

I MASINI TURING

Def. informala:

○ masina Turing este un "ansamblu" format dintr-o banda memoranta si un cap de citire-scriere. Capul de citire-scriere se poate deplasa atat la stanga cat si la dreapta.

Def. matematica a masinii Turing determinante

$M(Q, V, U, \delta, q_0, F, B)$, unde

1. Q = multimea finita de stari
2. V = alfabetul de intrare
3. U = alfabetul de lucru $V \subseteq U$
4. $q_0 \in Q$ = starea initiala
5. F = multimea stariilor finale
6. B = Blank
7. $\delta: (Q \setminus F) \times U \rightarrow Q \times (U \setminus \{B\}) \times \{L, R\}$

~~Masinile Turing pot fi: determinante si medeterminante~~

Def. matematica a masinii Turing medeterminante:

$M(Q, V, U, \delta, q_0, F, B)$

Def. matematica a masinii Turing cu mai multe bani determinante:

Diferenta apare la functia de transitiune:

$\delta: (Q \setminus F) \times U \rightarrow Q \times (U \setminus \{B\}) \times \{L, R\}$

$\delta: (Q \setminus F) \times U \rightarrow 2$

Def. matematica a masinii Turing cu mai multe bani determinante:

$M = (m, Q, V, U, \delta, q_0, F, B)$

$m = m$ de bani

$\delta: (Q \setminus F) \times U^m \rightarrow Q \times (U \setminus \{B\})^m \times \{L, R\}^m$

Def. matematica a masinii Turing medeterminanta cu mai multe bani:

$M = (m, Q, V, U, \delta, q_0, F, B)$

$\delta: (Q \setminus F) \times U^m \rightarrow 2^{Q \times (U \setminus \{B\})^m \times \{L, R\}^m}$

$\delta: (Q \setminus F) \times U^m \rightarrow 2$

Teorema 1

Orice matrice Turing M cu m liniile poate fi simulață ca o.
 Orice matrice Turing M' cu o singură bandă. În plus, dacă $M = \text{det}$
 $\Rightarrow M' = \text{determin}.$

Teorema 2

Orice matrice Turing medeterminată cu o bandă poate fi simulață de o matrice Turing determinată cu 3 liniile.

~~MT ca dispozitive de acceptare / calc. de funcții~~

- ~~Oricare dintre aceste mașini, privite ca dispozitive de acceptare, neumorse, liniile generate de această gramatică, limbi și recursiv enumerabile, generate de gramatica de tipul 0.~~
- ~~Oricare dintre aceste mașini, privite ca dispozitive de calcul, pot calcula orice funcție din clasa R (probleme decidaibile), echivalente cu limbajele recursive, adică pt. orice intrare finită mașina se va opri)~~

Demonstratie teorema 1:

Fie $M \in MT$ cu m liniile $M(m, Q, V, U, \Sigma, f_0, F, B)$

$M_1:$	1: $\boxed{\quad \quad \quad a_1 }_{\Delta_1}$
2:	$\boxed{\quad \quad a_2 }_{\Delta_2}$
\vdots	\vdots
$m:$	$\boxed{\quad \quad a_m }_{\Delta_m}$

$M'_1:$	1 < $\boxed{0 \quad 0 \quad 1 \quad \dots \quad 0}_{a_1}$
2 <	$\boxed{0 \quad 0 \quad \quad \quad 1 \quad 0}_{a_2}$
$2m <$	$\boxed{0 \quad 1 \quad \quad \quad \quad 0}_{a_m}$

Pe punctele de ordin împar: 0, 1
 Pe punctele de ordin par: simboluri.

- ~~1) Poziționăm capul de citire pe prima celule~~
~~2) Memorăm starea~~

Pas 1 Memorarea starea și apoi simbolurile de pe partea de ordin par (a_1, a_2, \dots, a_m)

Pas 2 M' se poate îmota de la dreapta la stânga, și să traseze simbolurile a_1, a_2, \dots, a_m cu lui $\leftarrow, \rightarrow, \downarrow, \uparrow$

Pas 3 M' se poate îmota de la stânga la dreapta, și să traseze pozițiile și pe partea înspate

Demonstratie teorema 2 (cu backtracking)

Considerăm $M = \Theta$ mașină Turing nedeterministă. Încercăm să construim o mt M' deterministă cu 3 biene care să simuleze M

Notăm W = multimea configurațiilor posibile ale lui M (finită)
 $W = \{(z, q, s, b, \Delta) \mid z = \text{stare mecanică}, q, b = \text{simboluri}, s = \text{stare}, b = \text{simbol} \neq \text{blank}, \Delta = \text{direcție (L sau R)}\}$

Deoarece W = finită \Rightarrow poate fi ordonată lexicografic

M' : - pe bioma 1 are configurația inițială
 - copiază conținutul lui B_1 pe B_3
 - scrie pe B_2 primul element din W

1. Citeste simbolul curent de pe B_2 (z, a, b, Δ)

2. Verif. dacă se află în starea q'
 2.1. Dacă nu, efectuează R (parul de retrur)

3. Verif. dacă simbolul citit de pe B_1 este a

4. Seriem b peste a

5. Schimbăm starea în q'

6. Deplasează capul de pe B_1 în direcția Δ .

7. Deplasează capul de citire de pe B_2 la dreapta

Se va relua de la parul 1, dacă există un simbol pe B_2 .
Dacă nu, se efectuează R .

- R :
- Repetă w_i de pe B_3 pe B_1
 - Scrie pe B_2 succesorul elementului de pe bandă
 - Întră în S_0

MT. ca dispozitive de acceptare / calcul ale funcții.

- MT pot fi fol. ca dispoz. de acceptare atunci când la intrare pe bandă se află un cuvant w_i , iar mașina se poate opri și să accepte/să nu accepte intrarea, dar poate să nu se oprească.
- MT. pot fi fol. ca dispozitive de calcul atunci când este practic o funcție parțială $f: \mathbb{N}^k \rightarrow \mathbb{N}$. Dacă mașina se oprește pe intrarea (x_1, \dots, x_k) , atunci acolo este rezultatul $f(x_1, \dots, x_k)$. Dacă mașina nu se oprește, atunci f nu este definită în (x_1, \dots, x_k) .

II Functă recursive. Functă Turing calculabile.

Functă calculabile cu programme standard

Definiții

1) Programul standard este format dintr-un set de instrucțiuni și terminarea se face fie prin salt la eticheta E, fie prin salt la o etichetă inexistentă, fie transferul se face la sfârșitul programului.

~~Output = valoarea cui și la finalul programului.~~

Limbajul abstract T va calcula $f: \mathbb{N}^* \rightarrow \mathbb{N}$.

- variabile → de intrare: x_1, \dots, x_m
→ de ieșire: y
→ locale: z_1, \dots, z_n

Variabilele de lumen și cea de ieșire sunt initializate cu 0.

- etichete: E, A₁, A₂, ..., A_m
- instrucțiuni:
 - $v \leftarrow v$ (efect nul)
 - $v \leftarrow v+1$ (incrementare)
 - $v \leftarrow v-1$ (decrementare dacă $v \neq 0$)
 - IF $y \neq 0$ GOTO L ; → $v=0$ efect nul, se face la intrare.
→ ~~v=0 transferul se face astfel;~~
→ la prima instrucție L dacă $L \neq$

2) Functii Turing calculabile:

Se numește f. Turing calculabilă funcția pt. care se poate scrie o MT.

3) Funcții recursive: $\exists f$ f. = recursivă dacă se poate obțin, din f. elementare, prin compunerea operațiilor alei comp. funcții, recursivitate, minimum, maxim.

• Funcții elementare:

(1) Succesor:

$$\text{succ}: \mathbb{N} \rightarrow \mathbb{N}.$$

$$\text{succ}(x) = x+1$$

(2) Constantă:

$$c_k^{(m)}: \mathbb{N}^m \rightarrow \mathbb{N}, c_k^{(m)}(x_1 \dots x_m) = k.$$

(3) Proiecție:

$$p_k^{(m)}: \mathbb{N}^m \rightarrow \mathbb{N}.$$

$$p_k^{(m)}(x_1, \dots, x_m) = x_k.$$

• Operări:

(1) Compunerea funcțională: (CF)

Functia $f: \mathbb{N}^k \rightarrow \mathbb{N}$ se obține din compunerea funcțională a funcțiilor $g_i: \mathbb{N}^k \rightarrow \mathbb{N}$, $i=1, m$ și $h: \mathbb{N}^m \rightarrow \mathbb{N}$. Dacă

$$f(x_1, \dots, x_k) = h(g_1(x_1, \dots, x_k), g_2(x_1, \dots, x_k), \dots, g_m(x_1, \dots, x_k))$$

(2) Recurență primitivă: (RP)

$f: \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ se obține prin recurență primitivă din funcția $g: \mathbb{N}^k \rightarrow \mathbb{N}$ și $h: \mathbb{N}^{k+2} \rightarrow \mathbb{N}$ dacă:

$$f(x_1, \dots, x_k, 0) = g(x_1, \dots, x_k)$$

$$f(x_1, \dots, x_k, t+1) = h(x_1, \dots, x_k, t, f(x_1, \dots, x_k, t)))$$

(3) Minimizare nemărginită (\mathbb{N}^m)

$f: \mathbb{N}^k \rightarrow \mathbb{N}$ se obține din funcția $g: \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ prin minimizare nemărginită dacă $f(x_1, \dots, x_k) = \min_t [g(x_1, \dots, x_k, t) = 0]$

$$\underbrace{\mathbb{N}^k, \mathbb{N}^k, \mathbb{N}^m}_{\mathbb{N}^k}, \underbrace{\mathbb{N}^{k+1}, \mathbb{N}^*, \mathbb{N}^{k+2}}_{\mathbb{N}^m}, \underbrace{\mathbb{N}^k, \mathbb{N}^{k+1}}_{\mathbb{N}^{k+1}}$$

Teorema 1

Orice funcție calculabilă cu programe standard este Turing calculabilă.

Teorema 2

Orice funcție recursivă este calculabilă cu programe standard

Teorema 3

Orice funcție Turing calculabilă este recursivă.

C	T
R	C
T	R

Demonstrare Teoremei 3

Fie $M \in MT$ deterministă și fie $f(x_1, \dots, x_m)$ funcția calculată de M

Părușumem că $m=1 \Rightarrow M$ calculează $f(x)$

Fie $\{s_0=0, s_1=1, \dots\}$ = simbolurile ce pot opărea pe bandă unei MT

Numerotăm celulele memorii cu $0, 1, \dots$

Fie $\{q_0, q_1, \dots\}$ = stări ce pot opărea în definiție MT .

Fie $\{a, b, c\}$ = simboluri de citire/scriere pe bandă memorii

la un pas nondeterminat următorul număr configurației

curente: $\langle a, b, c \rangle$

a = identificatorul stării curente

b = pos. copului de citire/scriere pe bandă memorii

c = numărul Gödel

Definim $C_M(x, m) =$ numărul atâtat configurației maximă la pasul m pe intrarea x . Dacă maximă se oprește după m pas, atunci definim $C_M(x, m) = C_M(x, m_0)$

Definim funcțiile auxiliare:

$\circ h_1(z) = \begin{cases} \text{numărul stării în care trece } M \text{ din config cu numărul } z \\ \text{dacă acest nr. codifică o config validă în } M \\ 0, \text{ altfel} \end{cases}$

$\circ h_2(z) = \begin{cases} \text{numărul celulei de pe bandă unde se poziionează eșul} \\ \text{i/o după config cu numărul } z, \text{ dacă ac. nr. codifică o config} \\ \text{validă în } M \\ 0, \text{ altfel} \end{cases}$

$$\bullet h_3(z) = \begin{cases} \text{numărul configurației binară după config cu numărul } s, \\ \text{dacă ac. nr. codificat este validă în } M \\ 0, \text{ altfel} \end{cases}$$

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

Fie a numărul unei stări și b numărul unui simbol. Definim

$$\bullet g_1(a, b) = \begin{cases} \text{numărul stării } s_m \text{ care trece } x_1 \text{ din starea } a, \\ \text{cînd } b, \text{ dacă } a \text{ și } b \text{ sunt valide} \\ 0, \text{ altfel} \end{cases}$$

$$\bullet g_2(a, b) = \begin{cases} \text{numărul simbolului } b \text{ scris pe locul } x_1 \text{ din starea } a \\ \text{cînd } b, \text{ dacă } a \text{ și } b \text{ sunt valide} \\ 0, \text{ altfel} \end{cases}$$

$$\bullet g_3(a, b) = \begin{cases} 0, \text{ dacă } z, \text{ dacă } M \text{ se deplasează la stările sau la} \\ \text{dreapta din starea } a \text{ cînd } b, \text{ dacă } a \text{ și } b \text{ sunt valide} \\ \text{altfel } 0. \end{cases}$$

$$K = \underbrace{l(z)}_{K_1}, \underbrace{(r(r(z)))}_{K_2} e(r(z)),$$

$$h_1(z) = g_1(l(z), (r(r(z))) e(r(z))) = g_1(K)$$

$$h_2(z) = l(r(z)) + g_3(l(z), (r(r(z))) \underbrace{e(r(z))}_{l(r(z))}) - 1 = l(r(z)) + g_3(K) - 1$$

$$h_3(z) = r(r(z)) / \underbrace{P_{l(r(z))}}_{K_2} * \underbrace{P_{l(r(z))}}_{g_2(l(z), r(r(z))) e(r(z))} = \cancel{\frac{r(r(z))}{P_{l(r(z))}}}$$

$(g_1, g_2, g_3, h_1, h_2, h_3, C_M)$ recursive

$m_{M'}(x) = \text{nr. de pași pe care } M \text{ îl face pt. a calcula } f(x)$

$f(x) = \text{rezolvă}$

$$f \text{ se poate scrie: } f(x) = L_t(r(r(C_M(x, m_{M'}(x)))))) - 1$$

$\Rightarrow f = \text{recursivă.}$

$$h_3(z) = r(r(z)) / \underbrace{P_{l(r(z))}}_{K_2} * \underbrace{P_{l(r(z))}}_{K}$$

III PROGRAMUL UNIVERSAL + PROBLEMA OPRIRII

Functia universală de m variabile:

$\sigma^{(m)}(x_1, \dots, x_m, t)$ unde: x_1, \dots, x_m = variabilele de intrare ale unui program standard
 t = codificarea

Teorema 1

Pentru orice $m \geq 1$, funcția $\sigma^{(m)}$ este calculabilă.

Demonstratie:

$T \leftarrow x_{m+1} + 1$ (numărul codificării lui σ)

$S \leftarrow \prod_{i=1}^m P_{2^i}^{x_i}$ (tarea programului)

$K \leftarrow 1$ (comptor)

c: $\text{IF}(K > L_T(T) \vee (K=0)) \text{ GOTO F}$ (dacă K a depășit nr. de instrucțiuni de nulat)

$i \leftarrow C_T K$ (numărul instr. de executat)

$V \leftarrow l(r(i))$ (numărul variabilei din instrucțiunea i)

$U \leftarrow r(r(i))$ (tipul instrucțiunii)

$\text{IF } U=0 \text{ GOTO N}$ (efect nul)

$\text{IF } U=1 \text{ GOTO A}$ (incrementare)

$\text{IF } U=2 \text{ GOTO D}$ (decrementare)

$\text{IF } P_V \neq S \text{ GOTO N}$

$K \leftarrow \min_R [U-2 = l((T)R)]$

GOTO C

A: $S \leftarrow S * R$

GOTO N

B: $\text{IF } P_V \neq S \text{ GOTO N.}$
 $S \leftarrow S / P_V.$

N: $K \leftarrow K + 1$

GOTO C

F: $y \leftarrow CS$

Simbolul # încearcă codificarea. Codificarea lui P este de formă:

$$\#(P) = [\#(i_1), \#(i_2), \dots, \#(i_3)]$$

#(i) este de formă $\langle a, b, c \rangle$ unde:
 $\begin{cases} a = \text{eticheta } (a \text{ dacă } i \text{ nu are etichetă}) \\ b = \text{numărul valorii} \\ c = \text{tipul construcției} \end{cases}$ #(L)dacă are

Problema oprișii

Dată codificarea unui program $\#(P)$, notată cu y , și o intrare pt acel program, notată cu x , se poate calcula $\text{HALT}(x, y)$?

$\text{HALT}(x, y) = 1$, dacă programul se term.
= 0 dacă programul nu se term.

Teorema 2 : Predicatul HALT nu se poate calcula.

Demonstratie:

- Pre presupunem HALT e calculabil

Consider

- Fie P' programul care calculează HALT

- Considerăm programul :

$A: z_{r+1} \leftarrow \text{HALT}(x_1, x_1) \quad \{ P_1 \}$

~~■~~ IF $z_{r+1} \neq 0$ GO TO A

$$\#(P_1) = t$$

$$\text{HALT}(x_1, t) \Leftrightarrow z_{r+1} = 0 \text{ în } P_1$$

$$\Leftrightarrow \neg \text{HALT}(x_1, x)$$

dar $x = t$; $\text{HALT}(t, t) \Leftrightarrow \neg \text{HALT}(t, t)$ contradicție

Codificarea la funcții recursive

IV Multimi recursive / recursive enumerabile / merecursive

Functia caracteristica a unei multimi = o functie predicat care primeste ca argument un element al unei multimi (x) si returneaza 1 daca elementul apartine unei multimi. Atunci $x \in S$ este de fapt

C multime X este recursive daca functia sa caracteristica este

Turing calculabila si masina se opreste pe fiecare intrare

C mult. X este recursive enumerabila daca f sa caracteristica =

Turing calculabila si masina nu se opreste pe fiecare intrare

C mult. X este merecursive daca f sa caracteristica ~~nu este~~ Turing calculabila

Teorema 1

~~Abia~~ X = recursive \Leftrightarrow ~~daca~~ recursive enumerabila \Leftrightarrow $C(x)$

Teorema 2

X, Y = recursive $\Rightarrow X \cup Y$ = recursive

Teorema 3

X, Y = recursive $\Rightarrow X \cap Y$ = recursive

Teorema 4

X, Y = recursive enumerabile $\Rightarrow X \cup Y$ = recursive enum.

Teorema 5

X, Y = recursive enum. $\Rightarrow X \cap Y$ = recursive enum.

(2-5): prop. de includere

prop de
includere

C proprietate a unei clase de limbaje C este o multime S a lui C Propri = triviala daca $S = C$ sau $S = \emptyset$. Altfel = metriviala.

Teorema Rice

Care prop. metrivială pe RE este medecidabilă?

Demonstratie Rice:

Fie S = prop. metrivială pe RE și $\varnothing \neq S$. Arătăm că L_S nu este recursiv.

Pp. că $L_S = L(M_S)$, M_S - se opn. pe fiecare intrare

Fie $L \in S$ a.s. $L = L(M_L)$. Alegem și fixăm $\langle\langle M \rangle, w \rangle$ și construim

M' a.s. $L(M') = \{L\}$, dacă $\langle\langle w \rangle\rangle \in L(M)$
 { altfel

Cum calculează M' :

- inițial x_1 ignorează intrarea $1aX$ și simulează M pe w
- dacă M acceptă atunci simulează M_L pe X
- acceptă și dacă M_L acceptă

Construim M_L astfel:

- Pe intrarea $\langle\langle M \rangle, \langle w \rangle\rangle$ determinăm $\langle\langle M' \rangle$
- Simulează M_S pe $\langle\langle M' \rangle$
- Acceptă și dacă M_S acceptă (M_S se opn. pe fiecare intrare)

Observație: M_L se opn. pe fiecare intrare

$L \in S \Leftrightarrow \langle\langle M' \rangle\rangle \in L(M_S) \Rightarrow \langle\langle M \rangle, \langle w \rangle\rangle \in L(M_L)$, contradicția că $\varnothing \neq S$.

Dem. prop. de închidere:

Că prop. a unei clase de limbi C este o submult. S a lui C . Prop = trivială dacă $S = C$ sau $S = \varnothing$. Altfel = metrivială.

Alegem că C este clasa limbielor recursiv enum. RE.

Pt o prop. S a clasei RE definim $L_S = \{\langle\langle M \rangle\rangle / L(M) \in S\}$

V CLASA DE COMPLEXITATE

TIIMP

Pt. calculul complexității timp se folosește o MT cu k benni' infinită la ambele capete, unde se oprește pe fiecare intrare, în acel de citire-scriere poate stopa.

Pt o MT vom nota: $\text{TIME}_M(m) = m$ numărul maxim de pasi pe care îl face MT M pe o intrare de lungime m .

$$\underline{\text{TIME}_K(f(m))} = \begin{cases} L | \exists \text{ o MT determin. cu } k \text{ benni' s.t. } L = L(M) \leq f(m) \\ \text{TIME}_M(m) \leq f(m), \forall m \geq m_0 \end{cases}$$

$$\underline{\text{TIME}_K(f(m))} = \begin{cases} L | \exists \text{ o MT medetermin. cu } k \text{ benni' s.t. } L = L(M) \leq f(m) \\ \text{TIME}_M(m) \leq f(m), \forall m \geq m_0 \end{cases}$$

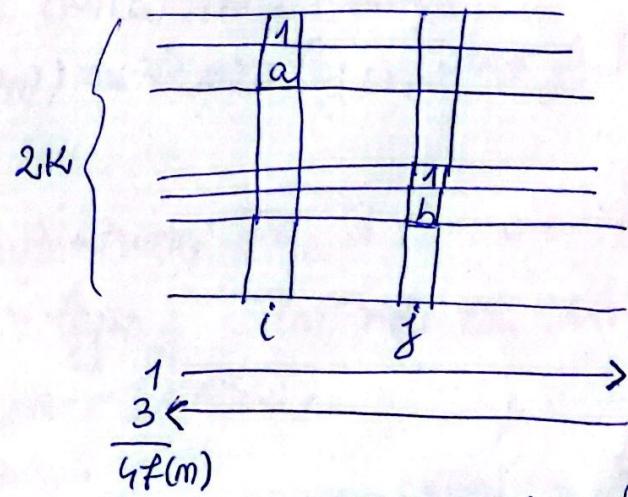
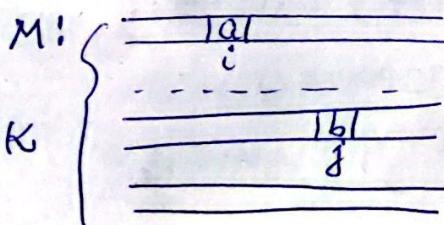
I Eliminarea comunitelor:

$$\lim_{m \rightarrow \infty} \frac{f(m)}{m} = +\infty \Rightarrow (N)(D) \text{TIME}_K(f(m)) = (N)(D) \text{TIME}_K(c \cdot f(m))$$

II Comprimarea bennilor:

$$\text{Dacă } L \in (N)(D) \text{TIME}_K(f(m)) \Rightarrow L \in (N)(D) \text{TIME}_1(4f^2(m)) \quad (\forall K \geq 1)$$

Demonstratie:



- Procedăm ca la simularea unei MT eu mai multe benni' de la face
- MT eu o bandă d.p.d.v al construcției
- MT eu mult $4f(m)$ mișcări pt fiecare mișcare a lui M
- M' face ul mult $4f(m) \cdot f(m) = 4f^2(m)$
- $\text{TIME}_M(m) \leq 4f(m) \cdot f(m) = 4f^2(m)$
- "Accelerat" M a.c. $\text{TIME}_M(m) \leq \frac{1}{2}f(m)$
- $\text{TIME}_{M'}(m) \leq 4 \left(\frac{1}{2}f(m) \right)^2 = f^2(m)$

mă: Rezolvare f(m) total recursivă ~~f(m)~~ excepțional recursiv

E: $L \in (N)(\Delta) \text{ TIME } (\mathcal{f}(m))$

Denumire: Fie $f(m)$ -recursivă

compr.: $L \in \{0, 1\}^*$, $L = \{w \mid M_w \text{ nu acceptă } w \text{ în timp } f(m)\}$

• L este recursiv:

Fie M' : -imput w :

- calc. $f(|w|)$ și marchează ~~f(|w|)~~ rețele pe o bandă
- găsește M_i
- simulatoare pe M_i pe intrarea w ; i acceptă dacă M_i nu acceptă w în $f(|w|)$ timp

• $L \in \Delta \text{ TIME } (\mathcal{f}(m))$

Preupoznem că $L(\bar{M}) = L$, și $\text{time}_M(m) \leq \mathcal{f}(m)$

- căresc j , a. s. $\bar{M} = M_j$, găresc w_j

- $w_j \in L \Leftrightarrow w_j$ este acceptat de M_j în timp $\mathcal{f}(m) \Leftrightarrow w_j \in L$

Relații:

(1) ~~(N)(Δ) TIME (f(m)) ⊆ (N)(Δ) SPACE (f(m))~~

(2) Dacă $L \in \Delta \text{SPACE}(\mathcal{f}(m))$, cu $\mathcal{f}(m) \geq \log m$, atunci $\exists C_L$ a. s.
 $L \in \Delta \text{TIME}(C_L \mathcal{f}(m))$

(3) Dacă $L \in N \text{TIME}(\mathcal{f}(m))$, atunci $\exists C_L$ a. s. $L \in \Delta \text{TIME}(C_L \mathcal{f}(m))$

(4) Savitch: Dacă $S(m) \geq \log m$ și $S(m)$ este scăzut, atunci
 ~~$N \text{SPACE}(S(m)) \subseteq \Delta \text{SPACE}(S^2(m))$~~ ✓

Teranjii de clase de complexitate:

Teorema 1: Dacă $L \in \text{TIME}_k(\mathcal{f}(m))$, atunci $L \in \text{TIME}_2(\mathcal{f}(m) + \log \mathcal{f}(m))$

3. funcție $f(m)$ este timp calculabilă/combinabilă/completă (TCC)

Dacă $\exists \sigma \in M$ a. s. pt orice $m \geq 1$ orice w cu $|w|=m$ avem:

$$\text{time}_M(w) = f(m)$$

VII CLASA DE COMPLEXITATE SPATIU.

Pt calculul complexității spațiu se folosesc o MT și o bandă de intrare și șase benzi auxiliare. Bandă de intrare este doar pt. citire, iar benzile auxiliare sunt mărginite la ambele dimuri capete. MT se oprește pe fiecare intrare

Pt o MT vom nota: $\text{SPACE}_M(m) = m$ nr maxim de celele folosite de MT pe o bandă auxiliară numărată de la oprirea sa pe o intrare de lungime m .

$$\Delta \text{SPACE}_K(f(m)) = \left\{ L \mid \exists \text{ o MT determin. cu } K \text{ benzi } \text{a.s.t. } L = L(M) \text{ și} \right. \\ \left. \text{SPACE}_M(m) \leq f(m), \forall m \geq m_0 \right\}$$

$$N \text{PACE}_K(f(m)) = \left\{ L \mid \exists \text{ o MT nondetermin. cu } K \text{ benzi } \text{a.s.t. } L = L(M) \text{ și} \right. \\ \left. \text{SPACE}_M(m) \leq f(m), \forall m \geq m_0 \right\}$$

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

I Eliminarea constantelor

$$(N)(\Delta) \text{PACE}_K(f(m)) = (N)(\Delta) \text{PACE}_K(C, f(m))$$

Dem: 6 mijloace ale mașinii M să numără mijloacele unei mașini M pe intervalul de lungime r și correspunțor dim M

$$\text{SPACE}_{M'}(m) \leq \left\lceil \frac{\text{SPACE}_M(m)}{m} \right\rceil \leq \frac{C f(m)}{r} + 1 \leq f(m)$$

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

$$\text{Dacă } f(m) = 1 \Leftrightarrow \text{SPACE}_M(m) \leq C = \text{SPACE}_{M'}(m) = 1 \leq f(m)$$

II Comprimarea benzilor

$$\text{Dacă } L \in (N)(\Delta) \text{PACE}_K(f(m)) \Rightarrow L \in (N)(\Delta) \text{PACE}_1(f(m))$$

Jerarhia de clase de complexitate

*Teorema: Dacă $S_1(m), S_2(m) \geq \log m$, $S_2(m)$ este scc și $\lim \frac{S_1(m)}{S_2(m)} = 0$. atunci $\Delta\text{SPACE}(S_2(m)) \setminus \Delta\text{SPACE}(S_1(m)) \neq \emptyset$

• funcție $f(m)$ este spațiu calculativă/comstruibilă complecț (SCC) dacă \exists o MT M a.t. pt orice m și x cu $|x|=m$ avem: $\text{SPACE}_M(x) = f(m)$

Relații:

- ① $(N)(\Delta)\text{TIME}(f(m)) \subseteq (N)(\Delta)\text{SPACE}(f(m))$
- ② Dacă $L \in (\Delta)\text{SPACE}(f(m))$ cu $f(m) \geq \log m$, atunci $\exists C_L$ a.t. $L \in (\Delta)\text{TIME}(e_L^{f(m)})$
- ③ Dacă $L \in (N)\text{TIME}(f(m))$, atunci $\exists C_L$ a.t. $L \in (\Delta)\text{TIME}(e_L^{f(m)})$
- ④ Savitch: Dacă $S(m) \geq \log m$ și $S(m)$ este scc, atunci $\text{NSPACE}(S(m)) \subseteq \Delta\text{SPACE}(S^2(m))$

~~În ce teorema + dem ???~~

Dem Teorema *

Idee: Construim o MT M a.t. $\text{SPACE}_M(m) \leq S_2(m)$ și limbiile său difere printr-o lungime cu un cuvant de orice limbaj acceptat de o MT M' cu spațiu $S_1(m)$.

Modelul de calcul al lui M :

- primește la intrare $w \in \{0,1\}^*$, $|w|=m$
- marchează o bandă auxiliară $S_2(m)$ celule și se va opri ori de câte ori va treb să folosească un spațiu mai mare. Dacă: $\text{SPACE}_M(m) \leq S_2(m)$
- identifică mașina Turing M' codificată cu Oy , unde $w = xoy$ și $x \in \{1\}^*$
- M simulează M' pe intrarea w . Presupunem că M se oprește mereu
- M neoprezepe w dacă simularea s-a încheiat cu acceptarea lui w de M' .

Fie M' o MT cu spațiu menținut de $S_1(m)$ și codificarea $\# \in \{0,1\}^*$. Putem presupune că M' are simboluri de bandă și o singură bandă auxiliară. \exists un cuv. $w' = 1^m \#$ cum să fie suficient de mare a.t. $[t]S_1(|w'|) \leq S_2(|w'|) \Rightarrow w' \in L(M)$ dacă $w' \notin L(M')$.

VII REDUCERI SI NP-COMPLETITUDINE

Reducere în timp polinomial:

Limbajul L' este redusibil în timp polinomial la limbajul L dacă există un determinat cu o bandă de reşire care se opreşte pe fiecare $(L' \leq_{tp} L)$ dacă și determinat cu o bandă de reşire care se opreşte pe fiecare intrare a.s. pt orice x la intrare produce un y pe bandă de reşire în timp polinomial $\exists M \in \mathbb{N}$ s.t. $x \in L' \Leftrightarrow y \in L$

Reducere în spațiu Logaritmic:

Limbajul L' este redusibil în spațiu logaritmic la limbajul L dacă există un determinat cu o bandă de reşire cu parcursere numărătore (pe care se poate obține serie) care se opreşte pe fiecare intrare a.s. pt orice x la intrare produce un y pe bandă de reşire folosind spațiu maxim logaritmic $x \in L' \Leftrightarrow y \in L$

$$L = \text{DSPACE}(\log n)$$

$$P = \bigcup_{i \geq 1} \text{NTIME}(m^i)$$

$$\text{NP} = \bigcup_{i \geq 1} \text{NTIME}(m^i)$$

$$\text{NSPACE} = \bigcup_{i \geq 1} \text{NSPACE}(m^i)$$

$$\text{PSPACE} = \bigcup_{i \geq 1} \text{PSPACE}(m^i)$$

¶

$$L \subseteq P \subseteq \text{NP} \subseteq \text{NSPACE} = \text{PSPACE}$$

Teorema 1

Fie $L' \leq_{tp} L$

$$a) \text{ Dacă } L \in P \Rightarrow L' \in P$$

$$b) \text{ Dacă } L \in \text{NP} \Rightarrow L' \in \text{NP}.$$

Dem: Dăm să arătăm $L \in P$ (pt $L \in \text{NP}$ se face analog)

Construim o M' determinată, cu $L(M) = L'$ și $\text{TIME}_{M'}(m) \leq p(x)$

$$\begin{aligned} \text{ip: } L' \leq_{tp} L &\Rightarrow \exists M \text{ determinat} \\ &\Rightarrow \text{Time}_{M'}(m) \leq p(m) \text{ și } x \in L \Leftrightarrow y \in L \end{aligned}$$

tip: $L \in P \Rightarrow \exists M \text{ determinista cu } L(M) = L$, $\text{TIME}_M \leq_{\text{P}} t$

$p_1(m)$ pt fiecare intrare de lungime m .

M' : - primește la intrare x
- simulează M pe intrarea x și ia timpul y și $\text{TIME}_{M'}(|x|) \leq p_2(|x|)$
- simulează M pe intrarea y și $\text{TIME}_M(|y|) \leq p_1(|y|)$
- M acceptă y

$\text{TIME}_{M'}(|x|) \leq p_2(|x|) + p_1(|y|) \leq p_2(|x|) + p_1(p_2(|x|)) \leq p(f(x))$
sau: $|y| \leq p_2(|x|)$

$\Rightarrow \text{TIME}_{M'}(|x|) \leq p_2(|x|) + p_1(p_2(|x|)) \Rightarrow L' \in P$

Teorema 2

Fie $L' \leq_{\text{P}} L$

a) Dacă $L \in P \Rightarrow L' \in P$

b) Dacă $L \in \text{SPACE}(\log^k m) \Rightarrow L' \in \text{SPACE}(\log^k m)$

c) Dacă $L \in \text{NSPACE}(\log^k m) \Rightarrow L' \in \text{NSPACE}(\log^k m)$

Teorema 3

Combinarea a două reduceri de același tip este o reducere de același tip.

Probleme dificile și complete

Fie C o clasă de limbi.

• Un limbaj L este C -dificil în raport cu o reducere \leq_{P} sau

• Un limbaj L' este C -oificil în raport cu o reducere \leq_{P} sau $L' \leq_{\text{P}} L$

• L este C -complet în raport cu o reducere \leq_{P} sau

• Un limbaj L este C -dificil în raport cu o reducere \leq_{P} sau

• L și $L \in C$

• L este C -NP-complet în raport cu o reducere \leq_{P} sau

• Un limbaj NP-complet dacă este NP-complet în raport cu o reducere \leq_{P} sau \leq_{NP}

~~afmprobif~~

mă NP-completă

Problema acoperirii unui graf (ACG)

Fie $G = (V, E)$ - un graf neorientat cu n vîrfuri și m muchii

$$V = \{x_1, x_2, \dots, x_m\}$$

$$E = \{e_1, e_2, \dots, e_m\}$$

(*) $e_i = \{x_{i1}, x_{i2}\}$ - muchie

O acoperire a lui G este o submultime a vîrfurilor a.i. $\{x_i\} \subseteq V$

O acoperire a lui G este o submultime a vîrfurilor a.i. $\{x_i\} \subseteq V$

$\subseteq E \rightarrow \{x_{ij}\} \cap X \neq \emptyset$ oricare 2 muchii au cel putin un vîrf

În X

Problema ACG (enunt):

Fie $K \geq 1 \in \mathbb{N}$. Se cere să se decidă dacă G are acoperire a lui G de cardinal maxim K .

a) ACG în XP.

$$V = \{x_1, x_2, \dots, x_m\} \text{ vîrfuri}$$

$$E = \{e_1, e_2, \dots, e_m\} \quad e_k = \{x_{ki}, x_{kj}\} \text{ muchie}$$

Grful este introdus pe baza MT. articol.

$$K(2) \# X^{1(2)} \# X^2 \quad x_1 x_{10} x_{11} \dots x_{\lceil \frac{m}{2} \rceil} x_{\lfloor \frac{m}{2} \rfloor} \{x_{\lceil \frac{m}{2} \rceil} x_{\lfloor \frac{m}{2} \rfloor}\} \dots$$

b) Arătă că problema xp-completă se poate reduce la ACG:

$$3-SAT \leq_{sl} ACG$$

$$\text{Fie } C = c_1 \wedge c_2 \wedge \dots \wedge c_m, \quad c_i = \{y_{i1}, \bar{y}_{i2}, \bar{y}_{i3}\}, \quad i=1, m$$

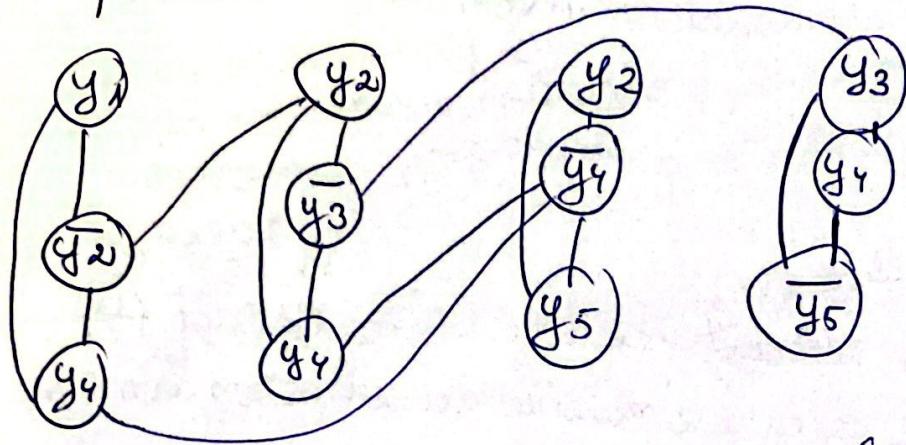
Construim grful: $G(V, E)$

$$V = \{y_{it} \mid i=1, m, t=1, 3\}$$

$$\{y_{is}, y_{it} \mid s, t \in \overline{1, 3}, \forall i \in \overline{1, m}, (\forall) i \leq s < t \leq 3$$

$$\{y_{is}, y_{it} \mid G \Rightarrow y_{is} = \bar{y}_{it}$$

Afirmatie: α = satisfacută dacă și numai dacă Γ are o acoperire de dimensiune $\leq m$.



DACĂ: Eliminând vîrfurile din acoperire ($2m$) rămân în moduri' cele în moduri' sunt distribuite cete unul în fiecare subgraf complet de 3 moduri'. Atunci în valoarea 1 literarului' corepunzător. Atribuirea facută este consistență și în urma ei $\Rightarrow \alpha$ satisfacută

NU MAI - DACĂ: α satisfacută \Rightarrow un literal din funcția clauză-are valoarea 1. Dând la o parte din graf modurile care corespund literarilor cu valoarea 1 rămân exact $2m$ moduri' ce reprez. acoperire

Spatiu logaritmic

Muchile grafului se construiesc întâi de-mă la nodurile din drepta și imediat ce terminăm o clauză nu ne mai întoarcem \Rightarrow pe hârtie de către vom merge pînă apă dreapta