Fontos

Mindig van pumpálás és Rice-tétel!

1. előadás

Jelölések

- Σ: karakterek/betűk halmaza
- Σ^i: i hosszú sorozatok/szavak halmaza
- Σ^*: összes szó
- $w \in \Sigma^*$: szó hossza: $|w|$
- ε: üres szó (0 hosszú)
- $L \subseteq \Sigma^*$: nyelv (szavak tetszőleges halmaza)
    - Σ^*: teljes nyelv, az üres szó is benne van
    - $\emptyset$: üres nyelv, üres szó sincs benne
    - $\{\varepsilon\}$: üres szóból álló nyelv
    - a^*: csupa a betűből álló szavak nyelve

Véges automata

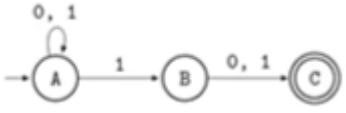
- Q: állapotok
- Σ: karakterek halmaza
- δ: átmeneti függvények, megmondják, hogy működnek a lépések
- q_n: kezdőállapot
- F: elfogadó állapotok

Formák

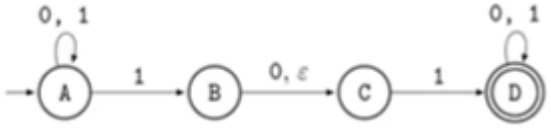
- **Determinisztikus**, ha minden állapot-karakter párra van átmenet.
- **Nemdeterminisztikus**, ha több átmenet is van ugyanarra.
- **Hiányos**, ha nincs mindenhol értelmezve.
- **Teljes**, ha mindenhol értelmezve van.

Az automata olvas a bemeneten, az alapján csinál valamit. **Számítási út**: legális lépések egy sorozata. Az automata **elfogad** egy szót, ha van számítási út F-be. Az elfogadott szavak halmaza az elfogadott nyelv. Egy nyelv **reguláris**, ha van hozzá véges automata, az mindegy, hogy milyen. Úgy is le szokás írni az automatát, hogy $L(M)$, vagyis L nyelvet elfogadó M automata.

Minden hiányos véges automatához van ekvivalens teljes, akár determinisztikus is. Ekkor ha például A-ból A-ba és B-be is megy a 0, akkor felveszünk egy AB állapotot, ahová és csak is ahová fut A-ból a 0. Egy példa átdolgozás arra a feladatra, hogy az utolsó előtti karakter 1:



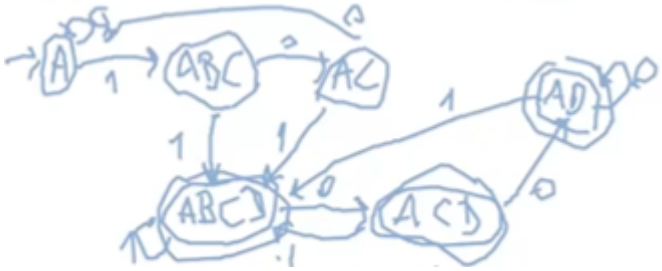
Lehet olvasás nélkül is átugrani egy másik állapotba, ez az ε átmenet, például ezzel a “van benne 101 vagy 11” feladat tömöríthető. Ez már alapból nemdeterminisztikussá teszi.



Tétel viszont, hogy nemdeterminisztikusból át lehet alakítani determinisztikussá. Például vegyük azt, hogy az előzőben hova lehet üres átmenettel jutni:

$E(A) = \{A\}, \quad E(B) = \{B, C\}, \quad E(C) = \{C\}, \quad E(D) = \{D\}$

A megoldás:



Az ABC úgy jön létre, hogy az A-ból az 1 tud menni A-ba és B-be is, viszont ezek E-jét kell nézni, vagyis a C is elérhető a nullátmenet miatt. A-ból 0-val az A-ba, B-ből C-be, C-ből sehová nem tudunk menni, éppen ezért az ABC egy AC állapotba jut, ha 0-t olvas. Ilyen logika szerint kell folytatni a feladatot.

Műveletek nyelvekkel

- Unió: $L_1 \cup L_2$ - valamelyikben benne van a szó
- Metszet: $L_1 \cap L_2$ - mindkettőben benne van a szó
- Különbség: $L_1 - L_2$ - az elsőben benne van a szó, de a másodikban nincs
- Komplementer: $\overline{L_2}$ - bármi, ami a Σ^* része, de nem L_2-nek
- Konkatenálás: $L_1 L_2$ - első fele L_1, második fele L_2 része, a vágás helye bárhol lehet
- Tranzitív lezárt: L_1^* - bárhányszor, akár 0-szor szerepel egymás után egy jó szó

A reguláris nyelvek ezekre zártak. Bizonyítja, hogy mindre fel lehet írni DVA-t (determinisztikus véges automata).

2. előadás

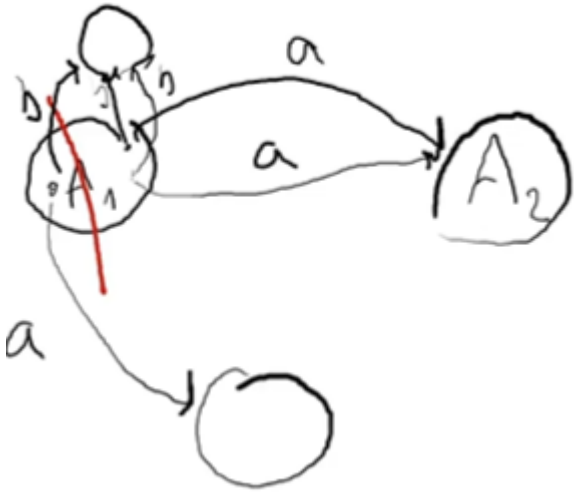
Minimalizálás

Adott nyelvhez minél kisebb automatát akarunk csinálni, vagyis a legkevesebb állapotú determinisztikus teljes automatát. Minden reguláris nyelvhez létezik minimálautomata. Ehhez az egyformán viselkedő (vagyis ekvivalens) átmeneteket össze kell vonni. Ha van két ekvivalens állapot, és azokból továbbmegyünk egy betűvel, azok az állapotok ugyanúgy ekvivalensek lesznek. Ha van két ekvivalens állapot, és az egyik elfogadó, akkor a másik is az.

L/x: olyan szavak, amiket x szó után írva L-ben érvényes szavak állnak elő. Szavak (pl. x és y) akkor ekvivalensek, ha $L/x = L/y$. Ha két szó ekvivalens, akkor ekvivalens úton jutnak kezdőállapotból végállapotba. Ha a két szó megkülönböztethető, akkor nem ekvivalens állapotba jutunk velük. Ebből az jön le, hogy n külön szó legalább n külön állapot. Ekvivalens szavak halmazát ekvivalenciaosztályoknak hívjuk. Ha annyi állapotunk van, ahány ekvivalenciaosztály, akkor minimál az automata. Ez algoritmizálható:

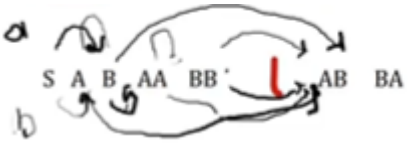
- Legyen két állapothalmaz, $A_1 = nem\ elfogadó$ és $A_2 = elfogadó$.
- Megnézzük az átmeneteket, és ha van két olyan, amik egy adott halmazból két különböző halmazba mennek át, akkor eszerint kettévágjuk a szóban forgó halmazt.
- Megállunk, ha nincs mit elvágni.

A vágás alapja szemléltetve, azt is bemutatva, hogy attól még, hogy az összes b ugyanoda mutat, az a-k közt még lehet eltérés:



Példának vegyünk egy olyan automatát, hogy legalább 2 hosszú, különböző utolsó 2 betűs szavakat fogad el (a BA-ból AB-be mutató b átmenet kimaradt):

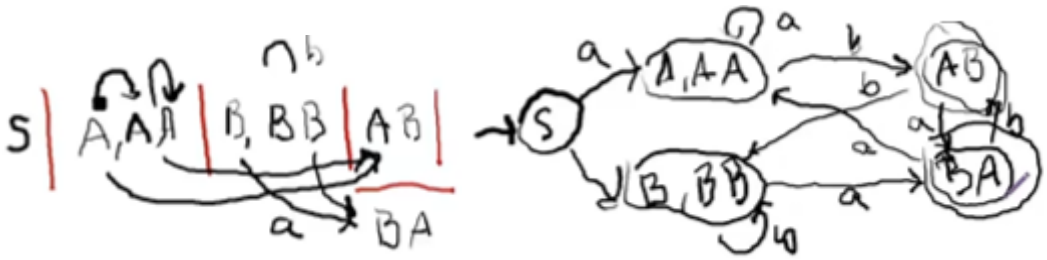
Egy nagyon jó trükk felírni a vágási pontot, és hogy melyik betű mit csinál:



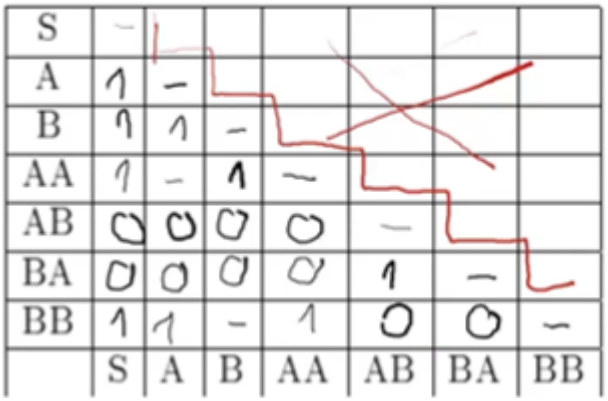
Itt ezek a halmazaink:

- a és b is marad (S)
- a marad, b átmegy, nem elfogadó (A, AA)
- b marad, a átmegy, nem elfogadó (B, BB)
- a marad, b átmegy, elfogadó (AB)
- b marad, a átmegy, elfogadó (BA)

Egyelemű halmazok nyilván nem vághatók szét, elég A-t és AA-t, illetve B-t és BB-t megnézni páronként. Mivel ugyanúgy viselkednek, nem vágódik semmi tovább, felrajzolható a minimálautomata:



Megoldható táblázatosan, ahol a sorok és oszlopok az állapotok. Ez szimmetrikus, a felső vagy alsó háromszög nélkül is jó. Első lépés, hogy ahol az egyik index elfogadó, a másik nem, ott írjunk be 0-t, máshol semmit. Innentől a j. körben j kerül a táblázatba, ha a két vizsgált elemnél ugyanaz a betű nem ugyanabba a halmazba megy, és addig van kör, amíg lehetséges beírni új számokat üres helyekre. Ahol nem az átlóban marad üres mező, azokat az állapotokat össze lehet vonni. Az előző példa megoldása így:



	S	A	B	AA	AB	BA	BB
S							
A	1						
B	1	1					
AA	1		1				
AB							
BA					1		
BB	1	1		1			

Nemdeterminisztikus véges automatára (NVA-ra) nincs ilyen eljárás. Két nyelv közel van egymáshoz, ha csak véges sok szóban különböznek. A minimalizálás kiterjeszthető, hogy a közeli nyelveket meghatározza.

Reguláris kifejezések

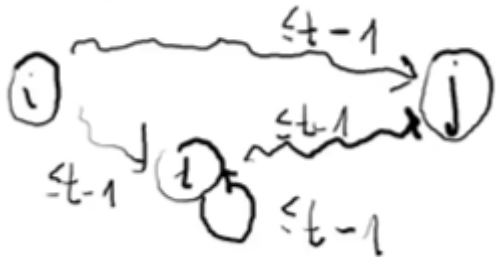
Van 3 elemi típus, az üres nyelv ($\emptyset$), az üres szó (ε), és a betű ($a \in \Sigma$). Az utóbbi kettőből van nyelv, pl. üres szónak megfelelő nyelv ($L = \{\varepsilon\}$). A nyelvek közt értelmezett az unió ($L_1 \cup L_2$, elfogadott szavak összevonása), konkatenálás ($L_1 L_2$, elfogadott szavak egymás után illesztése), és a lezárás (L_1^*, elfogadott szavak bárhányszor, akár nullaszor ismétlése).

Reguláris kifejezés az elemi típusokkal és ezzel a három művelettel leírható, pl. $a^* b a^*$ (legalább 1 b-t tartalmazó szavak) vagy $(a + b)^*$ (a-ból és b-ből álló szavak). Ha olyan analógiát állítunk, hogy $\emptyset = 0$ és $\varepsilon = 1$, akkor kijönnek a szorzás és összeadás képletei, pl. $R + 0 = R$. A reguláris kifejezésekkel leírható nyelvek a reguláris nyelvek.

3. előadás

$L(i,j,t)$: i-ből j állapotba legfeljebb t lépésben jutunk el. Ha az összes ilyet fel tudjuk írni reguláris kifejezésként, vagyis $R(i,j,t)$ alakban, kész vagyunk. Ha $t = 0$, akkor nincs köztes állapot, tehát az ilyenkor vonatkozó reguláris kifejezés a betűk összege, amik az átmenetet okozzák. Ha $i = j$, akkor hurokról van szó, azt is így kezeljük, de az ε is legyen a halmazban. Ha $t \neq 0$, rekurzívan visszavezethető arra, itt t bármely köztes állapotot jelenti:

$$R(i,j,t) = R(i,j,t-1) + R(i,t,t-1)\ R(t,t,t-1)^*\ R(t,j,t-1)$$



Nézzük azt a nyelvet, ami a b-ket tartalmazó szavakat fogadja el:

Példa:

Kell: $R(1,2,2)$

$R(1,1,0) = \varepsilon + a$ $\quad R(1,2,0) = b$

$R(2,1,0) = \emptyset$ $\quad R(2,2,0) = \varepsilon + a + b$

$R(1,1,1) = R(1,1,0) + R(1,1,0)\ R(1,1,0)^*\ R(1,1,0) = \varepsilon + a + (\varepsilon + a)(\varepsilon + a)^* = \varepsilon + a + a^* = a^*$

$R(1,2,1) = R(1,2,0) + R(1,1,0)\ R(1,1,0)^*\ R(1,2,0) = b + (\varepsilon + a)(\varepsilon + a)^* b = b + a^* b = a^* b$

$R(2,1,1) = R(2,1,0) + R(2,1,0)\ R(1,1,0)^*\ R(1,1,0) = \emptyset + \emptyset = \emptyset$

$R(2,2,1) = R(2,2,0) + R(2,1,0)\ R(1,1,0)^*\ R(1,2,0) = \varepsilon + a + b$

$R(1,2,2) = R(1,2,1) + R(1,2,1)\ R(2,2,1)^*\ R(2,2,1) = a^* b + a^* b (a+b)^* = a^* b (a+b)^*$

Reguláris-e egy nyelv?

Alapból az ekvivalenciaosztályok felsorolásával lehetne megoldani, de végtelen sok szó is lehet, szóval inkább van a **pumpálási lemma**, ami ilyesmi, csak a köröket jól kezeli. A lemma: ha van egy reguláris nyelvünk, akkor van egy olyan $p > 0$ pumpálási hossza, hogy minden szó felbontható egy olyan $u\,v\,w$ alakra, hogy v legalább 1 hosszú, és $u\,v^k\,w$ is a nyelv szava, ha $k \geq 0$. Ha egy nyelv nem reguláris, úgy bizonyítjuk, hogy nem igaz rá a pumpálási lemma. Például nem reguláris az $a^n b^n$ nyelv, ebben bármilyen $u\,v\,w$ felosztásra az $u\,v^2\,w$ már nincs a nyelvben. Van reguláris nyelv, ami pumpálható, tehát a nem reguláris nyelveket nem lehet vele keresni, csak azt bizonyítani, hogy valami nem reguláris.

4. előadás

Nyelvtanok

Egy formális G nyelvtanban van:

- V - változók nem üres halmaza
- Σ - ábécé
- $S \in V$ - kezdőváltozó
- P - levezetési szabályok, egy olyan alakból, amiben legalább egy változó van, bármivé átalakíthat

Levezetés = eljutottunk olyan alakra, ahol már csak ábécében lévő betűk vannak, nem pedig változók. Nyelvtant megadhatunk csak a szabályokkal, ezeket a bal oldalon össze is lehet vonni, pl. $X \to a$ és $X \to b$ esetén $X \to a \mid b$.

Chomsky-nyelvosztályok

A nyelvek osztályát az adja meg, hogy milyen osztályú nyelvtannal lehet generálni.

- 0. osztály - generatív nyelvtan: nincs megkötés, mindig generál valamit
- 1. osztály - környezetfüggő (CS) nyelvtan: csak változókat lehet helyettesíteni, de a változó előtt és után bal- és jobboldalt ugyanaz marad
- 2. osztály - környezetfüggetlen (CF) nyelvtan: baloldalt csak egy változó lehet
- 3. osztály - reguláris nyelvtan: csak $A \to a$ és $A \to aB$ forma megengedett

Vannak majdnem reguláris nyelvek, amiknek a szabályai $A \to a$ és $A \to \varepsilon$ alakúak, de az ε szabály reguláris nyelvben tilos. Tétel, hogy ez elhagyható. Először meg kell hozzá határozni az E halmazt (mikből lehet ε), az ebben lévő szabályokat ketté kell bontani ε-t megengedő szabályokra, illetve olyanokra, amikben az összes ε-ban végződő levezetés be van helyettesítve. Egy példa:

- $S \to aA \mid bB \mid \varepsilon$
- $A \to bB \mid \varepsilon$
- $B \to bS \mid a$

$E = \{S, A\}$

$S_0 \to \varepsilon \mid aA \mid a \mid bB$

$S \to aA \mid a \mid bB$

$A \to bB$

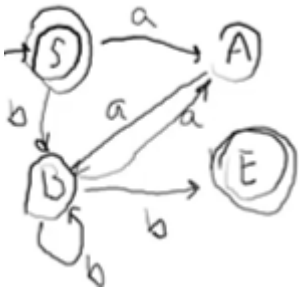
$B \to bS \mid b \mid a$

Minden reguláris nyelvtanhoz van reguláris nyelv. Ezt az bizonyítja, hogy automatát lehet belőle generálni:

- $A \to a$ szabályokra egy elfogadó átmenetet írunk fel,
- $A \to aB$ szabályokra sima átmenetet írunk fel.

Példa reguláris nyelvtanból felrajzolt automatára:

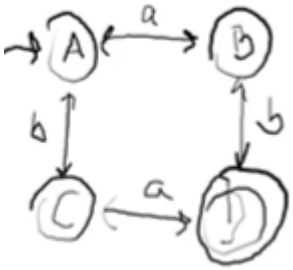
- $S \rightarrow aA \mid bB \mid \varepsilon$
- $A \rightarrow aB$
- $B \rightarrow bB \mid aA \mid b$



Minden reguláris nyelvhez van reguláris nyelvtan, ami az előző visszafelé:

- elfogadó átmenetekből $A \rightarrow a$ szabályt írunk fel,
- sima átmenetekből $A \rightarrow aB$ szabályt írunk fel.

Példa: a nyelv a páratlan sok a-t és páratlan sok b-t tartalmazó szavakat fogadja el:



Megoldás:

- $A \rightarrow aB \mid bC$
- $B \rightarrow cA \mid bD$
- $C \rightarrow bA \mid aD$
- $D \rightarrow aC \mid bB \mid \varepsilon$, de az ε-t megtanultuk eltüntetni.

Ha egy nyelvtan minden szabálya $A \rightarrow \alpha$ alakú (majdnem CF), akkor van CF nyelvtana, amihez szintén annyit kell csinálni, hogy az ε-okat felszámoljuk. Fel kell írni az i lépésben elenyésző ε-ok halmazait, de először elkülönítjük azokat, amik nem generálnak ε-t. Az elenyésző változók jele ismét E, nézzünk példát:

- $S \rightarrow ABCD$
- $A \rightarrow CD \mid AC \mid AB$
- $B \rightarrow Cb \mid Sb$
- $C \rightarrow a \mid \varepsilon$
- $D \rightarrow bD \mid \varepsilon$

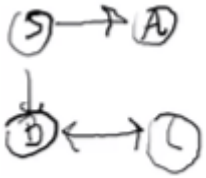
Ekkor $E_1 = \{C, D\}$, mivel azok mennek ε-ba. $E_2 = \{C, D, A\}$, mert az $A \rightarrow CD$ egy olyan szabály, ahol jobb oldalon csak E_1-beli elemek vannak. $E_3 = \{C, D, A\}$, mert se S, se B nem tud elenyészni. Mivel nem került új változó a halmazba, megállunk. Most az összes elenyésző változót felírjuk az összes állapotra, legyen az E_3 bármelyik, akár több eleme:

- $S \to ABCD \mid BCD \mid ABD \mid ABC \mid BD \mid BC \mid AB \mid B$
- $A \to CD \mid D \mid C \mid AC \mid A \mid C \mid AB \mid B$
- $B \to Cb \mid b \mid Sb$
- $C \to a$
- $D \to bD \mid b$

Egy másik átalakítás, hogy a láncszabályokat hagyjuk el, amikkel nincs haladás, pl. $A \to B$ és $B \to A$. Ezeket érdemes behelyettesíteni, és azt megoldani, hogy ne legyen kör. Ehhez először meg kell határozni minden változóhoz, hogy onnan mi érhető el egyszeres szabállyal, akár több lépésben is. Ez alapján már lehet összevonni, például a következő nyelvet:

- $S \to aB \mid A \mid B \mid a$
- $A \to aa \mid aC$
- $B \to aB \mid C$
- $C \to bS \mid B$

Segít egy rajz az egyszeres szabályokról:



Ekkor

- $N_S = \{A, B, C\}$ (a C tranzitív),
- $N_A = \emptyset$,
- $N_B = \{C\}$,
- $N_C = \{B\}$.

Eredmény behelyettesítésekkel egy esetben: $S \to aB \mid a \mid aa \mid aC \mid aB \mid bS$.

5. előadás

Felesleges szimbólumok

Legyen mondjuk

- $S \to a \mid Aa$
- $A \to Aa \mid AB \mid CC$
- $B \to bb$
- $C \to bC$
- $D \to abab$

Probléma: C-től nem szabadulunk (1. típus), D-t nem tudjuk elérni (2. típus), mindkettő felesleges szimbólum. **Először az 1., majd a 2. típusúakat kell kiszűrni**, ha fordítva kezdjük, maradhat még 2. típusú.

1. típus felszámolása

Kiindulunk a Σ-ból, majd bővítjük az így megkezdett T_0 halmazt azokkal a változókkal, amik T-beli szimbólumokká alakíthatók (van $A \to a$ alak), ezt ismételgetjük T_i minden elemére. A példára:

- $T_0 = \{a, b\}$
- $T_1 = \{a, b, B, D, S\}$
- $T_2 = \{a, b, B, D, S\}$ - nem változott

Itt leállt a dolog, kimarad az A és C → azokat tartalmazó minden kuka, az új nyelvtan

- $S \to a$
- $B \to bb$
- $D \to abab$

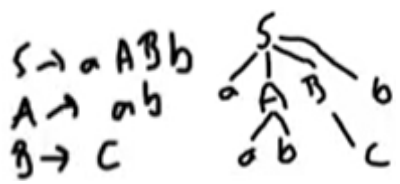
2. típus felszámolása

Az elérhetőket kell feltérképezni. Ehhez a kezdőváltozóból indulva úgy bővítünk ki egy S halmazt, hogy hova lehet onnan eljutni. Ami kimarad, eldobjuk a szabályait. Az előző kukázás utáni állapotot folytatva:

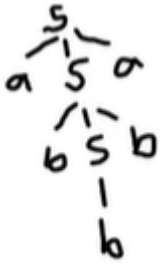
- $S_0 = \{S\}$
- $S_1 = \{S, a\}$
- $S_2 = \{S, a\}$ - végeztünk, egyedül az $S \to a$ szabály marad.

Levezetési fa

CF nyelvtanból elég evidens:



Példa a páratlan hosszú palindromok nyelvtana: $S \to aSa \mid bSb \mid a \mid b$, ennek legyen a példa levezetése az $S \Rightarrow aSa \Rightarrow abSba \Rightarrow abbba$:



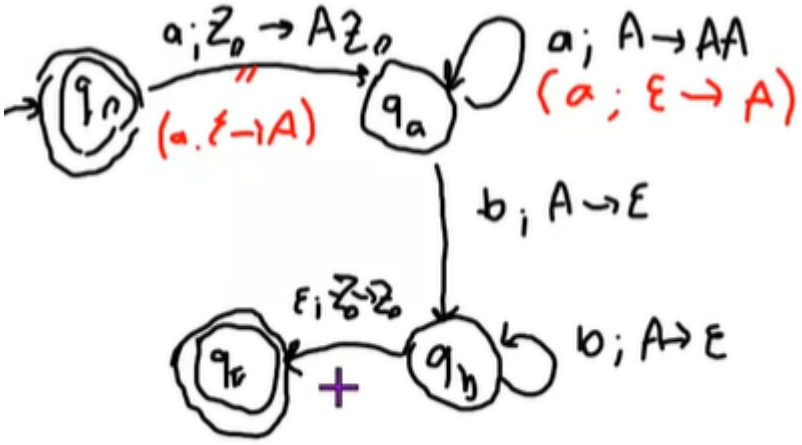
Ha minden levezethető szóhoz csak egy fa van, akkor egyértelmű a nyelvtan. Ha a nyelvhez van egyértelmű CF nyelvtan, akkor egyértelmű a nyelv. Van nem egyértelmű CF nyelv, és nincs algoritmus az egyértelműség eldöntésére. A CF-ek zártak az unióra, konkatenálásra, és tranzitív lezártakra, akárcsak a reguláris kifejezések, a többi műveletre viszont nem zárt.

Veremautomata

- Q - állapotok halmaza
- Σ - ábécé
- Γ - veremábécé (szimbólumok)
- $q_0 \in Q$ - kezdőállapot
- $Z_0 \in \Gamma$ - kezdőszimbólum (verem alja)
- $F \subseteq Q$ - elfogadó állapotok
- δ - átmeneti függvény

Mint egy sima automata, csak egy verembe pakolunk vagy abból veszünk ki. Egy átmenet $\delta(q, a, A)$ alakú, q állapotban vagyunk, a-t látunk a bemeneten, ettől a verembe A-t teszünk. Egy lépés formája $(q, ax, A\beta) \to (r, x, \alpha\beta)$. Itt lehet pl. törölni a veremből (ha $\alpha = \varepsilon$), vagy nem haladni a bemeneten (ha $a = \varepsilon$). Ha nincs alkalmazható átmenet, elakad. Elfogadó állapot, ha a bemenetről ε jön, és ha ilyenbe eljutunk, elfogadja az automata a szót. A szó végén még lehet lépni, de ha itt akad el, már elfogadta.

Példa: $L = \{a^n b^n : n \geq 0\}$, erre tipp, hogy a veremben tároljuk az a betűket, és szedjük ki őket minden b-re, csak az ezutáni üres verem elfogadó. Az ilyen automaták rajzán az élek szövege, hogy milyen bemenetre mi történik a verem tetejével. A példát megoldó automata:



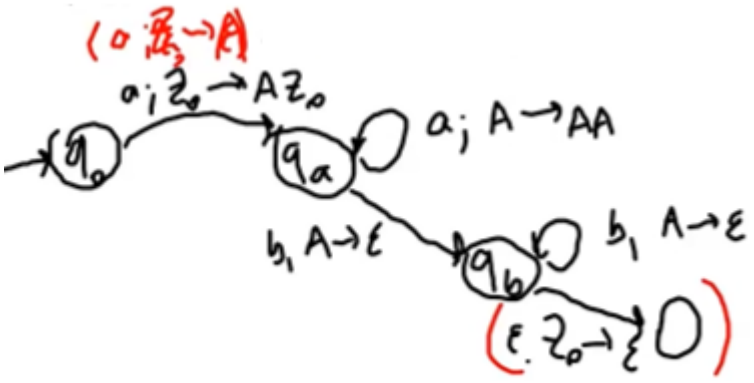
Vegyük például az $A \rightarrow AA$ átmenetet, ez az állapot úgy írható le, hogy $\delta(q_a, a, A) = (q_a, AA)$, vagyis a q_a állapotban, ha a bemeneten a érkezik, akkor az A tetejű verem teteje AA-vá változik, és maradunk a q_a állapotban. Az utolsó él azt ellenőrzi, hogy a verem állapota maradt-e az, amivel indult, tehát akkor fogadjuk csak el, ha nincs szemét az alján.

Determinisztikus veremautomata: mindig legfeljebb egy lehetőség van átmenetre. Determinisztikus egy CF nyelv, ha van hozzá determinisztikus veremautomata. Mivel van nemdeterminisztikus CF nyelv, ezért veremautomata nem determinizálható. Minden determinisztikus CF nyelv egyértelmű, de van nemdeterminisztikus is köztük (pl. palindromok nyelve).

6. előadás

Üres veremmel elfogadó veremautomata

Akkor fogad el egy szót, ha $(r, \varepsilon, \varepsilon)$ állapotba jut. Kimarad az F (elfogadó állapotok) halmaz, de amúgy ugyanúgy működik, mint a sima veremautomata. Egy ilyen példa az $\{a^n b^n : n \geq 1\}$ nyelvre:



(az ott fent $Z_0 \rightarrow A$ akar lenni)

Ha egy nyelv elfogadható veremautomatával, akkor elfogadható üres veremmel is. Ez visszafelé is érvényes. Emiatt inkább próbáljunk mindig üres veremmel elfogadót csinálni. CF nyelvtanból például úgy, hogy levezetjük a veremben, és a szó következő karaktere kell előálljon a verem tetején. Ebből jön, hogy minden CF nyelvre van azt elfogadó veremautomata.

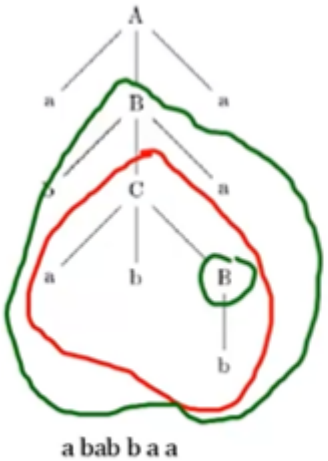
Példa: $S \rightarrow aSa \mid bSb \mid a \mid b$

Az üres veremmel elfogadó automata:

$Q = \{q\}, \qquad \Gamma = V \cup \Sigma = \{S, a, b\}$

$\delta(q, \varepsilon, S) = \{(q, aSa), (q, bSb), (q, a), (q, b)\}, \quad \delta(q, a, a) = \{(q, \varepsilon)\}, \quad \delta(q, b, b) = \{(q, \varepsilon)\}$

Hogyan lássuk be, hogy valami CF-e? Példa: $\{a^n b^n c^n : n \geq 0\}$, itt a megkülönböztethetőség és a pumpálás jön jól. A reguláris nyelveknél a DVA-ban van kör, tehát pumpálható, de a CF-ek esetében használt PDA nem determinisztikus, más kell. Ettől még van kör, lásd levezetési fa:



a bab b a a

A reguláris pumpálásnál az $u\,v\,w$-t pumpáltuk $u\,v^k\,w$ alakba, itt $u\,v\,w\,x\,y$-t pumpálunk $u\,v^k\,w\,x^k\,y$ alakba. A CF feltétel, hogy vx nem lehet üres, és vwx hossza $\leq p$. Egy példa, hogy $\{a^n b^n : n \geq 0\}$ pumpálható $u = a^{n-1}$, $v = a$, $w = \varepsilon$, $x = b$, $y = b^{n-1}$ alakban. Ha egy nyelv környezetfüggetlen, ez a lemma alkalmazható rá, tehát indirekt bizonyítható, hogy egy nyelv nem CF. Ha egy nyelvben lévő szót úgy pumpálunk, hogy az már nincs a nyelvben, akkor megdőlt a CF felvetés. Van olyan nyelv, ami pumpálható, de nem CF.

7. előadás

Algoritmikus kérdések

Ha egy nyelv reguláris, megadhatjuk leírással, véges automatával, reguláris nyelvtannal, reguláris kifejezéssel.

Beletartozás

- DVA-t végigkövetjük
- NVA-ban minden számítási utat bejárunk, de ez exponenciális, szóval determinizáljuk
- Reguláris kifejezésből vagy nyelvtanból: NVA → DVA
- CF: elágazás és korlátozás típusú módszer (nem tananyag)

Üresség

- VA: utat keresünk a gráfban a kezdőállapotból egy elfogadóba - bejárás, esetleg minimalizálhatunk, és ha az üres nyelv automatáját kapjuk, akkor üres
- Reguláris kifejezésből: szerepel minden tagban az ∅, akár $R = a\ \emptyset +\ b\ {}^*\emptyset$
- Reguláris nyelvtanból: levezethető-e bármilyen szó a felesleges szimbólumok eltüntetése után
- CF nyelvek: pumpálási lemma, a p-nél kisebb szavak benne vannak-e

Végesség

- Reguláris kifejezésnél minden véges, amiben nincs *
- VA: ha elfogadó utakban nincs kör, véges
- CF nyelveknél: pumpálási lemmával: ha $p + 1$ és $2p$ közti hosszú szavak nincsenek L-ben, hosszabbak sem

Diszjunktság

- VA: metszetet elfogadó automata elfogad-e valamit
- Reguláris kifejezéshez és nyelvhez VA-t kell készíteni
- CF nyelvre nincs algoritmus

Egyenlőség

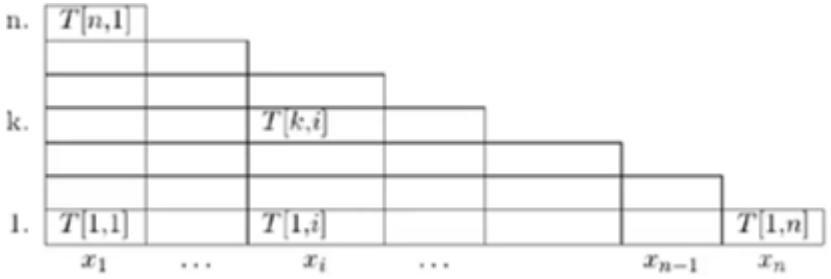
- Minimalizáljunk VA-kat, és ha a két automata izomorf, egyenlők
- CF nyelvre nincs algoritmus

Chomsky-normálformájú nyelvtan

CNF nyelvtan: olyan CF nyelvtan, amiben az $S \to \varepsilon$ szabályon kívül csak $A \to BC$ és $A \to a$ alakok lehetnek. Minden CF nyelvtan átalakítható CNF nyelvtanná. Ehhez minden a betűhöz, ami **nem egyedül** szerepel szabály jobb oldalán, bevezetjük az x_a változót és az $x_a \to a$ szabályt. Ezután minden $A \to a$ szabályban, ahol a nem egyedül volt, lecseréljük az a betűket x_a-ra. Utolsó lépésként simán feldaraboljuk a hosszú szabályokat. Ha van olyan, hogy $A \to BCDEF\ldots$, akkor legyen sok új változó, velük pedig lépegessünk mindig úgy, hogya következő tagot hozza ki sorban a szabály: $A \to BY_1$, $Y_1 \to CY_2$, $Y_2 \to DY_3\ldots$

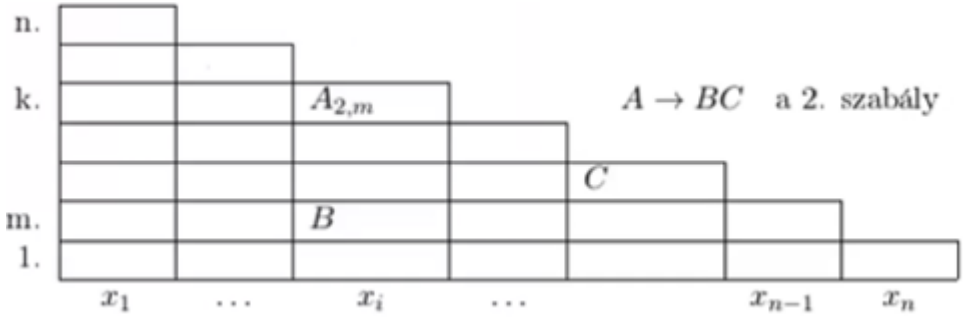
8. előadás

Algoritmus, ami CNF nyelvtant generál: CYK-algoritmus. Egy táblázatot kell felrajzolni, hogy az adott hosszú résszó milyen karakterekből vezethető le.

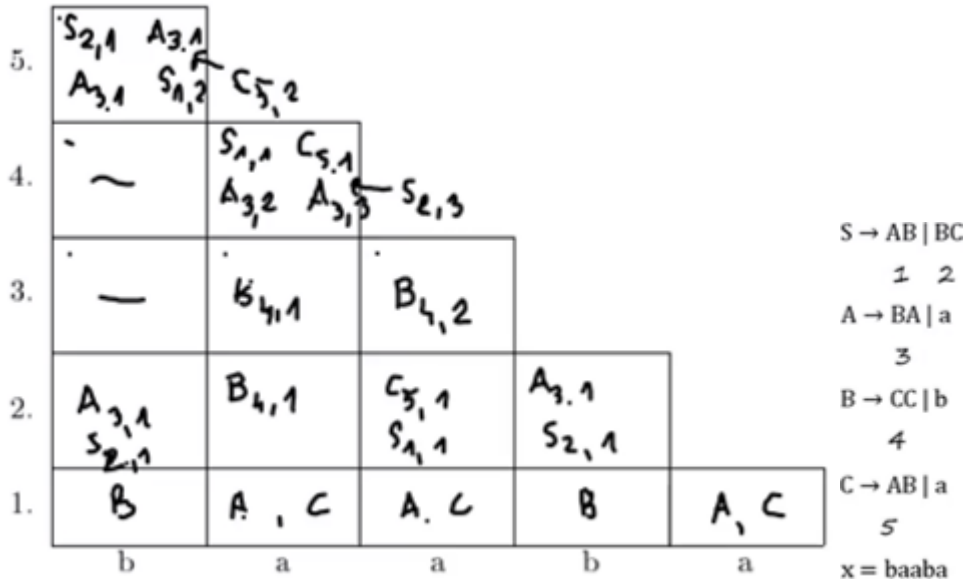


Kezdjük az alján, ahol 1 hosszú résszavakat vezetünk le, vagyis amihez van $A \to x_i$ szabály.

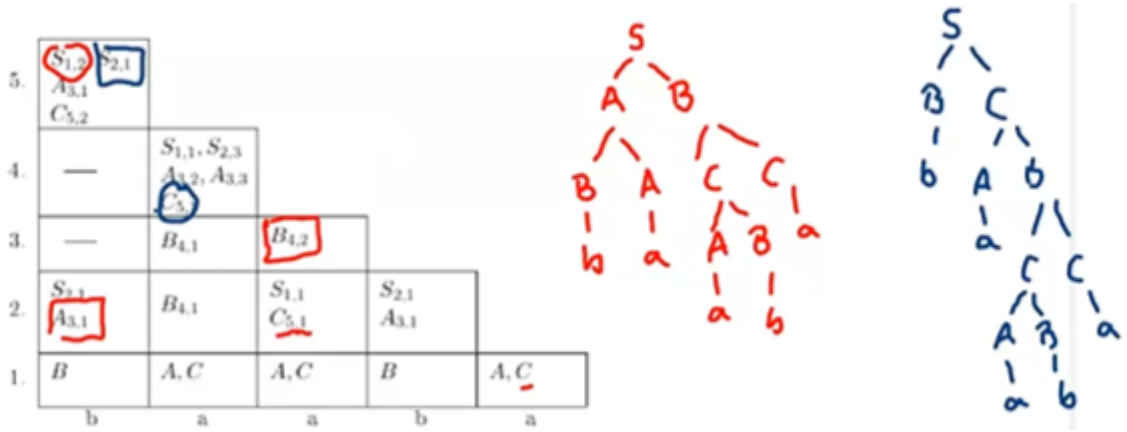
A feljebbi mező azt jelenti, hogy ami alatta van legalul, az onnan kezdődő k betűs résszó levezethető belőle. Nem csak a változót írjuk be, hanem hogy hogyan vezetjük le (alsó indexbe a szabály sorszáma, és az, hogy melyik sorban keressük az első elemét:



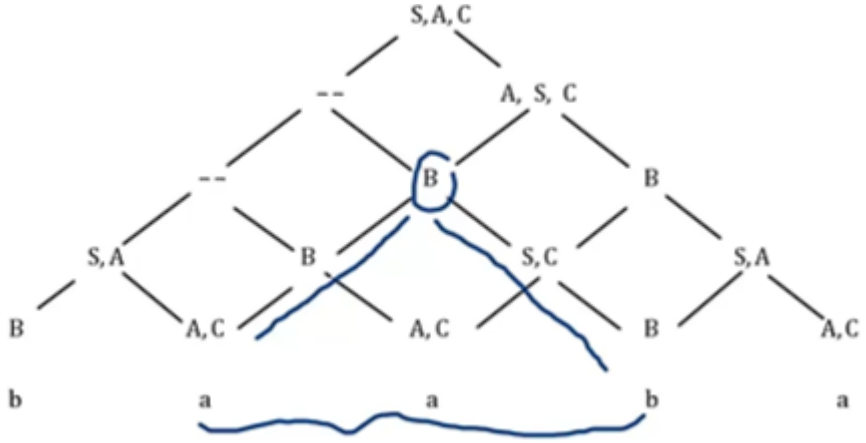
Itt például azt írja le, hogy a 2. szabály alapján az m. sor az első karakter, innen már evidens, hogy hol kell keresni a C-t. Ha egy sor kész, ugrunk feljebb, például a 2. sorban azt keressük, hogy az alattuk induló 2 hosszú szavak hogy jönnek ki. A harmadik sorban már több lehetőséget is meg kell nézni, a háromszögekre érdemes figyelni. Lehet például a 3. sor 1. mezője olyan, hogy a 2. sor 1. és 1. sor 3. mezője teszi ki, de az is, hogy az 1. sor 1. és 2. sor 2. Még egy lehetőség neki, hogy az 1. sor 1-3. mezőket is leírhatja egy szabály.



Elég az utolsó sorban a kezdőállapotokat leírni, abból lehet érvényes levezetési fát visszarajzolni. Az egyértelműség meghatározására is jó ez az algoritmus, ugyanis ha több kezdőállapot kerül a legfelső sorba, akkor több lehetséges levezetés is létezik:



Táblázat nélkül, eltolással egyszerűbben kiolvasható:

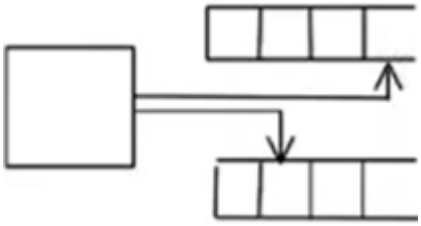


Reguláris nyelvtan lehet nem egyértelmű, de reguláris nyelv már egyértelmű.

9. előadás

Turing gépek

Jele M, van benne k darab szalag, akár a jobb oldalukon végtelenek, és az is megvan, hogy milyen betűk lehetnek rajtuk. A bemenet is szalag, az első szalagra rá van írva. Minden szalagnak van egy író-olvasó feje, ami balra-jobbra mozog.



- Q: állapotok
- Σ: ábécé
- Γ: szalagábécé, ezek kerülhetnek a szalagokra
- q_0: kezdőállapot
- F: elfogadó állapotok
- $*$: üres mező jele, alapból minden mezőben ez van, ami nem bemenet
- δ: átmeneti függvények

Egy átmeneti függvény formája $\delta(q, a_1, a_2, \ldots a_k) = (r, b_1, b_2, \ldots b_k, i_1, i_2, \ldots i_k)$. Itt q-ból r állapotba lép, és az a-kat (minden szalag jelenlegi karaktere) felülírja b-kkel (szalagok karaktereit felülíró karakterek), ez akkor történik, ha mind a k szalagon a megfelelő karakter van. Miután a felülírás megtörtént, az i-k jelentik azt, hogy merre lépnek a fejek. Ez lehet egyet balra, egyet jobbra, vagy semerre.

Ha nincs olyan állapotátmenet, ami az adott állapotban érvényes, akkor megáll, és ez lehet elfogadó állapot. Ha a gép nem áll meg (végtelen ciklus), akkor biztos nem fogad el. Az elfogadott nyelv $L(M)$, ami azokból a w szavakból áll, amiket a gép elfogad. Egy véges automata (VA) biztos megáll, elfogadás feltétele a végigolvasás. A veremautomata (PDA) nem feltétlenül áll meg, az elfogadás feltétele a végigolvasás, még ha végtelen ciklusban is ragad. A Turing gépnek (TG) már végig se kell olvasni az elfogadáshoz (pl. 1-gyel kezdődő szavak elfogadása).

Egy nyelv **rekurzívan felsorolható** (recursively enumerable, RE) avagy felismerhető, ha van azt elfogadó TG. Egy nyelv **rekurzív** (recursive, R) avagy eldönthető, ha van olyan TG, ami minden bemeneten véges számú lépésben megáll, és azt fogadja el. $R \subseteq RE$. Ez azt is jelenti, hogy ha RE gépünk van, az csak arra ad garanciát, hogy a nyelvbe tartozó szavakat jelzi, de ha már nem tartozik bele a szó, a nemleges választ sem biztos, hogy megkapjuk.

Church-Turing-tézis: Ha van algoritmus egy nyelvbe tartozási problémára, akkor $L \in R$, ez visszafelé is igaz. Ha van eljárás, ami nem biztos, hogy megáll a nyelvbe nem tartozó szavakra, akkor $L \in RE$, ez is igaz fordítva.

Van olyan nem CF nyelv, amihez van TG, pl. $\{a^n b^n c^n : n \geq 1\}$, ennek a megoldása egy szalaggal is megoldható, először átírunk egy a-t x-re, navigálunk egy b-re, ami y lesz, aztán

egy c lesz z. Ha ezt a kört ismételjük, akkor ha elfogynak az a-k, azt kell nézni, hogy elfogyott-e más is. Ez mindig megáll, tehát a nyelv R-ben van.

Minden nyelv, amire van TG, megoldható 1 szalaggal is. Erre az a trükk, hogy $2k$ sávra osztunk egyetlen szalagot, a páratlanok lesznek a régi szalagok adatai, a párosakat pedig arra használjuk, hogy a fej helyét rögzítsék. Végigpásztázzuk az állapotokért, majd még egyszer a visszaírásokért, és ettől a gép $O(n^2)$-es lesz. A szalagábécének része kell legyen az üres karakter ($*$) és az 1 (ez jelöli a fej helyét).

Létezik nemdeterminisztikus Turing-gép (NTG): van olyan számítási út, amin elfogad, de lehet olyan számítási út, amin meg sem áll. Minden NTG-hez van megfelelő determinisztikus TG.

Egy TG-t alapból átmeneti függvényekkel és egyéb paraméterekkel írunk le. Ha ezt mindet leírjuk, átfordítható egy 0-1 sorozatra, ezt mondhatjuk úgy, hogy a Turing-gép kódja. Innentől egy 0-1 sorozat lehet szó vagy Turing-gép is, és igazolni lehet, hogy egy sorozat TG-e, ráadásul ez R-beli nyelv, mert van rá algoritmus.

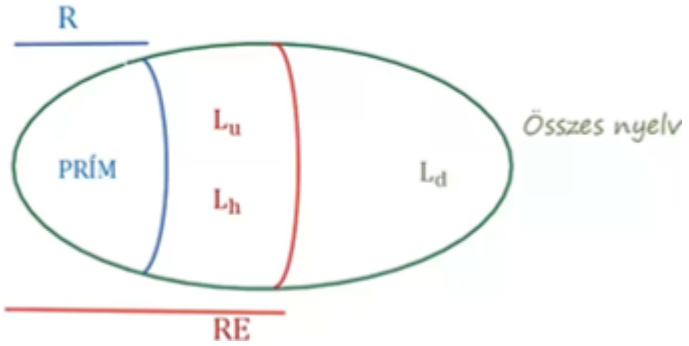
Diagonális nyelv: olyan nyelv, ami szabályos Turing-gép kódokat fogad el (0-1 karakterekből), bármilyet is kap bemenetként, kivéve saját magát. Ez nem RE, vagyis nincs ehhez algoritmus vagy TG. A bizonyítás, hogy ha van TG, akkor az illeszkedik a nyelvbe, ezt pedig el kellene fogadnia, de ez ellentmondás.

Univerzális nyelv: két részből áll a bemenet, a # elválasztó karakter: $w\#x$, ahol w egy TG kódja, x pedig egy bemenet, és az univerzális nyelv azokat a $w\#x$-eket tartalmazza, ahol w elfogadja x-et. Tehát ez a nyelv tartalmazza az összes lehetséges Turing-gép és bemenet párost, és ha benne van, akkor a gép elfogadja.

Megállási nyelv (halting problem): ugyanez, csak nem elfogadja x-et, hanem megáll rá. Az univerzális és megállási nyelv RE, de nem R. Ez azt jelenti, hogy nincs algoritmus, ami elmondaná, hogy egy TG elfogad-e vagy megáll.

10. előadás

PRÍM algoritmus: eldönti, hogy egy szám prím-e. Mivel algoritmus, R-ben van. R-nél kicsit nagyobb halmaz a rekurzívan felsorolható feladatok halmaza, ebbe tartozik, de nem rekurzív az univerzális és megállási nyelv. A diagonális nyelv nincs RE-ben.



Ha egy nyelv rekurzív, a komplementere ($\overline{L}$) is rekurzív. Egyszerűen az elfogadó és nem elfogadó állapotokat kell felcserélni hozzá. Ez a logika nem működik RE-re, mert sem így, sem úgy nem áll meg. Ha L és a komplementere is RE, akkor $L \in R$. Az univerzális (L_u) és megállási (L_h) nyelv komplementere nem RE.

Egy **nyelvosztály komplementere** (nem nyelv komplementere) $co\,X$, és azok a nyelvek tartoznak bele, amiknek a komplementere a nyelvosztály eleme. Tehát a komplementerek nem biztos, hogy benne vannak, de ha egy nyelv és a komplementere is eleme X-nek, akkor igen. Igaz, hogy $co\,co\,X = X$, és az is, hogy $X \subseteq Y \Rightarrow co\,X \subseteq coY$. Még pár ilyen: $co\,R = R$ és $RE \cap co\,RE = R$. R és RE zártak az unióra, metszetre, konkatenálásra, és tranzitív lezártra. L_ε: üres bemenetre megálló Turing-gépek osztálya, $L_\varepsilon \in RE - R$. Az $L_\emptyset$ osztály már csak a semmit elfogadó Turing-gépek (pl. végtelen ciklusok), ez a $co\,RE$ osztályban van, miközben RE-ben nincs. Ehhez annyit kell csak belátni, hogy $\overline{L_\emptyset} \in co\,RE$. Azt is tudjuk, hogy $\overline{L_\emptyset} \notin RE$, így R-ben sincs.

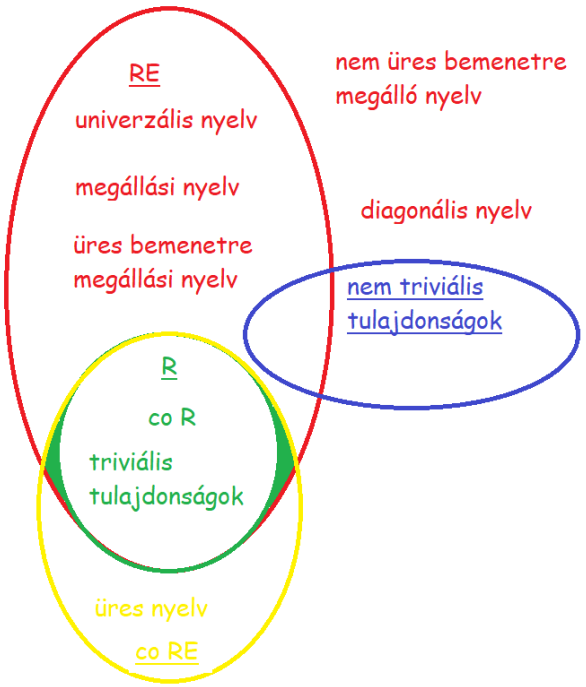
T egy nyelvi tulajdonság, pl. csupa páros hosszú szót tartalmaz. Egy nyelvi tulajdonság akkor nem triviális, ha létezik két olyan RE nyelv, amik közül csak az egyikre igaz. Másképp: T nyelvek egy halmaza, amik közt van RE nyelv, de a komplementerében is van RE nyelv. Lehet hozzá egy L_T nyelvet rendelni. Lehetséges tulajdonságok, hogy a nyelv üres, véges, vagy reguláris. Az már nem nyelvi tulajdonság, hogy megáll-e a TG. Két esetben lehet triviális a nyelv, ha vagy minden tartalmazott nyelv RE-beli (pl. érvényes Turing-gép kódok), vagy egyik sem (pl. a semmi).

Ha T egy triviális nyelvi tulajdonság, akkor $L_T \in R$ és $L_{\overline{T}} \notin R$, de ha nemtriviális, akkor már nem igaz ez. Ennek hozománya, hogy ha el akarjuk dönteni, hogy egy TG elfogad-e egy adott tulajdonságú nyelvet, az nem lehetséges algoritmikusan, mert nem lesz rekurzív. Triviális tulajdonság kizárólag a minden vagy a semmi.

Rice-tétel: Ha T egy nemtriviális nyelvi tulajdonság, akkor $L_T \notin R$. Ebből következik, hogy $L_\emptyset$ nem rekurzív, illetve minden a következő előadás elején. Rice-tételes feladat szinte mindig van zh-n.

Speedrun

- $L_u \in RE, L_h \in RE, L_d \notin RE$
- $L \in R \Leftrightarrow \overline{L} \in R$, RE esetében nem igaz
- $L \in RE$ és $\overline{L} \in RE \Leftrightarrow L \in R$
- $co\ X = \{L: \overline{L} \in X\}$
- $co\ co\ X = X$
- $X \subseteq Y \Rightarrow co\ X \subseteq coY$
- $co\ R = R$
- $RE \cap co\ RE = R$
- R és RE zártak az unióra, metszetre, konkatenálásra, és tranzitív lezártra
- Üres bemenetre megálló Turing-gépek osztálya: $L_\varepsilon \in RE - R$
- $L_\emptyset \in co\ RE, \overline{L_\emptyset} \notin co\ RE, \overline{L_\emptyset} \notin RE, L_\emptyset \notin R$
- L_T: T tulajdonsághoz rendelt nyelv
- Triviális nyelvi tulajdonság: $L_T \in R, L_{\overline{T}} \notin R$
- Nemtriviális nyelvi tulajdonság: $\exists\ L_1, L_2$: $L_1, L_2 \in RE$ és $L_1 \in T$ és $L_2 \notin T$, $L_T \notin R$, $T \cap RE \neq \emptyset, \overline{T} \cap RE \neq \emptyset$



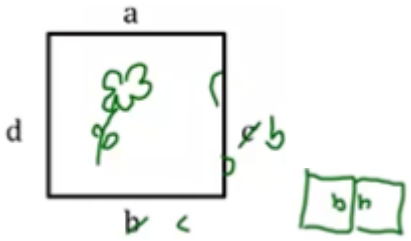
11. előadás

Ha azt akarjuk bizonyítani, hogy egy tulajdonság nem rekurzív (ami igazából szinte mindig igaz, mint beláttuk), **akkor csak adni kell egy olyan nyelvet, amire érvényes, és egy olyat, amire nem**:

1. T_1 : a nyelv nem üres $\quad L_{T_1} = \{w: \exists M_w,\ M_w$ elfogad valamilyen szót $\} \notin R$
   nem triv: $L_1 = \{\varepsilon\} \in RE \cap T, \quad L_2 = \emptyset \in RE - T$ (reguláris is)
2. T_2 : a nyelv véges $\quad L_{T_2} = \{w: \exists M_w,\ M_w$ véges sok szót fogad el $\} \notin R$
   nem triv: $L_1 = \emptyset \quad L_2 = 0^*$
3. $T_3 : |L| = 2020 \quad L_{T_3} = \{w: \exists M_w,\ M_w$ 2020 szót fogad el $\} \notin R$
   $L_1 = \{0, 0^1, \dots 0^{2020}\}, L_2 = \emptyset$
4. $T_4 : 011 \in L \quad L_{T_4} = \{w: \exists M_w,\ M_w$ elfogadja a 011 szót $\} \notin R$
   $L_1 = \{011\} \quad L_2 = \emptyset$
5. T_5 : L reguláris $\quad L_{T_5} = \{w: \exists M_w,\ L(M_w)$-hez van VA is $\} \notin R$
   $L_1 = \emptyset \quad L_2 =$ Palindrom
6. $T_6 : L = \Sigma^*$ $\quad L_{T_6} = \{w: \exists M_w,\ M_w$ mindent elfogad $\} \notin R$
   $L_1 = \Sigma^* \quad L_2 = \emptyset$
7. $T_7 : L \in R \quad L_{T_7} = \{w: \exists M_w, L(M_w)$ rekurzív $\} \notin R$
   $L_1 = \emptyset \quad L_2 = L_U$

Dominóprobléma

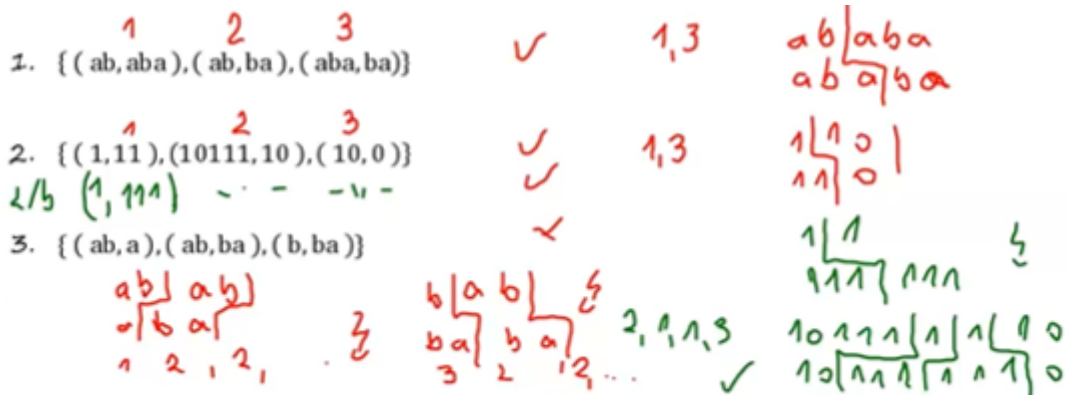
Vannak 1x1-es csempék, amiknek az oldalai címkézettek, és csak azonosan címkézett két oldal kerülhet egymás mellé:



A kérdés, hogy lefedhető-e a (végtelen) sík a készletünkkel, amit olyan formában kapunk meg, hogy $K = \{(a, b, b, b), (a, a, b, b)\}$. Ezekkel nem lefedhető, hiszen függőleges tengelyen mindkét elhelyezés rossz. A dominó nyelve $D = \{készlet: a\ sík\ lefedhető\ vele\}$. Igaz, hogy $K \in D \Leftrightarrow$ minden páros oldalhosszúságú négyzet lefedhető fele. Ez azt jelenti, ha el tudunk indulni 2x2-ből, majd köré tudunk épülni egy kerettel, aztán aköré is…, akkor le van fedve a sík. $D \in co\,RE \Leftrightarrow \overline{D} \in RE, D \notin R, D \notin RE$.

Post megfeleltetési problémája

Vegyünk egy véges szótárt, pl. magyar-angolt (azonos ábécé kell), és az a kérdés, hogy van-e olyan mondat, ami mindkét nyelven ugyanúgy néz ki. Erre néhány példa:



Itt a 3. azért nem igaz, mert az első nyelv minden szava b végű, a másiknál meg a, így eleve nem tudnak a mondatok ugyanúgy végződni.

A megfeleltetési probléma nyelve $PCP = \{szótár, amire\ van\ megoldás\}$. $PCP \in RE$, és legalább 2 betűs ábécékre $PCP \notin R$. $L = \{egyértelmű\ CF\ nyelvtanok\} \notin R$.

12. előadás

Minden (0. osztályú) nyelvtanhoz van olyan Turing-gép, aminek az elfogadása azonos azzal. Ez fordítva is igaz. Minden 1. osztályú G nyelvtan esetén $L(G) \in R$, de ez már nem igaz fordítva. CF nyelvekkel olyan 1 szalagos NTG ekvivalens, ami n hosszú bemenetre nem megy a szalagon $O(n)$-nél messzebb. Chomsky-nyelvosztályokra $L_3 \subset L_2 \subset L_1 \subset L_0 = RE$.

			melyik műveletre zártak				
nyelvtan	nyelvoszt	automata	∪	∩	kompl.	konkat	*
3	reguláris	VA	+	+	+	+	+
2	CF	PDA	+	-	-	+	+
	det CF	DPDA	-	-	+	+	+
1	CS	Lin.korlátos TG	+	+	+	+	+
0	RE	TG	+	+	-	+	+

Felsoroló Turing-gép

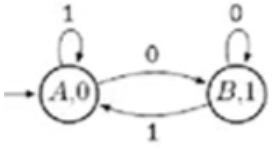
Olyan, mint a TG, de nincs elfogadó állapot, nem kap bemenetet, és lehetnek munkaszalagjai. Van kimeneti szalagja, ami csak írható, és visszalépni se lehet rajta. A kimeneti szalagon elválasztójel van a szavak közt. Az ilyen M által kiköpött szavakat felsorolt nyelvnek hívjuk. Ha M megáll, akkor csak véges nyelvet tud felsorolni. Egy nyelv csak akkor rekurzívan felsorolható (RE), ha van olyan TG, ami felsorolja, innen a neve. Egy végtelen nyelv csak akkor rekurzív, ha van a szavait sorrendben felsoroló TG. Véges nyelvekhez is van ilyen gép.

Kimenetes automata: valamit kiszámol, nem csak elfogad, rendes kimenete van. A VA és PDA fordító, a TG függvényt kiszámító. VA-nál két egyszerű modell: minden állapot vagy átmenet ad egy kimenetet. Az előbbit hívjuk úgy, hogy **Moore-automata**:

- Q: állapothalmaz
- Σ: ábécé
- Δ: kimeneti ábécé
- δ: állapotátmenetek
- μ: kimeneti függvény
- $q_0 \in Q$: kezdőállapot

Itt nincs olyan, hogy elfogadó állapot. Lényegében van egy szokásos teljes DVA-nk, ezt egészítjük ki azzal, hogy minden Q-hoz hozzárendelünk egy Δ-beli karaktert, ezt a μ függvény teszi meg. A q_0, q_1, q_2… számítási úton a kimenet $\mu(q_0), \mu(q_1), \mu(q_2)$… Ezt a μ-t ki lehet terjeszteni szavakra. Amit itt leírtunk, az hossztartó, de ez nem mindig van így.

Ilyen automata pl. az ellentétképző:



$\mu(000) =$ 111

$\mu(1011) =$ 0100

Az állapotban a neve mellett az van, hogy mit ír ki, ha belép az állapotba.

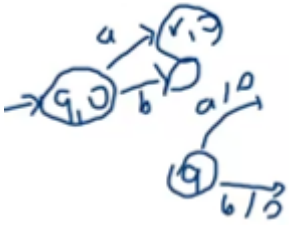
13. előadás

Mealy-automata

A Moore minden állapotra írt ki valamit, ez minden átmenetre. Ugyanúgy teljes DVA:

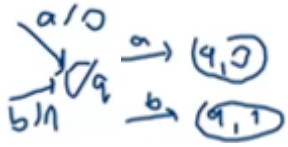
- Q: állapothalmaz
- Σ: ábécé
- Δ: kimeneti ábécé
- δ: állapotátmenetek
- τ: kimeneti függvény
- $q_0 \in Q$: kezdőállapot

Ebből is van olyan verzió, hogy az átmenet szavakra kiterjeszthető, de alapból hossztartó. Minden Moore-automata átalakítható Mealy-automatává. A módot nagyon egyszerűen ez a kép írja le:



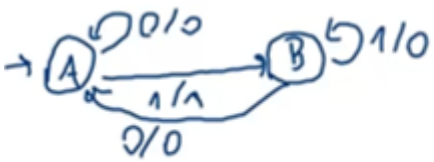
A Moore lépés után ír ki valamit, a Mealy ezt a lépés közben tegye meg. Az az automata, ami a bemenetet invertálja, így néz ki mindkét módon:

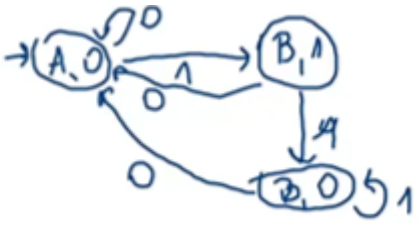
Minden Mealy-automata átalakítható Moore automatává. Ehhez olyan módszert használunk, ahol a célállapotot szedjük szét:



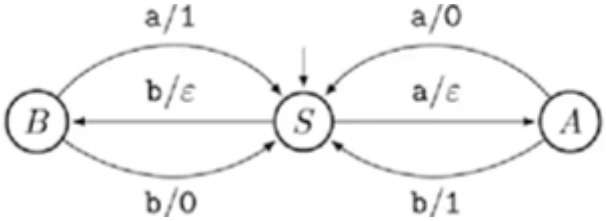
Példa az invertáló automata:

Példa arra, hogy minden 1-gyel kezdődő blokk első 1-e maradjon meg:





Egy Mealy-automata teljes értékkészlete reguláris. A **véges fordító** egy olyan Mealy-automata, ami megengedi tetszőleges hosszú karaktersor kiírását, vagyis az ε is lehetséges egy átmeneten. Ilyen például az automata, ami páronként kiírja, hogy azonosak vagy ellentétesek-e:



λ(aababbbba) =
0 1 0 0

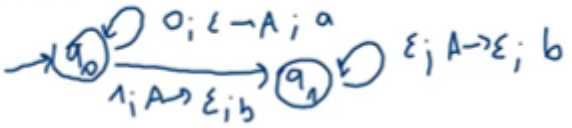
Véges fordító: egy nemdeterminisztikus VA, amiben lehetnek ε-mozgások is. Itt már fordításokról beszélünk, vagyis konkrét szópárokról, egy szótárról. **Véges fordítás** az a reláció, amihez létezik véges fordító. Reguláris nyelv fordítása szintén reguláris nyelv. CF nyelv fordítása szintén CF nyelv.

14. előadás

Ahogy az automatákat, a veremautomatákat is fordítóvá tehetjük, így jön létre a **veremfordító**, eredménye a **veremfordítás**. Ha elakad, sikertelen a fordítás. Ennek az átmenete kap egy extra paramétert a jobb oldalon, a kimenetre írt karaktert.

Példa: $Q = \{q_0, q_1\}$, $\Sigma = \{0,1\}$, $\Gamma = \{Z_0, A, B\}$, $\Delta = \{a, b\}$

$\delta: (q_0, 0, \varepsilon) = (q_0, A, a)$
$(q_0, 1, A) = (q_1, \varepsilon, b)$
$(q_1, \varepsilon, A) = (q_1, \varepsilon, b)$



Szintaxisvezérelt fordítási séma

Van két ABC alapján működő nyelvtan, de párhuzamosan írnak ki dolgokat. $G = (V, \Sigma, \Delta, P, S)$. A változók ugyanott helyezkednek el a szabályokban, csak a köztük lévő karakterek különböznek:

V: változók, Σ, Δ: két abc, $S \in V$: kezdőváltozó

P: nyelvtani szabályok: $A \to \alpha_1 ; \alpha_2$ $\quad \alpha_1 \in (\Sigma \cup V)^*$, $\alpha_2 \in (\Delta \cup V)^*$

$\alpha_1 = x_0 A_1 x_1 A_2 \cdots x_{k-1} A_k x_k$, $\quad \alpha_2 = y_0 A_1 y_1 A_2 \cdots y_{k-1} A_k y_k$

$x_i \in \Sigma^*$ $\quad y_i \in \Delta^*$ $\quad A_i \in V$

1. nyelvtan. $S \to 0S \mid 01$
2. $S \to 0S1 \mid 01$

1. *példa:* $S \to 0S ; 0S1 \mid 01 ; 01$

Egy levezetés: $S \Rightarrow 0S ; 0S1 \Rightarrow 00S ; 00S11 \Rightarrow 000S ; 000S111 \Rightarrow 00001 ; 00001111$

Fordítás = $\{(0^n 1, 0^n 1^n) : n \geq 1\}$

2. *példa:* $S \to AC ; 0A0C \mid C ; C \quad A \to aAbA ; 1AA \mid ab ; 2 \quad C \to c ; 3$

Egy levezetés: $S \Rightarrow \underline{A}C ; 0\underline{A}0C \Rightarrow a\underline{A}bAC ; 01\underline{A}A0C \Rightarrow$

$aabb\underline{A}C ; 012\underline{A}0C \Rightarrow aabbabC ; 01220C \Rightarrow$

$aabbabc ; 012203$

3. *példa:* $E \to E+T ; ET+ \mid T ; T$
$T \to T*F ; TF* \mid F ; F$
$F \to (E) ; E \mid a ; a$

$(a+a)*a$

$E \Rightarrow T ; T \Rightarrow T*F ; TF* \Rightarrow$
$F*F ; FF* \Rightarrow (E)*F ; EF* \Rightarrow$
$(E+T)*F ; ET+F* \Rightarrow \ldots$
$\Rightarrow (a+a)*a ; aa+a*$

$(a+a)*(a+a)$

15. előadás

Foglalkozunk azzal, hogy milyen gyorsan végez a Turing-gép. M gép időigénye $T_M(n)$, ez a lépések száma n hosszú bemeneten. A tárigény a maximálisan használt szalagmezők száma, jele $S_M(n)$. A tárigénybe nem számít a bemenet, ha nem írható felül. Ha tudunk felső becslést adni valamelyikre, akkor időkorlátos vagy tárkorlátos a TG. Ez nem elég jó besorolás, osztályozunk:

- $TIME(f(n))$: van DTG, ami felismeri, és időigénye $O(f(n))$
- $SPACE(f(n))$: van DTG, ami felismeri, és tárigénye $O(f(n))$
- $NTIME(f(n))$: van NTG, ami felismeri, és időigénye $O(f(n))$
- $NSPACE(f(n))$: van NTG, ami felismeri, és tárigénye $O(f(n))$

Még osztályok:

- P: polinomiális időben felismerhető nyelvek osztálya
- NP: nemdeterminisztikus gépekkel polinom időben felismerhető nyelvek osztálya
- P és NP zárt az unióra, metszetre, konkatenációra, tranzitív lezártra
- Tárhelyre ugyanez a $PSPACE$ és $NPSPACE$ (amúgy ezek egyenlők)
- EXP: exponenciális időben felismerhető nyelvek osztálya

Tulajdonságaik (leszopod magad):

- $TIME(f(n)) \subseteq NTIME(f(n)) \subseteq R$
- $SPACE(f(n)) \subseteq NSPACE(f(n)) \subseteq RE$
- $TIME(f(n)) \subseteq TIME(g(n))$, $ha\ f(n) \leq g(n)$, $n = 1, 2\ldots$
- $SPACE(f(n)) \subseteq SPACE(g(n))$, $ha\ f(n) \leq g(n)$, $n = 1, 2\ldots$
- $TIME(f(n)) = co\ TIME(f(n))$
- $SPACE(f(n)) = co\ SPACE(f(n))$
- $SPACE(f(n)) \subseteq R$
- $P \subseteq NP \subseteq PSPACE \subseteq EXP$
- $P \neq EXP$
- $P = co\ P$
- $P \subseteq co\ NP$
- Ha $f(n) \geq n$, akkor $NTIME(f(n)) \subseteq TIME(2^{c*f(n)})$
- Ha $f(n) \geq log\ n$, akkor $NSPACE(f(n)) \subseteq SPACE(f^2(n))$

16. előadás

- $L \in RE$: $x \in L$-re van véges hosszú bizonyíték
- $L \in co\,RE$: $x \notin L$-re van véges hosszú bizonyíték
- $L \in RE \cap co\,RE$: $x \in L$-re és $x \notin L$-re is van véges hosszú bizonyíték
- $L \in R$: $x \in L$ eldöntésére van algoritmus (megálló TG)
- $L \in NP$: $x \in L$-re van polinom hosszú bizonyíték
- $L \in co\,NP$: $x \notin L$-re van polinom hosszú bizonyíték
- $L \in NP \cap co\,NP$: $x \in L$-re és $x \notin L$-re is van polinom hosszú bizonyíték
- $L \in P$: $x \in L$ eldöntésére van polinom idejű algoritmus

Karp-redukció

Az összes lehetséges szóra (Σ^*) értelmezett L_1 és L_2 nyelv, ezek közt egy olyan $f(x)$ függvényt hívunk Karp-redukciónak, ami úgy képez le, hogy ami L_1-en belüli, az L_2-n belüli lesz, de ami azon kívüli, az kívül marad, és ez P. Ha létezik, akkor jele: $L_1 \prec L_2$. Ez azt szimbolizálja, hogy L_2 legalább olyan nehéz, mint L_1. Ha van Turing-gép az L_2-re, és meg is áll, akkor L_1-re is. Ha az egyik gép P vagy $PSPACE$, a másik is az lesz.

Egy nyelv ***NP*-teljes**, ha minden NP-beli nyelv visszavezethető rá, tehát a legnehezebbek közt van NP-n belül. Ugyanilyen logika szerint van $PSPACE$-teljes és P-teljes nyelv is, de utóbbinál a logaritmikus tárban számolhatóság a feltétel. NP-teljes feladat például gráfokban Hamilton-körök keresése, az utazóügynök, és a hátizsák probléma. $PSPACE$-teljes például a mindent elfogadó NVA, CS nyelvtanok nyelve, kvantoros Boole-formulák igazságának ellenőrzése.

A Turing-gépek jók, mert egy lépésben kevés dolog történik, jól lehet becsülni a lépésszámot. A mai számítógépek már közvetlen elérésű gépek (Random Access Machine), a becslések mégis jól használhatók, mivel a gép szintén idealizált és determinisztikus, csak szalag helyett memória van, de a lépések az utasításkészletben elég egyszerűek. Külön kezeljük a programmemóriát és adatmemóriát, illetve a bemeneti és kimeneti szalagot. Minden DTG-hez van ekvivalens RAM program.