= Calculabilitate & Complexitate =

Calculabilitate → ce pot / nu pot să calculez

one way functions → criptografie

Ce pot / nu pot calcula?
Ce pot / nu pot calcula EFICIENT?

Aho: "Mai multe probleme decât algoritmi"

Geodor Cantor: "infinitul corespunzător problemelor e mai mare decât infinitul corespunzător algoritmilor"

În finit: $|A| = |B|$ dc. $(\exists) f: A \to B$ bijectivă

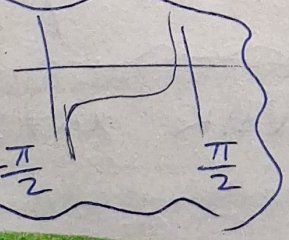
ex: $f: \mathbb{N} \to 2\mathbb{N}$
$f(x) = 2x$ $\Rightarrow |\mathbb{N}| = |2\mathbb{N}|$

$\aleph_0$ = cardinalul $\mathbb{N}$

Teoremă: $|\text{Programe}| = |\mathbb{N}|$

Orice program → string → nr. nat. în baza (de exp) 256

$\mathbb{R}: (0,1)$
$|\mathbb{R}| = |(0,1)|$



Teoremă (Cantor): Nu există o bijecție $f: \mathbb{N} \to (0,1)$
$\mathbb{R}$

(Sunt mai multe fct. de calculat decât programe care să le calculeze)

Dem: Pp prin red. la absurd că $(\exists) f: \mathbb{N} \to (0,1)$

$x_1, \ldots, x_n, \ldots$

0. $0,\boxed{a_{01}}a_{02} \ldots a_{0n} \ldots$
1. $0,a_{11}\boxed{a_{12}} \ldots a_{1n} \ldots$
2. $0,a_{21}a_{22}\boxed{} \ldots a_{2n} \ldots$

$x = 0,b_1 b_2 \ldots$
$b_1 \neq a_{01}, 9$
$b_2 \neq a_{12}, 9$

} x nu e în listă pt că diferă de ($\forall$) elem din listă măcar printr-o poziție

Concluzie $|\{f: \mathbb{N} \to \{0, \dots, 9\}\}| > |\mathbb{N}|$ ← programele

$\{f: \mathbb{N} \to \mathbb{N}\}$

Pb: String x → $\square$ → String $f(x)$

Ex. pb care NU are algoritm

INPUT: $p(x_1, \dots, x_n)$ cu coeficienți întregi

OUTPUT: Da / Nu Ec. $p(x_1, \dots, x_n) = 0$ are / nu are sol întregi?

Teoremă (Matiyasevic): Pb. de mai sus NU are algoritm

Fct. Ackermann

$$A(0, n) = n+1$$
$$A(m+1, 0) = A(m, 1)$$
$$A(m+1, n+1) = A(m, A(m+1, n))$$

pot calcula, dar, de exemplu, $A(5, 10)$ e imens

Ex: $x_0 = n$

$$x_{k+1} = \begin{cases} x_k/2 & \text{dc } x_k \text{ - par} \\ 3x_k + 1 & \text{dc } x_k \text{ - impar} \end{cases}$$

$x_0 = 8 \to 4 \to 2 \to 1$

Conjectura lui Collatz: $(\forall) n \ (\exists) k \quad x_n = 1$

("Toate descresc. la 1")

Ex: Șirul lui Goodstein

n exp 4 $4 = 2^2$

5 $5 = 2^2 + 2^0$

$\downarrow$

$3^3 + 3^0 - 1 = 27$

$27 \to 4$

$\downarrow$

$5 - 1$

Teoremă $(\forall) n (\exists) k \quad G(n, k) = 1$

Clarificări curs 1

Conjectura lui Collatz

- pornesc de la un nr intreg -> x
- dacă x e par -> $x' = x/2$
- dacă x e impar -> $x' = 3x+1$

La final, intotdeauna se ajunge la valoarea 1.
(mai au semn de interrogare, doar n-am găsit contraexemple)

Șirul lui Goodstein

→ ordinary base-n notation (n -> sumă de n^k, $k \in \mathbb{N}$)

$$\boxed{n = a_k n^k + a_{k-1} n^{k-1} + \dots + a_0}$$, $n > 1$, $n \in \mathbb{N}$, $0 \le a_i < n$, $a_k \neq 0$

ex: $35 = 32 + 2 + 1 = 2^5 + 2^1 + 2^0$ (baza 2)
$100 = 81 + 18 + 1 = 3^4 + 2 \cdot 3^2 + 3^0$ (baza 3)

→ hereditary base-n notation

- pas 1 -> se scrie și exponenții ca sumă de puteri ale lui n
- pas unu -> fac asta până când totul e scris în baza n (ignorăm coeficienții $0 \le coef < n$)

ex: $35 = 2^{2^1+1} + 2^1 + 1$
$100 = 3^{3^1+1} + 2 \cdot 3^2 + 1$

→ $\boxed{\text{Goodstein sequence } G(n)}$ pt un nr n

- primul elem e n
- al doilea e n scris în hereditary base-2 not.
- modif 2 cu 3 și scade 1

General: -> se ia hereditary base-(n+1) repr a lui G(n)(n)
inloc fiec apariție a lui base-(n+1) cu (n+2)
scade 1
continuă până ajungi la 0

ex: $3 \xrightarrow{\text{baza 2}} 2^1+1 \xrightarrow{\text{baza 3}} 3^1+1-1=3^1 \xrightarrow{\text{baza 4}} 4^1-1=3$

baza 5 → eram la baza 4 (nu apare 4) doar scad 1 → $3-1=2 \xrightarrow{\text{baza 6}} 2-1=1$

baza 7 → $1-1=0$

Baza	Hereditary notation	Value	Notes
$\boxed{2}$	$\boxed{2}+1$	3	Scrim 3 în baza-2 not.
$\boxed{3}$	$\boxed{3}+1-1=3^1$	3	$\boxed{2}$ devine $\boxed{3}$, scad 1
$\boxed{4}$	$\boxed{4}-1=3$	3	$\boxed{3}$ devine $\boxed{4}$, scad 1
5	$3-1=2$	2	nu apare 4 ce să-l schimb cu 5, scad 1
6	$2-1=1$	1	— 5 — 6 —
7	$1-1=0$	0	— 6 — 7 —

Se presupune că (∀) nr transf. în zin Goodstein ajunge într-un final la 0.

Asta scrie Wikipedia, profu zice că ajunge la 1.

Nu am idee, anyway ce să ajung la 0 trec prin 1.

Curs 2 (Fcț. calculabile în principiu)

Program $\xrightarrow{\text{calc}} f: \Sigma^* \to \Sigma^*$

de ex. nr în baza 10 → $\Sigma^* = \{0, \dots, 9\}$

$\Sigma \leftrightarrow$ alfabet

$\Updownarrow$

$f: \mathbb{N} \to \mathbb{N}$

$\Updownarrow$

$f: \mathbb{N}^2 \to \mathbb{N}$ $\quad \langle , \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$

Concluzia, există fcț $f: \mathbb{N} \to \mathbb{N}$ care NU sunt calculabile

T. lui Hatmarevich (nu are alg. rezol. pb)

INPUT: $p(x_1, \dots, x_n)$ cu coef. întregi

DE DECIS: Are ec $p(x_1, \dots, x_n)=0$ sol. întregi sau nu?

$$f(n) = \begin{cases} 1, & \text{ec } p(x_1, \dots, x_n)=0 \text{ are sol} \\ 0, & \text{altfel} \end{cases}$$

Pb de decizie

INPUT: $x \in \Sigma^*$
OUTPUT: Da/Nu

nr finit de inputuri

$$L = \{x \in \Sigma^* \mid \text{răsp e DA}\}$$

Vreau ceva care să meargă în general (are răsp pt o inf. de intrebări)

Calc fct $\equiv$ Pb de decizie (echivalent)

$f: \mathbb{N} \to \mathbb{N}$ $\quad n = \langle x, y \rangle$

De decis dc $f(x) = y$?

Prima incercare $\longrightarrow$ FCT. PRIMITIV RECURSIVE

Rez. matematice pot duce la lucruri paradoxale

Hilbert $\longrightarrow$ "Procedură mecanică" pt. verificarea adevărului unei prop. matematice arbitrare?

Răsp: NU

Arg I $\Rightarrow$ există pb. matematice care într-un anumit sens sunt adevărate, dar nu au o demonstrație (Gödel) adevărat $\neq$ demonstrabil

(Nu toate posibile prop. e adev)

Arg II $\Rightarrow$ Nu orice pb. are algoritm (Turing)

FCT. PRIMITIV RECURSIVE (și de teorie axiomatică)

↳ plec de la fct de bază + op cu fct de bază
↳ ajung la ceva mai complex.

Fct. de bază

$f(n) = 0$

$S(n) = n+1$ (succesor)

$P_i^n(x_1, \dots, x_n) = x_i$ (proiecția)

Op cu fct de bază → recursie primitivă (program care se autoapelează)

↓

compunere

$f: N^{k} \to N$

$g_1, \dots, g_k: N^{n} \to N$

$f \circ (g_1, \dots, g_k): N^{n} \to N$

recursie primitivă

$f: N^{n} \to N$

$g: N^{n+2} \to N$

$h: N^{n+1} \to N$

$h(x_1, \dots, x_n, 0) = f(x_1, \dots, x_n)$

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

Def $f: N^{*} \to N$ sn. **primitiv recursivă** dacă poate fi obținută din **fct de bază** prin **compunere** și **recursie primitivă**

OBS: (+) fct **primitiv recursivă** e **intuitiv calculabilă**

ex: fct Ackermann

$A(0,n) = n+1$

$A(n+1,0) = A(n,1)$

$A(n+1,n+1) = A(n, A(n+1,n))$

rec. primitivă?

↳ NU (aplic ac recursie asupra ei) g din def → distinctă de h aici am A & A

O pot calcula intuitiv (și pot scrie program)

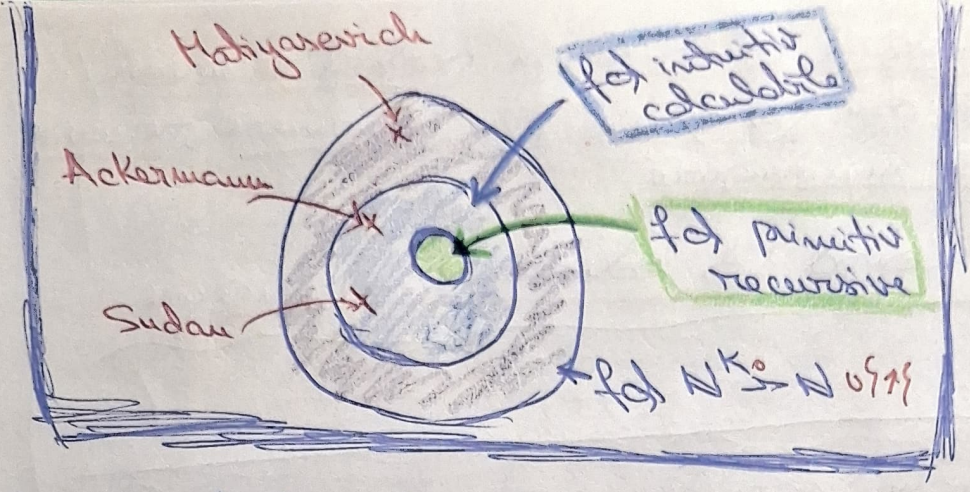
DAR TOT NU E primitiv recursivă

Teoremă: fct Ackermann **nu** e primitiv recursivă (crește prea rapid)

extra:

Hilbert → Ackermann + Gabriel Sudan → ex. de fct care au fost ulterior dem că nu sunt primitiv recursive

au dat exemple de fct (dar nu au și demonstrat că NU sunt primitiv recursive pt că nu exista noțiunea atunci



← operă de artă de dus la o expoziție pls.

<u>intuiție</u>: (Pot implementa op. de recursie primitivă cu un "for" loop)

↳ fct primitiv recursive → FOR, nu WHILE

<u>intuiție</u>: Trb să admit în modelul meu posibilitatea de fct care uneori nu ia valori

$\boxed{f(x) = \uparrow}$ ← f(x) nu ia valori (nedefinită)

$\boxed{f: N^k \to N}$ ← nu pe orice intrare ret valori

<u>ex</u>: de fct intuitiv calculabilă care NU e primitiv recursivă

<u>Pass</u>: enumăr toate fct primitiv recursive

$f_1, \dots, f_n, \dots$

$n \xrightarrow{\text{Program}} f_n$

$g(n) = f_n(n) + 1$ → intuitiv calculabil

g primitiv recursivă? NU

↙

$g \neq f_n \ (\forall) n$

(niacor pe input-ul n)

<u>Def</u>: $\boxed{\text{Minimizare}}$ (Pot să scrim μy)

$\boxed{g(x_1, \dots, x_n) = \mu y [f(x_1, \dots, x_n, y) = 0]}$

(cel mai mic y → poate nici să nu existe)

! f trb să fie $\boxed{\text{fct totală}}$ (bine def. peste tot)

Def O fct parțială $f: \mathbb{N}^k \overset{\circ}{\to} \mathbb{N}$ se. parțial recursivă dc o pot obț din fct de bază prin compunere, recursie primitivă & minimizare

Sunt consid. fct. calculabile în principiu.

Curs 3

Primitiv recursive

- Fct de bază
$$\begin{cases} f_0(x) = 0 \\ S(x) = x+1 \\ P_i^n(x_1, \dots, x_n) = x_i \end{cases}$$

- Op
  1. compunere
  $$\frac{f: \mathbb{N}^k \to \mathbb{N} \quad g_1, \dots, g_k: \mathbb{N}^n \to \mathbb{N}}{f \circ (g_1, \dots, g_k): \mathbb{N}^n \to \mathbb{N}}$$
  2. recursie primitivă
  $$\frac{g, h}{f \text{ a.î. } f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n) \quad f(x_1, \dots, x_n, y+1) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))}$$

Ackermann $\boxed{nu}$ e primitiv recursivă pt că se autoapelează de prea multe ori. (Curs)

Fct parțial recursive → fct de bază
→ op de bază → compunere, recursie primitivă, minimizare

Minimizare

$f(x_1, \dots, x_n, z)$ totală (se oprește întotdeauna)

$g(x_1, \dots, x_n) = \mu y$ (cel mai mic y nat)

a.î. $\{f(x_1, \dots, x_n, y) = 0\}$

Teza lui Church

Fct intuitiv calculab ≡ fct partial recursive

(Orice putem calcula în principiu, coincide cu conceptul de fct partial recursivă)

Fct intuitiv calculabile

→ mașini Turing, λ-calc, tag systems
→ fct calculab în Lisp
fct calculab în Python
→ automate celulare (au o mulțime de celule, fiecare celulă poate fi într-un nr de stări, fiec celulă evoluează după o anumită regulă; val vecinilor și a mea influențează val la momentul t+1)

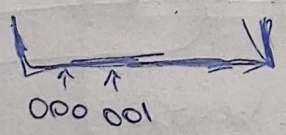
Regula 110

Game of Life

Paranteză

Automatul 110 (Regula) → în interiorul lui pot calcula (∀) fct partial recursivă

- scrii 110 în binar pe 8 biți



- NetLogo (pt vizualizare)

Game of Life → în int. lui pot calcula (∀) fct partial recursive

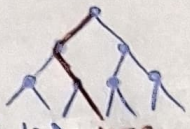
Accidentally Turing Complete (NU erau menite să calc. (∀) fct calculabile)

1. Templates în C++
2. Generics în Java
3. HTML3 + CSS
4. PowerPoint (e Turing Complete dc utilizezi face "slide")

Quantum Computers

Classic: $Pr\{YES\} = \sum$ amplitudinea YES

$\left(\frac{1}{2^n}\right)$



Quantic: amplitudinea $\to \mathbb{C}$

$$\boxed{Pr = \| \sum \text{amplitudinea} \|^2}$$

Acă parțial rec

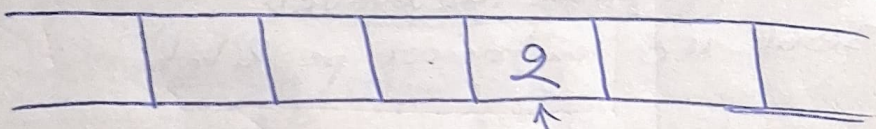
$7 = \mathbb{C}$

Un model mai puternic (Hyper computations)

(Timpul scade exponențial → primul pas 1 sec, al doilea 1/2 secunde, al treilea 1/4 sec)

Mașini Turing

- JFLAP (software, perm. simularea mas Turing și a automatelor finite)



Maz. Turing cu o bandă

$\boxed{\Sigma}$ alfabet al benzii

$\boxed{\sqcup}$ (sau $\square$) "blank"

$\boxed{S}$ stări

$\boxed{S_0}$ - stare init

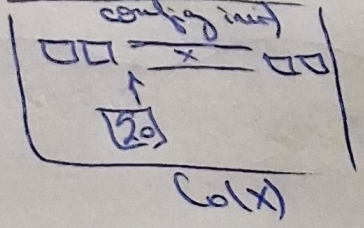
$\boxed{F \subseteq S}$ stări finale

$\boxed{\delta : \Sigma \times S \to \Sigma \times S \times \{\overrightarrow{\leftarrow}\}}$ fct de tranziție

1936/37 → a fost dem. că $\boxed{\exists}$ pb care nu pot fi rez cu o maz Turing (pb. cu oprirea)

$$\boxed{L(M) = \{x \in \Sigma^* \mid C_0(x) \xrightarrow{*} __ \to C_n(x) \text{ stare fin}\}}$$

snapshot (config)



$C_0(x) \to C_1 \to C_2 \dots \to C_k$ în care $S_k \in F$

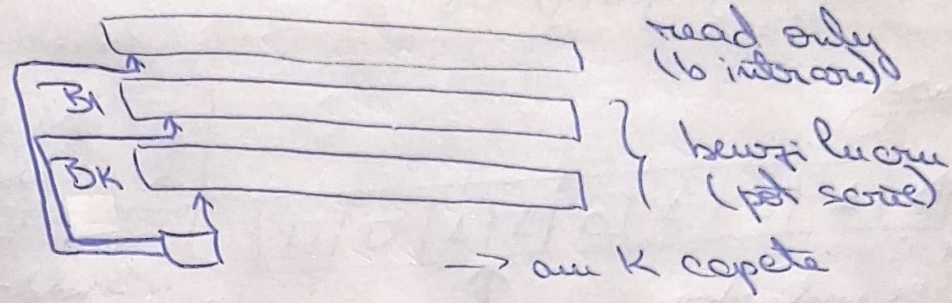
$\to$ STOP

Nu ajl la $S_k \in F$ $\Rightarrow$ maş poate rula la infinit

OBS: dc maş Turing are o bandă, ar trebui să îi dau voie să modifice input-ul.

Variante

① Maş. Turing multibandă



$$\delta : S \times \Sigma^{K+1} \to S \times \Sigma^{K} \times \{\rightarrow, \leftarrow, -\}^{K+1}$$

② Maş. Turing multibandă cu output
(una din benzile de lucru → bandă de ieşire)

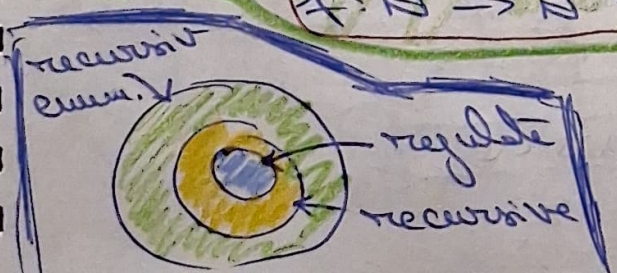
$f(x) = y$ $\quad C_0(x) \quad \dots \quad C_k(x)$

$\overline{y}$

Def: O mult. $A \subseteq \Sigma^*$ se numeşte recursiv enumerabilă (" computabilă enumerabilă) dc ($\exists$) maş Turing M a.î. $M(x) = \begin{cases} 1, & x \in A \\ \uparrow, & x \notin A \end{cases}$

Def: O mult. A se numeşte recursivă dc ($\exists$) maş Turing M a.î. $M(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}$

$$f : \mathbb{N} \to \mathbb{N} \equiv M_f = \{(x,y) \mid f(x) = y\}$$



Curs 4

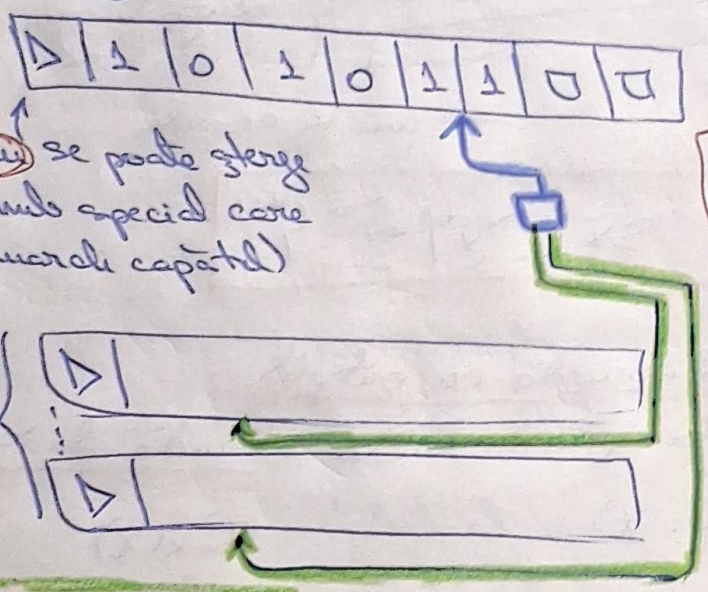
Fct partial rec = fct intuitiv calculabile
M. Turing

- fct de bază
- op. compuse
  - recursie primitivă
  - minimizare

$g(\vec{x}, y)$ totală

$h(\vec{x}) = \mu y[g(\vec{x}, y) = 0]$

M. Turing



(nu) se poate șterge (simbol special care marchează capătul)

Σ - alfabet limit
$\square$ - blank
S - stări
S_{init} - stări init
S_{hold} - stări accept

$\delta : S \times \Sigma^k \to S \times \Sigma^{k+1} \times \{\rightarrow, \leftarrow, -\}^k$

benzi de lucru

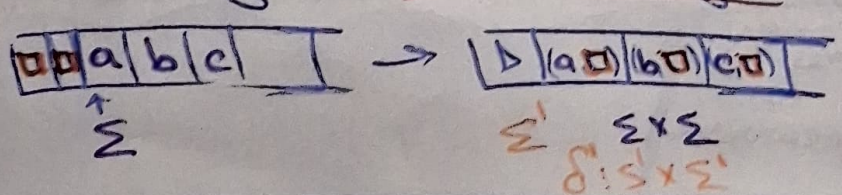
Teoremă O fct e calculabilă de o M. Turing ⟺ e partial rec.

→ Cum se calc. o fct cu o maș Turing?
→ Se consid o bandă de ieșire și outputul fct e scris pe aceasta.

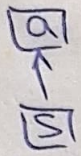
OBS. Singurul model de M. Turing care nu calc fct partial recursive e cu o banda read-only. În rest sunt toate echivalente.

(1) Cum simulez o M. Turing cu o bandă inf în ambele direcții cu o M. Turing cu o bandă semi-infinită?

M. T. 1.



② Cum simulez o M. Turing cu $|\Sigma|>2$ cu o M. Turing cu $|\Sigma|=2$



$\delta(s,a) = (s', a', \overset{\rightarrow}{\underset{\leftarrow}{-}})$

Idee: reprez. ac Σ în binar drept un cuv $w \in \{0,1\}^*$

ex: $\square \to 00$, $b \to 01$, $a \to 10$, $b \to 11$

| 0 | 1 | 0 |

string "a"

$(s,0) \to ((s,0),0,\rightarrow)$

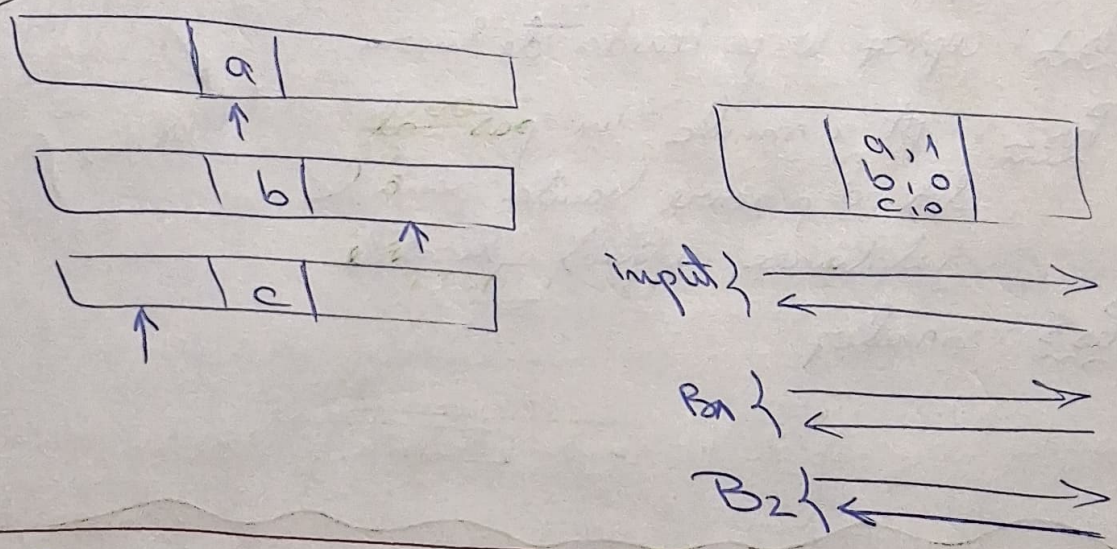
$\rightarrow\rightarrow\rightarrow ((s,01),1,\rightarrow)$

$\leftarrow\leftarrow\leftarrow ((s,01),0) \to ((s,010),0,\leftarrow)$

③ Cum simulez o M. Turing M cu k benzi cu o M. Turing M' cu o bandă.

$M(x)$ n paşi $\qquad M'(x')$ $\qquad O(n^2)$ paşi

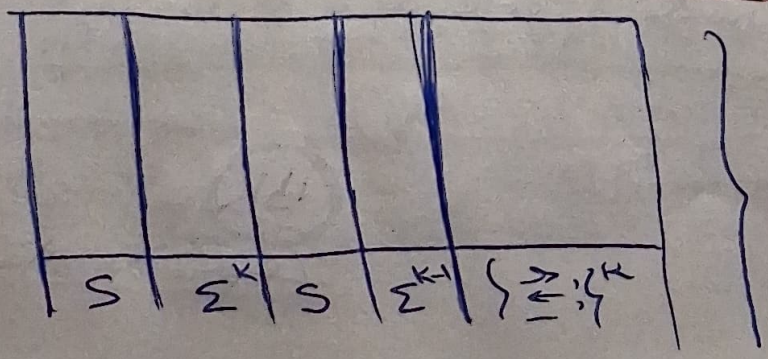
Idee:



Concl: Modelul precis de M. Turing nu contă (Toate pot. calc. exact fct. partial recursive)

M. Turing UNIVERSALA

$$\delta: S \times \Sigma^k \to S \times \Sigma^{k-1} \times \{\overset{\rightarrow}{\underset{\leftarrow}{-}}, ?\}^k$$



$\Rightarrow$ cuv. binar

$\#\#l_1\#l_2\ldots\#l_n\#\#$

Mașina $M_w(x)$ = mas cu progr. w pe intrarea x

$M_1, M_2, \dots, M_n, \dots$ toate MT.

MT Universală U

INPUT: $y = w\#x$ sau $y = \langle w, x \rangle$

$U(y)$ simulează funcționarea lui $M_w(x)$

UNIVERSAL ≡ INTERPRETOR

Teoremă: $(\exists)$ M. Turing Universală U. $\boxed{U(\langle w,x \rangle)}$ simulează T pași ai lui $\boxed{M_w(x)}$ în $\boxed{O(T\log(T))}$ pași.

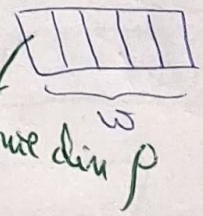
Idea: $w\#x$ - INPUT

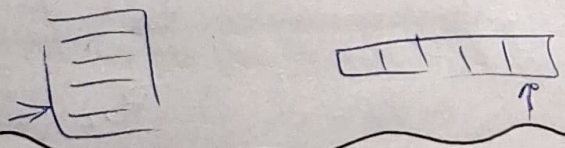
▷	w	#	x

Pas 1: copiez w pe banda de lucru

Pas 2: pt a simula un pas al fct δ parcurg banda până la întâlnirea instrucț aplicabile

Pas 3: simulez





ex: de fct recale de o M. Turing

$$w(x) = \begin{cases} 0, & \text{de } M_x(x) = 1 \\ 1, & \text{altfel} \end{cases}$$

→ simulez fct MT Universală $U(x\#x)$

Teoremă: Fct w $\boxed{NU}$ poate fi calc de o M. Turing

	λ	0	1	00	01	...
M_1						
⋮						
M_n			$\boxed{01}$			

Tp. RA w cod de o M. Turing $\Leftrightarrow w(x) = M_z(x)$ $(\forall) x$

$w \to M_z$

$$w(z) = \begin{cases} 0, & \text{dc } M_z(x) = 1 \\ 1, & \text{dc } M_z(x) = 0 \end{cases}$$

$$w(z) = \begin{cases} 0, & \text{dc } w(z) = 1 \\ 1, & \text{dc } w(z) = 0 \end{cases}$$

Curs 5

$\boxed{\text{fct intuitiv calculabile}} \equiv \boxed{\text{fct parțial recursive}} \equiv \boxed{\text{fct calculabile de M. Turing}}$

M. Turing Universală

$U(x)$ | $x = \langle i, w \rangle$ ← input; i ← codul unei MT

simula $M_i(w)$

Fct necalculabile (de MT)

$w : \mathbb{N} \to \{0,1\}$ Vreau ca $L(w) \neq L(M_i)$ $(\forall) i$

	0	1	2	3	—
M_0	□				
M_1		□			
M_2			□		
⋮					

$$w(x) = \begin{cases} 0, & \text{dc } M_x(x) = 1 \\ 1, & \text{altfel} \end{cases}$$

Teoremă: Fct w nu poate fi cod de o M. Turing M_x

$w(x) \neq M_x(x)$

Pb. opririi

INPUT: i, x

De Decis: Se oprește $M_i(x)$ after un nr. finit de pași sau nu

$K = \{ \langle i, x \rangle \mid M_i(x) = 1 \text{ (se oprește)} \}$

$$\text{HALT}(\langle i, x \rangle) = \begin{cases} 1, & \text{dc } M_i(x) \downarrow \\ 0, & \text{altfel} \end{cases}$$

Teoremă: Fct HALT $\boxed{\text{nu}}$ e parțial recursivă.

Dem: Pp că (∃) MTuring M_{HALT} care calc HALT

O să creez o M. Turing care fol M_{HALT} ca subrutină și vreau să calc $w \to M_w$.

$$M_w(\alpha) = \begin{cases} \text{input } \alpha \\ \underbrace{\text{simulez HALT}(\langle \alpha, \alpha \rangle)}_{M_{HALT}} \to 0/1 \end{cases}$$

dc HALT$(\langle \alpha, \alpha \rangle) = 0$ $\boxed{\text{ret } 1}$

altfel $\boxed{\text{ret } 0}$

NU pot calc corectitudinea unui program arbitrar cu un algoritm.

Met. formale

Model Checking
RunTimeVerification } corectitud anumite programe

↑ IN PRACTICĂ

"DA" pe majoritatea inputurilor (Hamrins, Miasnikov) → depinde modul de calc.

↑ IN TEORIE

Exemplu "concret" de pb. $\boxed{\text{nedecidabilă}}$ → pb de decizie care nu poate fi rez de MT

↓

Pavaje Wang (1961) (Echivalente: prese puzzle, tangram)

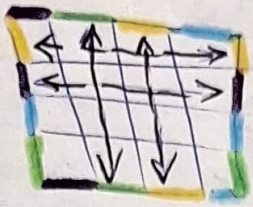
Se dau: Nr finit de template-uri $T_1, \ldots, T_k$ (pe fiec latură adiacentă, am o altă culoare)

Se decide: Pot pava tot planul cu $T_1, \ldots, T_k$ a.î. latura pe care se suprapun (două pătrate) are ac culoare

Teoremă: Pb de decizie privind pavaje Wang $\boxed{\text{NU}}$ e decidab.

Acoperire periodică

pătrat →



Wang credea: Dc $T_1, \dots, T_k$ acoperă planul $\Rightarrow$ $T_1, \dots, T_k$ acoperă periodic

$\Downarrow$

pb acoperire decidabilă

GREȘIT

Berger (1965) → există acoperiri neperiodice (~20000 modele)

Nr min de template-uri care au acop. neperiodică

↓

KNUTH (98) et al

↓

CULIK (19) → Grandal Raj (11) OPTIM

TILINGS (GRUNBAUM SHEPHARD)

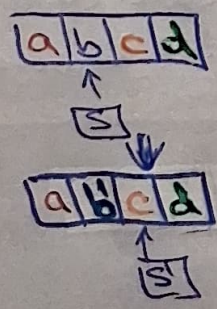
Dacă pot răsti forme:

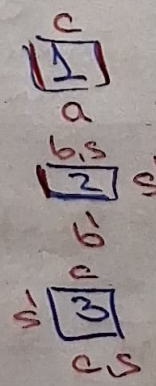
→ S-a crezut că nr. minim e 2 (pavage Penrose)

→ anul trecut → nr. min = 1 (2022)

(EINSTEIN MONOTILE)

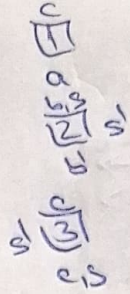
Pavage Wang simulează M. Turing





M. Turing

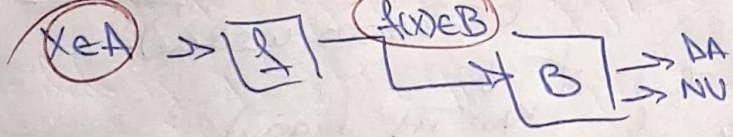
	a	b, s	c	d
	b', s'	c', s'	d	
1 a	2 b'	2 s'	3	



A, B - 2 pb de decizie

Def: $A \le_m B \iff (\exists) f: \Sigma^* \to \Sigma^*$ calculabilă de o M.T. care se oprește pe orice input a ?.

$x \in A \iff f(x) \in B$



A $\le_m$ B
B calculabilă de o MT
(B recursivă)
$\Rightarrow$ A recursivă

A $\le_m$ B
A nu e recursivă $\Rightarrow$ B nu e recursivă

ex: $W \le_m K$ $\Rightarrow$ nu e recursivă

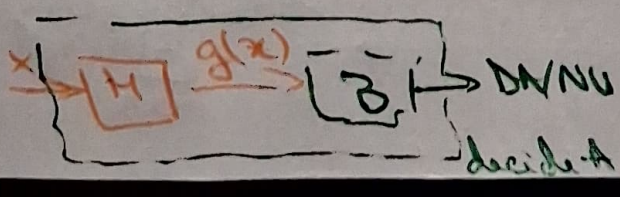
nu e recursivă

Curs 6:

$A \le_m B \iff (\exists) g: \Sigma^* \to \Sigma^*$, g calculabilă de o M. Turing M care produce rezultatul dorit

pb de decizie
$[f: \Sigma^* \to \{0,1\}]$
(N)

$(\forall) x \in A \iff g(x) \in B$



OBS: Dc B e decisă de o M. Turing/alg. at și pb A e decisă de o M. Turing.
Dc A nu e decisă de o M. Turing (și $A \leq_m B$) at nici B nu e decisă de o M. Turing.

$$\boxed{w \leq_m k = \{ \langle x, y \rangle \mid M_x(y) \text{ se oprește} \}}$$

Def: O pb de decizie e recursivă $\Leftrightarrow 1_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}$ calculabilă de o M.T.

w, k → nu sunt recursive

OBS: $A \leq_m B$ și B recursivă ⇒ A recursivă

Prop: A, B - pb de decizie a.î. A rec și $B \neq \emptyset, \Sigma^*$ $(x_0 \in B, x_1 \notin B)$ at. $A \leq_m B$

Dem: A rec ⇒ (∃) MT. care decide A. Constr. o fct g care arată că $A \leq_m B$ $(x \in A \Leftrightarrow g(x) \in B)$

$$g(x) = \begin{cases} x_0, & \text{dc } A(x) = 1 \\ x_1, & \text{dc } A(x) = 0 \end{cases}$$

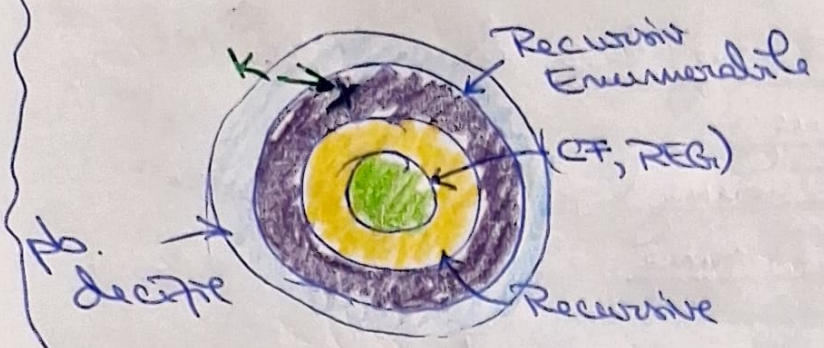
q.e.d

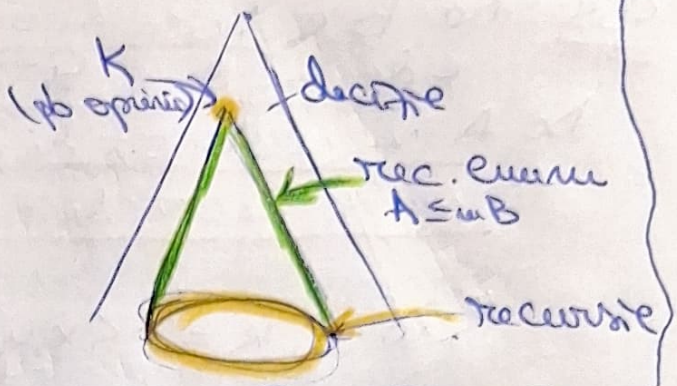
OBS: Dc $M_x(y)$ se oprește, pot verifica ușor acest lucru (rulez $U(\langle x, y \rangle)$ până se oprește, când se oprește, ret 1)

Def: O pb de dec A se n recursiv enumerabilă (semidecidabilă) $\Leftrightarrow$ $\boxed{f_A(x) = \begin{cases} 1, & x \in A \\ \uparrow, & x \notin A \end{cases}}$ calculabilă de o M.T.

OBS: A rec ⇒ A rec. enum

OBS: K e rec enum, dar NU e recursivă

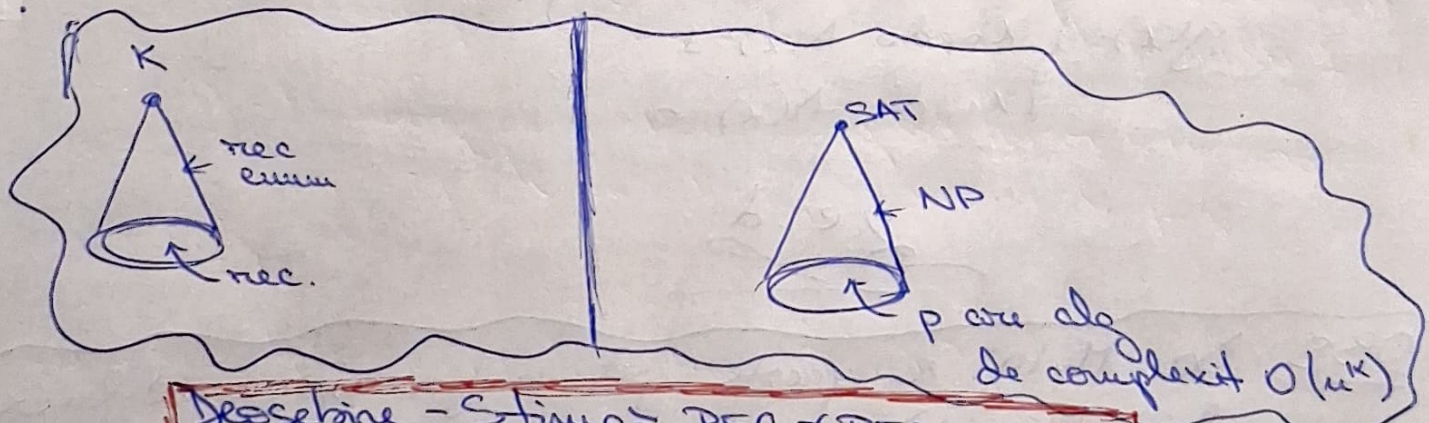




Teoremă: Dc A e o pb rec. enum, at. A se reduce la pb opriire ($A \leq_m K$)

Dem: A rec enum $\Rightarrow (\exists) x$ a.î. $A = L(M_x)$. Tb. să constr g a.î. $y \in A \Leftrightarrow g(y) \in K$; $g(y) = \langle x, y \rangle$ fct calculab. $y \in A \Leftrightarrow M_x(y)$ se oprește $\Leftrightarrow \langle x, y \rangle \in K$

Def: Pb A sn completă pt clasa $\mathcal{C}$ dc $A \in \mathcal{C}$ și $(\forall) B \in \mathcal{C}$, $B \leq_m A$



Observație: Știm că REC ≠ REC-ENUM
NU știm că P ≠ NP

Teoremă: A rec. enum
A rec-enum $\Leftrightarrow$ A recursivă

Dem: Fie $M_1 \to M_1(x) = \begin{cases} \uparrow, & x \in A \\ \uparrow, & x \notin A \end{cases}$

$M_2 \to M_2(x) = \begin{cases} \uparrow, & x \notin A \\ \uparrow, & x \in A \end{cases}$

$M(x)$ rulează în paralel $M_1(x)$ & $M_2(x)$

$M(x) = \begin{cases} 1, & M_1(x) = 1 \\ 0, & M_2(x) = 1 \end{cases}$

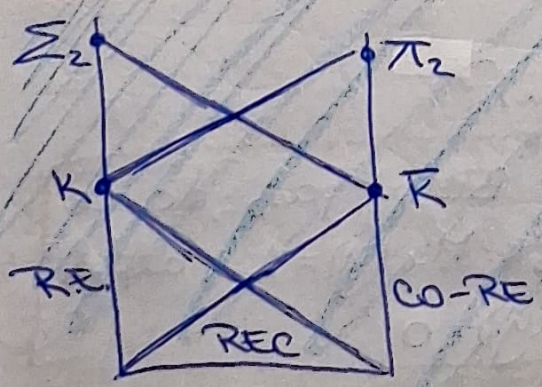
q.e.d

OBS: $\overline{K} = \{ \langle x, y \rangle \mid M_x(y)$ nu se oprește$\}$ nu e rec. enum

Dem: Pp RA $\overline{K}$ rec. enum
Stim K rec enum $\Big\} \Rightarrow$ K ar fi recursivă

Def: A este co-recursiv-enumerabilă $\Leftrightarrow$ $\overline{A}$. rec. enum

OBS: $\overline{K}$ completă pt cr recursiv enumerabile



$\Sigma_n, \Pi_n \to$ ierarhie analitică

$\Sigma_n = \{ A \mid \underbrace{(\exists) x_1 (\forall) x_2 (\exists) x_3 (\forall)}_{n \text{ cuantificatori}} \theta(x_1, \dots, x_n) \}$ recursiv

$\Pi_n = \{ A \mid \underbrace{\forall \exists \dots \exists}_{n} \}$

OBS: O pb de logică pred cu $n \geq 2$ cuantificatori are șansa să fie mai grea decât pb. opririi

Def: Complexit Kolmogorov

01010101 - nu e aleator
010010001 - nu e aleator
$\underbrace{011010010}_{n}$ - pare aleator

$$\boxed{K(y) = \min\{|x| \mid M_x(x) = y\}}$$

Teoremă ($\exists$) $c>0$ a.î. $(\forall) y, K(y) \leq |y| + c$

Def: Un cuv y se $\boxed{\text{aleator}}$ dc $\boxed{K(y) \geq |y| + O(1)}$

Teoremă: ($\exists$) cuv aleatoare oricât de lungi

$\boxed{\text{Fct } K(y)}$ $\boxed{\text{nu}}$ poate fi calc de o M. Turing

Kolmogorov complexity
Algorithmic Information Theory

Fct "Busy-Beaver" (Radó - 1962)

$\boxed{\Sigma(n)}$ - o M. Turing cu n stări și alfabet binar $\{0,1\}$ care pe intrarea vidă scrie oricât de multe 1

$\boxed{S(n)}$ - M. Turing cu n stări și alfabet unar $\{1\}$ care se oprește în cel mai mare nr de pași

$BB(n)$

Cum rez pb opririi dc știm $BB(n)$?

$M_x(y) \to M_{f(x,y)}(1)$ o rulez $BB(n)$ pași; dc se oprește $M_x(y) = 1$; altfel $M_x(y) = 0$

f: n stări

Teoremă: $\boxed{f(n) = BB(n)}$ $\boxed{\text{nu}}$ e calculabilă de o M. Turing

$BB(1) = 1$
$BB(2) = 6$
$BB(3) = 21$
$BB(4) = 107$
$BB(5) > 47.176870$

"Humanity might never know $BB(7)$"

$BB(6) \approx$ nr de 16 cifre

Clarificări (O să mă reîntorc aici. Nu fac numărări conexiuni momentan)

Kolmogorov Complexity

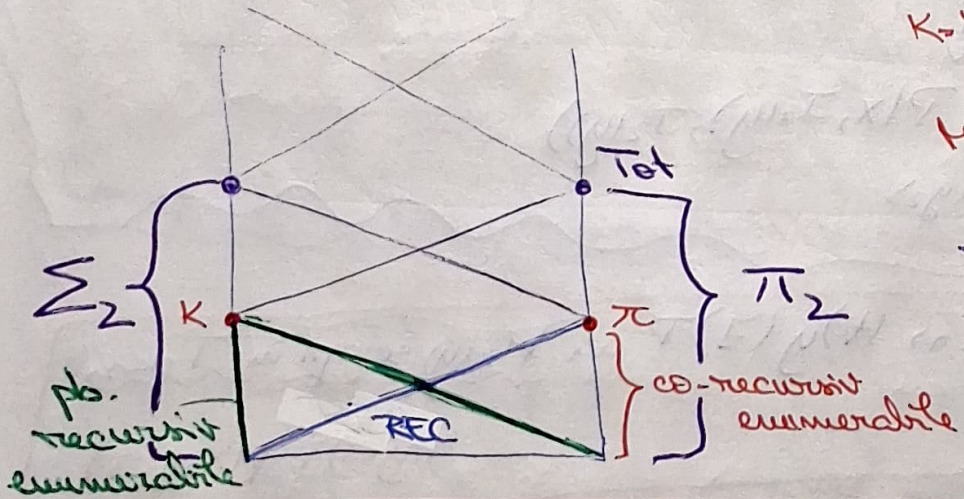
↳ lung celui mai scurt program care produce obiectul (dat) ca output.

↳ tot pt a dem problema opriri

Fct. Busy Beaver

↳ cuantifică scorul max pe care îl poate dat un "busy beaver" pe o măsură dată.

Curs 7: Complexitatea pb de decizie



K → "semicalculabilă" (rec. enumerabilă)

$M \to f_K(\langle x,y\rangle) = \begin{cases} 1, & (x,y)\in K \\ \uparrow, & \text{altfel} \end{cases}$

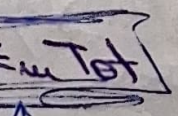
$Tot \in \Pi_2$ (cea mai grea pb din clasa Π_2)

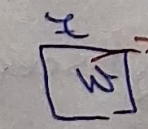
$$\boxed{\begin{array}{l} A \text{ cu Recursivă} \Leftrightarrow (\exists) \text{ M. Turing a.î.} \\ M(x) \text{ calc } 1_A(x) = \begin{cases} 1, & x\in A \\ 0, & x\notin A \end{cases} \end{array}}$$

$Tot = \{x \mid M_x \text{ se oprește pe } (\forall) \text{ input } y\}$

tot ar fi să verific pe o mul de inputuri (de aici e mai greu)

$\boxed{K \le_m Tot}$ dar $Tot \not\le_m K$





→ $w(z)$ simulează $M_x(y)$

$M_x(y)$ se oprește dacă $w(z)$ se oprește $(\forall) z$, altfel nu se oprește pe nicium z.

$\boxed{A \le_m B}$

Complementul M. Turing (vrem o maș care se oprește pt o mașină care nu se oprește)

$\boxed{\langle x,y\rangle \in K} \Leftrightarrow \boxed{M_x(y) \text{ se oprește}} \Leftrightarrow (\exists)\boxed{T}$ a.î.

$\boxed{M_x(y) \text{ se oprește în } T \text{ pași}}$

$\hookrightarrow P(x,y,t)$ – predicat recursiv

rec. enum. A $\quad x \in A \Leftrightarrow (\exists) y$ a.î. $\langle x,y\rangle \in P$

(o mulț. e rec. enum. $\Leftrightarrow$ se def. dim pred. recursiv + quantif. existențiali)

rec enum = $(\exists)$ recursiv

$(\exists) y_1 (\exists) y_2 \; P(x, y_1, y_2)$

$\Updownarrow$

$(\exists) y \; P(x, I_1(y), I_2(y))$

$y = \langle y_1, y_2 \rangle$

Tot $\quad x \in$ Tot $\Leftrightarrow (\forall) y (\exists) T$ a.î. $M_x(y)$ se opr în T pași

$\downarrow$

$(\forall) y (\exists) T \; P(x,y,T)$

Ierarhia aritmetică

(Σ_n, Π_n)

$\boxed{\Sigma_n \rightarrow \exists\forall\exists\forall \ldots P(\ )}$

$\boxed{\Pi_n \rightarrow \forall\exists\forall\exists \ldots P(\ )}$

Tic-tac-toe pe tablă inf (scop: K în linie)

(Jocul e determinant sau nu?)

Pb: are un jucător o strategie câștigătoare?

Șahul $\rightarrow$ determinant (un juc câștigă sau forțează remiza)

$k \le 5$ primul juc. câștigă
$k \ge 8$ al doilea juc. poate forța un joc infinit
$k = 6, 7 \rightsquigarrow$ Nobody knows ☺

Pb de decizie → mai grea decât orice clasă de pb polinomiale

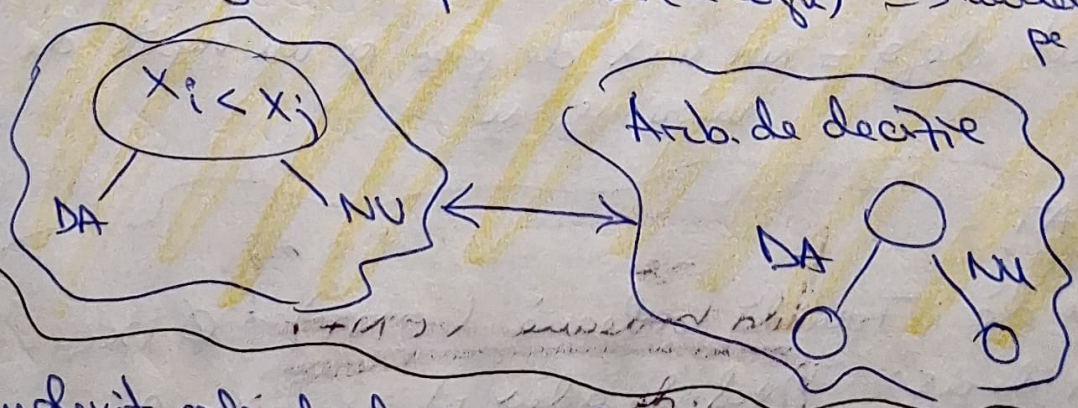
INPUT: Joc de 2 pers $M(x_1, y_1, \dots, x_n, y_n)$
De decis: Are primul jucător o strategie de câștig?

$$\boxed{\Sigma_n \cup \Pi_n \subset \Sigma_{n+1} \cap \Pi_{n+1}}$$

$\Sigma_n \neq \Sigma_{n+1}$ $\quad$ $\Pi_n \neq \Pi_{n+1}$
$\Sigma_n \subset \Sigma_{n+1}$ $\quad$ $\Pi_n \subset \Pi_{n+1}$
$\Sigma_n \neq \Pi_n$

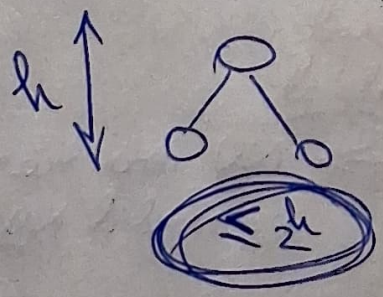
Complexitate computațională (Pb. rezolvabile eficient de cele care nu au alg eficienți)

ex: sorting are complexit $\Omega(n \log n)$ → modelul bazat pe comparații



Complexit cazului de decizie = lung celui mai lung drum de la răd către frunze

$\Omega(n \log n)$



nr perm n. elem → n!

2^h noduri

$\boxed{n! \le 2^n}$

$\boxed{\text{FORMULA LUI STIRLING}}$

$$\lim_{n\to\infty} \frac{n!}{\left(\frac{n}{e}\right)^n \sqrt{2\pi n}} = 1$$

$$h \ge \log_2(n!) \ge \log_2\left(\left(\frac{n}{e}\right)^n \sqrt{2\pi n}\right) - O(1)$$

$$\ge c\, n \log n$$

Ce putem calcula eficient?

$O(n \log n)$ – eficient

$O(n^2)$ – eficient

$O(n^{100})$ – eficient

2^n – $\boxed{\text{NU}}$ e eficient

($2^{100} \equiv$ nr de atomi din universul observat)

P (PTIME) – clasa de pb de decizie care au algo cu complexit $O(n^k)$

$\boxed{\text{Exponențial} \to \text{rău}}$; $\boxed{\text{polinomial} \to \text{bun}}$

Complexit algo = nr max de pași pe un input de mărime n

ex: SAT

Se dă: Formulă booleană (CNF) $\phi(x_1, \dots, x_n)$

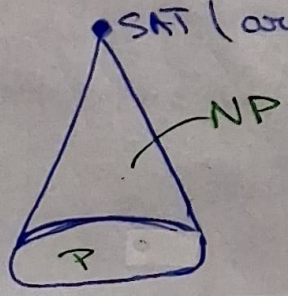
De decis: $(\exists)$ val de adev pt $x_1, \dots, x_n$ a.î.

$\phi(x_1, \dots, x_n) = \text{TRUE}$?

→ tabelă adevăr → exponențial

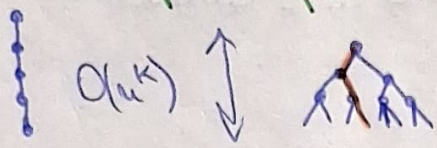
backtracking → exponențial

SAT (are sau nu alg polin?) → NU are răsp (probabil NU)



$P \ne NP$?

NP = clasa de pb de decizie care are alg. nedeterministi cu complexit. polinomială



NP - pb. de decizie pt care $(\Rightarrow)$ un algo polinomial P

$P(x,y) = 1$ (martor)

ex: $x = \emptyset$, $y = 010001$

$x \; (\exists) y \; P(x,y)$

$-(\forall) y$ a.î. $P(x,y) = 1$ $\quad |y| \le |x|^k$

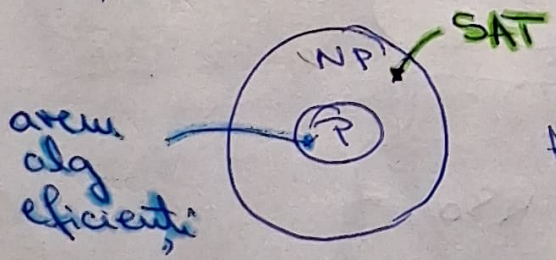
Def: O pb de decizie A aparț. clasei NP $\Leftrightarrow$

$\Leftrightarrow \begin{cases} (\exists) \; P(x,y) \text{ calculabil în timp polinomial } |x|+|y| \\ k > 0 \end{cases}$

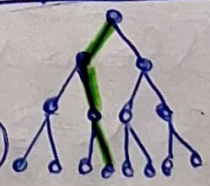
a.î. $x \in A \Leftrightarrow (\exists) y \; |y| \le |x|^k \; P(x,y)$

Curs 8: Pb de decizie

$P \neq NP$



Exp. backtracking



Algo verif:

INPUT: $\phi, \vec{x}$

VERIF. $\phi(\vec{x}) = TRUE$ $\quad O(|\vec{x}|^k)$

$NP = \{ A \mid (\exists) f(\cdot,\cdot)$ calculabil în timp polinomial$\}$

$f(x,y) \to O((|x|+|y|)^k)$

$(\exists)$ p polinom a.î.

$x \in A \Leftrightarrow (\exists) y \; |y| \le p(|x|)$ a.î. $f(x,y) = TRUE$

martor pt x

x = Form ϕ (CNF)

$\phi = \wedge(\vee \ldots \vee)$

$y = \cancel{FTFFTT}$

3-COL = {G graf | G poate fi colorat cu 3 culori}

NP

$f(\cdot,\cdot)$

$O(n^2)$

Algo

INPUT: G, $c = c_1 c_2 \ldots c_n \in \{1,2,3\}^n$

for v in V

for w in V

if $(v,w) \in E$ și $c_v = c_w$

ret. FALSE

ret. TRUE

3-COL =

= {G | (G) c a.î. $f(G,c)$ = TRUE}

HC (Hamiltonian Cycle)

$f(\cdot,\cdot)$

$O(n^2)$

INPUT: G

$c = c_1, \ldots, c_n \in V^n$

for i = 1 to n

for j = i+1 to n

if $c_i = c_j$

ret FALSE

for i = 1 to n

if $(c_i, c_{i+1}) \notin E$

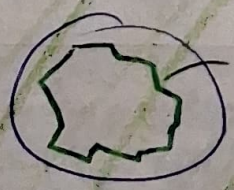
ret. FALSE

ret TRUE

TSP (Comis Voiajorului)

Se dă $G = K_n$, $d: E \to \mathbb{R}_+$, $K > 0$

be decis



$\sum d(e) \leq K$

$f((G,d,K), c_1, \ldots, c_n \in V^n)$

for i = 1 to n

for j = i+1 to n

if $c_i = c_j$

ret FALSE

INPUT: (G,d,K)

$c_1, \ldots, c_n \in V^n$

if $(\sum d(c_i, c_{i+1}) > K)$

ret FALSE

ret TRUE

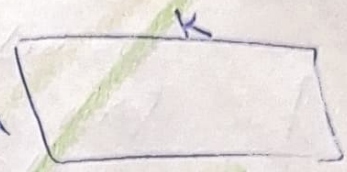
ex: KNAPSACK

$v_1 \dots v_n$

$c_1 \dots c_n > 0 \quad K > 0 \quad v > 0$

coloane (prog. dinam) = K

De decis: pot să iau obiecte de greutate $\leq K$ de val $\geq v$



K = în binar (100 biți)

K = exponențial în $|K|$

$\Omega(n \cdot K)$

① Dc. INPUT SIZE $|c_1| + \dots + |c_n| + |K| + |v|$ (codif binar) $\Rightarrow$ PROGR DINAMICĂ exponențial.

② Dc. INPUT SIZE $c_1 + c_2 + \dots + c_n + K + v$ $\Rightarrow$ PROGR. DINAMICĂ polinomial

$\underbrace{1 \dots 1}_{c_1 \text{ biți}}$

KNAPSACK ① $\in NP$

$\rho(\cdot,\cdot)$

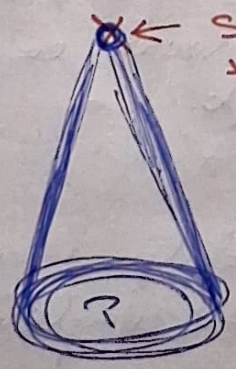
INPUT: $c_1 \dots c_n, K, v$ (codif. binar)

$y = y_1 \dots y_n \in \{0,1\}^n$

$O(n)$

```
if (sum [c_i for y_i = 1] > K)
    ret. FALSE
if (sum. [v_i for y_i = 1] < v)
    ret. FALSE
return TRUE
```

Concl: SAT, 3-COL, HC, TSP, KNAPSACK $\in NP$



$A \leq_m^P B$ $(\exists) f: \Sigma^* \to \Sigma^*$ calculabilă (în timp polinom) a.î. $(\forall) x \in A \Leftrightarrow f(x) \in B$

3-COL $\leq_m$ SAT e calculabil în timp polinomial

INPUT G

$O(|V|+|E|)$

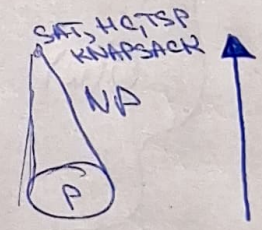
```
φ = vidă
for v in V
    add to φ
        x_{v,1} ∨ x_{v,2} ∨ x_{v,3}
        ¬x_{v,1} ∨ ¬x_{v,2}, ¬x_{v,1} ∨ ¬x_{v,3}, ¬x_{v,2} ∨ ¬x_{v,3}
for (v,w) in E
    for i=1 to 3
        add to φ
            ¬x_{v,i} ∨ ¬x_{w,i}
```

Def: O pb de decizie A se numește $\boxed{\text{NP-dificilă}}$ ($\boxed{\text{NP-hard}}$)

$\Leftrightarrow (\forall)\ \boxed{B \in NP} \rightarrow \boxed{B \leq_m^p A}$

Def: O pb de decizie A se numește $\boxed{\text{NP-completă}}$

$\Leftrightarrow \begin{cases} \boxed{A \in NP} \\ \boxed{A - NP\text{-hard}} \end{cases}$



Mii de pb sunt NP-complete (fizică, bio, st. politice)

Imp. pb NP-complete – de am un algo eficient pt o pb NP-completă at pt (în practică) toate solvabile ca să rez. alte pb.

Fizică statistică

- Model Ising – pb calculării modelului Ising în dim 2,3 dimens → NP-completă

Idee a dem SAT e NP-completă

M. Turing (nedeterministe)

$T_M(x)$ (pt o mag. dat) $\underbrace{C_0(x) \rightarrow C_1 \rightarrow C_2 \rightarrow _ \rightarrow C_k}_{T_M(x)=k}$ ACCEPT / REJECT

$DTIME[f] = \{A \mid (\exists) M.$ Turing $M.$ a.î. $(\forall) x$ $M(x)$ se oprește în $O(f(x))$ pași$\}$

$f: \mathbb{N} \to \mathbb{N}$

$P = \bigcup_{k \geq 0} DTIME[n^k]$

Curs 9 $\quad P \neq NP$ (probabil)

Def: $\boxed{A \leq_m^P B} \iff (\exists) f: \Sigma^* \to \Sigma^*$ calculabilă în timp polinomial $x \in A \iff f(x) \in B$

Def: A este $\boxed{NP\text{-completă}}$ $\begin{cases} \boxed{A \in NP} \\ \boxed{(\forall) B \in NP \quad B \leq_m^P A} \end{cases}$

Teoremă: SAT e NP-completă

Dem: $SAT \in NP$ în sor $\quad A: V \to \{0,1\}$

INPUT (ϕ, A)
Dc $\phi(A) = TRUE$
ACCEPT
altfel
REJECT

Trebuie să dem: $(\forall) B \in NP$, $B \leq_m^P SAT$

NP $\longrightarrow$ M. Turing NEDETERM $\longrightarrow$ circuit boolean

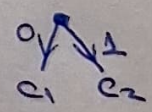
M. Turing obișnuită (deterministă)

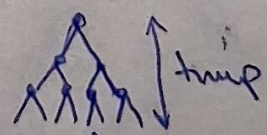
$$\boxed{\delta: S \times \Sigma^k \to S \times \Sigma^{k-1} \times \{\rightarrow, \leftarrow, -\}^k}$$

timp

M. Turing nedeterministă

$$\boxed{\delta: S \times \Sigma^k \times \boxed{\{0,1\}} \to S \times \Sigma^{k-1} \times \{\rightarrow, \leftarrow, -\}^k}$$





M(x) acceptă $\iff$ (∃) drum y a.î. M(x) acceptă pe dr y

Teoremă: $\boxed{A \in NP}$ $\Leftrightarrow$ există o M. Turing nedeterministă în timp polinomial care acceptă A

x ia pași $\leq p(|x|)$ polinom

SAT

ϕ A=01001

Fie $B \in NP$

$B = L(M)$

↳ maz Turing nedeterm polinomială

$x \longrightarrow \phi_x$ – Maz M pe intrarea x accepta

↓

există drum pe care M acceptă

$\phi_x = \boxed{\phi_x^1} \wedge \boxed{\phi_x^2} \wedge \boxed{\phi_x^3}$

ϕ_x^2 – tranziții

ϕ_x^1 – start în config init pe intrarea x

ϕ_x^3 – start în stare accept

config căsuța 1 = x_1 stare căs 1 = start

config căsuța 2 = x_2 stare căs 2 = –

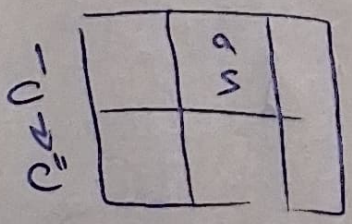
⋮

config căsuța n = x_n stare căs n = –

$\boxed{a \mid s}$ $\boxed{b \mid -}$

$\boxed{a \mid s}$ $y_{i,a} = \begin{cases} \text{TRUE } a \\ \text{FALSE } b \end{cases}$

$z_{i,s} = \begin{cases} \text{TRUE } s \\ \text{FALSE altfel} \end{cases}$

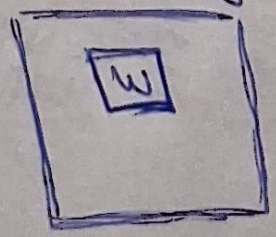


$\psi(\vec{z}) = \text{TRUE}$

$\Updownarrow$

$\vec{z}$ codif un pas al M. Turing

$\phi_x^2 = \bigwedge \psi_w$



Cum arăt că A e NP-completă?

$A \leq_m^p B$ și $B \leq_m^p C$ $\Rightarrow A \leq_m^p C$

$A \in NP$ $(\exists) B$ NP-completă a.î. $B \leq_m^p A$

$x \in A \Leftrightarrow f(x) \in B$

$\quad \| $

$y \in B \Leftrightarrow g(y) \in C$

$x \in A \Leftrightarrow g(f(x)) \in C$ $\quad$ $g \circ f$ calculabil în timp polinom. dc g, f calculabil în timp polinomial

$\boxed{\text{3-SAT} \in NP}$

INPUT: $\phi = \wedge C_i$ $\quad |C_i| \leq 3$

De decis: E ϕ SAT sau nu?

Teoremă: 3-SAT e NP-completă.

Dem: 3-SAT $\in NP$

SAT $\leq_m^p$ 3-SAT

$\phi \longrightarrow \tau_\phi$ $\quad$ 3-CNF

a.î. $\phi \in SAT \Leftrightarrow \tau_\phi \in SAT$

ϕ	τ_ϕ
$\bar{x} \vee \bar{y}$	$\bar{x} \vee \bar{y}$
$\bar{z} \vee t \vee \bar{w}$	$\bar{z} \vee t \vee \bar{w}$
$a \vee \bar{b} \vee \bar{c} \vee d$	$a \vee \bar{b} \vee u$ → var nouă
	$\bar{u} \vee \bar{c} \vee d$

$$\underbrace{a_1 \vee \bar{a}_2}_{b_1} \vee a_3 \vee \underbrace{a_4 \vee a_5}_{b_2} \longrightarrow \boxed{\begin{array}{l} a_1 \vee \bar{a}_2 \vee b_1 \\ \bar{b}_1 \vee a_3 \vee b_2 \\ \bar{b}_2 \vee a_4 \vee a_5 \end{array}}$$

var noi

OBS: 2-SAT $\in$ P

OBS: $\boxed{\left.\begin{array}{l} K\text{-SAT} \\ K \geq 3 \end{array}\right\} \text{NP-completă}}$

3-SAT $\leq$ 4-SAT

$x \vee y \vee z \longrightarrow x \vee y \vee z \vee \alpha$

$x \vee y \vee z \vee \bar{\alpha}$

Def: $\boxed{\text{1-IN-3 SAT}}$

INPUT: $\phi = \wedge C_i$ $\quad C_i = 1\text{-IN-}3\ (x, y, z)$

$\boxed{\text{Satisfying Assignment}}$ $\quad A: V \to \{0,1\}$ $\quad C_i$ $(\forall)\ C_i$ $\boxed{\text{exact una}}$ din $A(x), A(y), A(z)$ $\boxed{\text{e adevărată}}$.

Dem 3-SAT $\leq_m^P$ 1-IN-3 SAT

ϕ $\qquad\qquad$ $f(\phi)$

$x \vee y \vee z$

T F F

var noi

$R(x, a, d)$

$R(y, b, d)$

$R(a, b, e)$

$R(e, d, f)$

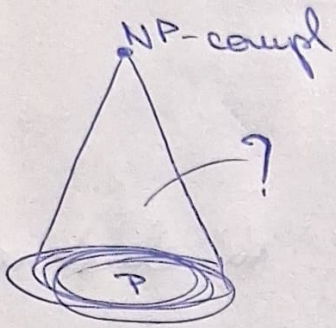
$R(z, c, \text{FALSE})$

x	y	z	a	b	c	d	e	f
T	F	F	F	T	T	F	F	T

OBS: Până acum avem variante ale pb SAT

P — 2-SAT, HORN-SAT

NP-COMPL — 3-SAT, (k-SAT) $k \geq 3$, 1-IN-3 SAT



T. Ladner

$P \neq NP \Rightarrow (\exists) A \in NP/P$, A nu e NP-completă

Pb. intermediare: Izomorfismul a două grafuri

Se dau: G_1, G_2 2 grafuri cu n vârfuri (matr. de adiacență)

De decis: $(\exists)$ bijecție $\pi: V_1 \to V_2$ a.î. $(x,y) \in E_1 \Leftrightarrow (\pi(x), \pi(y)) \in E_2$

Teorema lui Schaefer (Informal)

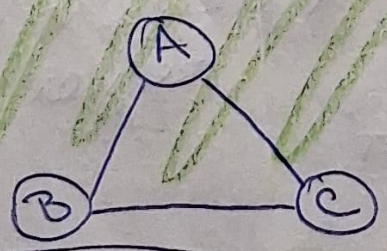
Var. pb SAT sunt fie în P fie NP-complete.

Horn-SAT
2-SAT
(negated HORN)
XOR-SAT

Independent-SET

Se dă: $G=(V,E)$ graf; $1 \leq k \leq n$

De decis: $(\exists)$ W mulț de vf de card $\leq k$ $(|W| \leq k)$ a.î. $(\forall)$ $(v,w) \in E$, $v \in W$, $w \in W$



$k=1$ NU

$k=2$ DA

Teoremă: Independent-SET (IS) e NP-completă.

Dem: IS ∈ NP este

Verifier $(G,K) \to W$

$3\text{-}SAT \leq_m^p IS$

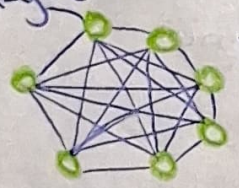
ϕ m clauze

n variabile

$x \vee y \vee z$

∃ sat assignment

$x=y=z=1$

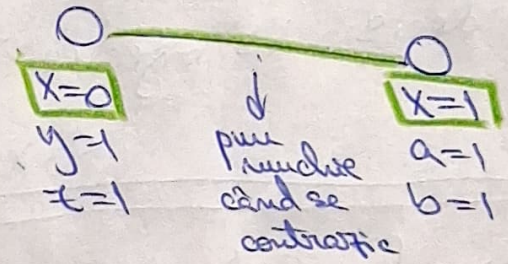


∃ sf

G

K

m clustere



Curs 10 — Independent Set (IS)

INPUT: G graf, $K \geq 1$

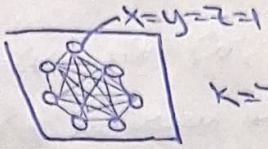
De decis: $(\exists) W \subseteq V$, $|W| \geq K$ a.î. $(\forall)(v_1, v_2) \in E$ $v_1 \notin W$ sau $v_2 \notin W$

IS ∈ NP

```
Algo G, K, W
  dc |W| ≤ K => REJECT
  altfel
    for (v1, v2) in E:
        if v1 ∈ W si v2 ∈ W
            REJECT
  ACCEPT
```

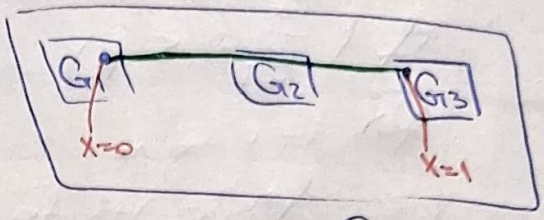
3-SAT $\leq_m^p$ IS

$\phi \xrightarrow{f} G_\phi, K_\phi$

$x \vee y \vee z$ (7 sol) $\xrightarrow[\text{cu 7 vf}]{\text{Asoc opal}}$ 

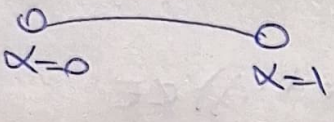
$k=7$

$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$



G_ϕ

Adaug muchii



$\boxed{K_\phi = m}$

ϕ satisfiabil $\Leftrightarrow$ G_ϕ are un IS de mărime m

Dem: "$\Rightarrow$" Pp ϕ satisfăcută de $A: V \to \{0,1\}$

$A \models C_1 \Rightarrow (\exists) v_1 \in C_1$ a.î. $A(v_1) = TRUE$

$\vdots$

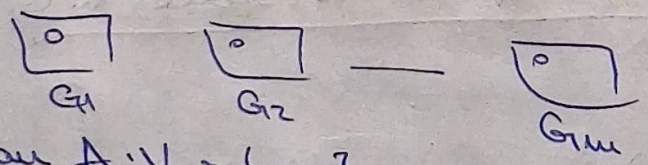
$A \models C_m \Rightarrow (\exists) v_m \in C_m$ a.î. $A(v_m) = TRUE$

$C_i = x_i \vee y_i \vee z_i$ $\quad W_i = (A(x_i), A(y_i), A(z_i))$

$W = \{W_1, W_2, \ldots, W_m\}$ e un IS

Din construcție, nu există o muchie între W_i și W_j

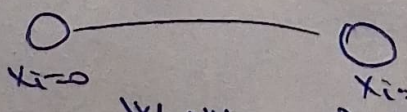
"$\Leftarrow$" Pp G_ϕ are un IS W de mărime $\geq m$



Iau $A: V \to \{0,1\}$ $\quad A \models \phi$

$C_i \models x_i, y_i, z_i$ aleg $A(x_i), A(y_i), A(z_i)$ conform W_i

A bine def; A nu dă val. def. la aceeași variabilă



W NU are fi IS

$A \models \emptyset \quad (\forall) i = \overline{1,m} \to A \models C_i$

Pb. CLIQUE $\in$ NP

INPUT: $G=(V,E)$, k, $w = w_1 \ldots w_n \in \{0,1\}^n$

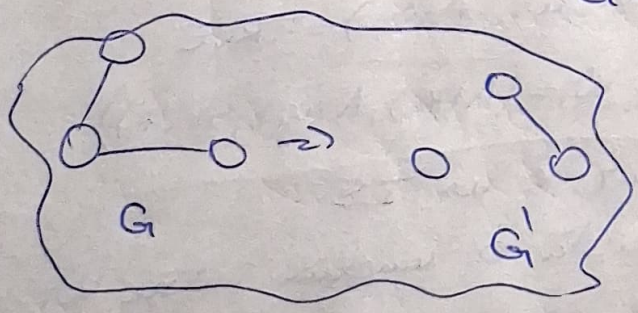
De decis: $(\exists)$ clică de mărime $\geq k$?

```
If ( Σ_{i=1}^{n} w_i < k) -> REJECT
for i := 1 to n
    for j = i+1 to n
        If w_i = 1 și w_j = 1 și (i,j) ∉ E
            REJECT
ACCEPT
```

$IS \leq^P_m$ CLIQUE

$(G,K) \longrightarrow (G', K')$ $\qquad K' = K$

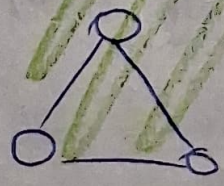
G' = complementul lui G



$\boxed{W \text{ clică în } G' \Leftrightarrow W \text{ IS. în } G}$.

Vertex Cover

Def: Se dă (G,K). Vreau o mulț de vf $S \subseteq V$, $|S| = K$ a.î. $(\forall)(v_1,v_2) \in E$, $v_1 \in S$ sau $v_2 \in S$



$K=2 \to DA$

$K=1 \to NU$

$\boxed{VC \in NP}$

$\boxed{IS \leq^p_m VC}$

$(G, K) \longrightarrow (G', K')$ $\quad G' = G$, $K' = n - K$

$S \subseteq V$ e IS în G $\quad |V \setminus S| = K'$

$\Updownarrow$

$V \setminus S$ e VC în G'

$\boxed{P}$ (vezi de rez) — 2-SAT
cuplaj perfect
PRIMES (de decis dc un nr scris în binar e prim)

$\boxed{NP\text{-complete}}$ (are anumite instanțe greu de rezolvat) — SAT, 3-SAT, IS, CLIQUE, TSP, HCP, VC

Necunoscut — GI (izom. grafurilor), factoring

"Tranziții de fază" în pb NP-complete

$\phi, \phi_1 = \phi \wedge c$ $\quad$ "ϕ_1 - mai puțin satisfiabilă decât ϕ"

$\boxed{\phi_1 \in SAT \Rightarrow \phi \in SAT}$

Generăm formule aleatoare de tip 3-SAT

$c = \frac{m}{n}$ $\quad m \leftarrow$ nr. clauze, $n \leftarrow$ nr. variabile

$n = 100$ $\quad c = 3$

$x_1 \vee x_2 \vee x_3$

$\vdots$

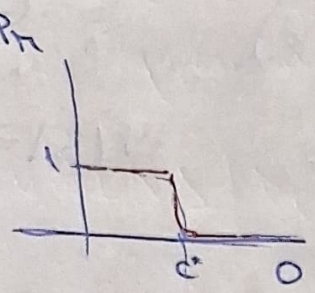
$\overline{x_{98}} \vee \overline{x_{99}} \vee \overline{x_{100}}$

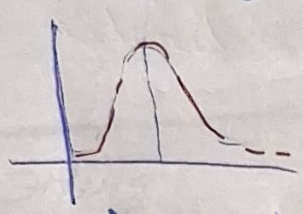
C_1 C_2 — C_{300}

$Pr\{\phi \in SAT\} = ?$

$c_1 < c_2 \Rightarrow Pr_{c_1}\{\phi \in SAT\} > Pr_{c_2}\{\phi \in SAT\}$

Când $n \to \infty$





(Timpul de rulare are un spike în jurul lui c)

Intuitie: multe pb NP-complete au "tranziții de fază" și cele mai grele instanțe se găsesc la pct. de tranziție

Există pb NP-complete care sunt ușoare "în medie"

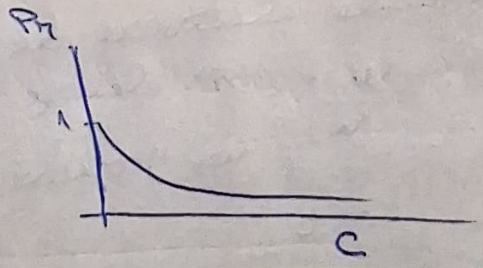
1-IN-3 SAT (1-IN-K SAT, $K \geq 3$)

$$c^*_K = \frac{1}{\binom{K}{2}}$$ ← comb de K luate câte 2

$\boxed{\text{1-IN-K SAT}}$ e "ușoară în medie" deși e NP-completă

Există pb $\boxed{\text{fără tranziții de fază}}$

$\boxed{\text{HORN-SAT}}$



$\boxed{\text{Complexit} \Leftrightarrow \text{tranz de fază pt alg Davis-Putnam}}$

Curs 11

Unde sunt "cele mai grele instanțe" pt o pb. NP-completă?

3-SAT : Param $c > 0$

$$\begin{cases} \#\text{var} = n \\ \#\text{clauze} = c \cdot n \end{cases}$$

$8\binom{n}{3}$ clauze posibile de lung 3

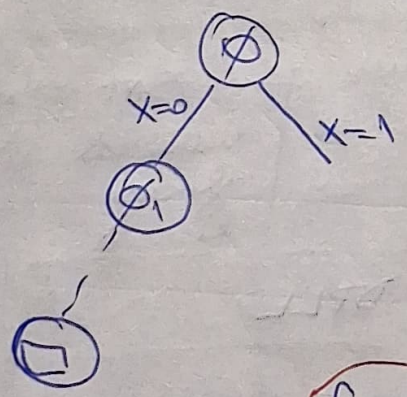
DPLL (Davis-Putnam (Longeman-Loveland))

↳ alg pt SAT backtracking complet

SAT : $\phi(x_1 \ldots x_n)$ CNF (formă normală conj)

$\phi = c_1 \wedge c_2 \wedge \ldots \wedge c_m$ → clauză

Clauză : $\bar{x} \vee y \vee z$



$$\phi \begin{bmatrix} x \vee \bar{y} \vee \bar{z} \\ \bar{x} \vee y \vee z \\ \bar{x} \vee \bar{y} \vee z \\ z \vee \bar{t} \vee \bar{q} \end{bmatrix}$$

$x=0$
$\Downarrow$
$\bar{x}=1$

$$\phi_1 \begin{bmatrix} \bar{y} \vee \bar{z} \\ \bar{z} \vee \bar{t} \vee \bar{q} \end{bmatrix}$$

Greșeș arde backtracking
Dc dă contradicție, mă întorc
înapoi un pas

Situații în care val. unei variabile e clară

① x - literal pur ⇒ val lui x det.

$$\phi = \begin{bmatrix} \bar{x} \vee \bar{y} \vee \bar{z} \\ \bar{x} \vee \bar{z} \vee \bar{q} \\ a \vee b \vee c \end{bmatrix}$$

x apare doar negat (dc. îi dau lui x val FALSE, satisf. ambele clauze simultan)

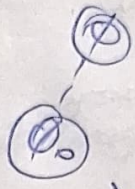
$\phi \in SAT \Rightarrow (\exists) A \models \phi$ $\quad A(x) = FALSE$

Def: x e literal pur în ϕ dc doar unul dintre x și $\bar{x}$ apare în ϕ

REGULA 1: Dc ϕ conț un literal pur x, setez $x = TRUE$

③ x - literal unitar

$\phi_0 = [\, x \,]$



(doar o clauză de lung 1)

REGULA 2: Dc o var. apare într-o clauză unitară (lung=1) îi dau val. care satisface clauza respectivă.

DPLL
- backtracking (1 pas în sus în caz de eroare)
+
- clauze pure
+
- clauze unitare

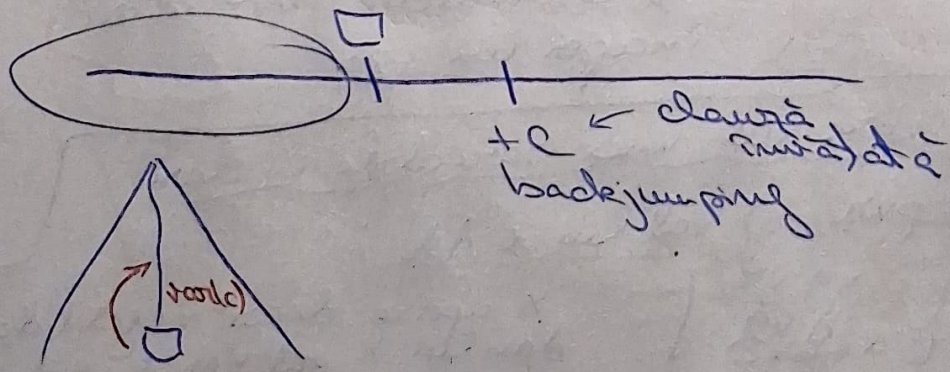
tranziții de fază → algo de tip DPLL

SAT solvere
- minisat
- glucose
- kissat

idee pt accelerarea algo de tip DPLL

↓

CDCL (Conflict Driven Clause Learning)

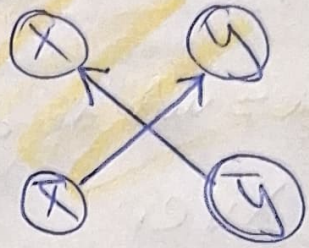


Exemplu 2-SAT

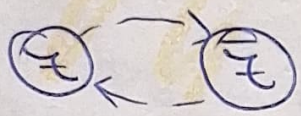
formula graph

$x \vee y \rightsquigarrow \bar{x} \to y$

$\bar{y} \to x$



UN SAT



Concluzii:

1. Algo proiectați pt SAT bazați pe paradigma DPLL
2. Cea mai imp. îmbunătățire → CDCL
3. Multe instanțe "din practică" pt SAT pot fi rez. cu CDCL (deși $\boxed{P \neq NP}$)
4. Există clase de formule pe care algo DPLL nu se calează bine

$\boxed{PHP^n_{n-1}}$ formă care codifică [Principiul cutiei / Princ. lui Dirichlet]

n porumbei

n-1 căsuțe $\Rightarrow$ IMPOSIBIL fără a avea 2 porumbei în aceeași căsuță

$\boxed{PHP^n_{n-1} \in \overline{SAT}}$ ← $x_{i,j} = \begin{cases} \text{TRUE} & i \to j \quad i=\overline{1,n},\ j=\overline{1,n-1} \\ \text{FALSE} & \text{altfel} \end{cases}$

$\boxed{x_{i,1} \vee x_{i,2} \vee \ldots \vee x_{i,n-1}}$ ← $i=\overline{1,n}$ porumbelul i merge în ≥ 1 căsuțe

$\boxed{\overline{x_{i,a}} \vee \overline{x_{j,a}} \quad a=\overline{1,n-1},\ 1 \leq i < j \leq n}$ ← doi porumbei diferiți nu merg în căsuța a

$n=15$ $\quad 15 \times 14 = 220$ variabile în PHP_{14}^{15}

$\boxed{\text{Timpul de rulare DPLL pe } PHP_{n-1}^{n} \text{ exponențial în } n}$

Rezoluție = mod de a dem că o formulă e $\boxed{\text{nesatisfiabilă}}$

$$\frac{x \vee C_1 \qquad \bar{x} \vee C_2}{C_1 \vee C_2}$$

ex: $\dfrac{x \vee a \qquad \bar{x} \vee b \vee c}{a \vee b \vee c}$

$\dfrac{x \qquad \bar{x}}{\square}$ ← contradicție

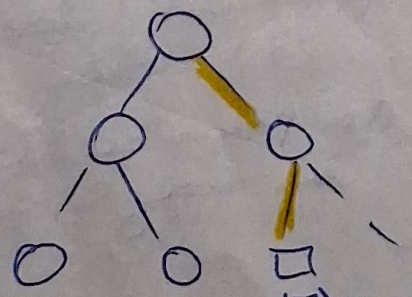
Teoremă: $\emptyset \notin SAT$ (e nesatisfiabilă) $\Leftrightarrow (\exists)$ demonstrație prin rezoluție pt $\square$ pornind de la $\emptyset$

Mă interesează (proof complexity) $\Rightarrow$ $\boxed{c(\emptyset)}$ = nr. min de clauze în cea mai scurtă demonstrație pt $\emptyset$ prin rezoluție

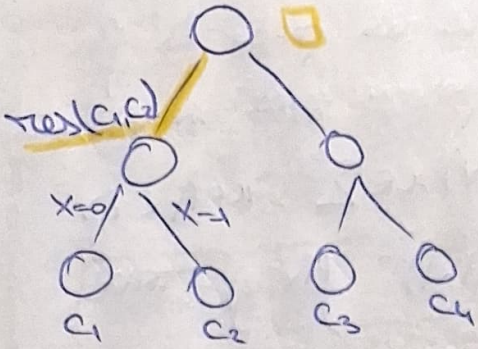
Teoremă (Haken): $(\exists)\lambda > 1$ a.î. $\boxed{c(PHP_{n-1}^{n}) > \lambda^{n}}$

Dream: DPLL $\Rightarrow$ dem prin rezoluție pt $\emptyset \in \overline{SAT}$

$\boxed{\text{timpul ALG} \geq \#\text{clauze din dem}}$



$(\exists)\, C \in \emptyset$ falsă cu assignmentul curent

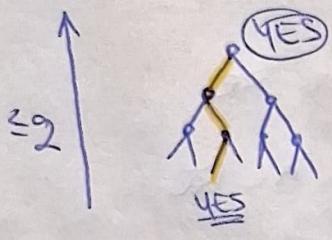


$c_1 = x \vee D_1$
$c_2 = \bar{x} \vee D_2$

Curs 12

Pb în afara clasei NP

NP

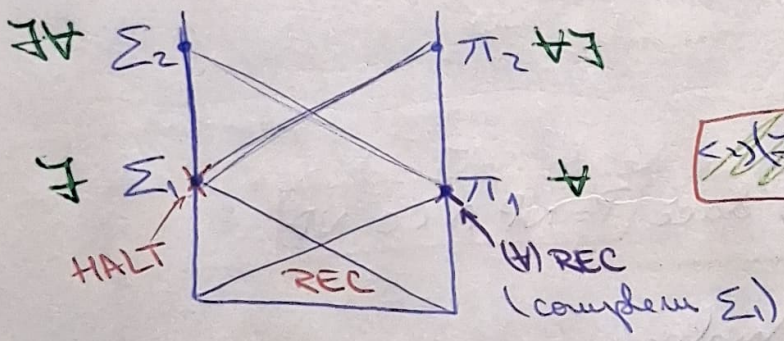


TAUT

se dă 1 form. în CNF

de decis: E o tautologie sau nu?

CALCULABILITATE



$A \in \Sigma_{1<2>} \Leftrightarrow (\exists)\ f \in REC$ a.î. $(\forall)\ x \in \Sigma^*$
$x \in A \Leftrightarrow (\exists)\ y$
$f(x,y) = TRUE$

Includerea pb rec. la o cuantificare

$A = L(M_i)$

$x \in A \Rightarrow M_i(x) \downarrow$

$x \notin A \Rightarrow M_i(x) \uparrow$

$$\boxed{f(x,t)} = \begin{cases} 1, & \text{dc } M_i(x) \text{ acceptă în } t \text{ pași} \\ 0, & \text{altfel} \end{cases}$$

$\boxed{f \text{ recursiv}}$ → simulez $U(\langle i,x\rangle)$ pt t pași ai lui $M_i(x)$; dc ACCEPT ⇒ ret 1; altfel ret. 0

Teoremă (FAGIN): $\boxed{A \in NP}$ $\Leftrightarrow$ există un predicat f calculabil în timp polinomial și există un polinom g a.î. $(\forall) x \in \Sigma^*$, $x \in A$ $\Leftrightarrow$ $(\exists) y$ $\boxed{|y| \leq g(|x|) \quad f(x,y) = TRUE}$

Analogie

$P \longleftrightarrow REC$

$NP \longleftrightarrow R.E. (\Sigma_1)$ $\qquad NP = \exists^{poly} P$

$NP \cap co\text{-}NP$ $\qquad RE \cap co\text{-}RE$

$\overset{!!?}{=}$ $\qquad\qquad \overset{\shortparallel}{REC}$

$A \in NP, \bar{A} \in NP$ $\qquad A \in RE, \bar{A} \in RE$

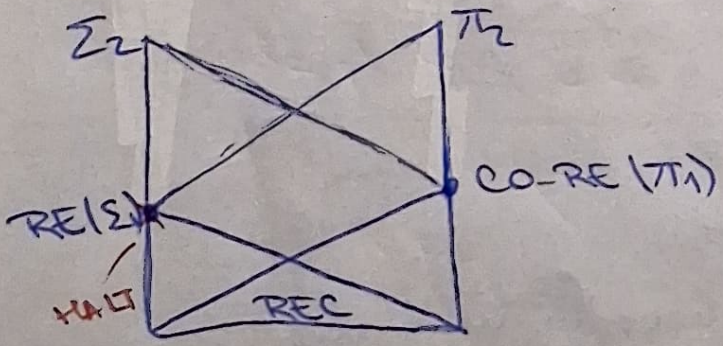
$co\text{-}NP = \{A, \bar{A} \in NP\}$

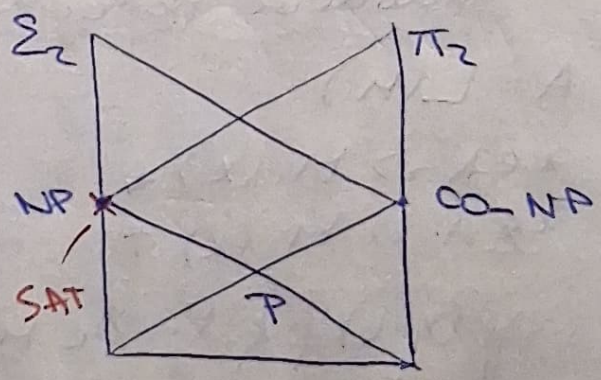
TAUT $\in$ co-NP

co-NP completă

$\boxed{\text{Nu știm}}$ (nu e clar) dacă $\boxed{P = NP \cap co\text{-}NP}$

Analogia





$\Sigma_k^P = \{A \mid$ există un predicat f calculabil în timp polinomial și există un polinom g a.î. $(\forall) x \in \Sigma^*$

$x \in A \iff (\exists) y_1, |y_1| \le g(|x|) (\forall) y_2, |y_2| \le g(|x|)$

$\ldots (Q y_k, |y_k| \le g(|x|))$

cuantif.

$f(x, y_1, y_2, \ldots, y_k) = \text{TRUE}$

$\Pi_k^P = \ldots (\forall) y_1 \ldots) ((\exists) y_2 \ldots)$

$\Sigma_k^P \subset \Sigma_{k+1}^P \cap \Pi_{k+1}^P$

$\Pi_k^P \subset \Sigma_{k+1}^P \cap \Pi_{k+1}^P$

$\boxed{PH}$ (ierarhie polinomială) $\boxed{PH = \bigcup_{k \ge 1} \Sigma_k^P = \bigcup_{k \ge 1} \Pi_k^P}$

Complexity Zoo → 547 clase de complexit def.

O altă perspectivă asupra PH

? Pot rez. pb $\overline{SAT}$ dc am o procedură P pt SAT?

Dc: $\emptyset \to \boxed{P}$

$\downarrow$

$\emptyset \in SAT?$ → DA → NU

→ NU → DA

$A \le_m^P B$ $\quad x \in A \to \square \xrightarrow{f(x)} \boxed{B} \to$ DA/NU → răsp. final

$\boxed{\text{CLIQUE (NP-compl)}}$

INPUT: $G = (V, E)$, $k \ge 1$

De decis: $(\exists) S \subseteq V$, $|S| = k$, S clică $\quad O(n^k)$

$\boxed{\text{MAX-CLIQUE}}$

Se dă: $G = (V, E)$ $\quad S \subset V$

De decis: E S o clică maximală?

Alg 1. S nu e dică $\Rightarrow$ NU

2. Dc $(G, K+1) \in$ CLIQUE? $\Rightarrow$ DA $\rightarrow$ NU

NU $\rightarrow$ DA

Max-clique $(\forall) S'$ $|S'| = K+1$, $(G, S') \notin$ CLIQUE

$\Downarrow$

$(\forall)(\exists) \Rightarrow \Pi_2^P$

$\Updownarrow$

$(\exists) v_1, v_2 \in S'$, $v_1 \not\sim v_2$

M. Turing cu Oracol

1. bandă "oracol"
2. Stări speciale

QUERY
YES
NO

$M(x) \longrightarrow$ QUERY $\boxed{y}$ $\rightarrow$ YES / $\rightarrow$ NO (şterg banda oracol)

$A = L(M, B)$

QUERY $\xrightarrow{y \in B}$ YES

QUERY $\xrightarrow{y \notin B}$ NO

Dc. am o procedură care decide SAT pot găsi o soluție (dc există)

Teoremă: $A \in \Sigma_k^P \Leftrightarrow$ există o M. Turing nedeterministă polinomială cu oracol M şi un oracol $B \in \Sigma_{k-1}^P$ a.î. $A = L(M, B)$

Pb completă pt clasa Σ_k^P

$\exists x_1 \quad \forall x_2 \quad \exists x_3 \quad \forall x_4$

la sfârşit rulez un predicat $p(x_1, \dots, x_k)$ calculabil în timp polinomial care îmi spune cine a câştigat

Se dă: (p, q)

De decis: Are primul jc o strategie de câştig în K paşi?

Exemple de pb care par să fie în PH

1. QBF (Quantified Boolean Formula)

INPUT: $\phi = \exists x_1 \forall x_2 __ Q x_m \; P(x_1 _ x_m)$

De decis: ϕ - Adev / Fals?

2. Se dă: Form. în teoria existențială a nr. reale.

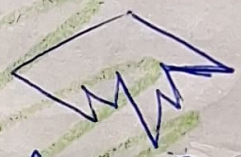
$x^2 > 0 \wedge xy - 1 < z^2 __ \wedge __$

De decis: Pot de val. reale pt x, y, z -- care fac formula adev?

3. ART GALLERY PB.

Se dă: Poligon arbitrar; $K \geq 1$

De decis: Pot plasa K paznici care să păzească (colectiv) interiorul poligonului?



Cum rez. pb din PH?

(1) QBF solvers

(2) Answer Set Programming CLINGO

Curs 13

Pb care (probabil) nu sunt în PH

1. Se dă: sist de ec. polinomiale peste $\mathbb{R}$

$$\begin{cases} x^2 + y - z > 0 \\ x + y + z^2 + t = 3 \end{cases}$$

Să se decidă dacă are sol în $\mathbb{R}$

2. Se dă un poligon convex în plan și $K \geq 1$

Se pot plasa K paznici a.î. să se supraveghieze tot?

Σ_k^P → solvere pt QBF
→ Answer Set Programming (CLINGO)

Clase de complexit. bazate pe spațiu

M mașină Turing

$M(x)$

$C_0 \to C_1 \to \dots \to C_m \to \dots C_n$ → YES / → NO

$\boxed{Space_M(x) = \max\{|C_i|, 1 \le i \le m\}}$

PSPACE = {A | există o maș Turing M a.î.
(i) $L(M) = A$
(ii) $(\forall)x$, $Space_M(x) = O(|x|^k), k>0$ }

Teoremă: $PH \subseteq PSPACE$

Teoremă: QBF e completă pt PSPACE $\le_m^P$

QBF

INPUT: Form booleană cuantificată complet

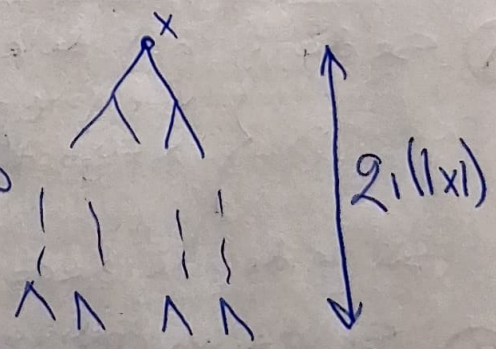
$((\exists)x_1)(\forall)x_2(\exists)x_3 \dots ((Q)x_n)\ P(x_1 \dots x_n)$

De decis: Adevărat sau nu?

Dem $\boxed{\Sigma_2^P \subseteq PSPACE}$

$A \in \Sigma_2^P$ există o M. Turing nedet. polinomială cu oracol M_1 a.î.

- $A = L(M_1, SAT)$
- $(\forall)x$ $time_{M_1}(x) \le q_1(|x|)$ ← polinom



$space_{M_1}(x) \le q_1(|x|)$

SAT $\to M_2$ nedet. în timp $\leq g_2(|x|)$

Creez M care face astfel:

- $M(x)$ simulează $M_1(x)$
- Când $M_1(x)$ pune o întrebare y, M simulează $M_2(y) \longrightarrow |y| \leq g_1(|x|)$

timp $M_1(y) \leq g_2(|y|)$

$\leq g_2(g_1(|x|))$

$\#$ query $\leq g_1(|x|)$

$\Rightarrow$ Sp total $\leq g_1(|x|) \cdot g_2(g_1(|x|))$

$\boxed{NPSPACE = \{A \mid (\exists) M \text{ Turing nedet } M \text{ a.î. } L(M) = A \text{ și } (\forall) x, \text{Space}_M(x) \leq g(x)\}}$

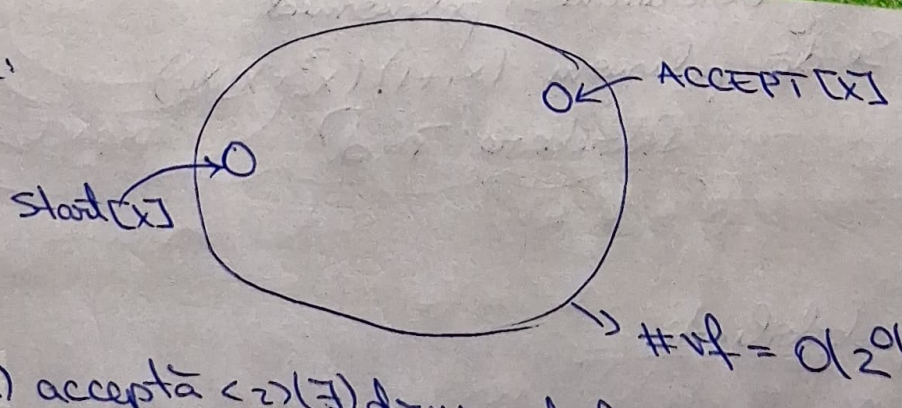
polinom

Teoremă $\boxed{PSPACE = NPSPACE}$

Dem: Arăt că

Teoremă (Savitch) $\boxed{NPSPACE[g(n)] \subseteq PSPACE[g^2(n)]}$

Idee:



$M(x)$ acceptă $\Leftrightarrow (\exists)$ drum de la start[x] la accept[x]

$\overset{C}{\underset{S(n)}{\vdash\!\dashv}}$ $\quad A \xrightarrow{t} B \quad A \xrightarrow{t/2} C$, $C \xrightarrow{t/2} B$

$\overset{D}{\vdash\!\dashv}$ $\quad A \xrightarrow{t} C \quad A \xrightarrow{t/2} D$, $D \xrightarrow{t/2} C$

$O(S(n))$ niveluri de recursie. Pt fiec nivel fol. spațiu $S(n)$

Algo probabiliști

I Alg Monte-Carlo

DA/NU

CORECT: DA → DA, prob ≥ 2/3

CORECT: NU → DA, prob ≤ 1/3

II Alg Las-Vegas

DA/NU/NU ȘTIU

sigur sigur prob ≤ 1/3

Pb PRIMES

Se dă un nr. n scris în binar.

DE DECIS: E n prim sau nu?

Complexit. unei instanțe e dată de nr. de cifre.

Alg. de școală – exponențial!

Alg. probabilist (Miller, Rabin)

p prim

$(a,p)=1 \Rightarrow a^{p-1} \equiv 1 \pmod p$

Din 2003, (∃) alg. determinist (PRIMES ∈ P)

Pb pt care se știe alg probabilist, dar nu se știe alg. determinist polinomial

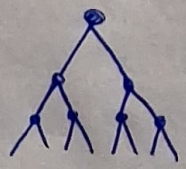
Se dă polinom $(x_1+1)(x_2^3+x_3x_4)$

Întrebare: P=0?

Curs 14

Alg probabiliști

Monte-Carlo



$x \in L \Rightarrow Pr[A(x)=YES] \geq 2/3$

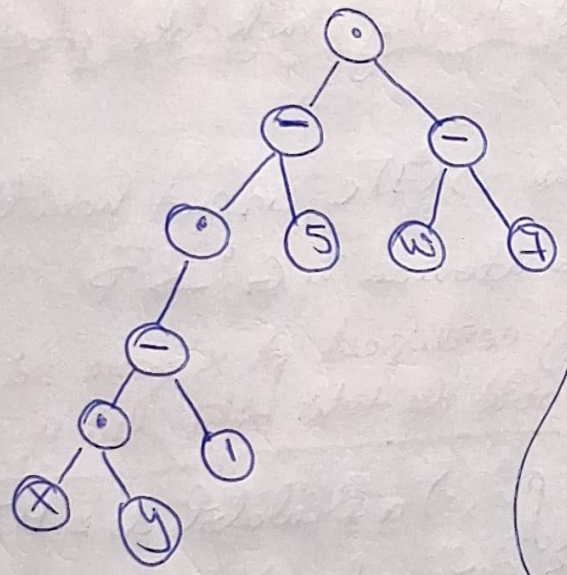
$x \notin L \Rightarrow Pr[A(x)=YES] \leq 1/3$

NP $x \in L \Leftrightarrow Pr[A(x)=YES] > 0$

ex: Polynomial identity testing

se dau: 2 polinoame P, Q

$$P = ((xy-1)(zt+1)-5)\cdot(w-7)$$



De decis: E $P = Q$?

Lema lui Schwartz - Zippel

Fie $p(x_1, \dots, x_n)$ un polinom de grad $\leq d$

Fie S o mulț cu m elem

Dacă: $p(x_1 \dots x_n) \neq 0 \Rightarrow \boxed{\Pr_{z \in S}[p(z) = 0] \leq \frac{d}{m}}$

$\boxed{BPP} = \{L \mid$ există o M. Turing probabilistă cu complexit polinomială a.î.

$(\forall) x, \boxed{T_M(x) \leq q(|x|)}\}$

polinom

$x \in L \Rightarrow \Pr(M(x) \text{ acceptă}) \geq 2/3$

$x \notin L \Rightarrow \Pr(M(x) \text{ acceptă}) \leq 1/3$

1) Alg probabilişti sunt foloositori

2) $P \subset BPP$

Probabil $BPP \not\subset NP$

Probabil $SAT \notin BPP$

$BPP \subset \Sigma_2^P$

Alg. aproximare: $A \to \rho \geq 1$ $\quad \boxed{\rho \leq \sup \frac{C_A(x)}{C_{OPT}(x)}}$

ex: VC are alg 2-aproximativ

Teorema PCP (probabilistically checkable proofs)

idee NP

G - graf hamiltonian $\Rightarrow (\exists)$ c = circuit hamiltonian în G

"C e circuit hamiltonian în G" $\to$ P

↳ verif că y certificat pt x pare să aibă nevoie de toți biții lui y

$L \in NP \Leftrightarrow (\exists)$ predicat $f(\cdot,\cdot)$ calculabil de o M. Turing probabilistă care rulează în timp polinomial a.î.

1) y martor pt $x \Rightarrow Pr(M(x,y) \text{ acceptă}) = 1$
2) y nu e martor pt $x \Rightarrow Pr(M(x,y) \text{ acc}) \leq 1/3$
3) M fol. $O(\log n)$ biți, $n = |x| + |y|$ și accesează $O(1)$ biți ai lui y

Clase de complexit în P

$\boxed{NL}$ (nondeterministic logspace)

Pb.: se dă: G=(V,E) graf orientat și 2 vf: s,t ∈ V $\quad$ s,t $\to$ nr cu $(\log n)$ biți ↳ nr vf.

de decis: $s \xrightarrow{*} t$ în G

Ghicesc s_2 ($\log_n$ biți); verific $s \to s_2$ ($O(1)$ spațiu) și repet

Alg. care ghicește un drum de la s la t și verif. $s_i \to s_{i+1}$ poate fi implem fol. spațiu $O(\log n)$

$\boxed{NL = \{A \mid (\exists) \text{M. Turing nedet. cu spațiu } O(\log n) \text{ care decide pe } A\}}$

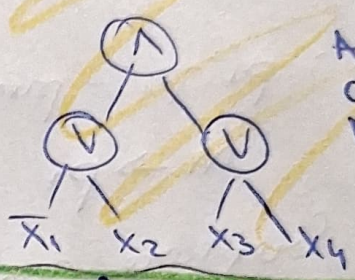
STCONN – cea mai grea pb din NL (completă)

Complexit circuitelor booleene

$f: \{0,1\}^n \to \{0,1\}$

ex: $f(x_1, \dots, x_n) = x_1 \oplus x_2 \oplus \dots \oplus x_n$ paritate

$c(f)$ = nr. porți



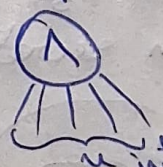
AND
OR
NOT

Teoremă (Shannon): Aproape toate fct booleene cu n variabile NU pot fi calc. de circuite cu $\leq 2^n/n$ porți.

Nu știm niciun ex. concret de fct booleană care să necesite $\geq 4n$ porți ($3{,}01\,n$)

Constrângeri suplimentare

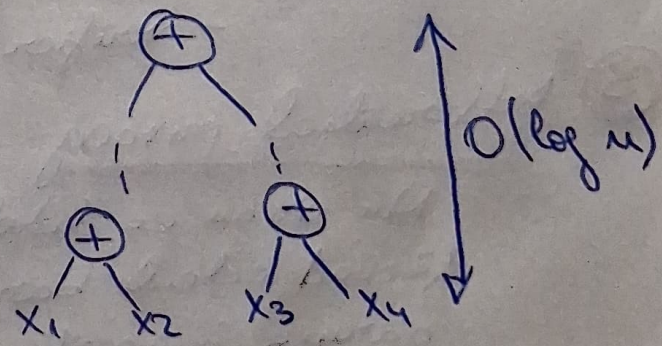
– înălțimea circuitelor să fie $O(1)$ dau voie doar ANDn, ORn



$AC^0 = \{f \mid f$ calculabilă de un circuit booleean cu adâncime $O(1)$ și # porți – poly$(n)\}$

Teoremă (Hastad) PARITY $\notin AC^0$

$x_1 \oplus \dots \oplus x_n$



Unde e imp complexitatea?

1. Criptografie → fct one-way & criptosisteme cu chei publice

fct one-way = fct ușor de calculat, dar greu de inversat

$x \to f(x) = y$ ușor
$y \to x$ greu

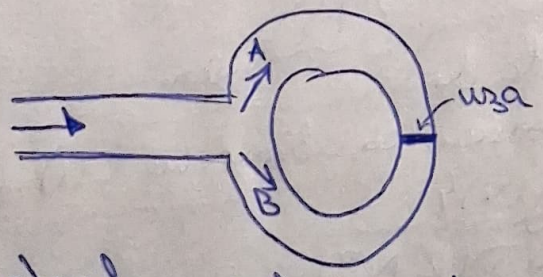
ex: p_1, p_2 – prime → $n = p_1 \cdot p_2$
$n \to p_1$ greu (FACTORING)

Criptare: am nevoie de n
Decriptare: am nevoie de p_1 și p_2

2. Zero-Knowledge

- protocol care convinge un verifier că știe un secret
- verifier-ul nu învață nimic despre secret rulând protocolul meu

ex: pp că există o ușă pe care să o deschid; am nevoie de un secret



Protocol: – intru pe A sau B
– verifier-ul alege la întâmplare A sau B

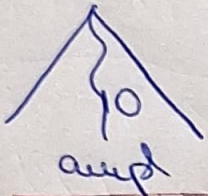
3. Differential privacy

Nu vreau ca datele mele să poată fi revelate făcând calcule bazate pe atributele unui grup căruia îi aparțin

9 Calcul cuantic

N Alg probabilisti DAR prob. sunt calc. diferite.
"Dau cu banul" cu amplitudini complexe

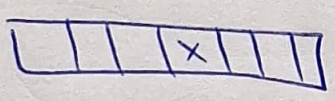
$\alpha, \beta \in \mathbb{C}$, $\alpha |0\rangle + \beta |1\rangle$ $\alpha, \beta \in \mathbb{C}$



$Pr(\text{eveniment}) = \| \ \|^2$

Teoremă (Shor): Factoring poate fi rez. în timp polin. pe un calc. cuantic.

Teoremă (Grover)



Cuantic: $x \rightarrow O(\sqrt{n})$ operații

Realitate: calc cuantice pe 1000 biți nu e clar dacă scalează

dovadă că nu putem rez. pb NP-compl. în timp polin. cu un alg. cuantic.