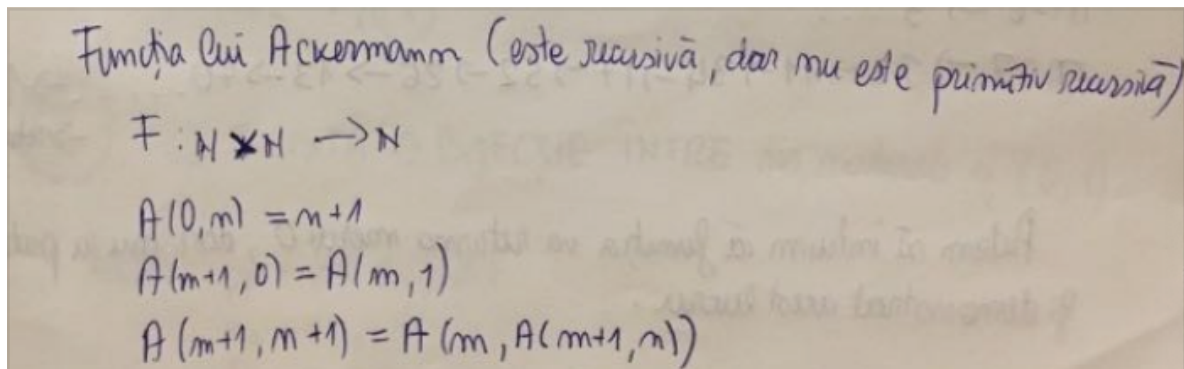


## CC



O funcție primitiv recursivă este o funcție definită din funcțiile de bază:

- Succesor
- Constanta 0
- Proiecție

și care se folosește de operații:

- Compunere
- Recursie primitivă

$f(x_1, \dots, x_n)$ ,  $h(x_1, \dots, x_{n+2})$ ,  $g(x_1, \dots, x_{n+1})$

G e recursie primitivă dacă :

- $f(x_1, \dots, x_n) = g(x_1, \dots, x_n, 0)$
- $g(x_1, \dots, x_n, y + 1) = h(x_1, \dots, x_n, y, g(x_1, \dots, x_n, y))$
-

$f(x, y) = x + y^n$  de primitivă recursivă  
 $f(x, 0) = S(x)$   
 $f(x, y+1) = S(f(x, y))$

Introducem si notiunea de partial recursiv cu operatia de minimizare.

Daca avem  $f$  cu  $n+1$  parametrii, definim  $g$  cu  $n$  parametrii care este cel mai mic  $y$  pt care  $f(x_1, \dots, x_n, y) = 0$  sau nedefinita daca  $y$  nu exista

## Masina turing

Avem  $S$  multimea de starti,  $\Sigma$  alfabetul si  $ro: S \times \Sigma \rightarrow S$  functia de tranzitie

Masina poate scrie peste simbolul pe care l-a citit si asa avem  
 $ro: S \times \Sigma \rightarrow S \times \Sigma \times \{left, right, pe\_loc\}$

Cu o masina touring cu o singura banda infinita la dreapta putem simula o masina touring infinita in ambele directii cu  $n$  benzi. Pentru asta facem alfabetul ca fiecare litera sa fie un tuplu de litere pt fiecare din cele  $n$  benzi + literele de la stanga si dreapta pozitie de inceput + offset

Exista o masina touring universală care poate simula orice masina touring given the code and input

## Functii necalculabile

Functia universală :

$$\bullet \quad U(x) = \begin{cases} 0 & \text{dacă } M_x(x) = 1 \text{ (mașina } x \text{ pe intrarea } x \text{ dă 1)} \\ 1 & \text{altfel} \end{cases}$$

$$U: \mathbb{N} \rightarrow \mathbb{N}$$

Calculz  $U(m) = M_m(m)$

$$\Rightarrow M_m \text{ calculează } U$$

$$U(m) = \begin{cases} 0 & \text{dacă } M_m(m) = 1 \\ 1 & \text{altfel} \end{cases}$$

Orice functie partial recursiva poate fi calculata de o MT (echivalenta)

Problema opririi este necalculabila

Nu avem algo pt Wang tiles (ala cu dominourile de acopera un plan) ca daca am avea am avea algo pt HALT

$S(n) \rightarrow$  nr de pasi pe care-i poate face o MT pana se opreste. nu putem calcula asta ca daca am putea am avea algo pt HALT (am stii in maxim cati pasi s-ar opri orice MT)

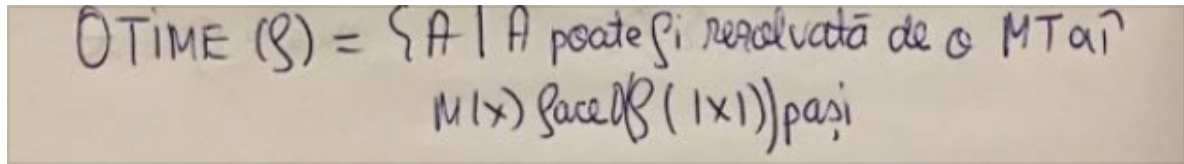
O multime este recursiva daca avem o fct  $f$  care ne zice 1/0 daca un element apartine sau nu multimii

E recursiv enumerabila daca ne zice 1 daca aprtine si nu returneaza altfel

Daca o multime este recursiv enumerabila si complementara multimii e recursiv enumerabila, atunci multimea e recursiva

Ne referim la multimi recursiv enumerabile cand ne interesaeaza doar

elementele care apartin multimii nu si cele care nu



PTIME (P) = { A | A poate fi rezolvată de o MT în  $M(x)$  pași } (1x1) pași

O problema de decizie este in clasa NP daca avem un algoritm care ne poate verifica in timp polinomial daca problema noastra accepta sau nu o solutie. Dar nu avem algoritm polinomial care sa ne dea solutiile efectiv.

O problema A se reduce la B daca avem o fct f care ne duce un x din A in B si B accepta f(x)

O problema A e NP completa daca A e NP si orice problema din NP se reduce la A

Daca A nu e din NP este NP hard

Probleme NP

- TSP
- SAT
- Colorare graf
- Sudoku

Primele 3 sunt chiar NP complete

Un automat nedeterministic poate fi simulat de unul deterministic dar cu mai multe stari

TSP si SAT sunt np complete

Se poate demonstra ca 3-SAT e np completa  
Si putem reduce SAT la 3-SAT

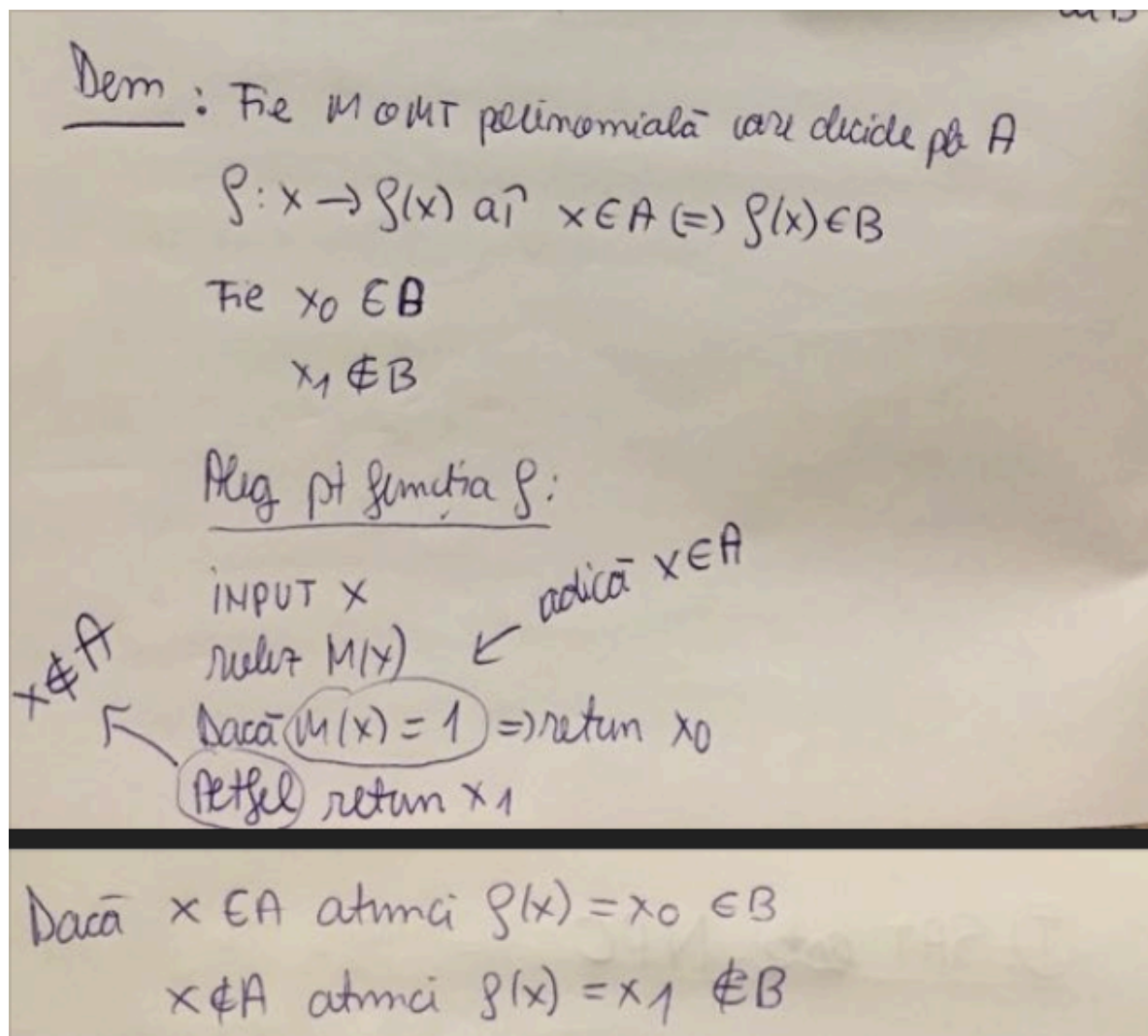
$$\begin{array}{l|l}
 \phi_0 & \mathcal{S}(\phi_0) \\
 xvy\bar{z}v^+ & xvyv\alpha \rightarrow \text{variabilă nouă} \\
 xvy\bar{z}v^+vm & \bar{x}\bar{v}\bar{z}v^+ \\
 & \hline
 & \begin{array}{l}
 xvy\bar{z}v\alpha \\
 \bar{x}v^+vm \\
 \hline
 xvyv\beta \\
 \bar{\beta}v\bar{z}v\alpha \\
 \bar{x}v^+vm
 \end{array}
 \end{array}$$

Pt o clauze cu 4 variabile bagam 2 cu cate 3 variabile. Introducem o variabila intermediara care in prima clauza e pozitiva si in a doua e negata

2-SAT e P

Integer linear programing, clique (se da un graf, daca avem sau nu miunim K noduri conectate toate intre ele), vertex cover sunt NPC

Orice problema din P o putem reduce la NP



NPC-urile se reduc unele la altele

O formula cu cat are mai multe clauze cu atat e probabilistic mai putin probabil sa fie satisfiabila

Pentru SAT avem algoritmul Davis Putnam care ne rezolva unele probleme. E un algoritm de backtracking:

- Literar pur (x apare doar pozitiv sau doar negativ -> alegem direct valoarea pt el)
- Literar unitar (x apare doar intr-o clauza -> alegem direct valoarea pt el)

Avem de asemenea si algoritmul de rezolutie. Daca stim  $C$  sau  $C_1$  si  $\neg C$  sau  $C_2$  atunci stim  $C_1$  sau  $C_2$

O multime se numeste p selectiva daca avem o functie calculabila in timp

polinomila a.i

- $f(x,y)$  = fie x fie y
- Daca x sau y aparitn A atunci  $f(x,y)$  apartine a

**Problema lui matiavelichi e recursiv enumerabila**