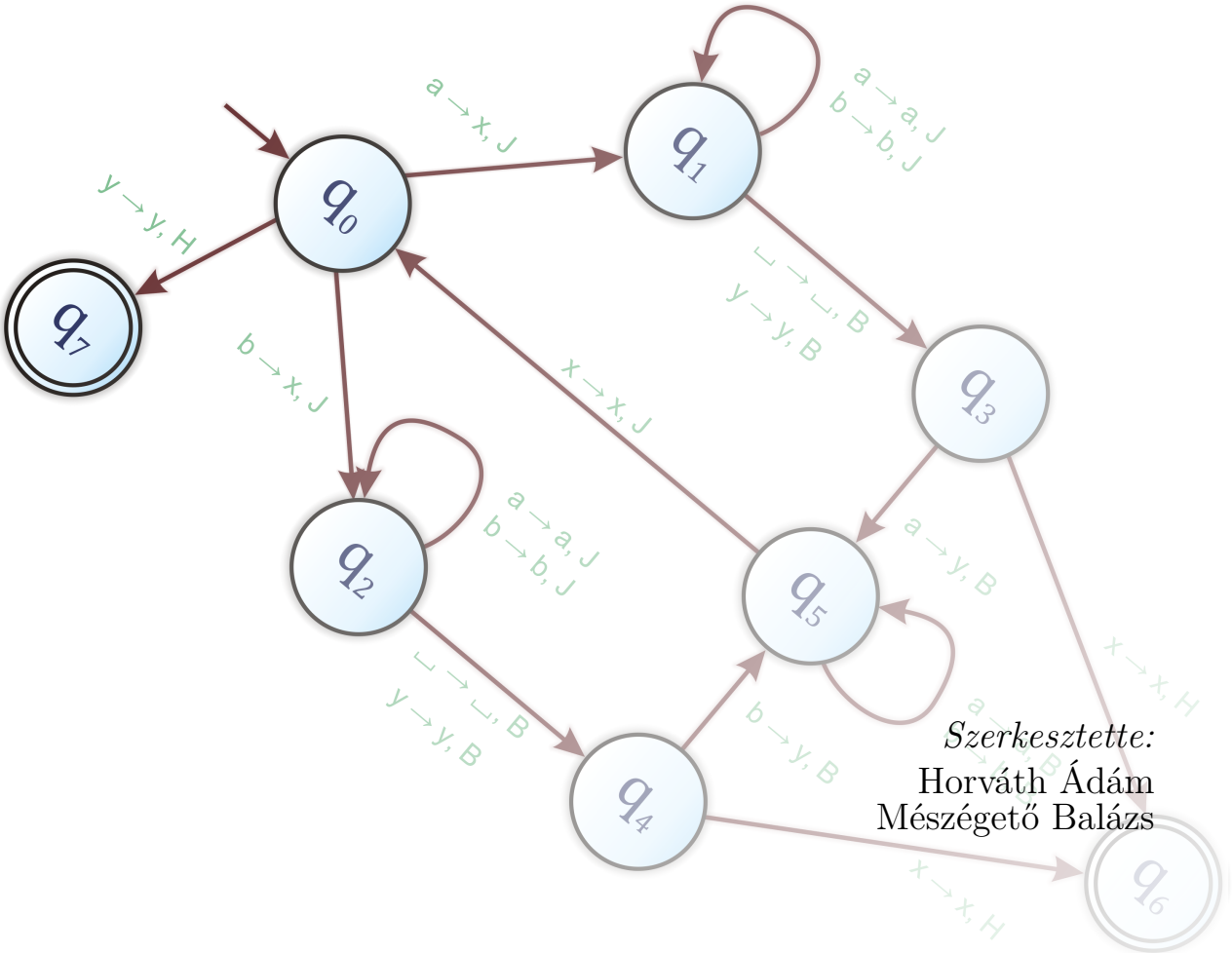
M Ű E G Y E T E M 1 7 8 2
Budapesti Műszaki és Gazdaságtudományi Egyetem

dr. Friedl Katalin

Nyelvek és Automaták

Órai jegyzet, 2010.



Szerkesztette:
Horváth Ádám
Mészégető Balázs

Előszó

A jelen jegyzet elsősorban a Budapesti Műszaki és Gazdaságtudományi Egyetem (BME) mesterképzéses mérnökinformatikus hallgatói számára készült. A *Nyelvek és Automaták* tantárgy a képzés – egy kivétellel – összes szakirányán kötelező, ezért reményeink szerint sok diáktársunk segítségére lehet a jegyzet.

A Szerkesztők maguk is a tárgy hallgatói voltak a 2010/2011-es tanév első félévében, ezért a jelenlegi, első kiadás ezen félév előadásai alapján íródott. Felépítésében, szerkesztettségében, kidolgozottságban, szó- és ábrahasználatában nagy mértékben épít az ott leírtakra és elhangzottakra.

A legnagyobb igyekezetünk ellenére előfordulhat, hogy a kedves Olvasó hibát talál a Jegyzetben. Minden hibajelzést szívesen fogadunk, kérjük ez esetben küldje el nekünk a `mesz86@gmail.com` címre!

Külön köszönet a tantárgy előadójának, *dr. Friedl Katalin*nak, aki készségesen válaszolt a kérdéseinkre, ezzel is segítve a munkánkat. Köszönjük, hogy a lektorálás munkáját is elvállalta.

Sikeres felkészülést kívánunk:

a Szerkesztők.

Tartalomjegyzék

1. fejezet

Reguláris nyelvek és véges automaták

1.1. Definíciók

Abc: Egy véges, nem üres halmaz. Jele: Σ

Szó: Σ elemeiből képzett véges hosszú sorozat.
Például: $\Sigma = \{0, 1\}$, akkor szavak lehetnek: 0110, 0, ε (üres szó)
Az i hosszú szavak halmazát így jelöljük: Σ^i. Például: $\Sigma^0 = \{\varepsilon\}$
Az összes Σ feletti szó jelölése: Σ^*

$$\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$$

Nyelv: $L \subseteq \Sigma^*$

Példák nyelvekre:

- $L_1 = \emptyset$
- $L_2 = \{\varepsilon\}$
- $L_3 =$ páros sok a-t tartalmazó szavak.

1.1. Feladat. Mekkora memória szükséges annak eldöntéséhez, hogy egy adott szó beltartozik a nyelvbe az alábbi nyelvek esetén? Minden szót csak egyszer olvasunk be.

- $\Sigma = \{0, 1\}$; $L_1 =$ Nullával kezdődő szavak.
- $\Sigma = \{0, 1\}$; $L_2 =$ Nullára végződő szavak.
- $\Sigma = \{0, 1\}$; $L_3 =$ Azok a szavak, amelyekben az utolsó előtti karakter nulla.
- $\Sigma = \{0, 1\}$; $L_4 =$ Páros sok egyest tartalmazó szavak.
- $\Sigma = \{(,)\}$; $L_5 =$ A helyes zárójelezések.

1.1. Megoldás. L_1 és L_2 esetén elegendő 1 bit, mert csak az első illetve az utolsó számjegyet kell nyilvántartanunk. L_3 esetén két bit elegendő, mert az utolsó és az utolsó előtti számjegyre van szükségünk. L_4 esetén ismét elég egyetlen bit, amelyet átbillentünk minden egyes beolvasásakor. L_5 esetén figyelnünk kell a zárójelezés mélységét. Egyrészt minden kinyitott zárójelet be kell zárnunk másrészt a zárójelezés mélysége nem mehet nulla alá. Például egy rossz zárójelezés: „) (”. Így összesen tehát „nyitó-csukó” tárolásához elegendő bit kell plusz még egy bit.

1.2. Véges automaták

Definíció (Véges automata)**.** $M = (Q, \Sigma, \delta, q_0, F)$, ahol:

- Q az automata állapotainak halmaz. Véges, nem üres halmaz.
- Σ az automata abc-je. Véges, nem üres halmaz.
- δ az állapotátmeneti függvény. $Q \times \Sigma \to Q$
- q_0 a kezdőállapot, $q_0 \in Q$
- F az elfogadó állapotok halmaza, $F \subseteq Q$

Számítás: A $w = a_1 a_2 \dots a_n \in \Sigma^*$ szó esetén $r_0 r_1 \dots r_n \in Q$ egy számítás, ha: $r_0 = q_0$ és $r_i = \delta(r_{i-1}, a_i)$, $i = 1 \dots n$ esetén.

M elfogadja w szót: ha w-hez tartozó számítás esetén $r_n \in F$.

M által elfogadott nyelv: azoknak a szavaknak a halmaza, amiket M elfogad. Jele: $L(M)$.

1.2. Feladat. Adjuk meg azt az automatát, amely a nullára végződő szavakat tartalmazó nyelvet fogadja el! $\Sigma = \{0, 1\}$.

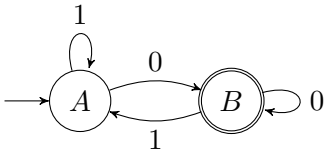
1.2. Megoldás.

$$\begin{aligned} Q &= \{A, B\} \\ \Sigma &= \{0, 1\} \\ \delta(A, 0) &= B \\ \delta(A, 1) &= A \\ \delta(B, 0) &= B \\ \delta(B, 1) &= A \\ q_0 &= A \\ F &= \{B\} \end{aligned}$$

Az állapotátmeneti függvény táblázattal is megadható:

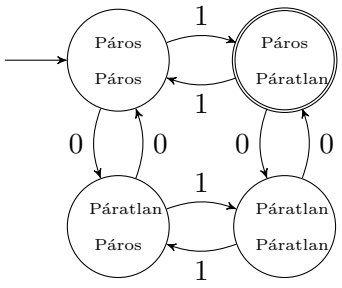
	0	1
A	B	A
B	B	A

Az automatát irányított gráffal is ábrázolhatjuk. A csomópontok az állapotokat, az élek pedig átmeneti függvényt jelölik. A kezdő állapotot egy bemenő nyíllal, az elfogadó állapotokat dupla körvonallal illusztráljuk.

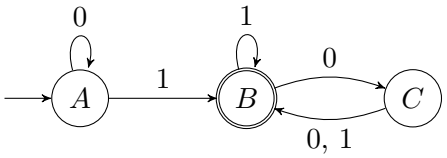


1.3. Feladat. Adjuk meg azt az automatát, amely a páros sok nulla és páratlan sok egyest tartalmazó szavakat fogadja el! $\Sigma = \{0, 1\}$.

1.3. Megoldás. Az automata az alábbi:



1.4. Feladat. Mely szavakat fogadja el az alábbi automata? $\Sigma = \{0, 1\}$.



1.4. Megoldás. Az egyes állapotok jelentése:

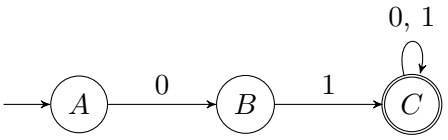
- A: nincs benne egyes
- B: van benne egyes, és a végén páros számú nulla van
- C: van benne egyes, és a végén páratlan számú nulla van

Ebből a B az elfogadó állapot, ezért az automata azokat a szavakat fogadja el, amelyekben van egyes, és a végén páros számú nulla van.

1.3. Hiányos véges automaták

Definíció. Hiányos véges automata esetén az állapotátmeneti függvény (δ) nincs mindenhol definiálva.

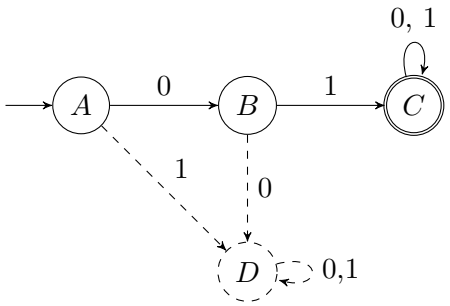
1.1. Példa. Az alábbi hiányos véges automata azokat a szavakat fogadja el, amelyek a 01 sorozattal kezdődnek, $\Sigma = \{0, 1\}$.



1.1. Tétel. Minden hiányos véges automata kiegészíthető teljes véges automatává úgy, hogy az elfogadott nyelv nem változik.

1.1. Bizonyítás. Vegyünk fel egy új állapotot, és minden hiányzó átmenetet kössünk be ebbe az új állapotba. Az új állapotból bármely bemenet hatására maradjunk az új állapotban. Az így kapott automata egy olyan teljes véges automata amely éppen ugyanazt a nyelvet fogadja el, mint amelyikből kiindultunk. □

1.2. Példa. A 01 sorozattal kezdődő szavakat elfogadó automatát például a következőképpen egészíthetjük ki:



Definíció. L nyelv *reguláris*, ha létezik olyan M véges automata, hogy $L(M) = L$.

1.2. Tétel. L_1 és L_2 reguláris, akkor

- $L_1 \cup L_2$
- $L_1 \cap L_2$
- $L_1 - L_2$

is reguláris.

1.2. Bizonyítás. Konstrukcióval. Legyenek M_1 és M_2 az egyes nyelveket elfogadó automaták:

$$M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$$
$$M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$$

Hozzunk létre egy olyan $M = (Q, \Sigma, \delta, q_0, F)$ automatát, amely éppen a két nyelv unióját, metszetét, illetve különbségét fogadja el! Ekkor M automata:

- $\Sigma = \Sigma_1 \cup \Sigma_2$. Előfordulhat, hogy Σ-ra nézve M_1 és M_2 hiányos. Ekkor egészítsük ki őket a korábbi tétel alapján, hogy teljesek legyenek!
- Az új állapothalmaz legyen az előzőek Descartes-szorzata: $Q = Q_1 \times Q_2$.
- Az új állapotátmeneti függvényt alakítsuk úgy, hogy az egyes „koordináták” mentén mozogjunk az egyes automaták állapotátmeneti függvénye szerint:

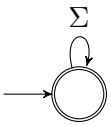
$$\delta((q, q'), a) = (\delta_1(q, a), \delta_2(q', a))$$

- Az új kezdőállapot: $q_0 = (q_1, q_2)$
- Az elfogadó állapotok halmaza:
  - Únió esetén: $F = \{(q, q') : (q \in F_1) \vee (q' \in F_2)\}$
  - Metszet esetén: $F = \{(q, q') : (q \in F_1) \wedge (q' \in F_2)\}$
  - Különbség esetén: $F = \{(q, q') : (q \in F_1) \wedge (q' \notin F_2)\}$

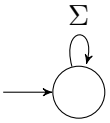
Az így létrehozott M automata éppen a megfelelő w szavakat fogadja el, mert ha $w \in L_i$, akkor M megfelelő állapotának i-edik koordinátája elfogadó az M_i automatában, és minden w szó esetén a hozzátartozó M-beli számítás i. kooridnátája egy számítás az M_i automatában. □

1.3. Tétel. Σ^* és az üres nyelv reguláris.

1.3. Bizonyítás. Egy-egy automatával bizonyítjuk a tételt. Az alábbi automata mindent elfogad:



Az alábbi automata semmit sem fogad el:



□

1.4. Tétel. Ha L nyelv reguláris, akkor $\overline{L}$ nyelv is reguláris.

1.4. Bizonyítás. Az 1.3 és az 1.2 tételek egyszerű következménye: $\overline{L} = \Sigma^* - L$ reguláris.
Megjegyzés: Közvetlenül is létrehozhatunk $\overline{L}$-t elfogadó automatát, ha L automatájában az elfogadó és nem elfogadó állapotokat felcseréljük. □

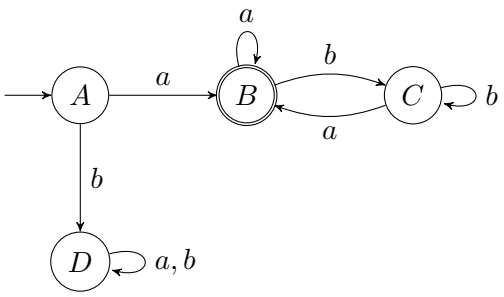
1.5. Feladat. Hozzunk létre olyan automatát, amely az alábbi L nyelvet fogadja el!

$$\Sigma = \{a, b\}$$
$$L = \{w \in \Sigma^* : w \text{ első és utolsó karaktere megegyezik}, w \text{ hossza} \geq 1\}$$

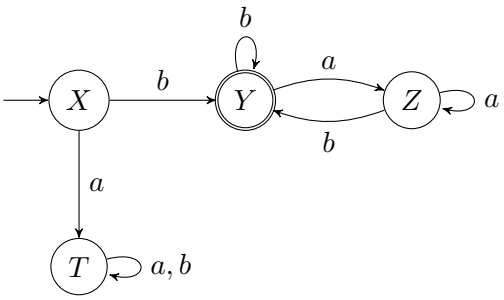
1.5. Megoldás. Legyen $L = L_a \cup L_b$ a következőképpen:

$$L_a : \{w \in \Sigma^* : w \text{ első és utolsó karaktere} = a, w \text{ hossza} \geq 1\}$$
$$L_b : \{w \in \Sigma^* : w \text{ első és utolsó karaktere} = b, w \text{ hossza} \geq 1\}$$

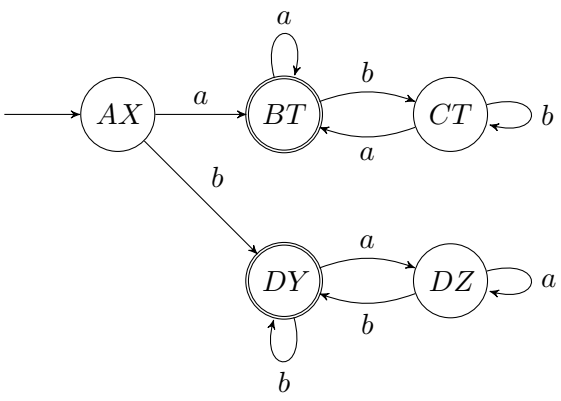
L_a:



L_b:



A korábbi konstrukció alapján előállíthatjuk a két automata unióját, azonban sok olyan állapot lesz, amelyet a kezdőállapotból nem érhetünk el. Érdemes ezért a kiinduló állapotból végigkövetni az elérhető állapotokat és így létrehozni az únionak megfelelő automatát. A végeredmény tehát:



1.4. Nemdeterminisztikus véges automaták

Definíció (Nemdeterminisztikus véges automata)**.** A nemdeterminisztikus véges automaták esetén egy bemenet hatására több állapotba is léphetünk és akár bemenet nélkül is átléphetünk egyik állapotból egy másikba.

Formálisan: $M = (Q, \Sigma, \delta, q_0, F)$ gép esetén Q, Σ, q_0 és F jelentése változatlan, azonban az átmeneti függvény ezúttal:

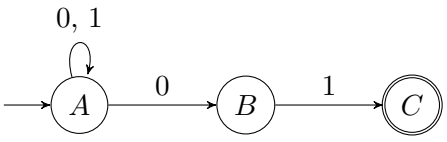
$$\delta(q, a) \subseteq Q,$$

ahol $q \in Q$ és $a \in \Sigma \cup \{\varepsilon\}$.

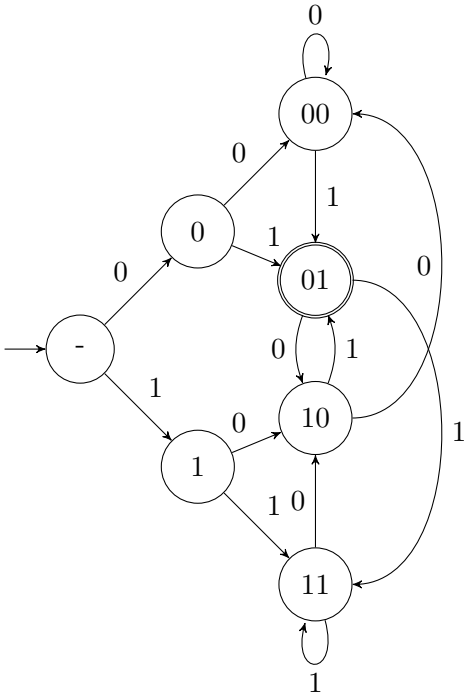
Számítás: $r_0 = q_0$ és $r_1 \dots r_k$, ahol $r_i \in \delta(r_{i-1}, \varepsilon) \cup \delta(r_{i-1}, a_j)$, ha az i-edik lépésig a_{j-1}-ig olvastuk a szót, és r_k-ig elolvastuk az egész szót.

M elfogadja w szót: ha van olyan számítás, amelyre $r_k \in F$.

1.3. Példa. A következő nemdeteminisztikus véges automata a 01-re végződő szavakat fogadja el:



Ugyenez determinisztikus véges automatával:



1.5. Tétel. Az L nyelvhez létezik nemdeterminisztikus véges automata $\Leftrightarrow$ L nyelv reguláris.

1.5. Bizonyítás. Külön bizonyítjuk a két irányt.

- $\Leftarrow$ irányban: L reguláris $\Rightarrow \exists L$-hez determinisztikus véges automata $\Rightarrow \exists L$-hez nemdeterminisztikus véges automata, mert a véges automata egyben nemdeterminisztikus is.

- $\Rightarrow$ irányban: Legyen az L-hez tartozó nemdeterminisztikus véges automata $M = (Q, \Sigma, \delta, q_0, F)$! Ehhez hozzunk létre olyan $M' = (Q', \Sigma', \delta', q_0', F')$ véges automatát, hogy $L(M') = L(M)$!

  Először tegyük fel, hogy nincs $\delta(q, \varepsilon)$ típusú átmenet. Ekkor M':

$$\begin{aligned} Q' &= \{R : R \subseteq Q\} \\ \Sigma' &= \Sigma \\ \delta'(R, a) &= \bigcup_{r \in R} \delta(r, a) \\ q_0' &= \{q_0\} \\ F' &= \{R : R \bigcap F \neq \emptyset\} \end{aligned}$$

  A konstrukció helyességének indoklása:

  Ha $w \in L(M)$, akkor $\exists r_0, r_1 \dots r_k$ számítás, hogy $r_k \in F$.

$$\begin{aligned} R_1 = \delta'(\{r_0\}, a_1) &\Rightarrow r_1 \in R_1 \\ R_2 = \delta'(R_1, a_2) \supseteq \delta(r_1, a_2) &\Rightarrow r_2 \in R_2 \\ &\vdots \\ r_k \in R_k &\Rightarrow R_k \in F' \end{aligned}$$

  Tehát ha M automata elfogadja w-t, akkor az M' is.

  Ha $w \in L(M')$, akkor a szóhoz tartozó $R_0, R_1 \dots R_k$ számítás esetén $R_k \in F' \Rightarrow$

k: $\exists r_k \in R_k$, hogy $r_k \in F$

$k-1$: $\exists r_{k-1} \in R_{k-1}$, amelyre $r_k \in \delta(r_{k-1}, a_k)$

$k-2$: $\exists r_{k-2} \in R_{k-2}$, amelyből a nemdeterminisztikus automatában az a_{k-1}, a_k lépésekkel r_k -ba juthatunk.

$\vdots$

0: $\exists r_0 \in R_0 = \{q_0\}$, amelyre a nemdeterminisztikus automatában az $a_1, a_2 \dots a_k$ lépésekkel r_k-ba juthatunk.

Tehát összességében $\exists r_0 = q_0, r_1, \dots, r_k$ számítás M-ben w szóra, hogy $r_k \in F$, azaz ha M' elfogadja w szót, akkor M is.

Most nézzük meg hogyan módosul a konstrukció, ha $\delta(q, \varepsilon)$ típusú átmeneteket is tartalmaz M! Ehhez vezessük be az alábbi jelölést:

$$E(R) = \{q \in Q : R\text{-ből a } q\text{-ba el lehet jutni csak } \varepsilon \text{ éleken}\} \cup R$$

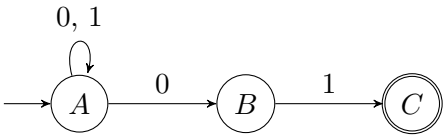
Ekkor a fenti konstrukció a következőképpen módosul:

$$\delta'(R, a) = \bigcup_{r \in R} E(\delta(r, a))$$
$$q_0' = E(\{q_0\})$$

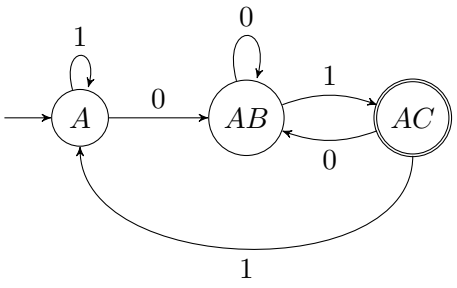
Az előző konstrukcióhoz hasonlóan igazolható az új konstrukció helyessége is.

□

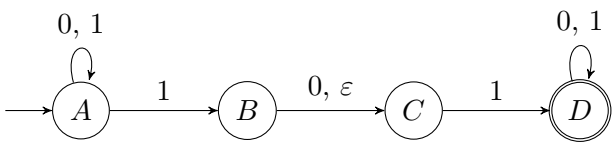
1.4. Példa. A fenti konstrukcióval állítsuk elő a 01 végződésű szavakat elfogadó determinisztikus automatát! $\Sigma = \{0, 1\}$.
Nemdeterminisztikus automata:



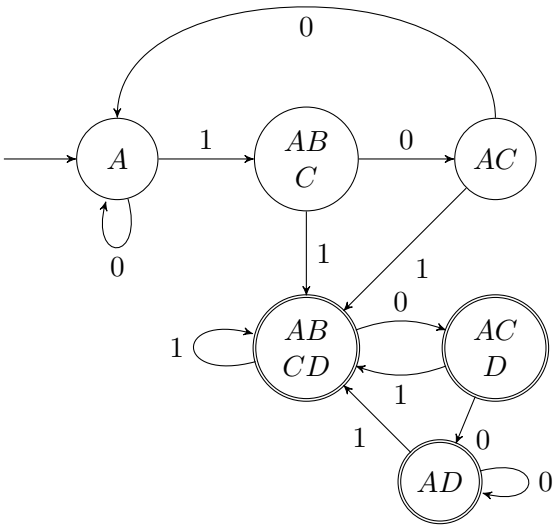
Determinisztikus automata:



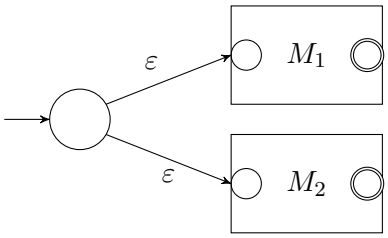
1.6. Feladat. Milyen nyelvet fogad el az alábbi automata? Készítsük el az ugyanezt a nyelvet elfogadó determinisztikus automatát! $\Sigma = \{0, 1\}$.



1.6. Megoldás. $L(M)$: Van benne 101 vagy 11.



A konstrukcióból kapott determinisztikus automata jelen esetben egyszerűsíthető az elfogadó állapotok összevonásával.
Az ε állapotok segítségével egyszerűen létrehozhatjuk azt az automatát, amely két nyelv unióját fogadja el. Legyen L_1 és L_2 a két nyelv. Valamint az őket elfogadó automaták rendre M_1 és M_2. Ekkor a két nyelv unióját az alábbi automata fogadja el:

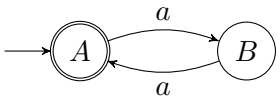


1.7. Feladat. Az előző konstrukció segítségével adjuk meg azt a determinisztikus automatát, amely az alábbi nyelvek unióját fogadja el! $\Sigma = \{a\}$.

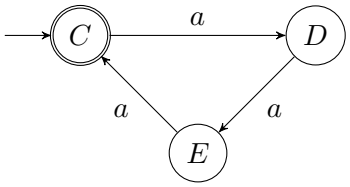
L_1 : páros sok a-ból álló szavak.

L_2 : hárommal osztható darab a-ból álló szavak.

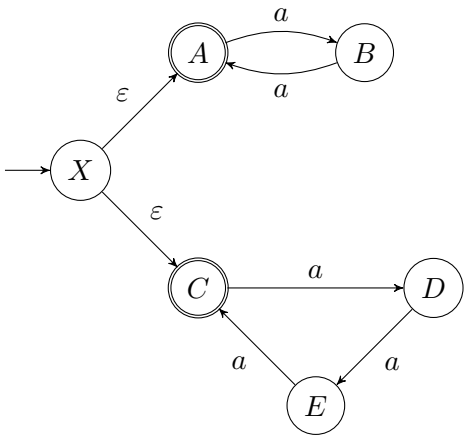
1.7. Megoldás. Az L_1 nyelvhez tartózó M_1 automata:



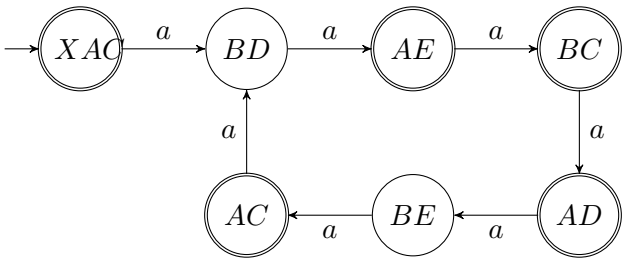
Az L_2 nyelvhez tartózó M_2 automata:



Az úniót elfogadó nemdeterminisztikus automata:



Az előző automata determinizálva:



Vegyük észre, hogy az „XAC” állapot összevonható az „AC”-vel.

1.5. Megkülönböztethetőség, ekvivalencia osztályok

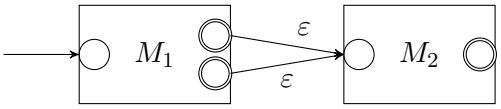
Definíció. Nyelvek *konkatenáltján* azt a nyelvet értjük, amelynek mindazok a szavak elemei, melyeket szét lehet vágni két részre úgy, hogy az első rész az első nyelv eleme, a második rész a másodiké. Formálisan:

$$L_1L_2 = \{w_1w_2 : w_1 \in L_1, w_2 \in L_2\}$$

1.6. Tétel. L_1 és L_2 reguláris, akkor L_1L_2 is reguláris.

1.6. Bizonyítás. Mivel L_1 és L_2 reguláris, léteznie kell olyan M_1 és M_2 automatáknak, melyekre: $L(M_1) = L_1$ és $L(M_2) = L_2$. Hozzuk lére a L_1L_2 nyelvet elfogadó automatát a következőképpen:

- M_1 elfogadó állapotaiból induljon egy-egy ε átmenet M_2 kezdőállapotába.
- Az automata kezdőállapota legyen azonos M_1 kezdőállapotával.
- Az automata elfogadó állapotai legyenek M_2 elfogadó állapotai.



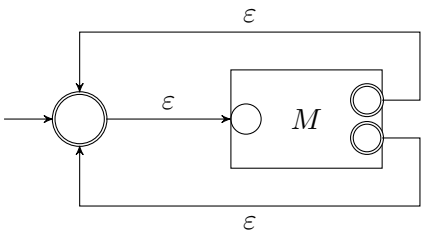
□

Definíció. Az L nyelv *tranzitív lezárt*ja: az L nyelv szavaiból nulla vagy több darabot egymás után konkatenálunk. Formálisan:

$$L^* = \{w_1w_2 \dots w_k, w_i \in L, k \geq 0\}$$

1.7. Tétel. Ha L reguláris, akkor L^* is reguláris.

1.7. Bizonyítás. Az alábbi automata L nyelv tranzitív lezártját fogadja el, ha $L(M) = L$.



□

1.5. Példa. Néhány példa a tranzitív lezártra:

- L : 01-re végződő szavak, $\Sigma = \{0, 1\} \Rightarrow L^* : \{\varepsilon\} \cup L$
- L : páratlan sok egyest tartalmazó szavak, $\Sigma = \{0, 1\} \Rightarrow L^* : \{\varepsilon\} \cup$ { Egyest tartalmazó szavak. }

Definíció. Ha $L \subseteq \Sigma^*$ egy nyelv és $x \in \Sigma^*$, akkor:

$$L/x = \{y \in \Sigma^*, xy \in L\}$$

1.6. Példa. Néhány példa az L/x jelölés jelentésére. Legyen

- L_1 : 01-gyel kezdődő szavak!
- L_2 : Páros sok egyest tartalmazó szavak!

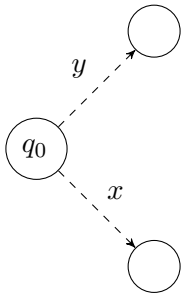
Ekkor:

- $L_1/011 = \Sigma^*$
- $L_1/10 = \emptyset$
- $L_2/011 = L_2$
- $L_2/010 = \overline{L_2}$
- Minden nyelvre: $L/\varepsilon = L$

Definíció. Ha $x, y \in \Sigma^*$, akkor x és y az L nyelvvel

- *megkülönböztethetetlen*, ha $L/x = L/y$.
- *megkülönböztethető*, ha $L/x \neq L/y$, azaz $\exists z \in \Sigma^*$, hogy ($xz \in L$ és $yz \notin L$) vagy ($xz \notin L$ és $yz \in L$).

Ha x és y megkülönböztethető L-el és L reguláris, valamint $M(L) = L$, akkor M automatában a kezdőállapotból x és y szavak hatására biztosan különböző állapotokba jutunk.



Következmény: Ha $\exists t$ darab páronként megkülönböztethető szó L-hez és $L(M) = L$, akkor M-nek legalább t állapota van.

1.7. Példa. Legyen L 01-re végződő szavakból álló nyelv:

- $x_1 = 0$ és $x_2 = 1$ szavakat a $z_1 = 1$ megkülönbözteti egymástól.
- $x_3 = 01$-et $z_2 = \varepsilon$ különbözteti meg x_1-től és x_2-től.

1.8. Példa. Legyen $L = \{ww : w \in \Sigma^*\}$, vagyis a „mindent kétszer mond" nyelv. Ekkor bármely két k hosszú szó megkülönböztethető egymástól (saját maguk segítségével). Mivel ez minden k-ra teljesül, ezért nem létezhet a nyelvhez véges automata, vagyis L nem reguláris.

Figyeljük meg, hogy a megkülönböztethetetlenség egy olyan reláció két szó között, amely:

- szimmetrikus, hiszen ha az egyik szó megkülönböztethetetlen a másiktól, akkor a másik is az egyiktől
- reflexív, mivel minden szó megkülönböztethetetlen saját magától
- tranzitív, ami a definíció következménye

A fenti tulajdonságok miatt a megkülönböztethetetlenség egy ekvivalenciareláció, amely a szavakat ekvivalenciaosztályokba osztja, úgy hogy az egyes osztályokba tartozó szavak mind páronként megkülönböztethetetlenek egymástól. A következőképpen beláthatjuk, hogy minden véges automata esetén az automata által elfogadott nyelv szavai legfeljebb $|Q|$ ekvivalencia osztályba sorolhatók. Vegyük az alábbi L_q részhalmazait a nyelvnek:

$$L_q = \{x \in \Sigma^* : x \text{ a } q \text{ állapotban ér véget}\}$$

Minden ilyen L_q valamely ekvivalenciaosztály részhalmazát képezi.
Ekvivalens állapotok: p és q állapot ekvivalens, ha $\forall y \in \Sigma^*$ esetén p-ből y szó beolvasásával pontosan akkor érkezünk elfogadó állapotba, ha q-ból is.

Az állapotok közötti ekvivalencia szintén ekvivalenciareláció, így az automata állapotait ekvivalenciaosztályokra osztja.

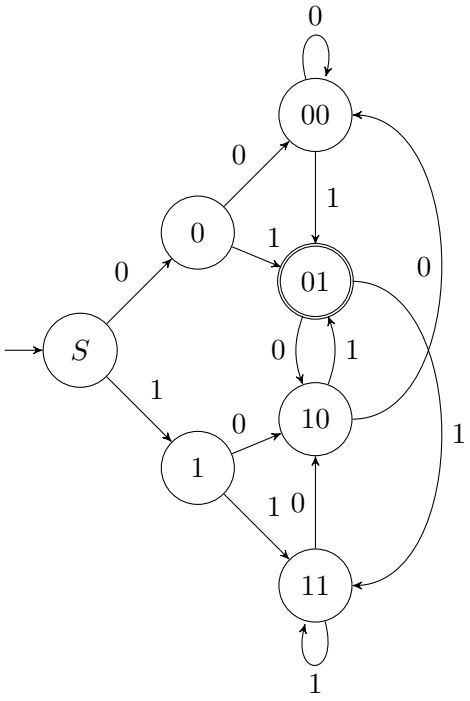
1.1. Algoritmus (Ekvivalencia osztályok meghatározása)**.** Az algoritmus teljes véges determinisztikus automatából indul ki. Az állapotok halmazát lépésenként fogjuk felosztani egyre kisebb halmazokra. Először osszuk fel $A_1 = F$ és $A_2 = Q - F$ halmazokra. Ezután minden lépésben vizsgáljuk meg az eddigi halmazainkban lévő állapotok viselkedését minden $a \in \Sigma$ karakterre. Ha az illető állapothalmaz elemei nem azonosan viselkednek, azaz van olyan karakter, amelynek hatására az egyik elemből más halmazba jutunk, mint a másikból, akkor vágjuk szét eszerint a halmazt. Az algoritmusnak akkor van vége, ha valamely lépésben már semelyik halmazt sem kellett tovább osztani. Az algoritmus biztosan véget ér, hiszen véges sok állapotunk van. A kialakult $A_1, A_2 \dots A_k$ állapothalmazok az automata állapotainak ekvivalencia osztályai.
A kapott osztályon definiálhatunk egy automatát úgy, hogy az állapotátmenetek éppen azok legyenek, mint az állapothalmazok közötti állapotátmenetek. Ez az automata ugyanazt a nyelvet fogadja el, mint az eredeti.

1.8. Tétel. Az 1.1 algoritmussal kapott automata minimális számú állapotot tartalmaz. (A teljes véges determinisztikus automatákat tekintve.)

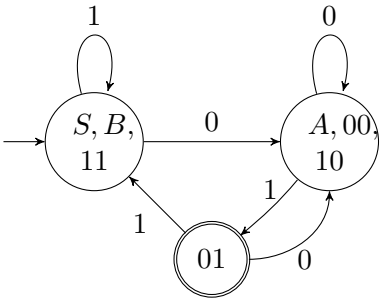
1.8. Bizonyítás. Vegyük az $x, y \in \Sigma^*$ megkülönböztethetetlen szavakat ($L/x = L/y$), melyek az M automatában a kezdőállapotból rendre a p és q állapotba visznek. M-ben ekkor p és q ekvivalens, hiszen ellenkező esetben létezne olyan folytatása x-nek és y-nak, amivel meg lehetne őket különböztetni. Az algoritmus szerint létrejövő automatában ekkor p és q biztosan ugyanabba

az új állapotba kerül. Tehát bármely két ekvivalens szóhoz tartozó állapot is ekvivalens lesz. Így a szavakon definiált ekvivalencia osztályok megfelelnek az állapotokon definiált ekvivalencia osztályoknak. Az automatának legalább annyi állapota kell, hogy legyen, mint ahány ekvivalencia osztály van a szavakra nézve. Mivel az állapotok ekvivalencia osztályaiból ugyanannyi van, mint a szavak ekvivalencia osztályaiból, így az automata állapotainak száma minimális. □

1.8. Feladat. A algoritmus használatával minimalizáljuk a korábbi automatánkat, amely a 01-re végződő szavakat tartalmazó nyelvet fogadja el.



1.8. Megoldás. Először két állapothalmazunk lesz: $A_1 = \{S, \overset{0}{\downarrow}A, 00, 10, 11, \overset{1}{\downarrow}B\}$, $A_2 = \{01\}$. A_1 minden eleme esetén 0 hatására A_1-ben maradunk, míg 1 hatására $S, 11$ és B esetén A_1-be, $A, 00$ és 10 esetén A_2-be jutunk. A_1 állapotot felosztjuk $A_1 = \{S, 11, B\}$, $A_3 = \{A, 00, 10\}$ állapotokra. (A_2-t nem kell vizsgálni, mivel eleve csak egy állapotot tartalmaz, így nem osztható tovább.) Az így kialakult A_1, A_2 és A_3 állapotok az ábécé minden elemére egységesen viselkednek, így ezt már nem tudjuk tovább felosztani. A végeredmény az alábbi:



1.2. Algoritmus (Táblázatos módszer minimalizálásra)**.** Az alábbi módszer akkor hasznos, ha a fenti minimalizáló algoritmust implementálni akarjuk.
Tekintsük az előző feladatban megadatott automatát. A következő táblázatban az egyes elemek azt mutatják meg, hogy az illető sornak és oszlopnak megfelelő állapotokat legalább mekkora hosszúságú szavak különböztetik meg. Mivel a táblázat az főátlóra szimmetrikus és a főátló elemeinek sincs értelme, tekinksük eleve csak az alsó háromszöget:

S							
A							
B							
00							
01							
10							
11							
	S	**A**	**B**	**00**	**01**	**10**	**11**

1. Első lépésben írjunk nullát minden olyan párhoz, amelyek közül az egyik állapot elfogadó, a másik nem. Ugyanis ezeket már a nulla hosszú szavak is megkülönböztetik.

S							
A							
B							
00							
01	0	0	0	0			
10					0		
11					0		
	S	**A**	**B**	**00**	**01**	**10**	**11**

2. Ezután az i-edik lépésben vizsgáljuk meg a táblázat üres elemeit, hogy a megfelelő állapotpárból egy karakter hatására nemüres mezőbe jutunk-e az ábécé egyes elemeivel. Ha igen, írjunk i-t az adott mezőbe. Jelen esetben például a $00 - S$ pár esetén 0 hatására $00 - A$ mezőbe, míg 1 hatására a $01 - B$ mezőbe jutunk, melyek közül az utóbbi nemüres, ezért a $00 - S$ mezőbe egy 1-est kell írnunk. A teljes első menet után a táblázat állapota:

S							
A	1						
B		1					
00	1		1				
01	0	0	0	0			
10	1		1		0		
11		1		1	0	1	
	S	**A**	**B**	**00**	**01**	**10**	**11**

3. Az algoritmus véget ér, ha valamelyik menetben nem töltöttünk ki új mezőt. Jelen esetben az algoritmus az első menet után véget ér, így a fenti táblázat egyben a végeredmény is. Ezután minden üresen maradt mező egy-egy ekvivalens állapotpárt jelképez.

A táblázatos módszer segítségével az algoritmus lépésszáma is megbecsülhető:

$$O(|Q|^2|\Sigma||Q|) = O(|Q|^3|\Sigma|)$$

A $|Q|^2$-es tag felső becslés a táblázat méretére, míg a $|\Sigma|$ az egyes próbálkozások számát jelöli. Az $|Q|$-s tag a menetek maximális száma, hiszen bármely állapotot, legfeljebb ennyiszer vághatunk szét. Megjegyezzük, hogy optimális esetben az algoritmus lépésszáma lehet $O(|Q|^2|\Sigma|)$ is.

Mivel az algoritmus minden esetben a minimális állapotszámú automatát hozza létre, az algoritmus eredménye független a kiindulási automatától és csakis a nyelvtől függ.

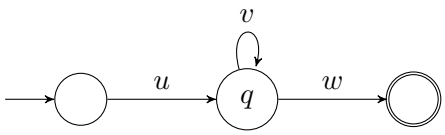
Definíció. A fenti minimalizáló algoritmussal létrehozott automata a *minimálautomata.*

1.9. Lemma (Pumpálási lemma)**.** Minden L reguláris nyelvhez létezik p pumpálási hossz, hogy $\forall x \in L$ és $|x| \geq p$ esetén létezik olyan $x = uvw$ felosztás, amire:

- $|uv| \leq p$
- $|v| \geq 1$
- $uv^k w \in L$, $\forall k \geq 0$

Ez tehát azt jelenti, hogy a szó középső része „pumpálható". Fontos azonban megjegyezni, hogy akár az is lehetséges, hogy az illető reguláris nyelv egyáltalán nem is tartalmaz n-hosszú vagy hosszabb szavakat, tehát nem minden reguláris nyelvben vannak „pumpálható" szavak.

1.9. Bizonyítás. Mivel L reguláris, biztosan létezik hozzá teljes determinisztikus véges automata. Legyen p egy ilyen automata állapotainak száma. Tekintsünk most egy x legalább p hosszú szóhoz tartozó számítást. A számítás során legalább p élen keresztül mentünk, ezért legalább $p+1$ állapotot érintettünk. Mivel az automatánknak ennél kevesebb állapota van, biztosan létezik egy olyan állapot, amit legalább kétszer érintünk. Legyen az első olyan állapot q, amit másodszor is érintünk. Legyen u az az élsorozat, amely a kiindulási állapotból q első érintéséig tart, v legyen a q első és második érintése közötti élsorozat, míg w a többi él. Az u és v együttes hossza biztosan kisebb vagy egyenlő p hiszen ellenkező esetben már korábban is lett volna ismétlődő állapot. Mivel az uvw szó a kiindulási állapotból elfogadóba visz, és v szakasz tuladonképpen egy hurok az $uv^k w$ szavak is a kiindulási állapotból elfogadóba visznek.



□

1.9. Feladat. Mutassuk meg, hogy az $L = \{0^i 1^i, i \geq 0\}$ nyelv nem reguláris! L szövegesen fogalmazva az a nyelv, amely azokat a szavakat tartalmazza, amelyekben valahány darab 0, majd ugyanennyi 1 van.

1.9. Megoldás. Indirekt úton bizonyítunk. Tegyük fel, hogy L reguláris, és p a pumpálási hossz. A $0^p 1^p$ szó nyilván benne van L-ben és a hossza nagyobb, mint p. Ennek a szónak bármilyen uvw felbontásában, ahol az uv hossza p-nél nem nagyobb és v nem üres, a v szó csakis nullákból áll. A v szót pumpálva így biztosan kikerülünk a nyelvből, vagyis az indirekt megközelítés ellentmondásra vezetett.

1.10. Feladat. Mutassuk meg, hogy az $L = \{$Ugyanannyi nullát és egyet tartalmazó szavak$\}$ nyelv nem reguláris!

1.10. Megoldás. Az $0^n 1^n$ szó szintén benne van a nyelvben így az előző bizonyítás itt is megfelelő.
Másik módszer: legyen az előző feladatban bemutatott nyelv L_1, valamint legyen

$$L_2 = \{0^i 1^j, i \geq 0, j \geq 0\},$$

vagyis az a nyelv, ahol amelynek szavaiban a 0-k megelőzik az 1-eseket. Ekkor:

$$L \bigcap L_2 = L_1$$

Könnyen megmutatható, hogy L_2 reguláris. És ha L reguláris lenne, akkor L_1-nek is regulárisnak kellene lennie, ezért L nem lehet reguláris.

1.11. Feladat. Mutassuk meg, hogy az $L =$ „Visszafelé is ugyanaz” nyelv nem reguláris.

1.11. Megoldás. Indirekt úton bizonyítunk. Tegyük fel, hogy L reguláris, és n a pumpálási hossz. Legyen x szó a következő:

$$x = \underbrace{1\ldots1}_{n}0\underbrace{1\ldots1}_{n} \in L$$

Az $|x| \geq n$ nyilván teljesül, továbbá ennek a szónak bármilyen uvw felbontásában, ahol az uv hossza n-nél nem nagyobb és v nem üres, a v szó csakis egyesekből áll. A v szót pumpálva így biztosan kikerülünk a nyelvből, vagyis az indirekt megközelítés ellentmondásra vezetett.

1.12. Feladat. Mutassuk meg, hogy az alábbi nyelv megfelelő szavai pumpálhatóak, de a nyelv nem reguláris:

$$L = \{a^i b^j c^j : i \geq 1, j \geq 0\} \cup \{b^j c^k : j, k \geq 0\}$$

1.12. Megoldás. Ha vesszük a megfelelő hosszúságú szavait, akkor azokat a lemmának megfelelő szabályok szerint fel tudjuk úgy bontani három részre, hogy a középső rész csak a-kat, vagy csak b-ket tartalmazzon. Ekkor a középső rész pumpálható lesz.
Másrészt a ab^i és ab^j szavak minden esetben megkülönböztethetőek a nyelvvel ha $i \neq j$, ezért végtelen sok olyan szó van, amely L nyelvvel megkülönböztethető, tehát L nem reguláris.

1.6. Reguláris kifejezések

Definíció (Reguláris kifejezés)**.** Ha Σ egy ábécé, akkor $\emptyset$, ε és $\forall a \in \Sigma$ reguláris kifejezés. Ha r_1 és r_2 reguláris kifejezés, akkor:

- $r_1 + r_2$
- $r_1 r_2$
- r_1^*

is reguláris kifejezések.

Nézzük meg mit jelent a reguláris kifejezések definíciója nyelvekkel leírva:

Reguláris kifejezés	**Nyelv**
$r = \emptyset$	$L = \emptyset$
$r = \varepsilon$	$L = \{\varepsilon\}$
$r = a$	$L = \{a\}$
$r = r_1 + r_2$	$L = L_1 \bigcup L_2$
$r = r_1 r_2$	$L = L_1 L_2$
$r = r_1^*$	$L = L_1^*$

1.1. táblázat. Reguláris kifejezések definíciója.

1.9. Példa. Néhány példa a reguláris kifejezések és nyelvek közötti kapcsolatra.

Reguláris kifejezés	**Nyelv**
$(a+b)^*$	a-ból és b-ből álló szavak
0^*10^*	1 darab egyest tartalmazó szavak
$(0+1)^*1(0+1)^*$	legalább egy darab egyest tartalmazó szavak
$\varepsilon + 0(0+1)^*0 + 1(0+1)^*1 + 0 + 1$	azon szavak, amelyekben az első karakter azonos az utolsóval

1.2. táblázat. Példák reguláris kifejezésekre.

Reguláris kifejezések tulajdonságai:

- $r + \emptyset = r$
- $r + \varepsilon = r \Leftrightarrow \varepsilon$-t generálja r
- $r\varepsilon = r$
- $r\emptyset = \emptyset$
- $\emptyset^* = \varepsilon$
- $(a + \varepsilon)^* = a^*$
- $(a + \varepsilon)(a + \varepsilon)^* = a^*$

1.10. Tétel. L nyelv leírható reguláris kifejezéssel $\Leftrightarrow$ L reguláris.

1.10. Bizonyítás. A két irányt külön bizonyítjuk.

$\Rightarrow$ irányban: az $\emptyset$, $\{\varepsilon\}$ és $\{a\}$ reguláris nyelvek, valamint az únió, a konkatenálás és a tranzitív lezárás nem vezet ki a reguláris nyelvek köréből, ezért L reguláris.

$\Leftarrow$ irányban: Legyen M az L-hez tartozó automata. Definiáljuk a következő nyelveket minden $p, q \in Q$-ra:

$$L(p, q) = \{x \in \Sigma^* : p\text{-ből indulva } x \text{ a } q\text{-ba visz.}\}$$

Ekkor:

$$L = \bigcup_{q \in F} L(q_0, q)$$

Így elegendő lenne az $L(p, q)$ típusú nyelvekhez reguláris kifejezést találni. Vezessük be ehhez a következő segédnyelveket:

$$\begin{aligned} L(p, q, t) = \{&x \in \Sigma^* : p\text{-ből indulva } x \text{ a } q\text{-ba visz úgy,} \\ &\text{hogy a közbenső állapotok az } \{1 \dots t\} \text{ halmazból kerülnek ki.}\} \end{aligned}$$

A definíció szerint ha $Q = \{1 \dots n\}$, akkor $L(p, q) = L(p, q, n)$, mivel így minden állapot meg van engedve közbenső állapotnak. Vegyük most szemügyre az $L(p, q, 0)$ nyelvet. Ha M-ben van p és q között él, akkor a p és q közötti éleken található betűk lesznek a nyelv elemei, valamint ha $p = q$, akkor ε is. Így tehát $L(p, q, 0)$ nyelvhez találtunk reguláris kifejezést. Tegyük most fel, hogy $L(p, q, t-1)$-re már találtunk reguláris kifejezést (legyen ez $r(p, q, t-1)$), és mutassuk meg, hogy ekkor $L(p, q, t)$-re is található.

A p állapotból most úgy kell q-ba jutnunk, hogy ehhez az $\{1 \dots t\}$ halmaz állapotait használhatjuk közbenső állapotnak. Ezt megtehetjük úgy, hogy az $\{1 \dots t-1\}$ halmaz állapotait használhatjuk közbenső állapotnak, vagy megtehetjük úgy is, hogy p-ből t-be megyünk $\{1 \dots t-1\}$ állapotok segítségével, itt tetszőlegesen sok (akár nulla) hurkot teszünk meg ugyanezen közbülső állapotokon át, majd t-ből q-ba megyünk szintén az $\{1 \dots t-1\}$ állapotok segítségével.

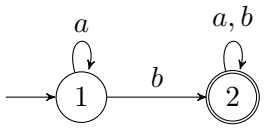
Azaz:

$$r(p, q, t) = r(p, q, t-1) + r(p, t, t-1)r(t, t, t-1)^* r(t, q, t-1)$$

Megyjegyzés: az iménti algoritmus abban az esetben is működik, ha az automata nem determinisztikus.

□

1.13. Feladat. Adjuk meg az alábbi automatával leírt nyelvhez tartozó reguláris kifejezést!



1.13. Megoldás. Nézzük először az $r(p, q, 0)$ típusú kifejezéseket:

	$\boldsymbol{r(p,1,0)}$	$\boldsymbol{r(p,2,0)}$
$\boldsymbol{p=1}$	$a+\varepsilon$	b
$\boldsymbol{p=2}$	$\emptyset$	$a+b+\varepsilon$

Használva az előző bizonyításban leírt képletet:

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

És így tovább, $t = 1$-re a reguláris kifejezések:

	$\boldsymbol{r(p,1,1)}$	$\boldsymbol{r(p,2,1)}$
$\boldsymbol{p=1}$	a^*	a^*b
$\boldsymbol{p=2}$	$\emptyset$	$a+b+\varepsilon$

$t = 2$-re a reguláris kifejezések:

	$\boldsymbol{r(p,1,2)}$	$\boldsymbol{r(p,2,2)}$
$\boldsymbol{p=1}$	a^*	$a^*b(a+b)^*$
$\boldsymbol{p=2}$	$\emptyset$	$(a+b)^*$

Az L-hez tartozó reguláris kifejezés $r(1,2,2)$, azaz: $a^*b(a+b)^*$.

2. fejezet

Nyelvtanok

2.1. Nyelvtanok

Definíció (Nyelvtan)**.** $G = (V, \Sigma, S, P)$, ahol:

- V a változók halmaza, egy véges nem üres halmaz
- Σ az ábécé, egy véges nem üres halmaz
- $S \in V$ a kezdő szimbólum
- P a levezetési (produkciós) szabályok halmaza, egy véges nem üres halmaz

A levezetési szabályokat a következőképpen adhatjuk meg:

$$\begin{aligned} P &: \alpha \to \beta \\ \alpha &: (V \cup \Sigma)^* V (V \cup \Sigma)^* \\ \beta &: (V \cup \Sigma)^* \end{aligned}$$

A levezetési szabály bal oldalán tehát mindenképpen áll legalább egy változó, ami mellett lehetnek további változók és az ábécé elemei, míg a jobb oldalán tetszőlegesen állhatnak változók és az ábécé elemei.

Definíció (Levezetés)**.** Egy olyan $S \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \cdots \Rightarrow \gamma_n$, ahol:

$$\begin{aligned} \gamma_i &= \delta_1 \alpha \delta_2 \\ \gamma_{i+1} &= \delta_1 \beta \delta_2 \\ (\alpha \to \beta) &\in P \\ \delta_1, \delta_2 &\in (V \cup \Sigma)^* \end{aligned}$$

Tehát egy levezetés során nulla vagy több levezetési szabályt alkalmazunk egymás után.

Definíció (Generált nyelv)**.** Ha G egy nyelvtan, akkor:

$$L(G) = \{w \in \Sigma^* : \exists S \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \cdots \Rightarrow w\}$$

Tehát a generált nyelv mindazon szavakból áll, amelyek S-ből levezethetőek.

2.1. Feladat. Határozzuk meg $L(G)$-t, ha $G = (\{S\}, \{a, b\}, S, P)$, ahol:

$$P = \{(S \to aS), (S \to bS), (S \to \varepsilon)\}$$

Rövidebben ezt így is írhatjuk:

$$S \to aS|bS|\varepsilon$$

2.1. Megoldás. Mivel a szabályok ismételgetésével bármely szót kirakhatjuk: $L(G) = \{a, b\}^*$

2.2. Feladat. Határozzuk meg $L(G)$-t, ha $G = (\{S\}, \{a, b\}, S, P)$, $P = \{S \to aSb|\varepsilon\}$!

2.2. Megoldás. A generált nyelv minden elemében csak az összes a után lehet b továbbá az a-k és b-k száma azonos, ezért: $L(G) = \{a^n b^n : n \geq 0\}$

A nyelvtan megadását tovább rövidíthetjük úgy, ha csak a levezetési szabályokat adjuk meg, ezáltal:

- V a levezetési szabályokban található összes nagybetű
- Σ a levezetési szabályokban található összes kisbetű
- S az elsőként felírt szabály bal oldala

2.1. Példa. Tekintsük az alábbi szabályokkal megadott nyelvtant:

$$S \to \underbrace{aSBC}_{1} \,|\, \underbrace{abC}_{2}$$
$$CB \to BC \tag{3}$$
$$bB \to bb \tag{4}$$
$$bC \to bc \tag{5}$$
$$cC \to cc \tag{6}$$

Figyeljük meg a levezetési szabályok alkalmazásának az alábbi sorrendjét (az aláhúzott rész jelöli a következőként alkalmazott szabály bal oldalát):

$$S \overset{1}{\Rightarrow} a\underline{S}BC \overset{1}{\Rightarrow} aa\underline{S}BCBC \overset{2}{\Rightarrow} aaa\underline{bC}BCBC \overset{5}{\Rightarrow} aaaabcB\underline{CB}C \overset{3}{\Rightarrow} aaabcBBCC$$

Látható, hogy a szabályok alkalmazása közben „elakadtunk”, vagyis a szabályokat a levezetés során nem lehet minden esetben tetszőlegesen alkalmazni. Könnyen megmutatható, hogy minden $\{a^i b^i c^i : i \geq 1\}$ típusú szó része a G által leírt nyelvnek:

$$S \underbrace{\overset{1}{\Rightarrow} \gamma_1 \overset{1}{\Rightarrow} \ldots \overset{1}{\Rightarrow} \gamma_2}_{n} \overset{2}{\Rightarrow} a^{n+1}b\underbrace{CBCB\ldots CB}_{n} \overset{3}{\Rightarrow} \gamma_k \overset{3}{\Rightarrow} \gamma_{k+1} \overset{3}{\Rightarrow} \ldots \overset{3}{\Rightarrow} a^{n+1}bB^nC^{n+1} \overset{4}{\Rightarrow} \gamma_m \overset{4}{\Rightarrow}$$
$$\gamma_{m+1} \overset{4}{\Rightarrow} \ldots \overset{4}{\Rightarrow} a^{n+1}b^{n+1}C^{n+1} \overset{5}{\Rightarrow} a^{n+1}b^{n+1}cC^n \overset{6}{\Rightarrow} \gamma_x \overset{6}{\Rightarrow} \gamma_{x+1} \overset{6}{\Rightarrow} \ldots \overset{6}{\Rightarrow} a^{n+1}b^{n+1}c^{n+1}$$

Belátható az is, hogy valójában az ilyen típusú szavakkal a teljes generált nyelvet lefedtük.

2.3. Feladat. Adjuk meg a palindromok, azaz a visszafelé olvasva is ugyanazt adó szavak nyelvtanát! ($\Sigma = \{a, b\}$)

2.3. Megoldás. Egy lehetséges megoldás: $S \to aSa|bSb|a|b|\varepsilon$
Triviálisan igaz, hogy az így generált nyelvben nem lehet olyan szó, amely nem palindrom. Azt is könnyű látni, hogy minden 0 és 1 hosszú palindrom része a generált nyelvnek. Mivel minden ennél hosszabb palindrom olyan, hogy az első és utolsó karaktere azonos, a kettő között pedig egy palindrom áll, a generált nyelv minden palindromot tartalmaz.

2.4. Feladat. $\Sigma = \{a, b\}$, $L = \{$ Az a-k száma azonos a b-k számával $\}$. Adjunk olyan G nyelvtant, amire $L(G) = L$.

2.4. Megoldás. Egy lehetséges megoldás: $S \to aSbS|bSaS|\varepsilon$
Minden esetben, amikor valamelyik szabályt alkalmazzuk, ugyanannyi a-t rakunk a szóba, mint ahány b-t, ezért biztosan nem generálunk olyan szót, amely nem tartozik a nyelvbe.

Bármely olyan $w \in L$ esetén ahol $|w| < 3$ igaz, hogy G generálja, mert minden ilyen szót megkaphatunk ha a megfelelő S-eket ε-nal helyettesítjük. A három vagy több karakterből álló szavak pedig a következők alakúak lehetnek: $aw'b$ vagy $bw'a$ vagy $aw'bbw''a$ vagy $bw'aaw''b$, ahol w' és w'' is L-bel szó. Ha az $aSbS$ illetve $bSaS$ szabályokban az utolsó S-t ε-nal helyettesítjük, akkor megkapjuk az első két típusba tartozó szavakat, ha pedig a nyelv egy w'' szavával, akkor a második két típusba tartozókat.

Nyelvtanok osztályozása: A 2.1 táblázat bemutatja a nyelvtanok Chomsky-féle osztályozását.

Osztály	Elnevezés	Megkötések
3. osztály	Reguláris	Csak $A \to aB$ és $A \to a$ alakú szabályok lehetnek, illetve lehet $S \to \varepsilon$ is, ha S a kezdőszimbólum és S semelyik szabály jobb oldalán nem szerepel
2. osztály	Környezet független (CF)	Csak $A \to \alpha$ alakú szabályok lehetnek, ahol $\alpha \in (V \bigcup \Sigma)^*$, $\alpha \neq \varepsilon$, illetve lehet $S \to \varepsilon$ is, ha S a kezdőszimbólum és S semelyik szabály jobb oldalán nem szerepel
1. osztály	Környezet függő (CS)	Csak $\beta A\gamma \to \beta\alpha\gamma$ alakú szabályok lehetnek, ahol $\alpha, \beta, \gamma \in (V \bigcup \Sigma)^*$, $\alpha \neq \varepsilon$, illetve lehet $S \to \varepsilon$ is, ha S a kezdőszimbólum és S semelyik szabály jobb oldalán nem szerepel
0. osztály	-	-

2.1. táblázat. Nyelvtanok osztályozása.

Nyelvek osztályozása: Az L nyelv i-edik osztályú, ha $\exists$ hozzá i-edik osztályú nyelvtan. Az i-edik osztályú nyelvek halmazát $\mathcal{L}_i$-vel jelöljük.
A definíció következménye: $\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$.

2.5. Feladat. Adjunk 3. osztályú nyelvtant az $\{a, b\}^*$ nyelvre!

2.5. Megoldás. Egy lehetséges megoldás:

$$S \to \varepsilon|aA|bA|a|b$$
$$A \to aA|bA|a|b$$

2.2. Reguláris nyelvtanok

2.1. Tétel. L nyelvhez létezik 3. osztályú nyelvtan $\Leftrightarrow$ L reguláris.

2.1. Bizonyítás. A két irányt külön bizonyítjuk:

- $\Leftarrow$ irányban: Legyen $M = (Q, \Sigma, \delta, q_0, F)$ az L nyelvet elfogadó automata Az automata alapján hozzunk létre egy olyan $G = (V, \Sigma, S, P\}$ nyelvtant, hogy $L(G) = L(M) = L$ legyen. Ehhez legyen $|V| = |Q|$, a q állapothoz tartozó változót jelölje A_q, $S = q_0$, valamint az automata minden $\delta(q, a) = p$ átmenetéhez vegyünk fel egy $A_q \to aA_p$ szabályt P-be. Ezen kívül vegyünk még fel minden $\delta(q, a) = p$, $p \in F$ átmenethez egy $A_q \to a$ szabályt. Mivel reguláris nyelv levezetése esetén mindig csak egy nem behelyettesített változónk lehet, az aktuális behelyettesítetlen változónk megmutatja, hogy az automata szerint, éppen melyik állapotban lennénk. Már csak azt az esetet nem kezeltük, ha $q_0 \in F$ vagyis ha $\varepsilon \in L$.

Ehhez vegyük fel az $S \to \varepsilon$ szabályt. Ez a szabály megsértheti a 3. osztályú nyelvtanok kritériumát, ha valamely másik szabályban szerepel az S jobb oldalon, azaz ha q_0-ba megy átmenet M-ben. Ennek a kezeléséhez még a konstrukció alkalmazása előtt vegyünk fel M-be egy extra q_0' állapotot. Minden q_0-ban végződő átmenetet töröljünk és vegyünk fel helyette egy ugyanolyan, de q_0'-ben végződőt, valamint minden q_0-ból induló átmenethez vegyünk fel egy ugyanolyan, de q_0'-ből indulót. Az így keletkezett módosított automatában már nincs q_0-ban végetérő átmenet így a fenti konstrukció alkalmazható rá, és az elfogadott nyelvet sem módosítottuk.

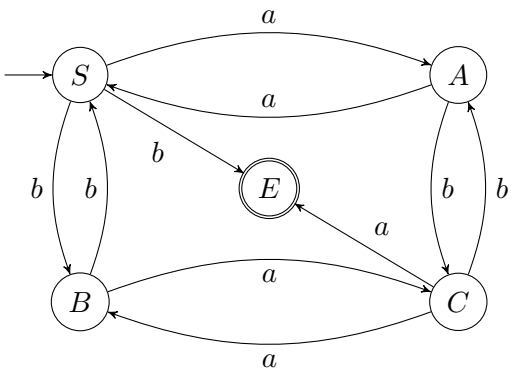
$\Rightarrow$ irányban: Legyen $G = (V, \Sigma, S, P\}$ az L nyelvet generáló nyelvtan. A nyelvtan alapján hozzunk létre egy M automatát L-hez. Legyen $Q = V \bigcup \{E\}$, ahol E egy extra elfogadó állapot, továbbá legyen $q_0 = S$. Minden $A \to aB$ alakú szabályhoz vegyünk fel egy A-ból B-be menő a karakterrel kiváltott átmentet, és minden $A \to a$ alakúhoz egy A-ból E-be menő a karakterrel kiváltottat. Ha van $S \to \varepsilon$ szabály P-ben, akkor S állapot legyen elfogadó. Könnyen belátható, hogy $L(G) = L(M)$. Megjegyzés: az így konstruált M automata nem feltétlenül determinisztikus.

□

2.6. Feladat. Az iménti bizonyításban használt konstrukció segítségével készítsünk automatát az alábbi nyelvtannal megadott nyelvhez:

$$S \to aA|bB|b$$
$$A \to aS|bC$$
$$B \to aC|bS$$
$$C \to aB|bA|a$$

2.6. Megoldás. A megoldás az alábbi:



A korábban definiált reguláris nyelvtanokat szokták *jobb-reguláris nyelvtan*oknak is hívni, mivel az $A \to aB$ alakú kifejezésekben a jobb oldalon van a változó. Ennek mintájára definiálhatunk bal-reguláris nyelvtanokat is.

Definíció (Bal-reguláris nyelvtan)**.** Olyan nyelvtan, amely esetén P-ben csak $A \to Ba$ és $A \to a$ alakú szabályok lehetnek, illetve lehet $S \to \varepsilon$ is, ha S a kezdőszimbólum és S semelyik szabály jobb oldalán nem szerepel.

Belátható, hogy a bal-reguláris nyelvtanok ugyanúgy a reguláris nyelveket generálják. A következő módon hozhatunk létre automatát egy bal-reguláris nyelvtanból:

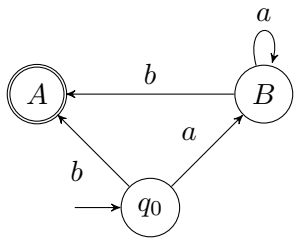
- $Q = V \cup q_0$ (tehát q_0 egy új állapot)

- Minden $A \to Ba$ alakú szabályhoz vegyünk fel egy B-ből A-ba mutató a-val kiváltott átmenetet
- Minden $A \to a$ alakú szabályhoz vegyünk fel egy q_0-ból A-ba mutató a-val kiváltott átmenetet
- $F = \{S\}$

2.7. Feladat. Az alábbi bal-reguláris nyelvből hozzunk létre automatát, majd az automata alapján konstruáljunk jobb-reguláris nyelvtant:

$$A \to Bb|b$$
$$B \to Ba|a$$

2.7. Megoldás. Az automata:



Ez alapján jobb-reguláris nyelvtan:

$$S \to aB|bA|b$$
$$B \to aB|bA|b$$

Vegyük észre, hogy semelyik levezetés során sem lehet az A változót kiküszöbölni, ezért az A-t tartalmazó szabályokat elhagyhatjuk:

$$S \to aB|b$$
$$B \to aB|b$$

Összegzés: Ha tehát L egy reguláris nyelv, akkor megadhatjuk nyelvtannal, reguláris kifejezéssel vagy véges automatával. Ha az a feladatunk, hogy eldöntsük hogy L üres-e, akkor véges automata esetén a kezdőállapotból indulva valamilyen bejárással ellenőrizhetjük, hogy eljuthatunk-e elfogadó állapotba, ha pedig reguláris kifejezéssel vagy nyelvtannal van megadva, akkor átalakíthatjuk ezeket automatává.
Ha két reguláris nyelv azonosságát kell megállapítanunk, akkor a hozzájuk tartozó automatákat minimalizálhatjuk, majd ha az így kapott automaták izomorfak, akkor a két nyelv is azonos. Vegyük észre, hogy ez egy nagyon speciális izomorfia vizsgálat, mivel a kezdő állapotokat megfeleltethetjük egymásnak és az állapotátmenetek segítenek a további állapotok azonosításában.
Ha a kérdés az, hogy az L nyelv véges-e, akkor a pumpálási lemma felhasználásával válaszolhatunk. Ha L minimálautomatájának n állapota van és találunk egy legalább ekkora szót, akkor biztosan van pumpálható szó is. A következő tétel egy reguláris nyelv végtelenségére ad szükséges és elégséges feltételt.
Megjegyzés: minden véges nyelv reguláris, tehát ha egy nyelv nem reguláris, akkor mindenképpen végtelen sok eleme van.

2.2. Tétel. Legyen L reguláris nyelv és n a hozzá tartozó minimálautomata állapotainak száma. L végtelen $\Leftrightarrow$ ha létezik $w \in L$ szó, hogy $n \leq |w| \leq 2n$.

2.2. Bizonyítás. A két irányt külön bizonyítjuk:

- $\Leftarrow$ irányban: w pumpálható $\Rightarrow |L| = \infty$
- $\Rightarrow$ irányban: $\exists w' \in L : |w'| \geq n$. Ha $|w'| \leq 2n$, akkor megtaláltuk a keresett szót, ha nem, akkor viszont pumpáljuk w'-t úgy, hogy a pumpálható részt 0-szor vesszük. Mivel a pumpálható rész legfeljebb n hosszú, ezért a pumpálás során legfeljebb n-nel csökkenhet w' hossza. Ha ezután egy $2n$-nél hosszabb szót kaptunk, akkor tovább csökkenthetjük a hosszát további pumpálással. Minden lépésben csak n-nel csökkenhet a hossza, ezért nem fogjuk átugrani az n-től $2n$-ig tartó sávot.

□

3. fejezet

CF nyelvek és veremautomaták

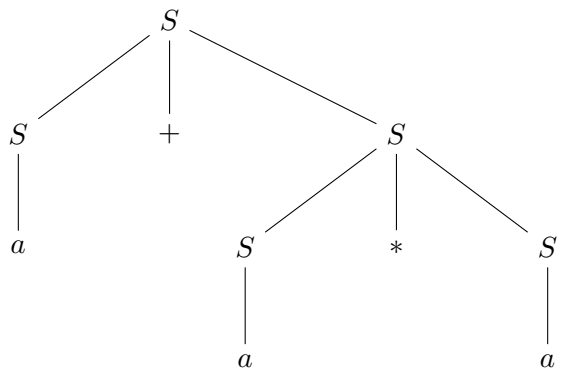
Definíció. Környezetfüggetlen (Context Free) nyelveknek azokat a nyelveket nevezzük, amelyekhez létezik 2. oszályú nyelvtan. Az ilyen nyelvtanok szabályai környezetfüggetlen változóhelyettesítéseket tartalmaznak.

3.1. Példa. Az alábbi CF nyelv az egyszerű aritmetikai kifejezéseket írja le:

$$S \to \underbrace{\underline{S} + \underline{S}}_{1} \mid \underbrace{\underline{S} * \underline{S}}_{2} \mid \underbrace{(S)}_{3} \mid \underbrace{a}_{4}$$

Egy lehetséges levezetés:
$S \overset{1}{\Rightarrow} \underline{S} + S \overset{4}{\Rightarrow} a + \underline{S} \overset{2}{\Rightarrow} a + \underline{S} * S \overset{4}{\Rightarrow} a + a * \underline{S} \overset{4}{\Rightarrow} a + a * a$
A levezetéshez levezetési fát is megadhatunk a következőképpen:

- Gyökér elem: A kezdő szimbólum
- Belső csúcsok: Változók
- Belső csúcs gyerekei: A változó helyettesítése
- Levelek: Σ elemei illetve lehet ε is.



A levezetési fából könnyen kiolvashatjuk a levezetett szót, ehhez csupán balról jobbra haladva kell a fa leveleit végigjárnunk. Az is könnyen látható, hogy bármely levezetés egyértelműen megadja a hozzá tartozó levezetési fát. Figyeljük meg, hogy ez visszafele nem igaz, vagyis egy levezetési fához többféle levezetés is tartozhat. Az alábbi levezetés eltér a korábbitól, mégis ugyanazt a fát adja:
$S \overset{1}{\Rightarrow} S + \underline{S} \overset{2}{\Rightarrow} S + \underline{S} * S \overset{4}{\Rightarrow} S + a * \underline{S} \overset{4}{\Rightarrow} \underline{S} + a * a \overset{4}{\Rightarrow} a + a * a$

Definíció (Bal levezetés)**.** Egy CF nyelvtan olyan levezetése, amely során minden esetben a legelső változót helyettesítjük be.

Definíció (Jobb levezetés)**.** Egy CF nyelvtan olyan levezetése, amely során minden esetben a legutolsó változót helyettesítjük be.

Figyeljük meg, hogy reguláris nyelvtanok esetén a levezetési fa egy olyan bináris fa lesz, amelynek minden levele a szülő csúcs bal oldalán van. Reguláris nyelvek esetén a bal levezetés megegyezik a jobb levezetéssel.

3.1. Veremautomaták

Definíció. A *veremautomata* egy olyan automata, amely a bemenet folyamatos olvasása mellett egy vermet (LIFO tulajdonságú tárolót) is használ. Minden lépésben olvashat a bemenetről nulla vagy egy karaktert és emellett kiveheti a veremből a tetején lévő szimbólumot, valamint írhat a verem tetejére tetszőleges (véges) számú szimbólumot. Fontos, hogy csak a verem tetején lévő szimbólumot olvashatja.
Formálisan: $M = (Q, \Sigma, \Gamma, q_0, Z_0, F, \delta)$, ahol:

- Q az állapotok halmaza, egy véges, nem üres halmaz.
- Σ a bemenet ábécéje, egy véges, nem üres halmaz.
- Γ a verem ábécéje vagyis a lehetséges veremszimbólumok halmaza, egy véges, nem üres halmaz.
- q_0 a kezdőállapot ($q_0 \in Q$)
- Z_0 a kezdetben a verem tetején lévő szimbólum ($Z_0 \in \Gamma$)
- F az elfogadó állapotok halmaza ($F \subseteq Q$)
- δ az állapotátmeneti függvény:

  $$(q, a, A) \to \{(q', \alpha) : q' \in Q, \alpha \in \Gamma^*\}$$

  Értelmezése: az automata $q \in Q$ állapotban van, a verem tetejéről az $A \in \Gamma \cup \{\varepsilon\}$ szimbólumot olvassa. A függvény értéke (q', α) párokból álló halmaz, ahol q' az új állapot, α a verem tetejére írandó szimbólumsorozat. A kiolvasott szimbólumot az olvasáskor az automata egyúttal el is távolítja a verem tetejéről. Ha $A = \varepsilon$, akkor az automata nem olvas (és távolít el) a verem tetejéről.

Számítás: Veremautomaták esetén is definiálhatjuk a számítás fogalmát. Ha az automata bemenete w:

$$\begin{aligned} &w = w_1 w_2 \dots w_m, \\ &w \in \Sigma^*, \\ &w_i \in \Sigma \bigcup \{\varepsilon\} \end{aligned}$$

akkor $r_0 = q_0, r_1, r_2 \dots r_m$, $r_i \in Q$ egy számítás, és az aktuális veremállapotok

$$\begin{aligned} &s_0 = \{Z_0\}, s_1 \dots s_m, \\ &s_i \in \Gamma^*, s_i = A t_i, \\ &A \in \Gamma \bigcup \{\varepsilon\}, t_i \in \Gamma^* \end{aligned}$$

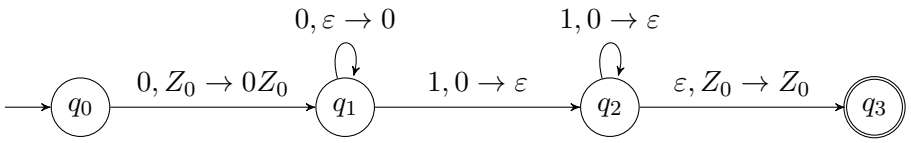
ha $(r_{i+1}, t_{i+1}) \in \delta(r_i, w_{i+1}, A)$, $s_{i+1} = t_{i+1} t_i$. Ekkor M elfogadja a w szót, ha van olyan számítása, amelyre $r_w \in F$.

$$L(M) = \{w \in \Sigma^* : M \text{ elfogadja } w\text{-t}\}$$

3.2. Példa. A következő automata az $L = \{0^n 1^n : n \geq 1\}$ nyelvet fogadja el:

$$(q_0, 0, Z_0) \to (q_1, 0Z_0)$$
$$(q_1, 0, \varepsilon) \to (q_1, 0)$$
$$(q_1, 1, 0) \to (q_2, \varepsilon)$$
$$(q_2, 1, 0) \to (q_2, \varepsilon)$$
$$(q_2, \varepsilon, Z_0) \to (q_3, Z_0)$$
$$F = \{q_3\}$$

A veremautomatákat is lehet gráffal ábrázolni:



Definíció (Determinisztikus veremautomata)**.** Olyan veremautomata, melyre:

- $\forall q \in Q, a \in \Sigma \bigcup \varepsilon, A \in \Gamma \bigcup \{\varepsilon\}$ esetén $|\delta(q, a, A)| \leq 1$
- Ha $\delta(q, \varepsilon, \varepsilon) \neq \emptyset$, akkor $\forall (a, A) \neq (\varepsilon, \varepsilon)$ esetén $\delta(q, a, A) = \emptyset$
- Ha $\delta(q, \varepsilon, A) \neq \emptyset$, akkor $\forall a \in \Sigma$ esetén $\delta(q, a, A) = \emptyset$
- Ha $\delta(q, a, \varepsilon) \neq \emptyset$, akkor $\forall A \in \Gamma$ esetén $\delta(q, a, A) = \emptyset$

3.1. Tétel. Van olyan L nyelv, amelyhez létezik M veremautomata, hogy $L = L(M)$, de nem létezik ilyen determinisztikus veremautomata.

3.1. Bizonyítás. A tételt nem bizonyítjuk, de megjegyezzük, hogy a palindromok nyelve egy ilyen nyelv. □

Definíció (Üres veremmel elfogadó veremautomata)**.** Olyan $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$ veremautomata, ahol az elfogadó állapotok halmazát (F) nem adjuk meg. Az automata akkor fogad el egy szót, ha a végigolvasása után a verem üres lesz. (Még Z_0 sincs benne).

3.2. Tétel. L nyelvhez $\exists$ veremautomata $\Leftrightarrow$ L-hez $\exists$ üres veremmel elfogadó veremautomata.

3.2. Bizonyítás. A két irányt külön bizonyítjuk.

$\Rightarrow$ irányban: legyen az L-et elfogadó veremautomata $M = (Q, \Sigma, \Gamma, q_0, Z_0, F, \delta)$. Hozzunk létre ebből egy L-et üres veremmel elfogadó $M' = (Q', \Sigma, \Gamma', q_0', Z_0', \delta')$ automatát:

- $Q' = Q \bigcup \{q_0', q_\varepsilon\}$ tehát q_0' és q_ε új állapotok, és q_0' az új kezdőállapot.
- $\Gamma' = \Gamma \bigcup \{Z\}$ tehát Z egy új veremszimbólum
- $Z_0' = Z_0$
- δ' legyen δ kiterjesztése az alábbi új állapotátmenetekkel:
  - $\delta'(q_0', \varepsilon, Z_0) = (q_0, Z_0 Z)$
  - $\delta'(q, \varepsilon, A) = (q_\varepsilon, \varepsilon)$, minden $q \in F$ és minden $A \in \Gamma'$ esetén
  - $\delta'(q_\varepsilon, \varepsilon, A) = (q_\varepsilon, \varepsilon)$, minden $A \in \Gamma'$-re

  Vagyis q_ε a vermet kiürítő állapot.

A konstrukció helyes, mert minden $w \in L(M)$ esetén M-nek van olyan számítása, amivel F-beli állapotba kerülünk, onnan pedig át tudunk menni a kiürítő állapotba. Ezen kívül az extra Z szimbólum csak a kiürítő állapotban tűnhet el, ahova csak F-beli állapotokból juthatunk.

$\Leftarrow$ irányban: legyen az L-et üres veremmel elfogadó veremautomata $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$. Hozzunk létre ebből egy L-et elfogadó $M' = (Q', \Sigma, \Gamma', q_0', Z_0', F, \delta')$ veremautomatát:

- $Q' = Q \bigcup \{q_0', q_F\}$ tehát q_0' és q_F új állapotok, és q_0' az új kezdőállapot
- $\Gamma' = \Gamma \bigcup \{Z_0'\}$ tehát Z_0' egy új veremszimbólum.
- $F = \{q_F\}$
- δ' legyen δ kiterjesztése az alábbi új állapotátmenetekkel:
  - $\delta'(q_0', \varepsilon, Z_0') = (q_0, Z_0 Z_0')$
  - $\delta'(q, \varepsilon, Z_0') = (q_F, Z_0')$ minden $q \in Q$ esetén

A kostrukció helyessége az előző irányú bizonyításhoz hasonlóan könnyen belátható.

□

3.3. Tétel. L-hez létezik M veremautomata, amelyre $L(M) = L \Leftrightarrow L$-hez $\exists$ 2. osztályú nyelvtan.

3.3. Bizonyítás. A két irányt külön bizonyítjuk.

$\Leftarrow$ irányban a 3.4 konstrukció adja a megoldást.

$\Rightarrow$ irányban a 3.6 konstrukció adja a megoldást.

□

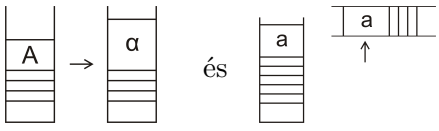
3.4. Konstrukció. Legyen $G = (V, \Sigma, S, P)$ az a 2. osztályú nyelvtan, amelyhez az $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$ üres veremmel elfogadó veremautomatát akarjuk létrehozni. Ha már létrehoztunk egy üres veremmel elfogadó automatát, akkor a korábban tárgyalt módszerrel könnyen átalakíthatjuk hagyományos veremautomatává.

- $Q = \{q\}$, azaz csak egy állapotunk lesz.
- $\Gamma = V \bigcup \Sigma \bigcup \{Z_0, Z_1\}$, tehát Z_0 és Z_1 új veremszimbólumok.
- $q_0 = q$

Az állapotátmenetek legyen a következők:

- $\delta(q, \varepsilon, Z_0) = (q, SZ_1)$
- Minden $A \in V$ esetén: $\delta(q, \varepsilon, A) = \{(q, \alpha) : A \to \alpha \in P\}$
- Minden $a \in \Sigma$ esetén: $\delta(q, a, a) = (q, \varepsilon)$
- $\delta(q, \varepsilon, Z_1) = (q, \varepsilon)$

Tehát minden válózó esetén beírjuk a helyettesítését a verembe, ha pedig az olvasott karakter azonos a verem tetejével, mindkettőt eldobjuk.



Alternatív lehetőségként a kezdőszimbólumnak választhatjuk közvetlenül S-t és akkor nincs szükség az első és az utolsó szabályra.

3.5. Állítás. A 3.4 konstrukció által előállított veremautomata pontosan az L nyelv szavait fogadja el.

3.5. Bizonyítás. Először megmutatjuk, hogy ha $x \in L(G)$, akkor $x \in L(M)$. Legyen $x \in L(G)$, tekintsük egy bal levezetését, amelyben az n. lépésben egy $A \to \gamma$ szabályt használtunk:

$$yA\alpha \Rightarrow y\gamma\alpha \Rightarrow \ldots \Rightarrow x$$

Legyen $y\gamma\alpha = yz\beta$, ahol β az α első változójával kezdődik, legyen továbbá $x = yzw$. Mivel ez egy bal levezetés, y nem tartalmaz változókat, és mivel β első eleme az α első változója ezért z sem tartalmaz változókat. Mivel y és z nem tartalmaz változókat közvetlenül meg fog jelenni x-ben. Teljes indukcióval megmutatható, hogy M-re mindig igaz, hogy x-szel elindítva amikor a bemenet hátralévő része w, akkor a verem tartalma βZ_1. Ez alapján a levezetés utolsó lépése esetén $\beta = \varepsilon$ (mivel ekkor már nincsenek változók) és ezért a verem tartalma ekkor Z_1.
A bizonyítás másik irányban hasonlóképpen teljes indukcióval működik. □

3.6. Konstrukció. Legyen $M = (Q, \Sigma, \Gamma, q_0, Z_0, \delta)$ üres veremmel elfogadó veremautomata, amelyhez $G = (V, \Sigma, S, P)$ CF nyelvtant akarjuk létrehozni.
Először alakítsuk át M-et úgy, hogy megszüntetjük azokat az átmeneteket, amelyekben nem veszünk le a verem tetejéről. Ezeket helyettesítsük azzal, hogy a verem tetején lévő elemet levesszük és utána visszatesszük. Világos, hogy az automata által meghatározott nyelv nem változik.
Formálisan: $(q', \alpha) \in \delta(q, a, \varepsilon)$ helyett: $(q', \alpha A) \in \delta(q, a, A), \forall A \in \Gamma$

Ez után írjuk fel a változókat: $V = \{[qAp] : q, p \in Q, A \in \Gamma\} \cup \{S\}$. Minden változót (az S kivételével) ezzel a hármassal jelölünk. Itt q jelöli azt az állapotot, amelyben A a verem tetejére kerül, illetve p azt az állapotot, amelyben A és az összes felette lévő kikerül a veremből.
Szabályok legyenek a következők:

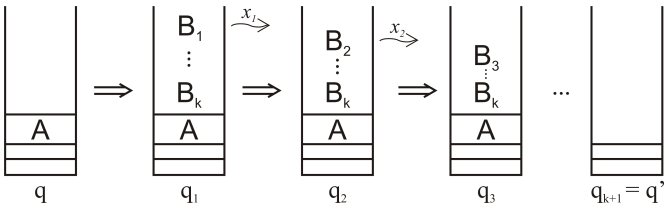
- $S \to [q_0 Z_0 q], \forall q \in Q$
- A $(q', \varepsilon) \in \delta(q, a, A)$ típusú átmenetekhez: $[qAq'] \to a$ ahol $q, q' \in Q, A \in \Gamma, a \in \Sigma \cup \{\varepsilon\}$
- A $(q', B_1 \ldots B_k) \in \delta(q, a, A)$ típusú átmenetekhez: $[qAq''] \to a[q_1 B_1 q_2][q_2 B_2 q_3] \ldots [q_k B_k q_{k+1}]$, ahol $q_1 = q', \forall q_{k+1} = q'', \forall q_2, q_3, \ldots, q_k \in Q$-ra.

3.7. Állítás. A 3.6 konstrukció által előállított nyelvtan az M veremautomata által elfogadott nyelvet generálja.

3.7. Bizonyítás. Ehhez először megmutatjuk, hogy $[qAq']$ változóból pontosan akkor tudjuk $x \in \Sigma^*$-ot levezetni, ha M automatában x bemenettel és A veremtartalommal a q' állapotba jutunk, úgy hogy a verem üressé válik és végigolvastuk x-et.
Ha ez igaz lenne, akkor az $S \to [qZ_0q]$ szabályok miatt S-ből pontosan akkor lehetne x-et levezetni amikor M elfogadja x-et.
A fenti állítás bizonyítása az egyik irányban: teljes indukcióval, x beolvasott szó(részlet) hossza szerint.

- $|x| = 1$ esetén: a konstrukció következménye.

- $|x| > 1$ esetén: $[qAq'] \Rightarrow a[q_1B_1q_2][q_2B_2q_3]\dots[q_kB_kq_{k+1}]$, $q' = q_{k+1}$, ekkor B_1-ből levezethető a bemenet x_1 része, $\dots B_k \Rightarrow x_k$.



A bizonyítás másik irányban hasonlóképpen, teljes indukcióval lehetséges. □

3.3. Példa. Az alábbi CF nyelvtanból konstruáljunk veremautomatát!

$$A \to \underbrace{a}_{1} \mid \underbrace{aA}_{2} \mid \underbrace{bAA}_{3} \mid \underbrace{AAb}_{4} \mid \underbrace{AbA}_{5}$$

δ elemei a következők:

$$\begin{aligned}
(q, \varepsilon, Z_0) &\to (q, AZ_1)\\
(q, \varepsilon, A) &\to \{(q, a), (q, aA), (q, bAA), (q, AAb), (q, AbA)\}\\
(q, a, a) &\to (q, \varepsilon)\\
(q, b, b) &\to (q, \varepsilon)\\
(q, \varepsilon, Z_1) &\to (q, \varepsilon)
\end{aligned}$$

Figyeljük meg a fenti nyelvtan esetén az *abbaaa* szó alábbi levezetését:

$$A \overset{5}{\Rightarrow} \underline{A}bA \overset{1}{\Rightarrow} abA \overset{3}{\Rightarrow} abb\underline{A}A \overset{2}{\Rightarrow} abba\underline{A}A \overset{1}{\Rightarrow} abbaaA \overset{1}{\Rightarrow} abbaaa$$

Nézzük hogyan juthatunk üres vermes állapotba az automatában ugyanerre a bemenetre:

$$\begin{aligned}
(q, abbaaa, Z_0) \Rightarrow (q, abbaaa, AZ_1) \overset{5}{\Rightarrow} (q, abbaaa, AbAZ_1) \overset{1}{\Rightarrow} (q, abbaaa, abAZ_1) \Rightarrow\\
(q, bbaaa, bAZ_1) \Rightarrow (q, baaa, AZ_1) \overset{3}{\Rightarrow} (q, baaa, bAAZ_1) \Rightarrow (q, aaa, AAZ_1) \overset{2}{\Rightarrow}\\
(q, aaa, aAAZ_1) \Rightarrow (q, aa, AAZ_1) \overset{1}{\Rightarrow} (q, aa, aAZ_1) \Rightarrow (q, a, AZ_1) \overset{1}{\Rightarrow} (q, a, aZ_1) \Rightarrow (q, \varepsilon, Z_1)
\end{aligned}$$

3.2. CF nyelvek átalakítása

3.8. Tétel. Ha G nyelvtan szabályai $A \to \alpha$ alakúak és $A \to \varepsilon$ is megengedett, akkor $\exists G'$ 2. osztályú nyelvtan, hogy: $L(G) = L(G')$.

3.1. Algoritmus. *Elenyésző* változóknak nevezzük azokat a változókat, amelyekből ε levezethető. A következő módon vezetjük be az elenyésző változók halmazát:

$$\begin{aligned}
N_0 &= \{A \in V : (A \to \varepsilon) \in P\}\\
&\vdots\\
N_{i+1} &= \{A \in V : \exists A \to \alpha : \alpha \in N_i^*\}
\end{aligned}$$

Világos, hogy $N_0 \subseteq N_1 \subseteq \dots \subseteq V$. Ilyenkor $\exists i$, amelyre az algoritmus leáll, azaz $N_i = N_{i+1}$.
Ekkor ha $A \to \alpha$ szabály G-ben, akkor G'-ben is, és mellé új szabályokat veszünk föl: α elenyésző változóit az összes lehetséges módon ε-nal helyettesítjük. Végül pedig elhagyjuk G'-ből az $A \to \varepsilon$ és $A \to A$ formátumú szabályokat.
Megjegyzés: Amennyiben $\varepsilon \in L(G)$ teljesül az eredeti nyelvre, akkor G'-ben fölveszünk egy új kezdőszimbólumot: S' és két új szabályt: $S' \to \varepsilon|S$, a többi szabályban S változatlan.

3.9. Állítás. Ha G-ben levezethető $x \in \Sigma^*$, akkor G'-ben is ($x \neq \varepsilon$). Vagyis: G-ben: $A \overbrace{\Rightarrow \cdots \Rightarrow}^{\text{n lépés}} x$, akkor G'-ben: $A \overbrace{\Rightarrow \cdots \Rightarrow}^{\text{m lépés}} x$.

3.9. Bizonyítás. Teljes indukcióval, a G-beli levezetés lépéseinek száma szerint.

- $n = 1$ lépés esetén: $G : A \to x$, $G' : A \to x$ mindig teljesül.
- $n > 1$ lépés esetén: $A \to \alpha = X_1 \dots X_k$, ahol $X_i = x_i \in \Sigma$ vagy $X_i \Rightarrow \cdots \Rightarrow x_i \in \Sigma^*$.
  Ekkor G'-ben:
  - Ha $x_i \neq \varepsilon$, akkor rendben.
  - Ha $x_i = \varepsilon$, akkor X_i kihagyva: $A \to X_1 \dots X_{i-1} X_{i+1} \dots X_k$.

Másik irány hasonlóan bizonyítható. □

3.1. Feladat. Legyen G nyelvtan az alábbi:

$$\begin{aligned} S &\to ABCD \\ A &\to CD|AC \\ B &\to Cb \\ C &\to a|\varepsilon \\ D &\to bD|\varepsilon \end{aligned}$$

Alakítsuk szabályos 2. osztálybeli nyelvtanná!

3.1. Megoldás. Először írjuk fel az elenyésző változók halmazát!

$$\begin{aligned} N_0 &= \{C, D\} \\ N_1 &= \{C, D, A\} \\ N_2 &= \{C, D, A\} \end{aligned}$$

Mivel $N_1 = N_2$, az algoritmus megállt. Továbbá S nem elenyésző változó, ezért ε nincs benne a nyelvben.
A meglévő szabályokat egészítsük ki újakkal, ahol az elenyésző változókat helyettesítjük ε-nal minden lehetséges módon. Ekkor G':

$$\begin{aligned} S &\to ABCD|BCD|BD|BC|B|ABD|ABC|AB \\ A &\to CD|C|D|AC \\ B &\to Cb|b \\ C &\to a \\ D &\to bD|b \end{aligned}$$

Definíció. *Egyszeres szabály*: $A \to B$ és $B \in V$

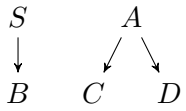
3.2. Algoritmus. Adott G (CF nyelvtan), cél az egyszeres szabályok kiküszöbölése. Ehhez készítsünk egy irányított gráfot, melyben minden nyelvtanbeli változóhoz vegyünk fel egy csúcsot. Minden $A \to B$ egyszeres nyelvtani szabályhoz vegyünk fel egy A-ból B-be mutató irányított élt. Jelöljük $U(A)$-val azokat a csúcsokat, amelyek a gráfban A-ból irányított úton (tetszőleges) bejárással elérhetők.
Ha $B \in U(A)$ és $B \to \beta \in P$, akkor $A \to \beta \in P'$ is legyen szabály G'-ben, viszont az egyszeres szabályokat kihagyjuk. Ekkor $L(G) = L(G')$ teljesül.

3.2. Feladat. Vegyük az előző feladat megoldását.

$$\begin{aligned}
&S \to ABCD|BCD|BD|BC|B|ABD|ABC|AB\\
&A \to CD|C|D|AC\\
&B \to Cb|b\\
&C \to a\\
&D \to bD|b
\end{aligned}$$

Távolítsuk el belőle az egyszeres szabályokat!

3.2. Megoldás. Rajzoljuk fel a gráfot:



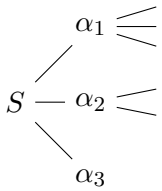
$$\begin{aligned}
&S \to ABCD|BCD|BD|BC|\overbrace{Cb|b}^{B\text{ helyett}}|ABD|ABC|AB\\
&A \to CD|\overbrace{a}^{C\text{ helyett}}|\overbrace{bD|b}^{D\text{ helyett}}|AC\\
&B \to Cb|b\\
&C \to a\\
&D \to bD|b
\end{aligned}$$

Probléma: Adott G nyelvtan és $w \in \Sigma^*$ szó. Kérdés, hogy a nyelvtan generálja-e a szót, azaz $w \overset{?}{\in} L(G)$.

Megoldás: Ha G nyelvtan *reguláris*, akkor elkészítjük hozzá a megfelelő véges determinisztikus automatát, innen pedig már könnyen ellenőrizhető, hogy a nyelv generálja-e a szót Ha G *környezet független* nyelvtan, akkor ahhoz tudunk verem automatát készíteni, viszont determinisztikus automatát már nem mindig, így a sokféle számítás miatt nem egyszerű az ellenőrzés. Emiatt célszerű a nyelvtannal tovább dolgozni. Kétféle módszerrel dolgozhatunk:

- összes lehetőség kipróbálása *elágazás és korlátozás* módszerrel
- *dinamikus programozás*

Nézzük meg az elágazás és korlátozás módszert! A dinamikus programozással a 3.3 algoritmus foglalkozik.
Tegyük fel, hogy a kezdő S változónkat $\alpha_1, \alpha_2 \ldots \alpha_n$ szavakkal helyettesíthetjük. Vizsgáljuk most a szó elejét. Ha például $\alpha_1 = a\beta$, $\alpha_3 = b\gamma$ és a szó az a karakterrel kezdődik akkor a levezetés α_3 ágát nem folytatjuk.



A továbbiakban feltesszük, hogy nincsenek egyszeres szabályok. Ekkor kétféle szabály lehet:

$$A \to \alpha, |\alpha| \geq 2$$
$$A \to a$$

Mivel minden lépésben nő a változók vagy a karakterek ($\in \Sigma$) száma, ezért *nem csökkenő nyelvtan*nak hívjuk. Fontos megjegyezni, hogy w levezetésében a lépések száma legfeljebb $2|w|$ lehet, vagyis az eljárás véges. Hatékonynak azonban nem mondható, mert az elágazás miatt exponenciális lépésszámú.

Definíció. Egy CF nyelvtan *Chomsky-féle normálformájú (CNF)*, ha a szabályai

$$A \to a$$
$$A \to BC$$

alakúak.

3.10. Tétel. Minden G CF nyelvtanhoz lehet készíteni G' nyelvtant, hogy G' CNF és $L(G') = L(G) - \{\varepsilon\}$.

3.10. Bizonyítás. Konstrukcióval bizonyítjuk.

1. G-ben megszüntetjük az egyszeres szabályokat valamint ha van benne $S \to \varepsilon$, akkor azt töröljük. Ez után a lehetséges szabályok:

$$A \to a$$
$$A \to \alpha, |\alpha| \geq 2$$

2. Minden $a \in \Sigma$ karakterhez, ami valamelyik szabály legalább 2 hosszúságú jobb oldalán fordul elő, fölveszünk egy új változót: X_a. Fölveszünk továbbá egy új szabályt: $X_a \to a$, illetve X_a-val helyettesítjük a-t, ahol nem egyedüli karakter egy szabály jobb oldalán.

3. Az előző lépés után ha egy szabály jobb oldala legalább kettő hosszú, akkor csakis változókból állhat. Jelöljük ezt így: $A \to B_1 B_2 \dots B_k$ ($B_i \in V, k \geq 3$). Ezek helyett vezessük be az alábbi szabályokat:

$$\begin{aligned} A &\to B_1 C_1 \\ C_1 &\to B_2 C_2 \\ &\vdots \\ C_{k-2} &\to B_{k-1} B_k \end{aligned}$$

   $C_1, \dots, C_{k-2}$ új változók.

□

3.4. Példa. Alakítsuk át a következő CF nyelvtant CNF formára: $S \to aSa|bSb|c$!

1.

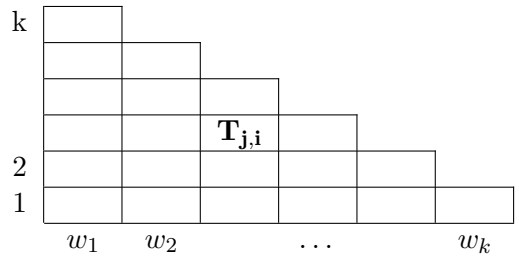
$$\begin{aligned} S &\to c|X_a S X_a|X_b S X_b \\ X_a &\to a \\ X_b &\to b \end{aligned}$$

2.

$$\begin{aligned} S &\to c|X_aA|X_bB \\ X_a &\to a \\ X_b &\to b \\ A &\to SX_a \\ B &\to SX_b \end{aligned}$$

3.3. Algoritmus (Cocke-Younger-Kasimi, CYK)**.** Az algoritmus egy CNF formára alakított G nyelvtan esetén keresi a választ arra a kérdésre, hogy a w szó benne van-e a G által generált nyelvben. Hozzunk létre egy táblázatot.

- A táblázat oszlopai a szó betűi: $w_1, w_2, \ldots, w_k$
- A táblázat sorai: $j = 1, 2, \ldots, k$
- $T_{j,i}$-edik cellába azok a változók kerülnek, melyekből $w_iw_{i+1}\ldots w_{i+j-1}$ szórészlet levezethető



Kitöltés:

- $j = 1$ esetén: A változó bekerül a $T_{1,i}$-edik cellába, ha $A \to w_i$ szabály G-ben
- $j > 1$ esetén: A változó bekerül a $T_{j,i}$-edik cellába, ha $A \to BC$ szabály G-ben, $B \in T_{l,i}$ és $C \in T_{j-l,i+l}$, valamilyen $1 \leq l \leq j-1$-re.

Megoldás: Ha a $T_{k,1}$-es cellában (bal felső) szerepel a kezdőszimbólum, akkor $w \in L(G)$.
Lépésszám:

- cellánként: $\approx k$
- táblázat: k^2
- összesen: $O(k^3)$

3.5. Példa.

$$\begin{aligned} S &\to AB|BC \\ A &\to BA|a \\ B &\to CC|b \\ C &\to AB|a \\ w &= baaba \end{aligned}$$

A táblázat alsó sorának kitöltése egyértelmű. Vegyük a $T_{2,1}$-es cellát. A cellához egy cellapár tartozik: $T_{1,1}$ és $T_{1,2}$. Keressük azokat a szabályokat, amik ezen cellapárosok változóit tartalmazzák jobb oldalon, vagyis $\to BA$, illetve $\to BC$ alakúak. Ilyenek az $A \to BA$ és $S \to BC$ szabályok, vagyis a cella tartalma A, S lesz. A $T_{3,1}$-es cellához két cellapár tartozik: $T_{2,1}$–$T_{1,3}$ és $T_{1,1}$–$T_{2,2}$. A többi cella ugyanígy kitölthető.

S, A, C				
–	S, A, C			
–	B	B		
A, S	B	C, S	A, S	
B	A, C	A, C	B	A, C
b	a	a	b	a

A táblázatból kitűnik, hogy w szó a megadott nyelvtanból levezethető, mert a bal felső cellában szerepel a kezdőszimbólum.

3.3. CF nyelvek tulajdonságai

Definíció. Egy $w \in \Sigma^*$ szó *egyértelműen levezethető* egy nyelvtanból, ha csak egy levezetési fája van. G nyelvtan egyértelmű, ha $\forall w \in L(G)$ egyértelmű. Nyelv egyértelmű, ha létezik egyértelmű nyelvtana.

3.6. Példa. A feltételes utasítás nyelvtana az alábbi formában nem egyértelmű:

```
if (felt) utasítás

if (felt) utasítás1 else utasítás2
```

Például: `if (felt1) if (felt2) utasítás1 else utasítás2`

3.7. Példa. Az

$$E \to E + E | E * E | (E) | a \tag{1}$$

nyelvtan nem egyértelmű (korábban már láttuk), de a nyelv egyértelmű, mert létezik egyértelmű nyelvtana:

$$\begin{aligned} E &\to E + T | T \\ T &\to T * F | F \\ F &\to (E) | a \end{aligned} \tag{2}$$

3.11. Állítás. Az előző példában bemutatott (2) nyelvtan egyértelmű és ugyanazt a nyelvet generálja, mint (1).

3.11. Bizonyítás. Vázlatosan bizonyítjuk, a szóhossz (n) szerint teljes indukcióval.

- $n = 1$ esetén: $w = a$ egyértelmű levezetése: $E \Rightarrow T \Rightarrow F \Rightarrow a$
- $n > 1$ esetén:
  1. Ha van a zárójelen kívüli $+$, akkor a levezetésnek így kell kezdődnie: $E \Rightarrow E \underset{\substack{\uparrow \\ \text{utolsó}}}{+} T$. Az itt lévő $+$ a levezetésben az utolsó. A folytatás az indukció miatt egyértelmű.
  2. Ha nincs a zárójelen kívüli $+$, de van zárójelen kívüli $*$, akkor a levezetésnek így kell kezdődnie: $E \Rightarrow E \underset{\substack{\uparrow \\ \text{utolsó}}}{*} T$. Az itt lévő $*$ a levezetésben az utolsó. A folytatás az indukció miatt egyértelmű.
  3. Ha se $+$, se $*$ nincs a zárójelen kívül, akkor a levezetése egyértelmű: $E \Rightarrow T \Rightarrow F \Rightarrow (E)$. A folytatás az indukció miatt egyértelmű.

□

3.12. Tétel. Létezik nem egyértelmű nyelv. (Lásd a 3.19 állítást)

3.8. Példa. Bizonyítás helyett figyeljük meg a következő nyelvet:

$$\{a^i b^j c^k : i = j \text{ vagy } j = k\}$$

3.13. Állítás. Ha L_1, L_2 CF nyelvek, akkor $L_1 \cup L_2$ is CF

3.13. Bizonyítás. Jelöljük a két nyelvtant:

$$\begin{aligned} G_1 &= (V_1, \Sigma, S_1, P_1) & L(G_1) &= L_1 \\ G_2 &= (V_2, \Sigma, S_2, P_2) & L(G_2) &= L_2 \\ G &= (V, \Sigma, S, P) & L(G) &= L = L_1 \cup L_2 \end{aligned}$$

Feltehetjük, hogy a két nyelvtan változói diszjunktak. ($V_1 \cap V_2 = \emptyset$). Ekkor:

$$\begin{aligned} P &= P_1 \cup P_2 \cup \{S \to S_1 | S_2\} \\ V &= V_1 \cup V_2 \cup \{S\} \end{aligned}$$

Végül pedig kiküszöböljük az ε-szabályokat. □

3.14. Állítás. Ha L_1, L_2 CF nyelvek, akkor $L_1 L_2$ is CF

3.14. Bizonyítás. Jelöljük ugyanúgy a nyelvtanokat, mint az előző bizonyításban! Feltehetjük, hogy a két nyelvtan változói diszjunktak. ($V_1 \cap V_2 = \emptyset$). Ekkor:

$$\begin{aligned} P &= P_1 \cup P_2 \cup \{S \to S_1 S_2\} \\ V &= V_1 \cup V_2 \cup \{S\} \end{aligned}$$

Végül pedig kiküszöböljük az ε-szabályokat. □

3.15. Állítás. Ha L_1 CF nyelv, akkor L_1^* is CF.

3.15. Bizonyítás. Továbbra is ugyanúgy jelöljük G_1-et és G-t.Ekkor:

$$P = P_1 \cup \{S \to S_1 S | \varepsilon\}$$

Végül pedig kiküszöböljük az ε-szabályokat. □

3.16. Állítás. Metszetekre, komplementerekre a CF nyelvek nem zártak.

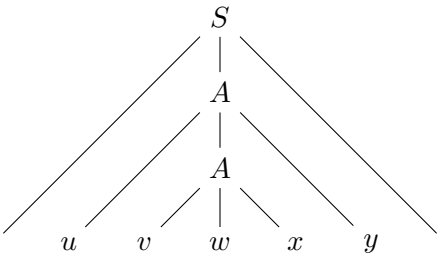
3.16. Bizonyítás. Lásd a 3.10 példát. □

3.17. Lemma (Pumpálási lemma CF nyelvekre)**.** Minden L CF nyelvhez létezik p pumpálási hossz, hogy $\forall z \in L$ és $|z| \geq p$ esetén létezik olyan $z = uvwxy$ felosztás, amire:

- $|vwx| \leq p$
- $|vx| \geq 1$

és $uv^k wx^k y \in L$, $\forall k \geq 0$.

3.17. Bizonyítás. Tekintsünk egy olyan L környezetfüggetlen nyelvtan, amelyben nincsenek egyszeres szabályok. (Ezt megtehetjük, mivel már láttuk, hogyan lehet az egyszeres szabályokat kiküszöbölni.) Legyen k a nyelvtanban használt változók száma és legyen d a nyelvtani szabályok jobb oldalain álló kifejezések közül a leghosszabb hossza. Legyen továbbá $p = d^{k+1}$. Megmutatjuk, hogy a legalább p hosszúságú szavak pumpálhatóak. Tekintsünk egy $z \in L$ teszőleges legalább p hosszú szót és vegyük szemügyre ennek egy levezetési fáját. Ebben a fában minden csúcsnak legfeljebb d fia lehet, valamint a levelek száma nyilván $|z|$. Emiatt viszont biztosan van út a gyökér és valamelyik levél között, amely legalább $k+2$ csúcsot tartalmaz. Ezen az úton legalább $k+1$ változó van, ezért biztosan van benne ismétlődő változó. Legyen A az a változó, amelyikből az iménti úton a levétől a gyökér felé haladva a leghamarabb találunk ismétlődő pédányt. Osszuk fel z szót a levezetési fa alapján az alábbi módon:



Ekkor A két előfordulása közötti részt akárhányszor megismételve továbbra is L nyelv szabályai alapján érvényes levezetési fát kapunk, így $uw^m wx^m y \in L$. Mivel nincs a nyelvben egyszeres szabály, v és x együttesen nem lehet üres. (Az A két előfordulása közötti részben biztosan generáltunk legalább egy levelet.) Az is elmondható, hogy $|vwx| \leq p$, mert a legmélyebben lévő ismétlődést választottuk, így a felső A alatti levelek száma legfeljebb $d^{k+1} = p$. □

3.9. Példa. A lemma segítségével bizonyítsuk be, hogy az $L = \{a^n b^n c^n : n \geq 0\}$ nyelv nem CF! Indirekt tegyük fel, hogy L CF és alkalmazzuk rá a pumplálási lemmát, a pumpálási hosszt jelöljük p-vel. Válasszuk a $z = \overbrace{a \ldots a}^{p}\overbrace{b \ldots b}^{p}\overbrace{c \ldots c}^{p}$ szót. Mivel $|vwx| \leq p$, ezért vwx legfeljebb kétféle betűből állhat.
Ez alapján persze v és x együttesen legfeljebb kétféle betűt tartalmaz, így ha ezeket többször vesszük, akkor a kapott szóban biztosan nem lesz mindhárom fajta betűből ugyanannyi. Itt ellentmondásra jutottunk L definíciójával, tehát L nem CF nyelv.

3.10. Példa. Megmutatjuk, hogy a CF nyelvek halmaza nem zárt a metszet műveletre. Ehhez tekintsük az alábbi nyelveket:

$$L_1 = \{a^n b^n c^m : n, m \geq 1\}$$
$$L_2 = \{a^m b^n c^n : n, m \geq 1\}$$
$$L = \{a^n b^n c^n : n \geq 1\}$$

Korábban már láttuk, hogy L nyelv nem CF-nyelv, valamint könnyen készíthető veremautomata L_1-hez és L_2-höz. Lévén, hogy $L = L_1 \bigcap L_2$, sikerült a metszet művelet segítségével kilépnünk a CF-nyelvek halmazából.
A CF nyelvek zártak az únióra de nem zártak a metszetre, ebből természetesen az is következik, hogy a komplementer képzésre sem lehetnek zártak, hiszen az alábbi módon megkaphatjuk a metszet műveletet úniók és kompelenterek segítségvel.

$$\overline{\overline{L_1} \bigcup \overline{L_2}} = L_1 \bigcap L_2$$

3.3. Feladat. Mutassuk meg, hogy az alábbi L nyelv nem környezetfüggetlen!

$$L = \{ww | w \in \{a, b\}^*\}$$

3.3. Megoldás. Indirek módon tegyük fel, hogy L egy CF nyelv és p a hozzá tartozó pumpálási hossz. Tekintsük a $z = a^p b^p a^p b^p$ szót, amely nyilván benne van a nyelvben és a hossza is megfelelő ezért pumpáható. Mivel $|vwx| \leq p$ ezért legfeljebb két egymást követő p hosszú blokkba lóghat bele. Ha v-t és x-et pumpáljuk, akkor az előzőek miatt vagy egyetlen blokkot tudunk növelni, vagy két egymás mellettit. Könnyen látható, hogy így mindenképpen kikerülünk a nyelvből.

3.18. Lemma (Ogden-Lemma)**.** Minden L környezetfüggetlen nyelvhez, $\exists p > 0$, hogy minden legalább p hosszú $z \in L$ szó esetén, ha tetszőlegesen kijelölünk legalább p darab pozíciót z-ben, akkor létezik z-nek egy olyan $z = uvwxy$ felbontása, hogy:

- vwx legfeljebb p darab kijelölt pozíciót tartalmaz
- v és x együttesen legalább 1 kijelölt pozíciót tartalmaz.
- minden $m \geq 0$ esetén $uv^m wx^m y \in L$

3.18. Bizonyítás. A lemma bizonyítását az olvasóra bízzuk. □

3.11. Példa. Az Ogden-Lemma segítségével mutassuk meg, hogy az alább L nyelv nem környezetfüggetlen!

$$L = \{a^i b^i c^j | i \neq j\}$$

Indirekt módon tegyük fel, hogy a nyelv környezetfüggetlen és legyen az Ogden-lemma szerinti pumpálási hossz p. Tekintsük az $a^p b^p c^{p+p!}$ szót, amely nyilván eleme a nyelvnek és megfelelő hosszúságú. Jelöljük ki az első p pozíciót, azaz a szó elején található a karaktereket. A lemma alapján bizonyos, hogy v és x együttesen tartalmaz a karaktert. Könnyen látható, hogy ha v vagy x többféle karatert is tartalmazna, akkor a pumpálásával kikerülnénk a nyelvből, ezért feltételezhetjük, hogy v és x is csupa azonos karakterből áll, és az előzőek alapján így valamelyikük csupa a karakter. Ekkor viszont az is igaz, hogy a másikuk csupa b karakter, mivel csak így tudjuk az a-kat és a b-ket egyenlő arányban növelni. Legyen v és x egyaránt k hosszúságú, ahol $1 \leq k \leq p$. A szó m-szeres pumpálása esetén a kapott új szó az alábbi alakú:

$$a^{p+(m-1)k} b^{p+(m-1)k} c^{p+p!}$$

A faktoriális definíciója miatt $m = \frac{p!}{k} + 1$ egész szám, és m ezen megválasztása esetén mindhárom karakterből azonos számú lesz, tehát kikerülünk a nyelvből.

3.19. Állítás. Az $L = \{a^i b^j c^k | i = j \text{ vagy } j = k\}$ nyelv nem egyértelmű, azaz nincs olyan CF nyelvtana, amelyben minden szó levezetési fája egyértelmű.

3.19. Bizonyítás. Vázlat. Mivel ez egy környezetfüggetlen nyelv, igaz rá az Ogden-Lemma. Legyen p a lemma által adott pumpálási hossz. Tekintsük a $z = a^p b^p c^{p+p!}$ szót és jeljük ki benne az első p pozíciót. Az előző feladat megoldásában leírtak miatt z felbontásáról tudjuk, hogy $v = a^k$ és $x = b^k$. Jelen esetben tudjuk, hogy ilyenkor x és v valóban pumpálható lesz. Ez egyúttal azt is jelenti, hogy létezik olyan A változó a nyelvtani szabályok változói között, hogy A-ból levezethető az $a^k A b^k$ formula. Tekintsük most a $s = a^{p+p!} b^{p+p!} c^{p+p!}$ szót. Világos, hogy $s \in L$, továbbá az s szót le tudjuk vezetni úgy, hogy a z szó levezetése közben az iménti $A \Rightarrow \cdots \Rightarrow a^k A b^k$ levezetést megfelelően sokszor alkalmazzuk. Hasonlóképpen elmondhatjuk, hogy a $q = a^{p+p!} b^p c^p$ szó levezetése közben is találhatunk egy olyan B változót amelyből levezethető egy $b^t B c^t$ formula, így tehát s-t le tudjuk vezetni úgy is, hogy q levezetése közben az iménti szabályt alkalmazzuk megfelelően sokszor. Láttuk tehát, hogy s szóra minden esetben legalább 2 levezetés található, amelyek levezetési fája is eltérő. □

Definíció. Egy nyelv *determinisztikus CF-nyelv (DCF)*, ha létezik hozzá determinisztikus veremautomata.

3.12. Példa. Az alábbi két nyelv DCF:

$$L_1 = \{a^i b^i c^j | i, j \geq 1\}$$
$$L_2 = \{a^j b^i c^i | i, j \geq 1\}$$

Korábban már láttuk, hogy a két nyelv metszete nem környezetfüggetlen, így természetesen nem is DCF. A DCF nyelvek tehát nem zártak a metszetre. Bizonyítás nélkül megjegyezzük, hogy zártak a komplementerkézésre, de nem zártak az únióra.

Tegyük fel, hogy az a feladatunk, hogy egy CF nyelvtannal megadott L nyelvről megállapítsuk, hogy üres-e. Egyrészt ha p a pumpálási hossz és nem találunk p-nél rövidebb elemet L-ben, akkor biztosan tudhatjuk, hogy üres. Másrészt a következő módszer szükséges és elégséges feltételt ad egy nyelv ürességére. Vezessük be a $V_0, V_1 \ldots V_k$ halmazokat a következőképpen:

$$V_0 = \{A : \exists A \to \alpha, \alpha \in \Sigma^*\}$$
$$V_{i+1} = \{A : \exists A \to \alpha, \alpha \in (\Sigma \bigcup V_i)^*\} \bigcup V_i$$

Nyilvánvaló, hogy a V_i halmazokat egy bizonyos határig tudjuk csak bővíteni:

$$V_0 \subseteq V_1 \subseteq \ldots V_k = V_{k+1}$$

Ekkor:

$$L = \emptyset \Leftrightarrow S \notin V_k$$

4. fejezet

Turing-gépek

4.1. Alapfogalmak

Definíció. A *Turing-gépeket* a következőképpen írhatjuk le:

$$T = (Q, \Sigma, \Gamma, q_0, \_, F, \delta)$$

ahol:

- Q egy véges, nem üres halmaz, az állapotok halmaza
- Σ egy véges, nem üres halmaz, a bemeneti ábécé
- Γ egy véges, nem üres halmaz, az szalagábécé
- $q_0 \in Q$ a kezdőállapot
- $\_ \in (\Gamma - \Sigma)$ a szalagon az üres jel
- $F \subseteq Q$ az elfogadó állapotok halmaza
- δ az átmeneti függvény

A gép rendelkezik egy belső állapottal, továbbá egy szalagon olvassa be a bemenetet, amit módosítani is tud, valamint tud a szalagon jobbra vagy balra lépni, esetleg helyben maradni. Az átmeneti függvény elemei a következő alakúak:

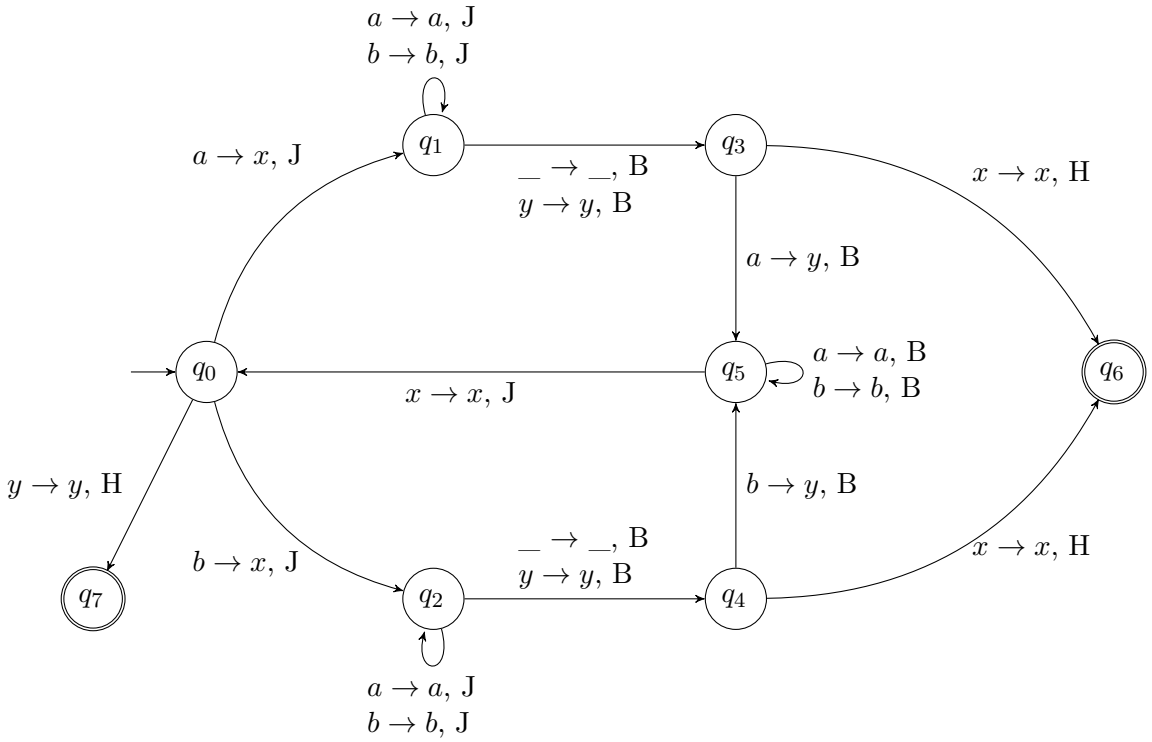
$$\begin{aligned} \delta : (q, a) &\to (q', b, D) \\ q, q' &\in Q \\ a, b &\in \Gamma \\ D &\in \{B, J, H\} \text{ azaz } \mathbf{b}\text{alra, } \mathbf{j}\text{obbra vagy } \mathbf{h}\text{elyben} \end{aligned}$$

Ez azt jelenti, hogy ha a gép q állapotban van, a betűt olvas a szalagról akkor q'-be kerül, emellett a-t felülírja b-vel majd a szalagot olvasó és író fej D irányba lép.
A gép kezdetben a q_0 állapotban van, a szalag elején a bemeneti szó található, a szalag többi része _ szimbólumokkal van feltöltve és a fej a szalag első karakterén áll.
A működés akkor ér véget, ha már nem tud lépni a gép, és akkor fogadja el a bemeneti szót, ha F-beli állapotban akadt el a működés.
A Turing-gép által elfogadott nyelv:

$$L(T) = \{w : T \text{ elfogadja } w\text{-t}\}$$

Fontos megjegyezni, hogy semmi nem garantálja, hogy adott bemenet hatására a Turing-gép valaha is le fog állni, tehát létre lehet hozni végtelen ciklusokat.

4.1. Példa. A 4.1 ábrán látható Turing-gép a $\Sigma = \{a, b\}$ feletti palindromokat fogadja el.



4.1. ábra. Palindromok nyelvét elfogadó Turing-gép.

Definíció. Az L nyelv *rekurzívan felsorolható*, ha van olyan T Turing-gép, hogy $L(T) = L$. Azt is szoktuk mondani, hogy az L nyelv ekkor *felismerhető*. A rekurzívan felsorolható nyelvek halmazát RE-vel jelöljük.

Definíció. Az L nyelv *rekurzív*, ha van olyan T Turing-gép, hogy $L(T) = L$ és T minden bemenet esetén véges lépésben megáll. Azt is szoktuk mondani, hogy az L nyelv ekkor *eldönthető*. A rekurzív nyelvek halmazát R-rel jelöljük. $R \subseteq RE$

4.1. Állítás. Van olyan L nyelv, amelyre: $L \notin RE$.

4.1. Bizonyítás. Egy Turing-gép leírás véges hosszú, ezért a Turing-gépek által felismerhető nyelvek száma megszámlálhatóan végtelen. Az összes nyelv számossága azonban kontinuum végtelen, ezért nyilván van olyan nyelv, amely nem ismerhető fel Turing-géppel. □

4.2. Speciális Turing-gépek

4.2. Példa. Mielőtt átnéznénk a speciális Turing-gépeket, nézzünk egy másfajta példát. Adott egy speciális veremautomata, ami visszafelé is léphet a bemeneten. Létezik-e olyan nyelv, amit a gép elfogad, de a normál veremautomata nem, vagyis nem CF nyelvet ír le?
Válasz igen, íme egy példa: $a^n b^n c^n$ nyelv nem CF, de ezzel az automatával el lehet fogadni.
Megjegyzés: ha véges automata léphetne visszafelé is a bemeneten, az ugyanúgy véges automata marad, legfeljebb kevesebb állapottal.

Definíció. A *k-szalagos Turing-gép* egy olyan Turing-gép változat, amely k darab szalagot használ ($k \geq 2$). Az első szalag elején helyezkedik el a bemenet és ha már úgy is több szalagunk van, akkor legyen csak olvasható. A többi szalagot ún. munkaszalagnak hívjuk, ezeket írni és

olvasni is lehet. Az átmeneti függvény elemeiben meg kell adnunk, hogy melyik szalagról mit olvastunk, melyik munkaszalagra mit írunk, és melyik szalagon merre lépünk.

$$\delta(q, a_1a_2 \dots a_k) \to (q', b_2b_3 \dots b_k, d_1d_2 \dots d_k)$$
$$a_i, b_i \in \Gamma$$
$$q, q' \in Q$$
$$d_i \in B, J, H$$

4.2. Tétel. Adott M egy k-szalagos ($k \geq 2$) Turing-gép. Ekkor létezik olyan normál (egy szalagos) Turing-gép, M', hogy $L(M) = L(M')$.

4.2. Bizonyítás. Ha M k-szalagos, akkor M' egyetlen szalagja $2k$ sávból álljon. Ezt a szalag abc megnövelésével tudjuk elérni.
Minden páros sávban a hozzá tartozó szalagot, a páratlanban a szalaghoz tartozó kurzort tároljuk. Formálisan:

- Az i-edik szalag M-ben = $(2i)$-edik sáv M'-ben
- M-ben az i-edik szalaghoz tartozó fejet az $(2i+1)$-edik sávban az adott helyen 1-es karakter jelöli M'-ben. (Minden más helyen üres karakter van)
- Γ szalag abc M-ben, akkor $\Gamma' = (\Gamma \times \{1, \_\})^k$

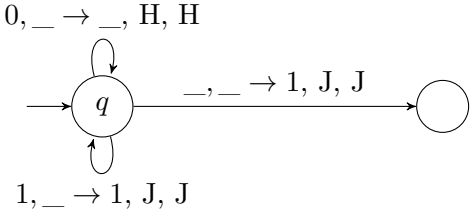
Szimuláljuk M egy lépését M'-ben! Az állapotváltás nem változik. Induljunk a szalag elejéről és a k darab fejpozíciónak megfelelő karaktereket kiolvassuk és állapotban tároljuk. Visszafelé mozogva változtatjuk M átmeneti függvénye szerint a karaktereket és a fejpozíciókat jelző egyeseket. □

4.3. Példa. A palindromok nyelvének felismerését gyorsabban elvégezhetjük egy 2-szalagos Turing-géppel úgy, hogy a bemenetet lemásoljuk a második szalagra, majd a két szalagot az ellentétes végéről indulva egyszerre olvassuk.

Definíció. Számoló Turing-gép. Ennek a Turing-gépnek egy csak olvasható bementi szalagja egy csak írható kimeneti szalagja és néhány munkaszalagja van, amelyeket olvasni és írni is lehet. A gép által kiszámított parciális (nem mindenhol definiált) függvény: f_M. Ha a gép bemenetén x van, akkor megálláskor a kimeneti szalagon $f_M(x)$ lesz. Ha f_M függvény nincs definiálva x helyen, akkor a gép nem áll meg.

4.1. Feladat. $\Sigma = \{0, 1\}$, $f(1^n) = 1^{n+1}$, többi helyen nincs definiálva. Például: $f(1111) = 11111$. Készítsük el az f függvényhez tartozó számoló Turing-gépet!

4.1. Megoldás. Megoldáskor ügyeljünk arra, hogy ha a függvény nem definiált (van benne 0), akkor az automata nem állhat meg. Ezt egy végtelen ciklussal valósíthatjuk meg.



Definíció. A Turing-gépekhez hasonlóan a függvényeket is osztályozhatjuk. Az f függvényt *parciálisan rekurzív*nak hívjuk, ha létezik M számoló Turing-gép, hogy $f_M = f$. A függvény *rekurzív*, ha emellett M minden bemeneten megáll. (Azaz az f minden x-re definiálva van.)

Church-Turing tézis: Ha egy nyelv valamilyen algoritmussal felismerhető, akkor Turing-géppel is felismerhető.

1. f parciális függvény rekurzív $\Leftrightarrow$ van olyan algoritmus, ami minden x bemenetre (ahol $f(x)$ értelmezve van) kiszámolja $f(x)$-et.
2. L nyelv rekurzív $\Leftrightarrow$ van olyan algoritmus, ami minden x bemenet esetén eldönti, hogy $x \overset{?}{\in} L$

Megjegyzés: az algoritmusra nincs jó definíció, például:

- biológiai számítás molekulákkal
- kvantum számítógép

A Turing-gép ezeket tudja szimulálni.

Definíció. *Univerzális Turing-gép*, ami bármely Turing-gépet tud szimulálni. Az általánosítás kedvéért szabványosítsuk a Turing-gépeinket! Belátható, hogy ha nem ilyen a gép, átalakítható erre a formára, anélkül, hogy megváltozna az elfogadott nyelv.

- 1 szalag
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, 1, \ldots, t-2, \_\}$, azaz összesen t darab szalag szimbólum van és az utolsó $\_$.
- $Q = \{0, 1, \ldots, r\}$ és q_0 a kezdőállapot
- $F = \{r\}$

Ekkor bármely M leírható a következő formátumban:

$$\underbrace{\#\#\#}_{\substack{\text{elválasztók}\\ \notin\Sigma}} \quad \underset{\substack{\uparrow\\ \text{szalagszimbólumok}\\ \text{száma}}}{t} \quad \#\# \quad \underset{\substack{\uparrow\\ \text{az utolsó állapot}\\ \text{sorszáma}}}{r} \quad \#\# \underbrace{q\#a\#q'\#b\#d}_{\substack{\delta(q,a)=(q',b,d)\\ \text{átmenet leírása}}} \#\# q'' \ldots \#\#\#$$

Ha a gép kódját w-vel jelöljük, akkor a hozzá tartozó Turing-gépet jelöljük M_w-vel. Ekkor U univerzális Turing-gép:

$$U(\underset{\substack{\uparrow\\ \text{gép}\\ \text{leírás}}}{w} \# \underset{\substack{\uparrow\\ \text{bemenet}}}{s}) = \begin{cases} M_w(s) & \text{ha } w \text{ Turing-gép kód} \\ \text{elutasít} & \text{ha } w \text{ nem Turing-gép kód} \end{cases}$$

4.3. Tétel. Létezik univerzális Turing-gép.

4.3. Bizonyítás. Konstrukcióval. U univerzális Turing-gépnek legyen három szalagja. Az első szalag a bemenet: $w\#s$ tartalommal.

1. w formális ellenőrzése. Amennyiben w nem Turing-gépet leíró kód, akkor U megáll nem elfogadó állapotban (elutasít).
2. U második szalagja M_w egyetlen szalagját szimulálja, tehát ide kell másolni M_w gép s bemenetét.
3. U harmadik szalagja tárolja M_w aktuális állapotát.

Belátható, hogy a szalagok fent leírt felhasználásával egy ilyen háromszalagos Turing-géppel megvalósítható az univerzális Turing-gép működése. Korábban láttuk, hogy bármely többszalagos Turing-gép szimulálható egyszalagossal. □

4.3. Rekurzív és Rekurzívan felsorolható nyelvek

Definíció. *Univerzális nyelv*:

$$L(U) = \{w\#s : \exists M_w \text{ Turing-gép és } M_w(s) = \text{ elfogad}\} = L_u$$

Tehát ebbe a nyelvbe mindazon $w\#s$ alakú szavak tartoznak, amelyek esetén a w egy Turing-gép leírása, és eme Turing-gép elfogadja s-t. Következmény: L_u Rekurzívan felsorolható (RE) nyelv.

Definíció. *Diagonális nyelv*:

$$L_d = \{w : \exists M_w \text{ Turing-gép és } w \notin L(M_w)\}$$

Tehát a diagonális nyelv olyan w szavakat tartalmaz, amelyek olyan Turing-gépeket írnak le, amelyek saját leírásukat nem fogadják el.

4.4. Tétel. $L_d \notin$ RE.

4.4. Bizonyítás. Indirekt, tegyük fel, hogy $L_d \in$ RE. Ekkor létezik M Turing-gép, hogy $L(M) = L_d$. Legyen M kódja w. Két lehetőség van $w \overset{?}{\in} L(M)$-re:

- Ha $w \in L(M) = L_d$, akkor a diagonális nyelv definíciója szerint $w \notin L(M_w)$, viszont $L(M_w) = L(M)$ miatt $w \notin L(M)$ lenne, ami ellentmond a feltételnek.
- Ha $w \notin L(M) = L_d$, akkor a diagonális nyelv definíciója szerint $w \in L(M_w)$, ekkor $w \in L(M)$ lenne, ami szintén ellentmondás.

Tehát L_d nyelv nem RE. □

4.5. Tétel (Turing)**.** L_u nyelv nem Rekurzív ($L_u \notin R$).

4.5. Bizonyítás. Indirekt, tehát $L_u \in$ R. Ekkor létezik M Turing-gép, hogy $L(M) = L_u$ és M minden bemenetre megáll. Definiáljunk egy M' gépet. Ha w nem Turing-gép kód, akkor M' elutasít. Egyébként, definiáljuk M'-t úgy, hogy ha $M(w\#w)$

- elfogad ($\Leftrightarrow w\#w \in L_u \Leftrightarrow w \in L(M_w)$), akkor M' elutasít.
- elutasít ($\Leftrightarrow w\#w \notin L_u \Leftrightarrow w \notin L(M_w)$), akkor M' elfogad.

Világos, hogy $L(M') = L_d$ és M' minden bemeneten megáll, vagyis $L_d \in$ R $\subseteq$ RE fönnállna, ami ellentmondás, tehát az indirekciós feltevés hibás volt. □

Definíció. *Megállási nyelv*:

$$L_h = \{w\#s : \exists M_w \text{ Turing-gép és } M_w(s) \text{ megáll}\}$$

Megjegyzés: $L_u \subseteq L_h$

4.6. Tétel. $L_h \in$ RE, de $L_h \notin$ R.

4.6. Bizonyítás. A két részt külön bizonyítjuk.

- $L_h \in$ RE. Jelölje M az L_h nyelvet elfogadó Turing-gépet. Ha M bemenete $w\#s$, akkor:
  1. Ha w nem Turing-gép kód, M elutasít.
  2. Egyébként ha $U(w\#s)$ kimenete
     (a) elfogad: M elfogad.

(b) elutasít: M elfogad.

(c) nem áll meg: M nem áll meg.

Mivel $L(M) = L_h$, ezért teljesül RE definíciója.

- $L_h \notin$ R. Indirekt, tehát $L_h \in$ R. Ekkor létezik M, hogy $L(M) = L_h$ és M megáll minden bemenetre. Legyen M', hogy $L(M') = L_u$. Ekkor M' bemenetén $w\#s$ hatására

  1. elutasít, ha w nem Turing-gép kód.
  2. ha $M(w\#s)$ kimenete
     (a) elfogad, akkor $w\#s \in L_h$ és $M_w(s)$
        i. elfogad, akkor M' is elfogad
        ii. elutasít, akkor M' is elutasít.
     (b) elutasít, akkor $w\#s \notin L_h$, M' elutasít.

  Mivel $L_u \notin$ R, ezért ellentondásra jutottunk.

□

4.4. Műveletek R és RE nyelvekkel

4.7. Állítás. Műveletek.

1. $L_1, L_2 \in \mathrm{R} \Rightarrow L_1 \cup L_2 \in \mathrm{R}$.
2. $L_1, L_2 \in \mathrm{R} \Rightarrow L_1 \cap L_2 \in \mathrm{R}$.
3. $L_1 \in \mathrm{R} \Rightarrow \overline{L_1} \in \mathrm{R}$.
4. $L_1, L_2 \in \mathrm{RE} \Rightarrow L_1 \cup L_2 \in \mathrm{RE}$.
5. $L_1, L_2 \in \mathrm{RE} \Rightarrow L_1 \cap L_2 \in \mathrm{RE}$.

4.7. Bizonyítás. Vázlat. Jelöljük L_1 és L_2-höz tartozó Turing-gépet M_1 és M_2-vel. Az 1. és a 2. esetén két Turing-gépet „*sorosan*” kötünk egymás után. Az 1.-nél M_1 *elutasít* ágára kötjük M_2 bemenetét és az elfogadó állapotok az M_1 és az M_2 elfogadó állapotai, a 2.-nál pedig az M_1 *elfogad* ágra kötjük M_2 bemenetét és az elfogadó állapotok az M_2 elfogadó állapotai.
A 3.-nál M jelölje azt a gépet, ami olyan, mint M_1, de $F = Q_1 - F_1$. Ekkor $L(M) = \overline{L}$.
A 4. esetén a két gépet „*párhuzamosan*” indítjuk és ha valamelyik megáll elfogadóban, akkor a másik leállítható (mert ilyenkor az únió is elfogad). Az 5. esetén „sorosan” kötjük egymás után.
□

Definíció. *Nyelvosztály*: nyelvek halmaza. X nyelvosztály komplementer nyelvosztálya:

$$\mathrm{co}\, X = \{L : \overline{L} \in X\}$$

Tehát minden X-beli nyelvnek külön-külön vesszük a komplementerét. Tulajdonságai:

1. Ha $X \subseteq Y$, akkor $\mathrm{co}\, X \subseteq \mathrm{co}\, Y$
2. $\mathrm{co}(\mathrm{co}\, X) = X$

4.8. Állítás. $\mathrm{co}\, R = R$

4.8. Bizonyítás. A fenti két tulajdonság felhasználásával:
$L \in R \Rightarrow \overline{L} \in R$. Vagyis $\operatorname{co} R \subseteq R \overset{1}{\Rightarrow} \operatorname{co}\operatorname{co} R \subseteq \operatorname{co} R \overset{2}{\Rightarrow} R \subseteq \operatorname{co} R$.
Tehát $R = \operatorname{co} R$ adódik. □

4.9. Állítás. $R = RE \cap \operatorname{co} RE$

4.9. Bizonyítás. A kétféle tartalmazást külön bizonyítjuk:

- $\subseteq$ irány: $R \subseteq \mathrm{RE} \overset{1}{\Rightarrow} \underbrace{\operatorname{co} R}_{R} \subseteq \operatorname{co} RE$.
- $\supseteq$ irány: Tegyük most fel, hogy $L \in RE$ és $L \in \operatorname{co} RE$, ekkor tehát van olyan M_1 Turing-gép, amely elfogad minden L-beli elemet és van olyan M_2 Turing-gép, amely elfogad minden $\overline{L}$-beli elemet. M_1-et és M_2-őt együttesen elindítva egy w szóval valamelyik biztosan meg fog állni úgy, hogy w-t elfogadja. Ekkor állítsuk meg a másikat is.

□

4.5. Az L_ε és az $L_\emptyset$ nyelv

Definíció.

$$L_\varepsilon = \{w : \exists M_w \text{ és } M_w \text{ az } \varepsilon \text{ bemenet hatására megáll } \}$$

4.10. Tétel (Church)**.**

$$L_\varepsilon \in (RE - R)$$

4.10. Bizonyítás. Először megmutatjuk, hogy az L_ε nyelv rekurzív felsorolható. Legyen M az L_h nyelvhez tartozó Turing-gép. (Ilyen létezik, hiszen $L_h \in RE$.) Tekintsünk most egy M' Turing-gépet, amely a w hatására az M Turing-gép működését szimulálja a $w\#\varepsilon$ bementen, azaz:

$$M'(w) = M(w\#\varepsilon)$$

Az M' Turing-gép minden L_ε-beli szón megáll, és mindet el is fogadja.
Most megmutatjuk, hogy L_ε nem rekurzív. Ehhez indirekt úton tegyük fel, hogy $L_\varepsilon \in R$, amely tehát azt jelenti, hogy létezik egy olyan M Turing-gép, amely L_ε minden szavát elfogadja és minden bemenetre megáll. A célunk, hogy létrehozzunk ebből egy olyan M' Turing-gépet, amely L_h minden szavát elfogadja, és minden bemenetre megáll. Működjön M' a következőképpen a $w\#s$ bemenet hatására:

- Ha w nem egy Turing-gép kódja, akkor utasítsuk el a bementet.
- Ha w egy Turing-gép kódja, akkor készítsük el belőle azt az M'' Turing-gépet, amelybe az s bemenet bele van építve, azaz: $M''(\varepsilon) = M_w(s)$. Ezt véges időben megtehetjük, például egy Turing-gép segítségével. Legyen M'' kódja w''.
- Szimuláljuk M működését a w'' bemeneten.

Sikerül tehát megépítenünk M'-t amely ellentmondás, hiszen $L_h \notin R$. □

Definíció.

$$L_\emptyset = \{w : \exists M_w \text{ és } L(M_w) = \emptyset\}$$

4.11. Tétel.

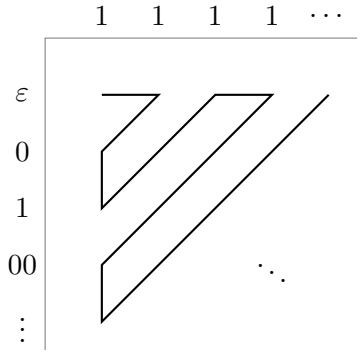
$$L_\emptyset \in (\operatorname{co} RE - R)$$

4.11. Bizonyítás. Először megmutatjuk, hogy $L_\emptyset \in \operatorname{co} RE$.

$$L_\emptyset \in \operatorname{co} RE \Leftrightarrow \overline{L_\emptyset} \in RE$$
$$\overline{L_\emptyset} = \{w : \neg\exists M_w \text{ vagy } (\exists M_w \text{ és } L(M_w) \neq \emptyset)\}$$

Konstruálni fogunk egy M Turing-gépet $\overline{L_\emptyset}$-hez. Működjön M a következőképpen a w bemeneten:

- Ellenőrzi, hogy w Turing-gép kód-e, és ha nem, akkor elfogadja.
- Ha w egy Turing-gép kódja, akkor futtassuk M_w-t a Σ^* i-edik elemén j lépésszám korláttal, minden i-re és j-re az ábrán látható bejárás szerint, amíg nem kapunk egy elfogadó számítást.



Tehát M minden olyan szóra megáll elfogadó állapotban amely eleme $\overline{L_\emptyset}$-nek.
Most még belátjuk, hogy $L_\emptyset \notin R$. Ehhez indirekt úton bizonyítva tegyük fel, hogy $L_\emptyset \in R$, és legyen az $L_\emptyset$ minden szavát elfogadó és minden bemenetre megálló Turing-gép M. Ebből egy olyan M'-t fogunk konstruálni, amely L_ε minden elemét elfogadja és minden bemenetre megáll. Működjön M' a következőképpen a w bemeneten:

- Ellenőrzi, hogy Turing-gép kód-e, és ha nem, akkor megáll és elutasítja a bemenetet.
- Ha w egy Turing-gép kódja, akkor hozzuk létre belőle a kövekezőképpen működő M'' Turing-gépet:
  - M'' nem foglalkozik a bemenetével, tehát minden x bemenetre ugyanúgy működik.
  - Ha $M_w(\varepsilon)$ megáll, akkor M'' megáll elfogadó állapotban.
  - Ha $M_w(\varepsilon)$ nem áll meg, akkor M'' sem áll meg.
  - Ez alapján $L(M'') = \Sigma^*$, ha $w \in L_\varepsilon$, és $L(M'') = \emptyset$, ha $w \notin L_\varepsilon$.
- Legyen M'' kódja w''.
- Szimuláljuk M-et s w'' bemenettel, és ha $M(w'')$ elfogad, akkor M' utasítsa el w-t, ha pedig $M(w'')$ elutasít, akkor M' fogadja el w-t.

Így tehát ha M' elfogadja w-t, akkor M elutasítja w''-t, azaz ilyenkor $w \in L_\varepsilon$. Hasonlóképpen ha M' elutasítja w-ta, akkor M elfogadja w''-t azaz ilyenkor $w \notin L_\varepsilon$. Azt is elmondhatjuk, hogy M' minden esetben megáll, mivel w'' létrehozható véges lépésben, és M mindig megáll. Így tehát az indirekt feltevéssel ellentmondásra jutottunk. □

4.6. Nyelvi tuladonságok és nyelvosztályok

Definíció. T egy *nem triviális nyelvi tulajdonság*, ha $\exists L_1, L_2 \in RE$, hogy L_1-re T teljesül és L_2-re T nem teljesül. A T tulajdonsággal rendelkező nyelvek halmazát is szokták T-vel jelölni, így megfogalmazva:

$$T \bigcap RE \neq \emptyset$$
$$T \bigcap RE \neq RE$$

4.12. Tétel (Rice)**.** Legyen:

$$L_T = \{w : \exists M_w \text{ és } L(M_w)\ T \text{ tulajdonságú } \}$$

Ha T egy nem triviális nyelvi tulajdonság, akkor $L_T \notin R$.

4.12. Bizonyítás. Az egyszerűség kedvéért tegyük fel, hogy $\emptyset \notin T$. (Ha nem így lenne, akkor T helyett nézhetnénk $\overline{T}$-t is, így a bizonyítás $L_{\overline{T}}$-re működne amiből a rekurzív nyelvek komplementerképzésre való zártsága miatt következtethetnénk L_T nemrekurzivitására, pár apróbb megfontolás mellett.) Legyen $L \in RE$ egy olyan nyelv, amelyre $L \in T$, és legyen M az a Turing-gép, amelyre $L(M) = L$. Indirekt úton bizonyítva tegyük fel, hogy $L_T \in R$, azaz létezik egy olyan M_T amely minden bemenetre megáll és $L(M_T) = L_T$. Ebből olyan M'-t fogunk konstruálni, amelyik minden bemenetre megáll és $L(M') = L_u$ Működjön M' a $w\#s$ bemenetre a következőképpen:

- Ha w nem egy Turing-gép kódja, akkor M' megáll és elutasíja a bemenetet.
- Ha w egy Turing-gép kódja, akkor konstruáljunk belőle egy M'' Turing-gépet, amely a következőképpen működik egy x bemeneten:
  - Ha $M_w(s)$ elfogad és $M(x)$ elfogad, akkor M'' megáll és elfogadja x-et
  - Ha $M_w(s)$ elfogad és $M(x)$ elutasít, akkor M'' megáll és elutasítja x-et
  - Ha $M_w(s)$ elfogad és $M(x)$ nem áll meg, akkor M'' sem áll meg.
  - Ha $M_w(s)$ elutasít, akkor M'' megáll és elutasítja x-et
  - Ha $M_w(s)$ nem áll meg, akkor M'' sem áll meg.
- Figyeljük meg, hogy $M''(x) = L$, ha $w\#s \in L_u$, és $M''(x) = \emptyset$ ha $w\#s \notin L_u$.
- Legyen M'' kódja w''.
- M' fogadja el a bemenetét, ha $M_T(w'')$ elfogad, és utasítsa el a bemenetét ha $M_T(w'')$ elutasít.

Mivel M_T mindig megáll, M' is mindig meg fog állni. M' akkor fogadja el a bementetét, amikor M_T elfogadja w'' azaz amikor az M'' által elfogadott nyelv T tulajdonságú. Mivel $L \in T$ és $\emptyset \notin T$ ez pontosan akkor lesz ha $w\#s \in L_u$. Ebből tehát $L_u \in R$ lenne, így az indirekt feltevésünk ellentmondásra vezetett. □

A Rice tétel következményei:

T : a nyelv nem üres $\Rightarrow L_T = \{w : \exists M_w, L(M_w) \neq \emptyset\} \notin R$

T : a nyelv véges $\Rightarrow L_T = \{w : \exists M_w, L(M_w)$ véges $\} \notin R$

T : a nyelvnek legalább 100 eleme van $\Rightarrow$
$L_T = \{w : \exists M_w, L(M_w)$-nek legalább 100 eleme van $\} \notin R$

T : a nyelvbe beletartozik egy rögzített s szó $\Rightarrow L_T = \{w : \exists M_w, s \in L(M_w)\} \notin R$

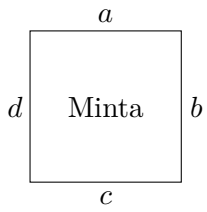
T : a nyelv reguláris $\Rightarrow L_T = \{w : \exists M_w, L(M_w)$reguláris $\} \notin R$

T : a nyelv $= \Sigma^* \Rightarrow L_T = \{w : \exists M_w, L(M_w) = \Sigma^*\} \notin R$

Ezekhez természetesen be kell látnunk, hogy az itt felsorolt T tulajdonságok nem triviálisak, ennek bizonyítását az olvasóra bízzuk.

4.7. A Dominó probléma

Definíció. A *dominó probléma* egy olyan sík lefedési feladat, ahol a teljes síkot kell megadott építőelemekkel (dominók) lefednünk. Adott egy dominókészlet, amely véges sok dominótípust tartalmaz és minden típusból megszámlálhatóan végtelen darab áll rendelkezésre. A dominók típusa azt adja meg, hogy milyen dominókat lehet melléjük helyezni. Egy ilyen típust az (a, b, c, d) négyessel adhatunk meg, amely a dominó oldalainak típusát írja le a felső oldallal kezdve az óramutató járásának megfelelő körüljárási irányban. Az oldaltípusok azt írják le, hogy csak ugyanolyan típusú oldal kerülhet mellé. tehát ha a felső oldal típusa a_1, akkor csak olyan dominót rakhatunk fölé, amelyiknek az alsó oldala a_1 típusú.



A dominókon mintázat is található, ezért nem szabad őket elforgatni. A kérdés, hogy adott készlettel le tudjuk-e fedni a síkot.

4.4. Példa. Nézzük meg, az alábbi készleket közül melyek alakalmasak a sík lefedésére:

- $\{(a, a, a, a)\}$
- $\{(a, b, b, b), (a, a, b, b)\}$

Az első készlettel le tudjuk fedni a síkot, hiszen bármely két dominó egymás mellé illeszthető bármilyen irányból, míg a másodikkal nem, mivel semelyik két dominót sem tudjuk egymás alá rakni.

Definíció. A *dominó nyelv* azon dominókészletek halmaza, amelyekkel a dominó probléma megoldható.

$$D = \{ \text{ a sík kirakására alkalmas dominó készletek } \}$$

4.13. Állítás. Egy dominókészlettel kirakható a sík $\Leftrightarrow$ minden $2k \times 2k$-as négyzet kirakható. $(k = 1, 2, \dots)$

4.13. Bizonyítás. A két irány külön bizonyítjuk:

$\Rightarrow$ irányban triviális.

$\Leftarrow$ irányban: Tekintsünk egy fát a sík kirakásához a következőképpen. A gyökér fiai a 2×2-es négyzetek kirakásai, amelyekből legalább egy, de legfeljebb t^4 darab van, ahol t a készletben a típusok száma. A 2×2-es négyzetek fiai legyenek az ő 4×4-es lehetséges folytatásaik. A 4×4-es folytatások fiai legyenek az ő 6×6-os folytatásaik, és így tovább. Ez egy gyökeres szintezett fa, minden szinten véges sok csúcs van, és minden nem-gyökér elemnek van apja, valamint összesen végetelen sok csúcs van. Induljunk ki a gyökérből. Mivel véges sok fia van, nyilván van olyan fia, aminek a részfája végtelen sok csúcsot tartalmaz. Válasszunk egy ilyet, majd tekintsük az ő fiait. Mivel neki is csak véges sok fia van ezek között biztosan van olyan aminek a részfája végtelen sok elemet tartalmaz. Figyeljük meg, hogy ez a bejárás sosem akad el, és végül találunk egy olyan utat amely a gyökérből indul és végtelen hosszú.

□

4.14. Állítás.

$$D \in \operatorname{co} RE$$

4.14. Bizonyítás.

$$D \in \operatorname{co} RE \Leftrightarrow \overline{D} \in RE \Leftrightarrow \exists M : L(M) = \overline{D}$$

Működjön M a következőképpen:

- Ha a bemenet nem egy dominó-készlet, akkor megáll és elfogad.
- Ha a bemenet egy dominó-készlet, akkor $k = 1, 2, \ldots$ esetekre kipróbálja az összes lehetőséget a $2k \times 2k$-as négyzet lefedésére. Ha valamely k-ra nincs megoldás, akkor megáll és elfogadja.

□

4.15. Tétel.

$$D \notin R$$

4.15. Bizonyítás. A tételt nem bizonyítjuk. □

4.8. Post megfeleltetés problémája

Definíció. *Post megfeleltetés problémája* szerint adottak a Σ ábécé feletti $\{(s_i, t_i) : i = 1, 2 \ldots k\}$ szópárok. A kérdés, hogy van-e olyan nem üres $i_1, i_2, \ldots i_n$ indexsorozat, hogy:

$$s_{i_1}, s_{i_1} \ldots s_{i_n} = t_{i_1}, t_{i_1} \ldots t_{i_n}$$

4.5. Példa. Néhány példa:
A szópárok: $\{(ab, aba), (ab, ba), (aba, ba)\}$.
$\underbrace{ab}_{1}\underbrace{aba}_{3}$, illetve $\underbrace{aba}_{1}\underbrace{ba}_{3}$ megfeleltetésekkel az $(1, 3)$ indexsorozat megfelelő.
A szópárok: $\{(1, 111), (10111, 10), (10, 0)\}$.
$\underbrace{10111}_{2}\underbrace{1}_{1}\underbrace{1}_{1}\underbrace{100}_{3}$, illetve $\underbrace{10}_{2}\underbrace{111}_{1}\underbrace{111}_{1}\underbrace{0}_{3}$ megfeleltetésekkel a $(2, 1, 1, 3)$ indexsorozat megfelelő.
A szópárok: $\{(ab, a), (ab, ba), (b, ba)\}$.
Figyeljük meg, hogy ebben az esetben nincs jó indexsorozat, mert a szópárok első tagja mindig b, míg a második mindig a karakterre végződik, ezért bárhogyan is próbálkoznánk nem találnánk utolsó szót.

Definíció. A *Post megfeleltetés probléma nyelve*:

$$PCP = \{(s_i, t_i) : i = 1, 2 \ldots k, \text{ van megfelelő indexhalmaz } \}$$

4.16. Tétel.

$$PCP \in RE - R$$

4.16. Bizonyítás. Először megmutatjuk, hogy $PCP \in RE$, ehhez a következő algoritmussal működő Turing-gépet vegyük szemügyre:

- Először szintaktikailag ellenőrzi a bementetét, ha nem megfelelő, akkor elutasítja
- Kipróbálja az összes lehetséges indexsorozatot úgy, hogy egyre hosszabb sorozatokat vesz. Ha van megoldás véges sok lépés után megáll és elfogadja a bementet, ha nincs megoldás nem áll meg.

Most még be kellene látnunk, hogy $PCP \notin R$. A tételnek ezt a részét nem bizonyítjuk. □

A tétel alkalmazásával további állításokat fogalmazhatunk meg.

4.9. CF nyelvtanokkal kapcsolatos eldönthetetlenségi eredmények

4.17. Állítás. Egy CF nyelvtan egyértelműsége algoritmikusan nem dönthető el, azaz a megfelelő nyelv nem rekurzív.

4.17. Bizonyítás. A bizonyításhoz egy $(s_i, t_i) : i = 1, 2 \dots k$ halmazhoz mutatunk olyan G környezetfüggetlen nyelvtant, hogy:

$$(s_i, t_i) : i = 1, 2 \dots k \notin PCP \Leftrightarrow G \text{ egyértelmű}$$

Legyenek G szabályai az alábbiak:

- $S \to A|B$
- $A \to s_i A a_i | s_i a_i$, minden $i = 1, 2 \dots k$-ra
- $B \to t_i B a_i | t_i a_i$, minden $i = 1, 2 \dots k$-ra

Itt a szavak indexelésére az $a_1, a_2, \dots a_k$ karaktereket használjuk, így ezekkel ki kell egészítenünk az ábécét. Ekkor ha a PCP probléma megoldása az $i_1, i_2, \dots i_n$ indexsorozat, akkor:

$$s_{i_1} s_{i_2} \dots s_{i_n} = t_{i_1} t_{i_2} \dots t_{i_n}$$

Ez viszont azt jelenti, hogy a következő két kifejezés levezethető a nyelvtanból

$$s_{i_1} s_{i_2} \dots s_{i_n} a_{i_n} a_{i_{n-1}} \dots a_{i_1}$$
$$t_{i_1} t_{i_2} \dots t_{i_n} a_{i_n} a_{i_{n-1}} \dots a_{i_1}$$

Ez a két kifejezés a fentiek miatt azonos szavakat ír le, de levezetési fájuk különböző, mivel más szabályokkal hoztuk őket létre. $L(G)$ pontosan akkor nem egyértelmű ha létezik olyan szó amely A-ból és B-ből is megkapható, ami tehát azt jelenti, hogy a PCP problémának nincs megoldása. □

4.18. Tétel. Adott G_1 és G_2 környezetfüggetlen nyelvtanok:

- $L(G_1) \stackrel{?}{=} L(G_2)$ algoritmikusan eldönthetetlen
- $L(G_1) \stackrel{?}{=} \Sigma^*$ algoritmikusan eldönthetetlen
- $L(G_1) \bigcap L(G_2) \stackrel{?}{=} \emptyset$ algoritmikusan eldönthetetlen

4.18. Bizonyítás. Az állítások a 4.19 tétel következményeiként előállnak. □

Korábban már láttuk, hogy bármely többszalagos, több elfogadó állapottal rendelkező Turing-gépet tudunk szimulálni olyan Turing-géppel, amelynek egy szalagja van, egy elfogadó állapta van és ebben az elfogadó állapotban megáll.

Turing-gép konfigráció: Ezzel leírhatjuk a fenti Turing-gépek pillanatnyi helyzetét. Legyen:

- x az olvasófej előtti lévő szalagrész
- q a pillanatnyi állapot
- y a szalag többi része (tehát az olvasófej y első karakterére mutat)

Ekkor xqy a Turing-gép pillanatnyi konfigurációjának kódja.

Definíció.

$$\text{elfogadó}(M) = \{w_1\#w_2^R\#w_3\#w_4^R \dots w_{2k+1}\} \bigcup \{w_1\#w_2^R\#w_3\#w_4^R \dots w_{2k}^R\}$$

Ahol:

- w^R a w konfigurációkód fordítottjának jelölése
- $\forall w_i$ egy lehetséges pillanatnyi állapota M-nek
- $w_1 = q_0 y$
- w_i-ből egy lépésben előáll w_{i+1}
- Az utolsó konfigurációkódban $q \in F$
- $\# \notin (\Gamma \bigcup Q)$

Tehát elfogadó(M) az M Turing-gép elfogadó számításait tartalmazó nyelv, ahol minden számításban minden második pillanatnyi állapot kódját fordítva írtuk, a konfigurációkódok közé pedig a # elválasztójelet raktuk.

4.19. Tétel. Van olyan algoritmus, amely adott M Turing-géphez meghatároz G_1, G_2, G_3 környezetfüggetlen nyelvtanokat úgy, hogy:

- $\text{elfogadó}(M) = L(G_1) \bigcap L(G_2)$
- $\overline{\text{elfogadó}(M)} = L(G_3)$

4.19. Bizonyítás. Vázlat. Legyen M_1, M_2 és M_3 rendre az $L(G_1)$, $L(G_2)$ és $L(G_3)$ veremautomatája. Ha ilyen veremautomatákat elő tudunk állítani, azokból már létre tudjuk hozni a megfelelő nyelvtanokat.

- M_1 működése:
  - Ellenőrzi, hogy w_1 megfelelő-e
  - Ellenőrzi, hogy $w_1 \to w_2$, $w_3 \to w_4$ … $w_{2i-1} \to w_{2i}$ átmenetek lehetségesek-e.
- M_2 működése:
  - Ellenőrzi, hogy $w_2 \to w_3$, $w_4 \to w_5$ … $w_{2i} \to w_{2i+1}$ átmenetek lehetségesek-e.
  - Ellenőrzi, hogy az utolsó w konfigurációkód megfelelő-e.
- M_3 működése:
  - Ellenőrzi, hogy a bemenet megfelelő alakú-e (erre egy véges automata is képes)
  - Ellenőrzi, hogy a w_1 megfelelő-e (erre egy véges automata is képes)
  - Ellenőrzi, hogy az utolsó w konfigurációkód megfelelő-e (erre egy véges automata is képes)

– A veremautomaták nemdetermizmusát kihasználva ellenőrzi, hogy valamely $w_i \rightarrow w_{i+}$ átmenet hibás-e.

□

Megmutatjuk, hogy a 4.18 tétel állításai következményként előállnak:

4.20. Állítás. Ha G_1 és G_2 környezetfüggetlen nyelvtanok, akkor $L(G_1) \bigcap L(G_2) \stackrel{?}{=} \emptyset$ algoritmikusan eldönthetetlen.

4.20. Bizonyítás. Indirekt úton tegyük fel, hogy van ilyen algoritmus, és tekintsünk egy M Turing-gépet. Legyen G_1 és G_2 a 4.19 tétel szerint M-ből létrehozható két környezetfüggetlen nyelvtan.

$$(L(G_1) \bigcap L(G_2) \neq \emptyset) \Leftrightarrow L(M) \neq \emptyset$$

Így tehát azt kaptuk, hogy ebben az esetben lenne algoritmus arra is, hogy az $L(M) \stackrel{?}{=} \emptyset$ kérdést megválaszoljuk, ami ellentmondás. □

4.21. Állítás. Ha G_1 és G_2 környezetfüggetlen nyelvtanok, akkor $L(G_1) \stackrel{?}{=} \Sigma^*$ algoritmikusan eldönthetetlen

4.21. Bizonyítás. Indirekt úton tegyük fel, hogy van ilyen algoritmus, és tekintsünk egy M Turing-gépet. Legyen G_3 a 4.19 tétel szerint M-ből létrehozható környezetfüggetlen nyelvtan.

$$L(G_3) = \Sigma^* \Leftrightarrow L(M) = \emptyset$$

Így tehát azt kaptuk, hogy ebben az esetben lenne algoritmus arra is, hogy az $L(M) \stackrel{?}{=} \emptyset$ kérdést megválaszoljuk, ami ellentmondás. □

A harmadik állítás bizonyítását az olvasóra bízzuk.

4.10. Felsorolós Turing-gépek

Definíció. A *Felsorolós Turing-gép* egy speciális Turing-gép változat:

- Nincs bemenete.
- Egy, csak írható kimeneti szalagja van, amin nem léphet vissza.
- Tetszőleges számú írható és olvasható munkaszalagja van.
- Nincs elfogadó állapota.
- A kimenete $x_1\#x_2\#x_3\ldots$ alakú, és lehet végtelen hosszú (ilyenkor nem áll meg a Turing-gép) vagy véges hosszú (ilyenkor megáll a Turing-gép)
- A felsorolt nyelv: $\{x_1, x_2, x_3 \ldots\}$
- Az elemek felsorolása közben lehet ismétlődés.

4.22. Tétel.

$$L \in RE \Leftrightarrow \text{van olyan felsorolós Turing-gép, ami } L\text{-et sorolja fel}$$

4.22. Bizonyítás.

$\Leftarrow$ irányban:

Most egy L elemeit felsoroló M Turing-gépünk van, amelyből egy L-et elfogadó M' Turing gépet akarunk konstruálni. Működjön M' a következőképpen az x bemeneten:

- Futtatja M-et és minden w kiírt szóra ellenőrzik, hogy $w \stackrel{?}{=} x$
- Ha igen, akkor megáll és elfogadja x-et
- Ha M megáll, de nem írta ki x-et, akkor megáll és elutasítja x-et

$\Rightarrow$ irányban:

Most egy L-t elfogadó M Turing-gépünk van, amelyből egy L elemeit felsoroló M' Turing gépet akarunk konstruálni. M' minden Σ^*-beli y_i szón futtatja M-et legfeljebb j lépésig. Ha M elfogadja y_i-t, akkor kiírja. A szokásos diagonális bejárást alakalmazva haladunk i és j szerint.

□

4.23. Tétel.

$L \in R \Leftrightarrow$ van olyan felsorolós Turing-gép, ami L elemeit sorrendben sorolja fel

4.23. Bizonyítás.

$\Leftarrow$ irányban:

Most egy L elemeit sorrendben felsoroló M Turing-gépünk van, amelyből egy L-t elfogadó, minden bemeneten megálló M' Turing gépet akarunk konstruálni. Működjön M' a következőképpen az x bemeneten:

- Futtatja M-et
- Ha M kiírja x-et akkor M' megáll és elfogad
- Ha M kiír egy olyan y-t, ami x után jön, vagy M megáll úgy hogy még nem írta ki x-et, akkor M' megáll és elutasít

$\Rightarrow$ irányban:

Most egy L-t elfogadó, minden bemenetre megálló M Turing-gépünk van, amelyből egy L elemeit felsoroló M' Turing gépet akarunk konstruálni. M' futtassa sorban Σ^* szavain M-et és írjon ki minden olyan szót, amit M elfogad.

□

4.24. Tétel.

$L \in R \Leftrightarrow$ Az alábbi módon definiált $f_L(x)$ függvény rekurzív

$$f_L(x) = \begin{cases} 1 & \text{ha } x \in L \\ 0 & \text{ha } x \notin L \end{cases}$$

4.24. Bizonyítás.

$\Rightarrow$ irányban:

Most egy L-t elfogadó, minden bemenetre megálló M Turing-gépünk van, amelyből egy olyan M' számoló Turing-gépet akarunk létre hozni, amely szintén megáll minden bemeneten, és $f_{M'} = f_L$. M' adjon vissza 1-et ha M elfogadja x-et, és adjon vissza 0-át ha M elutasítja x-et.

$\Leftarrow$ irányban:

Most egy olyan mindig megálló M Turing-gépünk van, hogy $f_M = f_L$ és ebből akarunk mindig megálló L-t elfogadó M' Turing-gépet konstruálni. Ha $M(x)$ kimenete 1, akkor M' fogadja el x-et, egyébként utasítsa el.

□

4.25. Tétel.

$L \in RE \Leftrightarrow$ Van olyan f parciális rekurzív függvény, amelynek az értékkészlete L

4.25. Bizonyítás.

$\Rightarrow$ irányban:

Legyen M egy olyan Turing-gép, hogy $L(M) = L$. Ekkor minden $x \in L$ bemenetre M megáll elfogadó állapotban.

$$f(x) = \begin{cases} x & \text{ha } x \in L \\ \text{nem definiált} & \text{ha } x \notin L \end{cases}$$

f értékkészlete nyilván L, és a következő M'-re igaz, hogy $f_{M'} = f$. M' az x bemeneten futtatja M-et és ha M elogadja x-et, akkor M' kiírja, ha elutasítja vagy nem áll meg rá, akkor M' végtelen ciklust hajt végre.

$\Leftarrow$ irányban:

Legyen M az a Turing gép, akire $f_M = f$. Ebből konstruálunk olyan M'-t, hogy $L(M') = \{f(x)\}$. M' az y bemenetre keres egy olyan x-et, hogy $f(x) = y$, azaz i és j szerinti diagonális bejárással minden x_i-n futtatja M-et j lépésig, és ha megáll és y-t írja ki, akkor M' megáll és elfogadja y-t.

□

4.11. Nemdeterminisztikus Turing-gépek

Definíció. A *nemdeterminisztikus Turing-gépeket* a következőképpen írhatjuk le:

$$T = (Q, \Sigma, \Gamma, q_0, \_, F, \delta)$$

ahol $Q, \Sigma, \Gamma, q_0, \_, F$ szerepe azonos a Turing gépek esetén definiálttal, valamint:

$$\delta(q, a) \subseteq Q \times \Gamma \times \{J, B, H\}$$

Azaz egy állapotból egy bemenet hatására többféle műveletet is végezhet a Turing gép. A nemdeterminizmusból fakadó választási lehetőségeket számítási fával is reprezentálhatjuk, melynek elágazásai az adott állapot után lehetséges további állapotokat jelentik. A nemdeterminisztikus Turing-gép akkor fogadja el a bemenetét., ha a számítási fában van elfogadó levél. Megjegyezzük, hogy nemdeterminisztikus Turing-gépek esetén is definiálható többszalagos gép, amely szimulálható egy egyszalagossal.

4.26. Tétel. Bármely nemdeterminisztikus Turing-gép szimulálható determinisztikus Turing-géppel.

4.26. Bizonyítás. Vázlat. Legyen M a nemdeterminisztikus Turing-gép, és tekintsük a számítási fáját. Ennek minden csúcsából legfeljebb $3|Q||\Gamma|$ irányba indulhatnak elágazások, tehát minden helyzetben véges sok lehetőség közül választatunk. Működjön az M' Turing-gép a következőképpen:

- Szélességi bejárással végigmegy M számítási fáján
- Ha elfogadó levelet talál, akkor megáll és elfogad
- Ha végigment a teljes fán, és nem talált elfogadó levelet, akkor elutasít.
- Egyébként (azaz a fa végtelen és nincs elfogadó levele) nem áll meg.

□

4.27. Tétel. G egy nyelvtan $\Rightarrow \exists M$ Turing-gép, hogy $L(M) = L(G)$.

4.27. Bizonyítás. Mivel már láttuk, hogy a nemdeterminisztikus Turing-gépek determinizálhatók, elegendő egy nemdeterminisztikus M Turing-gépet mutatnunk, amely megoldja a feladatot. Legyen 4 szalagunk, amelyből kezdetben az elsőre helyezzük a bemenetet, a másodikra pedig a G nyelvtan S kezdőváltozóját. A következő nemdeterminisztikus lépések leírják M működését:

1. Állítsuk az olvasófejet a 2. szalag valamely pontjára
2. A 3. szalagra írjunk valamilyen α, változókból és Σ-beli karakterekből álló sorozatot.
3. A 4. szalagra írjunk valamilyen β, változókból és Σ-beli karakterekből álló sorozatot.
4. Ellenőrizzük, hogy a 2. szalagon a fejtől kezdve megtalálható-e α és, hogy $\alpha \to \beta$ egy G-beli szabály-e. Ha igen, akkor a második szalagon α-át cseréljük β-ra.
5. Ellenőrizzük, hogy a 2. szalag tartalma egyezik-e az elsővel, ha igen, akkor elfogadjuk a bemenetet, ha nem akkor megismételjük az előbbieket az első lépéstől.

A konstrukció helyességének belátását itt nem részletezzük. □

4.28. Tétel. $L \in RE \Rightarrow \exists G$ nyelvtan, hogy $L(G) = L$.

4.28. Bizonyítás. Tekintsük az L nyelvnek megfelelő M Turing-gépet, amelyről feltételezzük, hogy egyszalagos, egy elfogadó állapota van, és az elfogadó állapotából nem tud kilépni. Legyenek a nyelvtan változói a következők:

- S, T_1, T_2
- $[t, x]$ felírású változók, ahol $t \in \Sigma \bigcup \{\varepsilon\}$ és $x \in \Gamma$
- Q elemei

Vegyük fel a következő nyelvtani szabályokat:

- $S \to q_0 T_1$
- $T_1 \to [a, a]T_1$ minden $a \in \Sigma$ esetén
- $T_1 \to T_2$
- $T_2 \to [\varepsilon, \_]T_2 | \varepsilon$

Ezután M átmeneti függvényének megfelelően vegyük fel a következő szabályokat:

- Minden $\delta(q, x) = (p, y, J)$ esetén vegyünk fel $q[a, x] \to [a, y]p$ szabályokat minden $a \in \Sigma \bigcup \{\varepsilon\}$-ra
- Minden $\delta(q, x) = (p, y, B)$ esetén vegyünk fel $[b, z]q[a, x] \to p[b, z][a, y]$ szabályokat minden $a, b \in \Sigma \bigcup \{\varepsilon\}$, $z \in \Gamma$-ra.

- Minden $\delta(q, x) = (p, y, H)$ esetén vegyünk fel $q[a, x] \to p[a, y]$ szabályokat minden $a \in \Sigma \bigcup \{\varepsilon\}$-ra

Végül vegyük még fel a következő szabályokat minden q elfogadó állapot esetén:

- $q[a, x] \to qaq$
- $[a, x]q \to qaq$
- $q \to \varepsilon$

A konstrukció helyességének belátását itt nem részletezzük. □

5. fejezet

Bonyolultságelmélet

5.1. A számítás költsége

Definíció. M Turing-gép *időigénye* $T_M(n)$ nemnegatív függvény, amely megadja, hogy az n hosszú bemeneten mennyi M maximális lépésszáma. Ha a gép valamelyik n hosszú bemenetén nem áll meg, akkor a függvény értéke ∞.
Hasonlóan, jelöljük M gép *tárigényét* $S_M(n)$-nel, mely megadja, hogy az n hosszú bemeneten M legfeljebb mennyi szalagcellát használ fel. Az egyszalagos Turing-gépnél ezalatt azt értjük, hogy a gép milyen „messze" ment el a szalagon. Többszalagos gép esetén a bemeneti (és csak olvasható) szalagok nem számítanak, csak a munkaszalagok összege.
Ha M nemdeterminisztikus Turing-gép, akkor $T_M(n)$, illetve $S_M(n)$ a maximális lépésszámot, illetve használt cellák számát adja az n hosszú bemenetek számítási fájának összes ága közül.

Probléma: $T_M(n)$ és $S_M(n)$ meghatározása nem mindig triviális, ezért pontos megadás helyett megelégszünk felső becsléssel.

Definíció. Az M Turing-gép

- $t(n)$ *időkorlátos*, ha minden n-re fennáll, hogy: $T_M(n) \leq t(n)$.
- $s(n)$ *tárkorlátos*, ha minden n-re fennáll, hogy: $S_M(n) \leq s(n)$.

Ökölszabályként azt mondhatjuk, hogy a bemenet beolvasása miatt általában $t(n) \geq n$, illetve a bemenet címzése miatt $s(n) \geq \log_2 n$ fennáll.

Tulajdonságok: Ha M $t(n)$ időkorlátos, akkor $L(M) \in R$. A tárkorlátosságra nem tudunk ilyen kijelentést tenni.

5.1. Állítás. M k-szalagos Turing-géphez létezik M' egyszalagos Turing gép, hogy $L(M) = L(M')$, és

- $T_{M'} = \mathrm{O}({T_M}^2(n))$
- $S_{M'} = \mathrm{O}(S_M(n) + n)$

5.1. Bizonyítás.

idő Amíg M egy lépést tesz, addig M' ezt a lépést azzal szimulálja, hogy elmegy a szalag „végéig" és utána visszafordul (plusz pár helyen át is írhat értéket). A szalag vége legfeljebb $T_M(n)$ messze van. Tehát M egy lépését $\mathrm{O}(T_M(n))$ lépésben tudja szimulálni M'. M a futása során legfeljebb $T_M(n)$ lépés után áll meg, így M' lépésszáma $\mathrm{O}({T_M}^2(n))$.

tár M' ugyanannyi munkaterületet használ, azonban a bemenetet is tárolnia kell.

□

5.2. Tétel. M k-szalagos Turing-géphez létezik M' kétszalagos Turing-gép, hogy:

- $L(M) = L(M')$
- $T_{M'} = \mathrm{O}(T_M(n) \cdot \log T_M(n))$
- $S_{M'} = \mathrm{O}(S_M(n))$

5.3. Tétel (Lineáris gyorsítás)**.** L nyelvhez létezik M, hogy $L(M) = L$ és $T_M(n) \leq cn$ ($c \geq 1$ konstans), akkor minden $\varepsilon \geq 0$ esetén létezik M', hogy $L(M') = L$ és $T_{M'}(n) \leq (1 + \varepsilon)n$

Megvalósítás: A bizonyítást mellőzzük, de megjegyezzük, hogy a gyorsítás például úgy lehetséges, M' k lépést tud olvasni/írni a szalag(ok)ról egyszerre.

5.4. Tétel (Gyorsítás)**.** Minden M Turing-géphez létezik $L \in R$ nyelv, hogy $L(M) = L$. Minden M-hez létezik M' Turing-gép, hogy $L(M') = L$ és

- $T_{M'}(n) = \mathrm{O}(\log T_M(n))$ végtelen sok n-re
- $T_{M'}(n) \leq T_M(n)$ minden n-re

5.2. Nyelvosztályok

Definíció (Nyelvosztályok)**.**

$$\begin{aligned}
\mathrm{TIME}(f(n)) = \{&L \text{ nyelv} : \exists M \text{ Turing-gép, hogy} \\
&L(M) = L \text{ és } M\ \mathrm{O}(f(n)) \text{ időkorátos}\} \\
\mathrm{SPACE}(f(n)) = \{&L \text{ nyelv} : \exists M \text{ Turing-gép, hogy} \\
&L(M) = L \text{ és } M\ \mathrm{O}(f(n)) \text{ tárkorátos}\} \\
\mathrm{NTIME}(f(n)) = \{&L \text{ nyelv} : \exists M \text{ nemdeterminisztikus Turing-gép, hogy} \\
&L(M) = L \text{ és } M\ \mathrm{O}(f(n)) \text{ időkorátos}\} \\
\mathrm{NSPACE}(f(n)) = \{&L \text{ nyelv} : \exists M \text{ nemdeterminisztikus Turing-gép, hogy} \\
&L(M) = L \text{ és } M\ \mathrm{O}(f(n)) \text{ tárkorátos}\}
\end{aligned}$$

Tulajdonságok:

- $\mathrm{TIME}(f(n)) \subseteq \mathrm{NTIME}(f(n))$
- $\mathrm{SPACE}(f(n)) \subseteq \mathrm{NSPACE}(f(n))$
- $\mathrm{TIME}(f(n)) \subseteq R$
- $\mathrm{SPACE}(f(n)) \subseteq RE$
- $\forall n$-re, ha $f(n) \leq g(n)$, akkor $\mathrm{TIME}(f(n)) \subseteq \mathrm{TIME}(g(n))$

5.5. Tétel. Létezik f függvény, hogy $\mathrm{TIME}(f(n)) = \mathrm{TIME}(2^{f(n)})$

5.6. Tétel (Hierarchia)**.** Legyen f függvény olyan, hogy $\exists M$ Turing-gép és $T_M(n) = f(n)$. Ekkor

- $\mathrm{TIME}(f(n)) \subset \mathrm{TIME}(f(n) \cdot \log^2 f(n))$
- $\mathrm{SPACE}(f(n)) \subset \mathrm{SPACE}(f(n) \cdot \log f(n))$

Definíció.

$$\mathrm{P} = \bigcup_{k=1}^{\infty} \mathrm{TIME}(n^k)$$
$$\mathrm{NP} = \bigcup_{k=1}^{\infty} \mathrm{NTIME}(n^k)$$
$$\mathrm{PSPACE} = \bigcup_{k=1}^{\infty} \mathrm{SPACE}(n^k)$$
$$\mathrm{EXPTIME} = \bigcup_{k=1}^{\infty} \mathrm{TIME}(2^{n^k})$$

Tulajdonságok: $P \subseteq NP$. A nemdeterminisztikus számítási fa bejárásának időigénye exponenciális a mélységben, tárigénye lineáris. Vagyis:

$\mathrm{NP} \subseteq \mathrm{EXPTIME}$
$\mathrm{NP} \subseteq \mathrm{PSPACE}$
$\mathrm{P} \subset \mathrm{EXPTIME}$, a többi egyenlőségről nem tudunk
$\mathrm{P} \subseteq \mathrm{coNP}$, mert $\mathrm{P} = \mathrm{coP}$

5.7. Állítás. $\mathrm{TIME}(f(n)) \subseteq \mathrm{SPACE}(f(n))$, azaz a lépésszából következtetni lehet a tárigényre.

5.7. Bizonyítás. $\mathrm{O}(f(n))$ időben legfeljebb $\mathrm{O}(f(n))$ tárat lehet használni. □

5.8. Tétel (Tár-idő)**.** Ha $L \in \mathrm{SPACE}(s(n))$, $s(n) \geq \log n$, akkor létezik $c_L > 0$ konstans, hogy $L \in \mathrm{TIME}({c_L}^{s(n)})$

5.8. Bizonyítás. Legyen M egy k-szalagos Turing-gép ($k > 1$), amelyre a tárkorlát $d > 0$ konstans esetén $S_M(n) \leq d \cdot s(n)$. Vegyük számításba, hogy hányféle különböző helyzete lehet a gépnek egy-egy tulajdonsága alapján.

Tulajdonság	Lehetséges helyzetek száma
aktuális állapot	$\lvert Q\rvert$
munkaszalagok tartalma	$\lvert\Gamma\rvert^{k \cdot S_M(n)}$
fejek a munkaszalagokon	${S_M(n)}^k$
fej a bemeneten	n

Mivel a tulajdonságok függetlenek, ezért az összes lehetséges helyzetek száma:

$$\lvert Q\rvert \cdot \lvert\Gamma\rvert^{k \cdot S_M(n)} \cdot {S_M(n)}^k \cdot n \leq \lvert Q\rvert \cdot \lvert\Gamma\rvert^{k \cdot d \cdot s(n)} \cdot 2^{(\log S_M(n))k} \cdot 2^{s(n)} \leq \underbrace{\lvert Q\rvert \cdot \left(\lvert\Gamma\rvert^{kd} \cdot 2^{kd+1}\right)^{s(n)}}_{t} = \mathrm{O}({c_L}^{s(n)})$$

Ötlet: Futtassuk M-et ez előző jelölése alapján $t + 1$ lépésig. Ha megállt, akkor elfogadunk. Ha nem állt meg, akkor legalább egy helyzet kétszer is előfordult a gép futása során, így az ismétlődés végtelen ciklushoz vezet, így megállíthatjuk a gépet és elutasíthatunk.

Probléma: t nem mindig számítható.

Javítás: Vegyük M két példányát: M_1, M_2. M_1-et lépésenként futtatjuk. Ha M_1 az l-edik lépésnél tart, akkor M_2-t a számítás elejéről indítjuk és $l - 1$ lépésig hagyjuk futni. M_2 minden lépése után összehasonlítjuk M_1 és M_2 helyzetét. Ha megegyezik, akkor végtelen ciklus lenne, ezért elutasítunk. Ha M_1 megáll, akkor elfogadunk. Ekkor M_1 legfeljebb $t + 1$ lépést, míg M_2

összesen $1 + 2 + \cdots + t = \Theta(t^2)$ lépést tesz. A rendszernek együttesen így $O(t^2) = O(c_L{}^{2s(n)})$ időigénye van, tárigénye $O(s(n))$.

□

Következmény:

- SPACE$(s(n)) \subseteq R$
- PSPACE $\subseteq$ EXPTIME
- P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXPTIME

5.9. Tétel (Tanú tétel)**.** $L \in$ NP $\Leftrightarrow \exists c > 0$ konstans és $L_1 \in P$, hogy

$$L = \{x : \exists \underset{\substack{\uparrow \\ \text{tanú}}}{y}, |y| \leq |x|^c, (x, y) \in L_1\}$$

5.9. Bizonyítás.

$\Rightarrow$ irányban: y egy elfogadó számítási út leírása. Ekkor $(x, y) \in L_1$ azt jelenti, hogy x bemeneten y egy létező jó számítási út. Tehát a nemdeterminisztikus Turing-gép számítási fájának egy megfelelő ága alkalmas tanú lesz.

$\Leftarrow$ irányban: Most egy olyan nemdeterminisztikus Turing-gépet kell L-hez mutatnunk, amely polinom időigényű. Mivel y hossza $|x|$-ben polinomiális vegyük nemdeterminisztikusan az egyik y lehetséges tanút majd nézzük meg, hogy $(x, y) \in L_1$ teljesül-e. Ezt polinom időben meg tudjuk nézni, mivel $L_1 \in P$ és a nemdeterminisztikus működés miatt az összes lehetséges y-ra ki tudjuk próbálni.

□

Definíció (Karp-redukció)**.** Legyen $f : \Sigma^* \to \Sigma^*$ egy polinom időben számolható függvény, $L_1, L_2 \subseteq \Sigma^*$ két nyelv. L_1 visszavezethető L_2-re, ha:

$$x \in L_1 \Leftrightarrow f(x) \in L_2$$

A továbbiakban így jelöljük: $L_1 \prec L_2$.

Tulajdonságok:

- Ha $L_1 \prec L_2$, akkor $L_2 \in$ P $\Rightarrow L_1 \in$P
- Ha $L_1 \prec L_2$, akkor $L_2 \in$ NP $\Rightarrow L_1 \in$NP
- Ha $L_1 \prec L_2$, akkor $\overline{L_1} \prec L_2$
- Ha $L_1 \prec L_2$ és $L_2 \prec L_3$, akkor $L_1 \prec L_3$

Definíció. L nyelv NP-teljes, ha $L \in$ NP és minden $L' \in$ NP és $L' \prec L$

Ahhoz tehát, hogy belássuk, hogy egy L nyelv NP-teljes, bizonyítanunk kell, hogy

1. $L \in$ NP
2. egy választott L' NP-teljes nyelvre be kell látni, hogy $L' \prec L$.

Vegyük észre, hogy ilyen módszerrel csak akkor tudunk bizonyítani, ha már létezik legalább egy NP-teljes nyelv.

Definíció. Legyen $\varphi(x_1, \ldots, x_n)$ Boole-formula, melyben szerepelhet

- $x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}$
- $\wedge$ (ÉS operátor)
- $\vee$ (VAGY operátor)
- zárójelek

Ekkor

$$\{\varphi(x_1, \ldots, x_n) : \exists b_1, \ldots, b_n \text{ behelyettesítés, hogy } \varphi(b_1, \ldots, b_n) = \text{igaz}\} = \text{SAT}$$

5.10. Tétel (Cook, Levin)**.** SAT nyelv NP-teljes.

5.10. Bizonyítás.

1. SAT $\in$ NP, mert $b_1, \ldots, b_n$ tanú, polinom hosszú és polinom ellenőrizhető
2. NP-teljes. Legyen $L \in$ NP nyelv, M egyszalagos, nemdeterminisztikus Turing-gép, hogy $L(M) = L$ és M polinom időkorlátos. Megmutatjuk, hogy M-hez és x-hez lehet olyan φ-t készíteni, hogy $x \in L$ pontosan akkor teljesül, amikor $\varphi \in SAT$. Ez tehát azt jelenti, hogy x nyelvbe való beletartozásának felismerését visszavezetjük arra, hogy egy x-hez konstruált b_x kielégít-e egy logikai kifejezést. Ehhez tekintsük M-et és vezessünk be logikai változókat:

   - x_{ij0}: az i-edik lépés után a j-edik cellában 0 van
   - x_{ij1}: az i-edik lépés után a j-edik cellában 1 van
   - $x_{ij\_}$: az i-edik lépés után a j-edik cellában _ van
   - y_{ij}: az i-edik lépés után a fej a j-edik cellán áll
   - z_{iq}: az i-edik lépés után az állapot q

   Ezzel tehát olyan logikai változókat adtunk meg, amelyek értékei az M Turing-gép működését írják le. Írjuk fel, hogy milyen megkötések szükségesek, hogy érvényes Turing-gép leírást, és elfogadó számítást kapjunk!

   - Adott i-re z_{iq} változók közül csak egy lehet igaz
   - Adott i-re y_{ij} változók közül csak egy lehet igaz
   - Adott i-re és j-re x_{ij0}, x_{ij1} és $x_{ij\_}$ változók közül csak egy lehet igaz
   - Két lépés között csak egyet léphet a gép jobbra vagy balra
   - Bekapcsoláskor a kezdőállapotnak kell lennie, és a szalagon az x bemenet van
   - Elfogadáskor elfogadó állapotban kell lennie

   Ezen logikai kifejezések összeésélésével tehát leírtuk az M Truring-gép működését, és látható, hogy ha egy x-hez létezik olyan ága a Turing-gép számítási fájának, amely elfogadó levélben ért véget (tehát M elfogadja x-et), akkor létezik egy b_x behelyettesítés amely kielégíti a fenti logikai kifejezést.

□

Definíció. Egy Boole-formula *konjunktív normál forma* ha az alábbi alakú:

$$(x_{i_1} \vee x_{i_2} \vee x_{i_3} \dots) \wedge (x_{i_j} \vee x_{i_{j+1}} \vee x_{i_{j+2}} \dots) \wedge \dots$$

Ahol az x_{i_k}-k a változókat vagy azok negáltjait reprezentáják.

5.11. Állítás. SAT nyelv a konjuktív normál formájú kifejezésekre is NP-tejes.

5.11. Bizonyítás. A Cook-Levin tétel bizonyítása működik akkor is ha a formula konjuktív normál forma. □

Definíció. A *3SAT* nyelv egy olyan része SAT nyelvnek, ahol a Boole-formula konjuktív disztjukt forma és minden zárójelben legfeljebb három változó van.

5.12. Tétel. A 3SAT nyelv NP-teljes.

5.12. Bizonyítás. Először belátjuk a 3SAT nyelv NP-beliségét. A megfelelő változóbehelyettesítés itt is jó tanú lesz tehát a 3SAT nyelv NP-beli. Most megmutatjuk, hogy SAT $\prec$ 3SAT. Ehhez mutatunk egy $\varphi \to \psi$ megfeleltetés minden φ kifejezésre, ami szerint:

$$\varphi \in SAT \Leftrightarrow \psi \in 3SAT$$

Tekintsük a φ kifejezés kiértékelési fáját és minden nem levél csomópontnak feleltessük meg egy z_i változót. Ekkor a φ kifejezés teljesülése felírható a z_i változók közötti szabályokkal. Például:

$$\begin{aligned}
\varphi &= (x_1 \vee \overline{x_2} \vee x_3 \vee x_4) \vee (x_2 \wedge \overline{x_5}) \\
z_1 &= x_1 \vee \overline{x_2} \\
z_2 &= z_1 \vee x_3 \\
z_3 &= z_2 \vee x_4 \\
z_4 &= x_2 \wedge \overline{x_5} \\
z_5 &= z_3 \vee z_4
\end{aligned}$$

Ezután a z_i változók közötti szabályokat az alább két módon logikai kifejezéssé alakíthatjuk:

$$\begin{aligned}
z_i = z_j \wedge z_k &\rightsquigarrow (z_i \vee \overline{z_j} \vee \overline{z_k}) \wedge (\overline{z_i} \vee z_j) \wedge (\overline{z_i} \vee z_k) \\
z_i = z_j \vee z_k &\rightsquigarrow (\overline{z_i} \vee z_j \vee z_k) \wedge (z_i \vee \overline{z_j}) \wedge (z_i \vee \overline{z_k})
\end{aligned}$$

A fenti példában ez a következőképpen néz ki:

$$\begin{aligned}
&(\overline{z_1} \vee x_1 \vee \overline{x_2}) \wedge (z_1 \vee \overline{x_1}) \wedge (z_1 \vee x_2) \\
&(\overline{z_2} \vee z_1 \vee x_3) \wedge (z_2 \vee \overline{z_1}) \wedge (z_2 \vee \overline{x_3}) \\
&(\overline{z_3} \vee z_2 \vee x_4) \wedge (z_3 \vee \overline{z_2}) \wedge (z_3 \vee \overline{x_4}) \\
&(z_4 \vee \overline{x_2} \vee x_5) \wedge (\overline{z_4} \vee x_2) \wedge (\overline{z_4} \vee \overline{x_5}) \\
&(\overline{z_5} \vee z_3 \vee z_4) \wedge (z_5 \vee \overline{z_3}) \wedge (z_5 \vee \overline{z_4})
\end{aligned}$$

Ezután ψ-t a fenti kifejezések összeéseléséből kaphatjuk meg. A fenti konstrukcióval ψ hossza legfeljebb φ hosszának konstansszorosa. □

Definíció. A *2SAT* nyelv egy olyan SAT nyelv, ahol a Boole-formula konjuktív disztjukt forma és minden zárójelben legfeljebb két változó van.

5.13. Tétel. A 2SAT $\in$ P

5.13. Bizonyítás. A tételt nem bizonyítjuk. □

Definíció. A *3SZÍN* nyelv az olyan G gráfok halmaza, amelyek csúcsait ki lehet színezni 3 színnel.

5.14. Tétel. A 3SZÍN nyelv NP-teljes.

5.14. Bizonyítás. A 3SZÍN nyelv NP-belisége könnyen belátható, mivel egy megfelelő színezés polinomhosszú tanú. Az NP-teljesség belátásához megmutatjuk, hogy 3SAT $\prec$ 3SZÍN. Ehhez egy tetszőleges ψ formulához konstruálunk egy olyan G gráfot, hogy:

$$\psi \in 3SAT \Leftrightarrow G \in 3SZÍN$$

Legyenek ψ változói $x_1, x_2 \dots x_n$. G csúcsai legyenek $x_1, \overline{x_1} \dots x_n, \overline{x_n}$ valamint U és V továbbá minden $(x_i \vee x_j \vee x_k)$ formulához vegyünk fel még 5-5 csúcsot $(K_{i_1} \dots K_{i_5})$ Minden x_i és $\overline{x_i}$ pontot kössünk össze éllel, továbbá minden $x_1 \dots x_n$ és $\overline{x_1} \dots \overline{x_n}$ pontot kössünk össze V-vel. Legyen még egy él U és V között, valamint az egyes $K_{i_1} \dots K_{i_5}$ pontokat kössük össze úgy, hogy egy 5 hosszú kört alkossanak. (Tehát ψ minden zárójelének meg fog felelni egy ötszög.) Ezek után minden olyan ötszög esetén, amely egy $(x_i \vee x_j \vee x_k)$ kifejezésnek felel meg kössük össze három pontját a megfelelő változokhoz tartozó csúcsokkal, a másik két pontját pedig U-val. Legyen a három színünk piros, fehér és zöld. Legyen V színe fehér. Ekkor ψ egy behelyettesítése annak felel meg, hogy a piros csúcsok hamisak a zöldek igazak. Mivel minden $x_1 \dots x_n$ és $\overline{x_1} \dots \overline{x_n}$ össze van kötve V-vel bármely x_i és $\overline{x_i}$ pár esetén az egyikük piros a másikuk zöld, így tehát ha x_i igaz, akkor $\overline{x_i}$ hamis. U nyilván piros vagy zöld. Tegyük fel, hogy piros. Ekkor a kiszínizhetőség pontosan akkor terjesül, ha az ötszögek esetén az U-hoz nem kötött pontok közül legalább az egyik zöld ponthoz van kötve, azaz a „vagy” kapcsolatban lévő változók közül legalább az egyik igaz. Ez amiatt van, mert ellenkező esetben az ötszög semelyik pontját sem színezhetnénk pirosra, vagyis az ötszög pontjaira csak két szín jutna. Ha pedig U zöld lett volna, akkor a zöld szín felelne meg a hamis változóbehelyettesítéseknek és ugyanezt elmonhatnánk zöld színnel. □

5.3. Az 1. Chomsky osztály

Az 1. Chomsky osztályt környezetfüggő nyelvtannal generálható nyelveknek is nevezzük. Az ilyen nyelvtanokban az alábbi típusú szabályok lehetnek:

$\alpha A \beta \to \alpha \gamma \beta$, ahol $\gamma \neq \varepsilon$

$S \to \varepsilon$ ahol S a nyelvtan kezdőváltozója, aki semelyik szabály jobb oldalán nem szerepel

Az ilyen nyelvtanokat a CS rövidítéssel szokták jelölni („Context Sensitive”),

5.15. Állítás. Egy adott G CS nyelvtan és egy $w \in \Sigma^*$ esetén a nemdeterminisztikus Turing gép $|w|$ lineáris tárral eldönti, hogy w benne van-e $L(G)$-ben.

5.15. Bizonyítás. Az egyszeres szabályok kiküszöbölésével elérhető, hogy minden egyes behelyettesítés során a generálandó szó hossza nem csökken, így minden lehetséges behelyettesítést kipróbálva a szó nyelvbe tartozása ellenőrizhető. □

5.16. Tétel (Savitch)**.** Ha $s(n) \geq \log(n)$, akkor $NSPACE(s(n)) \subseteq SPACE(s^2(n))$

5.16. Bizonyítás. A tételt nem bizonyítjuk. □

Ez tehát azt is jelenti, hogy egy szónak egy CS nyelvbe való tartozása a szóhossz négyzetével arányos tárban eldönthető.

5.17. Tétel.

$L \in CS \Leftrightarrow L \in NSPACE(n)$ azaz

$L \in CS \Leftrightarrow \exists M$ lineárisan korlátolt szalagú automata ami felismeri a nyelvet

5.17. Bizonyítás. A tételt nem bizonyítjuk, de megjegyezzük, hogy hasonló képpen lehet nyelvtant létrehozni az automatához mint a 0-ás nyelvosztály esetén. □

5.18. Állítás.

$$L \in CS \Rightarrow L \in R$$

5.18. Bizonyítás. Az 5.15 állítás következménye. □

5.4. Kolmogorov-bonyolultság

Definíció. Legyen $\Sigma = \{0, 1\}$. Az $y \in \Sigma^*$ szónak az M Turing-gépre vonatkozó Kolmogorov-bonyolultsága:

$$C_M(y) = \begin{cases} min\{|x| : f_M(x) = y\} & \text{ha van ilyen } x \\ \infty & \text{ha nincs ilyen } x \end{cases}$$

5.1. Példa. Néhány példa a C_M-re:

- $f_M(x) = x \rightsquigarrow C_M(y) = |y|$
- $f_M(x) = \begin{cases} y & \text{ha } x = \varepsilon \\ x & \text{egyébként} \end{cases} \rightsquigarrow C_M(y) = 0$
- $f_M(x) = 01 \rightsquigarrow C_M(y) = \infty$ ha $y \neq 01$

5.19. Tétel. Legyen M egy tetszőleges Turing-gép és U egy univerzális Turing gép, ekkor:

$$\exists k \text{ konstans, hogy } C_U(y) \leq C_M(y) + k$$

5.19. Bizonyítás. Legyen x olyan, hogy $f_M(x) = y$ és $|x|$ minimális. Ekkor $|x| = C_M(y)$. Legyen w az M Turing-gép kódja. Ekkor U a $w\#y$ bemeneten y-t ad ki. Állítsuk elő $w\#x$ bináris kódját a következő képpen:

- w-ben minden 0 helyett írjunk 00-át
- w-ben minden 1 helyett írjunk 11-et
- A # jel helyett írjunk 01-et.

Ekkor az átkódolt változat hossza:

$$\underbrace{2|w| + 2}_{k} + |x|$$

□

5.20. Állítás. Ha U_1 és U_2 univerzális Turing gépek, akkor:

$$\exists k : |C_{U_1}(y) - C_{U_2}(y)| \leq k$$

5.20. Bizonyítás. Az előző tétel következménye. □

Definíció. Az $y \in \Sigma^*$ szónak a Kolmogorov-bonyolultsága:

$$C(y) = C_U(y) \text{ (valamilyen rögzített } U \text{ univerzális Turing-gépre)}$$

5.2. Példa. Néhány példa a C-re:

- Nézzük meg azon y_n szavak Kolmogorov-bonyolultságát, amelyek n hosszúak és csupa nullából állnak. Egy ilyen szót előállítható egy $M(n) = y_n$ Turing-géppel, amelynek univerzális Turing-géppel való szimulálásához az n bemenetre van szükség. Az n bemenet $\log(n)$ biten leírható, így: $C(y_n) \leq \log(n) + K$.

- Nézzük meg azon n hosszú y_n szavak bonyolultságát, amelyekben pontosan két darab 1-es található. A szó hosszával (n), és a két 1-es pozíciójával (i és j) paraméterezhető egy univerzális Turing gép, amely előállítja ezeket a szavakat. Ekkor a bemenet $n\#i\#j$. Az első két szám esetén a 0-kat és 1-eseket megduplázva, használhatjuk a 01 kódot az elválasztó karakternek. Így: $C(y_n) \leq 5\log(n) + K$.

Definíció. Az $y \in \Sigma^*$ szó *összenyomhatatlan*, ha $C(y) \geq |y|$

5.21. Tétel. Minden n-re létezik n-hosszú összenyomhatatlan szó.

5.21. Bizonyítás. A legfeljebb $(n-1)$-hosszú lehetséges bemenetek száma $2^0+2^1+\cdots+2^{n-1} = 2^n - 1$. Ez alapján viszont létezik olyan n hosszú y szó, amit nem lehet kiszámolni legfeljebb $(n-1)$ hosszú bemenetekből, hiszen az n hosszú leheséges szavak száma 2^n. □

5.22. Tétel. A $C : \Sigma^* \to \Sigma^*$ Kolmogorov-bonyolultság függvény nem rekurzív.

5.22. Bizonyítás. Legyen $F(m) = min\{y : C(y) \geq m\}$. Ekkor $C(F(m)) \geq m$. Ha C rekurzív lenne, akkor $F(m)$ kiszámítható lenne m-ből a következőképpen:

- Vesszük sorban az y-okat
- kiszámoljuk rájuk $C(y)$-t.
- Ha $C(y) > m$, akkor kiírjuk y-t

Mivel m leírható $\log(m)$ bittel ebből az is következne, hogy $C(F(m)) \leq \log(m) + k$, ami ellentmondás, hiszen korábban láttuk, hogy $C(F(m)) \geq m$. □

5.23. Állítás. A palindromok nyelvének felismeréséhez az egyszalagos Turing-gépnek kell legalább cn^2 lépés.

5.23. Bizonyítás. Legyen y egy összenyomhatatlan k hosszú szó és legyen a Turing-gép bemenete $w = y0^k y^R$. Ekkor $|w| = 3|y|$. Legyen $q_j \dots q_{j+p}$ azon állapotok amelyekben a Turing-gép a középső nullák egyikének feldolgozása közben van. Tekintsünk egy M Turing-gépet, amelynek a bemenete a fenti állapotsorozat és hogy melyik állapot melyik 0-áshoz tartozik. Működjön M a kövekezőképpen:

- Vegyünk egy k hosszú z szót.
- A palindrom-felismerő Turing gépet szimuláljuk úgy, mintha a szalag elején z lenne y helyett.
- Ezután figyeljük meg, hogy az első 0-ás elérésekor melyik állapotban vagyunk.
- Ha ez nem q_j, akkor vegyük a következő z szót és kezdjük elölről.
- Ha ez q_j, akkor folytatjuk a szimulációt.

Ha minden $q_j \dots q_{j+p}$ állapot jó volt egy adott z-re, akkor $z = y$, hiszen $z0^k y^R$ és $y0^k y^R$ bemeneten M és a palindrom felismerő gép egyformán viselkedik. Ekkor:

$$C(y) \leq \mathrm{O}(p)$$

Mivel $C(y) \geq |y|$ ebből az is következik, hogy:

$$p \geq \frac{|y|}{K}$$

Ahol K valamilyen konstans. A K-t választhatjuk úgy, hogy az egyenlőtlenség mindegyik középső nulla esetén teljesüljön. Ekkor mind a k darab nullán legalább p-szer járt a Turing-gép:

$$\text{lépések száma } \geq K\frac{|y|}{K} = \Omega(|w|^2)$$

□

Definíció. Egy z végtelen hosszú 0-1 sorozat *Kolmogorov-véletlen* ha:

$$\frac{C(z_n)}{n} \to 1 \text{ ha } (n \to \infty)$$

ahol z_n a z n-hosszú kezdőszelete.

5.24. Tétel. A kövekező két állítás a Kolmogorov-véletlen és az algoritmusok kapcsolatát írja le:

- Ha z algoritmussal előállítható, akkor nem lehet Kolmogorov-véletlen
- Ha z bitjei $\frac{1}{2}$-$\frac{1}{2}$ valószínűséggel 0-ások vagy 1-esek, akkor:

$$P(z \text{ Kolmogorov-véletlen }) = 1$$