenta primitiva din $g: \mathbb{N}^m \rightarrow \mathbb{N}$ si $f(\vec{x}, 0) = g(\vec{x})$

$$\left\{ \begin{array}{l} f(\vec{x}, t+1) = h(\vec{x}, t, f(\vec{x}, t)) \end{array} \right.$$

(iii) minimizare membru unitate

$f: \mathbb{N}^m \rightarrow \mathbb{N}$ se obtine prin minimizare membru unitate din $\mathbb{N}^{m+1} \rightarrow \mathbb{N}$ $\Leftrightarrow f(\vec{x}) = \begin{cases} \min \{ t \mid t \geq 0, g(\vec{x}, t) = 0 \} & \text{daca } M \neq \emptyset \\ \gamma \text{ (medefinita)} & \text{daca } M = \emptyset \end{cases}$

Def. O functie se numeste functie primitiva recursiva \Leftrightarrow este o functie unituala sau se poate obtine din functie unitiale prin aplicarea de un nr. finit de ori a operatiilor (i) si (ii).

Def. O functie se numeste functie partial recursiva \Leftrightarrow este o functie unituala sau se poate obtine din functie unitiale prin aplicarea de un nr. finit de ori a operatiilor (i), (ii) si (iii).

Def. O functie se numeste functie recursiva \Leftrightarrow este partial sau totala.

PROGRAME STANDARD (LIMBAJUL S)

Def. - variabile de intrare: x_1, x_2, \dots

- variabile interne: $z_1, z_2 \dots$ sunt unitale

- variabile de ieșire: y

- etichete: A, B, C, D, E, A₂, B₂, ...

- instrucțiuni

(i) incrementarea: $v \leftarrow v + 1$, $y = \text{variabila}$

(ii) decrementarea: $v \leftarrow v - 1$

(iii) salt conditional: if $v \neq 0$ GOTO L, L - eticheta

(iv) dummy / efect nul $v \leftarrow v$

Deci, program standard: lista de instrucțiuni etichetate sau nu

- limbajul unui program standard: lista de instrucțiuni

- stare a programului: P la un moment dat e o lista de ecuatii de forma $v = m$, $v = \text{variabila}$, $m \in \mathbb{N}$. Aceasta lista contine cel putin o ecuatie pt fiecare variabila care apare in program

și cel mult o ecuație pt. oarecare variabilă.

- configuratie a lui P este o pereche (i, τ) , unde $i =$ nr. de op. al instrucțiunii care urmează să fie executată și $\tau =$ starea curentă.

$$(i, \tau) \vdash (j, \tau') \Leftarrow \Delta$$

Fie programul standard:

$$P: \begin{cases} \text{var. intrare: } x_1, \dots, x_m \\ \text{var. interne: } z_1, \dots, z_m \\ \text{var. ieșire: } y \\ \text{instrucțiuni: } I_1, I_2, \dots, I_k \end{cases}$$

(i) dacă $I_1 = v \leftarrow v+1$ atunci

$$j = i+1$$

și se obține din τ îndepărând ecuația $v = g$ cu $v = g+1$

(ii) dacă $I_1 = v \leftarrow v - 1$ atunci

$$j = i+1$$

și se obține din τ ce valoarea lui v decrementează

(iii) dacă $v = v$ atunci

$$j = i+1, \tau = \tau'$$

(iv) dacă $I_1 = \text{IF } v \neq 0 \text{ GO TO } L$ atunci

dacă $v = 0$ atunci $j = i+1$ și $\tau = \tau'$

dacă $v \neq 0$ și \exists în P o instrucțiune etichetată cu L ,

atunci $\tau = \tau'$ și $j =$ nr. de ordine al I instrucție etichetată cu L .

dacă $v \neq 0$ și nu \exists în P o instrucțiune etichetată cu L ,

atunci $j = K+1$ // se oprește

Convenție: Eticheta E se folosește pentru exit. ($I_{K+1} = \text{sf. programul}$)

Def. O funcție este Turing-calculabilă dacă mașina Turing asociată se oprește după un nr. finit de cehuri cu automatul într-o stare finală.

Def. O funcție este s-calculabilă dacă \exists un program care calculiază acea funcție.

ENUNȚURI:

- Dacă f este o funcție s-calculabilă $\Rightarrow f$ este funcție Turing-calculabilă
- Dacă f este o funcție Turing-calculabilă $\Rightarrow f$ este funcție recursivă
- Dacă f este o funcție recursivă $\Rightarrow f$ este funcție s-calculabilă

DEMONSTRATIE (f Turing-calculabilă $\Rightarrow f$ recursivă)

Fie $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ o MT deterministă.

$$\overline{x_1} \mid 0 \quad \overline{x_2} \mid 0 \quad \overline{x_3} \mid 0 \quad \cdots \mid 0 \quad \overline{x_K} \mid 0 \quad \{$$

$$f: N^m \rightarrow N, f(x_1, \dots, x_m)$$

$$f(x_1, \dots, x_m) \mid 000000B$$

Configurație: $\alpha\beta$

- numerotăm celele benzii: 0, 1, 2, ...

numerotările stării din α : $g_0=0, g_1=1, g_2=2, \dots, g_i=i$

numerotările simbolurilor din Γ : $0=0, 1=1, \dots$

Numerul atașat unei configurații:

$$g \rightarrow x$$

$|x|=j$ - poziția pe bandă a capului este j

Continutul benzii: $\alpha_p = s_1 s_2 \dots s_m \rightarrow$ nr. atașat este $p^{\# s_1} p^{\# s_2} \dots p^{\# s_m}$, unde $\# s_k$ este nr. simb. s_k .

Configurația $\rightarrow \langle x, \langle y, z \rangle \rangle$ $x =$ nr. stăru

$y =$ poz capului de citire / scriere
 $z =$ nr. atașat conținutului benzii

$$\langle a, b \rangle = 2^a(2b+1)-1$$

$C_M(x, m) =$ nr. atașat configurației de la pasul m al mazinii Turing M pe intrarea x .

Ne preparam să demonstrăm că $C_M(x, m)$ este funcție recursivă.

$$C_M(x, 0) = \langle 0, \langle 0, p_1 \cdot p_2 \cdot p_{x+1} \cdot p_{x+3} \cdots p_{x_1+x_2+2} \cdots p_{x_1+x_2+\dots+x_K} \rangle \rangle$$

$h_1(z) = \begin{cases} \text{nr. atașat stării } z \text{ care drece } M \text{ din config. curentă cu nr. } z, \\ \text{dacă } z \text{ este nr. unei config. valide} \\ \uparrow, \text{ altfel} \end{cases}$

$h_2(z) = \begin{cases} \text{poziția capului de citire / scriere din care depinde nr. din config. curentă cu nr. } z, \text{ dacă } z \text{ este nr. unei config. valide} \\ \uparrow, \text{ altfel} \end{cases}$

$h_3(z) = \begin{cases} \text{nr. atașat conținutului benzii din care drece } M \text{ din config. cu nr. } z, \text{ dacă } z \text{ este nr. unei config. valide} \\ \uparrow, \text{ altfel} \end{cases}$

$$C_M(x, m+1) = \langle h_1(C_M(x, m)), \langle h_2(C_M(x, m)), h_3(C_M(x, m)) \rangle \rangle$$

h_1, h_2, h_3 recursive?

$g_1(a, b) =$ nr. stării în care drece M din starea cu nr. a citind simbolul cu nr. b

$g_2(a, b) = \begin{cases} 0, M \text{ se deplasează la stânga din starea cu nr. } a \text{ citind simbolul } b \\ 1, M \text{ se deplasează la dreapta din starea cu nr. } a, citind simbolul } b \end{cases}$

$g_3(a, b) =$ nr. simbolului scris de M din starea cu nr. a , citind simbol. cu nr. b

g_1, g_2, g_3 sunt fct. de suport finit $\Rightarrow g_1, g_2, g_3$ fct. recursive.

$h_1(z) = g_1(\ell(z), (r(r(z)))_{\ell(r(z))})$ - f. recursivă

$h_2(z) = (\ell(z) + g_2(\ell(z), (r(r(z)))_{\ell(r(z))})) \stackrel{?}{=} f. \text{ recursivă}$

$h_3(z) = (r(r(z)) | P_{\ell(r(z))} \cdot P_{\ell(r(z))}^{g_3(\ell(z), r(r(z)))})$ - f. recursivă

$C_M(x, m) - M$ se oprește după m pagini

$C_M(x, m), \forall m > m_0$

$\mu(x) = \min_m [C_M(x, m) = C_M(x, m+1)] =$ nr. de pagini după care M se oprește pe intrarea x

$f(x) = L_2(\mu(x) + C_M(x, \mu(x))) - \Delta = f(x_1, \dots, x_n) \Rightarrow f$ funcție recursivă

SUBIECTUL 3

Functie (s-calcularibila) universală, program universal

PROGRAME STANDAR, LIMBAZUL S, FUNCTIE S-CALCULARIBILA... - SB 2

- Codificare / Decodificare

- $\langle \cdot, \cdot \rangle : N^2 \rightarrow N$ (bijectie de la N^2 la N)

$$\langle x, y \rangle = 2^{x_1} (2y + 1) - 1.$$

Demonstrare: $\langle \cdot, \cdot \rangle$ bijectivă

- Injectivitate

$$2^{x_1} (2y_1 + 1) - 1 = 2^{x_2} (2y_2 + 1) - 1$$

$$\Leftrightarrow 2^{x_1} (2y_1 + 1) = 2^{x_2} (2y_2 + 1)$$

$$\Rightarrow 2^{x_1} \cdot 2y_1 + 2^{x_1} = 2^{x_2} \cdot 2y_2 + 2^{x_2} \quad (\text{împărțim la } 2 \text{ pînă nu se mai poate})$$

$$\Rightarrow \begin{cases} x_1 = x_2 \\ y_1 = y_2 \end{cases}$$

- Surjectivitate

$$2^x (2y + 1) - 1 = z \Rightarrow \langle l(z), r(z) \rangle$$

$$2^x (2y + 1) = z + 1$$

$$x = \exp(z+1) / \text{împărțit la } 2 \text{ pînă nu se mai poate}$$

$$l, r : N \rightarrow N$$

$$l(z) = \max_{t \leq z} [2^t | z+1] = \max_{t \leq z} [7(2^{t+1} | z+1)] = \min_{t \leq z} [z \cdot \frac{1}{d} (\langle t, y \rangle = z)]$$

$$l(z) = \frac{\max_{t \leq z} [t | z+1 \text{ și } t \text{ împărțit}]}{2} = \frac{\frac{z+1}{\text{exp}(z)}}{2} = \min_{t \leq z} [z \cdot \frac{1}{d} (\langle t, y \rangle = z)]$$

$\langle \cdot, \cdot \rangle$ bijectivă și primitivă recursivă

$$\forall z \quad (l(z), r(z)) \leq z)$$

l, r primitivă recursivă

$$\langle l(z), r(z) \rangle = z.$$

- $[\dots] : \bigcup_{m \geq 0} N^m \rightarrow N$ - asociază unei liste un număr natural

$$[\emptyset] = 0$$

$[x_1, x_2, \dots, x_m] = p_1^{x_1} p_2^{x_2} \cdots p_m^{x_m}$, $p_i = \text{al } i\text{-lea nr. prim}$
(numerele lui Gödel)

$[\dots]$ primitivă recursivă pt o listă de anumită lungime (pt că nu sunt foarte mari).

Este surjectivă, pot găsi $[x_1 \dots x_m]$ prin alg. de descompunere în factori primi.

$$\text{ex: } [2, 3] = 2^2 \cdot 3^3 - 27^4 = [2, 3]_0 = [2, 3, 0, 0, 0, 0] \quad (\text{NU E INJECTIVĂ})$$

• $(\cdot)_i : \mathbb{N} \rightarrow \mathbb{N}$

$$([x_1, x_2, \dots, x_m])_i = x_i$$

$$p_1^{x_1} \cdot p_2^{x_2} \cdot \dots \cdot p_m^{x_m} = [(z)_1, (z)_2, \dots, (z)_m]$$

$$(z)_i = \max_{z \leq z} [p_i^{-1}|z] = \min [7 (p_i^{-1}|z)]$$

$$L : \mathbb{N} \rightarrow \mathbb{N}, L(z) = \begin{cases} \min [z_i \neq 0 \wedge \forall j (z_j \cdot \forall j \leq i)] & \text{dor } i \leq z \\ 1, \text{ dacă } z_i = 0 \rightarrow 0, \text{ dacă } z = 0 \end{cases}$$

[...], $(\cdot)_i$, L sunt primitive recursive

Codificarea programelor standard

- codificarea variabilelor:

V		y	x_1	z_1	x_2	z_2	x_3	z_3	\dots
#(V)		0	1	2	3	4	5	6	

- codificarea etichetelor

L		λ	A_1	A_2	A_3	\dots
#(L)		0	1	2	3	

- codificarea instrucțiunilor

tip IK	#(tip IK)
$v \leftarrow v$	0
$v \leftarrow v+1$	1
$v \leftarrow v-1$	2
IF $v \neq 0$ GOTO L	$\#(L)+2$

- codificarea instrucțiunii IK: $\#(IK) = \langle \#(\text{et IK}), \langle \#(\text{tip IK}), \#(\text{var IK}) \rangle \rangle$

eticheta

tipul inst.

nr variabilei în S

- codificarea stării curente: $\#(v) = 2^{p_1^{x_1} p_2^{z_1} p_3^{x_2} p_4^{z_2} \dots} = 2^{\prod_{i=1}^m p_{2i}^{x_i} \cdot \prod_{i=1}^m p_{2i+1}^{z_i}}$

Functia universală de m variabile

Def: Se numește funcție universală de m variabile funcția ($m \geq 1$)

$\phi^{(m)} : \mathbb{N} \rightarrow \mathbb{N}, \phi^{(m)}(\vec{x}, x_{m+1}) = \phi_{\#^{-1}(x_{m+1})}^{(m)}(\vec{x})$, deci $\phi^{(m)}(\vec{x}) = \phi^{(m)}(\vec{x}, \#(P))$

Sau: Fie P un program standard codificat astfel:

$$\#(P) = [\#(I_1), \#(I_2), \dots, \#(I_m)] - 1$$

$$\#(I_i) = \langle a, \langle b, c \rangle \rangle$$

și $f_p(x_1, \dots, x_m)$ funcția calculată de programul P

$\phi^{(m)}(x_1, x_2, \dots, x_m, x_{m+1})$ - funcția universală de m variabile

$$\phi^{(m)}(x_1, x_2, \dots, x_m) = f_p(x_1, \dots, x_m), \#(P) = x_{m+1}$$

ENUNT: Pe măsură că $m \geq 1$, $\phi^{(m)}(\vec{x}, x_{m+1}) = f_p(x_1, \dots, x_m)$ este s-calculară.

MONSTRATIE:

Scriem un program care calculeaza y_p .

- variabila de intrare x_{m+1} reprezinta codul progr.
- variabilele de intrare x_1, \dots, x_m reprezinta variab. de intrare ale progr.
- $K =$ variabila de intrare = nr de ordine al instructiunii pe care urmeaza sa se execute programul
- $s =$ variabilele de intrare = codul stocat curent al programului

IF $x_{m+1} = 0$ GOTO E

$$S \leftarrow \prod_{i=1}^m p_{2i}^{x_i}$$

$K \leftarrow 1$

[C]: IF $K = L(x_{m+1}) + 1$ GOTO F // final

IF $\ell(\tau(x_{m+1})_K) = 0$ GOTO N // incrementare (next)

IF $\ell(\tau(x_{m+1})_K) = 1$ GOTO P // plus

IF $\ell(\tau(x_{m+1})_K) = 2$ GOTO N

$K \leftarrow \min_{t \leq L(x_{m+1})} (\ell(\tau(x_{m+1})_t - \ell(\tau(x_{m+1})_K) - 2)$

[M]: GOTO C
 $S \leftarrow S / p_{2K}(\tau((x_{m+1})_K) + 1)$
GOTO N

[P]: $S \leftarrow S \cdot p_{2K}(\tau((x_{m+1})_K)) + 1$

[N]: $K \leftarrow K + 1$
GOTO C

[F]: $y \leftarrow S(1)$

SUBIECTUL 4

Multimi recursive, recursive enumerabile și NE-mui enumerabile.

Def: Pe集ă A ⊆ N, $\chi_A(x) = \begin{cases} 1, & \text{dacă } x \in A \\ 0, & \text{dacă } x \notin A \end{cases}$

Def: $A \subseteq N$, A se numește multime recursive (m. r-calcuabila sau m. decidabila) \Leftrightarrow e A oricare din afirmațiile:

- χ_A este Turing-calcuabilă

- χ_A este r-calcuabilă

- χ_A este funcție recursive

- problema apartenenței la A este decidiabilă

Def: $A \subseteq N$, A se numește multime recursive enumerabilă (m. semi-decidabila) \Leftrightarrow partial calculabilă sau m. semi-decidabilă \Leftrightarrow

- exist alg. care se oprește pe $x \in A$ și nu se oprește pt $x \notin A$

- $f: N \rightarrow N$ (f T-calc) și $A = \text{dom}(f)$

- $f: N \rightarrow N$ (f r-calc) și $A = \text{dom}(f)$

- $f: N \rightarrow N$ (f funcție parțial recursive) și $A = \text{dom}(f)$

$$R = \{A \mid A \in 2^N, A \text{ m. recursive}\}$$

$$RE = \{A \in 2^N, A \text{ m. recursive}\}$$

Def: $A \subseteq N$, A se numește multime merecursive enumerabilă dacă nu poate fi rezolvată de o masină Turing. (m. medecidabile)

ENUNȚURI:

(a) $RE \subseteq NRE$

(b) $R \subseteq RE$

(c) Proprietăți de închidere ale clasei R:

Dacă $L_1, L_2 \in R$ atunci $\bar{L}, L_1 \cup L_2, L_1 \cap L_2 \in R$.

Dacă $L_1, L_2 \in R$ atunci $\bar{L}_1, \bar{L}_2 \in R$.

Proprietăți de închidere ale clasei RE:

Dacă $L_1, L_2 \in RE \Rightarrow L_1 \cup L_2 \in RE, L_1 \cap L_2 \in RE$

Dacă $L, \bar{L} \in RE \Rightarrow L, \bar{L} \in R$.

Demonstratie:

(b) $R \subseteq RE$

\subseteq evident deoarece o funcție recursive este deoarece parțială

\supseteq Pp că $R \subseteq RE$. Cum, dacă $\bar{L} \in R \Rightarrow \bar{L} \in RE \Rightarrow$

(c) $\chi_L = \bar{\chi}_{\bar{L}} \Rightarrow \bar{L} \in R$

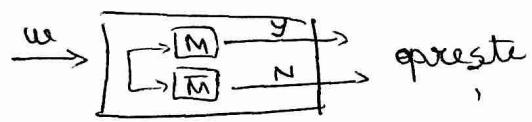
$$\chi_{L_1 \cup L_2} = \chi_{L_1} \vee \chi_{L_2} \Rightarrow L_1 \cup L_2 \in R$$

$$\chi_{L_1 \cap L_2} = \chi_{L_1} - \chi_{L_2} \Rightarrow L_1 \cap L_2 \in R$$

Fie M și \bar{M} 2 TM așă $L(M) = L, L(\bar{M}) = \bar{L}$

Comstr. TM A așă $\delta(w) = \begin{cases} 1, & \text{dc. } w \in M \\ 0, & \text{dc. } w \notin M \end{cases}$

- Δ simularea în paralel M și \bar{M}
- De undată ce M acceptă Δ se oprește în starea g_Y .
- De undată ce \bar{M} acceptă Δ se oprește în g_N



SUBIECTUL 5

Complexitate spațiu

Pentru calculul complexității spațiu se folosește modelul mașinii Turing offline care are o bandă de intrare și 2 benzi auxiliare. Banda de intrare este doar pentru citire, iar benzile auxiliare sunt marcate la unul din capete.

Mașina se oprește pe fiecare intrare.

$\text{SPACE}_M(m)$ - numărul maxim de călăi folosite de mașina M pe o bandă auxiliară până la oprirea sa pe o intrare de dimensiune m.

$\text{TIME}_M(m)$ - numărul maxim de pași pe care îl face mașina M pe o intrare de lungime m.

$$\text{ASPACE}_K(f(m)) = \{ L \mid \exists \text{ MT deterministă cu } K \text{ benzi cu } L = L(M) \text{ și } \text{SPACE}_M(m) \leq f(m), \forall m > m_0 \}$$

$$\text{NSPACE}_K(f(m)) = \{ L \mid \exists \text{ MT nedeterministă cu } K \text{ benzi cu } L = L(M) \text{ și } \text{SPACE}_M(m) \leq f(m), \forall m > m_0 \}$$

$$(N)(\Delta) \text{ SPACE}_K(f(m)) = \bigcup_{K \geq 1} (N)(\Delta) \text{ SPACE}_K(f(m))$$

Def: Funcția f se numește funcție spațiu-construibilă $\Leftrightarrow \exists \text{ TM offline} \text{ cu } m, f(w_m), |w_m| = m \text{ cu:}$

- $\text{space}_m(m) \leq f(m)$
- $\text{space}_m(w_m) = f(m)$

Def: Funcția f se numește funcție complet spațiu-construibilă $\Leftrightarrow \exists \text{ TM offline} \text{ cu } m, f(w_m), |w_m| = m \text{ cu } \text{space}_m(w_m) = f(m)$.

ENUNȚURI:

(a) Comprimarea spațiului de lucru cu un factor constant

$$(N)(\Delta) \text{ SPACE}_K(f(m)) = (N)(\Delta) \text{ SPACE}_K(c \cdot f(m)) + K \cdot c \cdot m^2.$$

(b) Reducerea nr. de benzi (nr. de benzi cu contrarăptirea spațiu)

$$(\Delta)(N) \text{ SPACE}_K(f(m)) = (\Delta)(N) \text{ SPACE}_1(f(m))$$

$$c) (\Delta)(N) \text{ TIME}(f(m)) \subseteq (\Delta)(N) \text{ SPACE}(f(m))$$

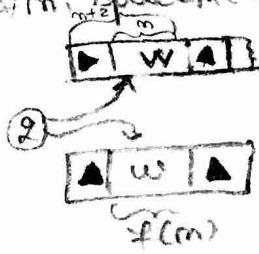
$$d) f(m) \geq \log m \Rightarrow \text{ASPACE}(f(m)) \subseteq \text{ATIME}(2^{\Omega(f(m))})$$

$$e) \text{ Teorema Savitch: } f(m) \geq \log m \\ f(m) \text{ complet} \\ \text{spațiu-construibilă}$$

$$\Rightarrow \text{NSPACE}(f(m)) \subseteq \text{ASPACE}(f(m))$$

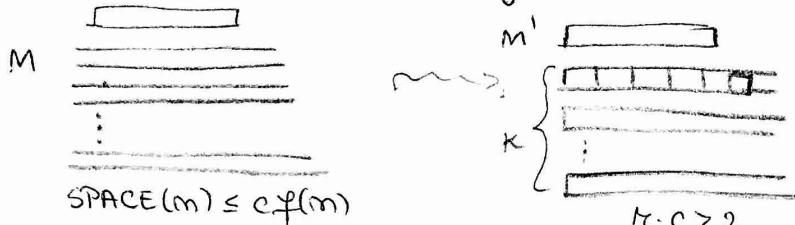
DEMONSTRATII:

(d) Fie $L \in \text{ASPACE}(f(m))$. Fie $M = (Q, \Sigma, \Gamma, \delta, q_0, F, \{g, f\})$ un TM offline cu $\text{space}_m(m) \leq f(m)$.



$$\begin{aligned} \text{config} &= (Q, \Sigma, \Gamma, \delta, q_0, F, \{g, f\}) \\ &\quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ &\quad \alpha_1 \quad m+2 \quad |P| \quad f(m) \quad f(m) \quad \dots \quad \leq f(m)+2 \\ \max &= |\alpha_1| (m+2) f(m) \cdot |P| = 2^{\log_2 |\alpha_1| + \log_2 (m+2)} \\ &\leq f(m) \cdot 2^{\log_2 |\alpha_1| + \log_2 (m+2)} = 2^{c_1 + c_2 f(m)} \leq 2^{f(m)}. \end{aligned}$$

(a) Fie M o mașină Turing cu K benzi.



O mișcare a mașinii M' simulează mișcările mașinii M pe intervalele de lungime k corespunzător dim M .

$$\text{SPACE}_{M'}(m) \leq \lceil \frac{\text{SPACE}_M(m)}{m} \rceil \leq \frac{c f(m)}{k} + 1 \leq f(m)$$

$$\Leftrightarrow \frac{f(m)+1}{2} \leq f(m) \Leftrightarrow f(m) \geq 2.$$

Dacă $f(m)=1 \Leftrightarrow \text{SPACE}_{M'}(m) \leq c \Rightarrow \text{SPACE}_{M'}(m)=1 \leq f(m)$.

! Complexitatea spațiu se măsoară cu funcția $\text{SPACE}_M(m): N \rightarrow N$.

(c) Fie M o TM. Presupunem că aceasta face x pași, ceea ce înseamnă că, făcând x pași, nu poate atinge mai mult de x căsuți, de unde rezulta concluzia.

SUBIECTUL 6 - COMPLEXITATE TEMP

Pentru calculul complexității timp se folosește modelul mașinii Turing cu K benzi, infinită la ambele capete și care se oprește pe fiecare intrare, iar capul de citire scriere poate stopa.

Complexitatea timp se măsoară cu funcția $\text{TIME}_M(m) : N \rightarrow N$.

- $\text{SPACE}_m(m)$ = numărul maxim de celule folosite de mașina M pe o bandă auxiliată până la oprirea sa pe o intrare de dimensiune m .
- $\text{TIME}_m(m)$ = numărul maxim de pași pe care îl face mașina M pe o intrare de lungime m .
- $\Delta\text{SPACE}_K(\varphi(m)) = \{L \mid \exists \text{ MT det. cu } K \text{ benzi cu } L = L(M) \text{ și } \text{TIME}_M(m) \leq \varphi(m), \forall m > m_0\}$
- $\Delta\text{TIME}_K(\varphi(m)) = \{L \mid \exists \text{ MT med. cu } K \text{ benzi cu } L = L(M) \text{ și } \text{TIME}_M(m) \leq \varphi(m), \forall m > m_0\}$
- $(\Delta)(N)\text{TIME}_K(\varphi(m)) = \bigcup_{K \geq 1} (\Delta)(N)\text{TIME}_K(\varphi(m))$.

Def: Fie $\varphi : N \rightarrow N$. Funcția φ se numește timp-construibilă dacă $m > 0$ și M e o TM care se oprește după exact $\varphi(m)$ pași pt $m > m_0$.

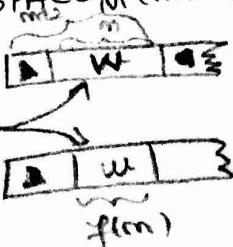
Def: Funcția φ se numește complet timp-construibilă dacă $\forall m$ și o mașină Turing M , M se oprește după exact m pași.

ENUNTURI:

1. Comprimarea timpului de lucru cu un factor constant
 $\lim_{m \rightarrow \infty} \frac{\varphi(m)}{m} = +\infty \Rightarrow (\Delta)(N)\text{TIME}_K(\varphi(m)) = (\Delta)(N)\text{TIME}_K(c \cdot \varphi(m)), c > 0$.
2. Reducerea numărului de benzi
 $(\Delta)(N)\text{TIME}_K(\varphi(m)) = (\Delta)(N)\text{TIME}_1(\varphi^2(m))$
3. $(\Delta)(N)\text{TIME}(\varphi(m)) \subseteq (\Delta)(N)\text{SPACE}(\varphi(m))$
4. $\varphi(m) \geq \log m \Rightarrow \Delta\text{SPACE}(\varphi(m)) \subseteq \Delta\text{TIME}(2^O(\varphi(m)))$
5. $\text{NTIME}(\varphi(m)) \subseteq \Delta\text{TIME}(2^O(\varphi(m)))$.

DEMONSTRATII:

4. Fie $L \in \Delta\text{SPACE}(\varphi(m))$. Fie $M = \{Q, \Sigma, \Gamma, d, q_0, \delta, \varphi\}$ o f.f.l.m. cu $\text{SPACE}_M(m) \leq \varphi(m)$.



 $\text{config} = (Q, \underbrace{w}_{\downarrow}, \underbrace{\Gamma}_{\downarrow}, \underbrace{w}_{\downarrow}, \underbrace{\Gamma}_{\downarrow}, \underbrace{\Gamma}_{\downarrow}, \underbrace{\Gamma}_{\downarrow})$
 $\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$
 $|a| \quad 1 \quad m+2 \quad |\Gamma|^{\varphi(m)} \quad |f(m)| \quad |f(m)| \quad |f(m)|$

$$\max = |a|(|m+2| \varphi(m) \lceil \varphi(m) \rceil) = 2^{\log_2 |a|} + 2^{\log_2 (m+2)} + 2^{\log_2 (\varphi(m))} \leq \varphi(m) + 2$$

$$+ \varphi(m) \log_2 |\Gamma| = 2^{C_1 + C_2 \cdot \varphi(m)} \in 2^{O(\varphi(m))}$$

5. Fie M o K-NTM. Construim M' ($K+2$) TM cu $L(M) = L(M')$.

B_1' - curăntul de intrare w care nu se va modifica. La stg. lui w punem $\#$ și la stânga $\#$ -ului avem o sau 1.

0 = nu am determinat arborele de calcul al lui M

1 = am determinat procesul de calcul.

B_2' - se rulează generația codurilor drumurilor prin arborele $T_f(m)$

$B_3' - B_{K+2}'$: cele K benzi rulează și folosite pentru simularea funcționalității

intrării: w , cu $w1=m$ pe B_1'

PAS 1: scriem $\#$ la stânga lui w pe B_1' și pozitionăm capul pe prima literă a lui w (4 pași - $O(1)$)

PAS 2: - generează codul următorului drum de simulat în $O(f(m))$ $T_f(m))$

- dacă am trecut pe următorul nivel din arborele $T_f(m)$ atunci, dacă avem $\#$ la stânga $\#$ pe B_1' , atunci M respinge.
 - dacă avem $\#$ la stânga $\#$ pe B_1' , atunci scriem $\#$ în locul lui și trecem la PAS 3.

PAS 3: Copiem w de pe B_1' pe B_3' și aducem capetele pe B_1' și B_3' pe prima literă a lui w .

PAS 4: Se simulează m pe urmărea w în mod deterministic pe $O(f(m))$ drumul codificat pe B_2' .

- dacă codul este invalid \Rightarrow PAS 5
- dacă M acceptă $\Rightarrow M'$ acceptă
- dacă M nu se oprește, atunci scriem $\#$ la stg. $\#$ pe B_1' și PAS 5
- dacă M respinge

PAS 5: - sterg $B_3' - B_{K+2}'$ și PAS 2:

$O(f(m))$

Total: $O(1) + O(f(m)) + O(f(m)) + O(m) \cdot 2^{O(f(m))} = 2^{O(f(m))}$.

SUBIECTUL 7

Terarhii de clase de complexitate timp și spațiu

- Modele de TM pt timp și spațiu
- Def. măsurii și a claselor de complexitate spațiu și timp
- Funcții spațiu-timp-construibile, complet și construibile.
- Codificarea binară a TM.

} SB. 6

$$M = (Q, \Sigma, \Gamma, \delta, q_1, F)$$

$$Q = \{q_1, q_2, \dots, q_s\}$$

$$\Sigma = \{0, 1\}$$

$$\Gamma = \{\square, \square, z_4, z_5, z_x\}$$

$$F = \{q_2\}$$

$$\delta: Q \times F \times \Gamma^* \rightarrow Q \times \Gamma^* \times \{ \leftarrow, \rightarrow \}^*$$

Q		2		21	22	...	2s
#(q)		0 ¹	0 ²		0 ^s		

Σ		0	1
#(Σ)		0 ¹	0 ²

Γ		z		0 1 \square z_4 ... z_s
#(Γ)		0 ²	0 ³	0 ⁴ ... 0 ^s

Δ		d		\leftarrow \rightarrow
#(Δ)		0 ¹	0 ²	0 ³

$$\delta = \{g_i(z_{i1}, \dots, z_{ik}), g_j(z_{j1}, \dots, z_{jk}), (\Delta_1, \dots, \Delta_K) \mid \delta(g_i, \vec{z}_i) = (g_i, \vec{z}_j, \vec{\Delta})\}$$

$$\#(t) = 0_1 0_1 \dots 0_1 0_1 0_1 \dots 0_1^{ik}$$

$$\#(\Delta_1) \dots \#(\Delta_K) \in (0^1)^{2K-2} (\{0, 0^2, 0^3\})^K$$

$$\# M = 1^3 \#(t_1), 1 \#(t_2) \dots \#(t_m) \in \bigcup_{K \geq 1} 1^3 (0^1)^{2K+2} (\{0, 0^2, 0^3\})_1^K$$

$$w \in \{0, 1\}^*, M w$$

$$W_m = \underbrace{1 \dots 1}_m \#(M) \mid m \geq 0 \rangle$$

ENUNȚURI:

1. Existența hierarhiilor de clase de complexitate spațiu și timp

(i) $T(m)$ recursive, $\exists L$ recursive ($L \notin \text{TIME}(T(m))$)

(ii) $S(m)$ recursive, $\exists L$ recursive ($L \notin \text{SPACE}(S(m))$)

2. Hierarhie refeinatoare de clase de complexitate ~~spatiu~~ spațiu

$$S_1(m) \geq \log m, \lim_{m \rightarrow \infty} \frac{S_1(m)}{S_2(m)} = 0$$

$$S_2(m) \text{ complet spațiu - construibile} \Rightarrow \text{SPACE}(S_1(m)) \subsetneq \text{SPACE}(S_2(m))$$

3. Hierarhie reafinată de clase de complexitate timp

$$\lim_{m \rightarrow \infty} \frac{T_1(m)}{T_2(m)} \log \frac{T_1(m)}{T_2(m)} = 0$$

$\Rightarrow \Delta\text{TIME}(T_1(m)) \subsetneq \Delta\text{TIME}(T_2(m))$

$T_2(m)$ complet spațiu-construibilă

Demonstratii:

1. Fie $T(m)$ recursivă. Fie $TM M$ care calculează $T(m)$. Consideră $L = \{w \in \{0,1\}^* \mid Mw \text{ nu este acceptat în } T(m) \text{ pașii}\}$

(a) L recursivă

(b) $L \notin \Delta\text{TIME}(T(m))$

(a) Construim M' așa că M' decide L
Descrierea lui M' :

- intrare curând următoare de lungime m
 - M' simulează pe M și calculează $T(m)$
 - decodifică Mw din w
 - dacă w nu e un cod valid, atunci M' respinge
 - M' simulează Mw pe intrarea w
 - dacă Mw nu se oprește după $T(m)$ pași $\Rightarrow M'$ acceptă
 - M' acceptă $\Leftrightarrow Mw$ respinge
- Evident, $L(M) = L$

(b) Prăvîl absurd că $L \in \Delta\text{TIME}(T(m))$. Fie $TM M_1$ cu $\begin{cases} L(M_1) = L \\ M_1 \text{ lucraza în } \Delta\text{TIME}_{M_1}(m) \leq T(m) \end{cases}$
Fie un cod w al lui M_1 .

$w \in L(M_1) = L \Leftrightarrow Mw = M_1$ nu acceptă w $\Leftrightarrow w \notin L(M_1)$ \Leftarrow

2. Fie M_2 o mașină care calculează $s(m)$.
Construim $TM M$ cu prop:

- (a) M lucraza în $\Delta\text{SPACE}(s_2(m))$ dacă $L(M) \in \Delta\text{SPACE}(s_2(m))$
- b) M diferă de orice $TM M_1$ ca mod de lucru, cu $L(M_1) \in \Delta\text{SPACE}(s_1(m))$ pe cel puțin o intrare $\Rightarrow L \notin \Delta\text{SPACE}(s_1(m))$.

(a) Intrare: w așă $|w| = m$

Descrierea mașinii M :

- M marchează pe o bandă $s_2(m)$
- M decodifică Mw din w
- dacă w nu e un cod valid $\Rightarrow M$ respinge
- M simulează Mw pe intrarea w pe banda marcață (cu $s_2(m)$ casută)
- dacă Mw nu se oprește în spațiul marcață $\Rightarrow M'$ respinge
- M respinge $\Leftrightarrow M'w$ acceptă

(b) Fie M_1 o TM cu prop. $L(M_1) \in \Delta\text{SPACE}(s_1(m))$.

$$\text{fim } M_1 = Mw_1 \quad \boxed{\dots z_1 z_2 \dots z_m \dots}$$

$$\text{card}(\Gamma_{Mw_1}) = y_1$$

$$\text{fim } M_1 \text{ simulează } Mw_1 \leq s_1(m) \log_2(y_1) \leq s_2(m)$$