

Curs 1. Calculabilitate și Complexitate.

Calculabilitate: programe standard

masini Turing - funcție Turing calc
funcții recursive

- echivalenta celor 3 modele.
- Gödelizare (codificare)
- a PS
- Problema oprire MT este undecidabilă

Complexitate: - modelele de OT

- clase de co-complexitate
- relații între aceste clase.
- NP - incompletitudine.

$$x = (x_1, \dots, x_n) \quad y_1, x_i \text{ secrete bineve.}$$

$$y = (y_1, \dots, y_n)$$

$$\exists k \in \mathbb{N} \text{ a.s. } \underbrace{x_1 x_2 \dots x_{k-1}}_{\text{deutschificare}} = y_1 y_2 \dots y_k$$

Demonstrări constructive

Af. $x = 0, x_1 x_2 x_3 \dots x_n \dots$

~~Există un număr $y \in \{0, 1\}^*$ așa că y apare de o infinitate de ori în scrierea lui x .~~

$$(p_n)_{n \geq 1} \in L \quad \begin{array}{ll} x_1 \neq 0 & p_1 = 1 \\ x_2 \neq 0 & p_2 = 1 \\ x_3 \neq 0 & p_3 = 1 \\ x_4 = 0 & p_4 = 0 \\ x = \dots & p_5 = 0 \end{array}$$

-2-

Există un k ce apare de o definiție de ori în sirul p_n .

Dacă Caz 1, scrierea decimală a lui x are o definiție de 0. $\rightarrow k = 0$
Caz 2, 0 nu apare de o definiție de ori.

$$x = 0, \dots \textcircled{0} x_{p+k} x_{p+2} \dots$$

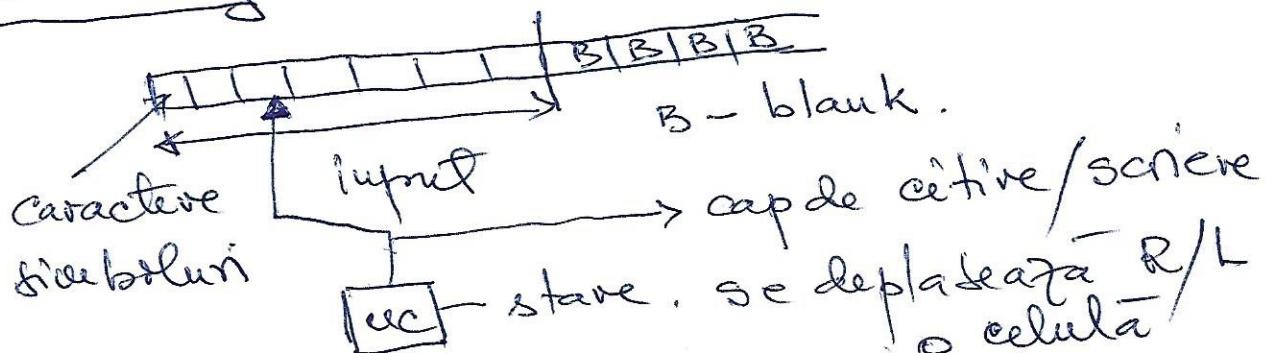
↑ multiplu de 0.

k k k

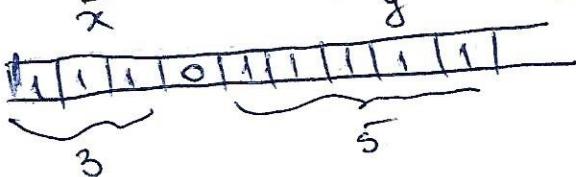
$$x = \sqrt{2} - 1.$$

Mașina Turing

→ infinită



Program: freccie de transiție.

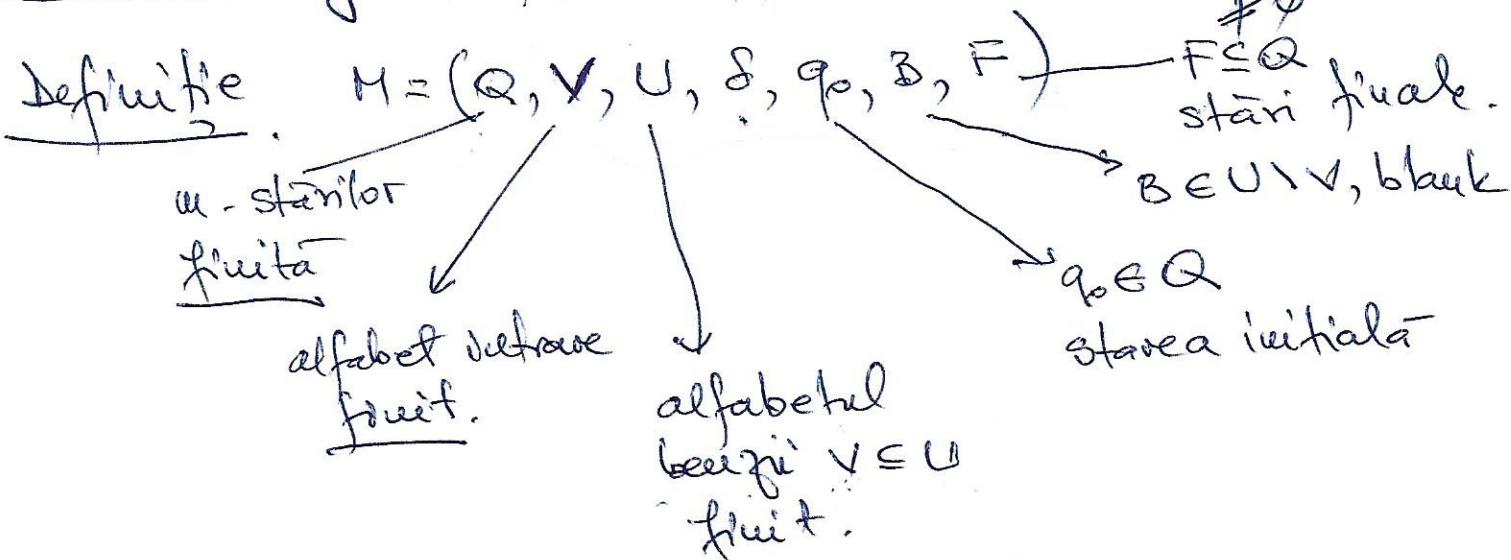


$\rightarrow 111100\dots0B\dots$

$$f(2,4) = 2 \log_2 4 = 4.$$

$$f(0,4) \rightarrow 01111$$

$$\text{Alte Turing} \quad 1350/8 + 20/8 = \frac{=3=}{22}$$



Masina este in starea "q" si cítete "a", poate trece in starea "s", scrie "b" peste "a" si se deplaseaza catre dreapta.

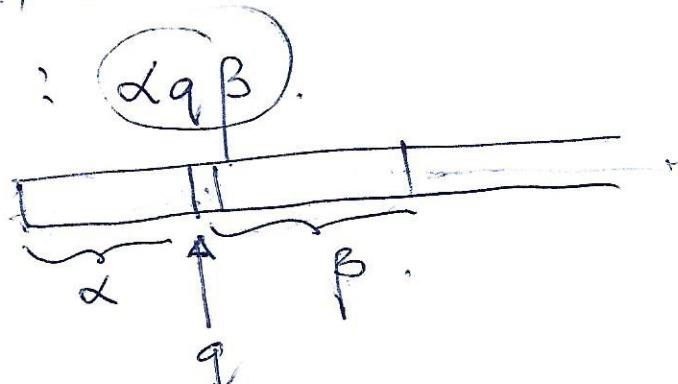
	a	b	c
s	(s, a)		
q		(q, b)	
q ₀			∅
t			

Restricții: $\exists (s, b, X) \in \delta(q, a) \quad | \Rightarrow b \neq B$, $a \neq B$.

Tranzitii: Coordonatelor: (x, y) .

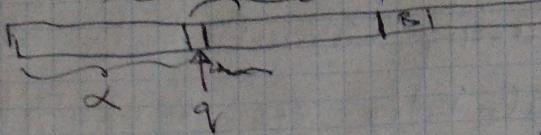
Ex. qP

Δq



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

$q\beta$ - configurație β



$dq\alpha \leftarrow dq\beta$ dacă $(s, b, R) \in \delta(q, \alpha)$

$dq \leftarrow dq\beta$ dacă $(s, b, R) \in \delta(q, \beta)$

$dq\beta \leftarrow \{ d\alpha \in \beta, \text{ dacă } (s, c, L) \in \delta(q, \alpha) \}$

$dq \leftarrow d\alpha\beta, \text{ dacă } (s, b, L) \in \delta(q, \beta)$

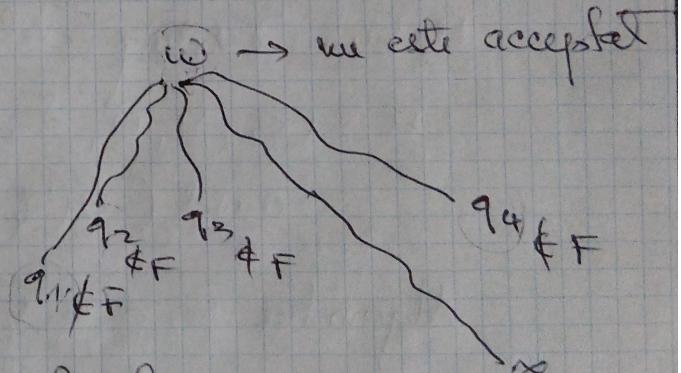
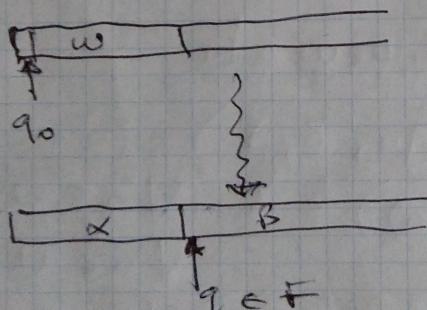
M - dispozitiv de acceptare

(acceptă limbajul)

$$L(M) = \{ w \in \Sigma^* \mid q_0 w \xrightarrow{*} q\beta, \quad \begin{cases} \alpha, \beta \in (\Sigma \setminus \{B\})^* \\ q \in F \end{cases} \}$$

echidondere reflexivă

și transiția a
relației $\xleftarrow{*}$

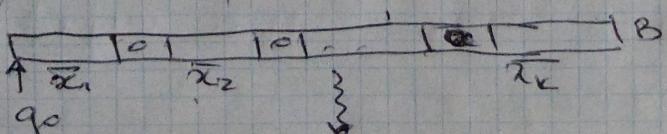


M ca dispozitiv de calcul
funcții numerale

$$f: N^k \rightarrow N$$

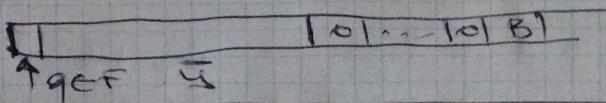
$f(x_1, \dots, x_k)$ este definită și $f(x_1, \dots, x_k) = y$

dacă



$$\bar{x} = \overbrace{1 \dots 1}^{x_1 \text{ ori}}$$

$x_1 \text{ ori}$



$f(x_1, \dots, x_n)$ nu este definită dacă
masina Turing nu se opreste pe intrarea
 (x_1, \dots, x_n)

f.s.u. Turing calculează dacă există
o rețea folosită ca dispozitor de calcul
care o calculează

Construiește o rețea:

- 1) Etapele bolotelor.
- 2) Fiecare etapă se face exceptă pe o reț.
- Routine rezvoacă
- definiție
(pseudocod)
- numărul de stări
în bulelori auxiliare
folosită este finit.

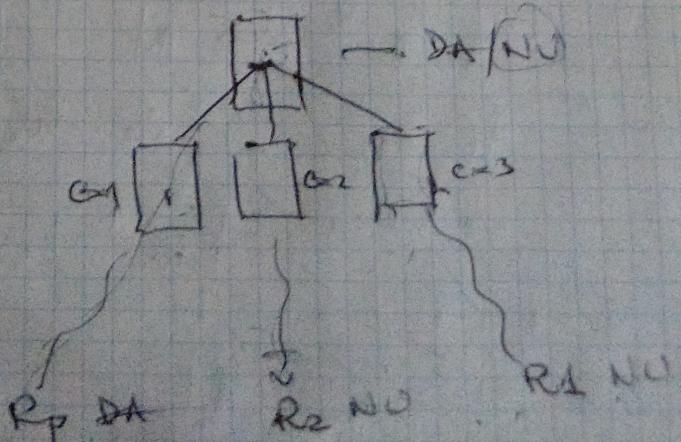
Masina Turing deterministă

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

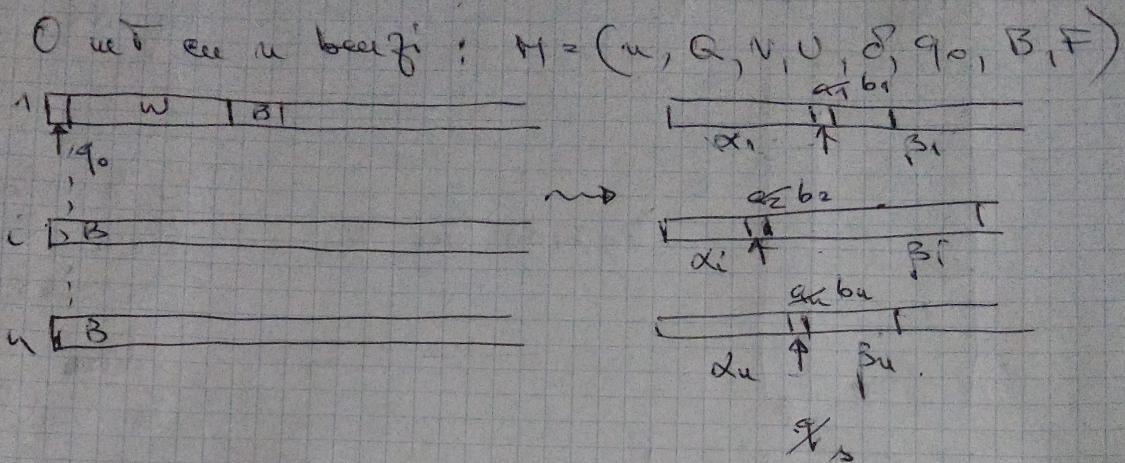
$$\text{card}(\delta(q, a)) \leq 1 \quad \forall q \in Q \\ \text{f.a.c } V$$

Algoritmic = o rețea deterministă care se
opreste pe fiecare intrare

MT deterministă \neq MT nedeterministă



- 5 -
 Scop: $MTD \xrightarrow{?} MTN$
 $\nwarrow \quad \swarrow$
 $w MTD$



$$\delta: (Q \times U)^n \rightarrow S_f(Q \times U^n \times \{L, R\}^n)$$

$$(s, b_1, \dots, b_u, x_1, \dots, x_u) \in \delta(q, a_1, \dots, a_u) \\ \in \{L, R\}^n$$

Teorema Pentru orice set nedeterministă M , există o setă deterministă, ce 3 benefici, și an. $L(M) = L(M')$.

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

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

M' :

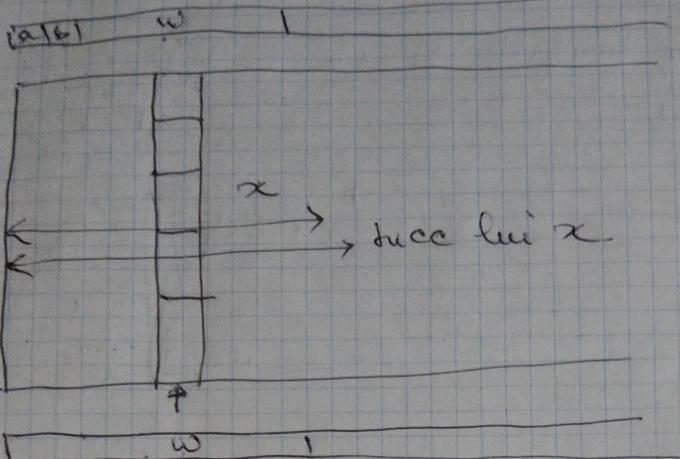
a	a	b	c	b	b
w					
B					

Etapa 1

Copiază w de la prima bandă la a treia bandă.

$$\delta(q_b, (a, \beta)) = (q_a, \underset{a}{\beta}, \underset{R, R \in}{\beta})$$

$$\begin{aligned}\delta(q_0, a, B) &= (q_0, a, a, R, R) \\ \delta(q_0, b, B) &= (q_0, b, b, R, R) \\ \delta(q_0, c, B) &= (q_0, c, c, R, R)\end{aligned}$$



Etapă 2.

Iguală bauda
2.

Greutatea
pe bauda 2
elementelor
succesor al
celorii curent
în vîrstă nu e
mai mare

E..

$$E = \{(q, a, s, b, x) \mid q, s \in Q, a \in U, b \in U - \{B\}, x \in \{L, R\} \}$$

$$\subseteq Q \times U \times Q \times U - \{B\} \times \{L, R\}. (s, b, x) \in \delta(q, a)$$

$\vee \{(B, \dots, B)\}$

Ticăre element al lui E va fi considerat

un stabil pe bauda 2.

$E = \{s_1, s_2, \dots, s_t\}$ ca urmare a
definiției (E, \leq)

$$\underline{\text{Ex}}. S = \{1, 2, 5, 10, 4\}$$

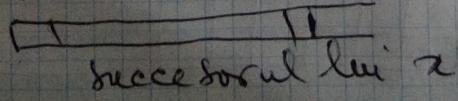
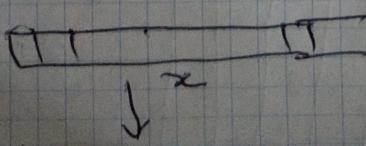
$$(S, \leq) = \{10, 5, 4, 1, 2\}$$

$$E = \{(\overset{\uparrow}{s_1}), (\overset{\uparrow}{s_2}), (\overset{\uparrow}{s_3}), \dots, (\overset{\uparrow}{s_n})\}$$

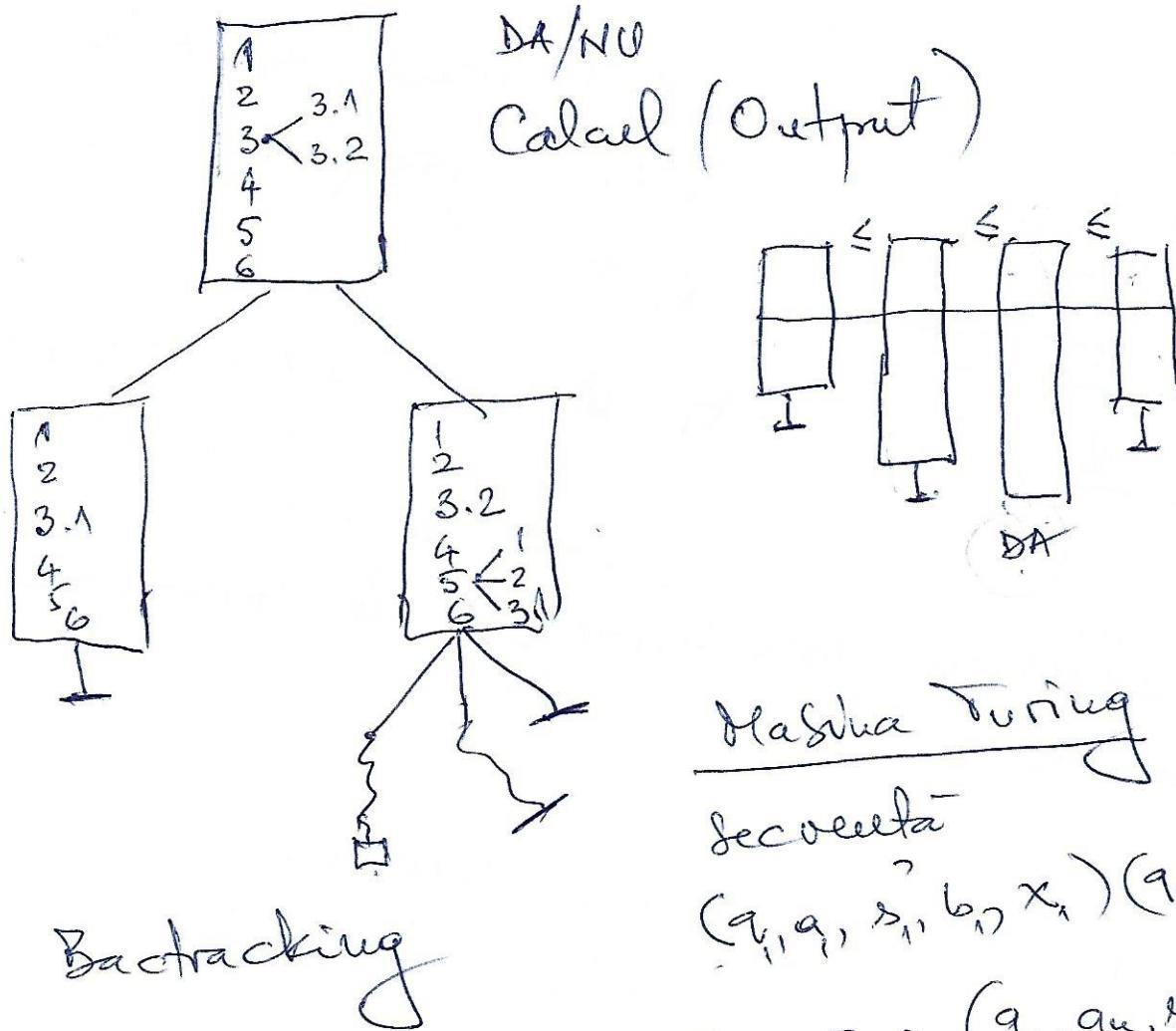
Subproblemă

$$21x04 \beta \neq$$

$$\begin{array}{c} 4 \\ 0 \\ \hline 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{array}$$



= CURS 3 =



Mashina Turing

secvență

$$(q_1, q_2, s_1, b_1, x_1) (q_2, q_2, s_2, b_2, x_2)$$

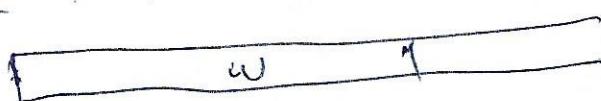
$$\dots \dots (q_n, q_n, s_n, b_n, x_n)$$

Teroarea Peatru orice set M se determină o set M' determinată de 3 beneficii există o set M' determinată cu 3 beneficii.

$$a.f. L(M) = L(M')$$

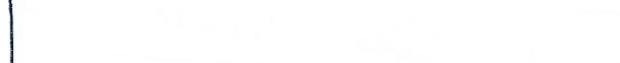
$$\text{Deu } f\text{ie } M = (Q, V, \Delta, \delta, q_0, B, F).$$

Construim M'



Etapă 1:

Copiem w de pe
bandă 1 pe bandă 3



w	1
s ₁	
a	
s ₂	
b	
x	

menajg $\in Q$

E^* = multimea decrivatorilor de cointelei
din E

(E, \leq)

$\Sigma = \{a, b\}$ $a \leq b$.

Pe ce posibile de situație.

~~a, aa, aaa,~~

a, b, aa, ab, ba, bb, aaa, - - -

aabb, $u = 1 \# 3$.

Etapa 3 Verifică dacă calculul de pe
baza 2 este valid pe intrarea afărată
pe baza 3.

Etapa 2.

Pe baza 2 se generează
succesiv în ordine
lexicografică în

E^*

$$E = \underbrace{\emptyset \times U \times Q \times U \times \{L, R\}}$$

(a, a, s, b, x)

decrivator de cointelei

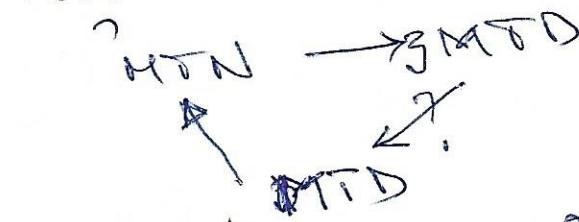
$abb \leq aaaa$

$aabba \geq aabba$

b ?

- E3.1. Calculul se blochează
- E3.2. Calcul se termină dacă starea
nu este finală (călăria
încasată)
- E3.3. Calculul se termină și
dacă este finală.
(Acceptă w)
- $L(M) = L(M')$
- ↓
M acceptă w .

E3.1 și E3.2.: Goto Etapa 1,
dacă M' este deterministă. QED.



Teorema 2 Pentru orice M cu un beneficiu H ,
există o rețea M' , cu o baza astfel.

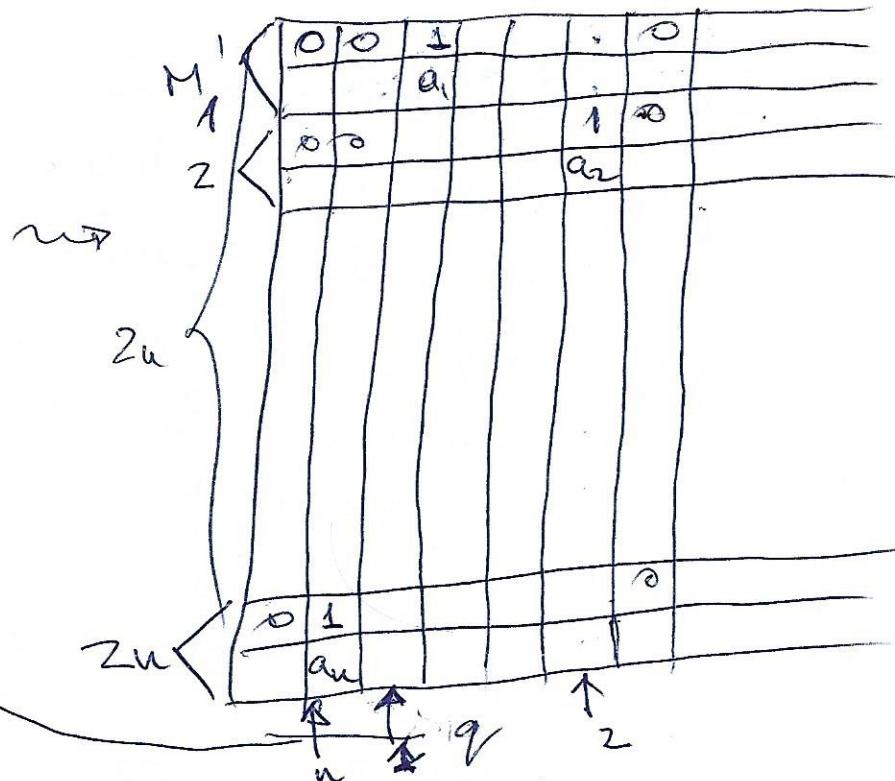
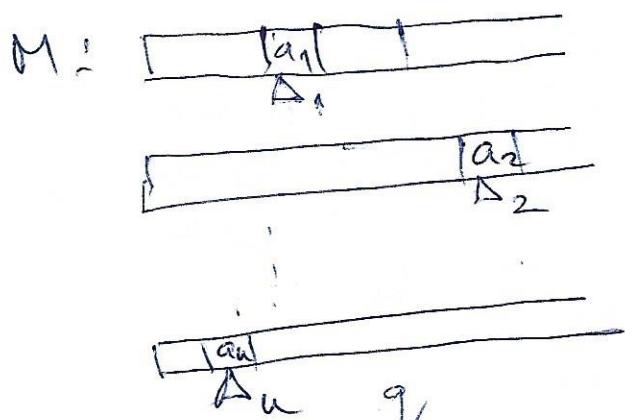
$L(M) = L(M')$.
Dacă M' este deterministă.

Dacă plus, M' este deterministă.

este deterministă.
 $\{ \}$

Dacă, $M = (\alpha, u, Q, V, U, \delta, q_0, B, F)$

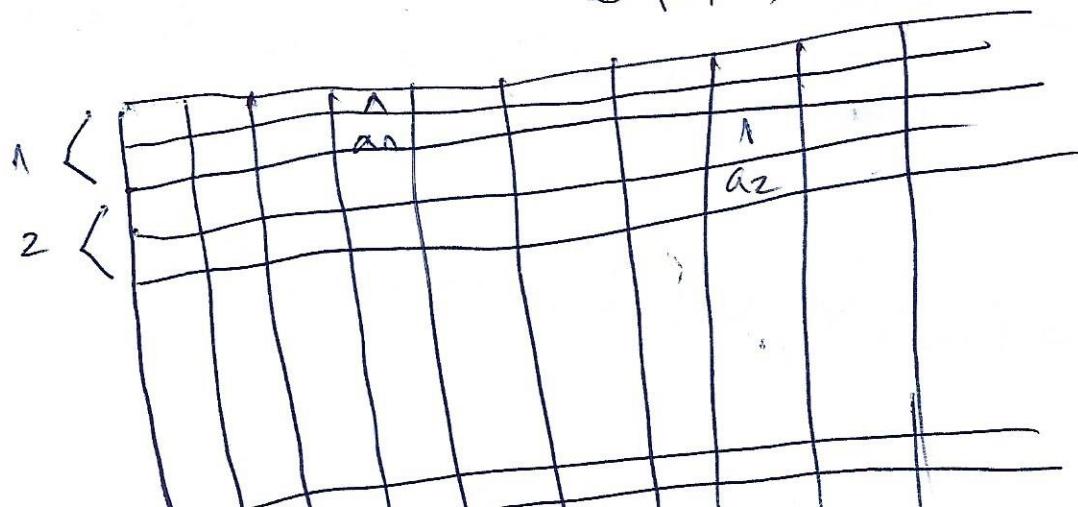
$$M' = (\alpha', v, V', \delta', q_0, B, F)$$



- Pe pistile de ordine superioare sunt elemente a_i, b_i .
- Pe pistile de ordine par este succesiunea lui U .
- Dacă o configurație C din M are o configurație

că correspunde în M' și $C_1 + C_2$
atunci $C_1 \rightarrow C_1$ și $C_2 \rightarrow C_2$

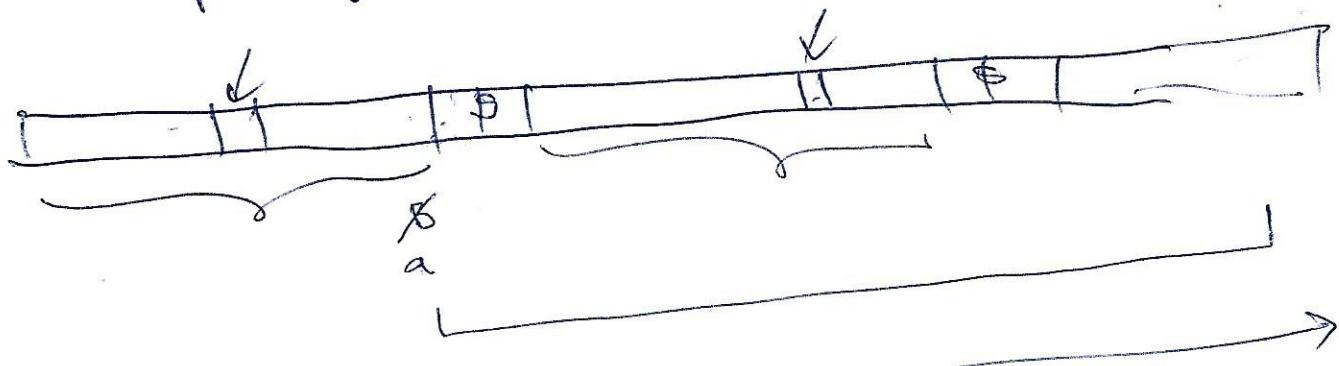
$$(a_1, b_1, \dots, a_n, b_n, x_1, \dots, x_n) = \delta(a_1, a_1, \dots, a_n)$$

$$\in \{h, R\}^n$$


Etape 1 M' scanază într-o reprezentare binară a
datele fizice ale sistemului a_1, a_2, \dots , an cofită
de M.

Etape 2 M' scanază banda de la
dreapta în stârge și actualizează
stările a_1, a_2, \dots an cu b_1, b_2, \dots, b_n .

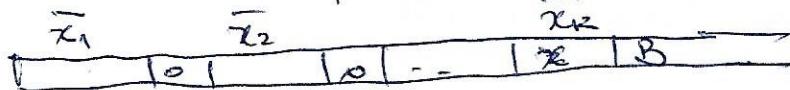
Etape 3 M' scanază banda de la stărg
și actualizează pozitia
în dreapta și actualizează pozitia
pe pistă împărțită.



= CURS 4 =

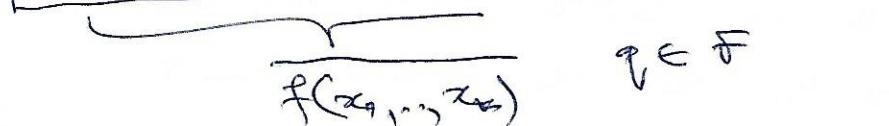
MT ca dispoziție de calcul al fructelor

$$f: \mathbb{N}^k \rightarrow \mathbb{N}$$



$$f(x_1, \dots, x_k)$$

$\Leftrightarrow f(x_1, \dots, x_k)$ este
de fructă



$$f(x_1, \dots, x_k) \quad q \in F$$

~~f(x_1, ..., x_k)~~ nu este de fructă (\Rightarrow nu se operează diferența x_1, \dots, x_k)

Discriminare: $f: \mathbb{N}^k \rightarrow \mathbb{N}$ restricțiv?

$$(i) f: \mathbb{N}^k \rightarrow \mathbb{N}^t \quad f(x_1, \dots, x_k) = (\underbrace{v_1, \dots, v_t}_{\in \mathbb{N}})$$

$$\downarrow \\ 2^{v_1} \cdot 3^{v_2} \cdots t^{v_t}$$

$$(ii) f: \mathbb{Z}^k \rightarrow \mathbb{N}$$

$$\mathbb{N}^{2^k} \quad (x_1, \dots, x_k) \in \mathbb{Z}^k$$

$$(1^{|x_1|}, 2^{|x_2|}, \dots, t^{|x_k|})$$

$$(iii) f: \mathbb{Q}^k \rightarrow \mathbb{N}$$

$$\downarrow \\ (\mathbb{Z}^k \times \mathbb{Z})^k \\ \mathbb{N}^3$$

$$(iv) f: \mathbb{R}^k \rightarrow \mathbb{N}$$

Ce "funcție" poate calcula cea MT ?
sunt Turing calculabile ?

Lucrările de programare abstractă

Sintaxă : variabile de intrare : x_1, x_2, x_3, \dots
variabile de iesire : y
variabile de lucru : z_1, z_2, z_3, \dots

Variabile de lucru și cea de iesire sunt
initializate cu 0. Orice variabilă menținută
o valoare materială

Etichete : E, A_1, A_2, \dots

Istrucțiuni : (1) $y \leftarrow v$, unde v este o
variabilă

Efect : Nul.

(2) $y \leftarrow y + 1$ Efect

(3) $y \leftarrow y - 1$ Efect : decrementar
dacă val lui $y > 0$
Nul, altfel.

Obs Orice instrucțiune poate fi
etichetată.

(4) IF $y \neq 0$ GOTO L

Efect : Dacă val lui y este 0
se trece la instrucțiunea următoare
Altfel, se face transfer la peinsă
instrucțiunea ce eticheta L.

Dacă nu există instrucțiune
ce eticheta L, programul se
oprește.

Program standard : o secvență fără încă de
de stoc și care :

Oprirea unui program standard :

- se termină instrucțiunile

- transfer la o instrucție care

~~nu~~ nu există. (vezi instrucțiunea)

IF $V \neq 0$ GOTO L

- transfer la eticheta E (Exit)

Care calculează $f(x)$? $f: N^k \rightarrow N$

$f(x_1, \dots, x_k)$

$x_1 \leftarrow x_1$ } nu apar în program -
 $x_2 \leftarrow x_2$ } sunt ori traligate
 \vdots
 $x_k \leftarrow x_k$

$f(x_1, \dots, x_k)$ este definită \Leftrightarrow Programul se
oprește și valoarea lui y este $f(x_1, \dots, x_k)$.
 $f(x_1, \dots, x_k)$ nu este definită \Leftrightarrow Programul
nu se oprește pe intrarea x_1, \dots, x_k .

Ex 1. $f(x) = x$. A₁: IF $x_1 \neq 0$ GOTO A₁

$z_1 \leftarrow z_1 + 1$

IF $z_1 \neq 0$ GOTO E

A₁: $x_1 \leftarrow x - 1$

$y \leftarrow y + 1$

IF $x \neq 0$ GOTO A₂.

Obs 1) GOTO L (Macro instrucție),
 2) $v \leftarrow v!$ ($\overbrace{\quad \quad \quad}^{n}$)

$A_2: \text{IF } X_1 \neq 0 \text{ GOTO } A_1$

= 4 =

GOTO ~~E~~ E

$A_1: X_1 \leftarrow X_1 - 1$

$Y \leftarrow Y + 1$

$Z_2 \leftarrow Z_2 + 1$

IF $X_1 \neq 0$ GOTO A_1

$A_3: X_1 \leftarrow X_1 + 1$

$Z_2 \leftarrow Z_2 - 1$

IF $Z_2 \neq 0$ GOTO A_3

① Funcție calculabilă ce PS. \Rightarrow Funcție Turing
calculabilă.

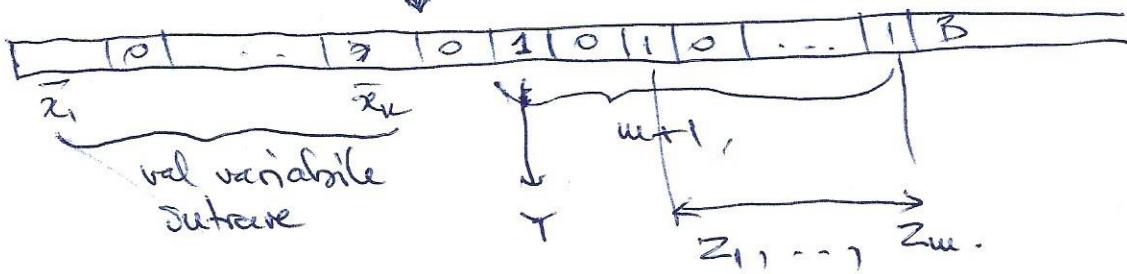
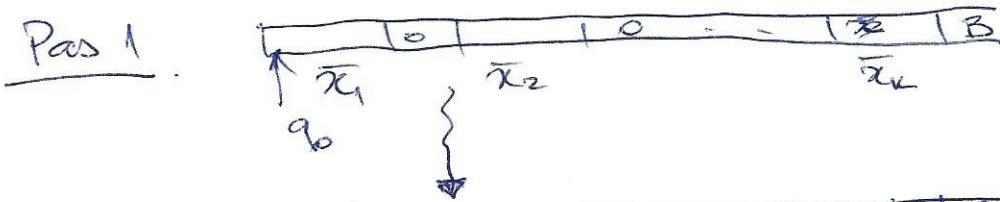
Dove. Fie $f: N^k \rightarrow N$ o funcție calculabilă ce PS.

Fie P programul standard care calculează f.

P \leftarrow n instrucțiuni : I_1, I_2, \dots, I_n .

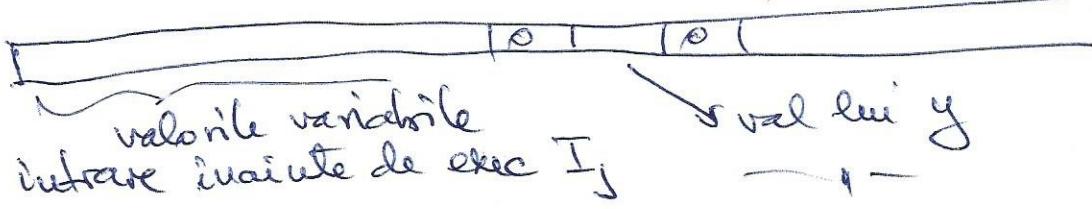
Z_1, \dots, Z_n variabile de lucru.

Cum calculează M o set :



Starea va fi $\langle I_1 \rangle$

Inductive pot spune că starea curentă a M este $\langle I_j \rangle$ și corespondentul său \langle val lui Z_i \rangle



val lui y

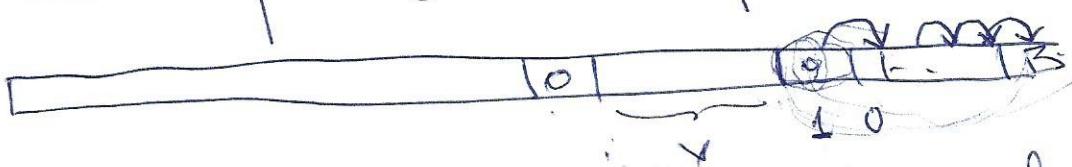
Pas 2. $I_j : V \neq V$.

uT nu efectuează nicio modificare a beneficiilor și schimbă starea $\langle I_{j+1} \rangle$.

$I_j : V \neq V+1$.

(i) uT aduce la formula de boole corectă factorii V

(ii) se plesează la dreapta toate posibilitățile de la dreapta formei corespunzătoare lui V .



(iii) ~~Repetă~~ plăresc formula corespunzătoare lui V care duce la 1.

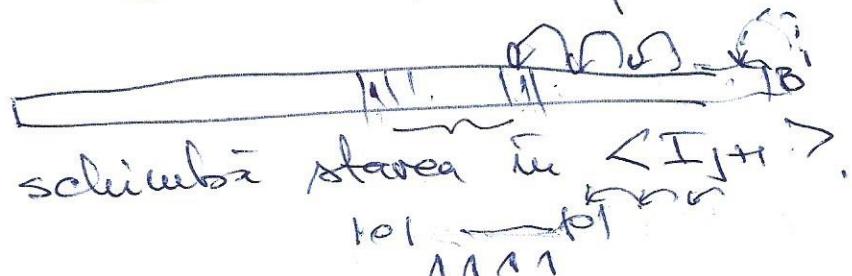
(iv) Schimbă starea în $\langle I_{j+1} \rangle$.

$I_j : V \neq V+1$.

(i) Ideea (i) anterior.

(ii) Dacă formula corespunzătoare lui V are doar un 1, schimbă starea în $\langle I_{j+1} \rangle$.

(iii) Altfel, sterge un 1 și corespundă formula lui V .



$I_j : IF V \neq 0 GOTO L$.

(i) Ideea (i) anterior.

(ii) Ideea (ii) anterior.

(iii) Altfel, lăsa boolea neschimbată -

starea curentă

$\langle I_p \rangle$ trebuie să este cel mai mic indice de instrucție a tipurilor având eticheta L.

dacă L este o etichetă,
diferită de E ~~și~~
casă există în P .

dacă $L = E$ sau nu
există starea devine
 $\langle I_{int} \rangle$ eticheta L ,

Multimea stării finale: $\{ \langle I_{int} \rangle \}$,
intermediare.

Pas final (3): Sterge tot ce pe boarde ce exceptă
lui y :

Iată boala: Există funcții $f: N \rightarrow N$
casă nu pot fi calculate cee PS ?

$$\text{card}(\{ f: N \rightarrow N \}) = \aleph_0$$

$$\text{card}(\{ \text{toate } PS \}) = \aleph_0$$

= CURS 5 =

Teorema Orice fracție (parțială) calculabilă
cu PS este '(parțial)' rezigă calculabilă.

Clasa funcțiilor (parțiali) recursive

Funcții elementare : succ : $N \rightarrow N$, $s(x) = x+1$

proiecții : $\pi_n^{(u)} : N^n \rightarrow N$, $\pi_n^{(u)}(x_1, \dots, x_n) = x_k$

constante : $c_n^{(u)} : N^n \rightarrow N$, $c_n^{(u)}(x_1, \dots, x_n) = k$.

(I) Sunt aceste funcții calculabile cu PS ?

succ : $Y \nrightarrow X$

$Y \nrightarrow Y+1$.

π_k : $Y \nrightarrow X_k$.

c_k : $\frac{Y \nrightarrow Y+1}{Y \nrightarrow Y+1}$ fără ori

Operări cu fracții

II Compozirea funcțională

$f \circ g : N^k \rightarrow N$ este definită prin comp. func.
a fracților $h : N^m \rightarrow N$ și $g_i : N^k \rightarrow N$, $i = 1, m$
dacă $f(x_1, \dots, x_k) = h(g_1(x_1, \dots, x_k), \dots, g_m(x_1, \dots, x_k))$

(II) Dacă funcțiile g_i și h sunt calculabile cu
PS este f calc. cu PS ?

$$z_1 \leftarrow g_1(x_1, \dots, x_k)$$

$$z_2 \leftarrow g_2(x_1, \dots, x_k)$$

$$\vdots \quad \vdots \quad \vdots$$

$$z_m \leftarrow g_m(x_1, \dots, x_k)$$

$$Y \nrightarrow h(z_1, \dots, z_m)$$

$$\text{Goto } f \\ A_j$$

$$A_j : z_e \leftarrow z_t$$

Recreceta primă

$f: N^{k+1} \rightarrow N$ este definită prin recreceta
primă dă $g: N^k \rightarrow N$ și $h: N^{k+2} \rightarrow N$

$$\text{daca } f(x_1, \dots, x_k, 0) = g(x_1, \dots, x_k)$$

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

(ii) Dacă funcția g nu își suport calc. cu PS, este
 f calc. cu PS? DA.

$$z_1 \leftarrow g(x_1, \dots, x_k) \quad // z_1 = f(x_1, \dots, x_k, 0)$$

IF $x_{k+1} \neq 0$, GOTO A1

$$Y \neq Z_1 \\ \text{GOTO E} \quad f(x_1, \dots, x_k, 0)$$

$$A_1 : z_2 \leftarrow h(x_1, \dots, x_k, \underbrace{z_3, z_1}_{\begin{smallmatrix} f \\ f \end{smallmatrix}}) \\ f(x_1, \dots, x_k, 1)$$

$$x_{k+1} \leftarrow x_{k+1} - 1.$$

$$z_3 \leftarrow z_3 + 1.$$

$$z_1 \leftarrow z_2$$

IF $x_{k+1} \neq 0$ GOTO A1.

$$Y \neq Z_1$$

Minimizare neîmpărțită

$f: N^k \rightarrow N$ este definită prin minimizare
neîmpărțită dă $g: N^{k+1} \rightarrow N$ dacă

$$f(x_1, \dots, x_k) = \min_{t \in N} [g(x_1, \dots, x_k, t) = 0]$$

(iv) Dacă g este calc. cu PS, este f calc. cu
PS?

= 3 =

A₂: $Z_1 \leftarrow g(x_1, \dots, x_k, Y)$

IF $Z_1 \neq 0$ GOTO A₁

GOTO E

A₁: $Y \leftarrow Y + 1$

GOTO A₂

Def. O funcție (partială) este recursivă dacă se poate obține din "funcții" elementare prin aplicarea consecutivă a op. de comp. funcțiivale, recurente și primitive și îninițială nu este nici o funcție.

Teorema Orice funcție (partială) recursivă este (partial) calculabilă cu PS.

Fie $\text{sum}(x_1, x_2) = x_1 + x_2$: Este recursivă ?

$$f(x_1, 0) = x_1 = \pi_1^{(0)}(x_1)$$

$$f(x_1, x_2+1) = f(x_1, x_2) + 1$$

$$= \text{succ}(\pi_3^{(0)}(x_1, x_2, f(x_1, x_2)))$$

$$\text{prod}(x_1, x_2) = x_1 \cdot x_2$$

$$\text{prod}(x_1, 0) = 0$$

$$\text{prod}(x_1, x_2+1) = \text{prod}(x_1, x_2) + x_1$$

$$x! . \quad 0! = 1$$

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

FR \rightarrow PS

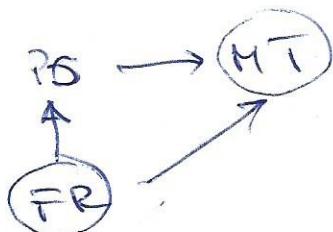
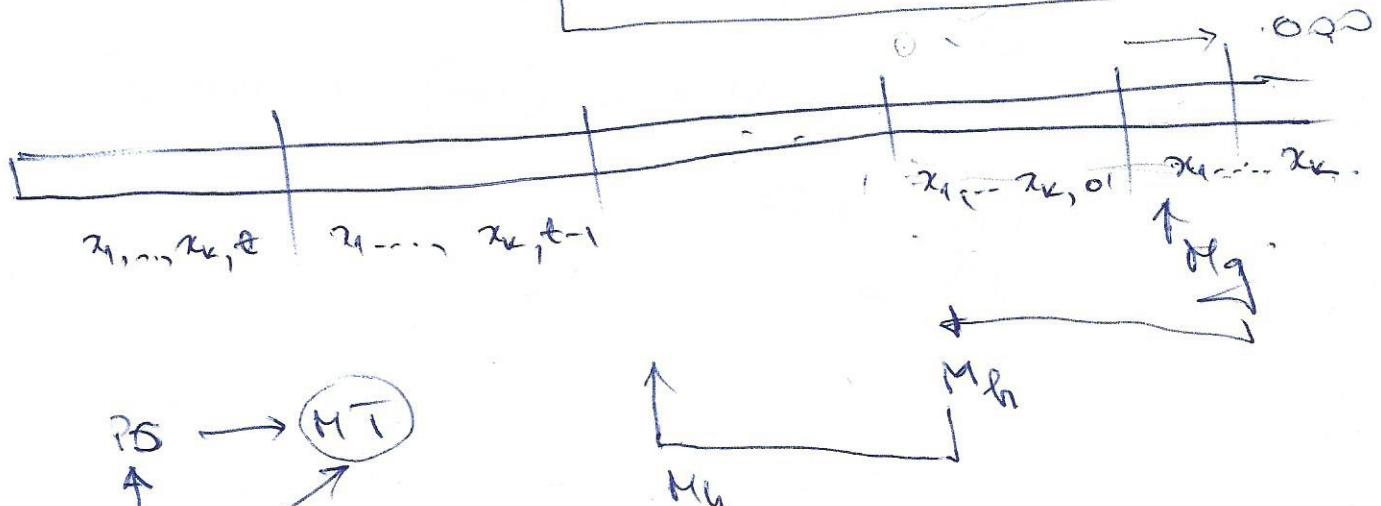
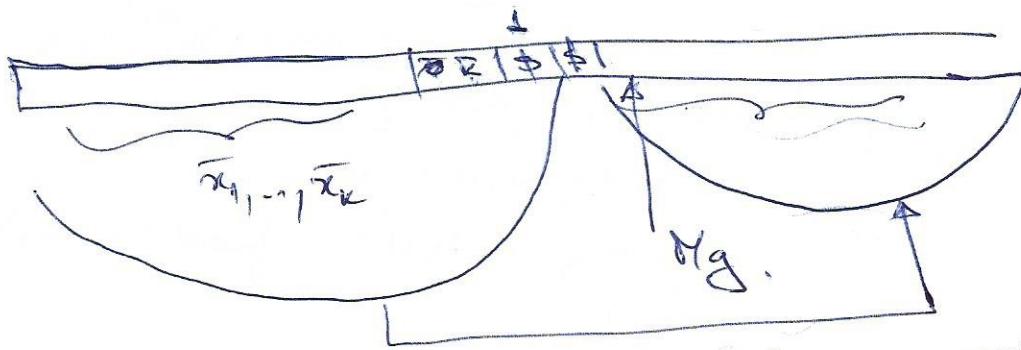
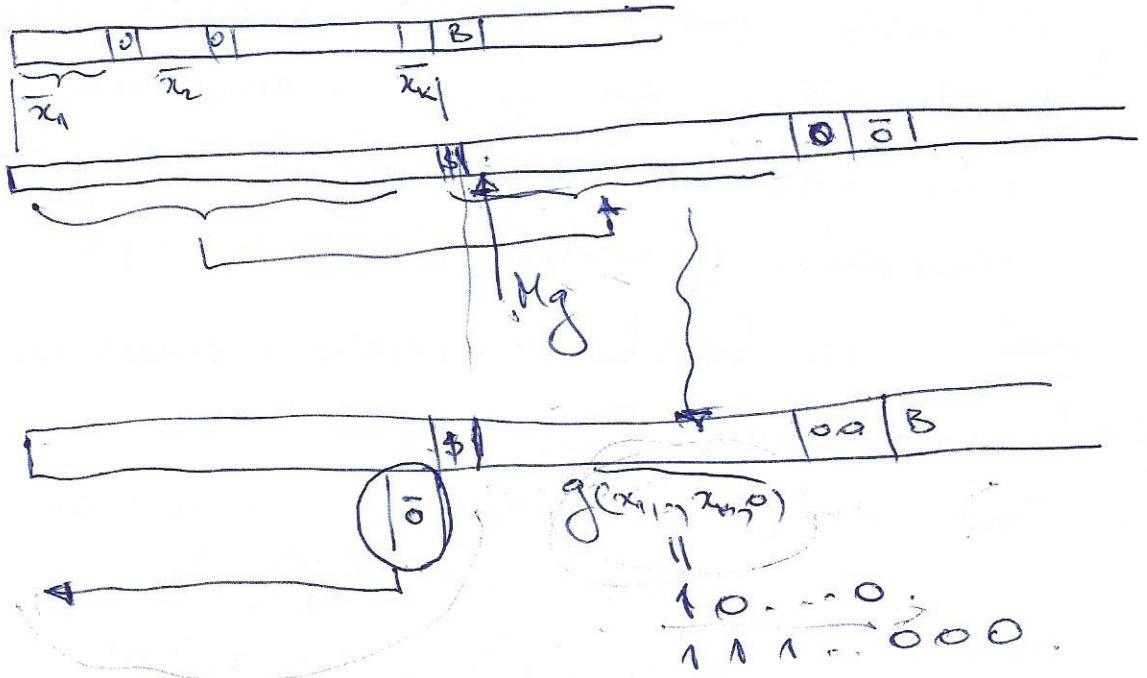


= 4 =

Codificarea PS

$P \mapsto n(P) (\#(P)) \in \mathbb{N}$

Dacă g este Turing calculabilă este cf, obțineți
mai multe informații relevante din g , Turing
calc? (0,2p)



Curs 6

November 11, 2021

Câteva funcții recursive

- ① Funcția sumă $f(x_1, x_2) = x_1 + x_2$.
- ② Funcția produs $f(x_1, x_2) = x_1 \cdot x_2$.
- ③ Funcția factorial $f(x) = x!$.
- ④ $f(x_1, x_2) = x_1^{x_2}$.

Câteva funcții recursive

- ① Funcția sumă $f(x_1, x_2) = x_1 + x_2$.
- ② Funcția produs $f(x_1, x_2) = x_1 \cdot x_2$.
- ③ Funcția factorial $f(x) = x!$.
- ④ $f(x_1, x_2) = x_1^{x_2}$.

$x_1^0 = 1$ (Atenție: convenim ca $0^0 = 1$)

$$x_1^{x_2+1} = x_1^{x_2} \cdot x_1.$$

- ⑤ Funcția predecesor $p(x) = \begin{cases} x - 1, & \text{dacă } x \neq 0 \\ 0, & \text{dacă } x = 0. \end{cases}$

Câteva funcții recursive

- ① Funcția sumă $f(x_1, x_2) = x_1 + x_2$.
- ② Funcția produs $f(x_1, x_2) = x_1 \cdot x_2$.
- ③ Funcția factorial $f(x) = x!$.
- ④ $f(x_1, x_2) = x_1^{x_2}$.

$$x_1^0 = 1 \text{ (Atenție: convenim ca } 0^0 = 1)$$

$$x_1^{x_2+1} = x_1^{x_2} \cdot x_1.$$

- ⑤ Funcția predecesor $p(x) = \begin{cases} x - 1, & \text{dacă } x \neq 0 \\ 0, & \text{dacă } x = 0. \end{cases}$

$$p(0) = 0$$

$$p(x + 1) = p.$$

- ⑥ $f(x_1, x_2) = \begin{cases} x_1 - \overset{\cdot}{x}_2, & x_1 \geq x_2, \\ 0, & x_1 < x_2. \end{cases}$

Câteva funcții recursive

- ① Funcția sumă $f(x_1, x_2) = x_1 + x_2$.
- ② Funcția produs $f(x_1, x_2) = x_1 \cdot x_2$.
- ③ Funcția factorial $f(x) = x!$.
- ④ $f(x_1, x_2) = x_1^{x_2}$.

$$x_1^0 = 1 \text{ (Atenție: convenim ca } 0^0 = 1)$$

$$x_1^{x_2+1} = x_1^{x_2} \cdot x_1.$$

- ⑤ Funcția predecesor $p(x) = \begin{cases} x - 1, & \text{dacă } x \neq 0 \\ 0, & \text{dacă } x = 0. \end{cases}$

$$p(0) = 0$$

$$p(x + 1) = p.$$

- ⑥ $f(x_1, x_2) = \begin{cases} x_1 - x_2, & x_1 \geq x_2, \\ 0, & x_1 < x_2. \end{cases}$

$$x_1 - 0 = 0$$

$$x_1 - (x_2 + 1) = p(x_1 - x_2).$$

Continuare funcții recursive

- 7 $f(x_1, x_2) = |x_1 - x_2|.$

Continuare funcții recursive

7 $f(x_1, x_2) = |x_1 - x_2|.$

$$|x_1 - x_2| = (x_1 - x_2) + (x_2 - x_1)$$

8 $\alpha(x) = \begin{cases} 1, & \text{dacă } x = 0 \\ 0, & \text{dacă } x \neq 0 \end{cases}$

Continuare funcții recursive

7 $f(x_1, x_2) = |x_1 - x_2|.$

$$|x_1 - x_2| = (x_1 - x_2) + (x_2 - x_1)$$

8 $\alpha(x) = \begin{cases} 1, & \text{dacă } x = 0 \\ 0, & \text{dacă } x \neq 0 \end{cases}$

$$\alpha(x) = 1 - x.$$

9 $f(x_1, x_2) \equiv (x_1 = x_2).$

Continuare funcții recursive

7 $f(x_1, x_2) = |x_1 - x_2|.$

$$|x_1 - x_2| = (x_1 - x_2) + (x_2 - x_1)$$

8 $\alpha(x) = \begin{cases} 1, & \text{dacă } x = 0 \\ 0, & \text{dacă } x \neq 0 \end{cases}$

$$\alpha(x) = 1 - x.$$

9 $f(x_1, x_2) \equiv (x_1 = x_2).$

$$(x_1 = x_2) \equiv \alpha(|x_1 - x_2|).$$

10 $f(x_1, x_2) \equiv (x_1 \leq x_2).$

Continuare funcții recursive

7 $f(x_1, x_2) = |x_1 - x_2|.$

$$|x_1 - x_2| = (x_1 - x_2) + (x_2 - x_1)$$

8 $\alpha(x) = \begin{cases} 1, & \text{dacă } x = 0 \\ 0, & \text{dacă } x \neq 0 \end{cases}$

$$\alpha(x) = 1 - x.$$

9 $f(x_1, x_2) \equiv (x_1 = x_2).$

$$(x_1 = x_2) \equiv \alpha(|x_1 - x_2|).$$

10 $f(x_1, x_2) \equiv (x_1 \leq x_2).$

$$(x_1 \leq x_2) \equiv \alpha(x_1 - x_2).$$

11 Dacă f și g sunt predicate recursive, atunci \bar{f} , $f \vee g$, $f \wedge g$ sunt recursive.

Continuare funcții recursive

7 $f(x_1, x_2) = |x_1 - x_2|.$

$$|x_1 - x_2| = (x_1 - x_2) + (x_2 - x_1)$$

8 $\alpha(x) = \begin{cases} 1, & \text{dacă } x = 0 \\ 0, & \text{dacă } x \neq 0 \end{cases}$

$$\alpha(x) = 1 - x.$$

9 $f(x_1, x_2) \equiv (x_1 = x_2).$

$$(x_1 = x_2) \equiv \alpha(|x_1 - x_2|).$$

10 $f(x_1, x_2) \equiv (x_1 \leq x_2).$

$$(x_1 \leq x_2) \equiv \alpha(x_1 - x_2).$$

11 Dacă f și g sunt predicate recursive, atunci \bar{f} , $f \vee g$, $f \wedge g$ sunt recursive.

$$\bar{f} \equiv \alpha(f), f \wedge g \equiv f \cdot g, f \vee g \equiv \overline{\bar{f} \wedge \bar{g}}$$

Continuare funcții recursive

- ⑫ Dacă f, g, P sunt funcții recursive de n variabile, atunci

$f(x_1, x_2, \dots, x_n) = \begin{cases} g(x_1, x_2, \dots, x_n), & \text{dacă } P(x_1, x_2, \dots, x_n), \\ h(x_1, x_2, \dots, x_n), & \text{altfel} \end{cases}$

este recursivă.

Continuare funcții recursive

- ⑫ Dacă f, g, P sunt funcții recursive de n variabile, atunci

$f(x_1, x_2, \dots, x_n) = \begin{cases} g(x_1, x_2, \dots, x_n), & \text{dacă } P(x_1, x_2, \dots, x_n), \\ h(x_1, x_2, \dots, x_n), & \text{altfel} \end{cases}$
este recursivă.

$$f \equiv g \cdot P + h \cdot \alpha(P).$$

- ⑬ Dacă $f(x_1, x_2, \dots, x_n, t)$ este recursivă, atunci funcțiile:

$$g(x_1, x_2, \dots, x_n, m) = \sum_{t=0}^m f(x_1, x_2, \dots, x_n, t),$$

$$h(x_1, x_2, \dots, x_n, m) = \prod_{t=0}^m f(x_1, x_2, \dots, x_n, t),$$

sunt recursive.

Continuare funcții recursive

- ⑫ Dacă f, g, P sunt funcții recursive de n variabile, atunci

$f(x_1, x_2, \dots, x_n) = \begin{cases} g(x_1, x_2, \dots, x_n), & \text{dacă } P(x_1, x_2, \dots, x_n), \\ h(x_1, x_2, \dots, x_n), & \text{altfel} \end{cases}$
este recursivă.

$$f \equiv g \cdot P + h \cdot \alpha(P).$$

- ⑬ Dacă $f(x_1, x_2, \dots, x_n, t)$ este recursivă, atunci funcțiile:

$$g(x_1, x_2, \dots, x_n, m) = \sum_{t=0}^m f(x_1, x_2, \dots, x_n, t),$$

$$h(x_1, x_2, \dots, x_n, m) = \prod_{t=0}^m f(x_1, x_2, \dots, x_n, t),$$

sunt recursive.

$$g(x_1, x_2, \dots, x_n, 0) = f(x_1, x_2, \dots, x_n, 0),$$

$$(x_1, x_2, \dots, x_n, m+1) = g(x_1, x_2, \dots, x_n, m) + f(x_1, x_2, \dots, x_n, m+1).$$

Continuare funcții recursive

- 14 Dacă predicatul $P(x_1, x_2, \dots, x_n, t)$ este recursiv, atunci:

$$g(x_1, x_2, \dots, x_n, m) = (\forall t)_{t \leq m} P(x_1, x_2, \dots, x_n, t),$$
$$h(x_1, x_2, \dots, x_n, m) = (\exists t)_{t \leq m} P(x_1, x_2, \dots, x_n, t),$$

sunt recursive.

Continuare funcții recursive

- ⑭ Dacă predicatul $P(x_1, x_2, \dots, x_n, t)$ este recursiv, atunci:

$$g(x_1, x_2, \dots, x_n, m) = (\forall t)_{t \leq m} P(x_1, x_2, \dots, x_n, t),$$

$$h(x_1, x_2, \dots, x_n, m) = (\exists t)_{t \leq m} P(x_1, x_2, \dots, x_n, t),$$

sunt recursive.

$$(\forall t)_{t \leq m} P(x_1, x_2, \dots, x_n, t) \equiv \left[\prod_{t=0}^m P(x_1, x_2, \dots, x_n, t) \right] = 1,$$

$$(\exists t)_{t \leq m} P(x_1, x_2, \dots, x_n, t) \equiv \left[\sum_{t=0}^m P(x_1, x_2, \dots, x_n, t) \right] \neq 0.$$

- ⑮ Funcția $f(x_1, x_2) \equiv x_2 | x_1$ este recursivă.

Continuare funcții recursive

- ⑭ Dacă predicatul $P(x_1, x_2, \dots, x_n, t)$ este recursiv, atunci:

$$g(x_1, x_2, \dots, x_n, m) = (\forall t)_{t \leq m} P(x_1, x_2, \dots, x_n, t),$$

$$h(x_1, x_2, \dots, x_n, m) = (\exists t)_{t \leq m} P(x_1, x_2, \dots, x_n, t),$$

sunt recursive.

$$(\forall t)_{t \leq m} P(x_1, x_2, \dots, x_n, t) \equiv \left[\prod_{t=0}^m P(x_1, x_2, \dots, x_n, t) \right] = 1,$$

$$(\exists t)_{t \leq m} P(x_1, x_2, \dots, x_n, t) \equiv \left[\sum_{t=0}^m P(x_1, x_2, \dots, x_n, t) \right] \neq 0.$$

- ⑮ Funcția $f(x_1, x_2) \equiv x_2|x_1$ este recursivă.

$$x_2|x_1 \equiv (\exists t)_{t \leq x_1} (x_2 \cdot t = x_1).$$

- ⑯ Predicatul $\text{Prime}(x) \equiv$ "este x număr prim?", este recursiv.

Continuare funcții recursive

- ⑭ Dacă predicatul $P(x_1, x_2, \dots, x_n, t)$ este recursiv, atunci:

$$g(x_1, x_2, \dots, x_n, m) = (\forall t)_{t \leq m} P(x_1, x_2, \dots, x_n, t),$$

$$h(x_1, x_2, \dots, x_n, m) = (\exists t)_{t \leq m} P(x_1, x_2, \dots, x_n, t),$$

sunt recursive.

$$(\forall t)_{t \leq m} P(x_1, x_2, \dots, x_n, t) \equiv \left[\prod_{t=0}^m P(x_1, x_2, \dots, x_n, t) \right] = 1,$$

$$(\exists t)_{t \leq m} P(x_1, x_2, \dots, x_n, t) \equiv \left[\sum_{t=0}^m P(x_1, x_2, \dots, x_n, t) \right] \neq 0.$$

- ⑮ Funcția $f(x_1, x_2) \equiv x_2|x_1$ este recursivă.

$$x_2|x_1 \equiv (\exists t)_{t \leq x_1} (x_2 \cdot t = x_1).$$

- ⑯ Predicatul $Prime(x) \equiv$ "este x număr prim?", este recursiv.

$$Prime(x) \equiv (x > 1) \wedge (\forall t)_{t \leq x} ((t = 1) \vee (t = x) \vee \overline{(t|x)}).$$

Continuare funcții recursive

- ⑯ Funcția "parte întreagă", $f(x_1, x_2) = [x_1/x_2]$ este recursivă.

Continuare funcții recursive

- ⑯ Funcția "parte întreagă", $f(x_1, x_2) = [x_1/x_2]$ este recursivă.
 $[x_1/x_2] = \min_t[(t + 1) \cdot x_2 > x_1]$.
- ⑰ Funcția $R(x_1, x_2)$, restul împărțirii întregi a lui x_1 la x_2 , este recursivă.

Continuare funcții recursive

- 17 Funcția "parte întreagă", $f(x_1, x_2) = [x_1/x_2]$ este recursivă.
 $[x_1/x_2] = \min_t[(t + 1) \cdot x_2 > x_1].$
- 18 Funcția $R(x_1, x_2)$, restul împărțirii întregi a lui x_1 la x_2 , este recursivă.
 $R(x_1, x_2) = x_1 - (x_2 \cdot [x_1/x_2]).$
- 19 Funcția p_n definită ca al n -lea număr prim, cu $p_0 = 0$, este recursivă.

Continuare funcții recursive

- ⑯ Funcția "parte întreagă", $f(x_1, x_2) = [x_1/x_2]$ este recursivă.
 $[x_1/x_2] = \min_t[(t + 1) \cdot x_2 > x_1].$
- ⑰ Funcția $R(x_1, x_2)$, restul împărțirii întregi a lui x_1 la x_2 , este recursivă.
 $R(x_1, x_2) = x_1 - (x_2 \cdot [x_1/x_2]).$
- ⑲ Funcția p_n definită ca al n -lea număr prim, cu $p_0 = 0$, este recursivă.
 $p_{n+1} = \min_t[Prime(t) \wedge (t > p_n)].$

⑳ Funțiile:

- ▶ $\langle x_1, x_2 \rangle = 2^{x_1}(2x_2 + 1) - 1$,
- ▶ $I(x) = z$, a.i. există t , $\langle z, t \rangle = x$,
- ▶ $r(x) = z$, a.i. există t , $\langle t, z \rangle = x$,

sunt funcții recursive.

- ㉑ Definim numărul lui Gödel atașat unei secvențe (a_1, a_2, \dots, a_n) ca fiind:

$$[a_1, a_2, \dots, a_n] = \prod p_i^{a_i}.$$

Exemplu: $[3, 0, 1, 8] = 2^3 \cdot 5 \cdot 7^8$.

Continuare funcții recursive

- 22 Funcția $(x)_i$ definită astfel: dacă $x = [a_1, a_2, \dots, a_n]$ atunci $(x)_i = a_i$. Această funcție este recursivă.

Continuare funcții recursive

- 22 Funcția $(x)_i$ definită astfel: dacă $x = [a_1, a_2, \dots, a_n]$ atunci $(x)_i = a_i$. Această funcție este recursivă.

$$(x)_0 = 0,$$
$$(x)_i = \underline{\min_t[p_i^{t+1}|x]}.$$

- 23 Funcția $Lt(x)$ definită astfel: dacă $x = [a_1, a_2, \dots, a_n]$ atunci $Lt(x) = n$. Această funcție este recursivă.

Continuare funcții recursive

- 22 Funcția $(x)_i$ definită astfel: dacă $x = [a_1, a_2, \dots, a_n]$ atunci $(x)_i = a_i$. Această funcție este recursivă.

$$(x)_0 = 0,$$
$$(x)_i = \underline{\min_t[p_i^{t+1}|x]}.$$

- 23 Funcția $Lt(x)$ definită astfel: dacă $x = [a_1, a_2, \dots, a_n]$ atunci $Lt(x) = n$. Această funcție este recursivă.

$$Lt(x) = \min_t[((x_t) \neq 0) \wedge (\forall j)_{j \leq x}((j \leq y) \vee ((x)_j = 0))].$$

Funcțiile Turing calculabile sunt recursive.

Demonstratie. Fie M o masina Turing determinista si fie $f(x_1, x_2, \dots, x_n)$ functia calculata de M . Fara a restrictiona generalitatea, vom presupune ca $n = 1$ deci masina calculeaza functia $f(x)$.

Fie $\{s_0 = 0, s_1 = 1, \dots\}$ o enumerare a tuturor simbolurilor ce pot aparea pe banda unei masini Turing. Identificam simbolul s_i prin indicele sau i . Numerotam celulele benzii cu $0, 1, \dots$ astfel incat fiecare celula se identifica prin numarul sau.

Fie $\{q_0, q_1, \dots\}$ o enumerare a tuturor stariilor ce pot aparea in definitia unei masini Turing. Identificam fiecare stare cu pozitia sa in aceasta enumerare.

La un pas oarecare, atasam urmatorul numar configuratiei curente: $\langle a, \langle b, c \rangle \rangle$, unde a este identifierul starii curente, b este pozitia capului de citire/scriere pe banda masinii, iar c este numarul Gödel asociat continutului benzii.

Functiile Turing calculabile sunt recursive.

Exemplu: starea curentă este q_2 , poziția capului pe banda este 3 iar conținutul benzii este: $s_1s_1s_0s_3B$, atunci numărul asociat acestei configurații este

Functiile Turing calculabile sunt recursive.

Exemplu: starea curentă este q_2 , poziția capului pe banda este 3 iar conținutul benzii este: $s_1s_1s_0s_3B$, atunci numărul asociat acestei configurații este

$$\begin{aligned} <2,<3,2 \cdot 3 \cdot 7^3>> = & <2,2^3 \cdot 2(2 \cdot 3 \cdot 7^3 + 1) - 1> = \\ & 2^2 \cdot 2(2^3 \cdot 2(2 \cdot 3 \cdot 7^3 + 1) - 1) - 1. \end{aligned}$$

Configurația initială are numărul:

Functiile Turing calculabile sunt recursive.

Exemplu: starea curentă este q_2 , poziția capului pe banda este 3 iar conținutul benzii este: $s_1s_1s_0s_3B$, atunci numărul asociat acestei configurații este

$$\begin{aligned} < 2, < 3, 2 \cdot 3 \cdot 7^3 >> = & < 2, 2^3 \cdot 2(2 \cdot 3 \cdot 7^3 + 1) - 1 > = \\ & 2^2 \cdot 2(2^3 \cdot 2(2 \cdot 3 \cdot 7^3 + 1) - 1) - 1. \end{aligned}$$

Configurația initială are numărul:

$$< 0, < 0, \prod_{i=1}^{x+1} p_i >> .$$

Functiile Turing calculabile sunt recursive.

Exemplu: starea curenta este q_2 , pozitia capului pe banda este 3 iar continutul benzii este: $s_1 s_1 s_0 s_3 B$, atunci numarul asociat acestei configuratii este

$$\begin{aligned} < 2, < 3, 2 \cdot 3 \cdot 7^3 >> = & < 2, 2^3 \cdot 2(2 \cdot 3 \cdot 7^3 + 1) - 1 > = \\ & 2^2 \cdot 2(2^3 \cdot 2(2 \cdot 3 \cdot 7^3 + 1) - 1) - 1. \end{aligned}$$

Configuratia initiala are numarul:

$$< 0, < 0, \prod_{i=1}^{x+1} p_i >> .$$

Definim functia $C_M(x, n) =$ numarul atasat configuratiei masinii la pasul n pe intrarea x . Evident, $C_M(x, 0) = < 0, < 0, \prod_{i=1}^{x+1} p_i >>$. Daca masina se opreste dupa n_0 pasi, atunci definim $C_M(x, n) = C_M(x, n_0)$, pentru orice $n \geq n_0$.

Functiile Turing calculabile sunt recursive.

Definim functiile auxiliare:

- $h_1(z) = \begin{cases} \text{numarul starii in care trece } M \text{ din configuratia cu numarul } z, \\ \text{daca acest numar codifica o configuratie valida in } M \\ 0, \text{ altfel} \end{cases}$
- $h_2(z) = \begin{cases} \text{numarul celulei de pe banda unde se pozitioneaza} \\ \text{capul I/O dupa configuratia cu numarul } z, \text{ daca acest numar} \\ \text{codifica o configuratie valida in } M \\ 0, \text{ altfel} \end{cases}$
- $h_3(z) = \begin{cases} \text{numarul configuratiei benzii dupa configuratia cu numarul } z, \\ \text{daca acest numar codifica o configuratie valida in } M \\ 0, \text{ altfel} \end{cases}$

Functiile Turing calculabile sunt recursive.

Definim functiile auxiliare:

- $h_1(z) =$

$$\begin{cases} \text{numarul starii in care trece } M \text{ din configuratia cu numarul } z, \\ \text{daca acest numar codifica o configuratie valida in } M \\ 0, \text{ altfel} \end{cases}$$

- $h_2(z) =$

$$\begin{cases} \text{numarul celulei de pe banda unde se pozitioneaza} \\ \text{capul I/O dupa configuratia cu numarul } z, \text{ daca acest numar} \\ \text{codifica o configuratie valida in } M \\ 0, \text{ altfel} \end{cases}$$

- $h_3(z) =$

$$\begin{cases} \text{numarul configuratiei benzii dupa configuratia cu numarul } z, \\ \text{daca acest numar codifica o configuratie valida in } M \\ 0, \text{ altfel} \end{cases}$$

$$C_M(x, n+1) = < h_1(C_M(x, n)), < h_2(C_M(x, n)), h_3(C_M(x, n))) >> .$$

Funcțiile Turing calculabile sunt recursive.

Fie a numarul unei stari si b numarul unui simbol. Definim

- $g_1(a, b) = \begin{cases} \text{numarul starii in care trece } M \text{ din starea } a, \\ \text{citind } b, \text{ daca } a \text{ si } b \text{ sunt valide} \\ 0, \text{ altfel} \end{cases}$
- $g_2(a, b) = \begin{cases} \text{numarul simbolului scris pe banda de } M \text{ din starea } a, \\ \text{citind } b, \text{ daca } a \text{ si } b \text{ sunt valide} \\ 0, \text{ altfel} \end{cases}$
- $g_3(a, b) = \begin{cases} 0 \text{ sau } 2, \text{ daca } M \text{ se deplaseaza la stanga sau dreapta} \\ \text{din starea } a \text{ citind } b, \text{ daca } a \text{ si } b \text{ sunt valide} \\ 0, \text{ altfel} \end{cases}$

Funcțiile Turing calculabile sunt recursive.

Fie a numarul unei stari si b numarul unui simbol. Definim

- $g_1(a, b) = \begin{cases} \text{numarul starii in care trece } M \text{ din starea } a, \\ \text{citind } b, \text{ daca } a \text{ si } b \text{ sunt valide} \\ 0, \text{ altfel} \end{cases}$
- $g_2(a, b) = \begin{cases} \text{numarul simbolului scris pe banda de } M \text{ din starea } a, \\ \text{citind } b, \text{ daca } a \text{ si } b \text{ sunt valide} \\ 0, \text{ altfel} \end{cases}$
- $g_3(a, b) = \begin{cases} 0 \text{ sau } 2, \text{ daca } M \text{ se deplaseaza la stanga sau dreapta} \\ \text{din starea } a \text{ citind } b, \text{ daca } a \text{ si } b \text{ sunt valide} \\ 0, \text{ altfel} \end{cases}$

$$h_1(z) = g_1(I(z), (r(r(z)))_{I(r(z))})$$

Funcțiile Turing calculabile sunt recursive.

Fie a numarul unei stari si b numarul unui simbol. Definim

- $g_1(a, b) = \begin{cases} \text{numarul starii in care trece } M \text{ din starea } a, \\ \text{citind } b, \text{ daca } a \text{ si } b \text{ sunt valide} \\ 0, \text{ altfel} \end{cases}$
- $g_2(a, b) = \begin{cases} \text{numarul simbolului scris pe banda de } M \text{ din starea } a, \\ \text{citind } b, \text{ daca } a \text{ si } b \text{ sunt valide} \\ 0, \text{ altfel} \end{cases}$
- $g_3(a, b) = \begin{cases} 0 \text{ sau } 2, \text{ daca } M \text{ se deplaseaza la stanga sau dreapta} \\ \text{din starea } a \text{ citind } b, \text{ daca } a \text{ si } b \text{ sunt valide} \\ 0, \text{ altfel} \end{cases}$

$$h_1(z) = g_1(I(z), (r(r(z)))_{I(r(z))})$$
$$h_2(z) = I(r(z)) + g_3(I(z), (r(r(z)))_{I(r(z))}) - 1$$

Funcțiile Turing calculabile sunt recursive.

Fie a numarul unei stari si b numarul unui simbol. Definim

- $g_1(a, b) = \begin{cases} \text{numarul starii in care trece } M \text{ din starea } a, \\ \text{citind } b, \text{ daca } a \text{ si } b \text{ sunt valide} \\ 0, \text{ altfel} \end{cases}$
- $g_2(a, b) = \begin{cases} \text{numarul simbolului scris pe banda de } M \text{ din starea } a, \\ \text{citind } b, \text{ daca } a \text{ si } b \text{ sunt valide} \\ 0, \text{ altfel} \end{cases}$
- $g_3(a, b) = \begin{cases} 0 \text{ sau } 2, \text{ daca } M \text{ se deplaseaza la stanga sau dreapta} \\ \text{din starea } a \text{ citind } b, \text{ daca } a \text{ si } b \text{ sunt valide} \\ 0, \text{ altfel} \end{cases}$

$$h_1(z) = g_1(I(z), (r(r(z)))_{I(r(z))})$$

$$h_2(z) = I(r(z)) + g_3(I(z), (r(r(z)))_{I(r(z))}) - 1$$

$$h_3(z) = r(r(z)) / p_{I(r(z))}^{(r(r(z)))_{I(r(z))}} * p_{I(r(z))}^{g_2(I(z), (r(r(z)))_{I(r(z))})}$$

Functiile Turing calculabile sunt recursive.

Afirmatii:

- Functiile g_1, g_2, g_3 sunt recursive.
- Functiile h_1, h_2, h_3 sunt recursive.
- Functia C_M este recursiva.

Definim $nr_M(x)$ ca fiind numarul de pasi pe care ii face M pentru a calcula $f(x)$, daca $f(x)$ este definita.

Functiile Turing calculabile sunt recursive.

Afirmatii:

- Functiile g_1, g_2, g_3 sunt recursive.
- Functiile h_1, h_2, h_3 sunt recursive.
- Functia C_M este recursiva.

Definim $nr_M(x)$ ca fiind numarul de pasi pe care ii face M pentru a calcula $f(x)$, daca $f(x)$ este definita.

$$nr_M(x) = \min_t [C_M(x, t) = C_M(x, t + 1)].$$

Deci nr_M este recursiva.

Atunci f se poate scrie:

Funcțiile Turing calculabile sunt recursive.

Afirmatii:

- Funcțiile g_1, g_2, g_3 sunt recursive.
- Funcțiile h_1, h_2, h_3 sunt recursive.
- Functia C_M este recursiva.

Definim $nr_M(x)$ ca fiind numarul de pasi pe care ii face M pentru a calcula $f(x)$, daca $f(x)$ este definita.

$$nr_M(x) = \min_t [C_M(x, t) = C_M(x, t + 1)].$$

Deci nr_M este recursiva.

Atunci f se poate scrie:

$$f(x) = Lt(r(r(C_M(x, nr_M(x))))) - 1.$$

In concluzie, f este recursiva.

Curs 7

November 24, 2021

Teza Church-Turing.

Orice functie efectiv/intutiv calculabila este recursiva/Turing calculabila.

Modele echivalente:

- ① Functii recursive
- ② Programe standard
- ③ λ -calcul (Alonzo Church)
- ④ Algoritm Markov (Andrey Markov, Jr.)
- ⑤ Automate coada
- ⑥ Sistem tag (Emil Post)
- ⑦ Masini cu registri (Marvin Minsky)
- ⑧ Automatul celular (Stanislaw Ulam si John von Neumann)
- ⑨ Sisteme de rescriere (include gramatica generativa Chomsky)
- ⑩ Etc.

Se poate depasi bariera Turing? **Hipercalcul**
(Hypercomputation):accelerare, invatare inductiva, etc.

Codificarea programelor standard.

Etichete: $\{E, A_1, A_2, \dots\}$; codificam o eticheta $\#(L)$ cu pozitia ei (numaratoarea incepe cu 1).

Variabile: $\{Y, X_1, Z_1, X_2, Z_2, \dots\}$; codificam o variabila $\#(V)$ cu pozitia sa (numaratoarea incepe cu 1).

Codificarea unei instructiuni I va fi $\#(I) = < a, < b, c >>$, unde

- daca I nu este etichetata, atunci $a = 0$ altfel $a = \#(L)$, unde L este eticheta ei.
- $c = \#(V) - 1$, unde V este variabila care apare in I .
- $b = 0, 1, 2$ daca $I = V \leftarrow V, V \leftarrow V + 1, V \leftarrow V - 1$.
- $b = \#(L) + 2$ daca I este $IF\ V \neq 0\ GOTO\ L$.

Exemplu: I este $A_1 : Z_2 \leftarrow Z_2 + 1$

$$\#(I) = < 2, < 1, 4 >> = < 2, 2 \cdot 9 - 1 > = < 2, 17 > = 4(2 \cdot 17 + 1) - 1 = 139.$$

Codificarea unui program P cu instructiunile I_1, I_2, \dots, I_k este

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

Codificarea programelor.

Exemplu: Ce numar are programul:

$$E : X_1 \leftarrow X_1 + 1$$

IF $X_1 \neq 0$ GOTO E

$$\#(I_1) = <1, <1, 1>> = <1, 5> = 21$$

$$\#(I_2) = <0, <3, 1>> = <0, 23> = 46.$$

$$\#(P) = 2^{21} \cdot 3^4 6 - 1.$$

Exemplu: Care este programul cu numarul 14999.

Codificarea programelor.

Exemplu: Ce numar are programul:

$$E : X_1 \leftarrow X_1 + 1$$

IF $X_1 \neq 0$ GOTO E

$$\#(I_1) = <1, <1, 1>> = <1, 5> = 21$$

$$\#(I_2) = <0, <3, 1>> = <0, 23> = 46.$$

$$\#(P) = 2^{21} \cdot 3^4 6 - 1.$$

Exemplu: Care este programul cu numarul 14999.

$$14999 + 1 = 15000 = 2^3 \cdot 3 \cdot 5^4$$

Deci programul are 3 instructiuni $I_1, I_{2,3}$ cu $\#(I_1) = 3$, $\#(I_2) = 0$,

$$\#(I_3) = 4.$$

- $< a, < b, c >> = 3 \Rightarrow 2^a(2 < b, c > + 1) - 1 = 3 \Rightarrow a = 2, 2 < b, c > + 1 = 1 \Rightarrow a = 2, b = 0, c = 0$

Deci $I_1 \equiv A_2 : Y \leftarrow Y$.

- $< a, < b, c >> = 0 \rightarrow a = b = c = 0$. Deci $I_2 \equiv Y \leftarrow Y$.

- $< a, < b, c >> = 4 \rightarrow a = 0, b = 0, c = 1$. Deci $I_3 \equiv X_1 \leftarrow X_1$.

Problema opririi.

Definim predicatul:

$\text{HALT}(x, t) \equiv$ programul codificat cu numarul t se opreste pe intrarea x .

Teorema. *Predicatul HALT nu este calculabil cu programe standard.*

Dem. Pp HALT calculabil si construim programul P :

$$A : \text{IF } \text{HALT}(X, X) \text{ GOTO } A$$

Functia calculata de P este

Problema opririi.

Definim predicatul:

$\text{HALT}(x, t) \equiv$ programul codificat cu numarul t se opreste pe intrarea x .

Teorema. *Predicatul HALT nu este calculabil cu programe standard.*

Dem. Pp HALT calculabil si construim programul P :

$A : \text{IF } \text{HALT}(X, X) \text{ GOTO } A$

Functia calculata de P este $\psi_P(x) = \begin{cases} 0, & \text{daca } \overline{\text{HALT}}(x, x) \\ \text{nedefinit, altfel} & \end{cases}$ Fie

$\#(P) = t$, deci $\text{HALT}(x, t) \equiv \overline{\text{HALT}}(x, x)$. Luam $x = t$ si obtinem $\text{HALT}(t, t) \equiv \overline{\text{HALT}}(t, t)$, contradictie.

Programul universal.

Pentru fiecare $n \geq 1$, definim functia universala de n variabile:

$$\Phi^{(n)}(x_1, x_2, \dots, x_n, t) = \varphi_P^{(n)}(x_1, x_2, \dots, x_n), \#(P) = t.$$

TEOREMA. Pentru orice n , functia universala de n variabile este (partial) calculabila cu programe standard.

DEM. Vom construi un program U_n care va fi programul universal pentru calculul functiei universale de n variabile.

Vom nota cu:

K : numarul instructiunii curente din programul cu numarul t ce urmeaza a fi simulate

S : "starea" curenta a programului cu numarul t (va memora valorile tuturor variabilelor la un moment dat).

Programul universal.

Pentru fiecare $n \geq 1$, definim functia universala de n variabile:

$$\Phi^{(n)}(x_1, x_2, \dots, x_n, t) = \varphi_P^{(n)}(x_1, x_2, \dots, x_n), \#(P) = t.$$

TEOREMA. Pentru orice n , functia universala de n variabile este (partial) calculabila cu programe standard.

DEM. Vom construi un program U_n care va fi programul universal pentru calculul functiei universale de n variabile.

Vom nota cu:

K : numarul instructiunii curente din programul cu numarul t ce urmeaza a fi simulate

S : "starea" curenta a programului cu numarul t (va memora valorile tuturor variabilelor la un moment dat).

$$Z \leftarrow T + 1(X_{n+1} + 1) // Z = [\#(I_1, I_2, \dots, I_m)] //$$

$$K \leftarrow 1$$

$$S \leftarrow \prod_{i=1}^n p_{2i}^{X_i}$$

Programul universal.

$C : IF (K > Lt(Z)) \vee (K = 0) GOTO F$

Programul universal.

$C : IF (K > Lt(Z)) \vee (K = 0) GOTO F$

$U \leftarrow r((Z)_K) // codul < b, c > al instructiunii K //$

Programul universal.

$C : IF (K > Lt(Z)) \vee (K = 0) GOTO F$

$U \leftarrow r((Z)_K) // codul < b, c > al instructiunii K //$

$V \leftarrow p_r(U) + 1 // codul variabilei de apare in instructiunea K //$

Programul universal.

$C : IF (K > Lt(Z)) \vee (K = 0) GOTO F$

$U \leftarrow r((Z)_K) // codul < b, c > al instructiunii K //$

$V \leftarrow p_{r(U)} + 1 // codul variabilei de apare in instructiunea K //$

$IF I(U) = 0 GOTO N // nu facem nimic //$

$IF I(U) = 1 GOTO I // incrementam variabila //$

$IF \overline{p_V | S} GOTO N // nu facem nimic //$

$IF I(U) = 2 GOTO D // decrementam variabila //$

Programul universal.

$C : IF (K > Lt(Z)) \vee (K = 0) GOTO F$
 $U \leftarrow r((Z)_K) // codul < b, c > al instructiunii K //$
 $V \leftarrow p_{r(U)} + 1 // codul variabilei de apare in instructiunea K //$
 $IF I(U) = 0 GOTO N // nu facem nimic //$
 $IF I(U) = 1 GOTO I // incrementam variabila //$
 $IF \overline{p_V | S} GOTO N // nu facem nimic //$
 $IF I(U) = 2 GOTO D // decrementam variabila //$
 $K \leftarrow \begin{cases} min_i[I((Z)_i) + 2 = I(U)], & \text{daca un astfel de } i \text{ exista,} \\ 0, & \text{altfel} \end{cases}$
 $GOTO C$

Programul universal.

$C : IF (K > Lt(Z)) \vee (K = 0) GOTO F$

$U \leftarrow r((Z)_K) // codul < b, c > al instructiunii K //$

$V \leftarrow p_{r(U)} + 1 // codul variabilei de apare in instructiunea K //$

$IF I(U) = 0 GOTO N // nu facem nimic //$

$IF I(U) = 1 GOTO I // incrementam variabila //$

$IF p_V | S GOTO N // nu facem nimic //$

$IF I(U) = 2 GOTO D // decrementam variabila //$

$K \leftarrow \begin{cases} min_i[I((Z)_i) + 2 = I(U)], & \text{daca un astfel de } i \text{ exista,} \\ 0, & \text{altfel} \end{cases}$

$GOTO C$

$I : S \leftarrow S * p_V$

$GOTO N$

Programul universal.

C : IF ($K > Lt(Z)$) \vee ($K = 0$) GOTO F

$U \leftarrow r((Z)_K)$ //codul $< b, c >$ al instructiunii K//

$V \leftarrow p_{r(U)} + 1$ //codul variabilei de apare in instructiunea K//

IF $I(U) = 0$ GOTO N //nu facem nimic//

IF $I(U) = 1$ GOTO I //incrementam variabila//

IF $p_V | S$ GOTO N //nu facem nimic//

IF $I(U) = 2$ GOTO D //decrementam variabila//

$K \leftarrow \begin{cases} min_i[I((Z)_i) + 2 = I(U)], \text{ daca un astfel de } i \text{ exista,} \\ 0, \text{ altfel} \end{cases}$

GOTO C

*I : $S \leftarrow S * p_V$*

GOTO N

D : $S \leftarrow S / p_V$

Programul universal.

C : IF ($K > Lt(Z)$) \vee ($K = 0$) GOTO F

U $\leftarrow r((Z)_K)$ //codul $< b, c >$ al instructiunii K//

V $\leftarrow p_{r(U)} + 1$ //codul variabilei de apare in instructiunea K//

IF $I(U) = 0$ GOTO N //nu facem nimic//

IF $I(U) = 1$ GOTO I //incrementam variabila//

IF $p_V | S$ GOTO N //nu facem nimic//

IF $I(U) = 2$ GOTO D //decrementam variabila//

K $\leftarrow \begin{cases} min_i[I((Z)_i) + 2 = I(U)], & \text{daca un astfel de } i \text{ exista,} \\ 0, & \text{altfel} \end{cases}$

GOTO C

*I : $S \leftarrow S * p_V$*

GOTO N

D : $S \leftarrow S / p_V$

N : $K \leftarrow K + 1$

GOTO C

Programul universal.

$C : IF (K > Lt(Z)) \vee (K = 0) GOTO F$

$U \leftarrow r((Z)_K) // codul < b, c > al instructiunii K //$

$V \leftarrow p_{r(U)} + 1 // codul variabilei de apare in instructiunea K //$

$IF I(U) = 0 GOTO N // nu facem nimic //$

$IF I(U) = 1 GOTO I // incrementam variabila //$

$IF p_V | S GOTO N // nu facem nimic //$

$IF I(U) = 2 GOTO D // decrementam variabila //$

$K \leftarrow \begin{cases} min_i[I((Z)_i) + 2 = I(U)], & \text{daca un astfel de } i \text{ exista,} \\ 0, & \text{altfel} \end{cases}$

$GOTO C$

$I : S \leftarrow S * p_V$

$GOTO N$

$D : S \leftarrow S / p_V$

$N : K \leftarrow K + 1$

$GOTO C$

$F : Y \leftarrow (S)_1$

Codificarea masinilor Turing.

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

- $\{q_0, q_1, q_2, \dots\}$ enumerare a tuturor statelor. Codificam q_i fie cu $qbin_i$ sau $q0^i$
- $\{s_0, s_1, s_2, \dots\}$ enumerare a tuturor simbolurilor. Codificam s_i fie cu $sbin_i$ sau $s0^i$. Convenim $s_0 = B$.
- Codificarea masinii Turing M va fi

$$\langle M \rangle = w_1 \$ w_2 \$ w_3 \$ w_4 \$ q_0 \$ s_0 \$ w_5,$$

unde

- ▶ w_1 este un string format din toate codificările statelor din Q ,
- ▶ w_2 este un string format din toate codificările simbolurilor din V ,
- ▶ w_3 este un string format din toate codificările simbolurilor din $U \setminus V$,
- ▶ w_4 este un string ce codifica funcția δ ; este o secvență de substringuri de forma $(qbin_{i_1} sbin_{j_1} qbin_{k_1} sbin_{l_1} L/R)$,
- ▶ w_5 este un string format din toate codificările statelor din F .

Deci $\langle M \rangle \in \{(,), 0, 1, q, s, L, R, \$\}^*$. Se poate chiar $\langle M \rangle \in \{0, 1\}^*$?

Codificarea binara a masinilor Turing.

- q_i se codifica cu 0^i ,
- s_i se codifica cu 0^i ,
- 1 se va folosi ca
- L si R se codifica cu 11 si, respectiv, 111,
- \$ se codifica cu 1111,
- (si) se codifica cu 11111 si, respectiv, 111111.

Orice cuvant se poate codifica printr-un sir binar care incepe cu 1.

Limbajul universal:

$$L_u = \{<< M >, < w >> |$$

sirul codificat cu $< w >$ este acceptat de masina codificata cu $< M >$.

Exista o masina Turing care accepta L_u , numita masina Turing universala.
Afirmatia rezulta si din constructia programului universal.

Codificarea masinilor Turing.

Specific, construim o masina Turing U cu 4 benzi astfel:

- Prima banda contine $\langle\langle M \rangle\rangle$.
- A doua banda contine $\langle w \rangle$.
- A treia banda memoreaza starea curenta a masinii M .
- A patra banda este auxiliara. Se foloseste pentru calcule de decodificare si codificare.
- U cauta o tranzitie codificata in $\langle M \rangle$ in care starea este cea pastrata pe banda 3 iar simbolul citit este cel de pe banda 2. Atentie: codificarile lor! Pentru aceasta identificare foloseste banda 4.
- Schimba banda 3 pentru a memora noua stare.
- Schimba banda 2 pentru a schimba simbolul citit cu cel scris.
- Muta capul de citire/scriere pe banda 2.
- U se opreste daca starea memorata pe banda 3 este finala in M .

Masini Turing cu $(15, 2), (9, 3), (6, 4), (5, 5), (4, 6), (3, 9), (2, 18)$
stari/simboluri. Depinde numarul de instructiuni.

Multimi/limbaje recursive si recursiv enumerabile.

Algoritm: masina Turing determinista care se opreste pe fiecare intrare.

Definitie

- Un limbaj L (multime A) este recursiv enumerabil (enumerabila) daca exista o masina Turing M a.i. $L(M) = L$ (χ_A este Turing calculabila.)
- Un limbaj L (multime A) este recursiv (recursiva) daca exista o masina Turing M , care se opreste pe fiecare intrare, a.i. $L(M) = L$ (χ_A este Turing calculabila.)

Echivalenta problema de decizie-limbaj: Fie P un predicat de n variabile.

Construim limbajul $L_P = \{< P(x_1, x_2, \dots, x_n) > \mid P(x_1, x_2, \dots, x_n) = 1\}$.

Multimi/limbaje recursive si recursiv enumerabile.

Algoritm: masina Turing determinista care se opreste pe fiecare intrare.

Definitie

- Un limbaj L (multime A) este recursiv enumerabil (enumerabila) daca exista o masina Turing M a.i. $L(M) = L$ (χ_A este Turing calculabila.)
- Un limbaj L (multime A) este recursiv (recursiva) daca exista o masina Turing M , care se opreste pe fiecare intrare, a.i. $L(M) = L$ (χ_A este Turing calculabila.)

Echivalenta problema de decizie-limbaj: Fie P un predicat de n variabile.

Construim limbajul $L_P = \{< P(x_1, x_2, \dots, x_n) > | P(x_1, x_2, \dots, x_n) = 1\}$.

P este decidabila ddaca L_P este recursiv.

TEOREMA.

1. Limbajul universal este recursiv enumerabil.
2. Limbajul $L_h = \{<< M >, < w >> | M \text{ se opreste pe intrarea } w\}$ este recursiv enumerabil.

Sunt ele recursive?

Multimi/limbaje recursive si recursiv enumerabile.

Fie M_1, M_2, \dots , o enumerare a masinilor Turing a.i. $\langle M_1 \rangle, \langle M_2 \rangle, \dots$ este o enumerare a codificarilor lor in ordine lexicografica. Analog, fie w_1, w_2, \dots o enumerare a cuvintelor binare.

TEOREMA. Limbajul (diagonal) $L_d = \{w_i \mid w_i \notin L(M_i)\}$ nu este recursiv enumerabil.

Multimi/limbaje recursive si recursiv enumerabile.

Fie M_1, M_2, \dots , o enumerare a masinilor Turing a.i. $\langle M_1 \rangle, \langle M_2 \rangle, \dots$ este o enumerare a codificarilor lor in ordine lexicografica. Analog, fie w_1, w_2, \dots o enumerare a cuvintelor binare.

TEOREMA. Limbajul (diagonal) $L_d = \{w_i \mid w_i \notin L(M_i)\}$ nu este recursiv enumerabil.

DEM. Pp. $L_d = L(M_j)$. Atunci $w_j \in L_d = L(M_j)$ ddaca $w_j \notin L(M_j) = L_d$.

TEOREMA. Limbajele L_u si L_h nu sunt recursive.

DEM. 1. Pp L_u este recursiv, $L_u = L(M)$ a.i. M se opreste pe fiecare intrare. Construim M' care lucreaza astfel pe un cuvant binar w primit la intrare:

- Determina j a.i. $w = w_j$.
- Din j determina M^* a.i. $M^* = M_j$.
- Simuleaza M pe intrarea $\langle\langle M_j \rangle, \langle w_j \rangle\rangle$. Daca M accepta, atunci M' respinge si vice versa.
- $L(M') = L_d$, contradictie.

Demonstratie pentru 2?

Multimi/limbaje recursive si recursiv enumerabile.

TEOREMA.

1. X este recursiva daca $\complement X$ sunt recursiv enumerabile.
2. Daca X si Y sunt recursive/recursiv enumerabile, atunci $X \cup Y$, $X \cap Y$ sunt recursive/recursiv enumerable.

DEM. Simplu exercitiu.

O proprietate a unei clase de limbaje \mathcal{C} este o submultime S a lui \mathcal{C} . Proprietatea se numeste *triviala* daca $S = \mathcal{C}$ sau $S = \emptyset$. Altfel, se numeste *netriviala*. Alegem \mathcal{C} ca fiind clasa limbajelor recursiv enumerabile RE . Pentru o proprietate S a clasei RE definim $L_S = \{\langle M \rangle \mid L(M) \in S\}$.

Teorema Rice.

TEOREMA. Orice proprietate netriviala pe RE este nedecidabila.

DEM. Fie S o proprietate netriviala pe RE , si $\emptyset \notin S$. Aratam ca L_S nu este recursiv. Pp ca $L_S = L(M_S)$, M_S se opreste pe fiecare intrare.

Fie $L \in S$ a.i. $L = L(M_L)$. Alegem si fixam $\langle\langle M \rangle, w \rangle$ si construim M' a.i. $L(M') = \begin{cases} L, & \text{daca } w \in L(M), \\ \emptyset, & \text{altfel} \end{cases}$ Cum calculeaza M' :

- Initial M' ignora intrarea sa x si simuleaza M pe w .
- Daca M accepta, atunci simuleaza M_L pe x .
- Accepta ddaca M_L accepta.

Construim M_u astfel:

- Pe intrarea $\langle\langle M \rangle, \langle w \rangle\rangle$ determina $\langle M' \rangle$.
- Simuleaza M_S pe $\langle M' \rangle$.
- Accepta ddaca M_S accepta. (M_S se opreste pe fiecare intrare.)

Observatie: M_u se opreste pe fiecare intrare!!!

$L \in S \Leftrightarrow \langle\langle M' \rangle \in L(M_S) \implies \langle\langle M \rangle, \langle w \rangle \rangle \in L(M_u)$, contradictie.

Cazul $\emptyset \notin S$?

Teorema Rice.

TEOREMA. Orice proprietate netriviala pe RE este nedecidabila.

DEM. Fie S o proprietate netriviala pe RE , si $\emptyset \notin S$. Aratam ca L_S nu este recursiv. Pp ca $L_S = L(M_S)$, M_S se opreste pe fiecare intrare.

Fie $L \in S$ a.i. $L = L(M_L)$. Alegem si fixam $\langle\langle M \rangle, w \rangle$ si construim M' a.i. $L(M') = \begin{cases} L, & \text{daca } w \in L(M), \\ \emptyset, & \text{altfel} \end{cases}$ Cum calculeaza M' :

- Initial M' ignora intrarea sa x si simuleaza M pe w .
- Daca M accepta, atunci simuleaza M_L pe x .
- Accepta ddaca M_L accepta.

Construim M_u astfel:

- Pe intrarea $\langle\langle M \rangle, \langle w \rangle\rangle$ determina $\langle M' \rangle$.
- Simuleaza M_S pe $\langle M' \rangle$.
- Accepta ddaca M_S accepta. (M_S se opreste pe fiecare intrare.)

Observatie: M_u se opreste pe fiecare intrare!!!

$L \in S \Leftrightarrow \langle\langle M' \rangle\rangle \in L(M_S) \implies \langle\langle M \rangle, \langle w \rangle\rangle \in L(M_u)$, contradictie.

Cazul $\emptyset \notin S$? $\emptyset \notin CS$

Problema Corespondentei lui Post.

Problema $PCP(x, y)$

Input: un alphabet V cu cel putin doua simboluri, $n \geq 2$ si doua liste Post

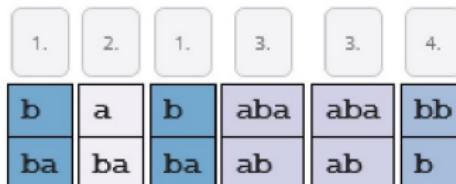
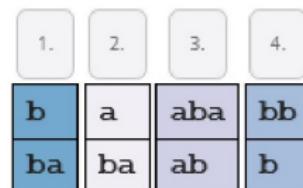
$x = (x_1, x_2, \dots, x_n)$,

$y = (y_1, y_2, \dots, y_n)$,

cu $x_i, y_i \in V^*, 1 \leq i \leq n$.

Output: Exista $k \geq 1$ si $1 \leq i_1, i_2, \dots, i_k \leq n$ a.i.

$x_{i_1}x_{i_2} \dots x_{i_k} = y_{i_1}y_{i_2} \dots y_{i_k}$?



Problema Corespondentei lui Post.

PCP modificata (MPCP): Input: un alphabet V cu cel putin doua simboluri, $n \geq 2$ si doua liste Post x, y .

Output: Exista $k \geq 1$ si $1 \leq i_1, i_2, \dots, i_k \leq n$ a.i.

$$x_1 x_{i_1} x_{i_2} \dots x_{i_k} = y_1 y_{i_1} y_{i_2} \dots y_{i_k} ?$$

Propozitie. Daca PCP este decidabila, atunci MPCP este decidabila.

Dem. Fie $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$, si $\beta = (\beta_1, \beta_2, \dots, \beta_n)$, doua liste Post peste alphabetul V . Vom folosi algoritmul pentru PCP pentru a arata ca $MPCP(\alpha, \beta)$ este decidabila.

Construim doua liste Post x, y peste alphabetul $V \cup \{\#, \$\}$ astfel:

$$x = (x_0, x_1, x_2, \dots, x_n, x_{n+1}), y = (y_0, y_1, y_2, \dots, y_n, y_{n+1}),$$

$$x_0 = \#x_1, x_i = h_1(\alpha_i), 1 \leq i \leq n, x_{n+1} = \$,$$

$$y_0 = y_1, y_i = h_2(\beta_i), 1 \leq i \leq n, y_{n+1} = \$\#,$$

$$h_1(a) = a\#, \quad h_2(a) = \#a, a \in V.$$

Exemplu: $\alpha = (b, a, aba, bb)$, atunci

$$x = (\#b\#, b\#, a\#, a\#b\#a\#, b\#b\#, \$).$$

Problema Corespondentei lui Post.

Afirmatie. $MPCP(\alpha, \beta)$ are solutie ddaca $PCP(x, y)$ are solutie. Fie $1, i_1, \dots, i_k$ o solutie pentru $MPCP(\alpha, \beta)$, deci:

$$\alpha_1\alpha_{i_1} \dots \alpha_{i_k} = \beta_1\beta_{i_1} \dots \beta_{i_k}.$$

Atunci: $x_0x_{i_1}x_{i_2} \dots x_{i_k}x_{n+1} = y_0y_{i_1}y_{i_2} \dots y_{i_k}y_{n+1}$, deci $PCP(x, y)$ are solutie.

Reciproc, $PCP(x, y)$ are solutie. Atunci primul indice este 0 iar ultimul este $n + 1$. Prin stergerea simbolului $\#$ se obtine o solutie pentru $MPCP(\alpha, \beta)$.

TEOREMA. Problema MPCP nu este decidabila.

DEM. PpA MPCP este decidabila. Vom construi doua liste Post α, β a.i. $MPCP(\alpha, \beta)$ are solutie ddaca o masina Turing data se opreste pe o intrare oarecare a sa.

Fie $M = (Q, V, U, \delta, q_0, B, F)$ o masina Turing determinista si $w \in V^*$ o intrare a sa.

Problema Corespondentei lui Post.

Lista α	Lista β	Conditie	Grup
#	# $q_0 w \#$	Fara conditii	0
a	a	pentru orice $a \in U \setminus \{B\}$	
\$	\$		
#	#		1

Pentru orice $q, p \in Q$, $a, b, c \in U \setminus \{B\}$:

qa	bp	daca $\delta(q, a) = (p, b, R)$	
cqa	pcb	daca $\delta(q, a) = (p, b, L)$	
$q\#$	$bp\#$	daca $\delta(q, B) = (p, b, R)$	
$cq\#$	$pcb\#$	daca $\delta(q, B) = (p, B, L)$	2

Pentru orice $q \in Q$ si $a \in U \setminus \{B\}$:

qa	\$	daca $\delta(q, a)$ nedefinita	
$q\#$	\$#	daca $\delta(q, B)$ nedefinita	3

Problema Corespondentei lui Post.

Lista α	Lista β	Conditie	Grup
$a\$b$	\$	pentru orice $a, b \in U \setminus \{B\}$	
$\$b$	\$	pentru orice $b \in U \setminus \{B\}$	
$a\$$	\$	pentru orice $a \in U \setminus \{B\}$	4
\$##	#	fara conditii	5

Afirmatie. $MPCP(\alpha, \beta)$ are solutie daca M se opreste pe intrarea w .

Dem. Fie $C_0, C_1, \dots, C_n, \dots$ secventa de configuratii pe intrarea w .

Atunci, pentru orice $j \geq 0$, listele partiale α si β vor arata astfel:

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \#$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# C_j \#.$$

Problema Corespondentei lui Post.

Inductie dupa j : $j \rightarrow j + 1$. Fie $C_j : xqay$ si $\delta(q, a) = (p, b, R)$. Atunci $C_{j+1} : xbpy$ (celelalte cazuri se trateaza similar).

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \#$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \#$$

Problema Corespondentei lui Post.

Inductie dupa j : $j \rightarrow j + 1$. Fie $C_j : xqay$ si $\delta(q, a) = (p, b, R)$. Atunci $C_{j+1} : xbpy$ (celelalte cazuri se trateaza similar).

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \#$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \#$$

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \# x$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \# x$$

Problema Corespondentei lui Post.

Inductie dupa j : $j \rightarrow j + 1$. Fie $C_j : xqay$ si $\delta(q, a) = (p, b, R)$. Atunci $C_{j+1} : xbpy$ (celelalte cazuri se trateaza similar).

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \#$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \#$$

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \# x$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \# x$$

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqa$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \# xbp$$

Problema Corespondentei lui Post.

Inductie dupa j : $j \rightarrow j + 1$. Fie $C_j : xqay$ si $\delta(q, a) = (p, b, R)$. Atunci $C_{j+1} : xbpy$ (celelalte cazuri se trateaza similar).

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \#$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \#$$

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \# x$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \# x$$

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqa$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \# xbp$$

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \#$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \# xbpy \#.$$

Problema Corespondentei lui Post.

Deci, M nu se opreste pe w implica $MPCP(\alpha, \beta)$ nu are solutie.

Pp. ca M se opreste la pasul $j + 1$.

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqa$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \# x \$$$

Problema Corespondentei lui Post.

Deci, M nu se opreste pe w implica $MPCP(\alpha, \beta)$ nu are solutie.

Pp. ca M se opreste la pasul $j + 1$.

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqa$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \# x \$$$

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \#$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \# x \$ y \#.$$

Folosim valorile listelor α, β din grupul 4 pana cand ajungem in situatia

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \# \dots \#$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \# x \$ y \dots \# \$ \#.$$

Folosim valorile din grupul 5:

$$\alpha : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \# \dots \# \$ \# \#$$

$$\beta : \#C_0 \# C_1 \# \dots \# C_{j-1} \# xqay \# x \$ y \dots \# \$ \# \#.$$

Problema Corespondentei lui Post.

Particularizari

- ① Alfabetul are o singura literă.

Problema Corespondentei lui Post.

Particularizari

- ① Alfabetul are o singura litera.(Decidabila)
- ② Lungimea listelor este cel mult 2.

Problema Corespondentei lui Post.

Particularizari

- ① Alfabetul are o singura litera.(Decidabila)
- ② Lungimea listelor este cel mult 2.(Decidabila)
- ③ Lungimea listelor este cel putin 7.

Problema Corespondentei lui Post.

Particularizari

- ① Alfabetul are o singura litera.(Decidabila)
- ② Lungimea listelor este cel mult 2.(Decidabila)
- ③ Lungimea listelor este cel putin 7.(Nedecidabila)
- ④ Lungimea listelor intre 3 si 6.

Problema Corespondentei lui Post.

Particularizari

- ① Alfabetul are o singura litera.(Decidabila)
- ② Lungimea listelor este cel mult 2.(Decidabila)
- ③ Lungimea listelor este cel putin 7.(Nedecidabila)
- ④ Lungimea listelor intre 3 si 6. (Nu se stie)
- ⑤ PCP marginita.

Problema Corespondentei lui Post.

Particularizari

- ① Alfabetul are o singura litera.(Decidabila)
- ② Lungimea listelor este cel mult 2.(Decidabila)
- ③ Lungimea listelor este cel putin 7.(Nedecidabila)
- ④ Lungimea listelor intre 3 si 6. (Nu se stie)
- ⑤ PCP marginita.(NP-completa)
- ⑥ Fiecare lista are toate elementele distincte.

Problema Corespondentei lui Post.

Particularizari

- ① Alfabetul are o singura litera.(Decidabila)
- ② Lungimea listelor este cel mult 2.(Decidabila)
- ③ Lungimea listelor este cel putin 7.(Nedecidabila)
- ④ Lungimea listelor intre 3 si 6. (Nu se stie)
- ⑤ PCP marginita.(NP-completa)
- ⑥ Fiecare lista are toate elementele distincte.(Nedecidabila)
- ⑦ PCP 1-marcata (nu exista 2 elemente care incep cu aceeasi litera)

Problema Corespondentei lui Post.

Particularizari

- ① Alfabetul are o singura litera.(Decidabila)
- ② Lungimea listelor este cel mult 2.(Decidabila)
- ③ Lungimea listelor este cel putin 7.(Nedecidabila)
- ④ Lungimea listelor intre 3 si 6. (Nu se stie)
- ⑤ PCP marginita.(NP-completa)
- ⑥ Fiecare lista are toate elementele distincte.(Nedecidabila)
- ⑦ PCP 1-marcata (nu exista 2 elemente care incep cu aceeasi litera)(Decidabila, PSPACE)
- ⑧ PCP 2-marcata.

Problema Corespondentei lui Post.

Particularizari

- ① Alfabetul are o singura litera.(Decidabila)
- ② Lungimea listelor este cel mult 2.(Decidabila)
- ③ Lungimea listelor este cel putin 7.(Nedecidabila)
- ④ Lungimea listelor intre 3 si 6. (Nu se stie)
- ⑤ PCP marginita.(NP-completa)
- ⑥ Fiecare lista are toate elementele distincte.(Nedecidabila)
- ⑦ PCP 1-marcata (nu exista 2 elemente care incep cu aceeasi litera)(Decidabila, PSPACE)
- ⑧ PCP 2-marcata.(Nedecidabila).

= Curs 9 =

Complexitate : măsură de complexitatea calculei

Relativ între clase.

P vs NP - reduceri

Complexitate : calculelor

Kolmogorov.

Descriptivă

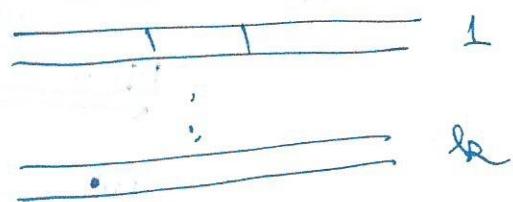
Măsura time (TIME)

spătii (SPACE)

Modelul de urmă : 1. urmă de opere

2. Are la bază
infinitele
asemănător capete

3. Poate staționa



Calcul

$$C(M, \alpha) = C_1 C_2 \dots C_n$$

α intrare C_i configurație

$$\text{time}(C(M, \alpha)) = n$$

$$\text{time}_M(x) = \max \{ \text{time}(C(M, \alpha)) \mid C(M, \alpha) \text{ calculează } x \}$$

$$\text{time}_M(n) = \max \{ \text{time}_M(x) \mid \forall x \text{ a.s.t. } |x|=n \}$$

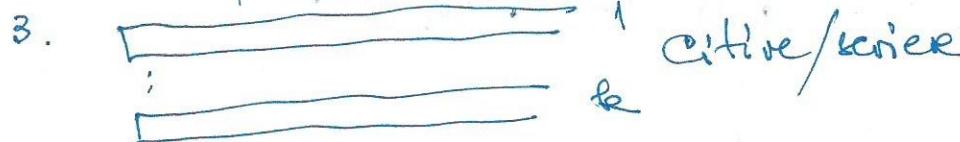
$$\text{TIME}_k(f(n)) = \{ L \mid \exists M \text{ o urmă ce la baza ei are } k \text{ capete a.s.t. } L = L(M) \text{ și } \text{time}_M(n) \leq f(n), \forall n \geq 0 \}$$

Obs ~~este~~ $f(n) \geq n+1$.

SPACE : Modelul : 1. urmă se operează

(off-line) 2. Are o bandă de intrare -

w intrare
disponibilită doar pentru citire



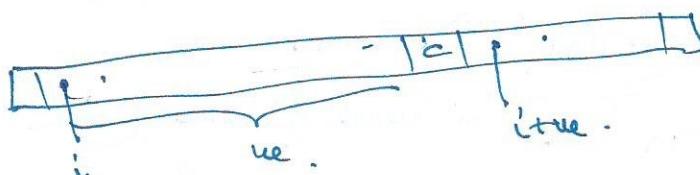
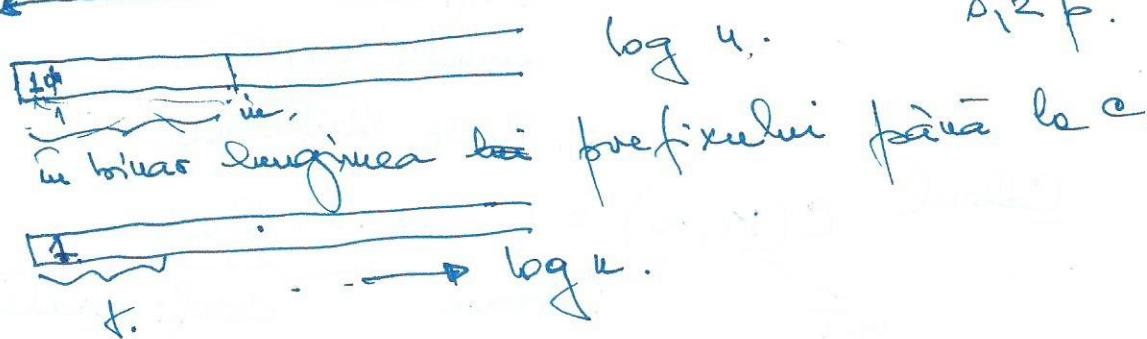
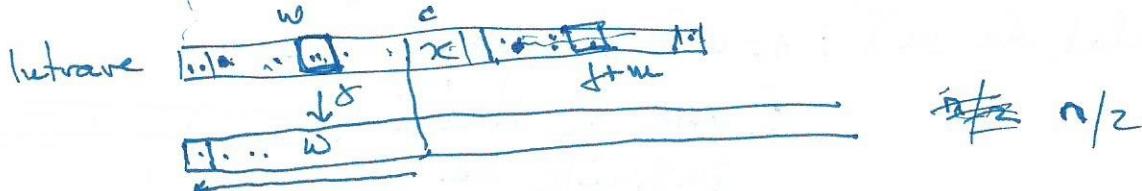
$$C(M, x) = c_1 \dots c_n$$

$\text{space}(M, x)$ = cel mai mare număr de celele folosite pe o bandă auxiliară $(1-k)$ în calculul C .

$$\text{space}_M(u) = \max \{\text{space}(M, x)\}$$

$$\text{SPACE}_k(f(u)) = \{L \mid \exists M \text{ s.t. } u \text{ are } k \text{ beneficii}, \\ \text{a.s. } L = L(M) \text{ și } \text{space}_M(u) \leq f(u) + \frac{H_u}{2}\}$$

Ex. $L = \{w \in w \mid w \in [a, b]^n\}^+$



! $O(f(u)) = O(cf(u))$!

Eliminarea constantei

① ~~$O(f(u)) \text{SPACE}_k(f(u)) = O(N) \text{SPACE}_k(cf(u))$~~

$$+ c > 0$$

Dacă aleg N . Este suficient să scriu

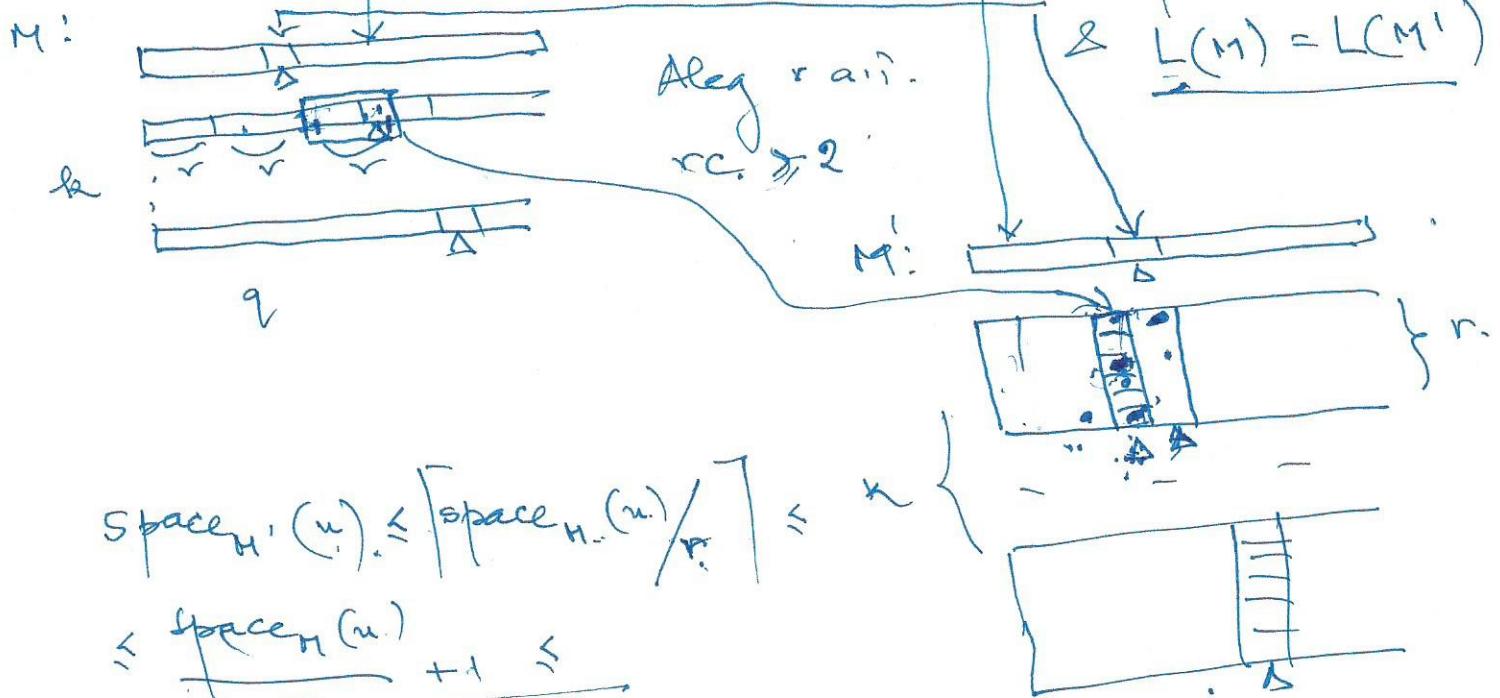
② *

$$\boxed{\text{NSPACE}_k(f(u)) = \text{NSPACE}_k(cf(u))} \\ O(c < 1)$$

$c \geq 1 \quad \text{NSPACE}_k(f(u)) \leq \text{NSPACE}_k(cf(u))$

= 3 =

\rightarrow Dem. M a.i. $\text{space}_M(u) \leq f(u)$ $\rightarrow M'$ a.i.
 $\text{space}_{M'}(u) \leq c f(u)$



$$\frac{c}{2} \cdot \text{space}_M(u) + 1 \leq c \text{space}_M(u) \quad ? \leq \left(\frac{c}{2}\right) \text{space}_M(u)$$

Dacă $1 \leq \frac{c}{2} \text{space}_M(u)$

$$\rightarrow \frac{c}{2} \cdot f(u) + 1 \leq c f(u) \quad (\Rightarrow \frac{1}{2} \leq \frac{c}{2} f(u))$$

Dacă $1 \leq \frac{c}{2} f(u)$, are terminat

~~$\text{Dacă } 1 \leq c f(u) \Rightarrow \frac{c}{2} f(u) \leq \frac{1}{2}$~~

M' va contine și baza a.i. spatială nău este
wânduit de 1.

$$f(u) < f'(u)$$

$$\begin{array}{c} \xrightarrow{\hspace{2cm}} f(u) \\ \xrightarrow{\hspace{2cm}} f'(u) \end{array}$$

phc 51

$\rightarrow M'$ a.i.

$\text{space}_{M'}(u) \leq c f(u)$

$$\& L(M) = L(M')$$

$$\begin{aligned} rc &\geq 2 \Rightarrow \\ c &\geq \frac{2}{r} \Rightarrow \\ \frac{1}{r} &\leq \frac{c}{2} \end{aligned}$$

= CURS 10 =

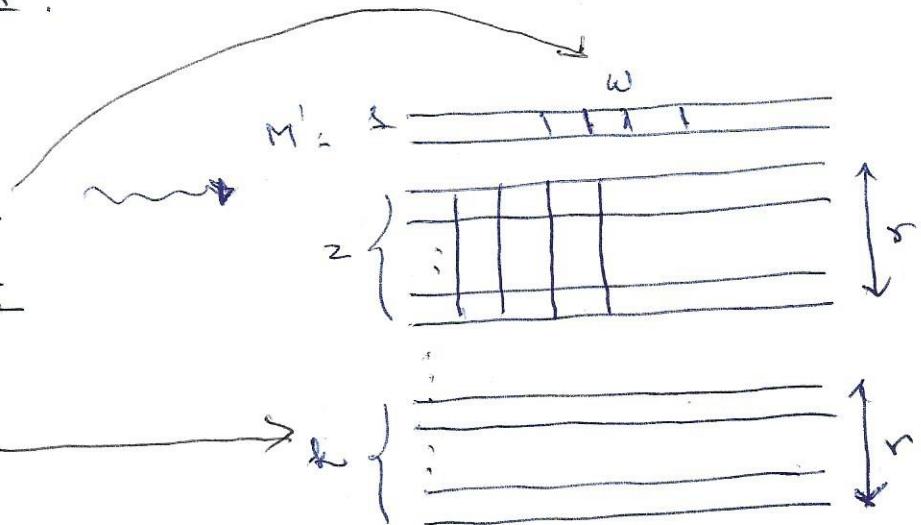
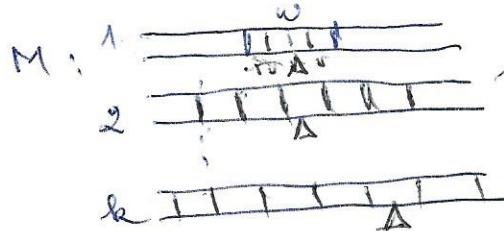
$$\textcircled{T} \quad (N)(\Delta) \text{SPACE}_k(f(u)) = (N)(\Delta) \text{SPACE}_k(c f(u)), \quad \forall c > 0.$$

$$\textcircled{T} \quad \lim_{k \geq 2} \frac{f(u)}{u} = \infty \quad \left\{ \begin{array}{l} (N)(\Delta) \text{TIME}_k(f(u)) = (N)(\Delta) \text{TIME}_k(c f(u)), \\ \quad \forall c > 0. \end{array} \right.$$

Dоказ.

Este important ca \exists $\text{time}_{\text{on}}(u) \leq f(u)$, cu u beingi si $c.a.i.$ $0 < c < 1$.

Construim M' :



M' : Etape

1. Citeste r substeuri de pe banda 1 si le trage intr-o nouă bandă adăugată pe banda 2

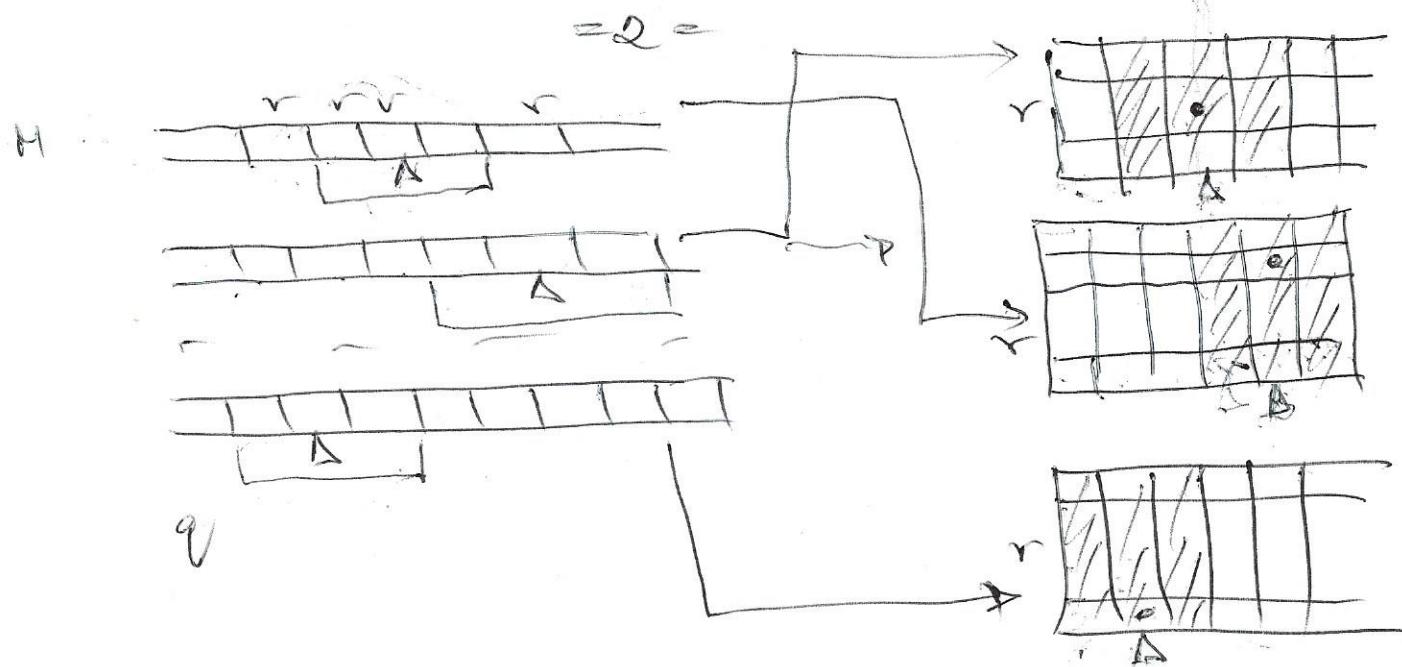
Timp: $O(n)$.

2. Repetă aceeași lucru pe banda 2 la început

Timp: $O(n)$.

3. Face o corespondență între benziile lui M' și cele ale lui M .

$$\begin{aligned} 1 &\rightarrow 2 \\ 2 &\rightarrow 1 \\ i &\rightarrow i \end{aligned}$$



q_1, q_2, q_3

4. M' este să mențină cele 3 stroboli aggregate în vecinătatea capului de cîrteze/sectorul pe flocare băndă.

Timp: 4

5. Așteaptă M să nu cănd:

- M se oprește

- cu cap ~~înainte~~ urmăză să părăsească segmentul de lg 3r pe una din banzi lejebul următor.

M lănează la stânga și construiește 3 sequențe de lg r pe flocare băndă.

Timp așteptare: 0.

M a lăsat minune r pasi.

6. Actualizare: stăre și colectivități

Timp: 4

Obs: Deceă M se oprește atunci și M' se oprește.

$\geq 3 \geq$

$$\underbrace{\text{time}_{\text{M1}}(u)}_{\leq u+1 + \lceil u/r \rceil + 8 \left\lceil \frac{f(u)}{r} \right\rceil} \leq c f(u).$$

$$\text{time}(u) \leq u+1 + \left\lceil \frac{u}{r} \right\rceil + 1 + \frac{8f(u)}{r} + 8 = u+10 + \frac{u}{r} + \frac{8f(u)}{r}.$$

$$\boxed{\text{Dacă } u \geq 10} \Rightarrow u+10 \leq 2u.$$

$$\text{time}(u) \leq 2u + \frac{u}{r} + \frac{8f(u)}{r}.$$

$$\text{Alegere: } r \in \mathbb{Z} \geq 16 \Rightarrow \frac{1}{r} \leq \frac{c}{16}.$$

$$\text{time}(u) \leq 2u + \frac{cu}{16} + \frac{8f(u) \cdot c}{16}$$

$$\text{Iată } \frac{f(u)}{u} = d \Rightarrow \text{Iată } \exists u_d \text{ a.i. } \boxed{u \geq u_d} \quad \frac{f(u)}{u} \geq d$$

$$\Rightarrow u \leq \frac{f(u)}{d}$$

$$\begin{aligned} \text{time}(u) &\leq \frac{2f(u)}{d} + \frac{8c \cdot f(u)}{(16d)} + \frac{8c \cdot f(u)}{16} = \\ &= f(u) \left[\frac{2}{d} + \frac{c}{16d} + \frac{8c}{16} \right] \leq c f(u). \end{aligned}$$

$$\frac{2}{d} + \frac{c}{16d} + \frac{8c}{16} \leq c \Rightarrow \frac{32}{16d} + \frac{c}{16d} + \frac{8cd}{16d} \leq c$$

$$32 + c + 8cd \leq 16cd \Rightarrow 32 + c \leq 8cd \Rightarrow$$

$$\Rightarrow \boxed{d \geq \frac{32+c}{8c}}$$

$$\text{Aleg } d \geq \frac{32+c}{8c} \text{ și } \boxed{u \geq \max(u_d, 10)}.$$

$$\Rightarrow \text{time}(u) \leq c f(u).$$

Dacă $u < \max(u_d, 10)$. Există un număr finit de astfel de u .

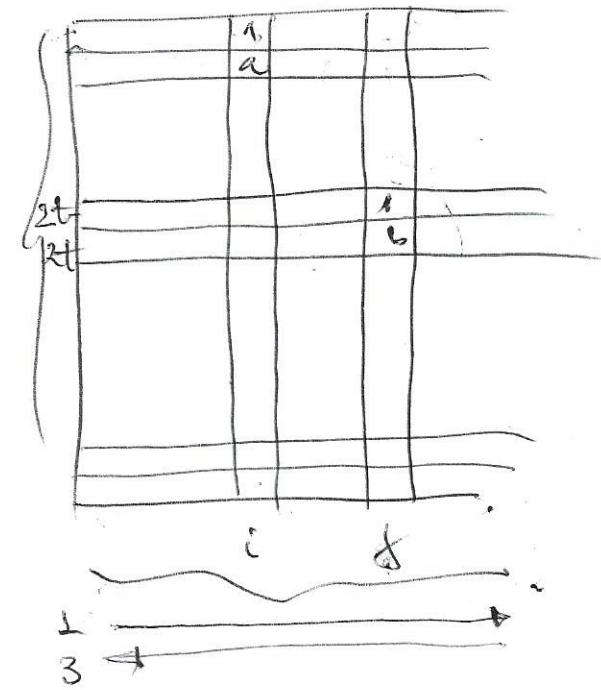
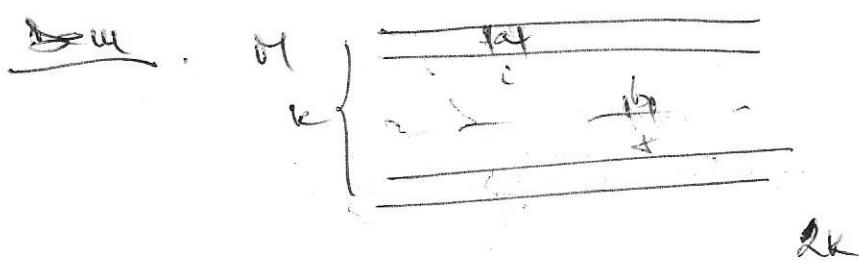
- 4 -

$$\textcircled{6} \quad (N)(D) \text{ TIME}_k(f(u)) = (N)(D) \text{ TIME}_{k+2}((1+2)f(u)), \quad k \geq 2.$$

$\forall \varepsilon > 0.$

Combinarea lucrărilor.

- \textcircled{7} Dacă $L \in (N)(D)$ SPACE_k(f(u)), atunci $L \in$ (N)(D) SPACE_{k+1}(f(u)), $\forall k \geq 1$.
- Dacă $L \in (N)(D)$ TIME_k(f(u)), atunci $L \in$ (N)(D) TIME_{k+1}(f²(u)), $\forall k \geq 1$.



TIME:

M' face cel mult $4f(u)$
măcar peste fiecare
măscăre a lui M.

$$\frac{1}{4}f(u).$$

$$\text{time}_M(u) \leq 4f(u) \cdot f(u) = 4f^2(u).$$

"Accesează" M și TIME_M(u) $\leq \frac{1}{2}f(u)$.

$$M' peste tot M și atunci$$

$$\text{time}_{M'}(u) \leq 4 \cdot \left(\frac{1}{2}f(u)\right)^2 = f^2(u).$$

(f(u))

⑦ Pentru orice funcție total recursivă există L
recursiv astfel încât $L \in \text{TIME}(f(u))$
 $L \in \text{SPACE}(f(u))$.

Dacă $L \in \text{TIME}$, atunci $f(u)$ recursivă
 Construim $L \in \{0,1\}^*$, $L = \{w \mid M_w \text{ nu acceptă } w\}$
 în $\text{Timep } f(u)\}$. $w_i \mid M_i$

1) L este recursiv.

Fie M' : se punt w_i :

calculează $f(1w_i1)$ și se verifică
 $f(1w_i1)$ este 0 sau 1.

găsește M_i :

S'calculă $f(M_i)$ pe intrarea w_i
 și se acceptă dacă M_i nu acceptă w_i
 și $f(1w_i1)$ este 0 sau 1.

2) $L \notin \text{TIME}(f(u))$, pp A că nu există.

$L(H) = L$ și $\text{Timep}(u) \leq f(u)$.

Găsește j. astfel încât, găseșc w_j

$w_j \in L \Leftrightarrow w_j$ este acceptat de H_j .
 în $\text{Timep } f(u) \Leftrightarrow w_j \notin L$.

Curs 11

December 22, 2021

lerarhii de clase de complexitate

Teorema. Daca $L \in TIME_k(f(n))$, atunci $L \in TIME_2(f(n) \log f(n))$.

Fara demonstratie.

Lema. Daca L este acceptat de o masina Turing (determinista) in spatiu $S(n) \geq \log n$, atunci L este acceptat de o masina Turing care se opreste pe fiecare intrare in spatiu $S(n)$.

Demonstratie. Fie M o masina Turing care accepta L in spatiu $S(n)$.

Atunci M accepta L printr-un calcul de lungime cel mult

$s(n+2)S(n)t^{S(n)}$, orice calcul mai lung avand configuratii care se repeta.

In formula de mai sus, s este numarul de stari iar t este numarul de simboluri pe banda auxiliara. Observam ca $s(n+2)S(n)t^{S(n)} \leq (4st)^{S(n)}$.

Construim M' care simuleaza M si care va folosi o banda pentru a numara tranzitiile in baza $4st$. Astfel M' se opreste fie cand M se opreste (si ia aceeasi decizie de acceptare) fie cand numarul de pasi depaseste $(4st)^{S(n)}$ (si respinge intrarea). Evident $space_{M'}(n) \leq S(n)$.

Ierarhii de clase de complexitate

Definitie. O functie $f(n)$ este *spatiu construibila* (sc) daca exista o masina Turing M a.i. $\text{space}_M(n) \leq f(n)$, pentru orice n si $\forall n \exists w_n$ a.i. $|w_n| = n$ si $\text{space}_M(w_n) = f(n)$.

Definitie. O functie $f(n)$ este *spatiu construibila complet* (scc) daca exista o masina Turing M a.i. pentru orice n si orice w a.i. $|w| = n$ avem $\text{space}_M(w) = f(n)$.

Definitii. Functiile *temp construibile* (tc) si *temp construibile complet* (tcc) se definesc analog.

Teorema. Daca $S_1(n), S_2(n) \geq \log n$, $S_2(n)$ este scc si $\lim \frac{S_1(n)}{S_2(n)} = 0$, atunci $\text{DSPACE}(S_2(n)) \setminus \text{DSPACE}(S_1(n)) \neq \emptyset$.

Dem. Idee: Construim o masina Turing M a.i. $\text{space}_M(n) \leq S_2(n)$ si limbajul sau difera prin cel putin un cuvant de orice limbaj acceptat de o masina Turing in spatiu $S_1(n)$.

Ierarhii de clase de complexitate

Continuare demonstratie. Modul de calcul al lui M :

- ① Primeste la intrare $w \in \{0, 1\}^*$, $|w| = n$;
- ② Marcheaza pe o banda auxiliara $S_2(n)$ celule si se va opri ori de cate ori va trebui sa foloseasca un spatiu mai mare. Deci $\text{space}_M(n) \leq S_2(n)$.
- ③ Identifica masina Turing M_{0y} codificata cu $0y$, unde $w = x0y$ si $x \in \{1\}^*$;
- ④ M simuleaza M_{0y} pe intrarea w . Putem presupune ca M se opreste intotdeauna (vezi lema anterioara).
- ⑤ M respinge w ddaca simularea s-a incheiat cu acceptarea lui w de M_{0y} .

Fie M' o masina Turing cu spatiul marginit de $S_1(n)$ si codificarea $z \in 0\{0, 1\}^*$. Putem presupune ca M' are t simboluri de banda si o singura banda auxiliara. Exista un cuvant $w' = 1^m z$, cu m suficient de mare a.i. $\lceil t \rceil S_1(|w'|) \leq S_2(|w'|)$. Rezulta ca $w' \in L(M)$ ddaca $w' \notin L(M')$. QED

Ierarhii de clase de complexitate

TEOREMA. Daca $T_2(n)$ este tcc si $\lim \frac{T_1(n) \log T_1(n)}{T_2(n)} = 0$, atunci $DTIME(T_2(n)) \setminus DTIME(T_1(n)) \neq \emptyset$.

DEM. Analog demonstratiei anerieoare cu urmatoarele observatii:

- Deoarece $T_2(n)$ este tcc, exista o mT M_0 care executa exact $T_2(n)$ pasi pe orice intrare de lungime n . M_0 se poate folosi ca sa marcam $T_2(n)$ celule pe o banda sau poate fi rulata alternativ cu masina M construita.
- Masina M_{0y} se poate reduce la o masina cu doua crescand complexitatea sa timp $T(n)$ cu un factor $\log T(n)$.

Relatii intre clase de complexitate

Teorema.

- ① $(N)(D)TIME(f(n)) \subseteq (N)(D)SPACE(f(n))$.
- ② Daca $L \in DSPACE(f(n))$ cu $f(n) \geq \log n$, atunci exista c_L a.i. $L \in DTIME(c_L^{f(n)})$.
- ③ Daca $L \in NTIME(f(n))$, atunci exista c_L a.i. $L \in DTIME(c_L^{f(n)})$.

Dem.

- ① Trivial.
- ② Fie M o masina Turing determinista care accepta L in spatiu $f(n)$. Atunci M accepta L printr-un calcul de lungime cel mult $s(n+2)f(n)t^{f(n)}$, orice calcul mai lung avand configuratii care se repeta. In formula de mai sus, s este numarul de stari iar t este numarul de simboluri pe banda auxiliara. Observam ca $s(n+2)f(n)t^{f(n)} \leq (4st)^{f(n)}$. Construim M' care simuleaza M si care va folosi o banda pentru a numara tranzitiile. Accepta daca M accepta si M' nu a efectuat mai mult de $(4st)^{f(n)}$ pasi.

Relatii intre clase de complexitate

Continuarea demonstratiei.

3 Fie M o masina Turing nedeterminista cu s stari, k benzi si t simboluri, care accepta L in timp $f(n)$. Numarul maxim de configuratii al lui M este $s(f(n) + 1)^k t^{f(n)} \leq d^{f(n)}$, unde $d = s(t + 1)^{3k}$. O mT determinista M' accepta limbajul lui M prin constructia unei liste cu toate configuratiile accesibile din configuratia initiala pe fiecare intrare. Aceasta lista are lungimea cel mult $d^{f(n)}$ si necesita un timp $(d^{f(n)})^2$. Fiecare configuratie are lungimea $1 + k(f(n) + 1)$ deci timpul total necesar lui M' este cel mult $(d^{f(n)})^2(1 + k(f(n) + 1))$ care se poate margini cu $c^{f(n)}$, pentru o constanta c convenabil aleasa.

Teorema lui Savitch

Teorema. Daca $S(n) \geq \log n$ si $S(n)$ este scc, atunci
 $\text{NSPACE}(S(n)) \subseteq \text{DSPACE}(S^2(n))$.

Dem. Fie M o mT nedeterminista cu spatiul marginit de $S(n)$. Pentru un input de lungime n , M accepta acest input printr-o seventa de cel mult $c^{S(n)}$ pasi.

Notam $C_1 \vdash; C_2$ procesul prin care configuratia C_2 este atinsa din C_1 printr-o seventa de cel mult 2^i pasi. Urmatorul pseudocod decide daca w , cu $|w| = n$, este acceptat de M .

$m \leftarrow \lceil \log c \rceil$

C_0 este configuratia initiala pe intrarea w

for fiecare configuratie finala C_f de lungime cel mult $S(n)$ **do**

if $\text{TEST}(C_0, C_f, mS(n))$ **then** accept

end if

end for

Teorema lui Savitch

Continuare demonstratie.

```
Function TEST( $C_1, C_2, i$ );
if ( $i = 0$ ) si ( $(C_1 = C_2)$  sau  $(C_1 \vdash C_2)$ ) then return true
end if
if  $i \geq 1$  then
    for fiecare configuratie  $C'$  de lungime cel mult  $S(n)$  do
        if  $TEST(C_1, C', i - 1)$  si  $TEST(C', C_2, i - 1)$  then return true
        end if
    end for
end if
return false
```

Teorema lui Savitch

Continuare demonstratie.

Acet algoritm poate fi implementat pe o masina determinista M' astfel:

- Foloseste o banda sau 3 benzi) ca o stiva pentru a salva argumentele functiei $TEST$, (C_1, C_2, i) .
- Fiecare configuratie este de lungime cel mult $S(n)$.
- Parametrul i se poate memora pe cel mult $mS(n)$ celule. (In binar chiar mai putin).
- Pozitia pe banda de intrare se salveaza in binar pe cel mult $\log n$ celule.
- Numarul de apelari este de cel mult $mS(n)$ (la fiecare apelare indicele i scade.)
- Spatiul pentru testarea $C_1 \vdash; C'$ poate fi refolosit pentru testarea $C' \vdash; C_2$.

Rezulta ca spatiul total al lui M' este $\mathcal{O}(S^2(n))$ si aplicand teorema de eliminare a constantelor demonstratia este completa.

Curs 12

January 11, 2022

Timp si spatiu polinomial

$$\mathbf{P} = \bigcup_{i \geq 1} DTIME(n^i)$$

$$\mathbf{NP} = \bigcup_{i \geq 1} NTIME(n^i)$$

$$\mathbf{PSPACE} = \bigcup_{i \geq 1} DSPACE(n^i)$$

$$\mathbf{NSPACE} = \bigcup_{i \geq 1} NSPACE(n^i)$$

$$\mathbf{L} = DSPACE(\log n).$$

L ⊆ P ⊆ NP ⊆ NSPACE = PSPACE.

Cel putin o incluziune este stricta: CARE?

Reduceri

Definitie. Masina Turing folosita in reduceri (similar unui translator): o masina determinista M care se opreste pe fiecare intrare a.i. pentru un input w produce un sir $f_M(w)$.

Un limbaj L' este *Turing-reductibil* la L daca exista o mT M a.i. $w \in L'$ ddaca $f_M(w) \in L$.

Definitie. L' este reductibil in timp polinomial la L ($L' \leq_{tp} L$) daca exista o mT M cu timp de lucru polinomial care reduce L' la L .

Translator off-line: o mT M care se opreste pe fiecare intrare, are o banda de intrare care se poate doar citi, o banda auxiliara, si o banda de iesire pe care se poate doar scrie fara a se putea intoarce.

Definitie. L' este reductibil in spatiu logaritmice la L ($L' \leq_{sl} L$) daca exista un translator off-line M cu spatiu de lucru logaritmice care reduce L' la L .

Proprietati ale reducerilor

Teorema. Fie $L' \leq_{tp} L$. Daca L este in $(N)P$ atunci L' este in $(N)P$.

Dem. Demonstram pentru $L \in P$. Fie $p_1(n)$ (polinom) timpul in care M reduce L' la L . Atunci pentru fiecare $|x| = n$, rezulta $|f_M(x)| \leq p_1(n)$.

Pentru ca $L \in P$, $f_M(x) \in L$ se poate face in timp $p_2(|f_M(x)|) \leq p_2(p_1(n))$. Asadar, $x \in L'$ se poate decide in timp $p_1(n) + p_2(p_1(n))$.

Teorema. Fie $L' \leq_{sl} L$.

1. Daca L este in P atunci L' este in P .

2. Daca $L \in (N)(D)SPACE(\log^k n)$ atunci $L' \in (N)(D)SPACE(\log^k n)$.

Dem. 1. Imediat, pentru ca orice reducere in spatiu logaritmice se poate face in timp polinomial.

Fie $L \in DSPACE(\log^k n)$. Fie M_1 translatorul care reduce L' la L si M_2 mT care accepta L in spatiu marginit de $\log^k n$. Lungimea iesirii lui M_1 pe intrarea $|x| = n$ este cel mult $s(n+2)t^{\log^k n}$, unde s este numarul de stari si t numarul de simboluri ale lui M_1 . Exista o constanta c a.i.

$$s(n+2)t^{\log^k n} \leq (2^c)^{\log^k n}.$$

Proprietati ale reducerilor

CONTINUARE DEM. Construim M_3 astfel:

- Pe banda de intrare are x , $|x| = n$.
- Are o banda auxiliara pe care simuleaza banda lui M_1 .
- Are o banda auxiliara pe care simuleaza banda lui M_2 .
- Are o banda auxiliara pe care va pastra pozitia i a capului de citire al lui M_2 pe banda sa de intrare, numar scris in baza 2^c . Deci spatiul folosit pe aceasta banda va fi cel mult $\log^k n$.
- Pentru fiecare astfel de i , M_3 restarteaza simularea lui M_1 pe intrarea x si numara simbolurile scrise la iesire de M_1 pana ajunge la i .
- Al i -lea simbol produs la iesire de M_1 este dat lui M_2 ca simbol curent pe banda sa de intrare.
- Simuleaza miscarea lui M_2 pentru acest simbol si actualizeaza i la $i - 1$ sau $i + 1$, si reia procesul.
- Cazuri particulare: (1) $i = 0$ inseamna ca M_2 citeste marginea stanga a benzii sale de intrare.
(2) Simularea lui M_1 se opreste fara a produce al i -lea simbol, inseamna ca M_2 citeste marginea dreapta a benzii sale de intrare.

Probleme dificile si complete

Teorema. Compunerea a doua reduceri de acelasi tip este o reducere de acelasi tip.

Fie \mathcal{L} o clasa de limbaje.

- Def.**
1. Un limbaj L este *dificil* pentru \mathcal{L} (\mathcal{L} -dificil) in raport cu reducerea \leq_{tp} sau \leq_{sl} daca pentru orice limbaj $L' \in \mathcal{L}$ avem $L' \leq_{tp} L$ sau $L' \leq_{sl} L$.
 2. Un limbaj L este *complet* pentru \mathcal{L} (\mathcal{L} -complet) in raport cu reducerea \leq_{tp} sau \leq_{sl} daca $L \in \mathcal{L}$ si L este \mathcal{L} -dificil in raport cu reducerea \leq_{tp} sau \leq_{sl} .
 3. Un limbaj este **NP-complet** daca este **NP-complet** in raport cu reducerea \leq_{tp} sau \leq_{sl} .

Probleme NP-complete

Teorema. (Cook-Levin) Problema satisfiabilitatii (SAT) este NP-completa.

SAT: Input: o formula in forma normala conjunctiva din calculul propositional

$$\begin{aligned}\alpha &= C_1 \wedge C_2 \wedge \cdots \wedge C_m, \text{ unde} \\ C_i &= (x_{i_1}^{e_1} \vee x_{i_2}^{e_2} \vee \dots x_{i_{k_i}}^{e_{k_i}}), \\ x^1 &= x \text{ si } x^0 = \bar{x}.\end{aligned}$$

Output: YES/NO daca exista/nu exista o asignare a variabilelor care o satisfac.

Limbajul L_{SAT} se defineste astfel: $L_{SAT} = \{<\alpha> | \alpha \text{ este satisfiabila}\}$.

Daca $\alpha = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, atunci

$$<\alpha> = < C_1 > \wedge < C_2 > \wedge \cdots \wedge < C_m >.$$

Daca $C_i = (x_{i_1}^{e_1} \vee x_{i_2}^{e_2} \vee \dots x_{i_{k_i}}^{e_{k_i}})$, atunci

$$< C_i > = (< x_{i_1}^{e_1} > \vee < x_{i_2}^{e_2} > \vee \cdots < x_{i_{k_i}}^{e_{k_i}} >).$$

$$< x_i > = xi_{(2)}, < \bar{x}_i > = \bar{x}i_{(2)}.$$

Probleme NP-complete

Exemplu. $\alpha = (x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_3)$

$<\alpha> = (x_1 \vee x_{10}) \wedge (x_1 \vee \bar{x}_{11}).$

Obs. Daca $|\alpha| = n$, atunci $|<\alpha>| \leq n \log n$. Deoarece vom lucra cu reduceri in spatiu logaritmice, vom considera ca $|<\alpha>| = n$, pentru ca $\log(n \log n) \leq 2 \log n$.

$L_{SAT} \in \textbf{NP}$: De ce? Se poate verifica in timp polinomial daca o asignare satisface formula. Cum?

Probleme NP-complete

Exemplu. $\alpha = (x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_3)$

$<\alpha> = (x_1 \vee x_{10}) \wedge (x_1 \vee \bar{x}_{11})$.

Obs. Daca $|\alpha| = n$, atunci $|<\alpha>| \leq n \log n$. Deoarece vom lucra cu reduceri in spatiu logaritmice, vom considera ca $|<\alpha>| = n$, pentru ca $\log(n \log n) \leq 2 \log n$.

$L_{SAT} \in \mathbf{NP}$: De ce? Se poate verifica in timp polinomial daca o asignare satisface formula. Cum?

- ① Se determina variabilele care apar in formula. (determinist)
- ② Se genereaza nedeterminist o asignare. (Nedeterminist)
- ③ Se verifica daca asignarea satisface formula. (determinist)
- ④ Complexitate?

Probleme NP-complete

Exemplu. $\alpha = (x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_3)$

$<\alpha> = (x_1 \vee x_{10}) \wedge (x_1 \vee \bar{x}_{11})$.

Obs. Daca $|\alpha| = n$, atunci $|<\alpha>| \leq n \log n$. Deoarece vom lucra cu reduceri in spatiu logaritmice, vom considera ca $|<\alpha>| = n$, pentru ca $\log(n \log n) \leq 2 \log n$.

$L_{SAT} \in \mathbf{NP}$: De ce? Se poate verifica in timp polinomial daca o asignare satisface formula. Cum?

- ① Se determina variabilele care apar in formula. (determinist)
- ② Se genereaza nedeterminist o asignare. (Nedeterminist)
- ③ Se verifica daca asignarea satisface formula. (determinist)
- ④ Complexitate? $O(n^2)$.

Probleme NP-complete

L_{SAT} este NP-dificil (schita).

Fie M o mT nedeterminista oarecare care decide L în timp polinomial $p(n)$.

Construim o formulă α a.i. $w \in L(M)$ dacă α este satisfiabilă.

Variabile:

- $T_{k,i}$: la pasul k , celula i conține simbolul j ; ($p^2(n)$)
- $RW_{k,i}$: la pasul k , capul R/W citește celula i ; ($p^2(n)$)
- $Q_{k,s}$: la pasul k , starea curentă este s . ($p(n)$)

Formulă:

$$\alpha = U \wedge I \wedge D \wedge N \wedge E,$$

- U : la orice pas, M se află într-o singură stare, capul R/W accesează o singură celulă, și fiecare celulă are o singură literă;
- I : configurația initială a mașinii, deci pasul $k = 0$;
- D : tranzitiiile posibile, conform funcției de tranzitie;
- N : toate celulele neprocesate păstrează litera de la un pas la altul;
- E : condiția de acceptare.

Probleme NP-complete

Teorema. Problema 3 – SAT este NP-completa.

Dem. Fie o clauza $C = l_1 \vee \dots \vee l_k$, $k \geq 1$.

- Cazul 1. $k > 3$. Inlocuim C cu

$$C' =$$

$$(l_1 \vee l_2 \vee y_1) \wedge (l_3 \vee \bar{y}_1 \vee y_2) \wedge \dots \wedge (l_{k-2} \vee \bar{y}_{k-4} \vee y_{k-3}) \wedge (l_{k-1} \vee l_k \vee \bar{y}_{k-3}).$$

C este satisfiabila daca C' este satisfiabila. De ce?

- Cazul 2. $k = 2$. Inlocuim C cu $C' = (l_1 \vee l_2 \vee y) \wedge (l_1 \vee l_2 \vee \bar{y})$.
- Cazul 3. $k = 1$. Inlocuim C cu

$$C' = (l_1 \vee y_1 \vee y_2) \wedge (l_1 \vee \bar{y}_1 \vee y_2) \wedge (l_1 \vee y_1 \vee \bar{y}_2) \wedge (l_1 \vee \bar{y}_1 \vee \bar{y}_2).$$

Transformarile se pot face in spatiu logaritmic.

Problema. Care este statutul problemei 2-SAT?

Probleme NP-complete

Teorema. Problema VERTEX COVER este NP-completa.

VERTEX COVER: Un graf neorientat $G = (V, E)$; $|V| = n$, $|E| = m$, o acoperire a lui G este o submultime $X \subseteq V$ a.i. $\{u, v\} \cap X \neq \emptyset$, pentru orice $\{x, y\} \in E$.

Problema: Pentru un graf G si $k \geq 1$, exista o acoperire a lui G de cardinal maxim k ?

Codificarea unei intrari:

Probleme NP-complete

Teorema. Problema VERTEX COVER este **NP-completa**.

VERTEX COVER: Un graf neorientat $G = (V, E)$; $|V| = n$, $|E| = m$, o acoperire a lui G este o submultime $X \subseteq V$ a.i. $\{u, v\} \cap X \neq \emptyset$, pentru orice $\{x, y\} \in E$.

Problema: Pentru un graf G si $k \geq 1$, exista o acoperire a lui G de cardinal maxim k ?

Codificarea unei intrari:

$k_{(2)} \# v1_{(2)} \# v2_{(2)} \# \dots \# vn_{(2)} \# (vi1_{(2)}, vj1_{(2)}) \# \dots \# (vim_{(2)}, vjm_{(2)})$.

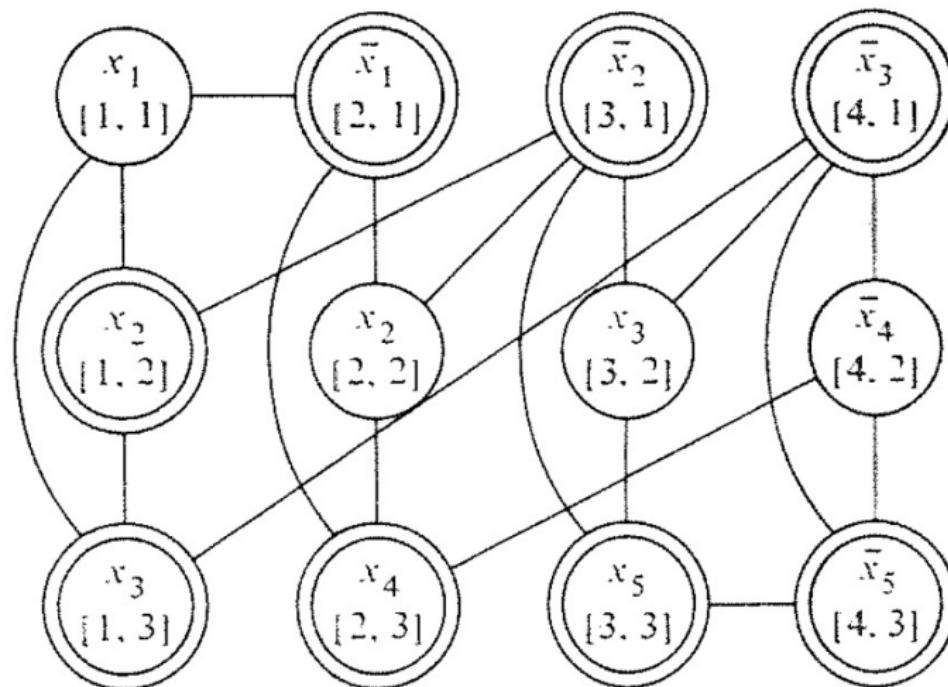
L_{VC} contine multimea tuturor codificarilor pentru care raspunsul este DA.

L_{VC} este in **NP**? De ce ?

Probleme NP-complete

L_{VC} este NP-dificil? Reducem 3-SAT la VERTEX COVER.

Ex. $\alpha = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (\bar{x}_2 \vee x_3 \vee x_5) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5)$.
 $k = 8$



Probleme NP-complete

Problema clicii este NP-completa

Problema clicii: Pentru un graf G si $k \geq 1$, exista un subgraf complet al lui G (clica) de cardinal minim k ?

Problema VERTEX COVER se poate reduce in spatiu logaritmice/timp polinomial la problema clicii. Cum?

Probleme NP-complete

Problema clicii este NP-completa

Problema clicii: Pentru un graf G si $k \geq 1$, exista un subgraf complet al lui G (clica) de cardinal minim k ?

Problema VERTEX COVER se poate reduce in spatiu logaritmice/timp polinomial la problema clicii. Cum? Se construieste graful complementar al grafului dat.

Alte probleme NP-complete:

- ① SUBSET SUM: Se dau numerele naturale a_1, a_2, \dots, a_n si b . Exista o submultime de suma b ?
- ② PARTITION: Se dau numerele naturale a_1, a_2, \dots, a_n . Se pot partitiona in doua submultimi avand aceeasi suma?
- ③ BIN PACKING: Se dau numerele naturale a_1, a_2, \dots, a_n , toate mai mici decat b si $k \geq 2$. Se pot partitiona in cel mult k submultimi fiecare avand suma cel mult b ?
- ④ Problema drumului (circuitului) hamiltonian.
- ⑤ Problema 3-colorarii unui graf (planar).
- ⑥ SET COVER.

Echivalenta expresiilor regulate

Problema 1. Pentru două expresii regulate date R_1, R_2 , este $L(R_1) = L(R_2)$?

- Este problema în NP?

Echivalenta expresiilor regulate

Problema 1. Pentru doua expresii regulate date R_1, R_1 , este $L(R_1) = L(R_2)$?

- Este problema in **NP**? Nu se stie.
- Schimbam problema in complementara sa: Sunt doua expresii date non-echivalente? Este in **NP**?

Echivalenta expresiilor regulate

Problema 1. Pentru doua expresii regulate date R_1, R_1 , este $L(R_1) = L(R_2)$?

- Este problema in **NP**? Nu se stie.
- Schimbam problema in complementara sa: Sunt doua expresii date non-echivalente? Este in **NP**? Nu se stie. ($w \in L(R_1) \setminus L(R_2)$ sau $w \in L(R_2) \setminus L(R_1)$.)
- Particularizam problema: Expresii regulate *-libere (expresii regulate fara operatia de inchidere Kleene). NEER: Este problema non-echivalentei a doua expresii regulate *-libere in **NP**?

Echivalenta expresiilor regulate

Problema 1. Pentru doua expresii regulate date R_1, R_1 , este $L(R_1) = L(R_2)$?

- Este problema in **NP**? Nu se stie.
- Schimbam problema in complementara sa: Sunt doua expresii date non-echivalente? Este in **NP**? Nu se stie. ($w \in L(R_1) \setminus L(R_2)$ sau $w \in L(R_2) \setminus L(R_1)$.)
- Particularizam problema: Expresii regulate *-libere (expresii regulate fara operatia de inchidere Kleene). NEER: Este problema non-echivalentei a doua expresii regulate *-libere in **NP**? Da:
$$R = ((a + b)aa(a + b) + aba(a + b)b).$$
- $SAT \leq_{tp} NEER$.

$SAT \leq_{tp} NEER$

Fie $\alpha = C_1 \wedge C_2 \wedge \dots \wedge C_p$ cu variabilele x_1, x_2, \dots, x_n ; construim două expresii regulate *-libere peste $\{0, 1\}$:

$$R_1 = (0 + 1)^n$$

$$R_2 = Z_1 + Z_2 + \dots + Z_p,$$

$$Z_i = Z_i^1 \cdot Z_i^2 \cdot \dots \cdot Z_i^n \text{ și } Z_i^j = \begin{cases} 0, & \text{daca } x_j \text{ apare în } C_i \\ 1, & \text{daca } \bar{x}_j \text{ apare în } C_i \\ (0 + 1), & \text{altfel} \end{cases}$$

Z_i descrie asignările care invalidează C_i .

Probleme complete pentru alte clase

- Problema satisfiabilitatii unei formule din calculul cu predicate (logica de ordinul intai) fara variabile libere este **PSPACE**-completa.
- Problema apartenentei pentru o gramatica dependenta de context (monotona) este **PSPACE**-completa. (Surprinzator pentru ca este in $NSPACE(n)$).
- Problema trivialitatii limbajului generat de o gramatica independenta de context este **P**-completa.
- Problema reachabilitatii este **NL**-completa.

$$\mathbf{EXP} = \bigcup_{i \geq 1} \mathbf{DTIME}(2^{n^i})$$

$$\mathbf{NEXP} = \bigcup_{i \geq 1} \mathbf{NTIME}(2^{n^i}).$$

$$\mathbf{L} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{NSPACE} = \mathbf{PSPACE} \subseteq \mathbf{EXP} \subseteq \mathbf{NEXP}.$$

Teorema Daca $\mathbf{P} = \mathbf{NP}$ atunci $\mathbf{EXP} = \mathbf{NEXP}$.

P, NP, co-NP, PSPACE

O problema este in **co-NP** daca complementara sa este in **NP**. Definitie cu masini Turing?

P, NP, co-NP, PSPACE

O problema este in **co-NP** daca complementara sa este in **NP**. Definitie cu masini Turing?

Exemplu de problema: TAUT=este tautologie o formula data?

Teorema. TAUT este **co-NP**-completa.

Dem. 1. TAUT este in **co-NP**. De ce?

P, NP, co-NP, PSPACE

O problema este in **co-NP** daca complementara sa este in **NP**. Definitie cu masini Turing?

Exemplu de problema: TAUT=este tautologie o formula data?

Teorema. TAUT este **co-NP**-completa.

Dem. 1. TAUT este in **co-NP**. De ce?

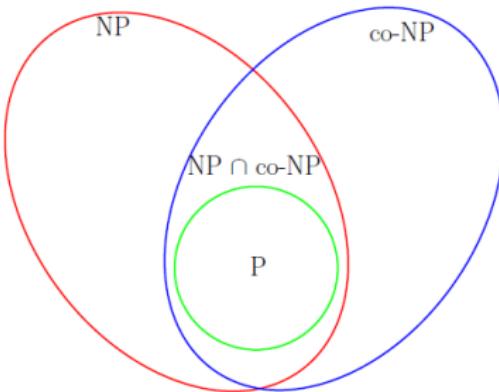
Pentru ca $\overline{\text{TAUT}}$ este in **NP**.

2. TAUT este **co-NP** dificila. De ce?

Orice problema **NP**-completa are complementara **co-NP**-completa. Fie L **NP**-complet, atunci $\bar{L} \in \text{co - NP}$. Fie $L' \in \text{co - NP}$, deci $\bar{L}' \in \text{NP}$; prin urmare $\bar{L}' \leq_{tp} L$. Rezulta ca $L' \leq_{tp} \bar{L}$.

Dar $SAT \leq_{tp} \overline{\text{TAUT}}$.

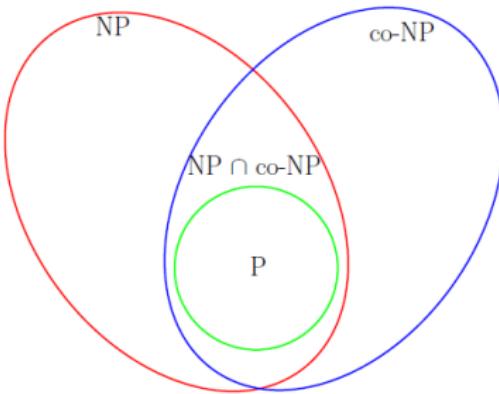
P, NP, co-NP, PSPACE



Relatii.

- ① $P = \text{co-}P$.
- ② $P \subseteq NP \cap \text{co-NP}$. Sunt egale? (Conjectura NU). Extrem de important in criptografie.
- ③ $NP = \text{co-NP}$? (Conjectura: NU) Ar fi egale daca complementul unei probleme **NP**-complete are fi in **NP**. De ce?

P, NP, co-NP, PSPACE



Relatii.

- ① $P = \text{co-}P$.
- ② $P \subseteq NP \cap \text{co-NP}$. Sunt egale? (Conjectura NU). Extrem de important in criptografie.
- ③ $NP = \text{co-NP}$? (Conjectura: NU) Ar fi egale daca complementul unei probleme **NP**-complete are fi in **NP**. De ce? Daca $L' \leq_{sl} L$ atunci $\bar{L}' \leq_{sl} \bar{L}$.