
Januar 2015
RegexRangers
Nastevanje besed regularnega jezika
Tadej Borovšak
Aleš Omerzel

Kazalo

1. Osnovne operacije
2. Drevesna struktura
3. Prevedba v avtomat
4. Obiskovanje avtomata
5. Capturing Groups
6. Problemi
7. Počitnice

Uvod: Kaj so regularni izrazi?

- Zaporedje znakov
- Vzorec za niz

Operacija	Primer
*	ab* = a, ab, abb, …
+	ab+ = ab, abb, abbb, …
?	ab? = a, ab
|	a | b = a, b
{n}	ab{3} = abbb
{n,}	ab{2,} = abb, abbb, abbbb, …
{n, m}	ab{2,3} = abb, abbb

Email: /^([a-z0-9_\.-]+)@([\da-z\.-]+)\.([a-z\.]{2,6})$/

Naloga

- Naštevanje besed regularnega jezika
- Urejeno po
  - dolžini
  - abecedi
    - a, aa, ab, bbb, bcd, …
  - (a | b)*c

1. Osnovne operacije

Izraz	Opis
a	Znak
e*	Kleene closure
e1e2	Catenation
e1|e2	Alternation

Primer dveh enostavnih izrazov:

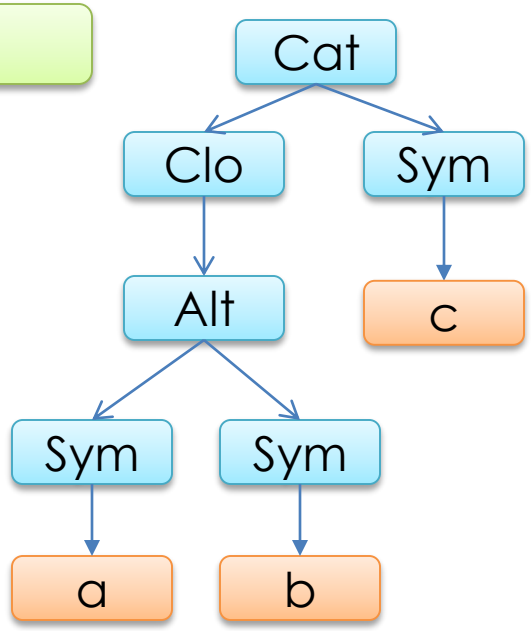
- a|ba*c
- (a|b)*c

2. Drevesna struktura

- Prevedba regularnega izraza v drevesno strukturo

(a | b)*c => Cat (Clo (Alt a b)) c

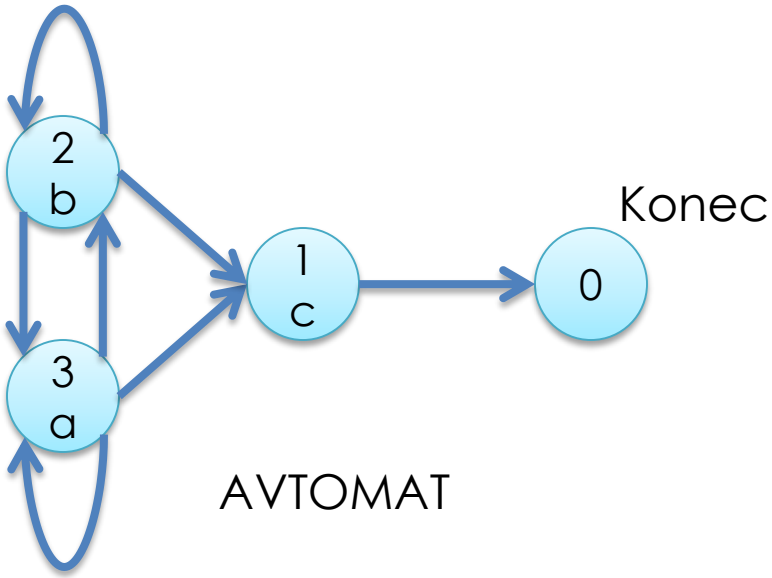
Izraz	Opis
∅	Nil
ε	Eps
a	(Sym "a")
e*	(Clo e)
e1e2	(Cat e1 e2)
e1 | e2	(Alt e1 e2)



- Parser

3. Prevedba v avtomat

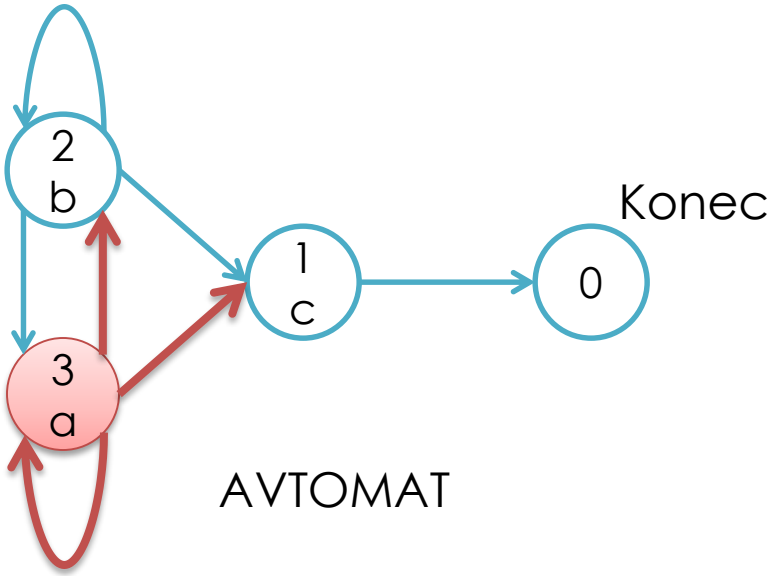
(a|b) * c



Kaj je avtomat? = graf

3. Prevedba v avtomat

(**a** | b) * c



3. Prevedba v avtomat

((abb)*|ba)* c|d*|(ab)*

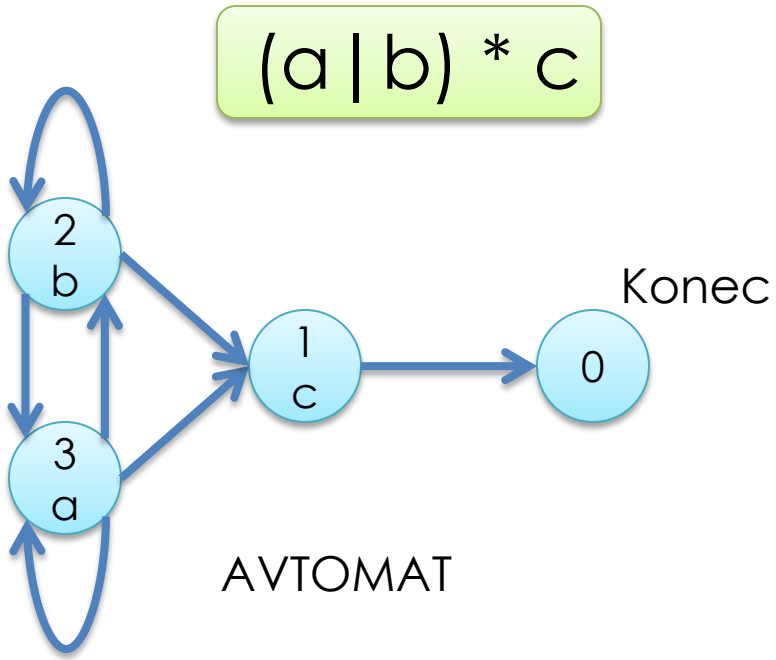
Prevedimo sedaj tole:

3. Prevedba v avtomat

((abb)* | ba)* c | d* | (ab)*

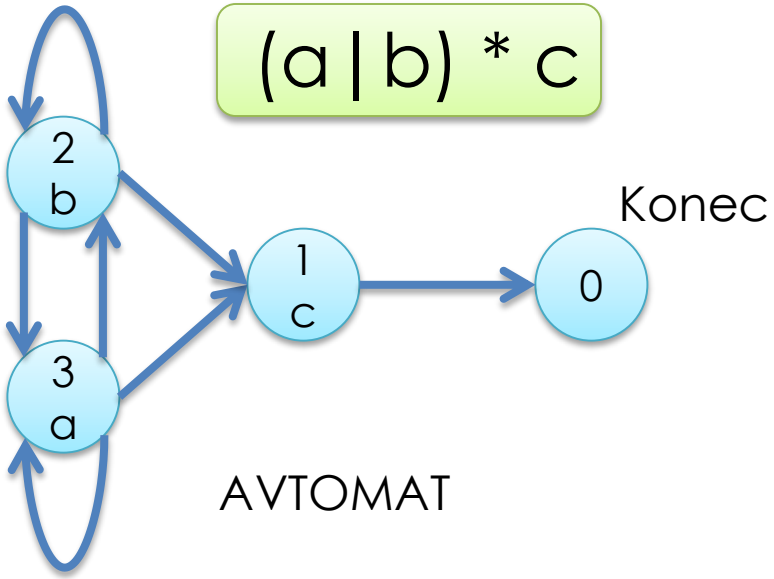
4. Obiskovalec avtomata

- = avanturist & printer
- Izpisuje besede urejene po **dolžini** in **abecedi** (CILJ?)

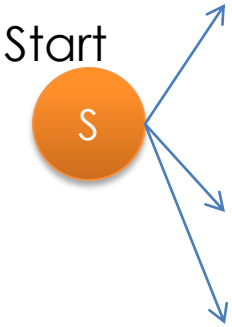


Izpis: [c, ac, bc, aac, abc, …]

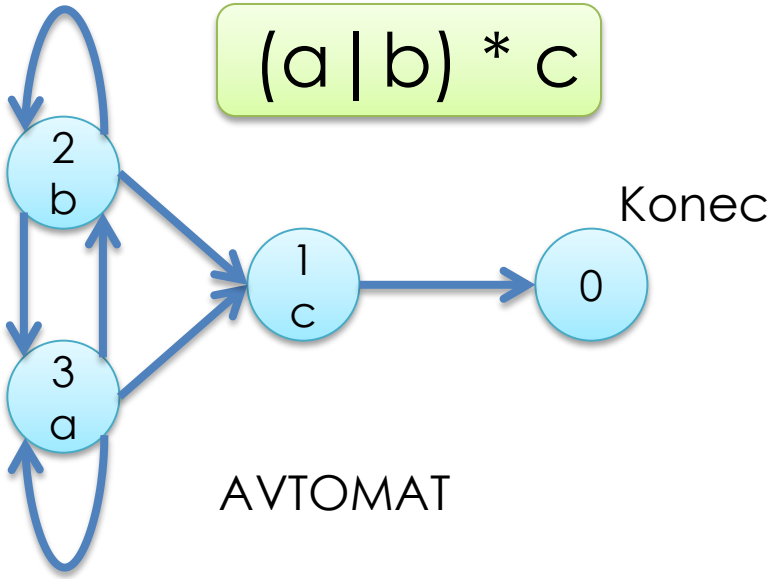
4. Obiskovalec avtomata



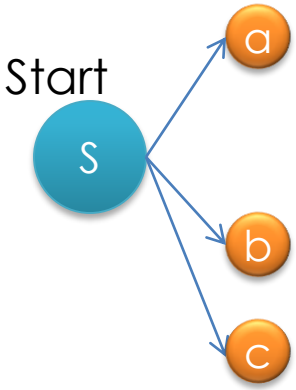
Vrsta stanj:
Izpis:



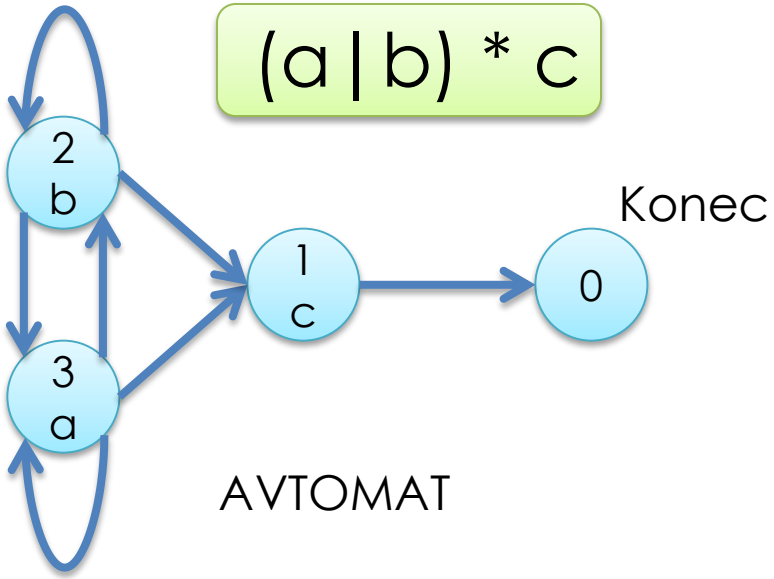
4. Obiskovalec avtomata



Vrsta stanj: **+ Stanje a, Stanje b, Stanje c**
Izpis:

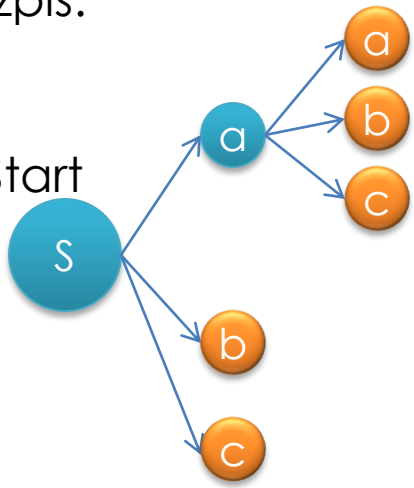


4. Obiskovalec avtomata

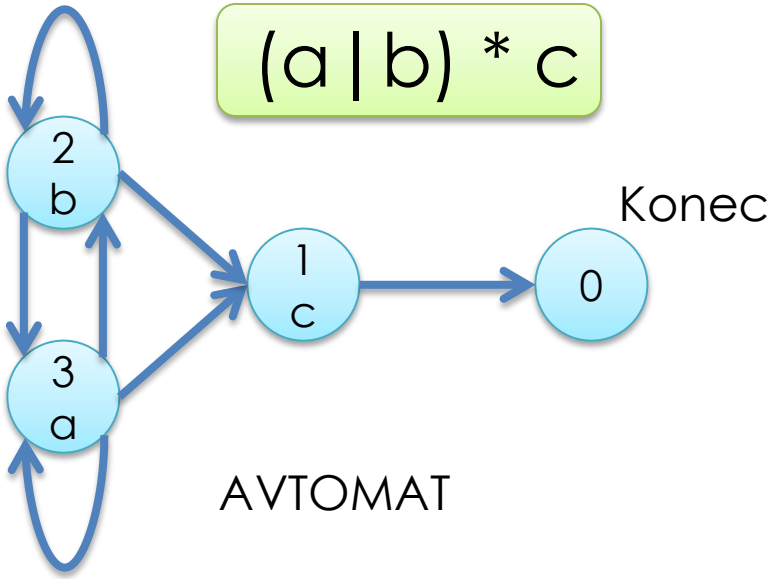


Vrsta stanj: ~~Stanje a~~, Stanje b, Stanje c, **+ Stanje a, Stanje b, Stanje c**

Izpis:

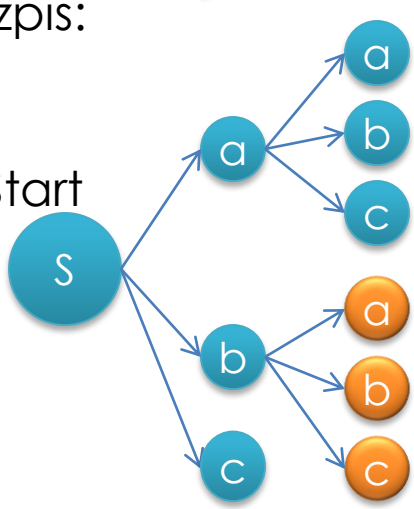


4. Obiskovalec avtomata



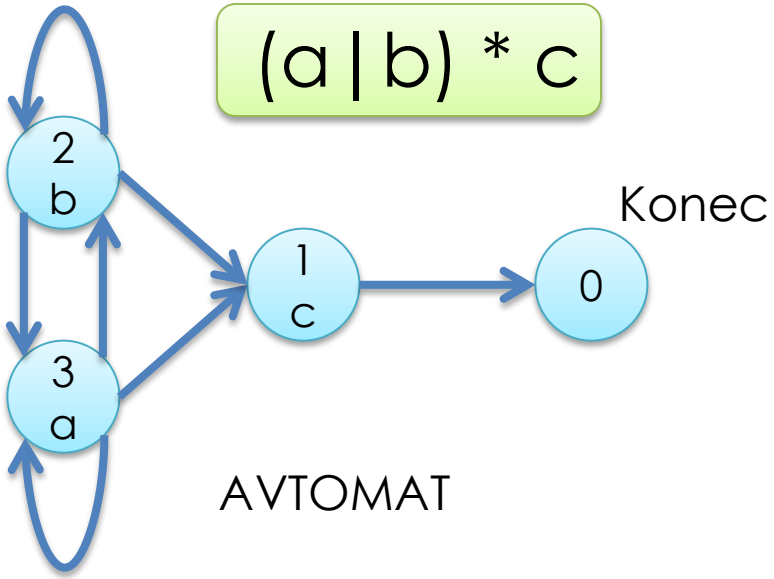
Vrsta stanj: ~~Stanje b, Stanje c, Stanje a, Stanje b, Stanje c,~~ **+ Stanje a, Stanje b, Stanje c**

Izpis:



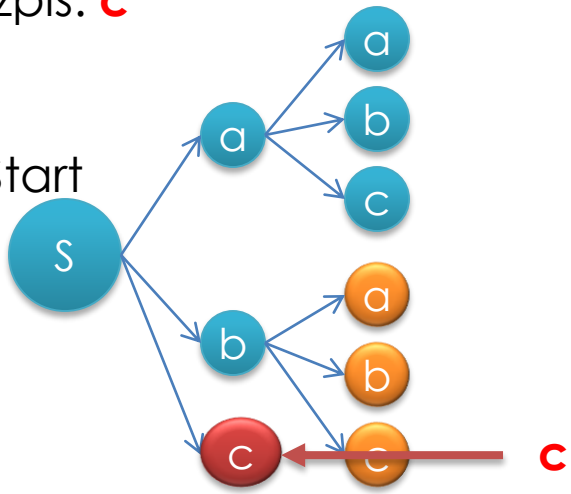
...

4. Obiskovalec avtomata



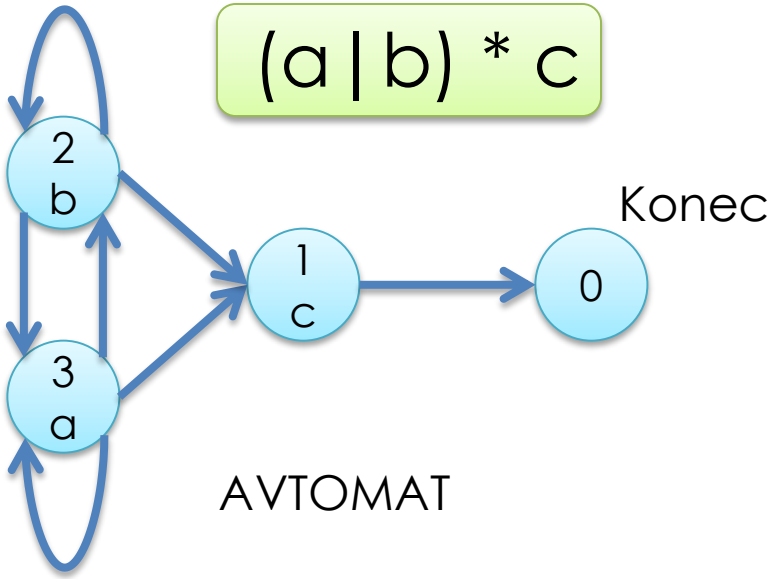
Vrsta stanj: ~~Stanje c~~, Stanje a, Stanje b, Stanje c, **+ Stanje a, Stanje b, Stanje c**

Izpis: **c**



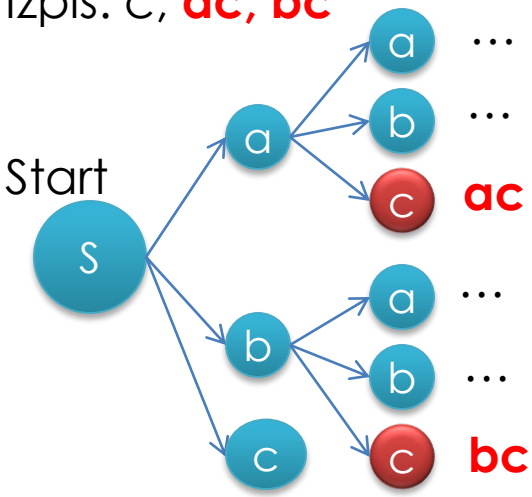
...

4. Obiskovalec avtomata



Vrsta stanj: Stanje a, Stanje b, Stanje c, **+ Stanje a, Stanje b, Stanje c**

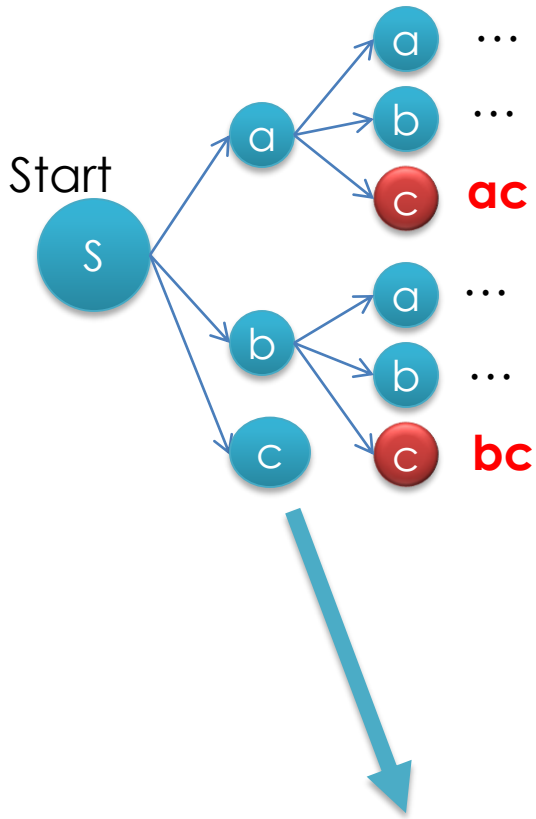
Izpis: c, **ac, bc**



Opomba

(a|b)*c

Novo besedo dobimo, ko pridemo do **c**.



(**c**|b)*c

- Vsak c ima svoje stanje
- Indeksiranje

data State = State Ident Action NFA deriving Show

Demonstracija

- enumerate “ab*”
- enumerate “a|b”
- enumerate “(a|b)*”

Kaj še?

- Izpis dreves – po dolžini in abecedi
- Ne da zapisati z avtomatom
  - Lema o napihovanju
    - Ni konteksno neodvisna gramatika
    - Zato ne obstaja avtomat


Capturing Groups

Capturing Groups???

(ab) b (c) **1** = abcab

(ab) b (c) **2** = abcc

- Kopiranje podniza
- Gnezdenje?

Kako deluje?

- Obiskovalec vozlišč
  - Špega po znakih

NEW

if "**(**" then "start a new group"
if "**)**" then "close *the last* group"
if "**2**" then return group 2

NEW

1) memory za vse grupe
2) memory za trenutne grupe (gnezdenje)
   - za dodajanje znaka vsem trenutnim grupam

(a(b))

Memory grup: ab, b
Memory trenutnih: 1, 2 (če na poziciji b)

Problemi

- a(b)1* = ab**bbb**... ? ab**111**...
- (a(b)*)2 = a b..b **b** ? a b..b **b..b**
- (a)(1)2 = a a **a** (run time) ? a a **1** (compile time)

- (a | b*)1 = ali je še urejen po dolžini?
- (aaa | b*)1 = ali je še urejen po dolžini?
- a(bb)(1 | a) = ali je še urejen po dolžini?

- Kako pa zapisati številke v reg. izrazu? 5, 9, 14

Popravilo

1. En memory (prepisovanje vsebine grupe)
2. a(..)+ => a **bc de**
   - Množenje števila iste grupe = napaka
3. Char -> data Action = Symbol Char
   | Open
   | Close
   | Ref Int
   | Accept
4. Popravilo parserja
5. Obiskovalec avtomata prilagoditi na Action


Hollidays

Za konec še …

- Testiranje
- Primeri
- Dokumentacija
- Program za konzolo


Hvala za pozornost