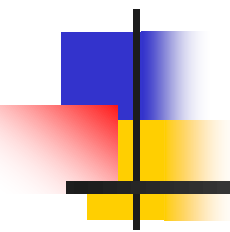


Programski prevodioci 1

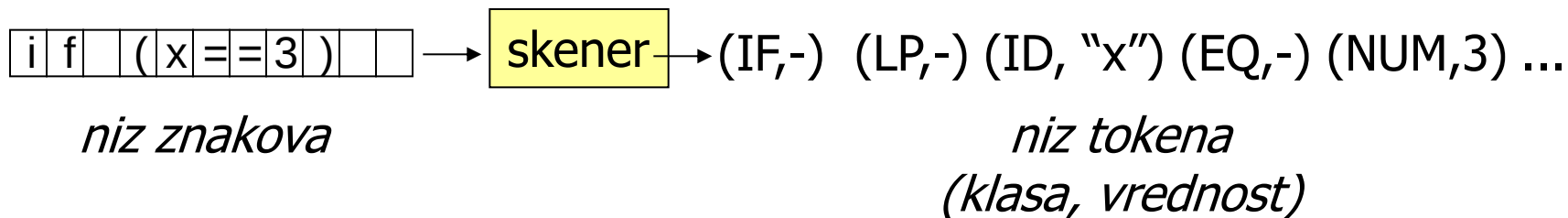
Lekcija 2 – Leksička analiza



Uvod

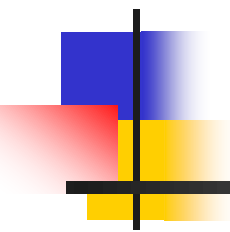
Uloga leksičkog analizatora (skenera)

1. Isporučuje sintaksnom analizatoru terminalne simbole (tokene)



2. Ignoriše neke delove ulaznog teksta

- razmake
- tabulatore
- znake za kraj linije (CR, LF)
- komentare



Deterministički konačni automati

Konačni automati

Startno stanje
označeno
strelicom
(ako je nema,
onda stanje
u prvoj vrsti
tabele

	0	1
A	D	A
B	A	C
C	A	F
D	B	C
E	B	C
F	E	A

Ulazni simboli

A je stanje
odbijanja

E je stanje
prihvatanja

Stanja

δ - funkcija prelaza
 $\delta(F, 1) = A$

- za određeno stanje i ulaz jednoznačno je određeno sledeće stanje => *deterministički* konačni automat (DKA)

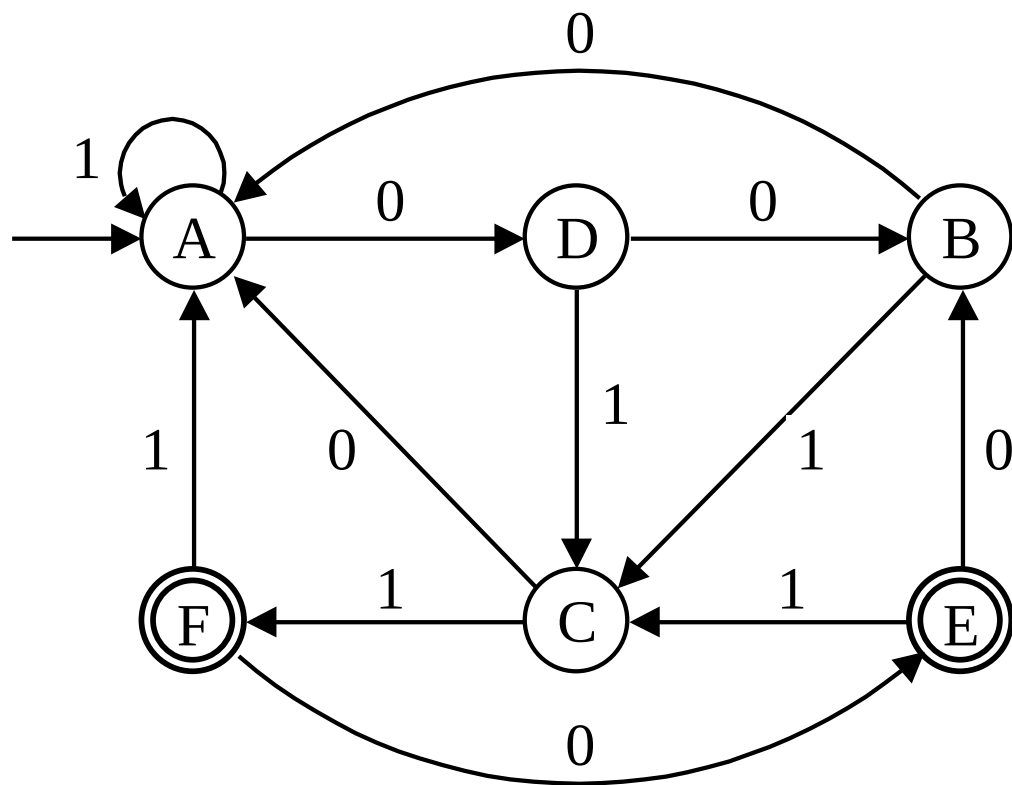


Konačni automati

Deterministički konačni automat opisan je uređenom petorkom (S, U, δ, S_t, P) gde:

- S - skup stanja automata $S = \{A, B, C, D, E, F\}$;
- U - skup ulaznih simbola (azbuku automata), $U = \{0, 1\}$;
- $\delta: S \times U \rightarrow S$ predstavlja funkciju prelaza; za svako stanje i svaki ulazni simbol definiše novo stanje u koje automat prelazi iz stanja $s \in S$ za ulazni simbol $u \in U$; u ulaz tabele u vrsti X i koloni Y upisana je vrednost $\delta(X, Y)$;
- $S_t \in S$ je startno stanje automata $S_t = A$;
- $P \subseteq S$ skup stanja prihvatanja, $P = \{E, F\}$. Stanja iz skupa $S-P$ su stanja odbijanja.

Graf prelaza konačnog automata



- Čvorovi: stanja
- Grane: prelazi
- Stanja prihvatanja: poduplano

Algoritam rada DKA

- Algoritam rada automata:

```
tekuće_stanje :=  $S_t$ ; {  $S_t$  – startno stanje }
tekući_ulaz := prvi simbol ulazne sekvence;
while not (kraj ulazne sekvence)
    tekuće_stanje :=  $\delta$ ( tekuće_stanje, tekući_ulaz );
    tekući_ulaz := sledeći znak ulazne sekvence;
end while;
if ( tekuće_stanje  $\in$  P )      {P – skup stanja prihvatanja}
    then ulazna sekvenc a se prihvata;
    else ulazna sekvenc a se ne prihvata;
endif;
```

- Primer:*

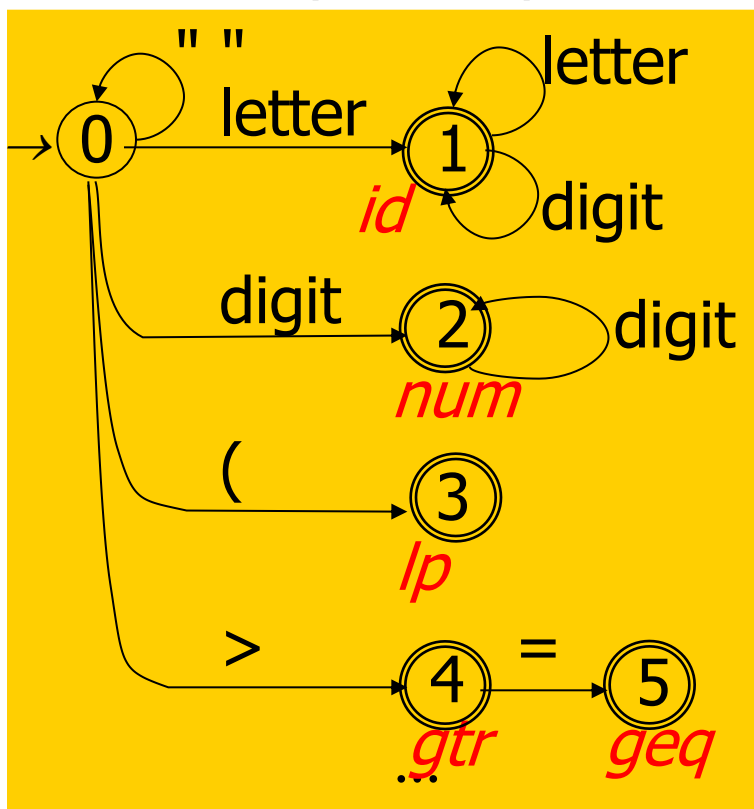


- završno stanje E je stanje prihvatanja $\Rightarrow$ 0110 pripada jeziku $L(K)$ automata K

- ## Programski prevodioci

Skener u obliku DKA

Konceptualni primer: skener za tokene id, num, lp, gtr, geq



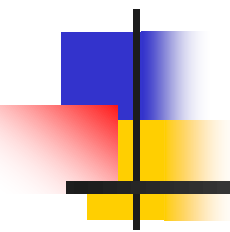
Ulazni niz: `max >= 30`

$s_0 \xrightarrow{\text{max}} s_1$ • Prepoznat *id*

$s_0 \xrightarrow{>=} s_5$ • Preskače beline na početku
• Prolazi kroz stanje 4
• Prepoznat *geq*

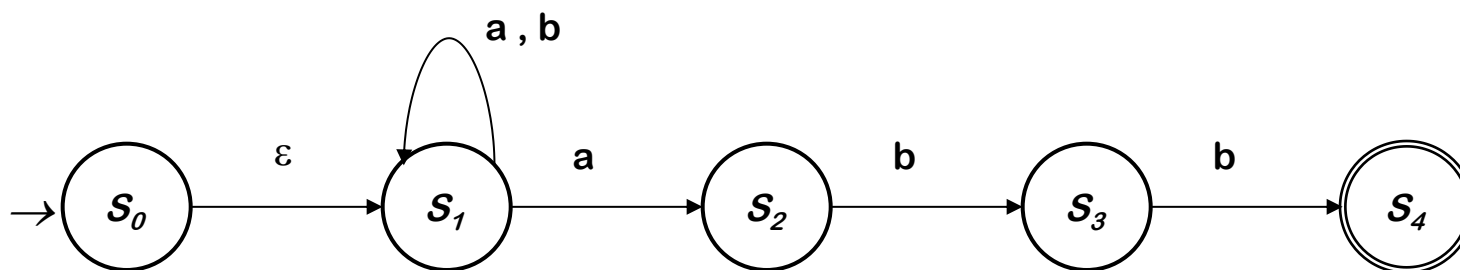
$s_0 \xrightarrow{30} s_2$ • Preskače beline na početku
• Prepoznat *num*

Posle svakog prepoznatog tokena skeniranje kreće ponovo od s_0 .



Nedeterministički konačni automati

Nedeterministički konačni automati



Specifičnosti automata:

- s_0 ima prelaz za ε (tzv. prazna sekvenca, automat može da pređe iz s_0 u s_1 a da pri tom ne konzumira nijedan simbol sa ulaza)
- s_1 ima dva različita prelaza za a

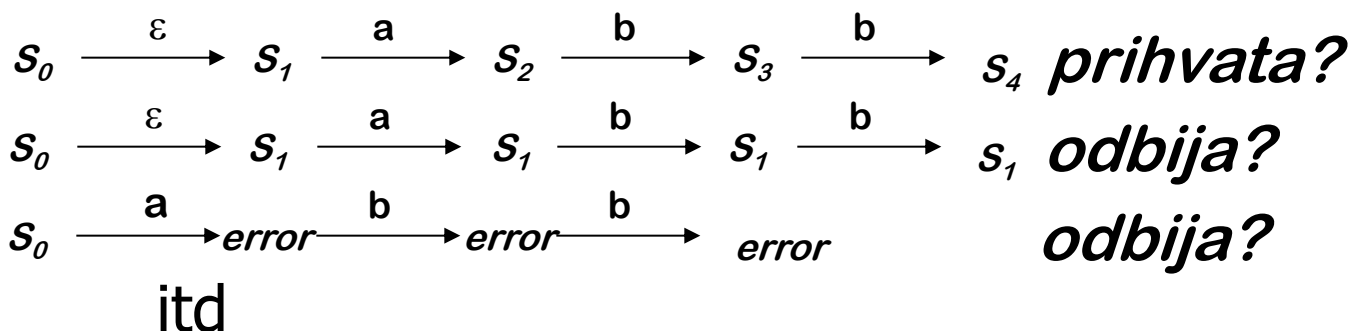
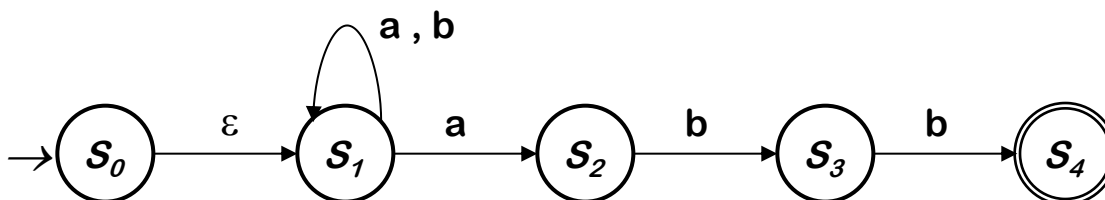
U pitanju je ***nedeterministički konačni automat (NKA)***

Nedeterministički konačni automat

Opisan je uređenom petorkom (S, U, δ, S_t, P) gde:

- • S predstavlja skup stanja automata, za dati primer $\{s_0, s_1, s_2, s_3, s_4, \text{error}\}$
- • U predstavlja skup ulaznih simbola (azbuku automata), $U = \{a, b\}$
- • $\delta: S \times (U \cup \{\varepsilon\}) \rightarrow P(S)$ predstavlja funkciju prelaza (P - partitivni skup); za svako stanje i svaki ulazni simbol definiše **skup** $S_N \subseteq S$ mogućih novih stanja u koje automat može preći iz stanja $s \in S$ za ulazni simbol $u \in U$ **ili za simbol prazne sekvence** ε
- • $S_t \subseteq S$ je **skup** startnih stanja automata. $S_t = \{s_0\}$
- • $P \subseteq S$ predstavlja skup stanja prihvatanja, u konkretnom slučaju $P = \{s_4\}$. Stanja iz skupa $S-P$ nazivaju se stanjima odbijanja.

Rad datog NKA za ulaz abb



- **Generalno**, NKA prihvata ulaznu sekvencu x ako i samo ako za x postoji scenario promene stanja iz nekog od startnih stanja do nekog od stanja prihvatanja
- *U ovom primeru, NKA prihvata sekvencu abb*

Algoritam rada NKA

Za dati NKA i datu ulaznu sekvencu na izlazu daje podatak da li se sekvenca prihvata ili odbija.

Algoritam koristi funkciju ε -zatvaranja:

ε -closure(s) daje skup stanja dostižnih iz s po ε

Napomena: s je uključeno u skup

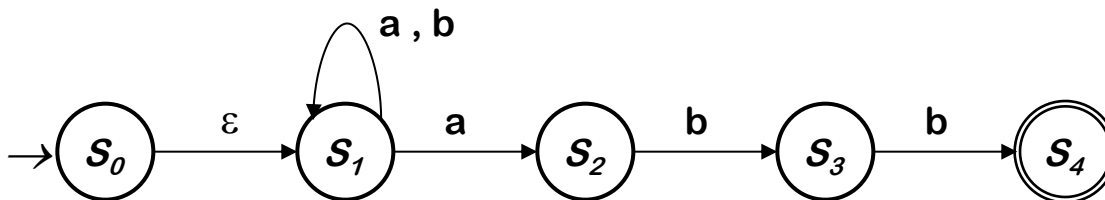
1. $S'' = S' = \{s\}$
2. Za svako p iz S'
3. $S'' = S'' \cup \delta(p, \varepsilon)$
4. Ako je $S'' = S'$ onda kraj, rezultat je S''
5. $S' = S''$;
6. Idi na korak 2

Algoritam rada NKA

- Moramo voditi evidenciju, u svakom koraku rada automata, o skupu svih mogućih stanja u kojima se automat može nalaziti.

```
tekući_skup_stanja :=  $S_t$ ;  
tekući_ulaz := prvi simbol ulazne sekvence;  
while not (kraj ulazne sekvence)  
    novi_skup_stanja :=  $\emptyset$ ;  
    for ( $\forall s \in \text{tekući\_skup\_stanja}$ )  
        novi_skup_stanja := novi_skup_stanja  $\cup \delta(s, \text{tekući\_ulaz})$ ;  
    end for;  
    tekući_skup_stanja :=  $\epsilon$ -closure(novi_skup_stanja);  
    tekući_ulaz := novi simbol ulazne sekvence;  
end while;  
if (tekući_skup_stanja  $\cap P \neq \emptyset$ )  
    then ulazna sekvencu se prihvata;  
    else ulazna sekvencu se ne prihvata;  
end if.
```

Rad datog NKA za ulaz abb



$s_0 s_1 \xrightarrow{a} s_1 s_2 \xrightarrow{b} s_1 s_3 \xrightarrow{b} s_1 s_4$

$\epsilon\text{-closure}(s_0)$

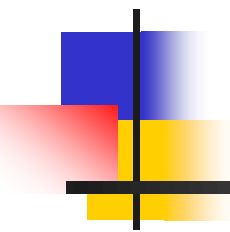
prihvata zbog s4



Odnos NKA i DKA

DKA je specijalan slučaj NKA

- DKA ima tačno jedno startno stanje
- DKA nema ε prelaza
- Funkcija prelaza DKA ima jedinstvenu vrednost
- Uvek se može konstruisati DKA koji je ekvivalentan (prepoznaje isti jezik) kao zadati NKA:
 - Svako stanje DKA je skup mogućih stanja NKA do kojih NKA može doći u toku rada
- Moguće eksponencijalno povećanje broja stanja



Konverzija NKA u DKA



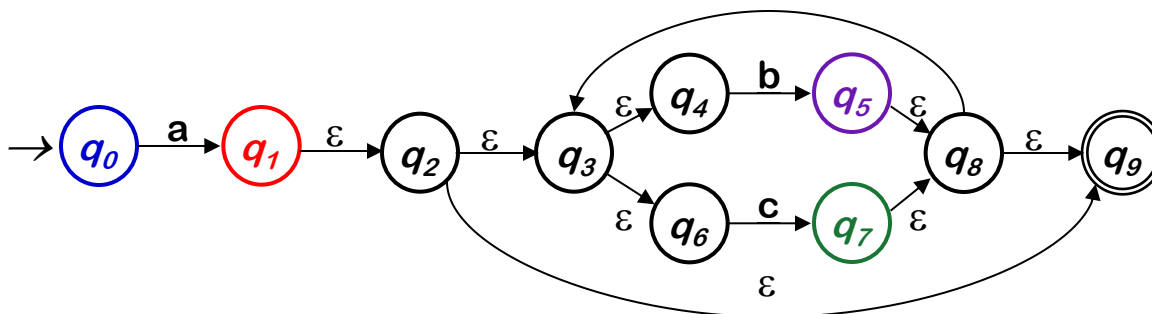
NKA $\rightarrow$ DKA

Algoritam koristi istu ideju kao algoritam rada NKA, samo što ne posmatra jednu ulaznu sekvencu, nego sistematski određuje tabelu prelaza DKA.

Algoritam:

1. Neka je s_0 startno stanje NKA.
2. Odrediti $S_0 = \varepsilon\text{-closure}(s_0)$ koje postaje startno stanje DKA
3. $\text{ново_stanje_DKA} = \{\}$
4. Za svako s iz S_0 ,
5. Za svako α iz ulazne azbuke:
 $\text{ново_stanje_DKA} = \text{ново_stanje_DKA} \cup \varepsilon\text{-closure}(\delta(s, \alpha))$
6. Označiti ново_stanje_DKA kao stanje prihvatanja, ako u svom skupu sadrži bar jedno stanje prihvatanja NKA
7. Iterativno ponavljati tačke 3-6 za sva dobijena stanja, dok postoji promena u skupu stanja DKA.

NKA → DKA prethodni primer



Određivanje ekvivalentnog DKA

stanja DKA	stanja NKA	ε-zatvaranje q1		
		a	b	c
→ s_0	q_0	$q_1, q_2, q_3, q'_4, q'_6, q'_9$		
s_1	$q_1, q_2, q_3, q_4, q_6, q_9$		$q_5, q_8, q_9, q'_3, q'_4, q'_6$	$q_7, q_8, q_9, q'_3, q'_4, q'_6$
s_2	$q_5, q_8, q_9, q_3, q_4, q_6$		s_2	s_3
s_3	$q_7, q_8, q_9, q_3, q_4, q_6$		s_2	s_3

0

1

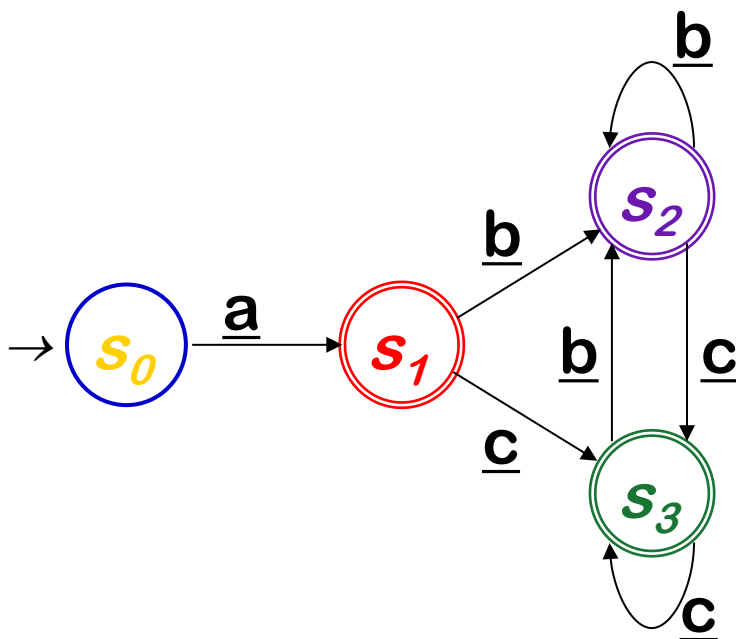
1

1

Samo tamo gde se u skupu stanja NKA nalazi završno stanje q_9

NKA → DKA prethodni primer (2)

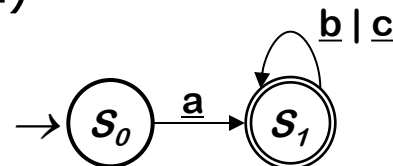
Rezultujući DKA

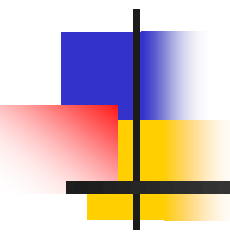


→ s_0
 s_1
 s_2
 s_3

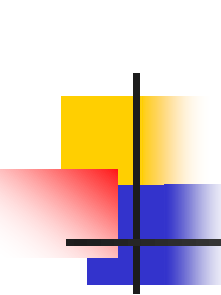
	a	b	c	
→ s_0	s_1	-	-	0
s_1	-	s_2	s_3	1
s_2	-	s_2	s_3	1
s_3	-	s_2	s_3	1

Stanja s_1 , s_2 , s_3 se ponašaju isto pa se mogu objediniti (tzv. minimizacija automata, biće obrađena na vežbama)



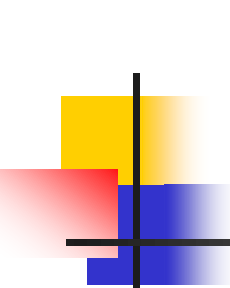


Regularni izrazi



Regularni izrazi

- Alternativni formalizam za opis jezika
- Regularni izrazi opisuju tačno isti skup jezika kao konačni automati:
 - Za svaki regularni izraz može se naći konačni automat koji opisuje isti jezik kao taj izraz
 - Za svaki konačni automat može se naći regularni izraz koji opisuje isti jezik kao taj automat



Definicija regularnog izraza

- Regularni izraz s opisuje regularni skup nizova znakova $L(s)$.
- Regularni izrazi definišu se na sledeći način:
 - $\emptyset$ je regularan izraz koji opisuje prazan skup $L(\emptyset)=\emptyset$.
 - ε je regularan izraz koji opisuje skup $\{\varepsilon\}$ $L(\varepsilon)=\{\varepsilon\}$.
(sekvenca dužine nula znakova, tzv. *prazna* sekvenca)
 - niz znakova n je regularni izraz koji opisuje skup $\{n\}$ $L(n)=\{n\}$.

Definicija regularnog izraza (2)

- složeniji regularni izrazi dobijaju se od regularnih izraza A i B primenom operacija:

- Opcije $A?$, $L(A?) = L(A) \cup \{\varepsilon\}$
- unije $A|B$, $L(A|B) = L(A) \cup L(B) = \{x \mid x \in A \text{ ili } x \in B\}$
- konkatencije (nadovezivanja) AB ili $A \cdot B$
 $L(AB) = L(A) \times L(B) = \{xy \mid x \in A, y \in B\}$

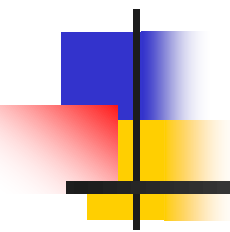
Konvencija:

$$A^n \equiv \underbrace{A \cdot A \cdot \dots \cdot A}_n \quad A^0 \equiv \{\varepsilon\}$$

- zvezdastog zatvaranja A^*
 $A^* = A^0 | A^1 | A^2 | A^3 | \dots$ (beskonačan broj članova)
- pozitivnog zatvaranja A^+
 $A^+ = A^1 | A^2 | A^3 | \dots$

Primeri regularnih izraza

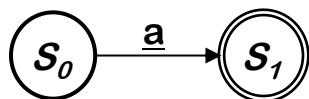
- $L(abc) = \{abc\}$ $L(a \mid b \mid c) = \{a, b, c\}$
- $L[(a \mid b)(c \mid d)] = \{ac, ad, bc, bd\}$ male zagrade su obavezne zbog prioriteta
- $L(a^*) = \{\varepsilon, a, aa, aaa, \dots\}$
- $L[(a \mid b)^+] = \{a, b\} \cup \{a, b\} \times \{a, b\} \cup \{a, b\} \times \{a, b\} \times \{a, b\} \cup \dots = \{a, b\} \cup \{aa, ab, ba, bb\} \cup \{aaa, aab, aba, abb, baa, bab, bba, bbb\} \cup \dots = \{a, b, aa, ab, ba, bb, \dots\} =$ svi mogući nizove slova a i b dužine 1 ili više
- Slovo = $A \mid B \mid \dots \mid Z \mid a \mid b \mid \dots \mid z$
Cifra = $0 \mid 1 \mid \dots \mid 9$
Identifikator = $\text{Slovo} (\text{Slovo} \mid \text{Cifra})^*$



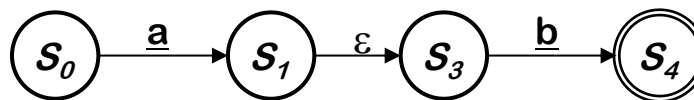
Konstrukcija konačnog automata iz regularnog izraza

Tompsonov algoritam

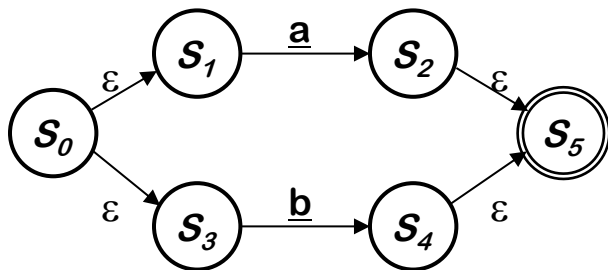
- Za zadati reg. izraz određuje nedeterministički konačni automat



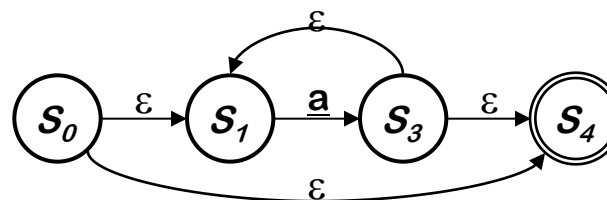
NKA for a



NKA for ab



NKA for a | b



NKA for a*

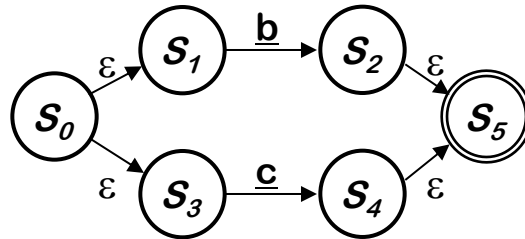
Ken Thompson, CACM, 1968

Primer primene Thompsonovog algoritma

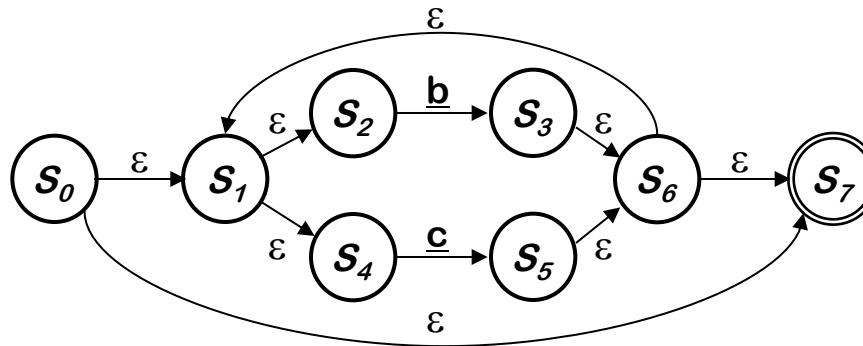
Zadati reg. Izraz: $\underline{a} (\underline{b} \mid \underline{c})^*$, krećemo od osnovnih komponenta pa kombinujemo u veće podizraze:



2. $\underline{b} \mid \underline{c}$

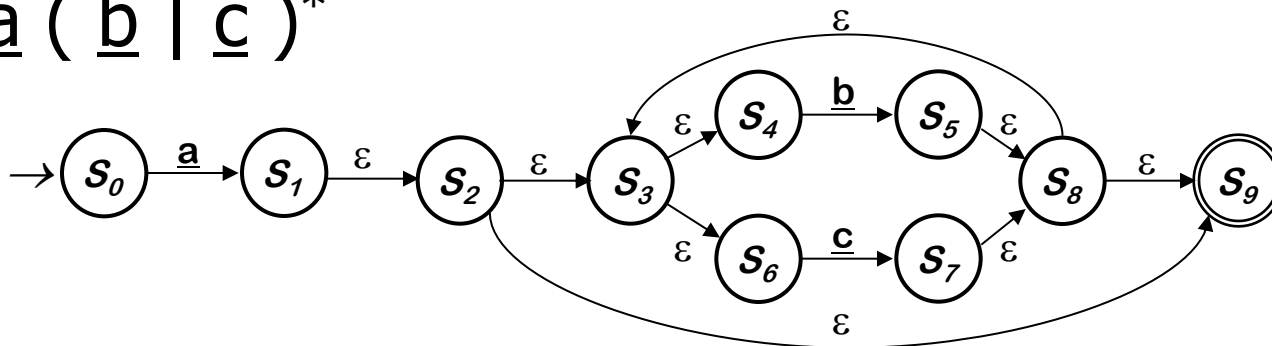


3. $(\underline{b} \mid \underline{c})^*$

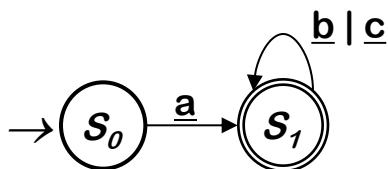


Primer primene Thompsonovog algoritma 2

4. $\underline{a} (\underline{b} \mid \underline{c})^*$



Posle konverzije u DKA i minimizacije broja stanja:





Automatizacija konstrukcije skenera

Formalna metodologija dobijanja programskog koda na osnovu specifikacije jezika:

- 1 Napisati RE za zadati ulazni jezik
- 2 Konstruisati NKA za dati RE Tompsonovim alg.
- 3 Konstruisati DKA ekvivalentan dobijenom NKA
- 4 (Po potrebi) odrediti minimalan DKA
- 5 Implementirati DKA iz tačke 4. programom

Generatori skenera rade po opisanom principu:

- Lex, Flex, jlex, jflex,...