Programiranje I

Beleške sa vežbi

Smer *Informatika*
Matematički fakultet, Beograd

Sana Stojanović

November 1, 2007

1 URM mašine

Zadatak 1 *Napisati* URM *program koji izračunava funkciju*

$$f(x,y) = \begin{cases} 0 & , \textit{ako } x \leq y \\ 1 & , \textit{inače} \end{cases}$$

Rešenje:
Predloženi algoritam se zasniva na sledećoj osobini:

$$x \leq y \Leftrightarrow (\exists k)(k \in \mathbf{N}) \;\; y = x + k$$

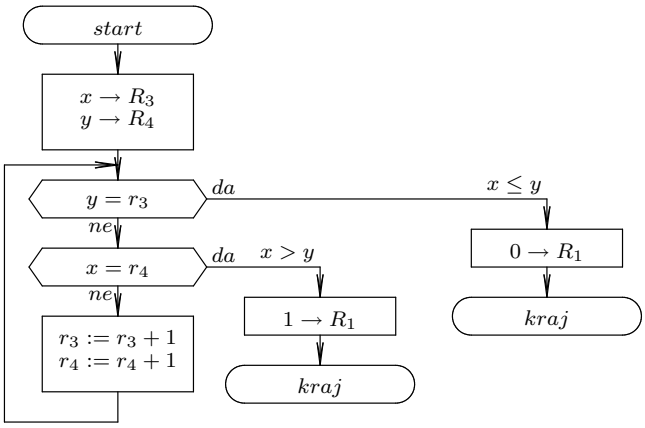
Dakle, vrednosti x, odnosno y se sukcesivno dodaje vrednost 1 sve dok se ne dostigne y, odnosno x. Prva dostignuta vrednost predstavlja broj ne manji od onog drugog. U skladu sa tim zaključkom i definicijom funkcije f, izračunata vrednost je 0 ili 1. Odgovarajući URM program podrazumeva sledeću početnu konfiguraciju:

R_1	R_2	R_3	$\ldots$
x	y	0	$\ldots$

i sledeću radnu konfiguraciju:

R_1	R_2	R_3	R_4	$\ldots$
x	y	$x+k$	$y+k$	$\ldots$

gde k dobija redom vrednosti $0, 1, 2 \ldots$.

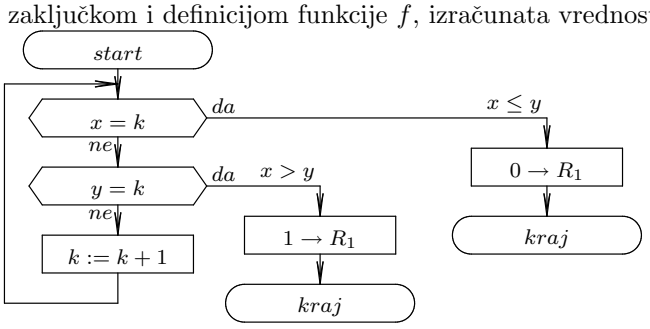


1.	$T(1,3)$	$x \to R_3$
2.	$T(2,4)$	$y \to R_4$
3.	$J(2,3,8)$	$y = r_3$?
4.	$J(1,4,10)$	$x = r_4$?
5.	$S(3)$	$r_3 := r_3 + 1$
6.	$S(4)$	$r_4 := r_4 + 1$
7.	$J(1,1,3)$	
8.	$Z(1)$	$0 \to R_1$
9.	$J(1,1,100)$	kraj
10.	$Z(1)$	
11.	$S(1)$	$1 \to R_1$

Isti problem može biti rešen i na drugi način. Alternativni URM program koristi sledeću radnu konfiguraciju:

R_1	R_2	R_3	...
x	y	k	...

gde k dobija redom vrednosti $0, 1, 2 \ldots$ sve dok ne dostigne vrednost x, odnosno y. Prva dostignuta vrednost predstavlja broj ne manji od onog drugog. U skladu sa tim zaključkom i definicijom funkcije f, izračunata vrednost je 0 ili 1.



1.	$J(1,3,5)$	$x = k$?
2.	$J(2,3,7))$	$y = k$?
3.	$S(3)$	$k := k + 1$
4.	$J(1,1,1)$	
5.	$Z(1)$	$0 \to R_1$
6.	$J(1,1,100)$	kraj
7.	$Z(1)$	
8.	$S(1)$	$1 \to R_1$

Zadatak 2 (DOMAĆI)

(a) Ako je data gramatika $G = (N, \Sigma, P, S)$*, gde je:*
$N = \{S, A, B\}$,
$\Sigma = \{a, b, c\}$,
$P = \{S \to AcB \ \ (1°), A \to aB \ \ (2°), A \to c \ \ (3°), B \to bA \ \ (4°), B \to c \ \ (5°)\}$

ispitati da li se reč `abccc` *može izvesti u gramatici* G *i ako može prikazati njeno izvođenje (tj. navesti redom sva pravila koja dovode do te reči,* $S \to AcB \to \ldots \to abccc$*).*

(b) Napisati URM *program koji izračunava funkciju*

$$f(x,y) = \begin{cases} x - y & , \text{ ako } x > y \\ 0 & , \text{ inače} \end{cases}$$

Zadatak 3 *Napisati* URM *program koji izračunava funkciju*

$$f(x,y) = xy$$

Rešenje:

Predloženi algoritam se zasniva na sledećoj osobini:

$$xy = \underbrace{x + (\overbrace{1 + \ldots + 1}^{x}) + \ldots + (\overbrace{1 + \ldots + 1}^{x})}_{y}$$

Odgovarajući URM program podrazumeva sledeću početnu konfiguraciju:

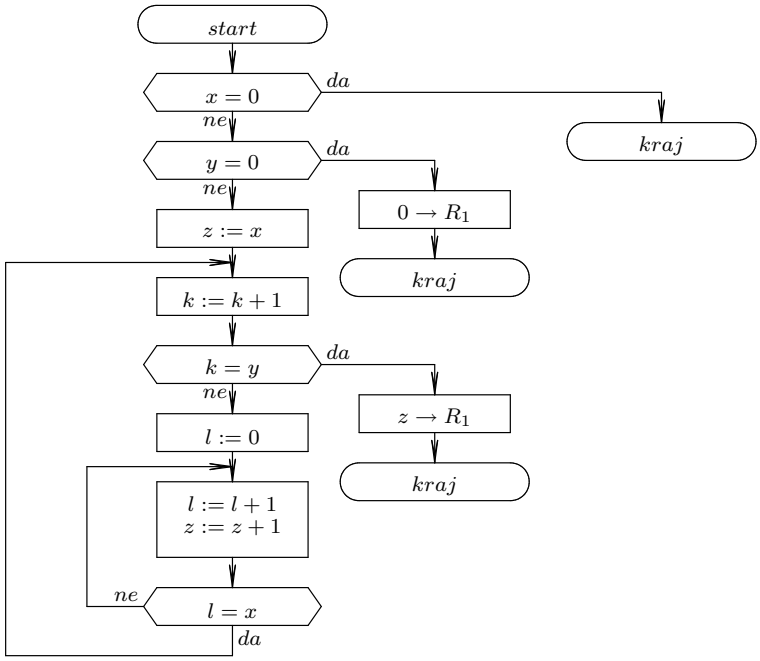
R_1	R_2	R_3	$\ldots$
x	y	0	$\ldots$

i sledeću radnu konfiguraciju:

R_1	R_2	R_3	R_4	R_5	$\ldots$
x	y	z	k	l	$\ldots$

gde k dobija redom vrednosti $0, 1, \ldots, y$, a za svaku od ovih vrednosti l dobija redom vrednosti $0, 1, \ldots, x$.

1.	$J(1,10,100)$	ako je $x=0$, onda kraj
2.	$J(2,10,13)$	$y=0$?
3.	$T(1,3)$	$z := x$
4.	$S(4)$	$k := k+1$
5.	$J(4,2,11)$	$k=y$?
6.	$Z(5)$	$l := 0$
7.	$S(5)$	$l := l+1$
8.	$S(3)$	$z := z+1$
9.	$J(5,1,4)$	
10.	$J(1,1,7))$	
11.	$T(3,1)$	$z \to R_1$
12.	$J(1,1,100)$	
13.	$Z(1)$	$0 \to R_1$



Zadatak 4 *Napisati* URM *program koji izračunava funkciju*

$$f(x) = 2^x$$

Zadatak 5 *Napisati* URM *program koji izračunava funkciju*

$$f(x,y) = \begin{cases} 1 & , \textit{ ako } x|y \\ 0 & , \textit{ inače} \end{cases}$$

Zadatak 6 *Napisati* URM *program koji izračunava funkciju*

$$f(x) = [\sqrt{x}]$$

Rešenje:

Predloženi algoritam se zasniva na sledećoj osobini:

$$n = [\sqrt{x}] \Leftrightarrow n^2 \leq x < (n+1)^2$$

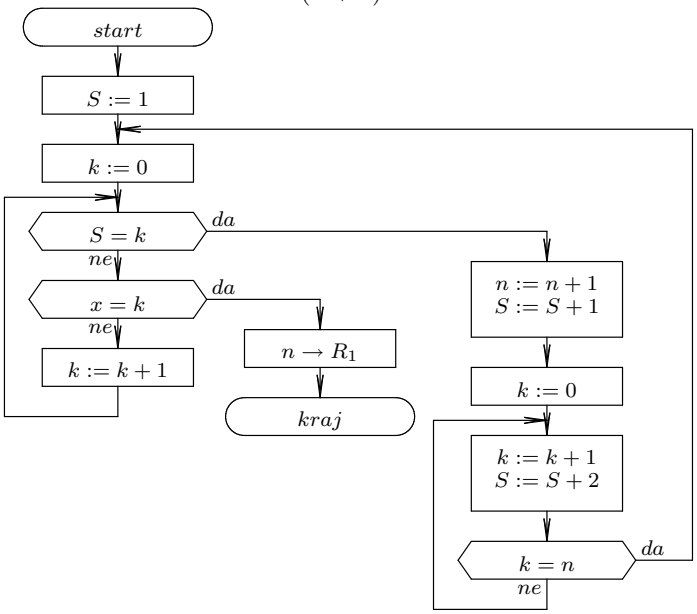
Odgovarajući URM program podrazumeva sledeću početnu konfiguraciju:

R_1	R_2	...
x	0	...

i sledeću radnu konfiguraciju:

R_1	R_2	R_3	R_4	...
r_1	r_2	$S = (n+1)^2$	k	...

gde za vrednosti n i S važi $S = (n+1)^2$.



1.	$S(3)$	$S := 1$
2.	$Z(4)$	$k := 0$
3.	$J(3,4,7)$	$S = k$?
4.	$J(1,4,15)$	$x = k$?
5.	$S(4)$	$k := k + 1$
6.	$J(1,1,3)$	
7.	$S(2)$	$n := n + 1$
8.	$S(3)$	$S := S + 1$
9.	$Z(4)$	$k := 0$
10.	$S(4)$	$k := k + 1$
11.	$S(3)$	$S := S + 1$
12.	$S(3)$	$S := S + 1$
13.	$J(4,2,2)$	$k = n$?
14.	$J(1,1,10)$	
15.	$T(2,1)$	$n \rightarrow R_1$

Zadatak 7 *Napisati* URM *program koji izračunava funkciju*[1]

$$f(x) = \begin{cases} x/3 & , \textit{ako } 3|x \\ \uparrow & , \textit{inače} \end{cases}$$

Zadatak 8 *Napisati* URM *program koji izračunava funkciju* $f(x) = \left[\frac{2x}{3}\right]$.

Zadatak 9 *Napisati* URM *program koji izračunava funkciju*

$$f(x,y) = \begin{cases} \left[\frac{y}{x}\right] & , \textit{ako } x \neq 0 \\ \uparrow & , \textit{inače} \end{cases}$$

Zadatak 10 *Napisati* URM *program koji izračunava funkciju* $f(x) = x!$.

Zadatak 11 *Napisati* URM *program koji izračunava funkciju*

$$f(x,y,z) = \begin{cases} \left[\frac{y}{3}\right] & , \textit{ako } 2|z \\ x+1 & , \textit{inače} \end{cases}$$

2 Specifikacija sintakse programskih jezika, meta jezici

Za opis programskih jezika često se koriste kontekstno-slobodne gramatike. Bekus-Naurova forma (BNF) je konvencija za zapisivanje pravila kontekstno-slobodnih jezika. Proširena Bekus-Naurova forma (EBNF) dodaje određene sintaksičke izraze BNF notaciji i omogućava jednostavniji zapis pravila gramatike.

2.1 BNF (Backus-Naur form)

Meta jezik je jezik koji služi da se pomoću njega opiše neki drugi jezik ili isti taj jezik. Tako se na primer služimo srpskim jezikom da bismo opisali gramatiku srpskog jezika. Bitno je razlikovati term meta jezika od terma jezika koji se opisuje.

BNF(Bekus-Naurova forma) je formalni meta jezik za predstavljanje kontekstno-slobodnih gramatika odnosno gramatika programskih jezika.

U BNF se koriste sledeće konvencije za zapisivanje pravila gramatike:

- Umesto simbola $\rightarrow$ koristi se simbol `::=`
- Pomoćni simboli se navode među zagradama $\langle$ i $\rangle$
- Vertikalna crta `|` (izbor) razdvaja desne strane pravila koja odgovaraju istom simbolu sa leve strane pravila
- Prazna niska ε se zapisuje kao $\langle$empty$\rangle$.

[1] Ukoliko je funkcija nedefinisana za neke vrednosti argumenta, umesto odgovarajuće vrednosti funkcije pisaćemo $\uparrow$.

Sledi nekoliko primera BNF-a :

- BNF za cifru:

```
<cifra> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
```

- BNF za neoznačen ceo broj:

```
<NeoznacenCeoBroj> ::=  <cifra> | <cifra> <NeoznacenCeoBroj>
<cifra> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
```

  Drugo pravilo može da se napiše i kao:

```
<NeoznacenCeoBroj> ::=  <cifra> | <NeoznacenCeoBroj> <cifra>
```

- BNF za ceo broj:

```
<CeoBroj> ::= <NeoznacenCeoBroj>
            | + <NeoznacenCeoBroj>
            | - <NeoznacenCeoBroj>
<NeoznacenCeoBroj> ::=  <cifra> | <cifra> <NeoznacenCeoBroj>
<cifra> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
```

- BNF za realne brojeve:

```
<RealanBroj>  ::= - <NeoznacenRealanBroj>
                | + <NeoznacenRealanBroj>
                | <NeoznacenRealanBroj>
<NeoznacenRealanBroj> ::= <NeoznacenCeoBroj>
                        | <NeoznacenCeoBroj> . <NeoznacenCeoBroj>
<NeoznacenCeoBroj> ::=  <cifra> | <cifra> <NeoznacenCeoBroj>
<cifra> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
```

- BNF za identifikator:

```
<identifikator> ::= <slovo>
                  | <identifikator> <slovo>
                  | <identifikator> <cifra>
<cifra> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
```

2.2 EBNF (Extended Backus-Naur form)

EBNF (proširena Bekusova normalna forma) je takođe meta jezik za predstavljanje kontekstno-slobodnih gramatika koji ima istu izražajnu moć kao i BNF, samo je zapis takav da je lakši za razumevanje. Zapravo, BNF koristi rekurziju da bi se izrazile relacije a EBNF koristi iteraciju.

Skup meta simbola BNF-a proširen je sa sledećim meta simbolima:

- Pravougaone zagrade "[..]" označavaju da se ono što se nalazi u njima pojavljuje opciono (ili se pojavljuje jednom ili se ne pojavljuje uopšte).

  Drugi način da se opiše nula ili jedno pojavljivanje simbola je korišćenje sufiksa "?".

- sufiks "*" označava da se simbol pojavljuje 0 ili više puta.
  Istu ulogu imaju i vitičaste zagarade "{..}".

- sufiks "+" za jedno ili više pojavljivanja simbola

- super/subscripts označava između m i n pojava simbola.

- završni simboli se navode između navodnika (na primer, "1", "2")

Sve ove konstrukcije mogu biti izražene i u BNF-u što je pokazatelj toga da sve što se može zapisati u EBNF-u može i u BNF-u.

Sledi nekoliko primera EBNF-a :

- EBNF za neoznačen ceo broj.

  ```
  <NeoznacenCeoBroj> ::= ( <cifra> ) + ;

  <cifra> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" |
  "0" ;
  ```

- EBNF za ceo broj.

  ```
  <CeoBroj> ::= [ "+" | "-" ] <NeoznacenCeoBroj>; <NeoznacenCeoBroj>
  ::= ( <cifra> )+; <cifra> ::= "1" | "2" | "3" | "4" | "5" | "6" |
  "7" | "8" | "9" | "0" ;
  ```

  Pri čemu se prva dva pravila mogu napisati i kao:

  ```
  <CeoBroj> ::= [ "+" | "-" ] ( <cifra> ) +
  ```

- EBNF za realne brojeve.

  ```
  <RealanBroj> ::= [ "+" | "-"] <cifra>+ ("." <cifra>+)? <cifra> ::=
  "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "0" ;
  ```

- EBNF za identifikator.

  ```
  <identifikator> ::= <slovo>(<slovo>|<cifra>)*
  ```

  ili

  ```
  <identifikator> ::= <slovo> { <slovo> | <cifra> }
  ```

Zadatak 1 *napisati BNF/EBNF za aritmeticki izraz i sl.*

Rešenje:
BNF:

```
<ArIzraz> ::= <term> | <ArIzraz> "+" <term> <term> ::= <factor> |
<term> "*" <factor> <factor> ::= <var> | "(" <ArIzraz> ")" <var>
::= <identifikator> | <RealanBroj>
```

EBNF:

```
<ArIzraz> = <term> { "+" <term> } <term> = <factor> { "*"
<factor>} <factor> = <cifra> | "(" <ArIzraz> ")"
```

Zadatak 2 *napisati BNF/EBNF za klauzu (nad nekim fiksnim skupom iskaznih slova)*

Zadatak 3 *napisati BNF/EBNF za klauzu duzine npr. 5 (nad nekim fiksnim skupom iskaznih slova); mozda ih ovo zbuni posle ovog prvog, ali zapravo dovoljno je:*

```
<klauza> ::== <literal> \/ <literal> \/ <literal> \/ <literal> \/
<literal>
```

Zadatak 4 *Napisati BNF/EBNF za iskaznu formulu (nad nekim fiksnim skupom iskaznih slova)*

2.3 Sintaksni dijagrami

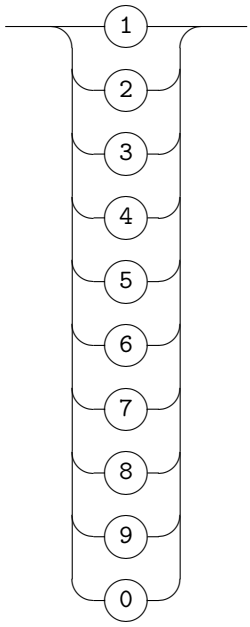
Sintaksni dijagrami predstavljaju grafičku notaciju za predstavljanje kontekstnoslobodnih gramatika. To su dijagrami slični dijagramima toka. Čitanje sintaksnog dijagrama znači kretanje od leve ka desnoj strani prateći strelice. Ono što je bitno to je da oni imaju istu izražajnu moć kao i BNF ili EBNF.

Pogledajmo kako izgledaju sintaksni dijagrami nekoliko već pomenutih pravila:

- Sintaksni dijagram za pomoćni simbol `cifra`:

  ```
  <cifra> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" |
  "0" ;
  ```

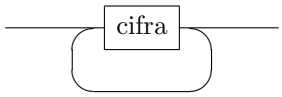
cifra



- Sintaksni dijagram za pomoćni simbol `NeoznacenCeoBroj`:

  ```
  <NeoznacenCeoBroj> ::= ( cifra ) +
  ```

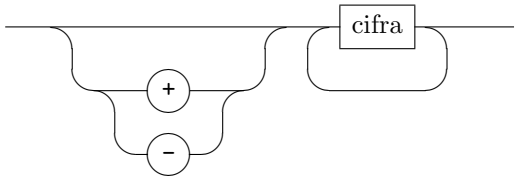
  broj



- Sintaksni dijagram za `CeoBroj`:

  ```
  <CeoBroj> ::= [ "+" | "-" ] ( <cifra> )+ ;
  ```

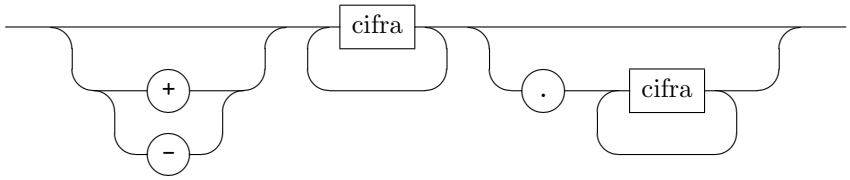
  CeoBroj



- Sintaksni dijagram za `RealanBroj`:

```
<RealanBroj> ::= "-"? <cifra>+ ("." <cifra>+)?
```

RealanBroj



- Sintaksni dijagram za `identifikator`:

```
<identifikator> ::= <slovo> { <slovo> | <cifra> }
```

ili

```
<identifikator> ::= <slovo> ( <slovo> | <cifra> ) *
```

identifikator