

Decembar 2002.

1. a) Konstruišimo automat u kome će svako od stanja da ima sledeću interpretaciju :

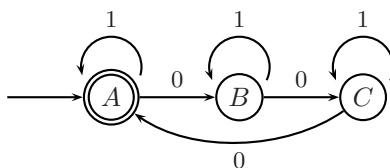
Neka je n broj nula u dosada prepoznatom delu niske.

A - n je deljiv sa 3

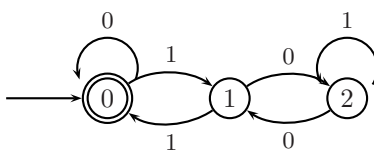
B - ostatak pri deljenju n sa 3 je 1

C - ostatak pri deljenju n sa 3 je 2

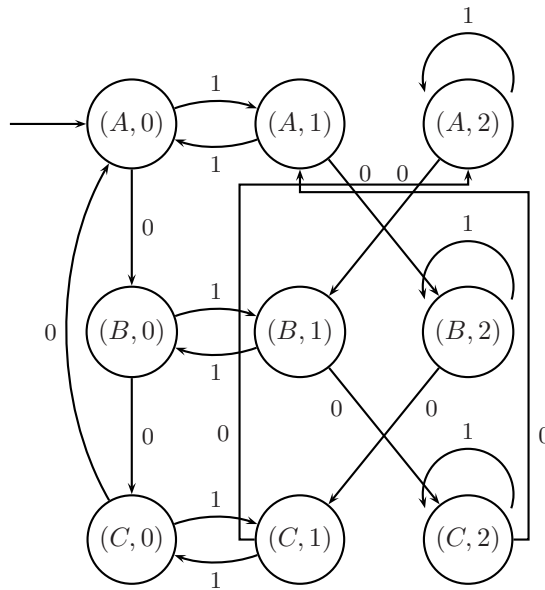
Početno stanje automata je A zato što prazna niska ne sadrži nijednu nulu. Ako je sledeći simbol 1, stanje se ne menja. Ako je, pak, sledeći simbol 0 prelazi se u stanje koje je sledeće po modulu 3. Završno stanje je A . Dakle traženi automat je :



b) Većina automata u kojima se zahteva ispitivanje deljivosti nekim brojem se konstruiše na prilično sličan način. Neka dakle konstruišemo automat koji određuje deljivost brojem n , broja zapisanog u osnovi m . Ideja je da nekako simuliramo algoritam deljenja. Svako stanje automata će da predstavlja jedan od mogućih ostataka pri deljenju sa brojem n . Stanja automata su, dakle, $0, 1, \dots, n-1$. Konstrukcija se zasniva na činjenici da se samo na osnovu ostatka pri deljenju broja zapisanog sa prvih k cifara našeg broja sa n i na osnovu $k+1$ -ve cifre možemo odrediti ostatak pri deljenju broja zapisanog sa prvih $k+1$ cifara našeg broja brojem n . Zaista, "spuštanje" $k+1$ -ve cifre predstavlja množenje dosadašnjeg ostatka brojem m i zatim dodavanjem $k+1$ cifre. Novi ostatak je ostatak pri deljenju tako dobijenog broja brojem n . Početno stanje je nula, jer na početku pretpostavljamo da nema nikakvog ostatka. Na osnovu rečenog možemo konstruisati automat:



c) Pošto ova dva automata predstavljaju presek i razliku prethodna dva automata koji su imali svojstvo da su deterministički, potpuni automati, možemo ih konstruisati standardnim postupkom za konstrukciju preseka i razlike. Ideja se zasniva na simuliranju istovremenog rada dva automata. Dakle konstruišemo automat čiji skup stanja se sastoji od proizvoda skupova stanja prethodna dva automata. Stanje (q_1, q_2) ima interpretaciju da bi prilikom prepoznavanja niske prvi automat bio u stanju q_1 , a drugi u stanju q_2 . Konstruišimo taj automat:



Sada odredimo završna stanja na osnovu uslova zadatka. U delu pod i) je potrebno da prvi automat ne prepoznaje nisku, dok je drugi prepoznaje. Dakle završna stanja su $(B,0)$ i $(C,0)$. U delu pod ii) je potrebno da oba automata prepoznaju nisku tako da je jedino završno stanje $(A,0)$. \square

3. Imajući u vidu sledeće jednakosti, do rešenja se veoma lako dolazi :

$$a^+ = aa^*$$

$$a? = a|\varepsilon$$

gde je a proizvoljni regularni izraz.

Karacterske klase se zamenjuju disjunkcijom karaktera članova. Npr.

$$[a - c] = a|b|c$$

Gramatika regularnih izraza je prilično uobičajena:

```

ri      :   ri BAR riili
          |   riili
          ;

riili   :   riili riputa
          |   riputa
          ;

```

```

riputa : ( ri )
        | riputa ?
        | riputa +
        | riputa *
        | [slovo-slovo]
        | slovo
        ;

```

Faza leksičke analize je jako jednostavna, imajući u vidu azbuku nad kojom su definisani regularni izrazi. Leksički analizator je dat u datoteci **decembar2002.lex**.

```
%option noyywrap
```

```
%{
#include "y.tab.h"
%}
```

```
%%
[a-z0-9_] return SLOVO;
"("      return LOZ;
")"      return DOZ;
"["      return LUZ;
"]"      return DUZ;
"?"      return UPITNIK;
"*"      return ZVEZDA;
"-"      return CRITICA;
"+"      return PLUS;
"|"      return BAR;
```

```
.      printf("Nedozvoljen karakter : %s\n",yytext);
%%
```

Prevodjenje vršimo koristeći YACC-ov interni stek. Važno je da naglasimo da će se na steku čuvati niske karaktera (tj. pokazivači na njih). Kada se, prilikom prevodjenja, naide na slovo azbuke ono se stavlja na stek. Zatim se u skladu sa primenjenim gramatičkim pravilima, i koristeći jednakosti sa početka teksta, vrednosti sa vrha steka zamenjuju odgovarajućim zamenama. Proizvod regularnih izraza a i b se zamenjuje sa ab , zatim se regularni izraz a^+ menja sa aa^* , a^* ostaje a^* , $a?$ se menja sa $a|\varepsilon$ i slično. Kada se izvrši redukcija celog regularnog izraza, štampa se vrednost sa vrha steka.

Kod je priložen u datoteci **decembar2002.y**

```
%{
#define yyerror printf
extern char* yytext;
#define YYSTYPE char*
```

```

#include <string.h>
%}

%left BAR
%token LOZ DOZ LUZ DUZ SLOVO UPITNIK ZVEZDA CRTICA PLUS
%start poc

%%
poc      :      ri      { printf("Neprosireni regularni
                             izraz je :\n%s\n", $1);
                             free($1);
                             }

ri  :   ri BAR riili   { $$=ili($1,$3);
                             free($1); free($2);
                             }
      |   riili        { $$=$1; }
      ;

riili  :   riili riputa { $$=puta($1,$2);
                             free($1); free($2); }
      |   riputa       { $$=$1; }
      ;

riputa :   LOZ ri DOZ   { $$=zagradi($2);
                             free($2);
                             }
      |   riputa UPITNIK { $$=upitnik($1);
                             free($1);
                             }
      |   riputa PLUS   { $$=plus($1);
                             free($1);
                             }
      |   riputa ZVEZDA { $$=zvezda($1);
                             free($1);
                             }
      |   LUZ slovo CRTICA slovo DUZ
      {   $$=klasa($2[0], $4[0]);
          free($2); free($4);
      }
      |   slovo        { $$=$1; }
      ;

slovo  :      SLOVO      { $$=strdup(yytext); }

%%

```

```

char* zagradi(char* s)
{
    char* r=(char *)malloc(strlen(s)+3);
    sprintf(r,"%s",s);
    return r;
}

char* ili(char* s, char* t)
{
    char* r=(char *)malloc(strlen(s)+strlen(t)+2);
    sprintf(r,"%s|%s",s,t);
    return r;
}

char* puta(char* s, char *t)
{
    char* r=(char *)malloc(strlen(s)+strlen(t)+1);
    sprintf(r,"%s%s",s,t);
    return r;
}

char* upitnik(char* s)
{
    char* r=(char *)malloc(strlen(s)+7);
    sprintf(r,"%s|eps",s);
    return r;
}

char* plus(char* s)
{
    char* r=(char *)malloc(2*strlen(s)+4);
    sprintf(r,"%s(%s)*",s,s);
    return r;
}

char* zvezda(char* s)
{
    char* r=(char *)malloc(strlen(s)+2);
    sprintf(r,"%s*",s);
    return r;
}

char* klasa(char a, char b)
{
    char* r;
    char c;
    if (b>=a)
    {
        r=(char *)malloc(2*(b-a)+4);
        r[0]='(';
        for (c=a; c<=b; c++)
        {
            r[2*(c-a)+1]=c;

```

```

        r[2*(c-a)+2]='|';
    }
    r[2*(b-a)+2]=')';
    r[2*(b-a)+3]='\0';
}
else
{
    r=(char *)malloc(1);
    r[0]='\0';
}
return r;
}

main()
{
    yyparse();
}

```