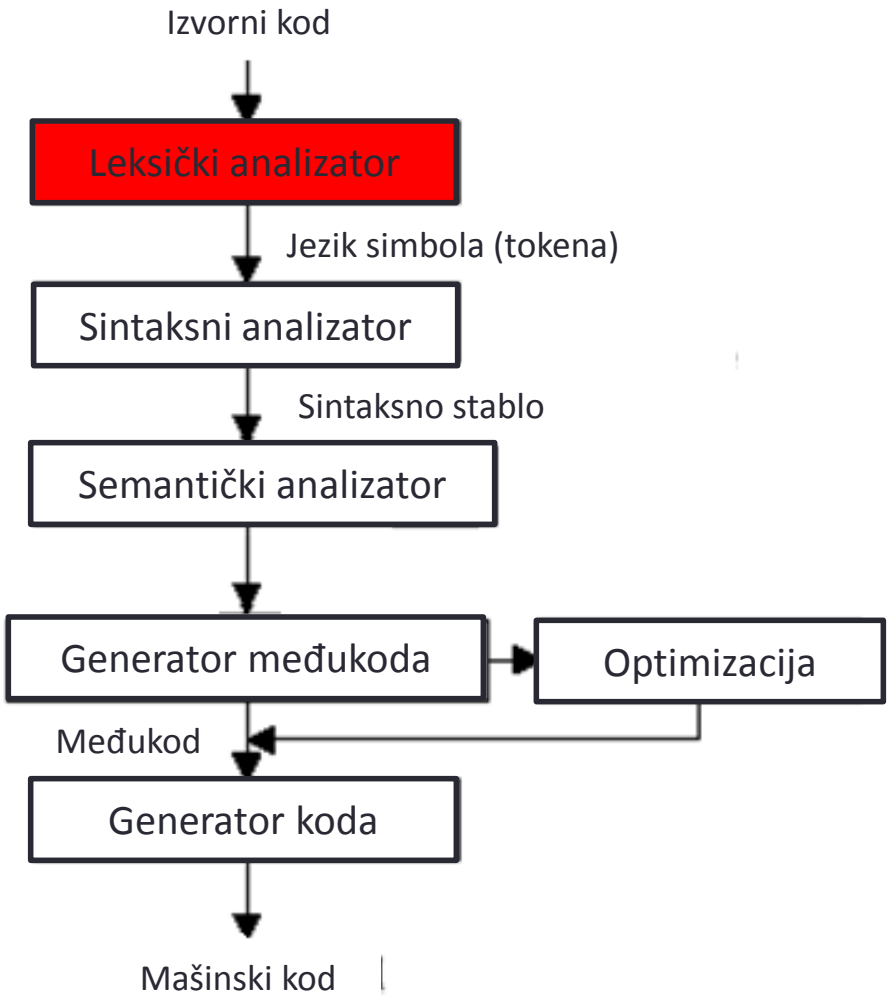
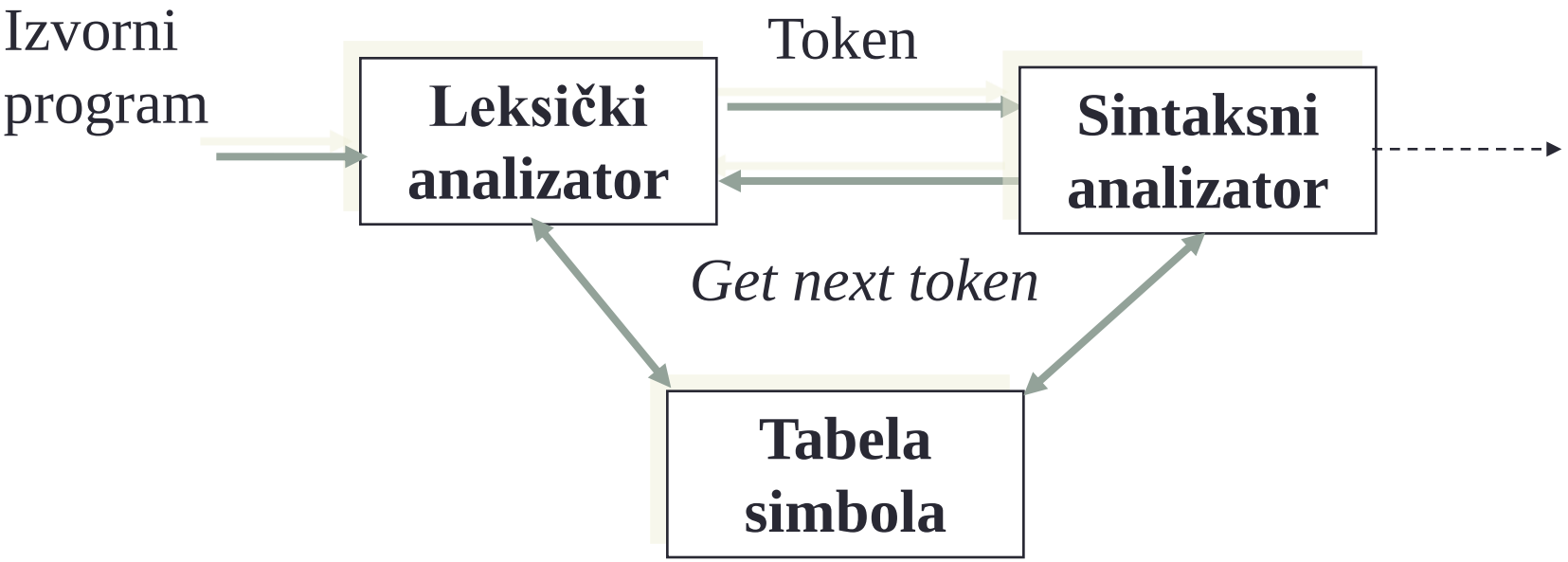
PROGRAMSKI PREVODIOCI
- Leksička analiza -

Struktura kompilatora



Uloga leksičkog analizatora



Leksički analizator je prva faza kompilatora. Njegova glavna uloga je da čita ulazni kod i gneriše sekvencu simbola (tokena) koju prosleđuje sintaksnom analizatoru.

Dodatni zadaci leksičkog analizatora

- Izbacuje iz ulaznog koda delove koji nisu značajni za sintaksnu analizu: komentare i „bele simbole“ (blanko znake, tabulatore i *newline* simbole).
- Usklađuje listing gršaka sa ulaznim kodom. Npr. vodi računa o broju *newline* simbola, što se koristi kod referenciranja na grške.

Zašto leksički analizator kao posebna komponenta?

- Jednostavnija realizacija
- Prenosivost kompilatora – U fazi leksičke analize eliminišu se svi mašinski zavisni elementi i generiše mašinski nezavisna sekvenca tokena koja se prosleđuje sintaksnom analizatoru.

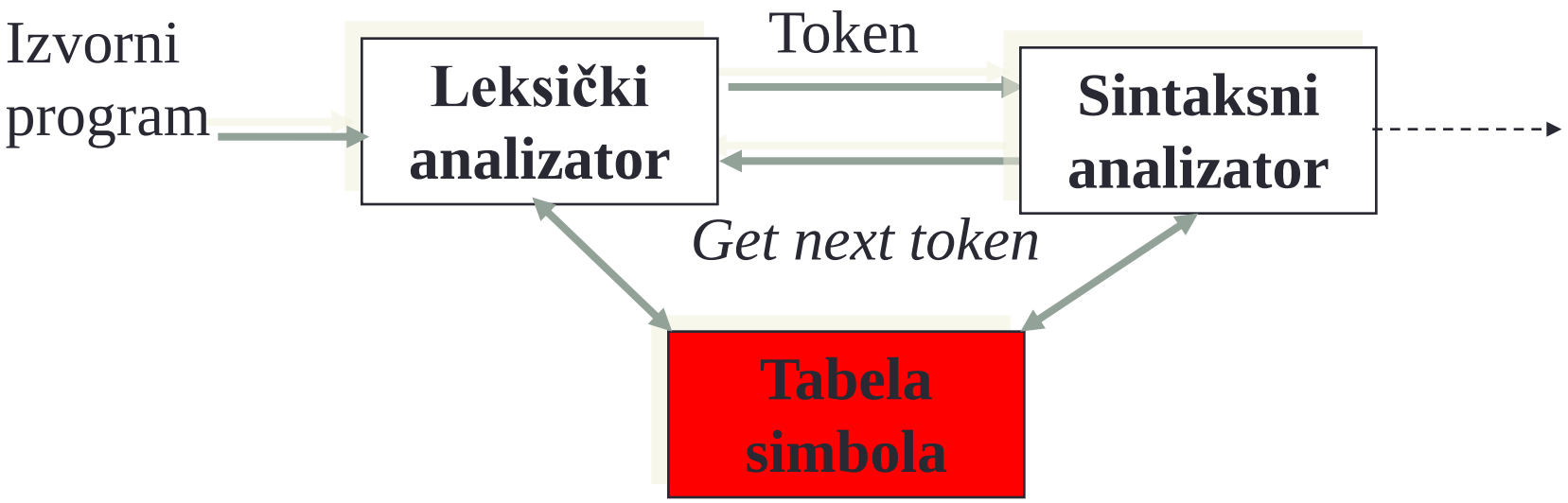
Lekseme, tokeni i šabloni

- LEKSEMA – Izdvojena reč.
- TOKEN – Značenje izdvojene reči (obično celobrojna konstanta).
- ŠABLON (pattern) – Formalni opis kako izgleda reč koja se izdvaja. Najčešće se definiše regularnim izrazom (može i prirodnim jezikom).

Tokeni, šabloni i lekseme

TOKEN	LEKSEME	ŠABLON
if	if	if
rel	<,<=,=,<>,>,>=	< | <= | = | <> | > | >=
id	pi, C, P2	Niz slova i cifara u kojima je prvi znak obavezno slovo
num	3.1416, 0, 6.02E23	Niz dekadnih cifara koji opciono sadrzi tacku i/ili eksponent
literal	“Ovo je podatak”	Niz znakova između navodnika

Tabela simbola



- Uz tokene leksički analizator generiše i neke atribute izdvojenih reči. Dok su sami tokeni bitni za sintaksnu analizu, atributi su bitni u fazi generisanja koda.
- Ako je izdvojeni token identifikator (simboličko ime), on kao dodatni atribut ima pointer (referencu) na element u tabeli simbola u kojoj se čuvaju informacije o tom simbolu.

Primer - Atributi tokena

Primer naredbe dodeljivanja:

E = M * C + 2

Tokeni sa pridruženim atributima:

<**id**, pointer na TS za E>

<**op_dod>**

<id, pointer na TS za M>

<**op_mul**>

<**id**, pointer na TS za C>

< **op_add**>

<**const**, integer vrednost 2>

Regularni izrazi

Pravila kojima se definišu regularni izrazi nad azbukom V.

1. ε je regularni izraz i označićemo ga sa {ε }, skup koji sadrži prazan niz.
2. Ako je a simbol azbuke V tada je a regularni izraz i označavaćemo ga sa {a}. Svako slovo azbuke je regularni izraz.
3. Ako su r i s regularni izrazi koji opisuju jezike L(r) i L(s) tada su i:
   - (r) | (s) regularni izraz koji predstavlja $L(r) \cup L(s)$.
   - (r)(s) regularni izraz koji predstavlja L(r)L(s).
   - (r)* regularni izraz koji predstavlja L(r)*

Operator * je najvišeg prioriteta

Operator nadovezivanja je sledeći po prioritetu

Operator | je najnižeg prioriteta

Svi ovi operatori su levo asocijativni

Algebarska svojstva regularnih izraza

AKSIOM	OPIS
r | s = s | r	Unija je komutativna operacija
r | (s | t) =(r | s) | t	Unija je asocijativna operacija
r(st) = (rs)t	Nadovezivanje ja asocijativna operacija
r(s | t) = rs | rt (s | t)r = sr | tr	Distributivnost nadovezivanja u odnosu na uniju
εr = r rε = r	ε je neutralni element za operaciju nadovezivanja
r* = (r | ε)*	Relacija između * i ε
r** = r*	Iteracija je idempotentna operacija

Primeri regularnih izraza

- Numerička konstanta je niz dekadnih cifara koji opciono sadrzi tacku i/ili eksponent
- Identifikator je niz slova i cifara u kojem je prvi znak obavezno slovo

cifra = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 ili **cifra** = [0 – 9]

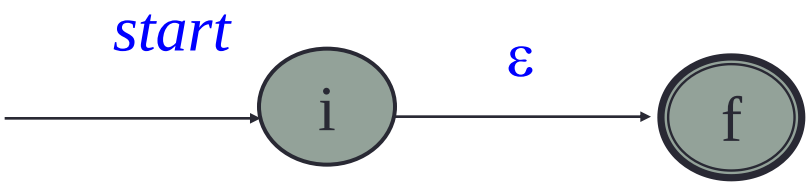
cifre = cifra cifra* ili **cifre = cifra$^+$**

slovo = A | B | C | … | Z | a | b | … | z ili **slovo** = [A - Z a - z]

id = slovo (slovo | cifra)*

num = (+ | -)? **cifra**$^+$ (.**cifra**$^+$)? (E (+ | -)? **cifra**$^+$)? (r)? = r | ε

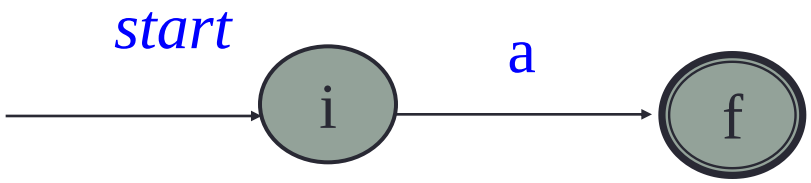
Generisanje grafa automata na osnovu regularnih izraza



i - novo početno stanje

f – novo završno stanje

Automat prepoznaje {ε}

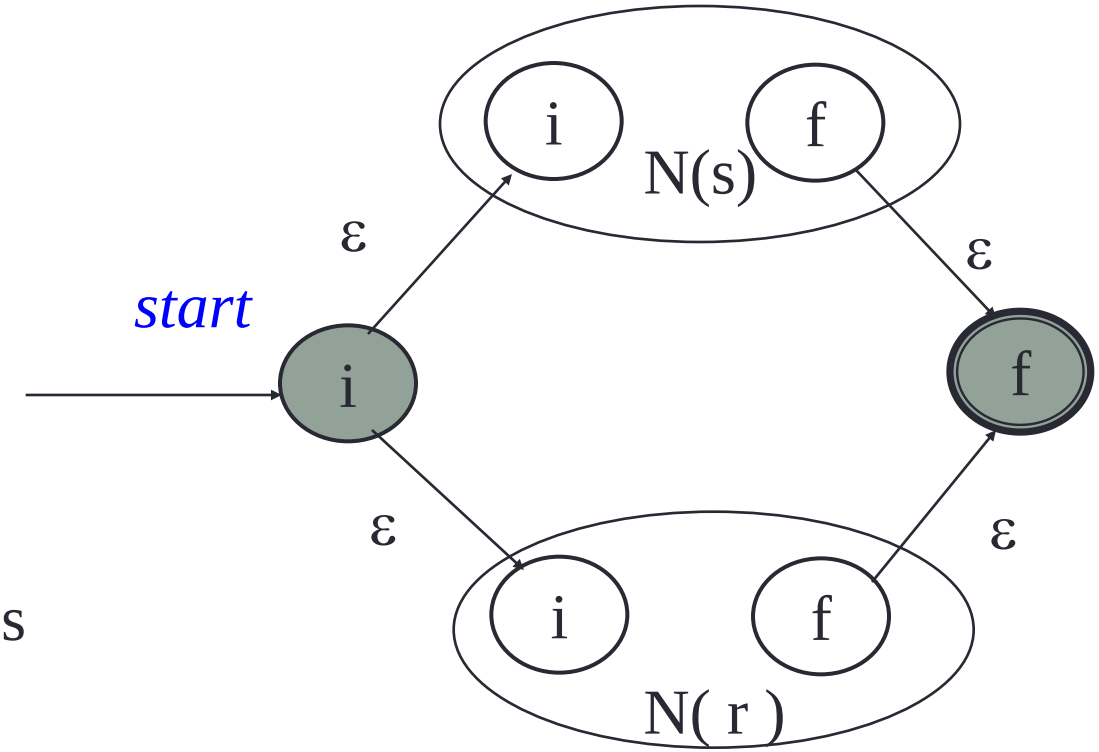


i - novo početno stanje

f – novo završno stanje

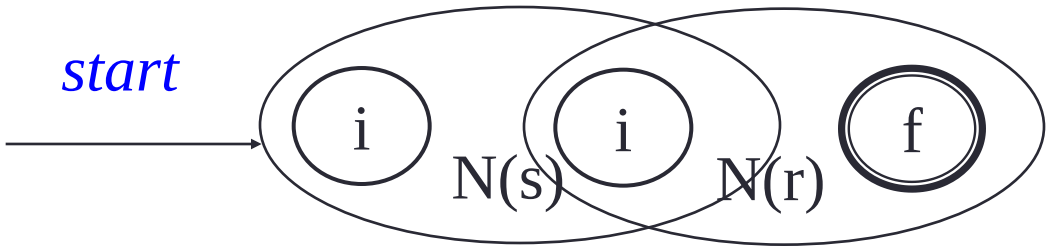
Automat prepoznaje {a}

Generisanje grafa automata N(s | r)



N(s) – podautomat koji realizuje regularni izraz s
N(r) – podautomat koji realizuje r
Automat prepoznaje s|r

Generisanje grafa automata N(s r)

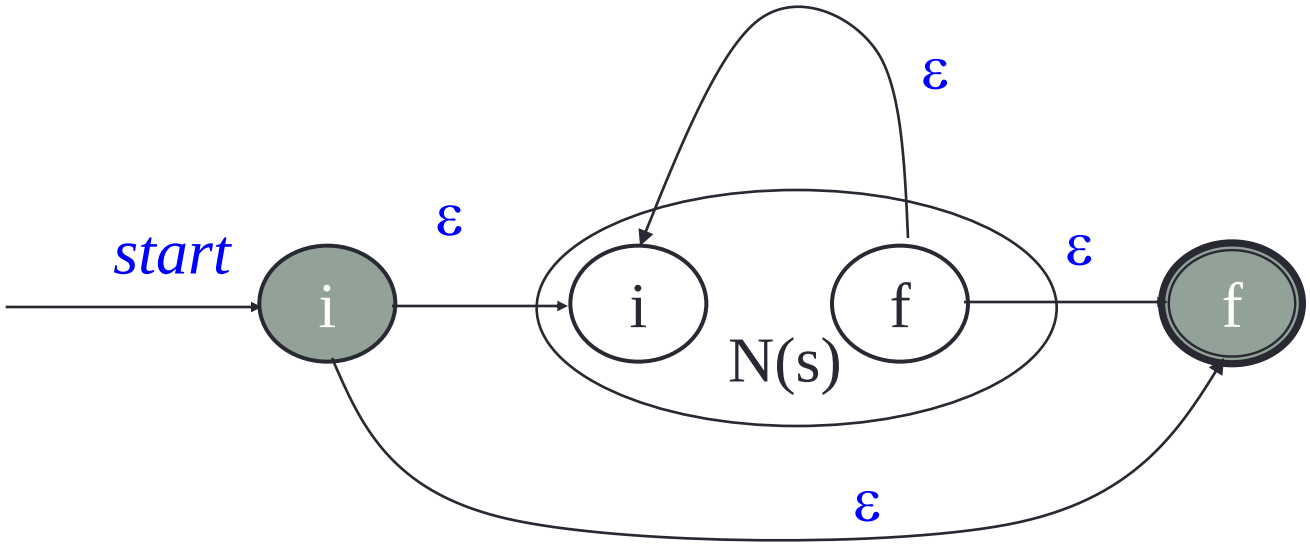


N(s) – podautomat koji realizuje s

N(r) – podautomat koji realizuje r

Automat prepoznaje sr

Generisanje grafa automata N(s*)

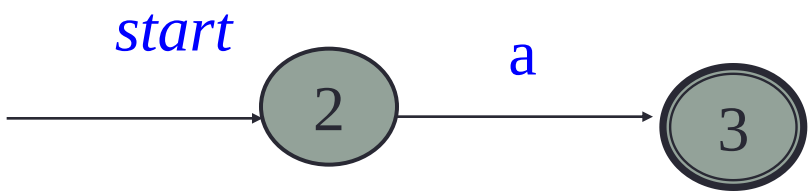


N(s) – podautomat koji realizuje regularni izraz s

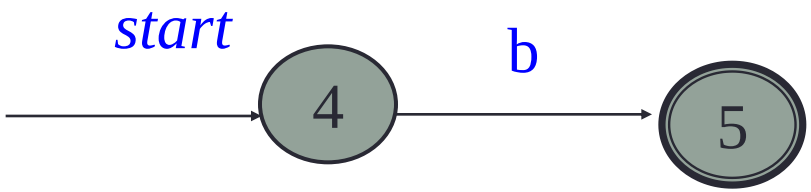
automat prepoznaje **N(s*)**

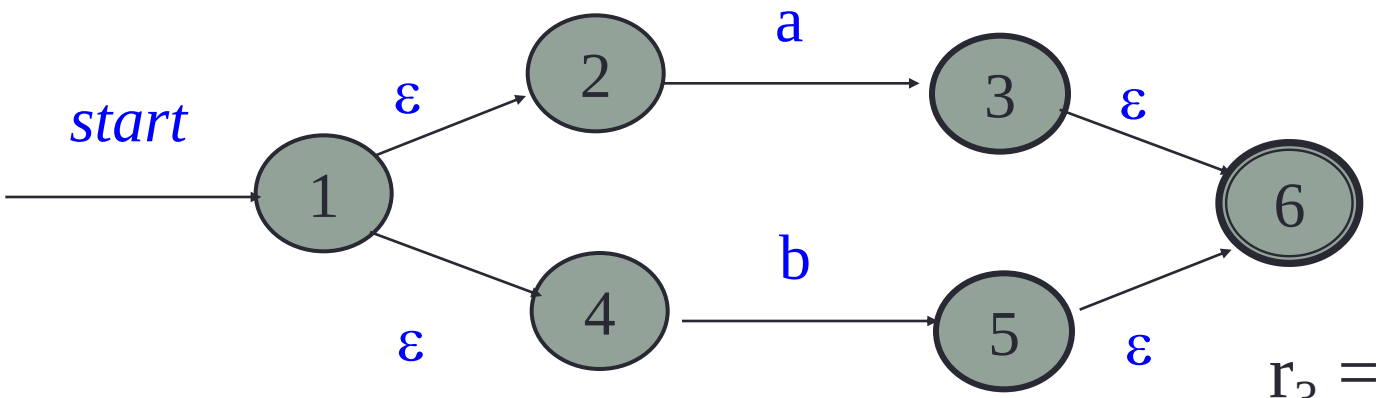
Primer: r=(a | b)*abb

$r_1 = a$



$r_2 = b$

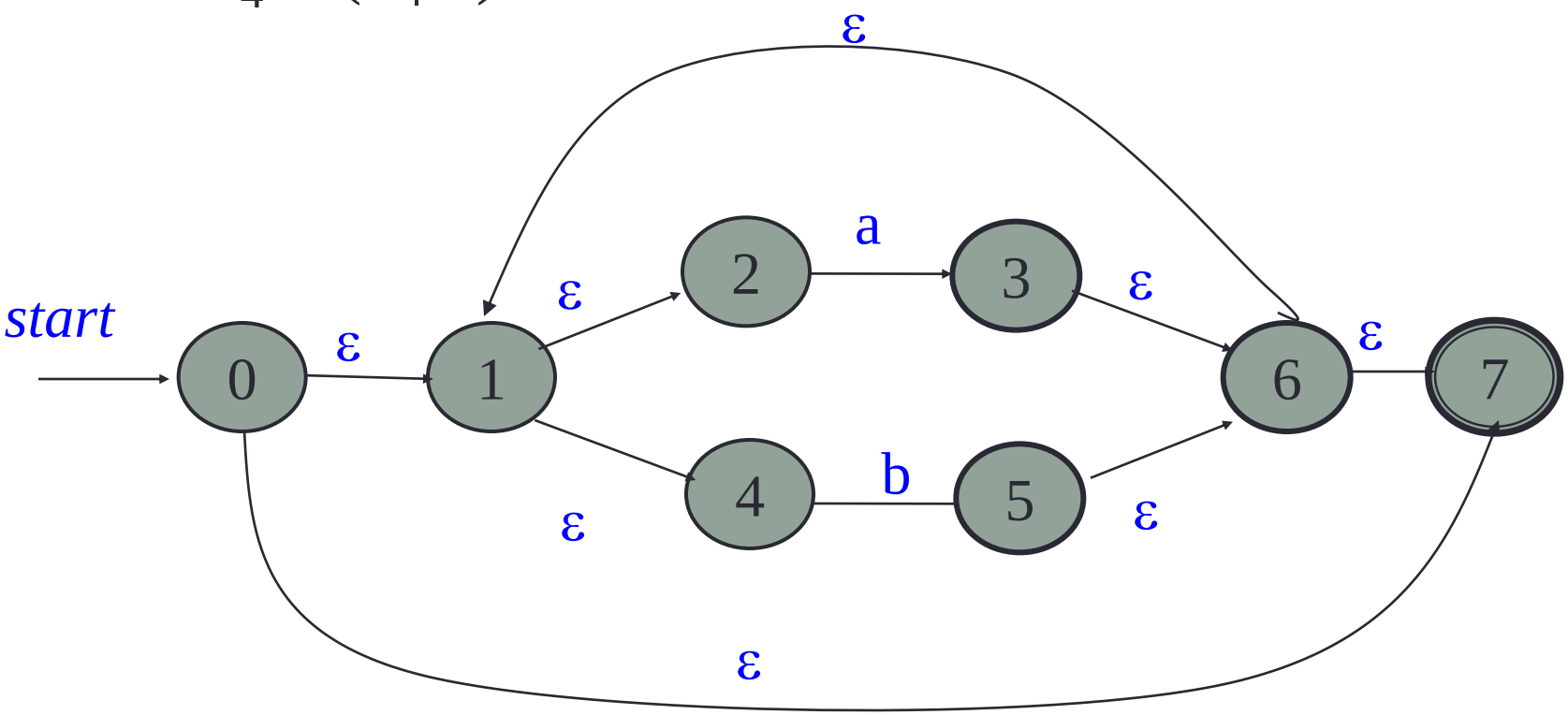




$r_3 = a \mid b$

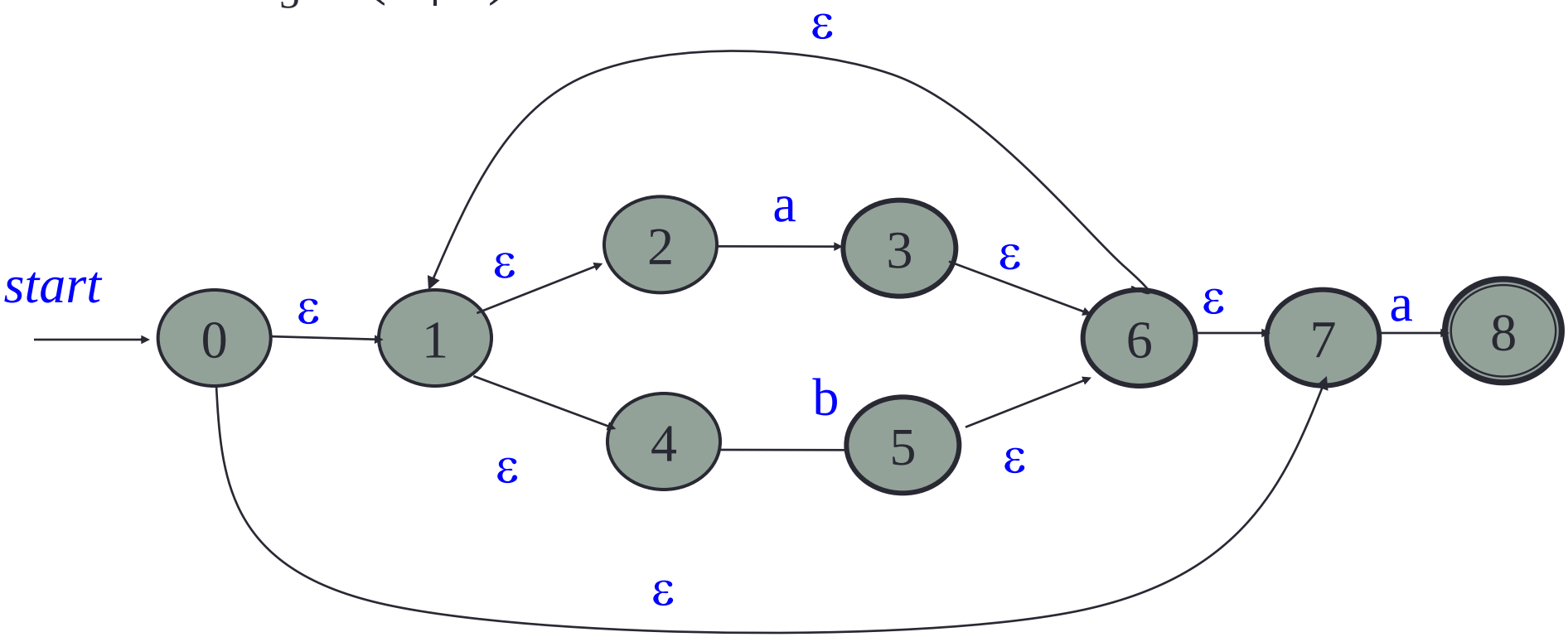
Primer: r=(a | b)*abb

$r_4 = (a \mid b)^*$



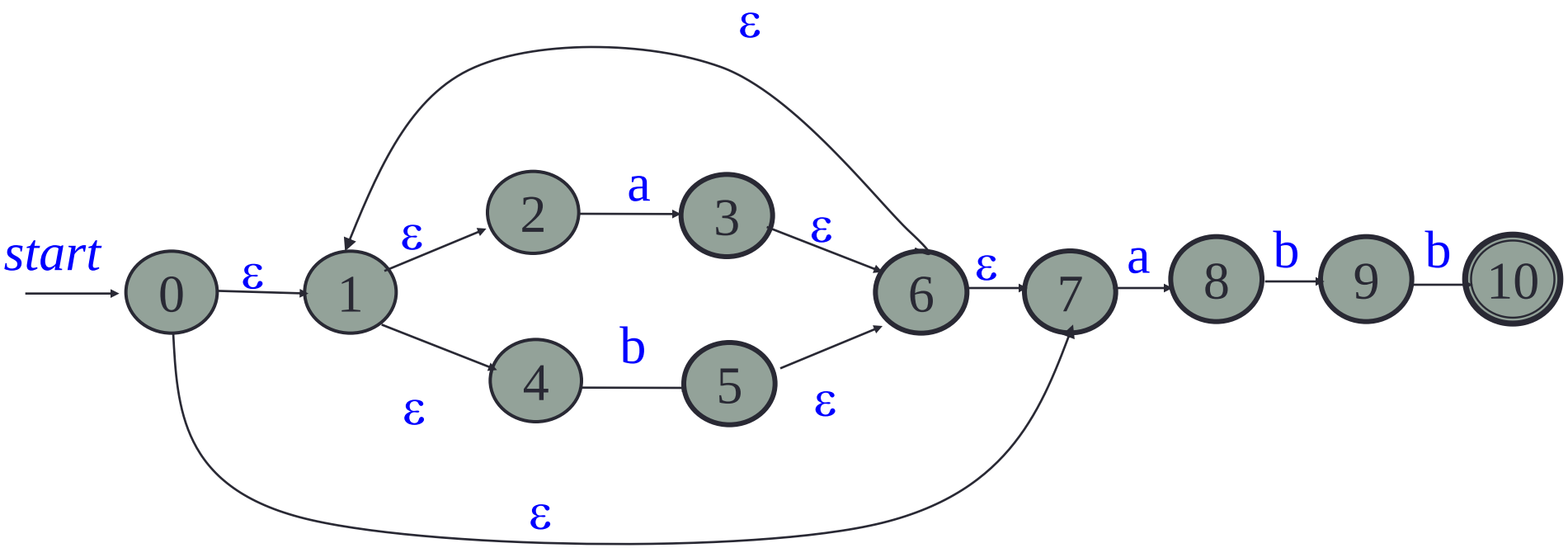
Primer: r=(a | b)*abb

$r_5 = (a \mid b)^*a$



Primer: r=(a | b)*abb

$r_6 = (a \mid b)^*abb$

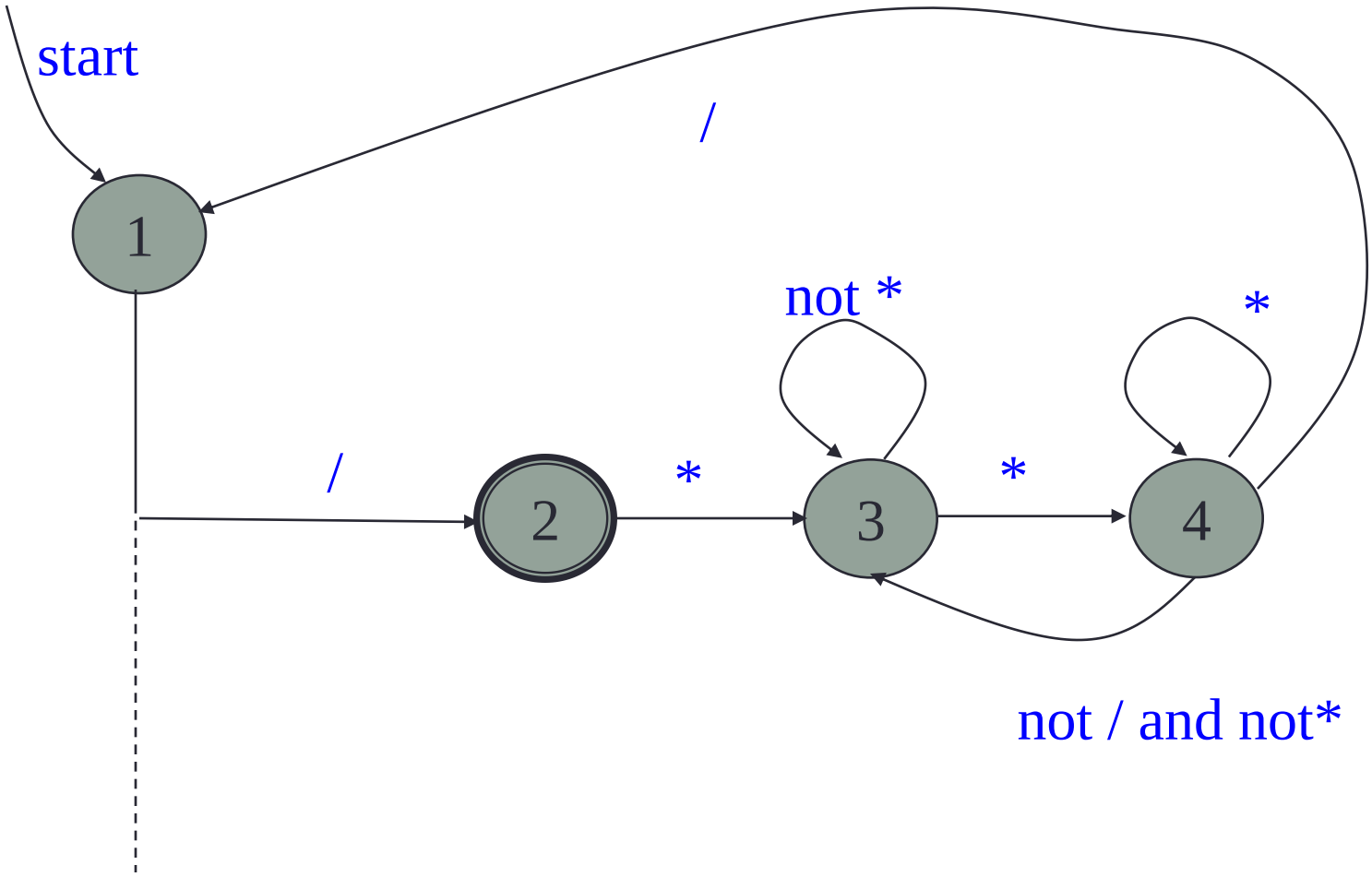


Primer leksičkog analizatora

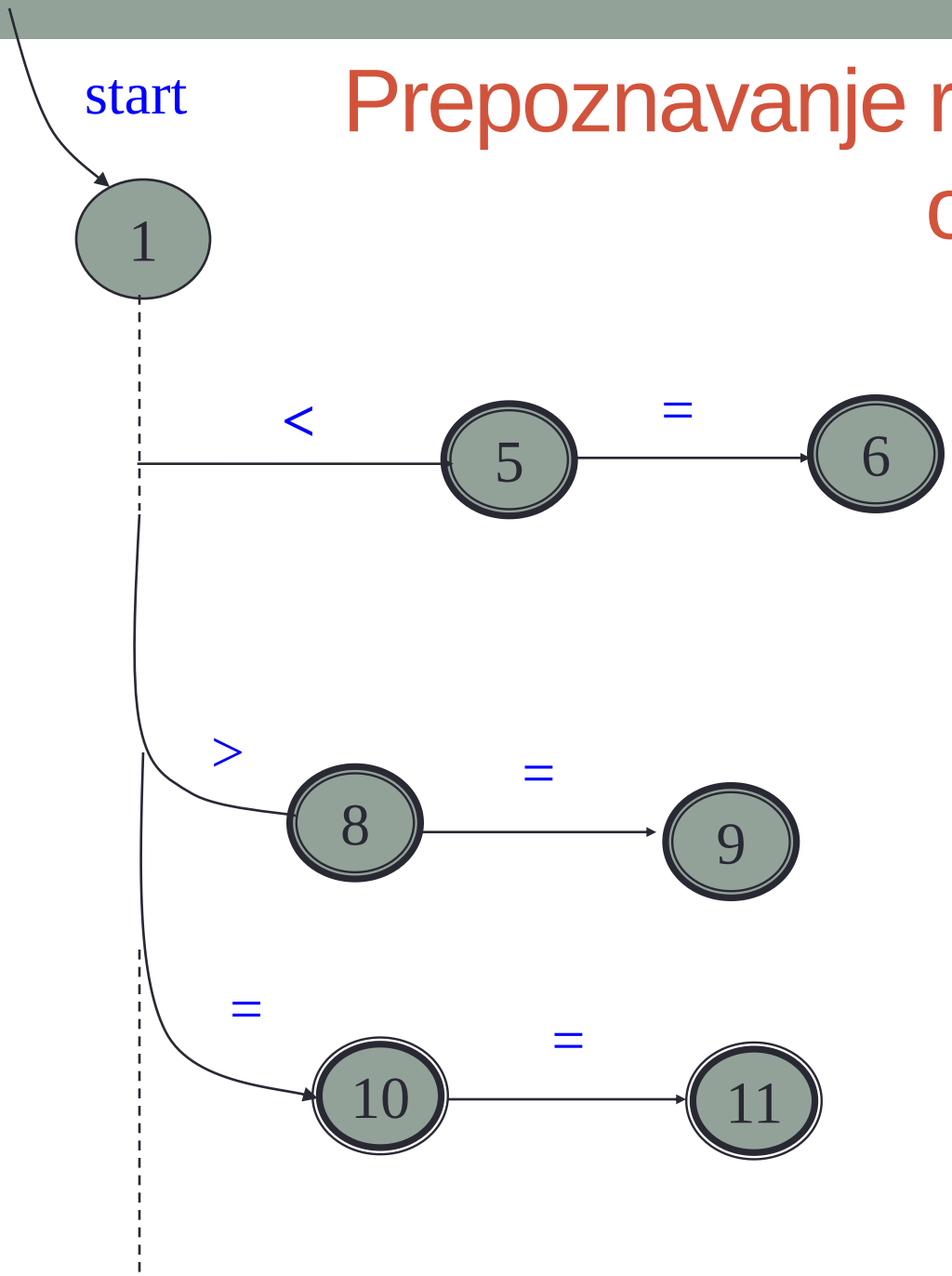
- Uzećemo kao primer jedan jednostavan jezik koji sadrži:
  - Komentare /*Ovo je komentar */
  - Relacione operatore: <, <=, ==, >, >=,
  - Aritmetičke operatore: + , - , / , *
  - Separatore = , ; , (,)
  - Ključne reči
  - Identifikatore

  Nizovi slova i cifara u kojima je prvi znak obavezno slovo
  - Literale (stringove, ‘Ovo je podatak’)
  - Numeričke konstante

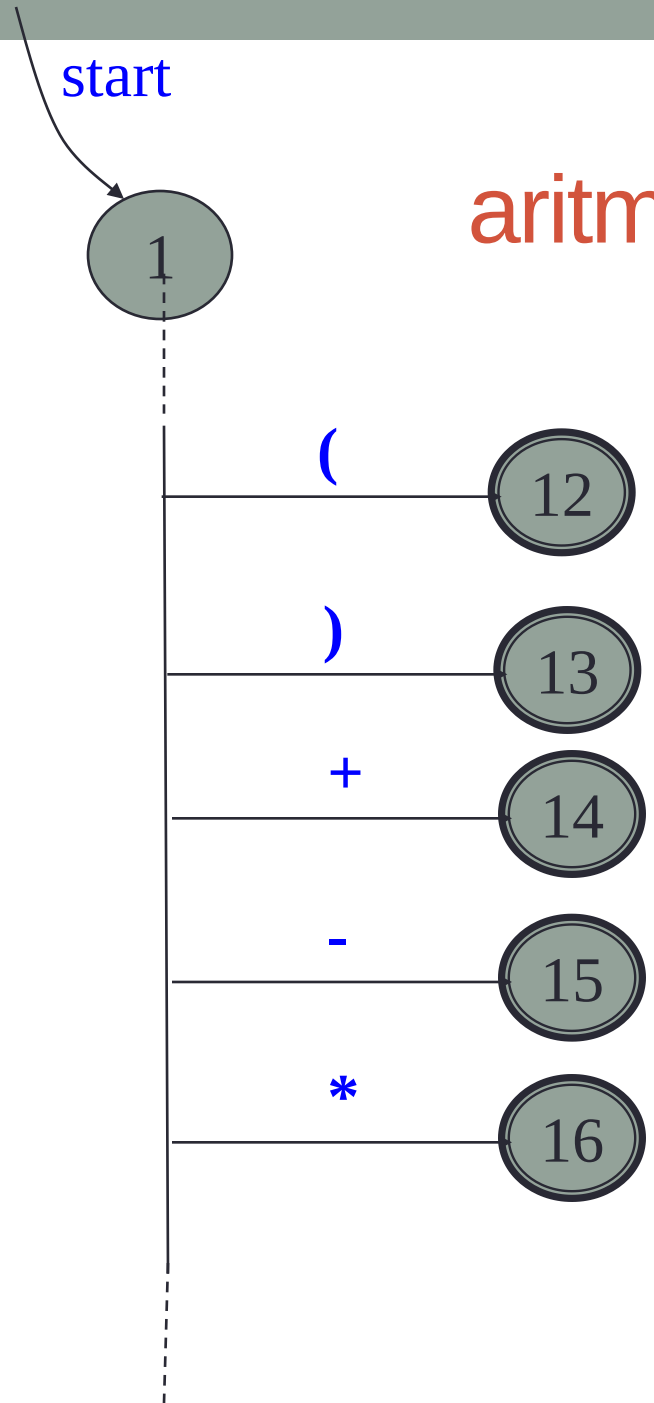
Deo automata za ignorisanje komentara



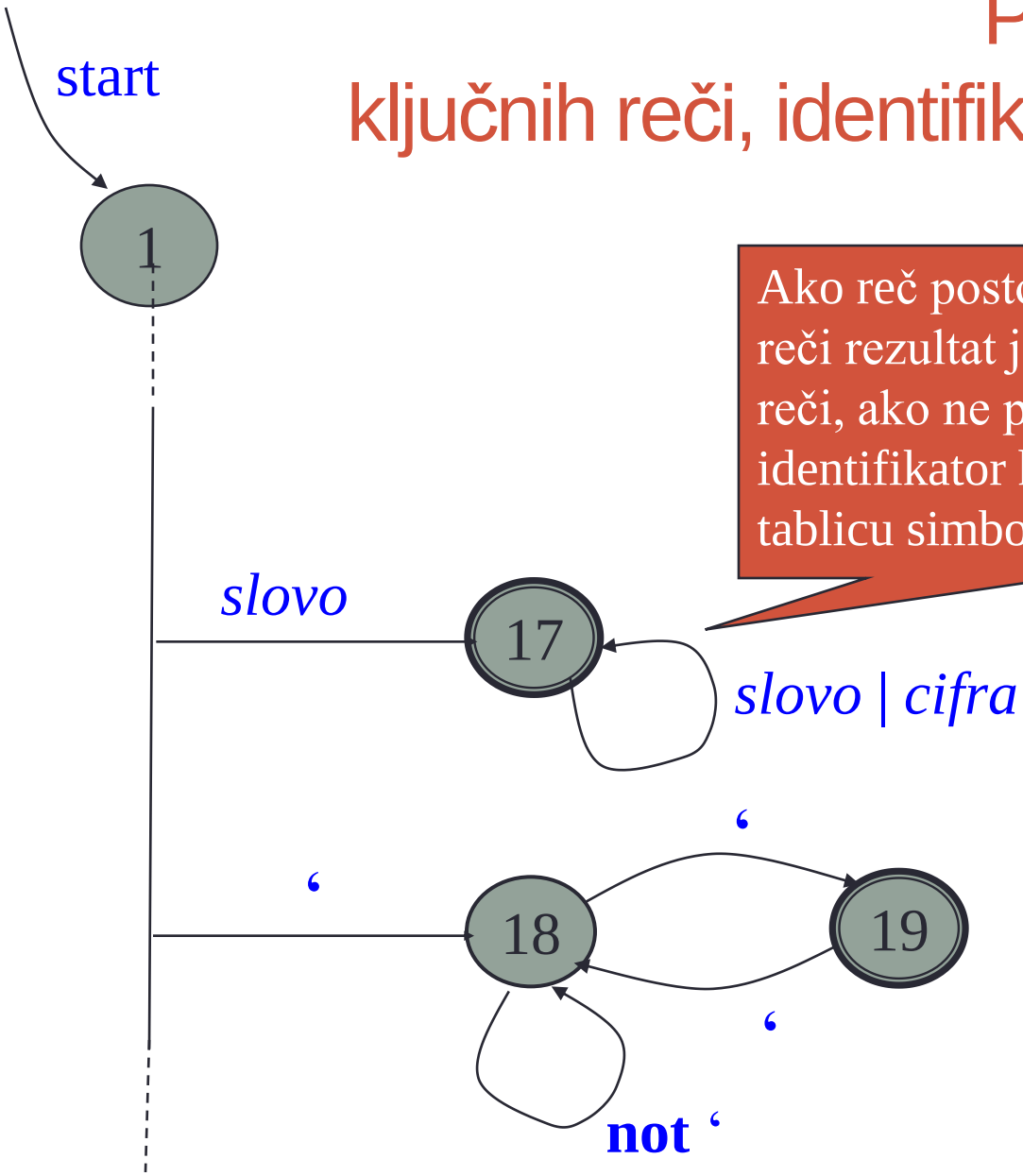
Prepoznavanje relacionih operatora



Prepoznavanje aritmetičkih operatora i separatora

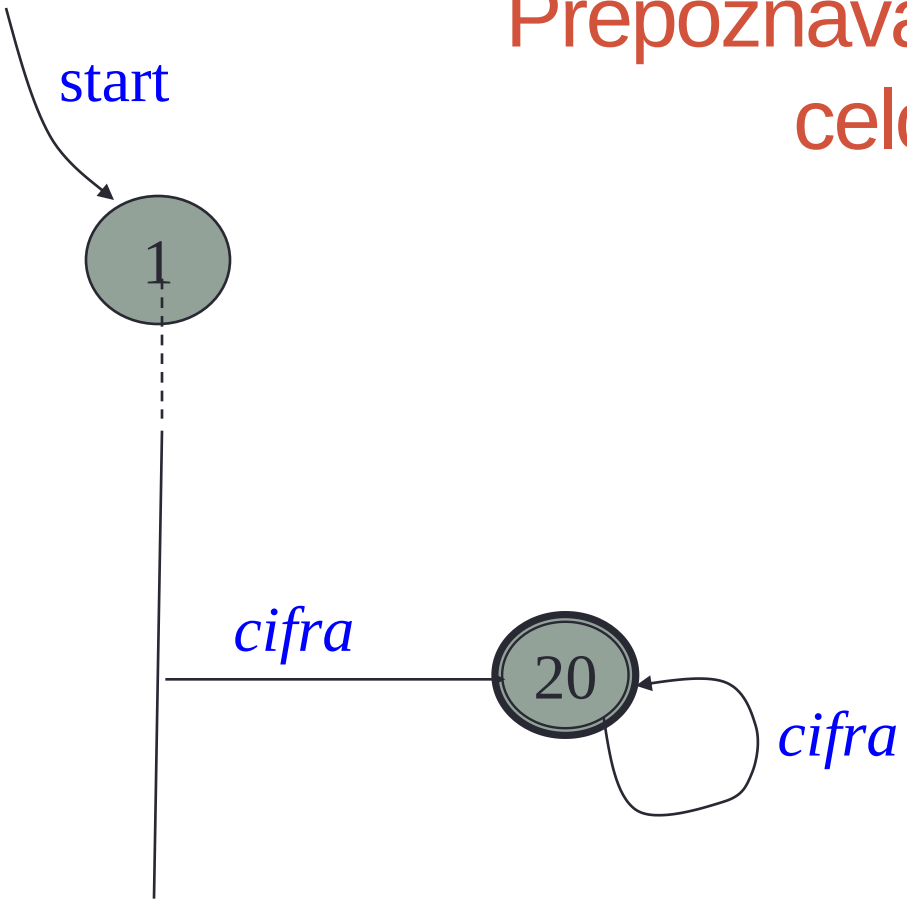


Prepoznavanje ključnih reči, identifikatora i literala

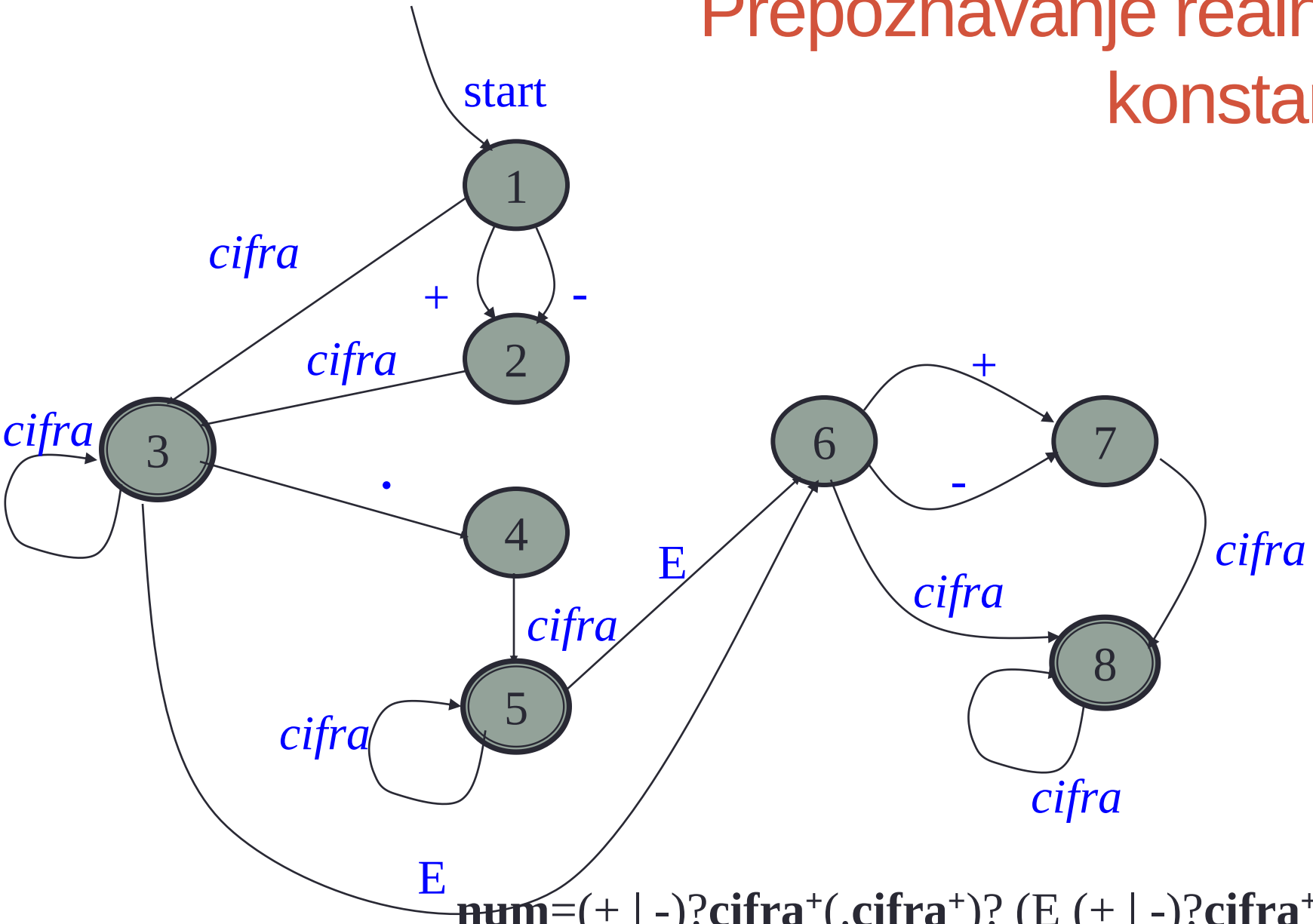


Ako reč postoji u tablici ključnih reči rezultat je značenje te ključne reči, ako ne postoji onda je to identifikator koji se upisuje u tablicu simbola.

Prepoznavanje neoznačenih celobrojnih constanti

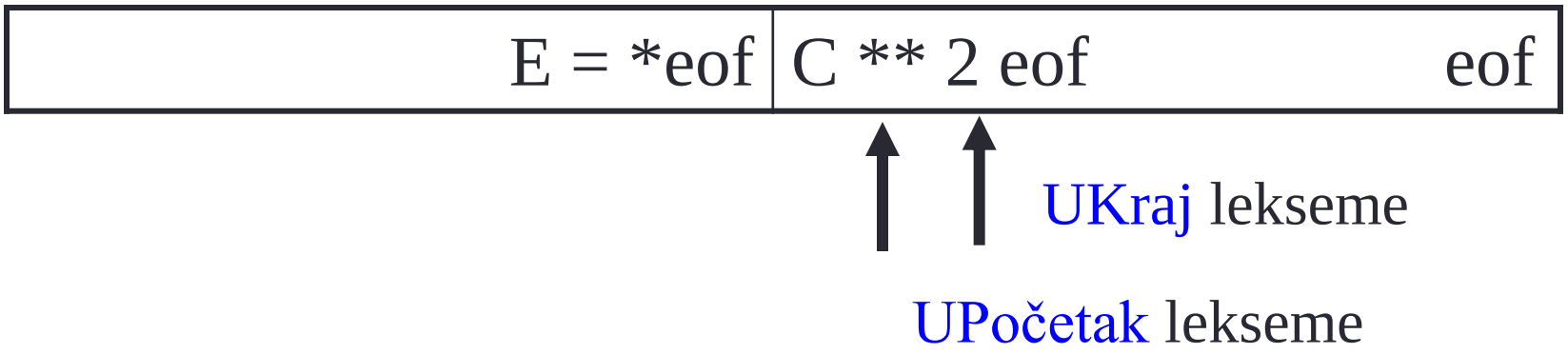


Prepoznavanje realnih konstanti



num=(+ | -)?**cifra**$^{+}$(.**cifra**$^{+}$)? (E (+ | -)?**cifra**$^{+}$)?

Realizacija leksičkog analizatora – tehnika dvostrukog baferovanja



Dužina bafera 1024 ili 4096 znaka

Realizacija leksičkog analizatora – tehnika dvostrukog baferovanja

```
Ukraj := Ukraj + 1;
if (Ukraj /= eof ) then begin
        if Ukraj na kraju prve polovine  then begin
                reload druge polovine;
                Ukraj := UKraj + 1;
        end
        else if  Ukraj na kraju druge polovine then  begin
            reload prve polovine;
            pomeriti UKraj na početak prve polovine
           end
   else Kraj lekseme
end
```

Leksičke greške

Leksički analizator može da otkriva samo ograničen broj grešaka zbog svog veoma lokalizovanog sagledavanja programa:

Tipične greške su:

- Dodavanje suvišnog znaka
- Izostavljanje znaka
- Zamena jednog znaka drugim
- Permutacija redosleda znakova

Leksičke greške - primer

Primer iz jezika C fi(a = = f(x))

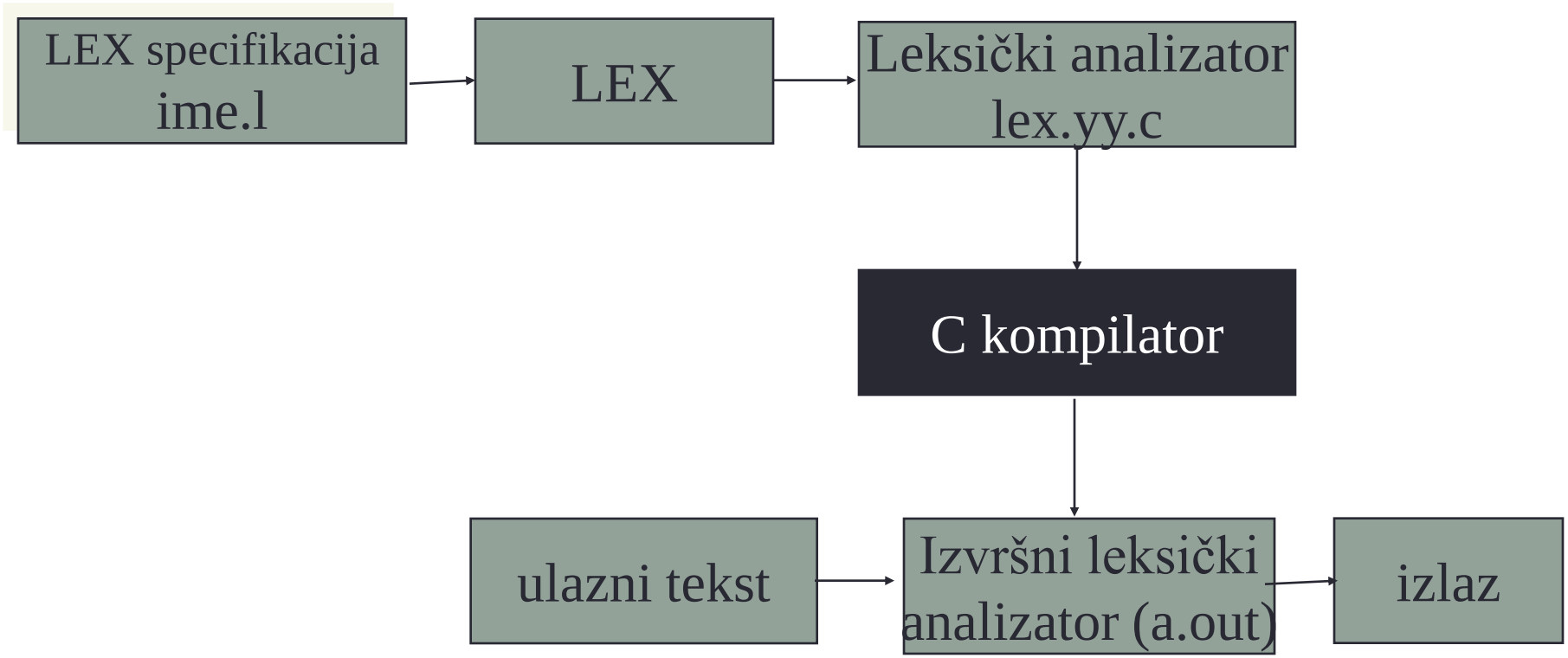
Leksički analizator ne može da utvrdi da li je fi permutovano if ili predstavlja ime funkcije tako da može da generiše dva simbola.

Generatori leksičkih analizatora

- Lex - lexical analyzer generator
- Flex (flex++) – fast lexical analyzer generator (open source)
- Jlex - lexical analyzer generator for java
- Jflex – unapredjenje Jlex-a

LEX

LEX


LEX specifikacija
ime.l
LEX
Leksički analizator
lex.yy.c
C kompilator
ulazni tekst
Izvršni leksički
analizator (a.out)
izlaz

Struktura LEX specifikacije

definicije
%%
pravila
%%
korisnički potprogrami

Pravila – Za svako završno stanje grafa konačnog automata kreira se jedno pravilo. Pravilo sadrži **uzorak** (šablon) koji se prepoznaje i **akciju** koja će se izvršiti kada se definisani uzorak pronađe.

Akcija se najčešće definišu kao pozivi funkcija koje su definisane u delu sa korisničkim potprogramima

Struktura LEX specifikacije

Definicije – Eksterne definicije, globalne promenlive, #include i #define naredbe Lokalne promenljive i zaglavlje koje se direktno umeće u generisani C- kod. Ovaj kod se piše između %{ i }%.

Korisnički potprogrami – potprogrami koji se pozivaju u delu sa akcijama.

LEX - zadavanje šablona
Regularni izrazi u lex specifikaciji

[ ]	Alternativa
( )	Grupisanje elemenata
+	Element uz koji stoji može da se ponovi više puta
?	Element koji mu prethodi može ali i ne mora da postoji
*	Element uz koji stoji može da se ponovi više puta i nijednom
-	Definiše interval unutar alternative
^	Početak reda
\	Ispred specijalnih znaka poništava njihovo „specijalno značenje“

Primer LEX specifikacije

Lex specifikacija analizatora koji prepoznaje celobrojne konstante u programskom jeziku C.

```
%%
[+-]?0|([1-9][0-9]*)     puts(“ Prepoznata dekadna konstanta”);
[+-]? 0[0-7]+            puts(“ Prepoznata oktalna konstanta”);
[+-]?0[xX][0-9A-Fa-f]+   puts(“ Prepoznata heksadekadna konstanta”);
```