

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 - complexitate.

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

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

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

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 .~~

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

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. $\underbrace{+0}$

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

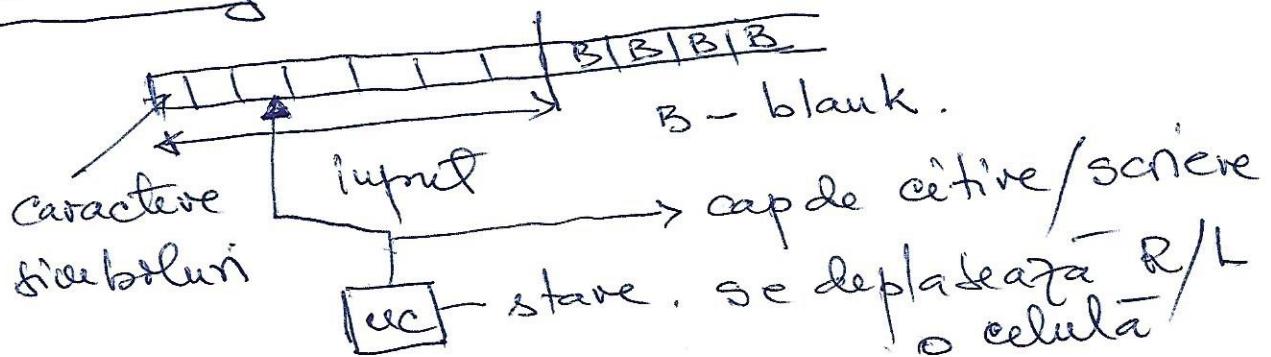
↑ multiplu de 0.

k k k

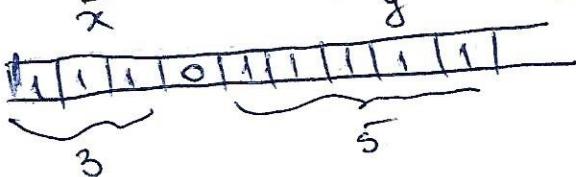
$$x = \sqrt{2} - 1.$$

Mașina Turing

\rightarrow infinită



Program: freccie de transiție.

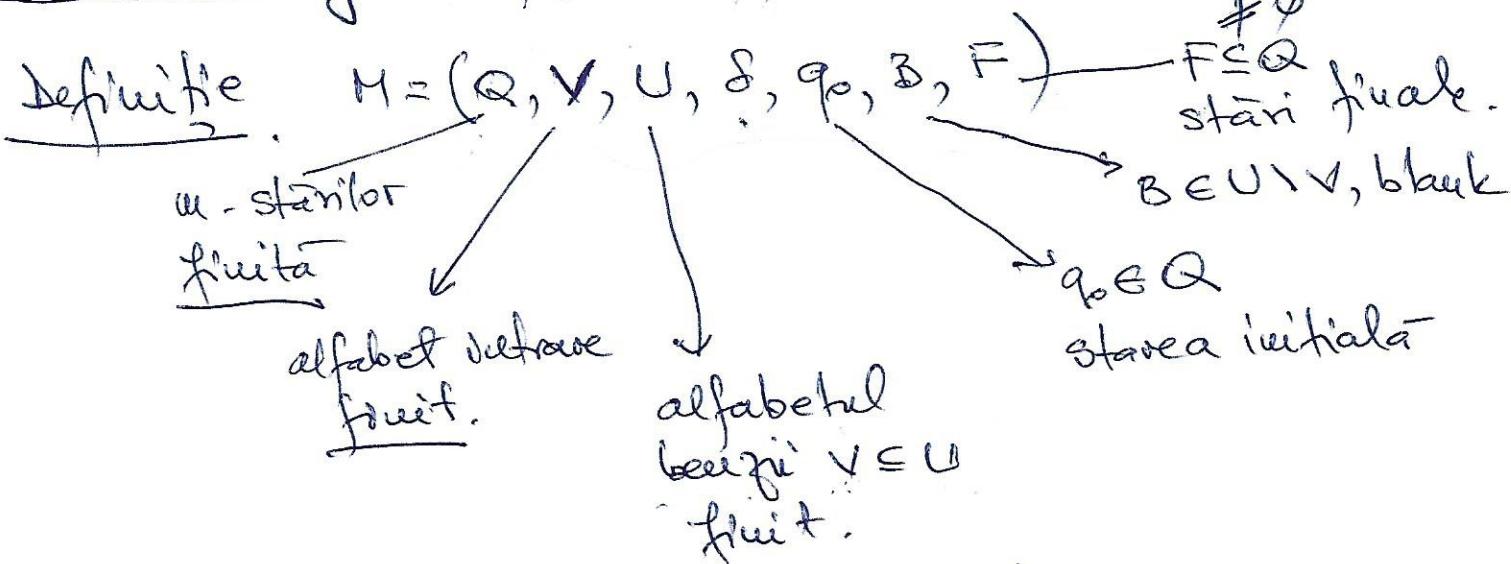


\rightarrow 111100...0B... .

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

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

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

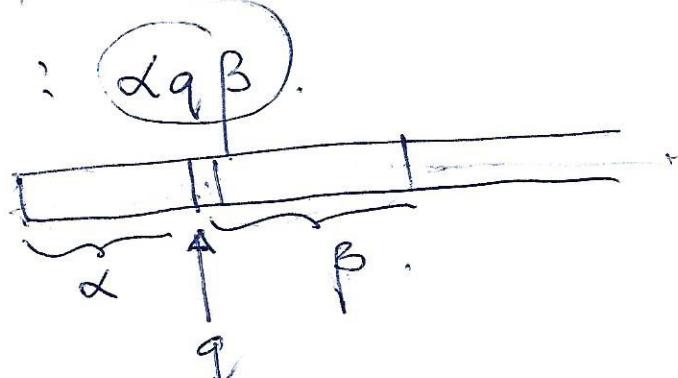


Masina este in starea "q" si cítete "a", poate scrie "b" peste "a" si trece in starea "s", se deplaseaza catre dreapta.

| | 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) .

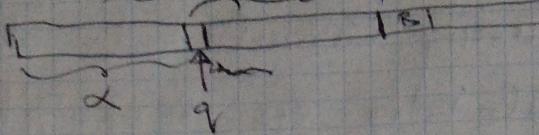


Ex. qP

$\xrightarrow{q} q'$

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

$q\beta$ - configurație β



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

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

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

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

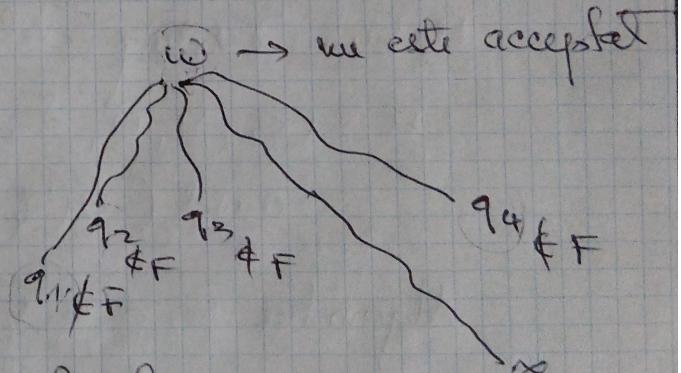
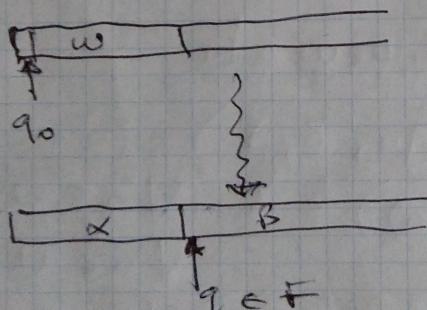
M - dispozitiv de acceptare

(acceptă limbaj)

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

echidondere reflexivă

și transiția a
relației $\xleftarrow{*}$

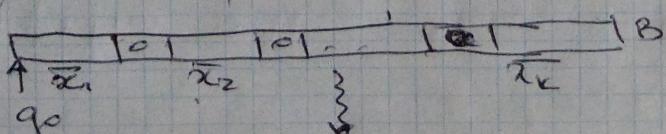


M ca dispozitiv de calcul
funcții numerale

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

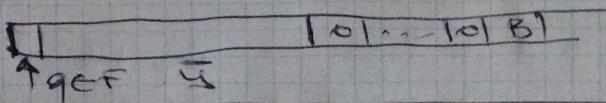
$f(x_1, \dots, x_k)$ este definită și $f(x_1, \dots, x_k) = y$

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.

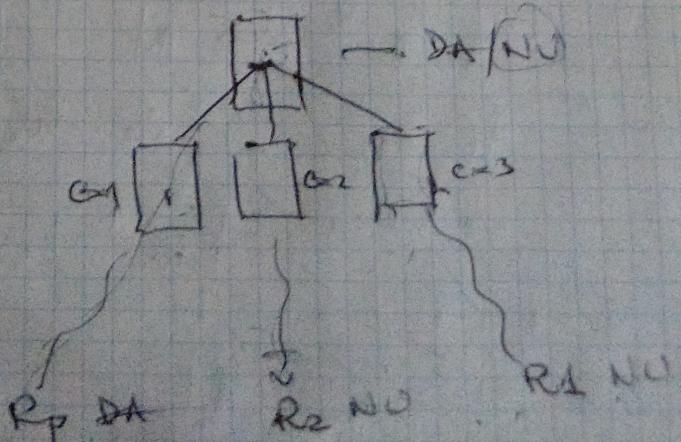
Masina Turing deterministă

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

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

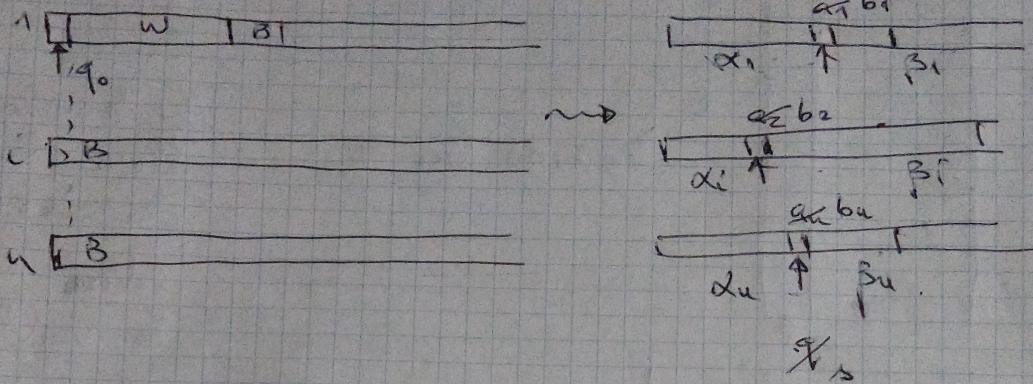
Algoritmic = o rețea deterministă care se
opreste pe fiecare intrare

MT deterministă \neq MT nedeterministă



Scop: $MTD \rightleftharpoons MTN$

Omtree n bærfi: $H = (n, Q, V, U, \delta, q_0, B, F)$



$$f: (\mathbb{Q} \times \mathbb{U}_{\mathbb{F}}^n) \longrightarrow \mathcal{G}_f(\mathbb{Q} \times \mathbb{U}^n \times \{L, R\}^n)$$

$$(s, b_1, \dots, b_n, \underbrace{x_1, \dots, x_n}_{\in \{h, R\}}) \in \delta(q, a_1, \dots, a_n)$$

Teorema Pentru orice multime finită M , există o multime determinată, cu 3 elemente, și astfel încât $L(M) = L(M')$.

$$\underline{\text{Defn}} \quad M = (Q, V, U, \delta, q_0, B, F)$$

$$H' = \left(z, Q', V, U, S', q_p, B, \tilde{\tau}' \right)$$

1

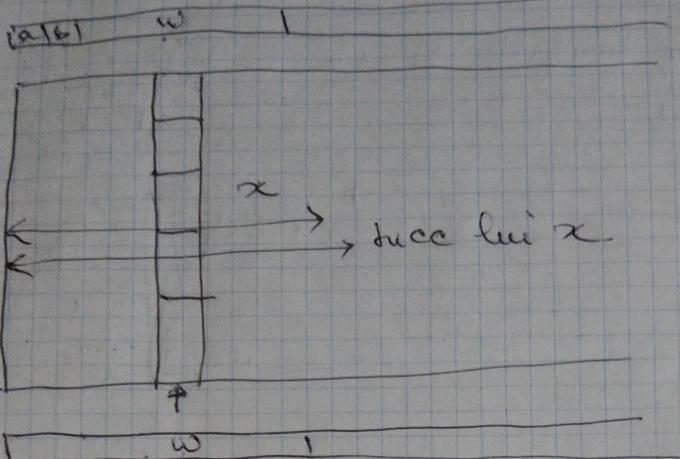
| | | | | | | |
|--|---|---|---|---|---|--|
| | a | b | c | b | b | |
| | a | b | c | b | b | |
| | a | b | c | b | b | |

Etapa 1.

Copiază și de
pe prima bordă
pe bordă 3.

$$d(q_0, (a_1, \beta)) = (q_1, a_1 -$$

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



Etapă 2.

Iguală bauda
2.

Greutatea
pe bauda 2
elementelor
succesorul
celorii curent
în vîrstă nu e
E..

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

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

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

Ticcare element al lui E va fi considerat

în stările pe bauda 2.

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

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

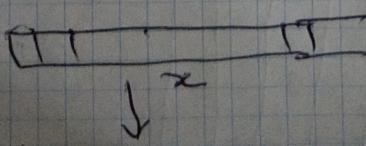
$$(S, \leq) = \{10, 5, 4, 1, 2\}$$

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

Subproblemă

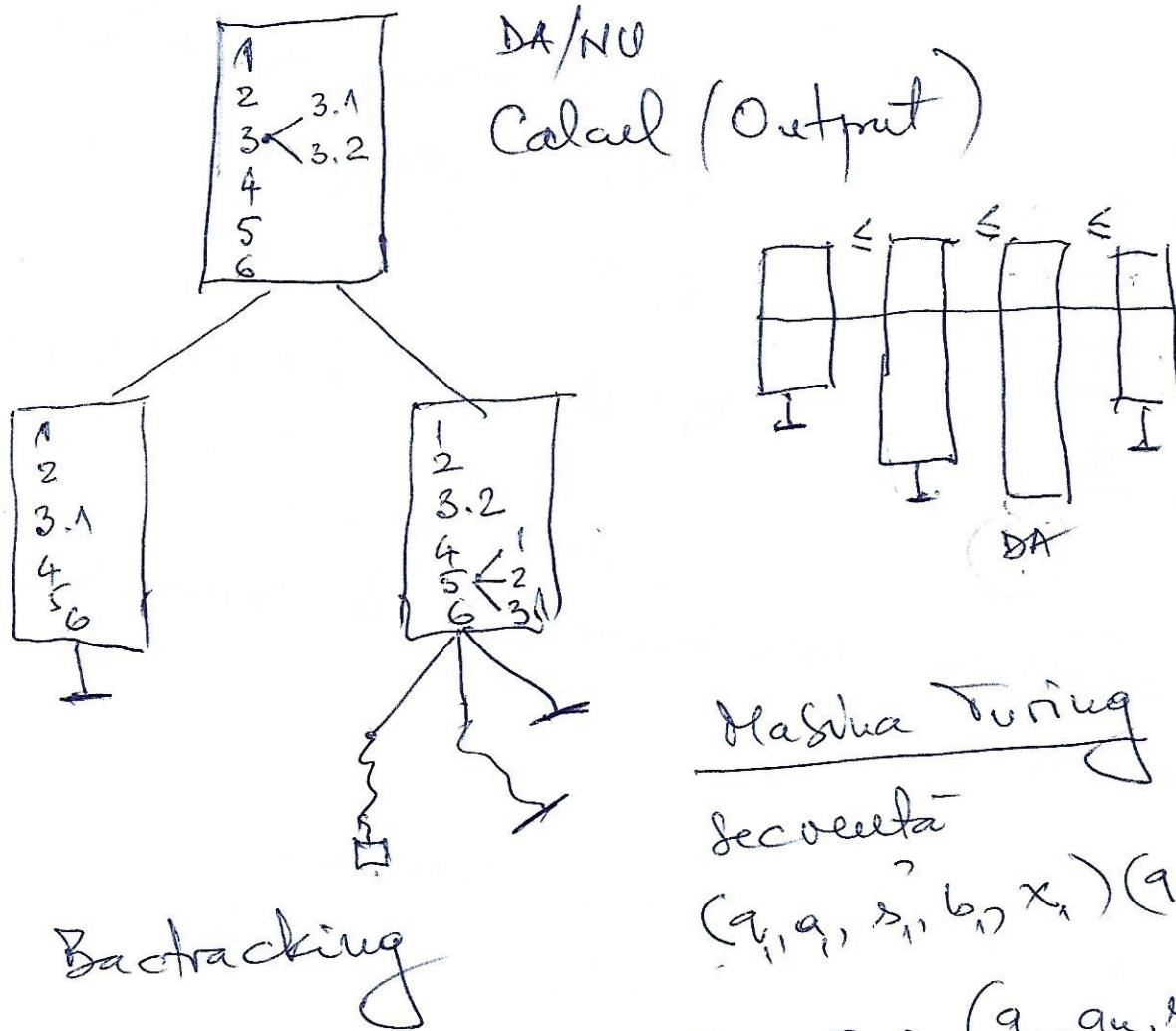
$$21x04\beta \neq$$

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



succesorul lui x

= CURS 3 =



Mashina Turing

secvență

$$(q_1, q_2, s_1, b_1, x_1) (q_2, a_2, b_2, b_2, x_2)$$

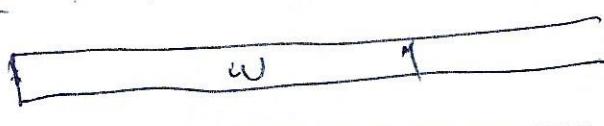
$$\dots \dots (q_n, a_n, b_n, b_n, x_n)$$

Teroarea Parțial orice set M determinanta
există o set M' determinanta cu 3 beneficii

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

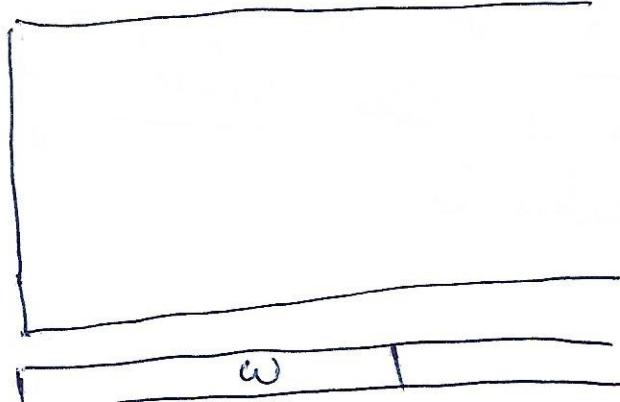
$$\text{Deu } f\text{ie } M = (Q, V, \Delta, \delta, q_0, B, F).$$

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

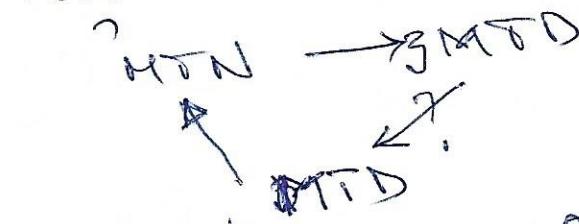
$abb \leq aaaa$

$aabba \geq aabba$

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.

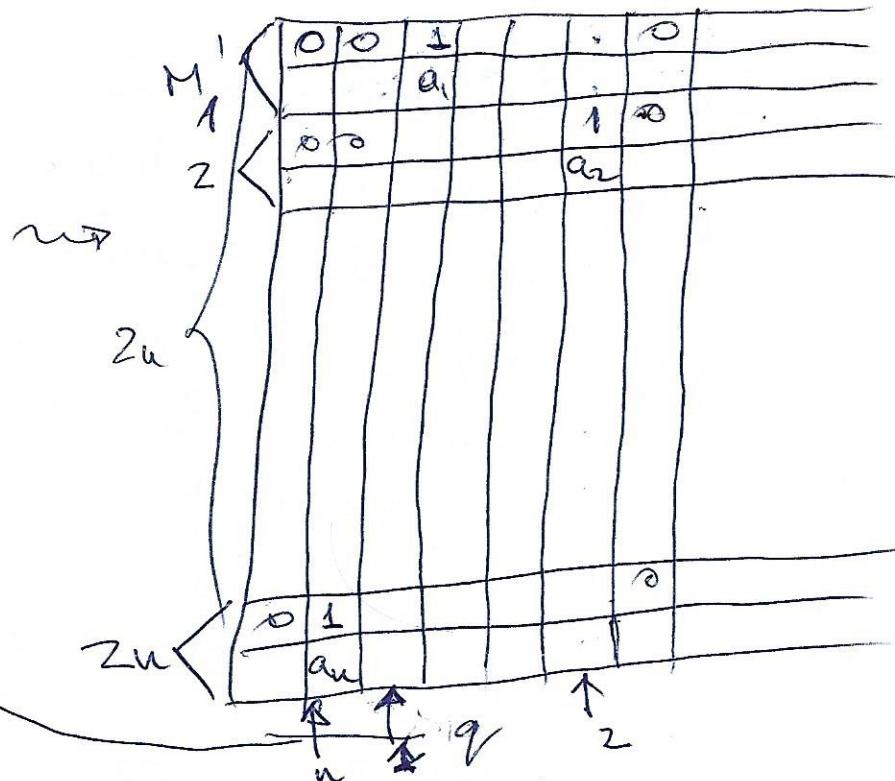
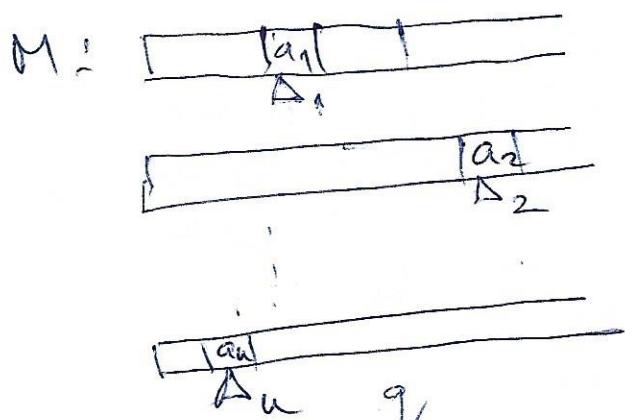
$L(M) = L(M')$.
Dacă M' este deterministă.

Dacă plus, M' este deterministă.

este deterministă.
 $\{ \}$

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

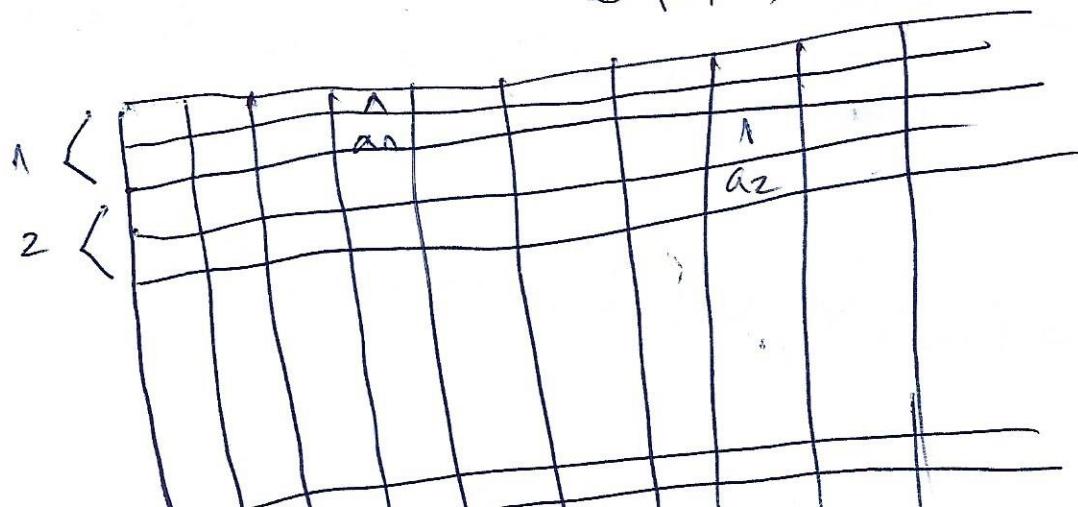
$$M' = (\alpha', v, V', \delta', q_0, B, F)$$



- 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 $C_2 \rightarrow C_2$

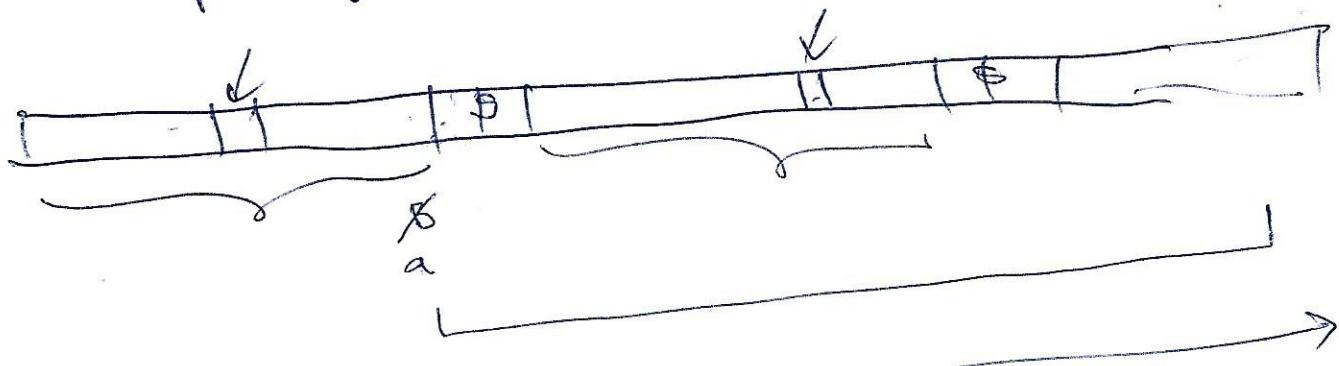
$$(a_1, b_1, \dots, a_n, b_n, x_1, \dots, x_n) = \delta(a_1, a_1, \dots, a_n)$$

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


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 .

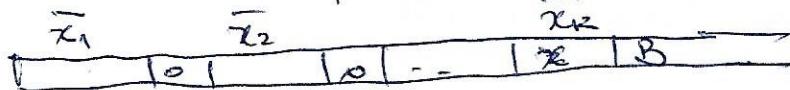
Etape 3 M' scanază banda de la stărg
și actualizează pozitia
în dreapta și actualizează pozitia
pe pistă împărțită.



= 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?

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

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

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

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

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

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

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

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

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.

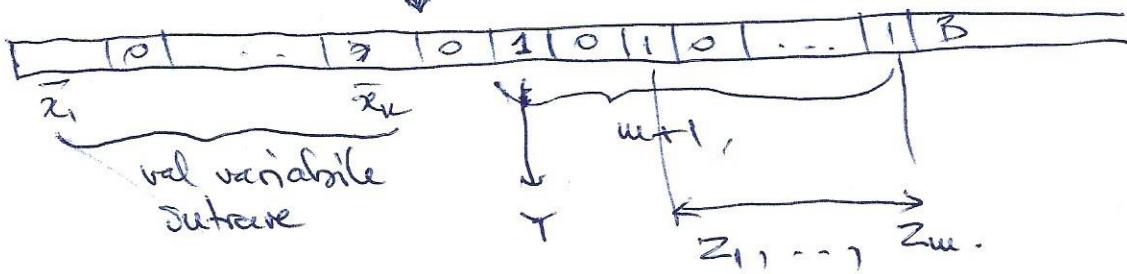
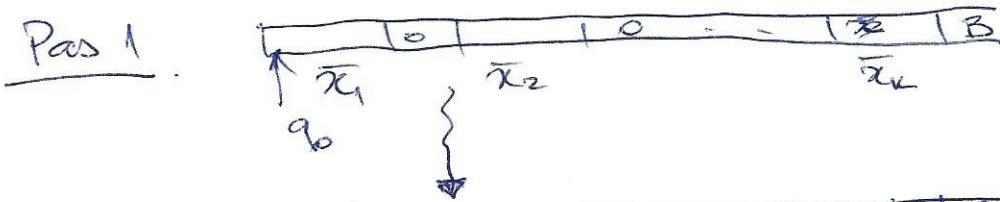
Dove. Fie $f: N^k \rightarrow N$ o funcție calculabilă ce PS.

Fie P programul standard care calculează f.

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

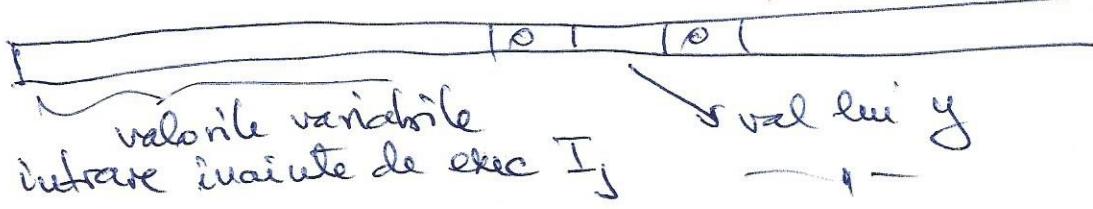
Z_1, \dots, Z_n variabile de lucru.

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 $\langle \text{val lui } Z_i \rangle$



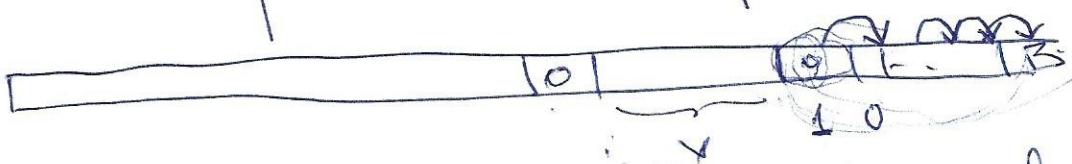
Pas 2. $I_j : V \neq V$.

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 lui V .

(ii) Deplasează la dreapta toate pozitivile de la dreapta formei corespunzătoare lui V .



(iii) ~~Rezolvă~~ Răvășează formula corespunzătoare lui V care duce la 1.

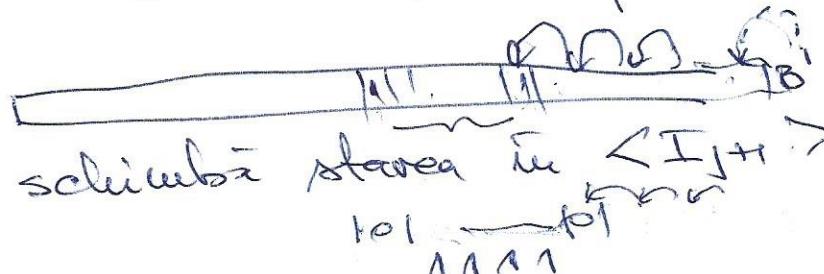
(iv) Schimbă starea în $\langle I_{j+1} \rangle$.

$I_j : V \neq V+1$.

(i) Ideea (i) anterior.

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

(iii) Altfel, șterge 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 număr de linii de instrucțiuni a I_p 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 : $Y \nleftarrow X$

$Y \leftarrow Y + 1$.

π_k : $Y \leftarrow X_k$.

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

Operări cu fracții

II Compozirea funcțională

$f \circ g : N^k \rightarrow N$ este definită prin comp. func.
a fracții $h : N^m \rightarrow N$ 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 ?

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

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

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

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

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

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

$$A_j : z_e \leftarrow z_t$$

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 nicio 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

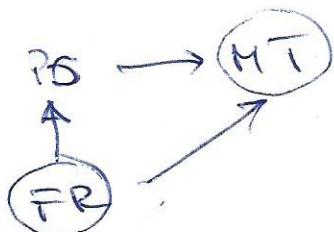
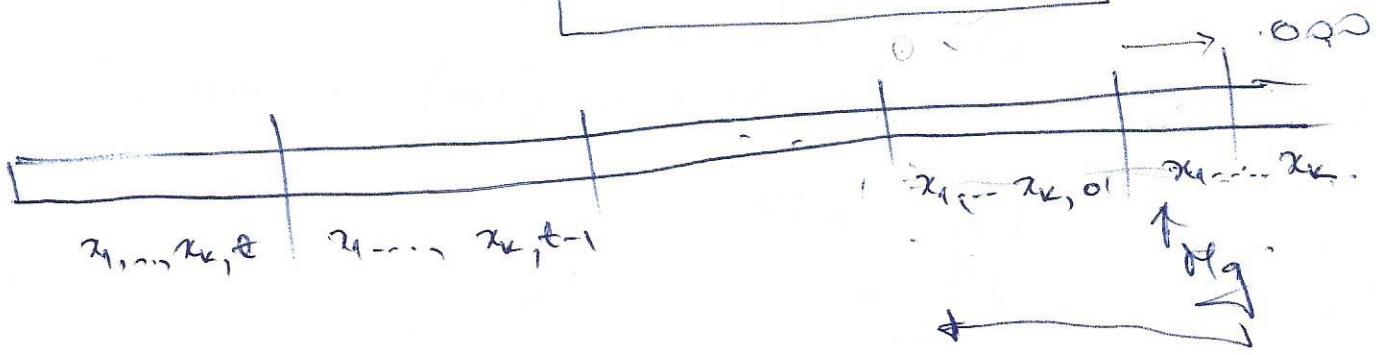
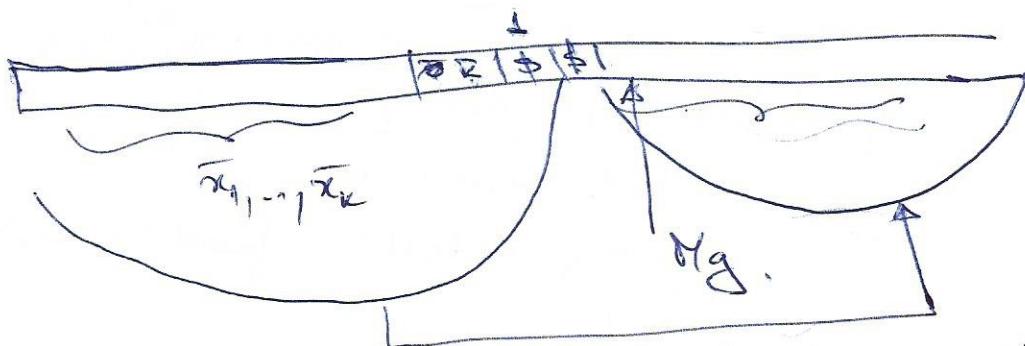
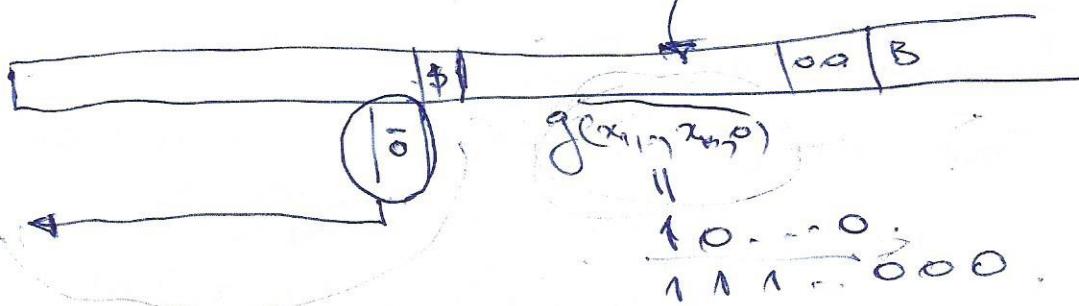
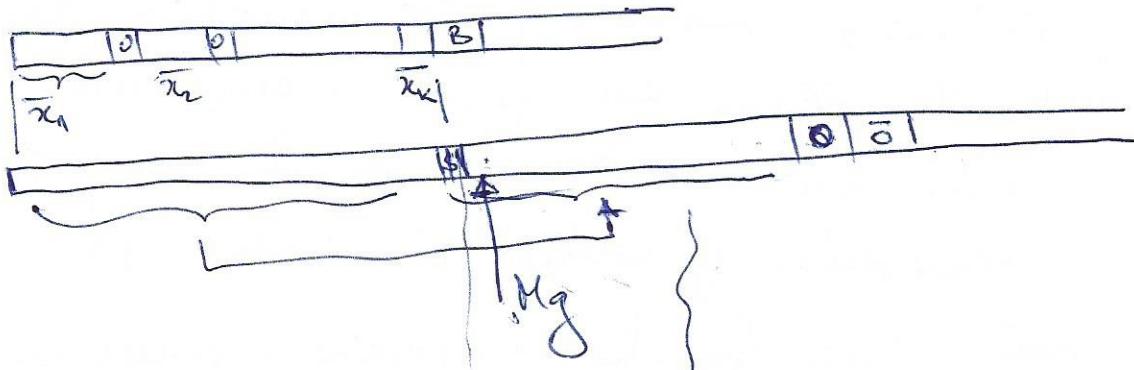


= 4 =

Codificarea PS

$P \mapsto n(P) (\#(P)) \in N$

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 , pozitia capului pe banda este 3 iar continutul benzii este: $s_1s_1s_0s_3B$, atunci numarul 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.

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:

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.

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

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

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

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

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

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

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

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

$GOTO C$

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

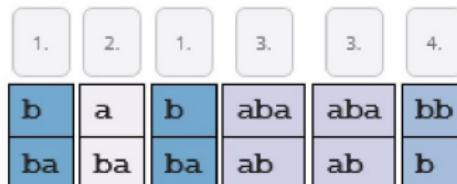
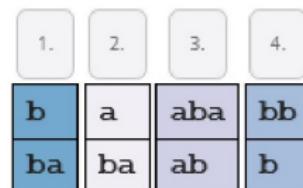
$x = (x_1, x_2, \dots, x_n)$,

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

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 daca $PCP(x, y)$ 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 daca 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.

| Listă α | Listă 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 3 =

Complexitate: măsură de complexitatea calculei

Relativ între clase.

P vs NP - reduceri

Complexitate: calculelor

Kolmogorov.

Descriptive

Măsura time (TIME)

spătar (SPACE)

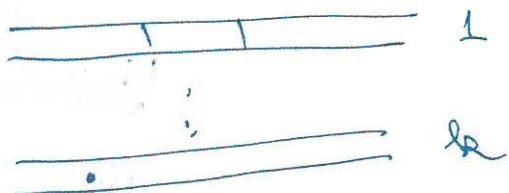
Modelul de urmă: 1. urmă de opere

2. Are la bază

inființate la

aceleși capete

3. Poate staționa



Calcul

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

$\underbrace{w_1}_{\text{intrare}} \quad \underbrace{\text{intrare}}_{\text{configuratie}}$

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

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

calcul al lui x

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

$$\text{TIME}_k(f(n)) = \{ L \mid \exists M \text{ o u.m. ce la baza a.i. } L = L(M) \text{ și } \text{time}_M(n) \leq f(n), \forall n \geq 0 \}$$

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

SPACE: Modelul:

1. urmă se oprește

(off-line) 2. Are o bandă de intrare

w intrare

disponibilită doar pentru citire

3.

citire/scris

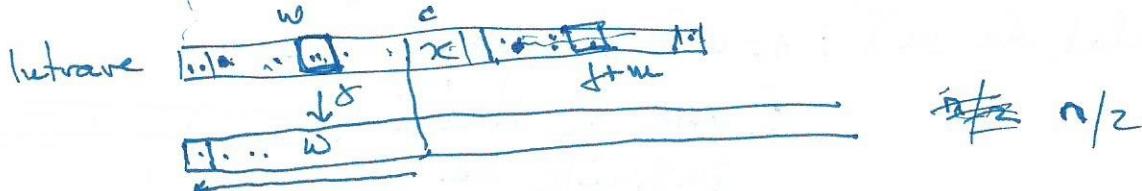
$$C(M, x) = c_1 \dots c_n$$

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

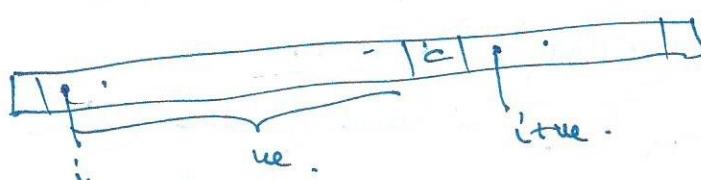
$$\text{space}_M(u) = \max \{\text{space}(M, x)\}$$

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

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



intrare w c $n/2$
 $\log n.$ $O(2^n)$.
 în totdeauna la prefixul până la c.



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

Eliminarea constantei

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

$$+ c > 0$$

Dacă aleg N . Este suficient să ducem

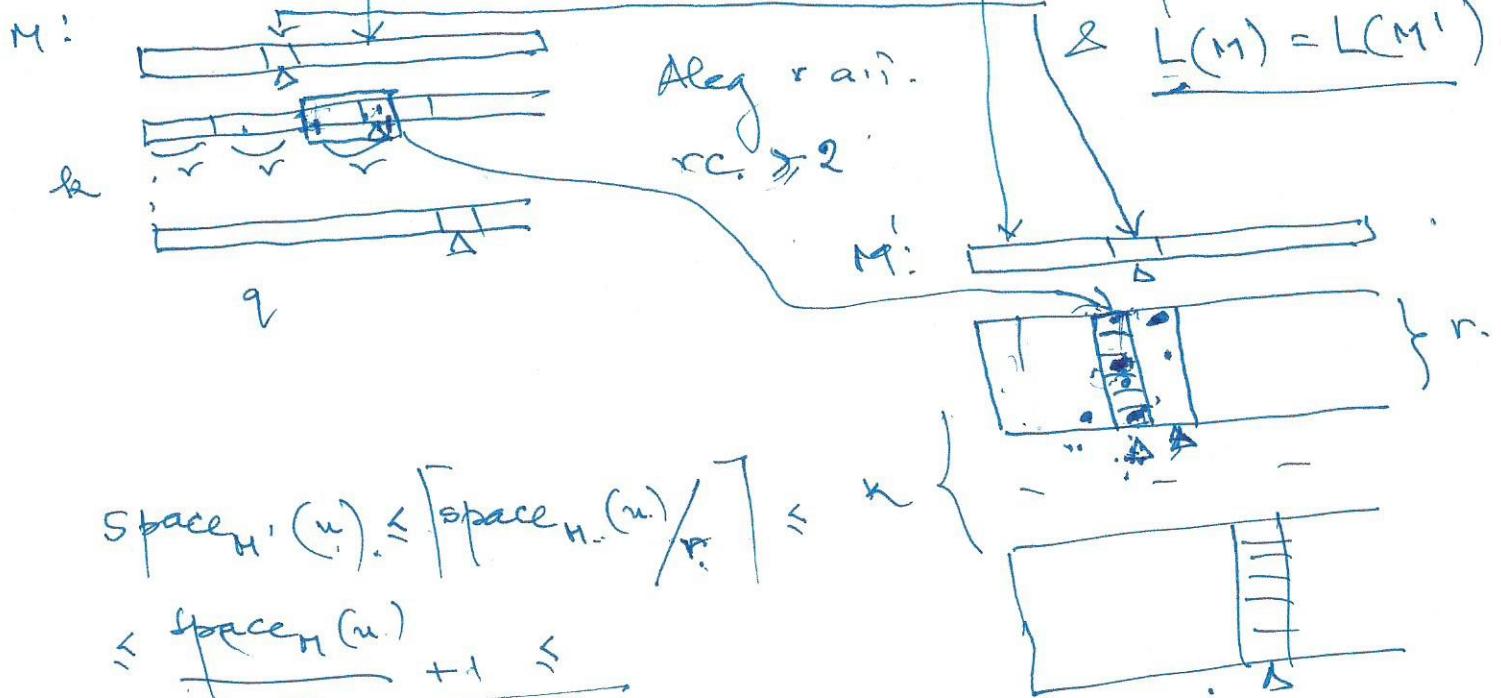
$$\textcircled{2} \quad \boxed{NSPACE_k(f(u)) = NSPACE_k(cf(u))}$$

$$0 < c < 1$$

$$c \geq 1 \quad NSPACE_k(f(u)) \leq NSPACE_k(cf(u))$$

= 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.

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

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