Formalne metode
u softverskom inženjerstvu

06 Regularni jezici

ETFBL 24-25

Dunja Vrbaški

Teorija formalnih jezika i automata

- Definisali smo formalni jezik: podskup reči nad nekim alfabetom

- Definisali smo konačne automate
  (ekvivalencija eNKA, NKA, DKA; minimizacija, zatvorenost)

- Koja je veza između konačnih automata i nekih jezika?
- Kakvi jezici odgovaraju konačnim automatima?
- Šta znači ”odgovaraju”?

- Kakvi još jezici postoje?
- Kakvi automati još postoje?
- Kakve to veze ima sa računarima i programiranjem?

Kako izgleda automat koji prepoznaje palindrome?

Konačni automati prihvataju regularne jezike.

- jezik u kom sve reči imaju neparan broj jedinica
  - imamo DKA
  - regularan jezik

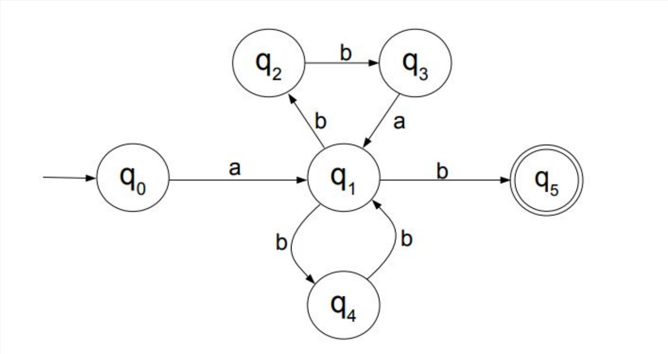
- jezik u kom su svi palindromi
  - ne može se konstruisati DKA
  - nije regularan jezik

Regularni izrazi

- Regularni izrazi služe za definisanje jezika, za opis reči

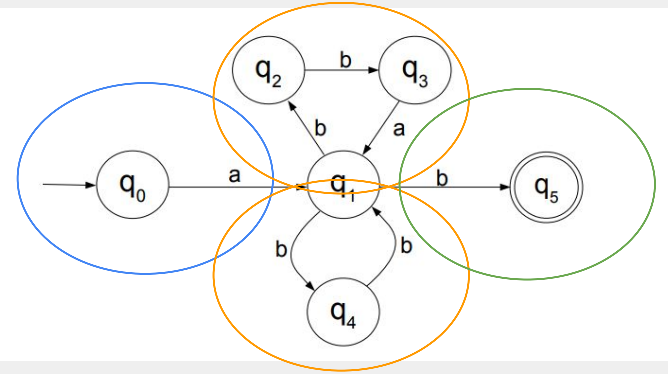
- Ispostavlja se da upravo definišu regularne jezike
- RE $\leftrightarrow$ DKA

- DKA prepoznaje jezik, prihvata ili ne prihvata reči
- RE su deklarativan način za opis jezika, mehanizam za građenje reči



a(bba|bb)*b

regularni izraz



Regularni izrazi i jezici koje ovi izrazi definišu nad nekim alfabetom Σ se definišu rekurzivno:

- **∅** je regularni izraz i označava jezik L(∅) = {},
- **ε** je regularni izraz i označava jezik L(ε) = {ε},
- za svaki simbol **a** ∈ Σ, a je regularni izraz i označava jezik L(a) = {a},
- ako su r i s regularni izrazi koji označavaju jezike L(r) i L(s), onda važe sledeće definicije:
    - **r | s** je regularni izraz koji označava jezik L(r | s) = L(r) ∪ L(s) (često se koristi i operator +)
    - **rs** je regularni izraz koji označava jezik L(rs) = L(r) L(s)
    - **r*** je regularni izraz koji označava jezik L(r*) = L(r)*

Regularni izrazi i jezici koje ovi izrazi definišu nad nekim alfabetom Σ se definišu rekurzivno:

- ∅ je regularni izraz i označava jezik L(∅) = {},
- **ε** je regularni izraz i označava jezik L(ε) = {ε},
- za svaki simbol **a** ∈ Σ, a je regularni izraz i označava jezik L(a) = {a},
- ako su r i s regularni izrazi koji označavaju jezike L(r) i L(s), onda važe sledeće definicije:
  - **r | s** je regularni izraz koji označava jezik L(r | s) = L(r) ∪ L(s) (često se koristi i operator +) — alternative
  - **rs** je regularni izraz koji označava jezik L(rs) = L(r) L(s) — spajanje
  - **r*** je regularni izraz koji označava jezik L(r*) = L(r)* — ponavljanje

Primer: a(bba|bb)*b

- unarni operator * je levo asocijativan i ima najveći prioritet
  r * s * t = (r * s) * t

- operator konkatenacije je levo asocijativan i ima veći prioritet od operatora |

- operator | je levo asocijativan i ima najmanji prioritet

01*|1	0 1* | 1
(01)*|1	(01)* | 1
a|bc*d	a | b c* d

$L(a|b) = \{a, b\}$

$L((a|b)(a|b)) = \{aa, ab, ba, bb\}$

$L(a^*) = \{\varepsilon, a, aa, aaa, aaaa, \ldots\}$

$L(a^*b^*) = \{\varepsilon, a, b, ab, aab, abbbb, aaaaa, \ldots\}$

$L((a|b)^*) = \{\varepsilon, a, b, ab, ba, aabb, bbaa, \ldots\}$

$L((a|b)a(a|b))$ - jezik čije reči sadrže bar jedan simbol a

$L(a(a|b)^*b)$ - jezik čije reči počinju simbolom a i završavaju simbolom b

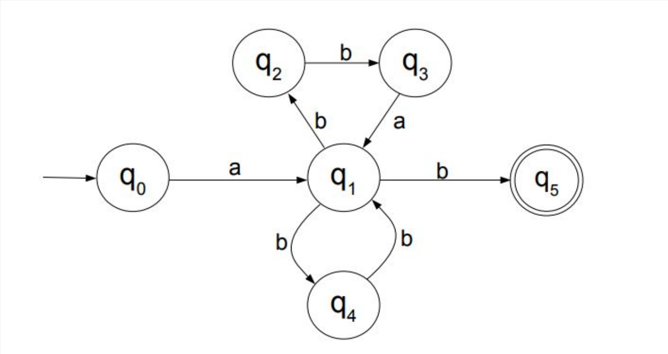
Zadatak: Napisati regularan izraz koji definiše jezik čije su reči sačinjene od naizmeničnog pojavljivanja 0 i 1

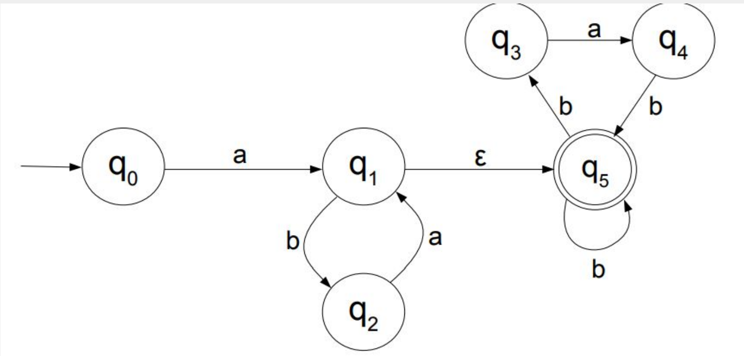
{ε, 0, 1, 10, 01, 101, 010, 1010, 0101, 10101, 01010,...}

$r \mid s = s \mid r$
$r \mid \varnothing = r = \varnothing \mid r$
$r \mid r = r$
$(r \mid s) \mid t = r \mid (s \mid t)$

$r\varepsilon = \varepsilon r = r$
$r\varnothing = \varnothing r = \varnothing$

$(rs)t = r(st)$
$r(s \mid t) = rs \mid rt$
$(r \mid s)t = rt \mid st$





a(bba|bb)*b

a(ba)*(b|bab)*

Formalno, za regularne izraze definišemo samo tri operacije: |, $\cdot$ i *

Postoje proširenja radi jednostavnijeg zapisa koja se koriste u različitim bibliotekama i alatima.

Neki zapisi imaju svoju ekvivalentnu formu
$R^+ = RR^*$

POSIX (IEEE) standard: BRE (Basic) i ERE (Extended)

Obratiti pažnju prilikom korišćenja biblioteke

```
^[\s]*(.*?)[\s]*$
```

trim whitespace (početak i kraj)

```
<([a-z]+)([^<]+)*(?:>(.*)<\/\1>|\s+\/>)
```

HTML element

```
\B#(?:[a-fA-F0-9]{6}|[a-fA-F0-9]{3})\b
```

boja heksadecimalno

```
\b[\w.!#$%&'*+/=?^`{|}~\-]+@[\w\-]+(?:\.[\w\-]+)*\b
```

email
dovoljno dobro?

Oznaka	Primer	Formiranje	Skup reči
		Prazan string	
	a	Svaki simbol (karakter) je RE	
	ab	Vrši se konkatenacija dva RE	“ab”
*	ab*	RE na koji se odnosi se ponavlja 0 ili više puta	“a”, “ab”, “abb”,....
+	ab+	RE na koji se odnosi se ponavlja 1 ili više puta	“ab”, “abb”,...
?	ab?	RE na koji se odnosi se ponavlja 0 ili 1 put	“a”, “ab”

<table>
<tr><th>Oznaka</th><th>Primer</th><th>Formiranje</th><th>Skup reči</th></tr>
<tr><td>|</td><td>a|b</td><td>Alternativa</td><td>“a”, “b”</td></tr>
<tr><td>( )</td><td>a(b|c)</td><td>Grupisanje</td><td>“ab”, “ac”</td></tr>
<tr><td>[ ]</td><td>[abc]</td><td>Alternative</td><td>“a”, “b”, “c”</td></tr>
<tr><td>[-]</td><td>[a-c]</td><td>Opseg</td><td>“a”, “b”, “c”</td></tr>
<tr><td>[^]</td><td>[^abc]</td><td>Alternative koje ne odgovaraju navedenim</td><td>“d”, “e”,….(ostalo iz azbuke)</td></tr>
<tr><td>{m, n}</td><td>a{1, 3}</td><td>Broj ponavljanja</td><td>“a”, “aa”, “aaa”</td></tr>
<tr><td>.</td><td>a.</td><td>Bilo koji znak (osim nove linije)
[a.b] – tu se često interpetira baš kao znak “.”</td><td>“aa”, “ab”, “ac”…</td></tr>
<tr><td>\</td><td>a\.</td><td>Escape, specijalne karaktere tretiramo kao obične</td><td>“a.”</td></tr>
</table>

```java
String[] tekst = {    "Marko Markovic +387(65)000-000", "Janko Jankovic +387(51)111-111",
                      "Jovan Jovanovic +387(51)222-222", "Nenad Nenadovic +387(65)333-333",
                      "Marko Jankovic +387(65)444-444", "ABC", "AABBCCC dodatni tekst", "AAABBBCC", "tekst123",
                      "office@etf.unibl.org", "nevalidan@etf.tv" };

String[] regularniIzrazi = { "tekst", "tekst$", ".+[A-Za-z]{3}ko", "^[A-Za-z]{2}ko ", "^[A-Za-z]{3}ko ",
                             "(Jan)+", "(Jan(.{3})){2}", "\\+387\\(51\\)[0-9]{3}-[0-9]{3}",
                             "[a-z]{1}[a-z0-9]*@([a-z0-9]+\\.)+[a-z]{3}$" };

for (String regex : regularniIzrazi) {
   Pattern pattern = Pattern.compile(regex);
      for (String ulaz : tekst) {
         Matcher matcher = pattern.matcher(ulaz);
         if (matcher.find())
            System.out.println("Izraz " + regex + " prihvata sekvencu: " + matcher.group() + " u ulazu " + ulaz);
         System.out.println();
      }
}
```

java

```csharp
string pattern = "(Mr\\.? |Mrs\\.? |Miss |Ms\\.? )";
string[] names = { "Mr. Henry Hunt", "Ms. Sara Samuels",
                   "Abraham Adams", "Ms. Nicole Norris" };

foreach (string name in names)
    Console.WriteLine(Regex.Replace(name, pattern, String.Empty));

//

Regex rx = new Regex(@"\b(?<word>\w+)\s+(\k<word>)\b",
        RegexOptions.Compiled | RegexOptions.IgnoreCase);

        // Define a test string.
        string text = "The the quick brown fox  fox jumps over the lazy dog dog.";

        // Find matches.
        MatchCollection matches = rx.Matches(text);

        // Report the number of matches found.
        Console.WriteLine("{0} matches found in:\n   {1}",
                          matches.Count,
                          text);

        // Report on each match.
        foreach (Match match in matches)
        {
            GroupCollection groups = match.Groups;
            Console.WriteLine("'{0}' repeated at positions {1} and {2}",
                              groups["word"].Value,
                              groups[0].Index,
                              groups[1].Index);
        }
```

C#

```bash
#!/bin/bash

echo ispisuje sve redove koji sadrze rijec \"tekst\"
grep --color -E "tekst" tekst.txt
echo -e "\n"

echo ispisuje sve redove kod kojih se rijec \"tekst\" nalazi na kraju reda
grep --color -E "tekst$" tekst.txt
echo -e "\n"


echo ispisuje sve redove koji sadrze fiksni broj telefona iz Banjaluke u formatu
+387\(51\)xxx-xxx
grep --color -E "\+387\(51\)[0-9]{3}-[0-9]{3}" tekst.txt
echo -e "\n"
```

Linux, grep

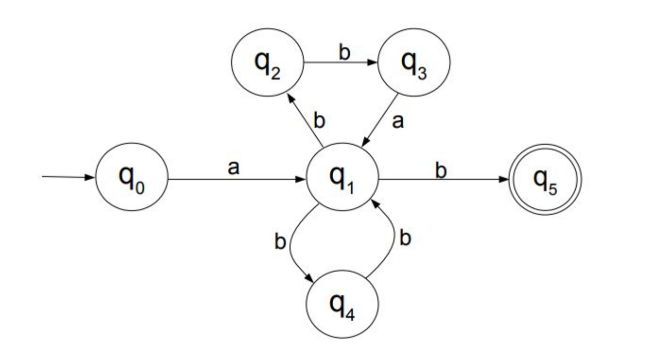
Globally search a Regular Expression and Print

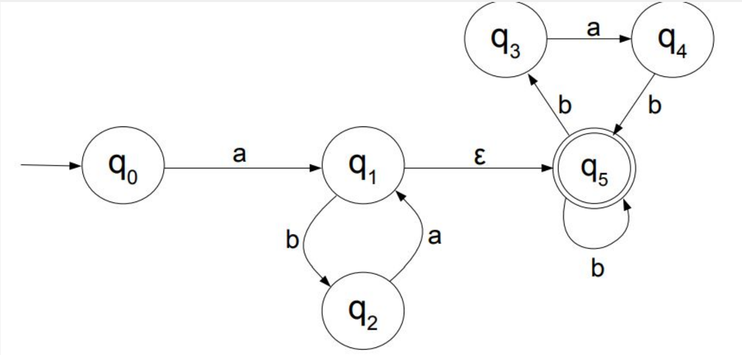
Kako biste vi implementirali?

RE → regularni jezici

regularni jezici → konačni automati

Da li postoji sistematičan način da od proizvoljnog regularnog izraza kreiramo konačni automat?





a(bba|bb)*b

a(ba)*(b|bab)*

Konstrukcija automata za RE

- $\varnothing$ je regularni izraz i označava jezik $L(\varnothing) = \{\}$,
- **ε** je regularni izraz i označava jezik $L(\varepsilon) = \{\varepsilon\}$,
- za svaki simbol $\mathbf{a} \in \Sigma$, a je regularni izraz i označava jezik $L(a) = \{a\}$,
- ako su r i s regularni izrazi koji označavaju jezike L(r) i L(s), onda važe sledeće definicije:
    - **r | s** je regularni izraz koji označava jezik $L(r \mid s) = L(r) \cup L(s)$ (često se koristi i operator +)
    - **rs** je regularni izraz koji označava jezik $L(rs) = L(r)\,L(s)$
    - **r*** je regularni izraz koji označava jezik $L(r^*) = L(r)^*$

$\varnothing$

$L(\varnothing) = \varnothing$

$A = (\{q_0, q_1\}, \Sigma, \sigma, s_0, \{q_1\})$

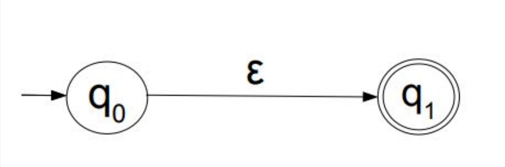
Nema prelaska
Ni za ε

Ne prihvata nijednu reč.

ε

$L(\varepsilon) = \{\varepsilon\}$

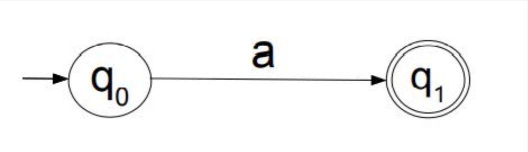
$A = (\{q_0, q_1\}, \Sigma, \sigma, s_0, \{q_1\})$



$a \in \Sigma$

$L(a) = \{a\}$

$A = (\{q_0, q_1\}, \Sigma, \sigma, s_0, \{q_1\})$



$r_1 | r_2$

$L(r_1 | r_2) = L(r_1) \cup L(r_2)$

r_1: $A_1 = (S_1, \Sigma_1, \sigma_1, s_1, F_1)$, r_2: $A_2 = (S_2, \Sigma_2, \sigma_2, s_2, F_2)$

Automati dobijeni prethodnim koracima.
Imaju samo jedno završno stanje iz kojih nema prelaza.

$F_1 = \{f_1\}$, $F_2 = \{f_2\}$

$A = (S_1 \cup S_2 \cup \{s_0, f\}, \Sigma_1 \cup \Sigma_2, \sigma, s_0, F)$

s_0 - novo početno stanje
$F = \{f\}$ - novo završno stanje iz kog nema prelaza
$\sigma = ?$

$r_1 | r_2$

$L(r_1 | r_2) = L(r_1) \cup L(r_2)$

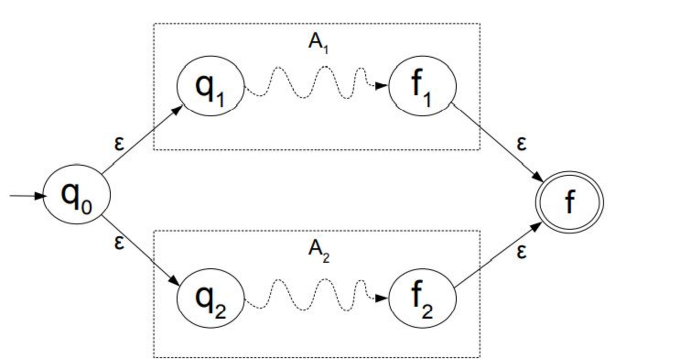
r_1: $A_1 = (S_1, \Sigma_1, \sigma_1, s_1, F_1)$, r_2: $A_2 = (S_2, \Sigma_2, \sigma_2, s_2, F_2)$

Automati dobijeni prethodnim koracima.
Imaju samo jedno završno stanje iz kojih nema prelaza.

$F_1 = \{f_1\}$, $F_2 = \{f_2\}$

$A = (S_1 \cup S_2 \cup \{s_0, f\}, \Sigma_1 \cup \Sigma_2, \sigma, s_0, F)$

s_0 - novo početno stanje
$F = \{f\}$ - novo završno stanje iz kog nema prelaza
σ = ?



Na slici je oznaka za stanje q, u definiciji s

r_1: $A_1 = (S_1, \Sigma_1, \sigma_1, s_1, F_1)$, r_2: $A_2 = (S_2, \Sigma_2, \sigma_2, s_2, F_2)$

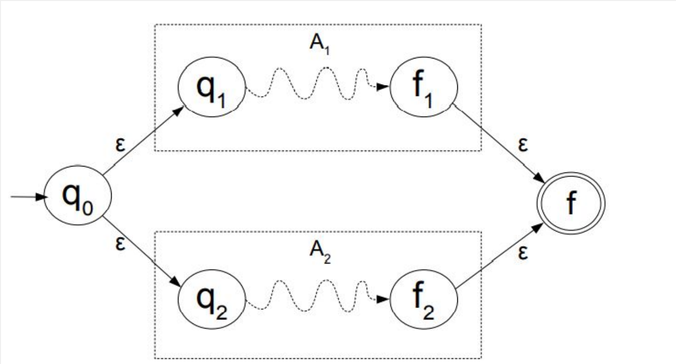
$A = (S_1 \cup S_2 \cup \{s_0, f\}, \Sigma_1 \cup \Sigma_2, \sigma, s_0, F)$

s_0 - novo početno stanje
$F = \{f\}$ - novo završno stanje iz kog nema prelaza
$\sigma = ?$

$\sigma(s_0, \varepsilon) \quad = \{s1, s2\}$

$\sigma(s, a) \quad = \sigma_1(s, a), \quad a \in \Sigma_1$ i $s \in S_1 \setminus \{f_1\}$
$\quad\quad\quad\quad = \sigma_2(s, a), \quad a \in \Sigma_2$ i $s \in S_2 \setminus \{f_2\}$

$\sigma(f_1, \varepsilon) = \{f\}$
$\sigma(f_2, \varepsilon) = \{f\}$



Na slici je oznaka za stanje q, u definiciji s

$r_1 | r_2$

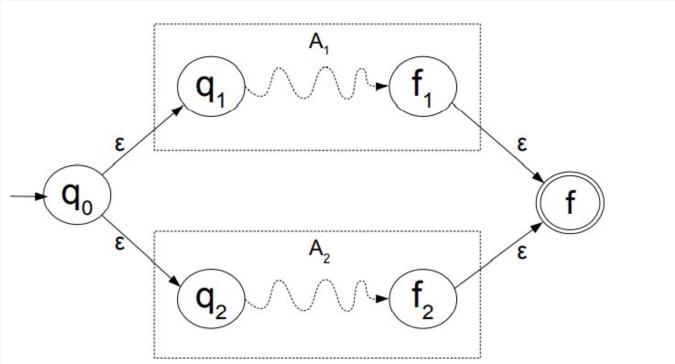
$L(r_1 | r_2) = L(r_1) \cup L(r_2)$

Iz početnog prelazi u početna stanja A_1 i A_2

Ponaša se isto kao A_1 ili A_2, zavisno od stanja

Iz završnih stanja A_1 i A_2 prelazi u novo završno

obezbeđujemo preduslov da automat ima jedno završno stanje, bez prelaza



Na slici je oznaka za stanje q, u definiciji s

$r_1 \cdot r_2$

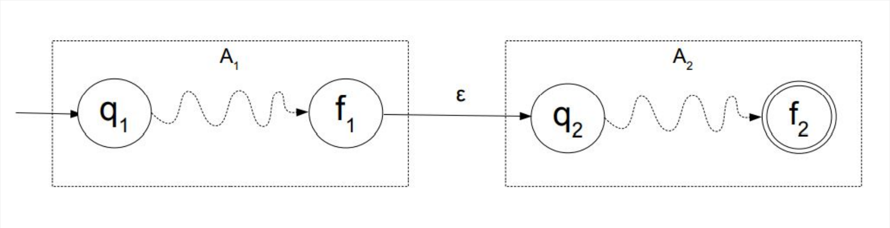
$L(r_1 \cdot r_2) = L(r_1)\,L(r_2)$

r_1: $A_1 = (S_1, \Sigma_1, \sigma_1, s_1, F_1)$, r_2: $A_2 = (S_2, \Sigma_2, \sigma_2, s_2, F_2)$

Automati dobijeni prethodnim koracima.
Imaju samo jedno završno stanje iz kojih nema prelaza.

$F_1 = \{f_1\}$, $F_2 = \{f_2\}$

$A = (S_1 \cup S_2, \Sigma_1 \cup \Sigma_2, \sigma, s_1, F)$



$F = \{f_2\}$ - f_2 nema prelaza

$\sigma = ?$

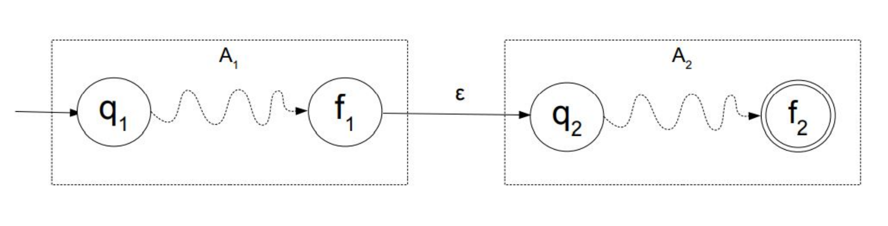
$r_1 \cdot r_2$

$L(r_1 \cdot r_2) = L(r_1)\, L(r_2)$

Počinje kako počinje A_1

Završava se gde se završava A_2

Ponaša se isto kao A_1 ili A_2, zavisno od stanja



r_1^*

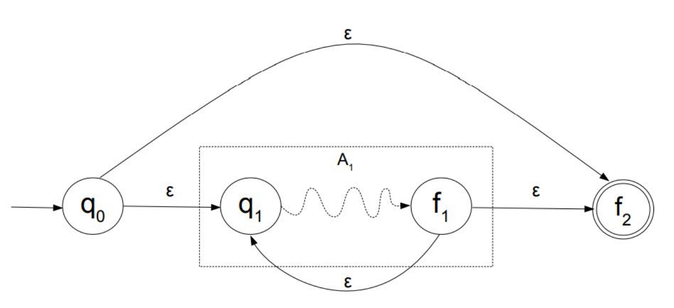
$L(r_1^*) = L(r_1)^*$

r_1: $A_1 = (S_1, \Sigma_1, \sigma_1, s_1, F_1)$

Automati dobijeni prethodnim koracima.
Imaju samo jedno završno stanje iz kojih nema prelaza.

$F_1 = \{f_1\}$

$A = (S_1 \cup \{s_0, f\}, \Sigma_1, \sigma, s_0, F)$



s_0 - novo početno stanje
$F = \{f\}$ - novo završno stanje

$\sigma = ?$

r_1^*

$L(r_1^*) = L(r_1)^*$

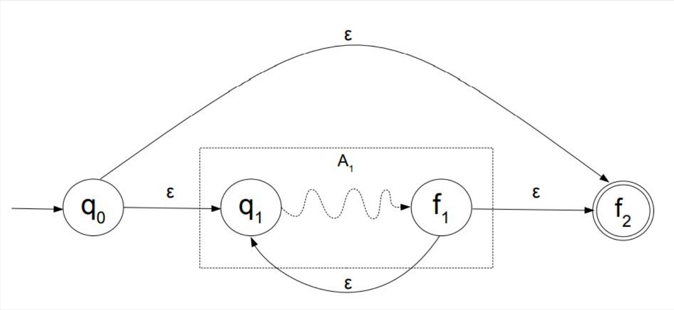
Omogućava praznu reč

(broj ponavljanja = 0)

Ponaša se isto kao A_1

Omogućava ponavljanje

(iz završnog A_1 se vrati na početak ili završi)



PRIMER: r = b(ab)*|ab*a

$r = b(ab)^* \mid ab^*a$

$r = r_1 \mid r_2$
$r_1 = b(ab)^*$
$r_2 = ab^*a$

$r_1 = r_3 r_4$
$r_3 = b$
$r_4 = (ab)^*$

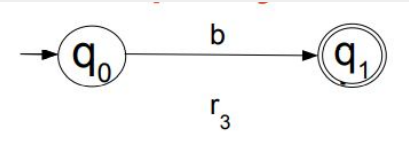
$r_4 = r_5^*$
$r_5 = ab$

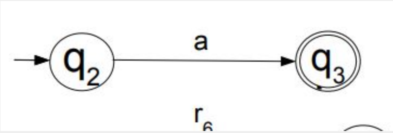
$r_5 = r_6 r_7$
$r_6 = a$
$r_7 = b$

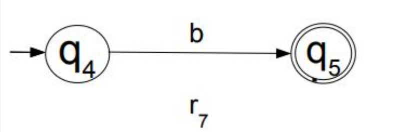
$r_2 = r_8 r_9$
$r_8 = a$
$r_9 = b^*a$

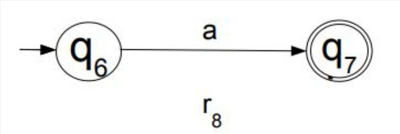
$r_9 = r_{10} r_{11}$
$r_{10} = b^*$
$r_{11} = a$

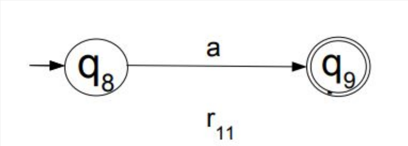
$r_{10} = r_{12}^*$
$r_{12} = b$

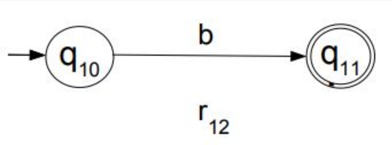












$r = b(ab)^* \mid ab^*a$

$r = r_1 \mid r_2$
$r_1 = b(ab)^*$
$r_2 = ab^*a$

$r_1 = r_3 r_4$
$r_3 = b$
$r_4 = (ab)^*$

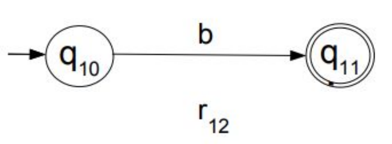
$r_4 = r_5^*$
$r_5 = ab$

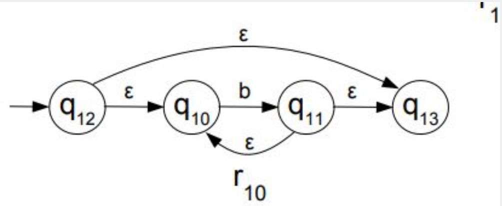
$r_5 = r_6 r_7$
$r_6 = a$
$r_7 = b$

$r_2 = r_8 r_9$
$r_8 = a$
$r_9 = b^*a$

$r_9 = r_{10} r_{11}$
$r_{10} = b^*$
$r_{11} = a$

$r_{10} = r_{12}^*$
$r_{12} = b$





$r = b(ab)^* \mid ab^*a$

$r = r_1 \mid r_2$
$r_1 = b(ab)^*$
$r_2 = ab^*a$

$r_1 = r_3 r_4$
$r_3 = b$
$r_4 = (ab)^*$

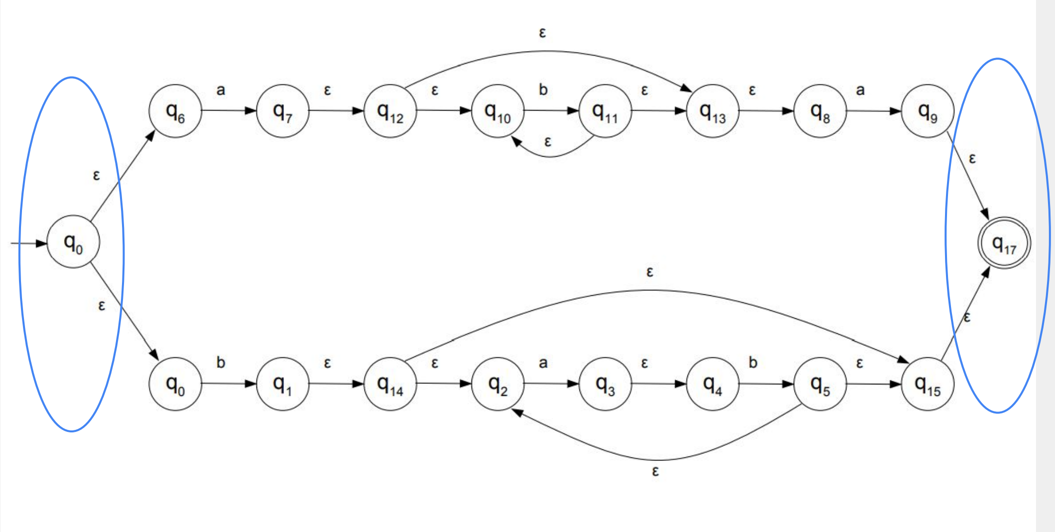
$r_4 = r_5{}^*$
$r_5 = ab$

$r_5 = r_6 r_7$
$r_6 = a$
$r_7 = b$

$r_2 = r_8 r_9$
$r_8 = a$
$r_9 = b^*a$

$r_9 = r_{10} r_{11}$
$r_{10} = b^*$
$r_{11} = a$

$r_{10} = r_{12}{}^*$
$r_{12} = b$



RE $\longrightarrow$ e-NKA $\longrightarrow$ NKA $\longrightarrow$ DKA

RE $\longleftarrow$ e-NKA $\longleftarrow$ NKA $\longleftarrow$ DKA ??