Az informatika számítástudományi alapjai
3. előadás

Vaszil György

vaszil.gyorgy@inf.unideb.hu

1. emelet 110-es szoba

Mi is volt a „kontextus”?

- **Automata**: matematikai “gép”
  - a **számítógép hardver** elvi, matematikai leírása
  - a **számítási folyamat** elvi, matematikai leírása
- Formális **nyelv**: jelsorozatok/sztringek/”szavak” sokasága/halmaza
  - az automaták **számítási képességeinek** jellemzése
  - a számítási **feladatok** formális/elvi/matematikai leírása

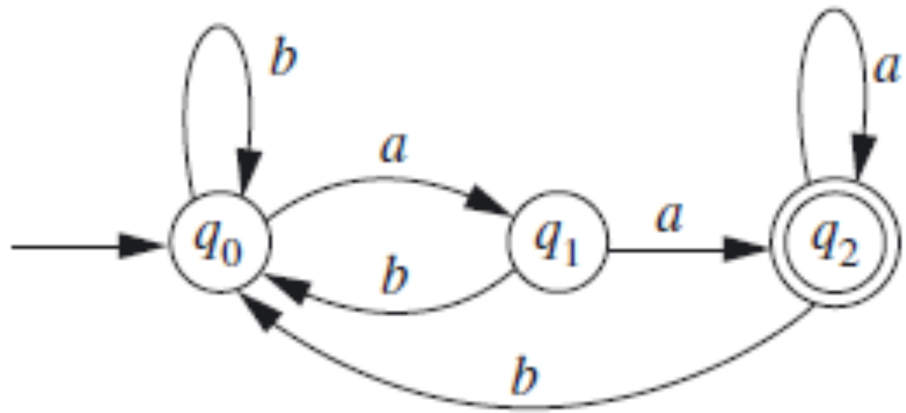
A múlt órán

- Véges automaták, véges automaták által definiált nyelvek (elfogadott nyelvek), példák
- A véges automaták állapotainak számáról
- Nyelvek amiket nem lehet véges automatával elfogadni

Példa

Az ábécé: $V = \{a, b\}$

Az automata:



Milyen nyelvet fogad el?

Azaz:
Milyen szavak viszik a kezdőállapotból a végállapotba?

(Írjuk fel a nyelvet szóhalmazok és múlt órai műveletek segítségével)

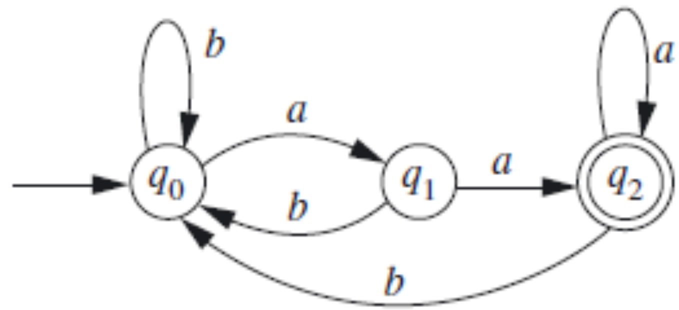
Formális definíció

egy véges automata $M = (Q, \Sigma, q_0, A, \delta)$,

ahol:

- Q : véges állapothalmaz
- Σ : véges bemeneti ábécé
- $q_0 \in Q$: kezdőállapot
- $A \subseteq Q$: végállapotok (elfogadó állapotok)
- $\delta : Q \times \Sigma \to Q$ állapotátmenet függvény

Például



$M = (\{q_0, q_1, q_2\}, \{a, b\}, q_0, \{q_2\}, \delta)$, ahol

δ : a következő:

$$\boxed{\begin{array}{ll} \delta(q_0, a) = q_1 & \delta(q_2, a) = q_2 \\ \delta(q_0, b) = q_0 & \delta(q_2, b) = q_0 \\ \delta(q_1, a) = q_2 & \\ \delta(q_1, b) = q_0 & \end{array}}$$

Az elfogadott nyelv

Legyen $M = (Q, \Sigma, q_0, A, \delta)$, és legyen $x \in \Sigma^*$.

M elfogadja x-et, ha

$$\delta^*(q_0, x) \in A$$

Az M által elfogadott nyelv

$$L(M) = \{ x \in \Sigma^* \mid \delta^*(q_0, x) \in A \}.$$

A múlt órán

- Véges automaták, véges automaták által definiált nyelvek (elfogadott nyelvek), példák
- A véges automaták állapotainak számáról
- Nyelvek amiket nem lehet véges automatával elfogadni

(szünet?)

Hány állapotra an szüksége a véges automatának egy adott nyelv elfogadásához?

<u>Definíció</u>: Két szó **megkülönböztethetősége** egy ***L* nyelvre** nézve:

x és y L-megkülönböztethető, ha létezik z, hogy

$xz \in L$ és $yz \notin L$ vagy $xz \notin L$ és $yz \in L$

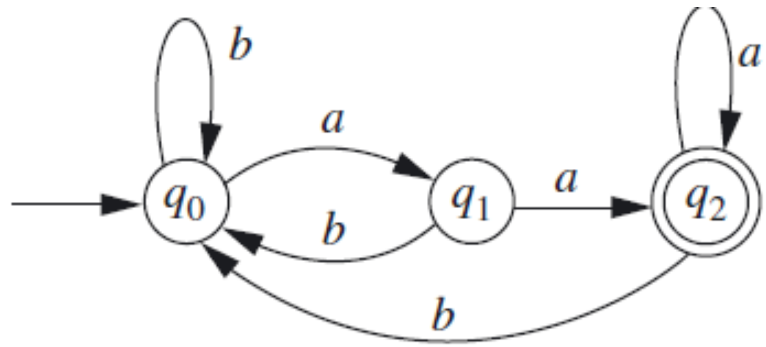
(példa?)

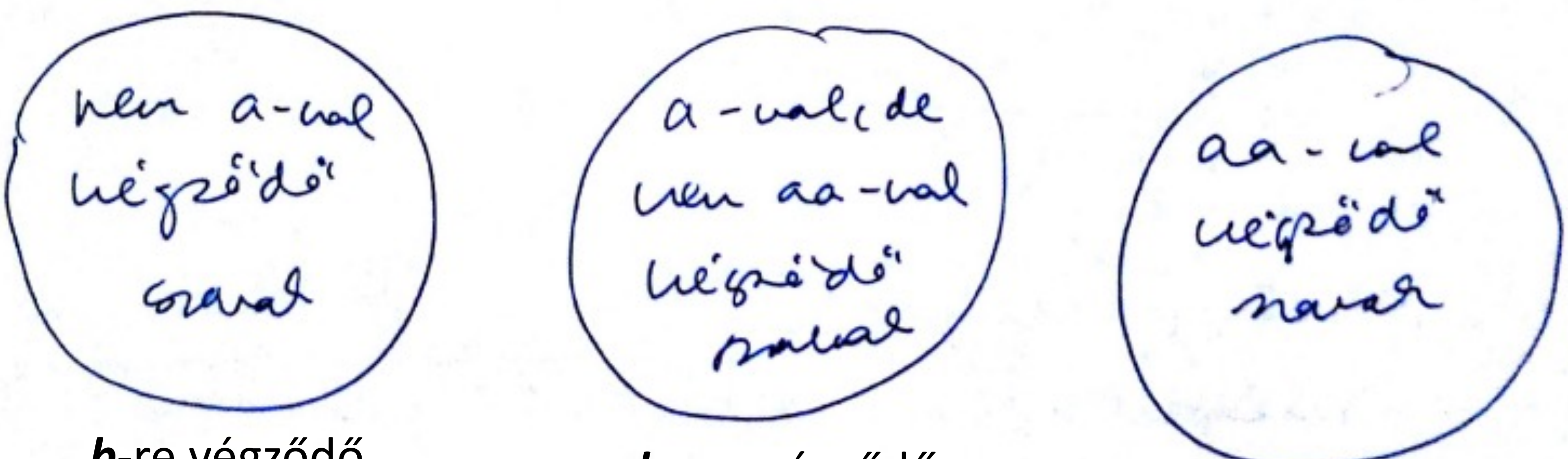
Tanítható

tétel: Ha $L \subseteq \Sigma^*$, és $L = L(M)$ valamilyen M véges automata esetén továbbá Σ^*-ban van n páronként egymástól L-megkülönböztethető szó, akkor M-nek legalább n állapota van.

Az **L nyelvet** elfogadó véges automatának **annyi állapotra** van szüksége, mint az **L szerinti** megkülönböztethetőség **ekvivalencia osztályainak száma**.

A szavak L megkülönböztethetőség alapján 3 csoportba, azaz **3 ekvivalencia osztályba** sorolhatók.





b-re végződő szavak

ba-ra végződő szavak

aa-ra végződő szavak

A múlt órán

- Véges automaták, véges automaták által definiált nyelvek (elfogadott nyelvek), példák
- A véges automaták állapotainak számáról
- Nyelvek amiket nem lehet véges automatával elfogadni

Nyelvek, amelyeket nem lehet véges automatával el-fogadni

tétel: Ha egy L nyelvre nézve végtelen sok páronként L-megkülönböztethető szó van, akkor $L \neq L(M)$ semilyen M véges automatára sem.

Példa: Palindrómák

$L = \{x \in \{a,b\}^* \mid x \text{ palindróma}\}$

{a,b} ábécé felett végtelen sok páronként L megkülönböztethető szó van, például:

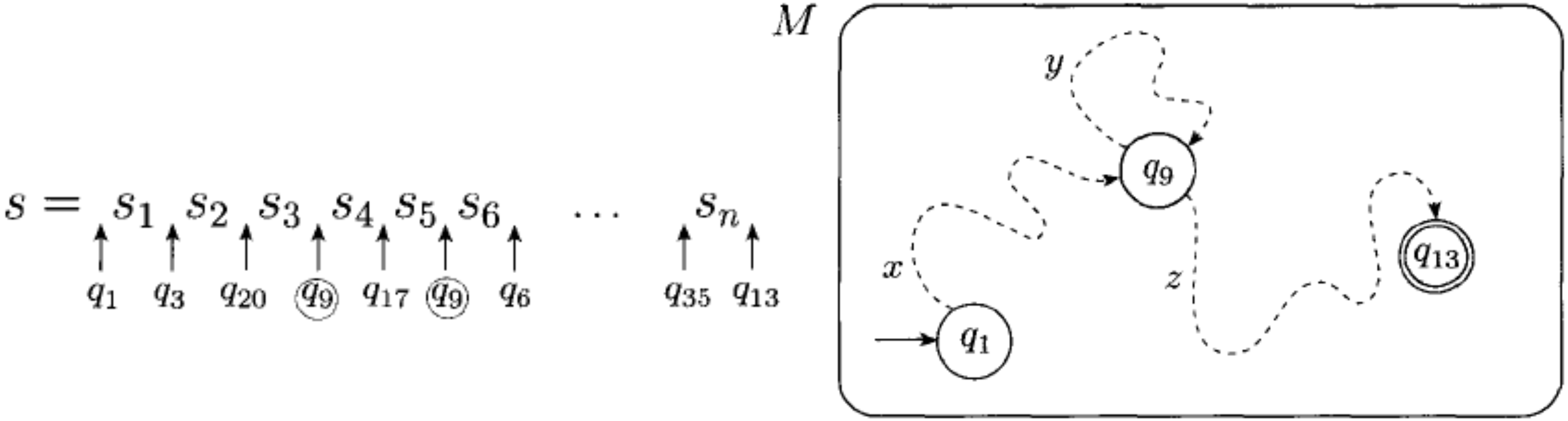
- *ab* és *aab,* mert *ab·baa* nem L-beli*, aab·baa* L-beli
  -->ez 2 osztály: *[ab], [aab]*
- Vegyük most azt, hogy *aaab.* Ez egy harmadik osztály, mert *aaab·baaa L-beli,* de *ab·baaa, aab·baaa* nem L-beli
  -->3 osztály: *[ab], [aab], [aaab]*
- Vegyük most azt, hogy *aaaab.* Ez egy negyedik osztály, mert *aaaab·baaaa,* L-beli, de *ab·baaaa, aab·baaaa, aaab·baaaa* nem L-beli
  --> 4 osztály: *[ab], [aab], [aaab], [aaaab]*
- És így tovább…

Másik példa

$$\{a^n b^n \mid n \geq 0\}$$

Ha $i \neq j$, akkor $a^i b^i \in L$ és $a^j b^i \notin L$.

Azaz $a^i b^i$ *és* $a^j b^i$ megkülönböztethető L= $\{a^n b^n \mid n \geq 0\}$-re nézve (a z=üresszó „megkülönbözteti" őket).



Az automata elfogadja az

$$s_1 s_2 s_3\ {\color{red} s_4 s_5\ s_4 s_5\ \ldots\ s_4 s_5}\ s_6 \ldots s_n$$

alakú szavakat is

Azaz:

Ha $L \subseteq \Sigma^*$ nyelvet elfogadja $M = (Q, \Sigma, q_0, A, \delta)$ véges automata és $n = |Q|$, akkor minden olyan $x \in L$ L-beli szó, amely $|x| \geq n$, felírható

$$x = uvw$$

alakban, ahol:

1. $|uv| \leq n$
2. $|v| > 0$ (azaz $v \neq \lambda$)
3. Minden $i \geq 0$-ra, $uv^iw \in L$

Ha egy nyelv nem teljesíti a pumpálási lemma feltételét, nem lehet véges automatával elfogadni.

Pl.: $\nexists M$, hogy $L = L(M)$, ahol

$L = \{a^i b^i \mid i \geq 1\}$ (M véges automata)

Miért?

És mi a helyzet az {a,b} feletti palindrómákat tartalmazó nyelvvel?

Ezen kívül:

A pumpálási lemma feltételének teljesülése nem elégséges feltétele a nyelvet elfogadó véges automata létezésének.

Például:

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

(Gondolkodni való: tényleg teljesül a pumpálási lemma feltétele? lehet kérdezni is)

<u>Tehát</u> :

Van olyan M véges automata, hogy

- $L = L(M)$ $\Rightarrow$ teljesül a pumpálási tulajdonság

- teljesül a pumpálási tulajdonság $\not\Rightarrow$ Van olyan M véges automata, hogy $L = L(M)$

<u>Ellenben</u> :

- <u>nem</u> teljesül a pumpálási tulajdonság $\Rightarrow L \neq L(M) \; \forall M$
$(\nexists M, L = L(M))$

(„létezik” jel)

A múlt órán

- Véges automaták, véges automaták által definiált nyelvek (elfogadott nyelvek), példák
- A véges automaták állapotainak számáról
- Nyelvek amiket nem lehet véges automatával elfogadni

Figyeljük meg, hogyan jellemzik a nyelvek az automaták képességeit („számítási erejét”)!

A mai órán

- Reguláris kifejezések, reguláris nyelvek
- Nemdeterminisztikus véges automaták

Példák nyelvekre

$V_1 = \{0, 1\}$ felett

- $\emptyset$
- $\{\lambda\}$
- V_1^*
- $L_1 = \{00, 101, 010\}$
- $L_2 = \{\underbrace{0 \dots 0}_{n} \underbrace{1 \dots 1}_{n} \mid n \geq 1\}$
- L_3: a) $\lambda \in L_3$

  b) ha $w \in L_3$, akkor $0w1$, $1w0 \in L_3$

  c) $u \in L_3$ és $v \in L_3$, akkor $uv \in L_3$

  d) L_3-ban nincs más szó, csak amit az a–c szabályok véges számú alkalmazásával kapunk

Reguláris nyelvek

Legyen Σ egy ábécé, a Σ feletti reguláris nyelvek halmaza R, a következő:

1. $\emptyset \in R$
2. $\{a\} \in R$ minden $a \in \Sigma \cup \{\lambda\}$
3. Ha $L_1, L_2 \in R$, akkor

$$L_1 \cup L_2 \in R, \; L_1 \cdot L_2 \in R, \text{ és } L_1^* \in R.$$

Reguláris kifejezések

Reguláris nyelv	Reguláris kifejezés
$\emptyset$	$\emptyset$
$\{\Lambda\}$	Λ
$\{a, b\}^*$	$(a+b)^*$
$\{aab\}^*\{a, ab\}$	$(aab)^*(a+ab)$
$(\{aa, bb\} \cup \{ab, ba\}\{aa, bb\}^*\{ab, ba\})^*$	$(aa+bb+(ab+ba)(aa+bb)^*(ab+ba))^*$

Reguláris kifejezések

1. a , ahol $a \in \Sigma \cup \{\lambda\}$ $\leftrightarrow$ $\{a\}$
2. $\emptyset$ $\leftrightarrow$ $\emptyset$
3. $R_1 + R_2$, ahol R_1 és R_2 reguláris kifejezés.
   (Néha $R_1 \cup R_2$) $\leftrightarrow$ (R_1 nyelv)∪(R_2 nyelv)
4. $R_1 R_2$, ahol R_1 és R_2 reg. kif.
   $\leftrightarrow$ (R_1 nyelv) · (R_2 nyelv)
5. R^* , ahol R reg. kif.
   $\leftrightarrow$ (R nyelv) *

Példák

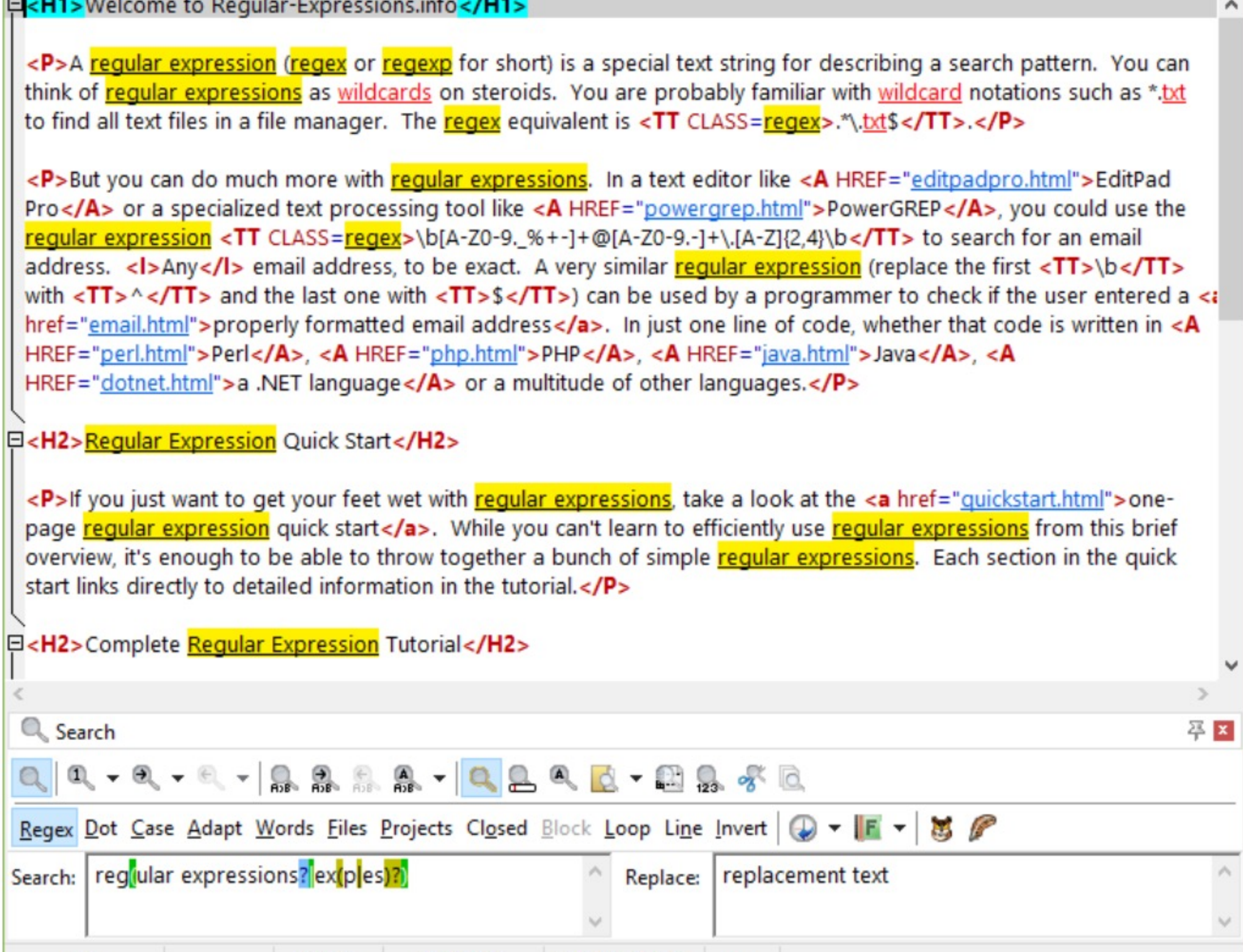
$\Sigma = \{a, b\}$ feletti nyelv, ahol az "a" betűk száma páratlan.

$$b^* a b^* (b^* a b^* a b^*)^*$$

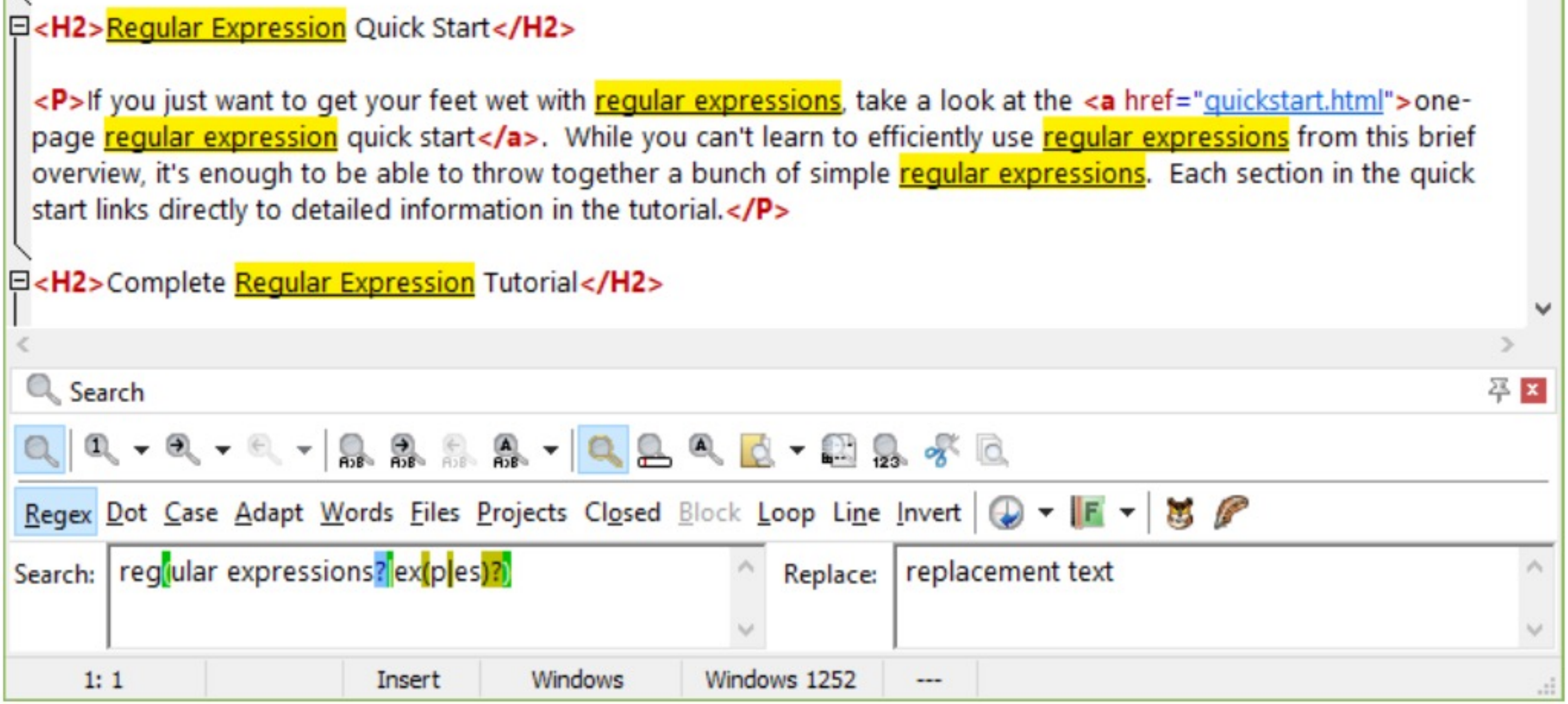
(Lehetne másképp felírni?) $b^* a (b + a b^* a)^*$

Reguláris kifejezések a „mindennapokban” - egyszerűsített jelölés és „komoly” jelölés

- *gr[ae]y* ← *gray, grey* → *gr(a+e)y*
- *gr . y* ← *gray, grby, grcy, ...* → *gr (a+b+c+...+w) y*
- *cat|dog* ← *cat, dog* → *cat + dog*
- *colou?r* ← *color, colour* → *colo(u+ λ)r*
- *[123..9][012..9]** ← *számok* → *(1+2+..+9)(0+1+..+9)**
- *[123..9][012..9]+* ← *többjegyű számok* → *(1+2+..+9)(0+1+..+9)*$^{+}$
- *Set(Value)?* ← *Set, SetValue* → *Set(Value+ λ)*



<H1>Welcome to Regular-Expressions.info</H1>
<P>A regular expression (regex or regexp for short) is a special text string for describing a search pattern. You can think of regular expressions as wildcards on steroids. You are probably familiar with wildcard notations such as *.txt to find all text files in a file manager. The regex equivalent is <TT CLASS=regex>.*\.txt$</TT>.</P>
<P>But you can do much more with regular expressions. In a text editor like <A HREF="editpadpro.html">EditPad Pro</A> or a specialized text processing tool like <A HREF="powergrep.html">PowerGREP</A>, you could use the regular expression <TT CLASS=regex>\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b</TT> to search for an email address. <I>Any</I> email address, to be exact. A very similar regular expression (replace the first <TT>\b</TT> with <TT>^</TT> and the last one with <TT>$</TT>) can be used by a programmer to check if the user entered a properly formatted email address. In just one line of code, whether that code is written in <A HREF="perl.html">Perl</A>, <A HREF="php.html">PHP</A>, <A HREF="java.html">Java</A>, <A HREF="dotnet.html">a .NET language</A> or a multitude of other languages.</P>
<H2>Regular Expression Quick Start</H2>
<P>If you just want to get your feet wet with regular expressions, take a look at the one-page regular expression quick start. While you can't learn to efficiently use regular expressions from this brief overview, it's enough to be able to throw together a bunch of simple regular expressions. Each section in the quick start links directly to detailed information in the tutorial.</P>
<H2>Complete Regular Expression Tutorial</H2>
Search
Regex Dot Case Adapt Words Files Projects Closed Block Loop Line Invert
Search: reg(ular expressions?|ex(p|es)?)
Replace: replacement text



reg(ular_expressions?|ex(p|es)?)

← regular_expression, regular_expressions, regexp, regexes, regex →

reg (ular_expression (s+ λ) + ex (p+es+ λ))

Például: a JAVA-ban...

```
public class StringMatcher {
    // returns true if the string matches exactly "true"
    public boolean isTrue(String s){
        return s.matches("true");
    }
    // returns true if the string matches exactly "true" or "True"
    public boolean isTrueVersion2(String s){
        return s.matches("[tT]rue");
    }

    // returns true if the string matches exactly "true" or "True"
    // or "yes" or "Yes"
    public boolean isTrueOrYes(String s){
        return s.matches("[tT]rue|[yY]es");
    }

    // returns true if the string contains exactly "true"
    public boolean containsTrue(String s){
        return s.matches(".*true.*");
    }
```

```
    // returns true if the string contains of three letters
    public boolean isThreeLetters(String s){
        return s.matches("[a-zA-Z]{3}");
        // simpler from for
//      return s.matches("[a-Z][a-Z][a-Z]");
```

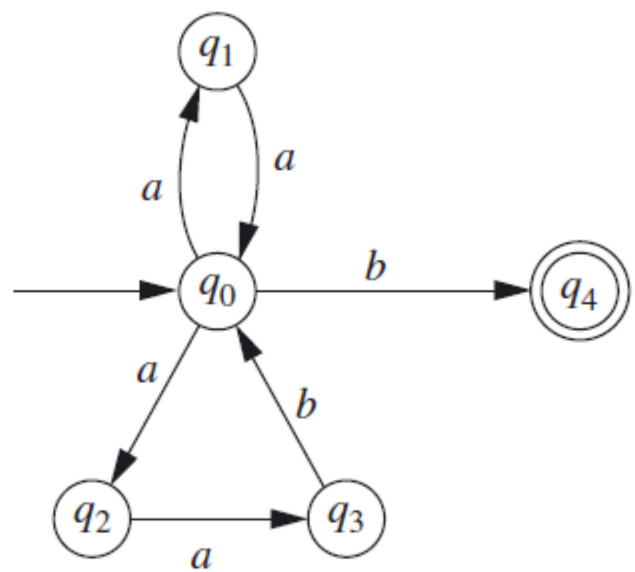
```
    // returns true if the string does not have a number at the beginning
    public boolean isNoNumberAtBeginning(String s){
        return s.matches("^[^\\d].*");
    }
    // returns true if the string contains a arbitrary number of characters
except b
    public boolean isIntersection(String s){
        return s.matches("([\\w&&[^b]])*");
    }
    // returns true if the string contains a number less than 300
    public boolean isLessThenThreeHundred(String s){
        return s.matches("[^0-9]*[12]?[0-9]{1,2}[^0-9]*");
    }

}
```

A mai órán

- Reguláris kifejezések, reguláris nyelvek
- Nemdeterminisztikus véges automaták

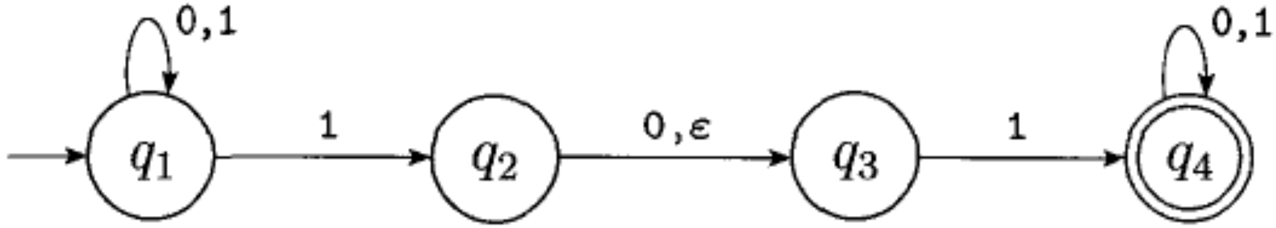
„Lassítsunk" a véges automata definícióján

$L = \{aa, aab\}^*\{b\}$ vagy $(aa + aab)^*b$



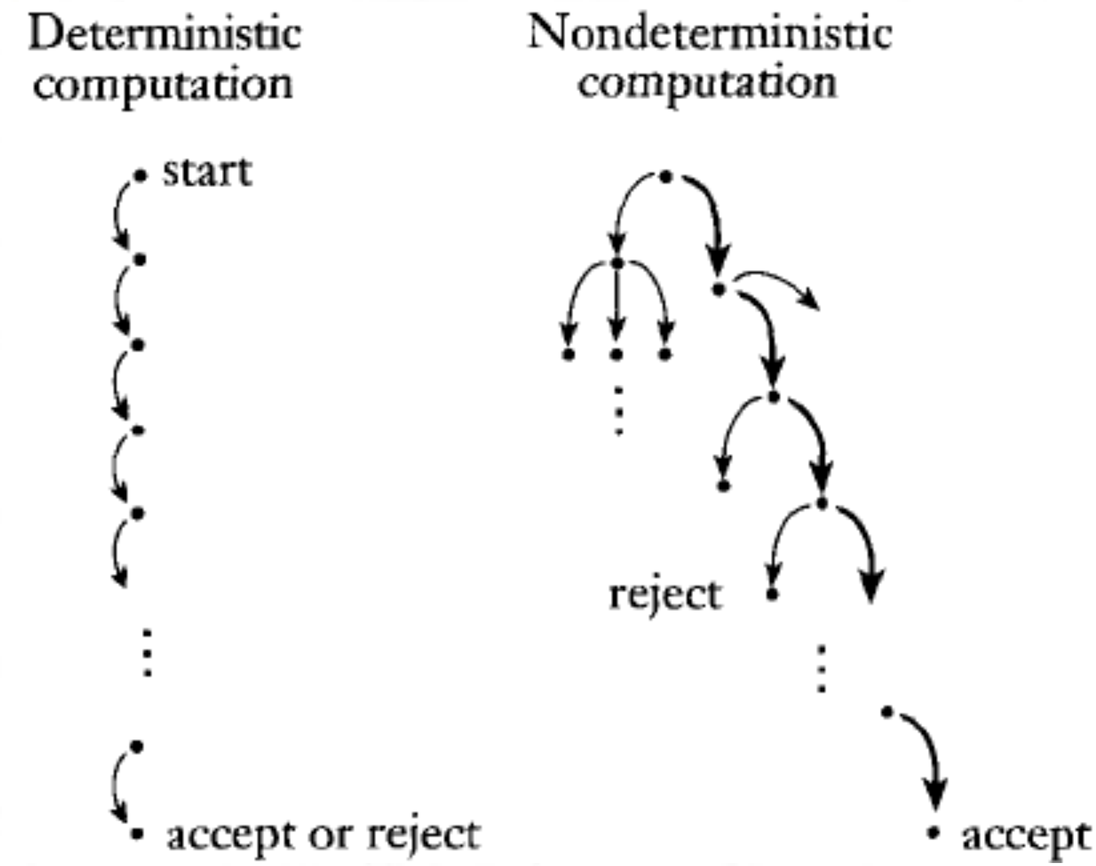
Mit jelentene aabb elolvasása?

Nemdeterminisztikus véges automaták



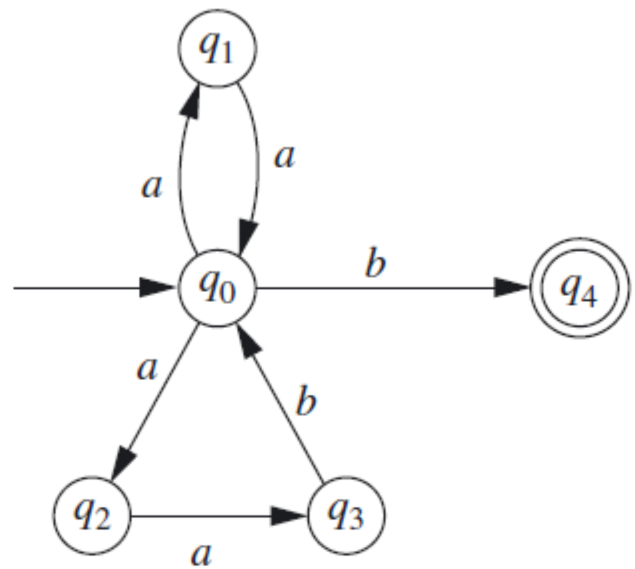
- több lehetőség ugyanarra a bemenő jelre
- üresszó átmenet (ε vagy λ)

Nemdeterminisztikus számítás



Elfogadja a szót, ha létezik olyan ~~...~~ „lehetőség" amivel elfogadó állapotba jut.

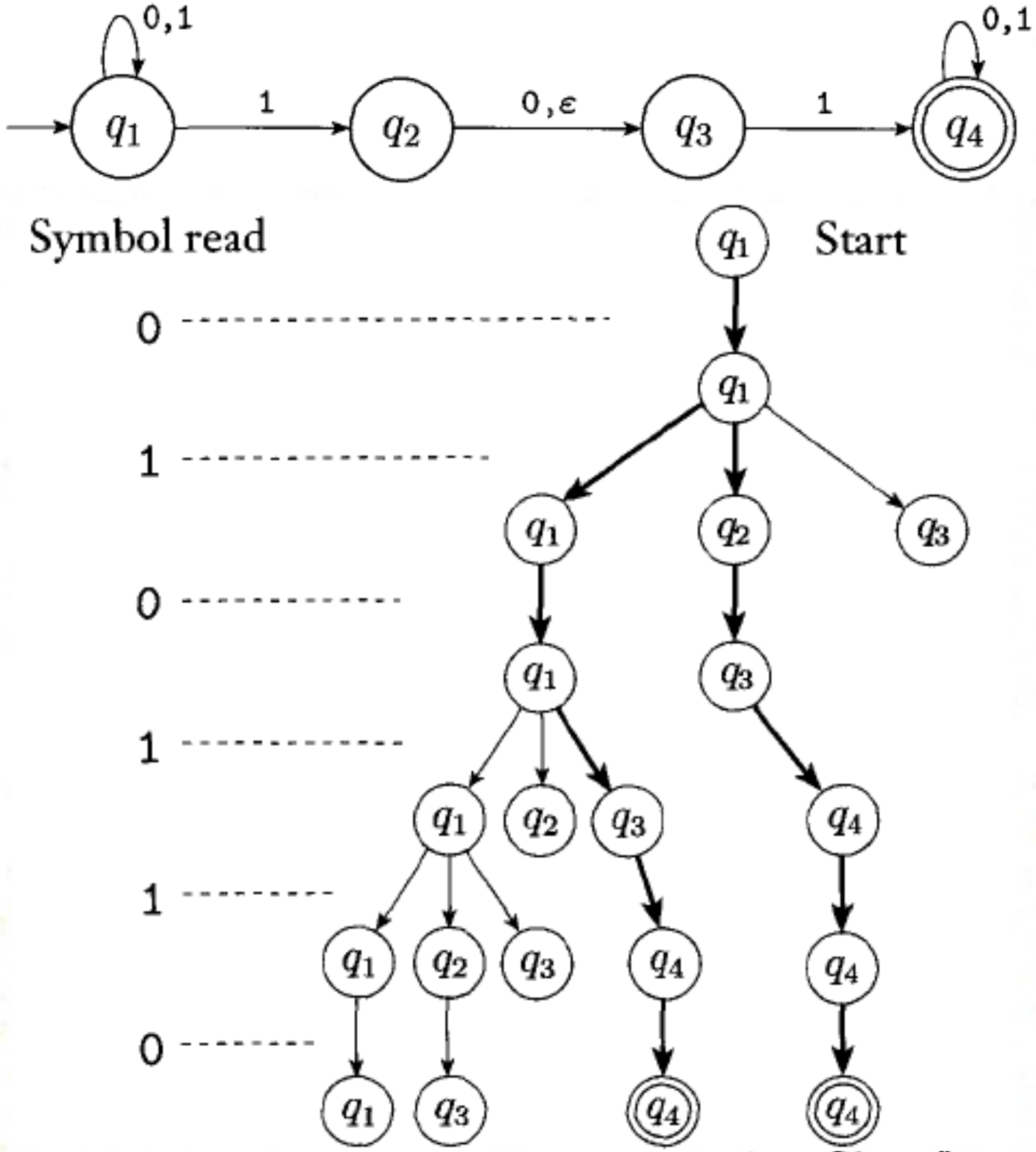
$L = \{aa, aab\}^*\{b\}$ vagy $(aa + aab)^*b$



Mit jelentenek aabb elolvasások?

Az automata elfogadja az *aabb* szót.

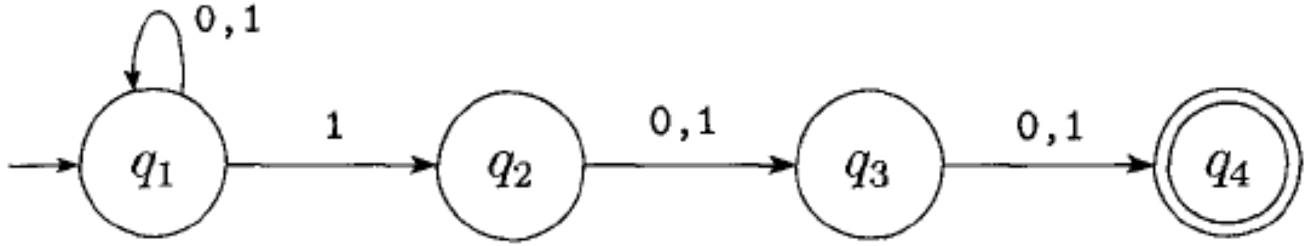
Nemdeterminisztikus számítás



Bemenet: 010110 – (elfogadja)

Példa: Mit fogad el ?

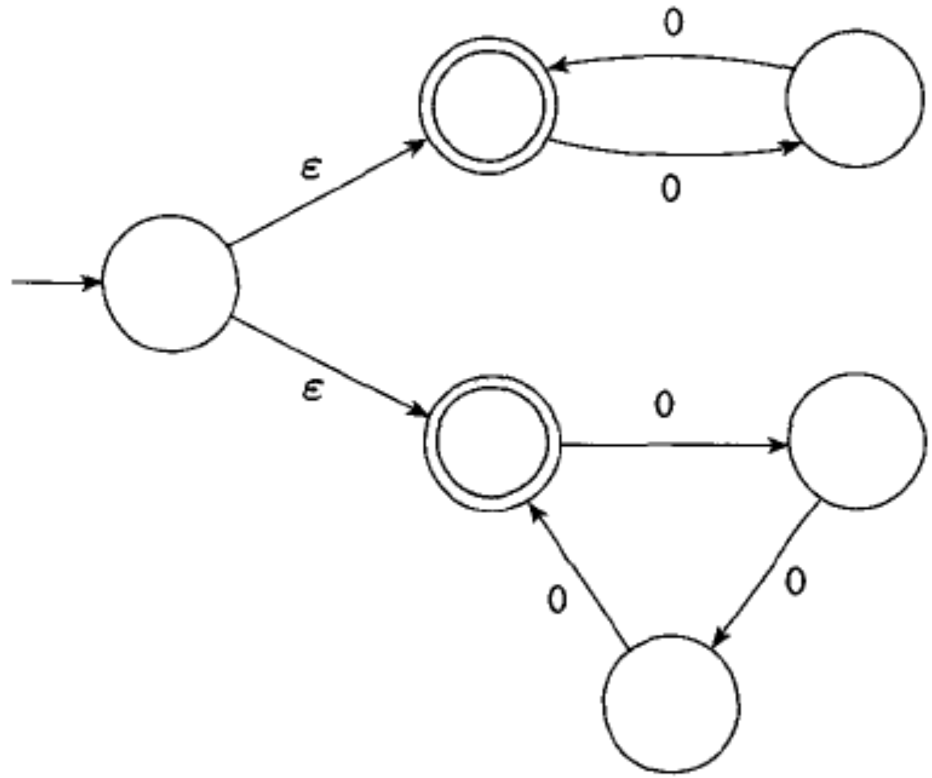
M:



$L(M) = ?$

Másik példa

M :



L(M) = ?

Nemdeterminisztikus véges automata

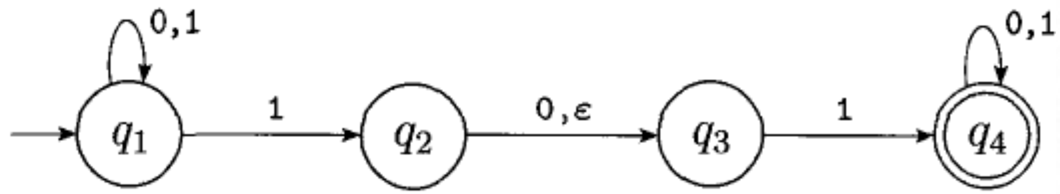
$M = (Q, \Sigma, q_0, A, \delta)$, ahol Q, Σ, q_0, A mint a determinisztikus esetben, és

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \to 2^{Q}$$

Q részhalmazai

Például

M:

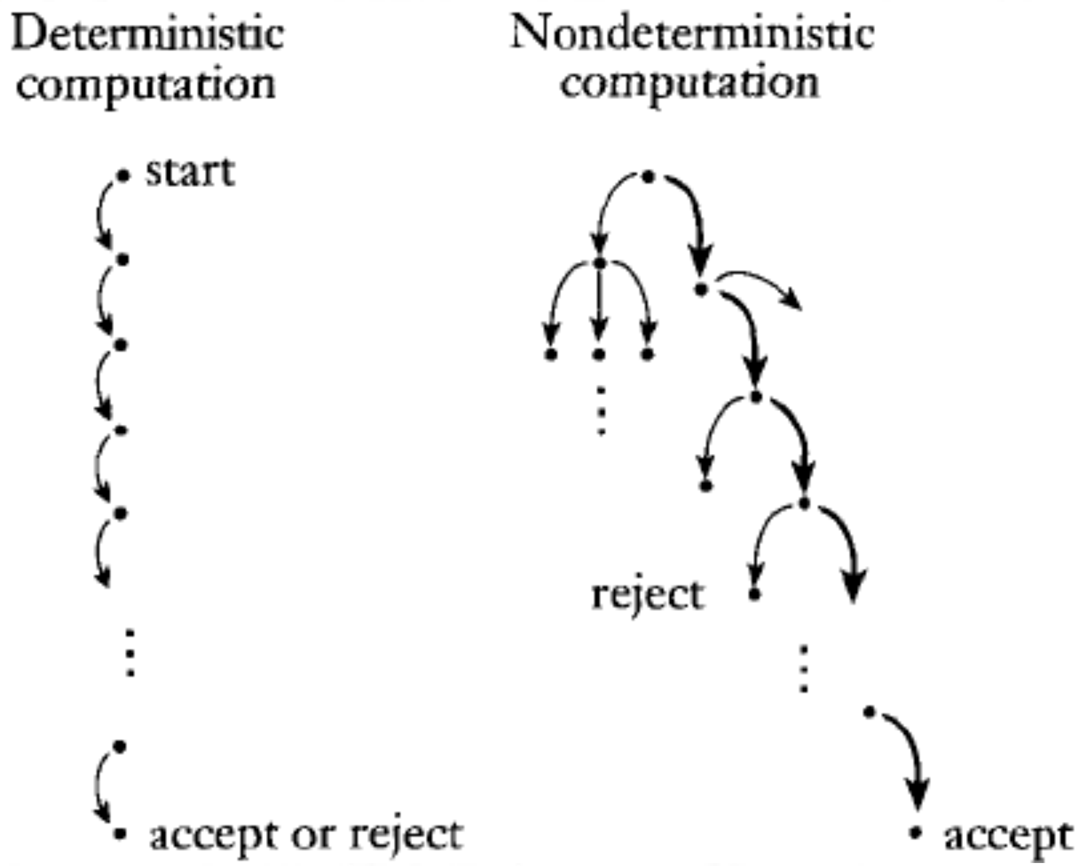


$M = (Q, \Sigma, q_1, A, \delta)$, ahol

- $Q = \{q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0, 1\}$
- $A = \{q_4\}$
- δ :

	0	1	ε
q_1	q_1	q_1, q_2	
q_2	q_3		q_3
q_3		q_4	
q_4	q_4	q_4	

Nemdeterminisztikus számítás



Elfogadja a szót, ha létezik olyan „lehetőség” amivel elfogadó állapotba jut.

(Az elfogadott szavakat szeretnénk formálisan felírni)

Definíció

$M = (Q, \Sigma, q_0, A, \delta)$ nemdet. véges automata.

Legyen $S \subseteq Q$ és jelölje $E(S)$ a következőt:

- $S \subseteq E(S)$
- Minden $q \in E(S)$-re $\delta(q, \lambda) \subseteq E(S)$

$E(S) \subseteq Q$ az az állapothalmaz, amibe S állapotaiból λ átmenetek mentén el lehet jutni.

Definíció

$M = (Q, \Sigma, q_0, A, \delta)$, legyen

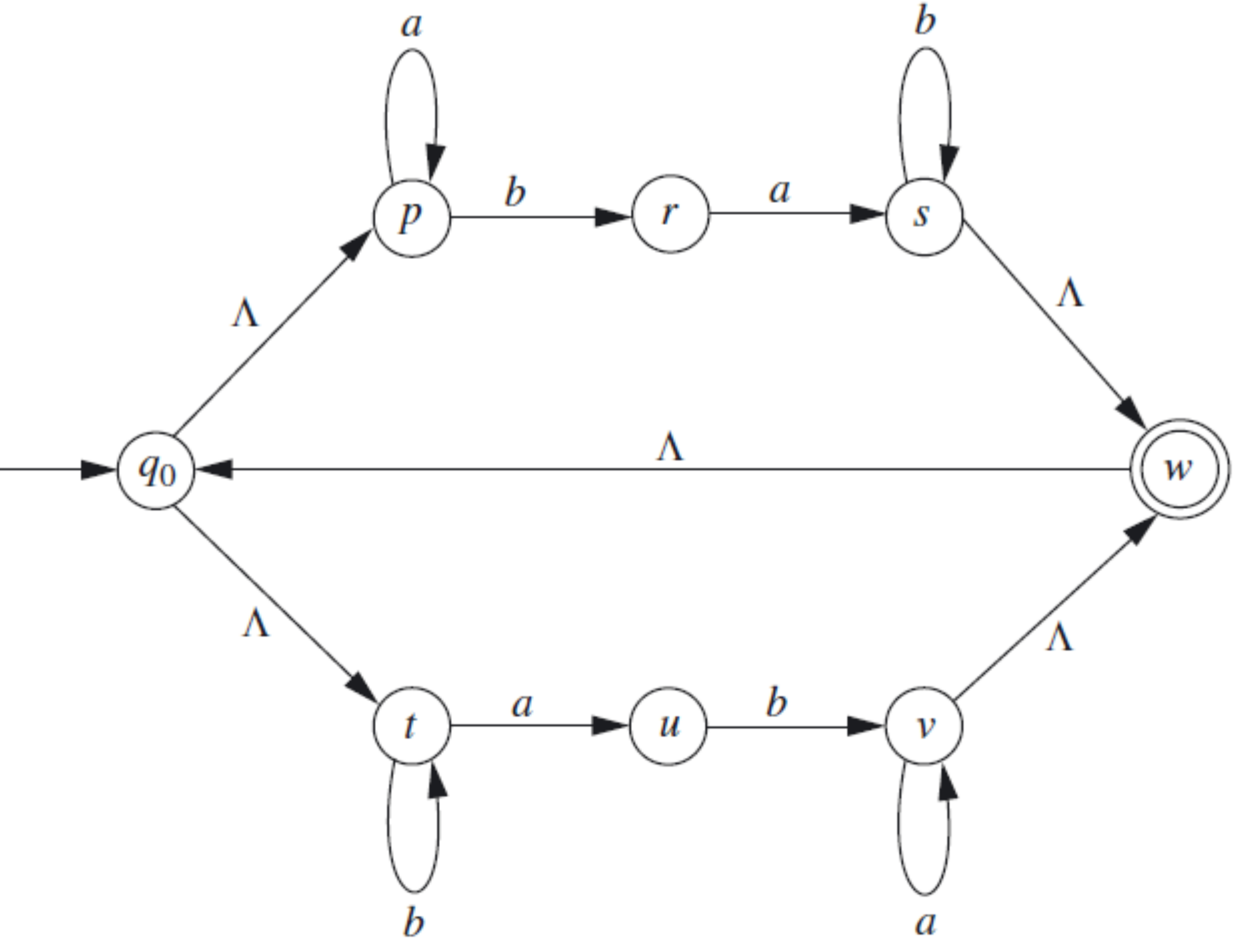
$$\delta^*: Q \times \Sigma^* \to 2^Q$$

a következő:

- $\delta^*(q, \lambda) = E(\{q\})$ minden $q \in Q$-ra
- $\delta^*(q, xa) = E\left(\bigcup \{\delta(p, a) \mid p \in \delta^*(q, x)\}\right)$

(Azaz az állapotok, ahova a w szó elolvasása viheti az automatát q-ból indulva $\delta^*(q, w)$.)

például

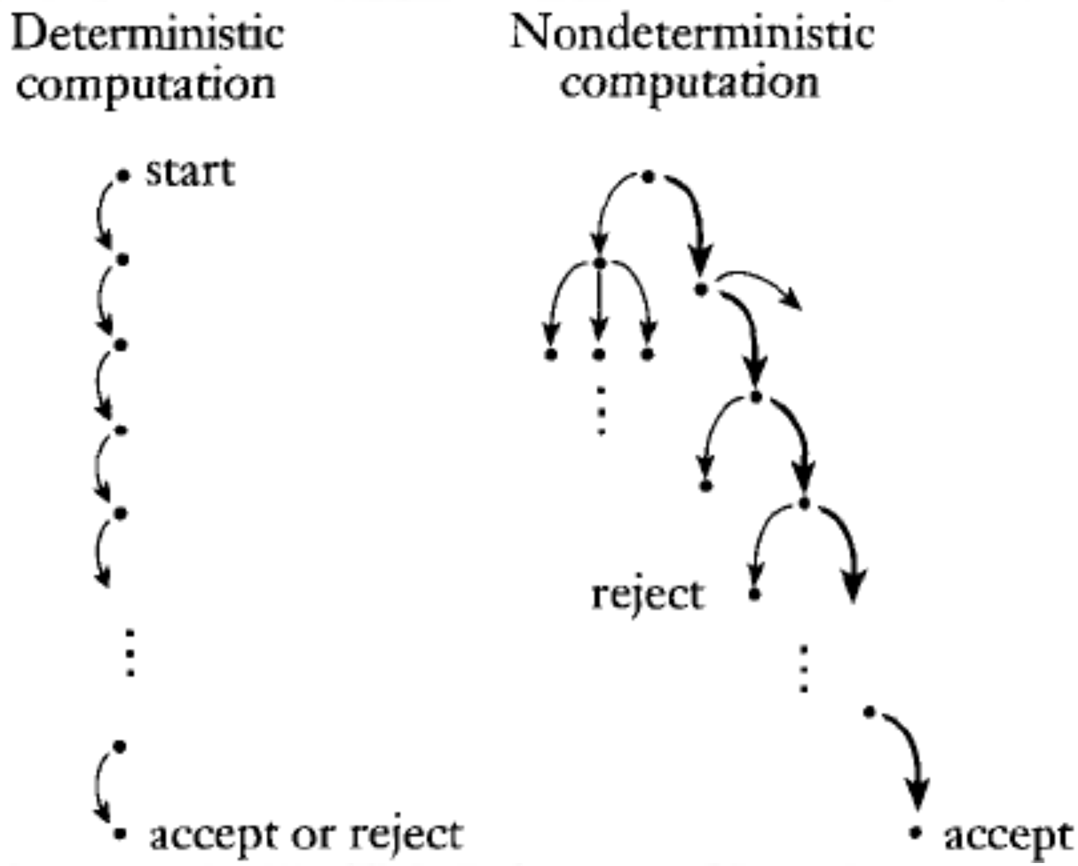


$$\delta^*(q_0,\lambda) = E(\{q_0\}) =$$
$$= \{q_0, p, t\}$$

$$\delta^*(q_0, ab) =$$

$$= \{r, v, w, q0, p, t\}$$

Nemdeterminisztikus számítás



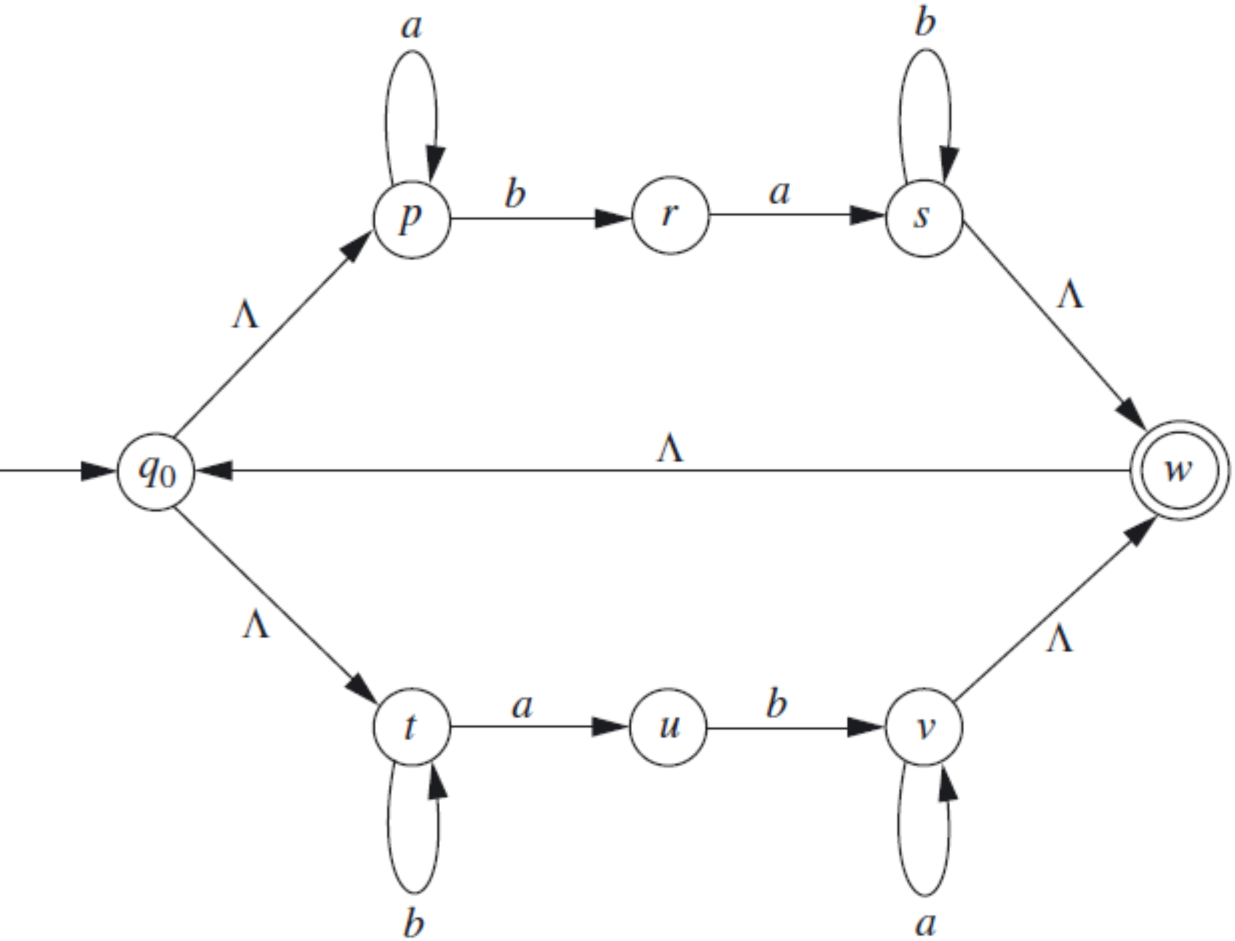
Elfogadja a szót, ha létezik olyan „lehetőség" amivel elfogadó állapotba jut.

Az elfogadott nyelv

$M = (Q, \Sigma, q_0, A, \delta)$ által elfogadott nyelv $L(M)$ a következő:

$$L(M) = \{ w \in \Sigma^* \mid \delta^*(q_0, w) \cap A \neq \emptyset \}.$$

például



$$\delta^*(q_0, \lambda) = E(\{q_0\}) =$$
$$= \{q_0, p, t\}$$

$$\delta^*(q_0, ab) =$$

= {*r, v, w, q0, p, t*}

Az *ab* szó benne van az elfogadott nyelvben?

A mai órán

- Reguláris kifejezések, reguláris nyelvek
- Nemdeterminisztikus véges automaták

A könyvekben

Reguláris kifejezések:

- **J. Martin: 3.1 fejezet, 92 – 96. oldal**
- **Dömösi et al.: 5.2 fejezet, 77 – 80. oldal**

Nemdeterminisztikus véges automaták:

- **J. Martin: 3.2 fejezet, 96 – 104. oldal**
- **Dömösi et al.: 5.4 fejezet, 83 – 86. oldal**