Az informatika számítástudományi alapjai
9. előadás

Vaszil György

vaszil.gyorgy@inf.unideb.hu

I. emelet 110-es szoba

Múltkor

- Turing gépek, sztringek elfogadása Turing géppel, elfogadott nyelv
- Több szalagos Turing gépek, nemdeterminisztikus Turing gépek
- Rekurzív és rekurzívan felsorolható nyelvek
- Általános (szabály-)alakú grammatikák és Turing gépek

Emlékeztető

Miről szól ez az egész?

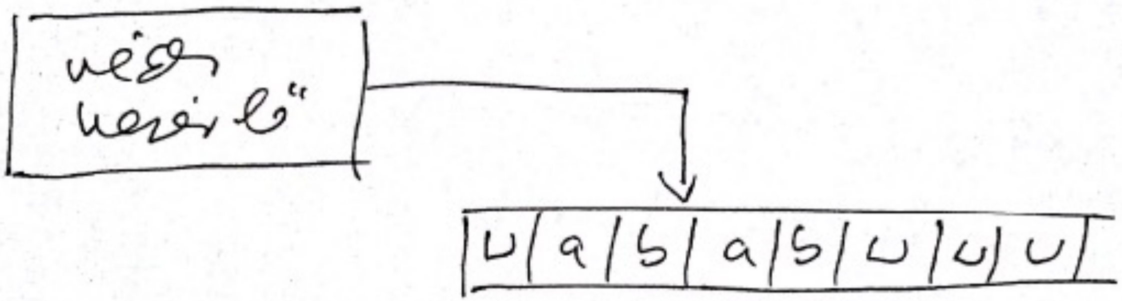
A számítógépek képességeiről és a képeségek ***elvi*** korlátairól.

- A számítógépek modellezése, a **számítási folyamat “hardverfüggetlen”) leírása** (automaták)
- A számítógéppel ***elvileg is*** megoldhatatlan feladatokról (kiszámíthatóság)
  - Mikor megoldható egy feladat?
- A számítógéppel elvileg megoldható feladatok **gyakorlati megoldhatósága** (számítási bonyolultság)
  - Mitől **egyszerű** és mitől **nehéz** egy feladat megoldása?

Múltkor

- Turing gépek, sztringek elfogadása Turing géppel, elfogadott nyelv
- Több szalagos Turing gépek, nemdeterminisztikus Turing gépek
- Rekurzív és rekurzívan felsorolható nyelvek
- Általános (szabály-)alakú grammatikák és Turing gépek

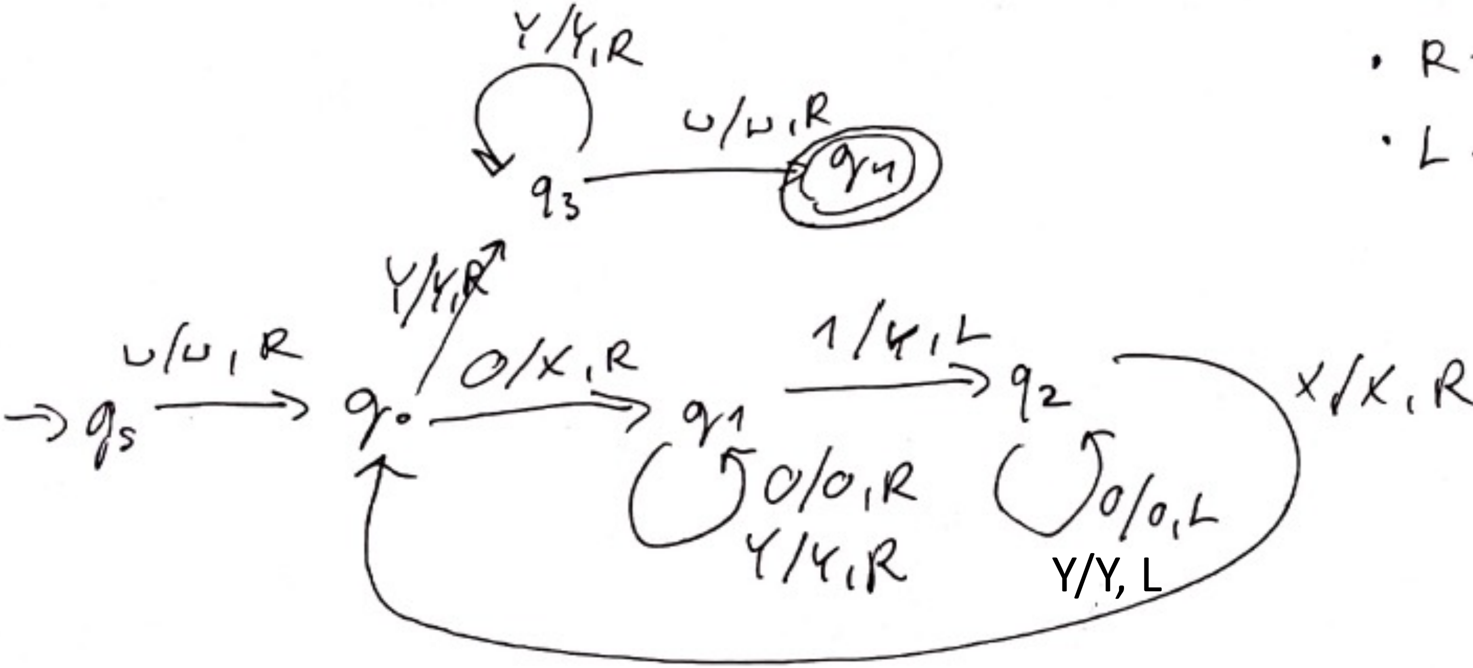
Turing gépek



1. A gép írhat és olvashat is a szalagról
2. Az író-olvasó fej jobbra és balra is mozoghat
3. A szalag végtelen
4. speciális állapot: elfogadás

Például: Turing gép 0^n1^n alakú szavak felismerésére

- ⊔ vagy Δ üres szalag
- R - right jobbra
- L - left balra



kezdő konfiguráció: … ⊔ | ⊔ | ⊔ | 0 | 0 | 1 | 1 | ⊔ | ⊔ …

$\uparrow$ q_s

Gyakorlaton: 00011, 00111

Tehát: A T Turing gép által elfogadott nyelv L(T).

- Ha $w \in L(T)$, akkor w bemenetre T **elfogadó állapotban áll meg**,
- Ha $w \notin L(T)$, akkor w bemenetre T **nem-elfogadó állapotban** áll meg, vagy **egyáltalán nem áll meg**.

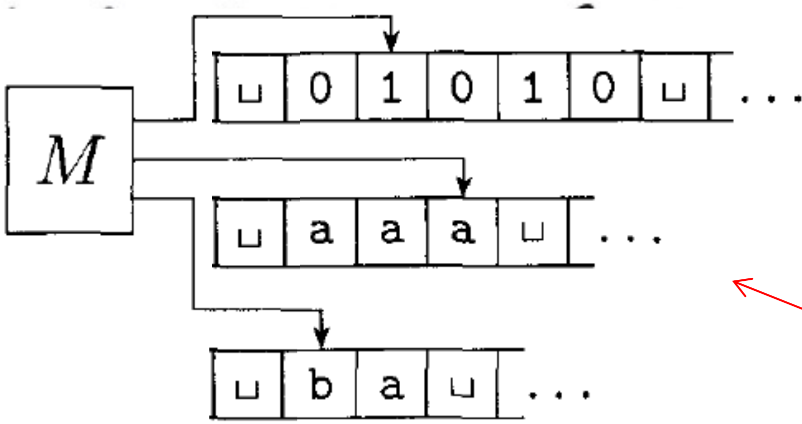
Múltkor

- Turing gépek, sztringek elfogadása Turing géppel, elfogadott nyelv
- Több szalagos Turing gépek, nemdeterminisztikus Turing gépek
- Rekurzív és rekurzívan felsorolható nyelvek
- Általános (szabály-)alakú grammatikák és Turing gépek

Variációk Turing gépre

- Egy szalag helyett k szalag, az input az első szalagon van.

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k$$



Ezen a példán k=3

k szalagos és 1 szalagos Turing gépek

Tétel: Minden k szalagos T_1 Turing géphez létezik olyan 1 szalagos T_2 Turing gép, hogy $L(T_1) = L(T_2)$

Variációk Turing gépre

- Nemdeterminisztikus Turing gép

$$\delta: Q \times \Gamma \to 2^{Q \times \Gamma \times \{L, R\}}$$

A Nemdeterminisztikus Turing gép szimulálható determinisztikus Turing géppel.

Azaz: Minden nemdeterminisztikus Turing gép **átalakítható determinisztikussá** úgy, hogy ne változzon meg a **nyelv amit elfogad**.

Múltkor

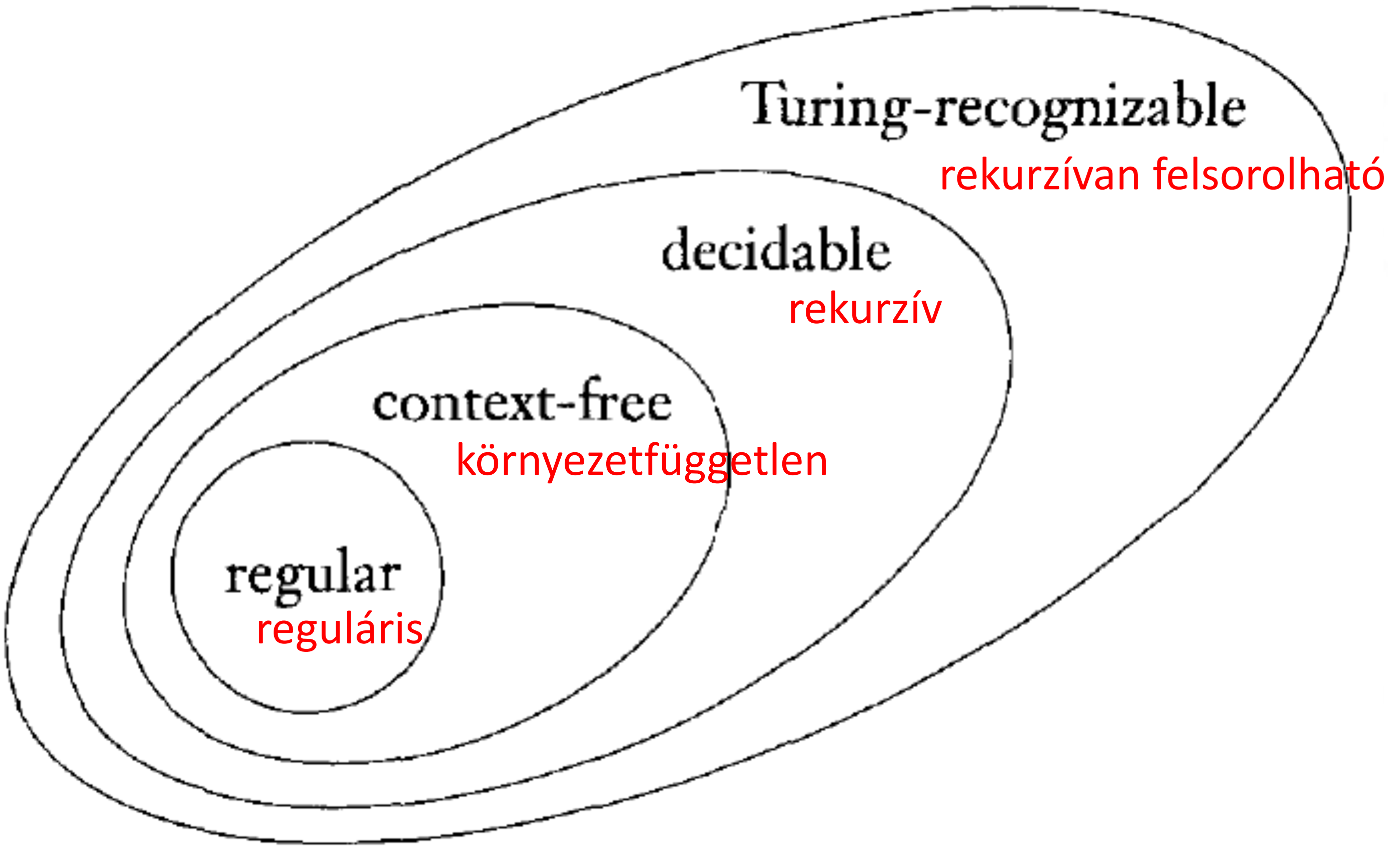
- Turing gépek, sztringek elfogadása Turing géppel, elfogadott nyelv
- Több szalagos Turing gépek, nemdeterminisztikus Turing gépek
- Rekurzív és rekurzívan felsorolható nyelvek
- Általános (szabály-)alakú grammatikák és Turing gépek

Rekurzívan felsorolható és rekurzív nyelvek

- Egy L nyelv rekurzívan felsorolható, ha van olyan Turing gép, ami minden $x \in L$ szó bemeneten elfogadó állapotban áll meg. A $x \notin L$ szavakra vagy nem elfogadó állapotban áll meg, vagy egyáltalán nem áll meg. -azaz, ha $L=L(T)$ egy T Turing gépre, vagyis ha **T elfogadja** L-et.
- Egy L nyelv rekurzív, ha van olyan Turing gép ami minden bemeneten megáll, a $x \in L$ szavakon elfogadó állapotban áll meg, a $x \notin L$ szavakon nem elfogadó állapotban áll meg.

Ha van ilyen T Turing gép, akkor **T eldönti** L-et.

Láttuk, látni fogjuk



reguláris, környezetfüggetlen, rekurzív (Turing géppel eldönthető), rekurzívan felsorolható (Turing géppel elfogadható) nyelvek

Múltkor

- Turing gépek, sztringek elfogadása Turing géppel, elfogadott nyelv
- Több szalagos Turing gépek, nemdeterminisztikus Turing gépek
- Rekurzív és rekurzívan felsorolható nyelvek
- Általános (szabály-)alakú grammatikák és Turing gépek

Általános alakú grammatikák

$G=(V,\Sigma,S,P)$ ahol P szabályai

$$\alpha \to \beta$$

alakúak, $\alpha, \beta \in (V \cup \Sigma)^*$, α tartalmaz nemterminálist

Tézis :

Az általános alaki nyelvtanok által generálható nyelvek megegyeznek a Turing géppel elfogadható nyelvekkel

L nyelv „elfogadása” T-vel: ha w szó L-beli, akkor T elfogadó állapotban áll meg w-

különben: T nem-elfogadó állapotban áll meg, vagy

T egyáltalán nem áll meg

Elnevezés : Rekurzívan felsorolható nyelvek

Rekurzívan felsorolható grammatikák

Múltkor

- Turing gépek, sztringek elfogadása Turing géppel, elfogadott nyelv
- Több szalagos Turing gépek, nemdeterminisztikus Turing gépek
- Rekurzív és rekurzívan felsorolható nyelvek
- Általános (szabály-)alakú grammatikák és Turing gépek

Ma

- Környezetfüggő grammatikák, a Chomsky féle nyelv-hierarchia
- Az univerzális Turing gép
- A Turing gép mint az *algoritmus* fogalmának matematikai megfogalmazása, a Church tézis
- Problémák mint formális nyelvek, algoritmikusan megoldható problémák és rekurzív nyelvek
- Algoritmikusan megoldhatatlan problémák, a megállási probléma eldönthetetlensége, egy konkrét nem rekurzívan felsorolható nyelv

Mi az algoritmus fogalmának „nem matematikai” megfogalmazása?

Környezetfüggő nyelvek

$G=(V,\Sigma,S,P)$ környezetfüggő, ha P szabályai

$$\alpha \to \beta, \text{ akkor } |\alpha| \leq |\beta|$$

alakúak.

L környezetfüggő, ha generálja környezetfüggő grammatika

(Ha λ benne van egy környezetfüggő L-ben, akkor egyetlen törlő szabály, S→ λ megengedett.)

Mit tudunk a nyelvek hierarchiájáról:

$$\mathcal{L}(REG) \subset \mathcal{L}(CF) \subseteq \mathcal{L}(CS) \subseteq \mathcal{L}(RE)$$

pumpálási
lemma

- Miért teljesül a tartalmazás reláció?
- „Szigorú”-e a tartalmazás?

<u>Jelölések</u>:

- reguláris – REG
- környezetfüggetlen – CF (context-free)
- környezetfüggő – CS (context-sensitive)
- rekurzívan felsorolható – RE (recursively enumerable)

$\mathcal{L}(CF) \in \mathcal{L}(CS)$

1. $\{a^n b^n c^n\} \notin \mathcal{L}(CF)$ ← a pumpálási lemma szerint

2. $\{a^n b^n c^n\} \in \mathcal{L}(CS)$ ← generálható környezet függő grammatikával

Pl. G=({S,B,C}, {a,b,c}, S, P), ahol

P={ S→aSBC, S→abC, CB→BC, B→bb, bC→bc, cC→cc }

Mit tudunk a nyelvek hierarchiájáról:

$$\mathcal{L}(REG) \subset \mathcal{L}(CF) \subset \mathcal{L}(CS) \subseteq \mathcal{L}(RE)$$

pumpálási lemma
reguláris nyelvekre

pumpálási lemma
környezetfüggetlen nyelvekre

Miért teljesül ez a
tartalmazás? (A múlt
héten láttuk)

A Chomsky féle hierarchia

$REG \subset CF \subset CS \subseteq REC \subseteq RA \subseteq$ a lehetséges nyelvek

?

?

?

pumpálási lemma reguláris nyelvekre:

$\{a^n b^n \mid n \geq 1\} \in CF$

$\notin REG$

pumpálási lemma környezetfüggetlen nyelvekre:

$\{a^n b^n c^n \mid n \geq 1\} \in CS$

$\notin CF$

Jelölés:

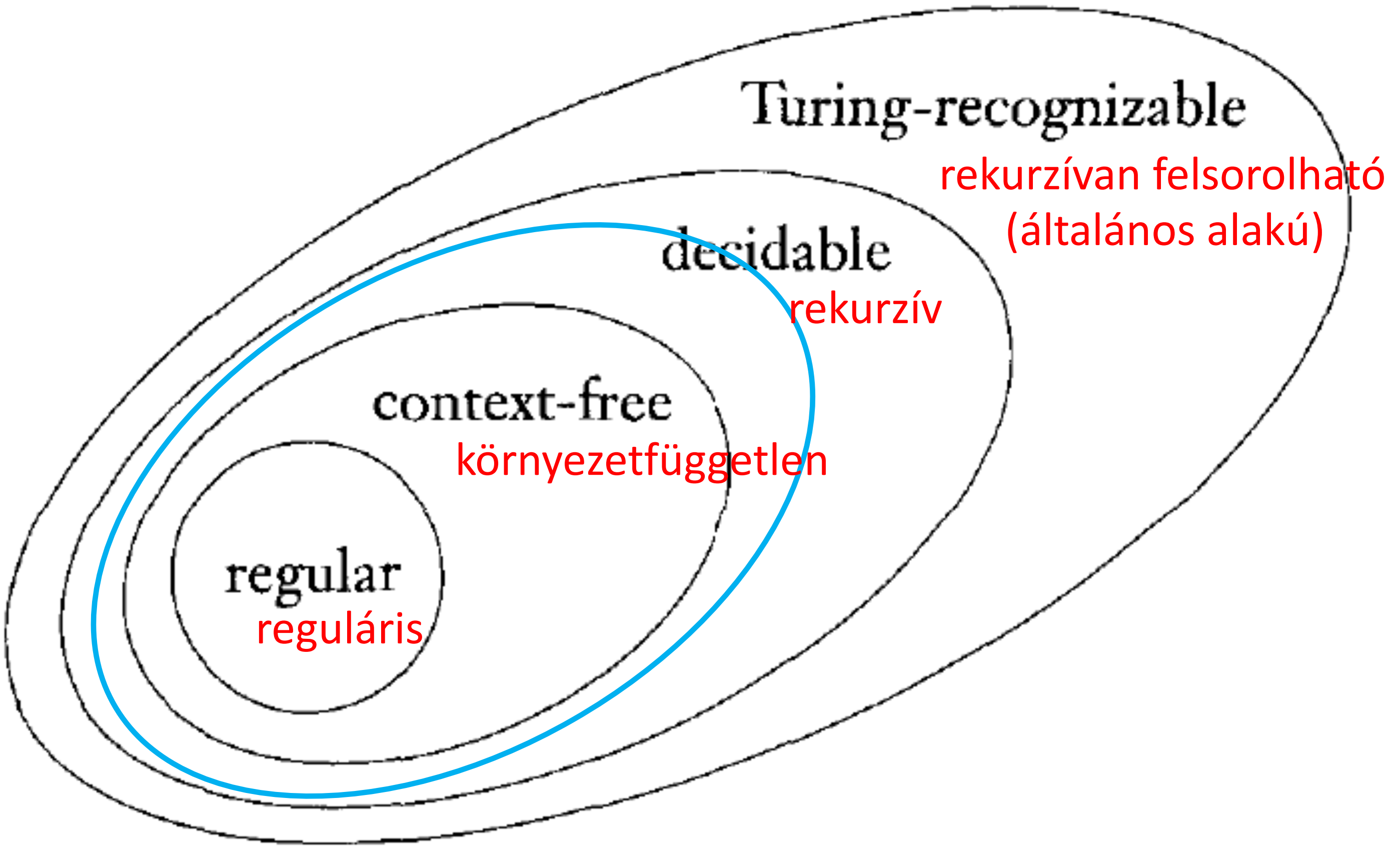
- REC - rekurzív

A Chomsky-féle hierarchia

$\mathrm{REG} \subset \mathrm{CF} \subset \mathrm{CS} \subset \mathrm{REC} \subset \mathrm{RE} \subset$ a lehetséges gépek

„ex cathedra"

Hamarosan
látni fogjuk



reguláris, környezetfüggetlen, környezetfüggő,
rekurzív, rekurzívan felsorolható

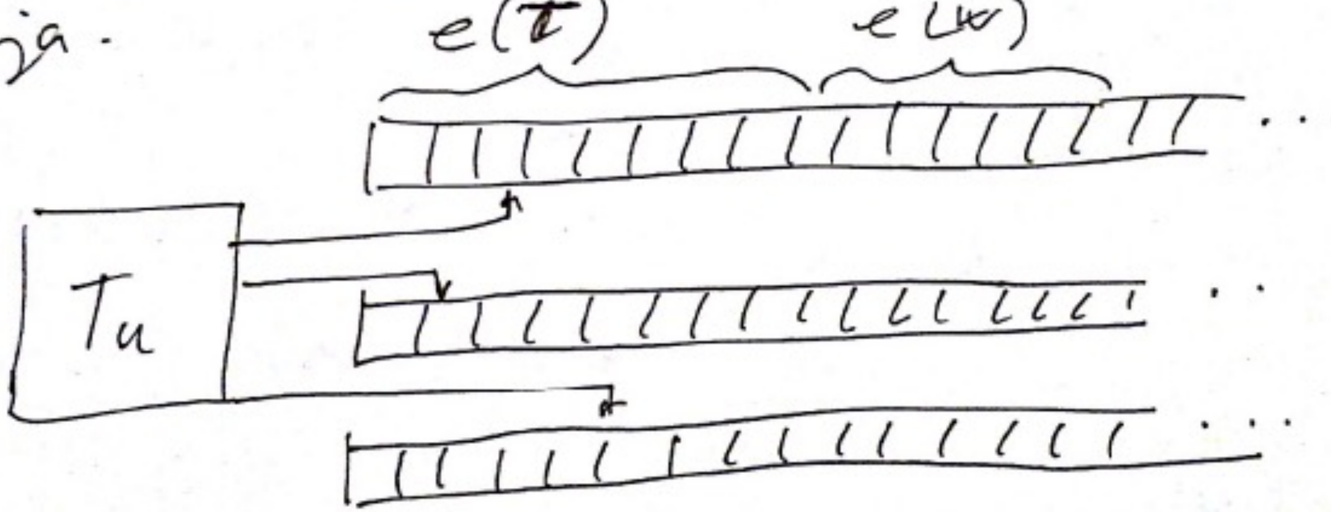
Ma

- Környezetfüggő grammatikák, a Chomsky féle nyelv-hierarchia
- Az univerzális Turing gép
- A Turing gép mint az *algoritmus* fogalmának matematikai megfogalmazása, a Church tézis
- Problémák mint formális nyelvek, algoritmikusan megoldható problémák és rekurzív nyelvek
- Algoritmikusan megoldhatatlan problémák, a megállási probléma eldönthetetlensége, egy konkrét nem rekurzívan felsorolható nyelv

Mi az algoritmus fogalmának „nem matematikai” megfogalmazása?

Univerzális Turing gép

Egy olyan T_u Turing gép, amelyik tetszőleges másik Turing gép működését szimulálni tudja.



Az első szalagra T kódját (e(T)) és w kódját (e(w)) írva, T_u szimulálja T működését w-n.

(T és w kódját néha <T> és <w> is fogja jelölni)

Hogy nézhet ki egy kódolás

string of 0's. Let

$$\left.\begin{array}{l} s(\Delta) = 0 \\ s(a_i) = 0^{i+1} \quad \text{(for each } a_i \in \mathcal{S}) \end{array}\right\} \text{szalag ábécé}$$

$$\left.\begin{array}{l} s(h_a) = 0 \\ s(h_r) = 00 \\ s(q_i) = 0^{i+2} \quad \text{(for each } q_i \in \mathcal{Q}) \end{array}\right\} \text{vezérlő-állapotok}$$

$$\left.\begin{array}{l} s(\mathrm{S}) = 0 \\ s(\mathrm{L}) = 00 \\ s(\mathrm{R}) = 000 \end{array}\right\} \text{fejmozgás irányok}$$

Each move m of a TM, described by the formula

$$\delta(p, a) = (q, b, D)$$

is encoded by the string

$$e(m) = s(p)1s(a)1s(q)1s(b)1s(D)1$$

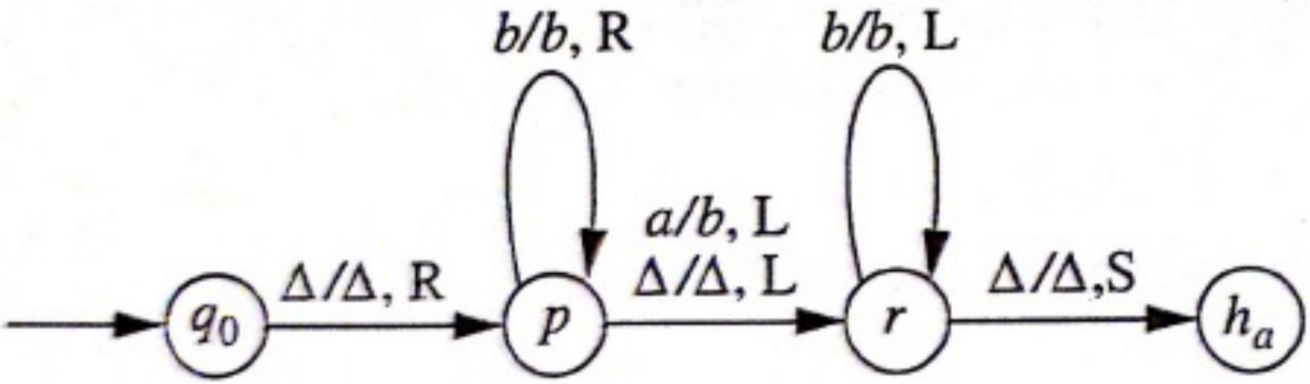
and for any TM T, with initial state q, T is encoded by the string

$$e(T) = s(q)1e(m_1)1e(m_2)1 \cdots e(m_k)1$$

where $m_1, m_2, \cdots, m_k$ are the distinct moves of T, arranged in some arbitrary order. Finally, any string $z = z_1 z_2 \cdots z_k$, where each $z_i \in \mathcal{S}$, is encoded by

$$e(z) = 1s(z_1)1s(z_2)1 \cdots s(z_k)1$$

Rèldail



0001 0001010000101000110000100010000100010001100001001000001000100110000101000001010011000001000100000010001001100000101010101011 string of 0's. Let

$$s(\Delta) = 0$$
$$s(a_i) = 0^{i+1} \quad (\text{for each } a_i \in \mathcal{S})$$
$$s(h_a) = 0$$
$$s(h_r) = 00$$
$$s(q_i) = 0^{i+2} \quad (\text{for each } q_i \in Q)$$
$$s(\mathrm{S}) = 0$$
$$s(\mathrm{L}) = 00$$
$$s(\mathrm{R}) = 000$$

Each move m of a TM, described by the formula

$$\delta(p, a) = (q, b, D)$$

is encoded by the string

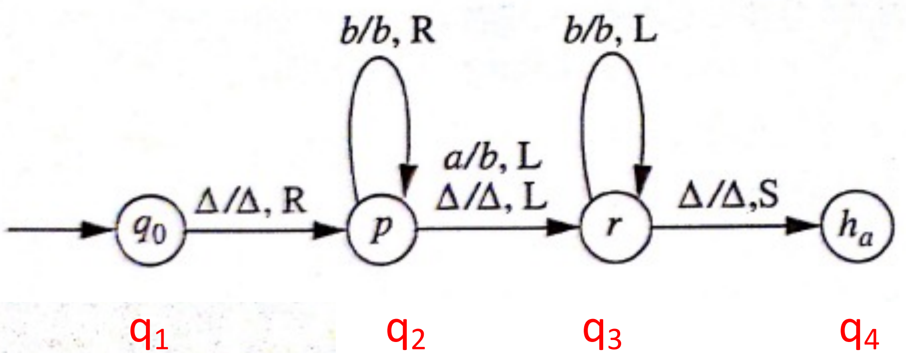
$$e(m) = s(p)1s(a)1s(q)1s(b)1s(D)1$$

and for any TM T, with initial state q, T is encoded by the string

$$e(T) = s(q)1e(m_1)1e(m_2)1 \cdots e(m_k)1$$

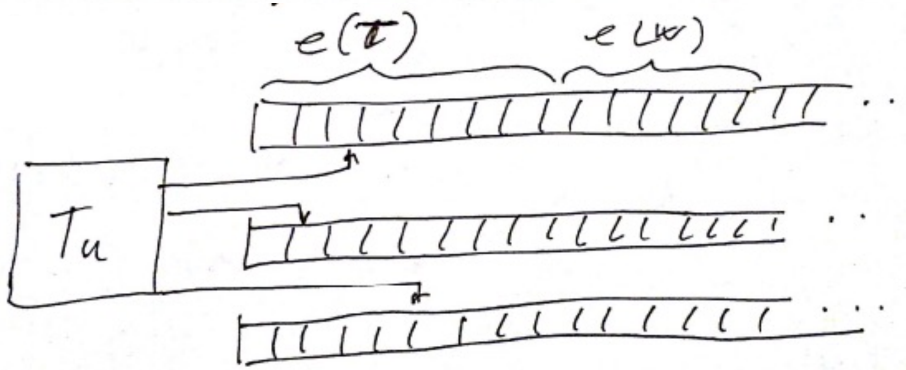
where $m_1, m_2, \cdots, m_k$ are the distinct moves of T, arranged in some arbitrary order. Finally, any string $z = z_1 z_2 \cdots z_k$, where each $z_i \in \mathcal{S}$, is encoded by

$$e(z) = 1s(z_1)1s(z_2)1 \cdots s(z_k)1$$



0001 000101000010100011 00001000100001000100011 0000100100000100010011
0000101000001010011 000001000100000100010011 000001010101011

Tu működése



1. Tu átmásolja e(w)-t a 2. szalagra, T kezdőállapotát pedig a 3. szalagra
2. A 2. szalagról leolvassa az aktuális betűt, a 3. szalagról az aktuális állapotot.
3. Ezek alapján megkeresi az 1. szalagon az ehhez tartozó állapot átmenetet
4. Ebből kiolvassa az új állapotot és azt, hogy mit kell a 2. szalagra írni

Ma

- Környezetfüggő grammatikák, a Chomsky féle nyelv-hierarchia
- Az univerzális Turing gép
- A Turing gép mint az *algoritmus* fogalmának matematikai megfogalmazása, a Church tézis
- Problémák mint formális nyelvek, algoritmikusan megoldható problémák és rekurzív nyelvek
- Algoritmikusan megoldhatatlan problémák, a megállási probléma eldönthetetlensége, egy konkrét nem rekurzívan felsorolható nyelv

Mi az algoritmus fogalmának „nem matematikai” megfogalmazása?

A Turing gép mint „számítógép modell" – számítási modell

1. A Turing gép általános célú számítási modell
2. Ráadásul programozható
3. Matematikailag precíz

⇒ Alkalmasnak tűnik a „számítás" másnéven az „algoritmus" fogalmának formalizálására

Azaz

A Turing gép által már az intuitív algoritmus-fogalom formalizálására.

precíz szabályok alapján, mechanikusan végrehajtható eljárás, ami véges sok lépésben eredményre vezet.

Church - Turing tézis

Alonzo Church logikus, matematikus
Alan Turing matematikus, XX. század első fele, közepe

„A Turing gép mindent tud", azaz, ha van egy problémára „mechanikus eljárással" véges sok lépésben megoldás (algoritmus), akkor egy Turing gép is meg tudja oldani.

Ha egy problémát Turing géppel nem lehet megoldani, akkor nincs „mechanikus eljárás" nincs algoritmus sem. (azaz nem lehet rá programot írni)

Ma

- Környezetfüggő grammatikák, a Chomsky féle nyelv-hierarchia
- Az univerzális Turing gép
- A Turing gép mint az *algoritmus* fogalmának matematikai megfogalmazása, a Church tézis
- Problémák mint formális nyelvek, algoritmikusan megoldható problémák és rekurzív nyelvek
- Algoritmikusan megoldhatatlan problémák, a megállási probléma eldönthetetlensége, egy konkrét nem rekurzívan felsorolható nyelv

Mi köze a Turing gépeknek a problémamegoldáshoz?

„Eldöntési" problémák: Amire igen - nem válasz adható.

Pl.: Egy algoritmus benne van-e egy adott halmazban.

Konkrétan pl.: $(\mathbb{N}, M)$ ← $M \subseteq \mathbb{N}$ vagy (Σ^*, L)

Bemenet: n

Kimenet: YES, ha $n \in M$

NO, ha $n \notin M$

Például

$$PRIM = \{ 2, 3, 5, 7, 11, 13 \dots \}$$

A $(\mathbb{N}, PRIM)$ feladat azt kéri, döntsük el egy adott bemenő n számról, hogy prím-e.

Ha van olyan algoritmus, ami megoldja $(\mathbb{N}, M)$-et, akkor $(\mathbb{N}, M)$

- <u>algoritmikusan megoldható</u>
- <u>eldönthető</u>

<u>Pl</u>: $(\mathbb{N}, \mathbb{N}_{páros})$ eldönthető

$(\mathbb{N}, PRIM)$ eldönthető

A Turing gép futása nem biztos, hogy befejeződik

Ezek szerint:

egy Turing gép elfogadja M halmazt, ha:

- $u \in M$ bemenetre elfogadó állapotban áll meg
- $u \notin M$ bemenetre nem-elfogadó állapotban áll meg

  vagy „végtelen ideig” dolgozik

Rekurzívan felsorolható és rekurzív nyelvek

- Egy L nyelv <u>rekurzívan felsorolható</u>, ha van olyan Turing gép, ami minden $x \in L$ szó bemeneten elfogadó állapotban áll meg. A $x \notin L$ szavakra vagy nem elfogadó állapotban áll meg, vagy egyáltalán nem áll meg. -azaz, ha L=L(T) egy T Turing gépre, vagyis ha **T elfogadja** L-et.
- Egy L nyelv <u>rekurzív</u>, ha van olyan Turing gép ami minden bemeneten megáll, a $x \in L$ szavakon elfogadó állapotban áll meg, a $x \notin L$ szavakon nem elfogadó állapotban áll meg.

Ha van ilyen T Turing gép, akkor **T eldönti** L-et.

Tehát:

Egy probléma algoritmikusan megoldható, ha van olyan Turing, ami **eldönti** a hozzá tartozó nyelvet (mindig megáll és IGEN-t és NEM-et válaszol a megfelelő módon).

Tehát:

Egy probléma algoritmikusan megoldható, ha van olyan Turing, ami **eldönti** a hozzá tartozó nyelvet (mindig megáll és IGEN-t és NEM-et válaszol a megfelelő módon).

Azaz:

Egy probléma algoritmikusan megoldható, ha a hozzá tartozó nyelv **rekurzív**.

Ma

- Környezetfüggő grammatikák, a Chomsky féle nyelv-hierarchia
- Az univerzális Turing gép
- A Turing gép mint az *algoritmus* fogalmának matematikai megjelenése, a Church tézis
- Problémák mint formális nyelvek, algoritmikusan megoldható problémák és rekurzív nyelvek
- Algoritmikusan megoldhatatlan problémák, a megállási probléma eldönthetetlensége, egy konkrét nem rekurzívan felsorolható nyelv

Még egy példa: Hilbert 10. problémája

Létezik-e „eljárás" ami eldönti, hogy egy egész együtthatós polinomnak van-e egész gyöke.

Pl.: $6x^3yz^2+3xy^2-x^3-10$

egész gyöke: $x=5,\ y=3,\ z=0$

Hilbert problémája mint formális nyelv, a polinomok mint sztringek

Az ábécé: *{0, 1, 2, ..., 9, x, +, -, *}*

- a számok 10-es számrendszerben
- a változók: $x_1, x_2, \ldots$
- a műveleti jelek: +, -, *

Pl.:

$$6x^2 y z^2 + 3x y^2 - x^3 - 10 \longleftrightarrow 6x_1^2x_2x_3^2+3x_1x_2^2-x_1^3-10$$

A polinomot leíró sztring:

6 x1*x1* x2* x3*x3 + 3* x1* x2*x2 - x1*x1*x1 - 10*

Hilbert tizedik tétele:

$L_p = \{<p> \mid p$ polinomnak van egész gyöke$\}$

L_p rekurzív? Jurij Matijasevics orosz matematikus mutatta meg 1970-ben, hogy L_p nem rekurzív.

Egyszerűen látható viszont, hogy L_p rekurzívan felsorolható

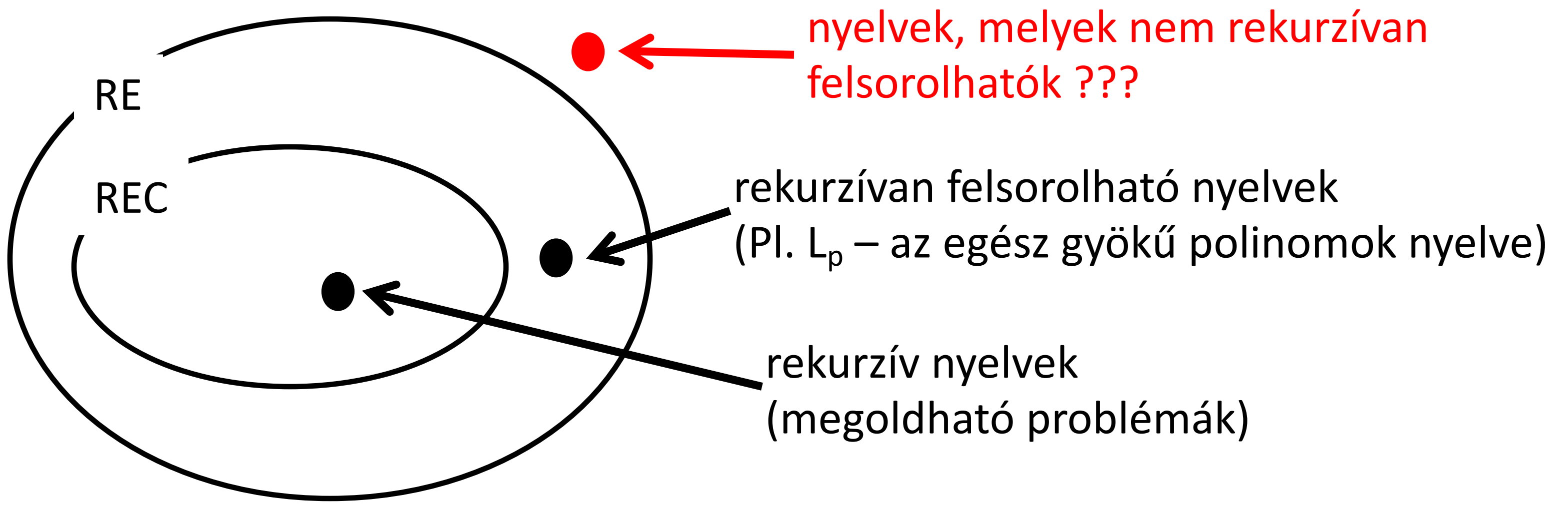
Az egész gyökű polinomok nyelve rekurzíve felsorolható

Adott egy P polinom k változóval. Veszünk egy Turing gépet, ami végigmegy az összes ~~az~~ egész szám k-ason, és kipróbálja, hogy gyöke-e P-nek. ~~Ha igen, akkor~~ Ha talál ilyet, elfogadja P-t.

(Eldönteni nem fogja tudni)

Eddig láttuk – Az algoritmikus kiszámíthatóság/megoldhatóság határai

- Church tézis – Az algoritmikusan megoldható feladatok pontosan azok, amelyek Turing gépekkel *eldönthetők.*
- *„eldönthető”* : a hozzá tartozó nyelv rekurzív



Ma

- Környezetfüggő grammatikák, a Chomsky féle nyelv-hierarchia
- Az univerzális Turing gép
- A Turing gép mint az *algoritmus* fogalmának matematikai megjelenése, a Church tézis
- Problémák mint formális nyelvek, algoritmikusan megoldható problémák és rekurzív nyelvek
- Algoritmikusan megoldhatatlan problémák, a megállási probléma eldönthetetlensége, egy konkrét nem rekurzívan felsorolható nyelv

<u>Megoldhatatlan / eldönthetetlen problémák</u>

<u>Egy problémakör</u>: Ellenőrizhető-e a programok helyessége?

Van-e olyan algoritmus / program ami tetszőleges másik programról megmondja, hogy <u>helyesen működik</u>-e?

(Nincs) Sőt

megáll-e adott bemeneten?

A megállási probléma

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ elfogadja } w\text{-t} \}$$

1. A_{TM} Turing-felismerhető / rekurzív felsorolható:
   - Univerzális Turing gép: $\langle M, w \rangle$ bemenete szimulálja M működését w-n.
     - megállhat $w \in L(M)$, $w \notin L(M)$
     - végtelen ciklusba kerülhet $w \notin L(M)$ 4

$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ elfogadja } w\text{-t} \}$$

Tegyük fel, hogy A_{TM} rekurzív.

→ Létezik H, hogy

$$H(\langle M, w \rangle) = \begin{cases} \text{elfogad, ha } M \text{ elfogadja } w\text{-t} \\ \text{elutasít, ha } M \text{ nem fogadja el } w\text{-t} \end{cases}$$

Az “<, >” zárójelek a kódolásra utalnak: ha X egy objektum, akkor <X> jelöli az objektumot kódoló sztringet

Konstruáljunk egy D Turing gépet, ami ellenkezőképp viselkedik mint H:

D, bemenete $\langle M \rangle$:

1. Szimulálja H-t $\langle M, \langle M \rangle \rangle$ bemeneten
2. Elfogadja $\langle M \rangle$-et, ha H nem fogadja el $\langle M, \langle M \rangle \rangle$-et
3. Elutasítja $\langle M \rangle$-et, ha H elfogadja $\langle M, \langle M \rangle \rangle$-et.

Az “<, >” zárójelek a kódolásra utalnak: ha X egy objektum, akkor <X> jelöli az objektumot kódoló sztringet

Kérdés: Mi van, ha D-t $\langle D \rangle$ bemenettel indítjuk?

$$D(\langle M \rangle) = \begin{cases} \text{elutasítás}, & \text{ha } M \text{ elfogadja } \langle M \rangle\text{-et} \\ \text{elfogadás}, & \text{ha } M \text{ nem fogadja el } \langle M \rangle\text{-et} \end{cases}$$

$$\Downarrow$$

$$D(\langle D \rangle) = \begin{cases} \text{elutasítás}, & \text{ha } D \text{ elfogadja } \langle D \rangle\text{-t} \\ \text{elfogadás}, & \text{ha } D \text{ nem fogadja el } \langle D \rangle\text{-t} \end{cases}$$

$\Rightarrow$ azaz D nem létezhet, azaz H nem létezhet

Az “<, >” zárójelek a kódolásra utalnak: ha X egy objektum, akkor <X> jelöli az objektumot kódoló sztringet

Hol itt a diagonális módszer?

Ha ATM rekurzív volna, ki lehetne tölteni egy ilyen táblázatot:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*	
M_4	*accept*	*accept*	*reject*	*reject*	
$\vdots$			$\vdots$		$\ddots$

Ebben az összes Turing gép, és az összes Turing gép leírás (kód) szerepel, azaz D is.

Hol itt a diagonális módszer?

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	$\cdots$	$\langle D\rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*		*accept*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*		*reject*	
M_4	*accept*	*accept*	*reject*	*reject*		*accept*	
$\vdots$			$\vdots$		$\ddots$		
D	*reject*	*reject*	*accept*	*accept*		?	
$\vdots$			$\vdots$				$\ddots$

A_{TM} nem rekurzív nyelv,

Lássuk, hogy vannak nem rekurzívan felsorolható nyelvek is, mutatunk egy ilyet!

(de rekurzívan felsorolható)

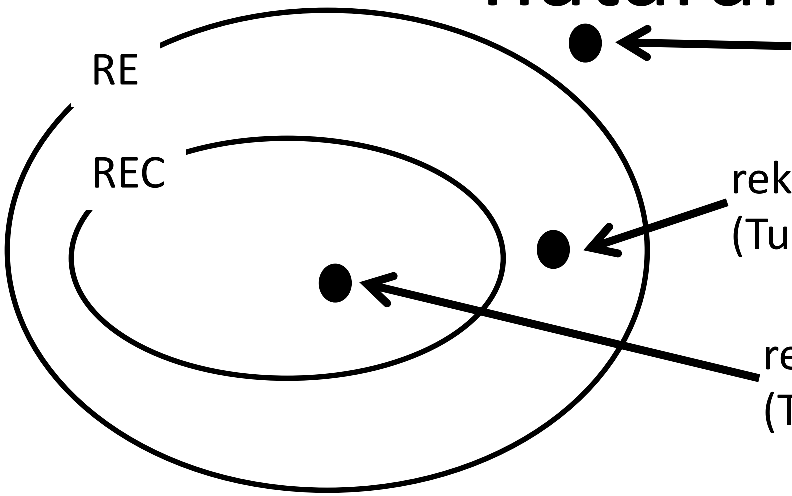
Állítás: Ha L és $\bar{L}$ is rekurzívan felsorolható, akkor L rekurzív.

L komplementere

Miért van ez így?

A_{TM} rekurzívan felsorolható, de nem rekurzív $\Rightarrow$ $\overline{A}_{TM}$ nem rekurzívan felsorolható

Az algoritmikus kiszámíthatóság/megoldhatóság határai



nyelvek, melyek nem rekurzívan felsorolhatók, pl. A_{TM} komplementere

rekurzívan felsorolható nyelvek (Turing gép elfogadja őket, pl. A_{TM}, L_p)

rekurzív nyelvek (Turing gép eldönti őket, megoldható problémák)

- Láttunk rekurzívan felsorolható de nem rekurzív nyelvet - pl. A_{TM} ilyen

Ma

- Környezetfüggő grammatikák, a Chomsky féle nyelv-hierarchia
- Az univerzális Turing gép
- A Turing gép mint az *algoritmus* fogalmának matematikai megjelenése, a Church tézis
- Problémák mint formális nyelvek, algoritmikusan megoldható problémák és rekurzív nyelvek
- Algoritmikusan megoldhatatlan problémák, a megállási probléma eldönthetetlensége, egy konkrét nem rekurzívan felsorolható nyelv