

Rešenja - februar 2003.

1. Pre određivanja jezika ovog automata poželjno je izvršiti minimizaciju ovog automata. Stanje G je očigledno nedostižno, tako da možemo da ga zanemarimo.

$$\{B, C, D, E\}\{A, F\}$$

$$\{C, D\}\{B, E\}\{A, F\}$$

$$\{C, D\}\{B, E\}\{A, F\}$$

Označimo skup stanja $\{A, F\}$ sa 1, $\{B, E\}$ sa 2 i $\{C, D\}$ sa 3. Dakle dobijeni automat ima 3 stanja. Stanje 1 je i početno i završno. Određivanje jezika ovog automata može da se izvrši na mnogo načina. Npr. metodom eliminacije stanja ili rešavanjem sistema regularnih jednačina. Rešimo ovaj put sistem jednačina:

$$X_1 = aX_1 + bX_2 + \varepsilon$$

$$X_2 = aX_2 + bX_3$$

$$X_3 = aX_3 + bX_1$$

Rešimo li prvo treću jednačinu, dobijamo :

$$X_3 = a^*bX_1$$

Ubacivanjem u drugu jednačinu i rešavanjem dobijamo :

$$X_2 = aX_2 + ba^*bX_1 = a^*ba^*bX_1$$

I na kraju ubacivanjem u prvu jednačinu i njenim rešavanjem dobijamo:

$$L = X_1 = aX_1 + ba^*ba^*bX_1 + \varepsilon = (a + ba^*ba^*b)^*$$

2.

3. Ideja je da se svakom aritmetičkom izrazu pridruži matrica karaktera koja predstavlja njegovu "lepu" reprezentaciju. Onda se od jednostavnijih matrica grade složenije i to postavljanjem jedne do druge ili jedne iznad druge, u zavisnosti od toga o kojoj se računskoj operaciji radi.

februar2003.l

```
%{
#include "y_tab.h"
%}

%%
"+" return PLUS;
"-" return MINUS;
"*" return PUTA;
"/" return PODELJENO;
"(" return LZ;
")" return DZ;
[0-9]+|[a-zA-Z] return LITERAL;
%%
```

februar2003.y

```

%{
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

enum VrstaOperacije {LIT, ADITIVNA, MULTIPLIKATIVNA};

/* Svakom izrazu odgovara matrica karaktera.
   Posto ne znamo kolike ove matrice mogu biti pravimo
   ih dinamickim */

typedef struct _mat
{ char **mat;
  int bv,bk;

  /* Vrsta matrice u kojoj se nalazi glavna
     razlomacka crta */
  int raz_crt;
  /* Zbog pravilnog postavljanja zagrada potrebno je
     znati i vrstu operacije na "vrhu" izraza */
  int vrsta_operacije;
} matrica;

#define YYSTYPE matrica*
#define yyerror printf
extern char* yytext;

#define max(a,b) ((a>b)?(a):(b))
%}
%left PLUS,MINUS
%left PUTA,PODELJENO
%token LITERAL,LZ,DZ
%start poc
%%
poc : izraz { ispisi($1); oslobodi($1);}
izraz : izraz PLUS izraz
      { $$=matrica_pored_matrice($1,'+', $3); oslobodi($1); oslobodi($3);}
      | izraz MINUS izraz
      { $$=matrica_pored_matrice($1,'-', $3); oslobodi($1); oslobodi($3);}
      | izraz PUTA izraz
      { $$=matrica_pored_matrice($1,'*', $3); oslobodi($1); oslobodi($3);}
      | izraz PODELJENO izraz
      { $$=matrica_iznad_matrice($1,$3); oslobodi($1); oslobodi($3);}
      | LZ izraz DZ
      { $$=$2;}
      | LITERAL
      { $$=napravi_matricu(yytext); }
      ;
%%
/* Funkcija alokira memoriju za matricu koja ima
   bv vrsta i bk kolona */
matrica* alociraj(int bv, int bk)

```

```

{   int i,j;
    matrica* pm=(matrica*) malloc(sizeof(matrica));
    pm->bv=bv;
    pm->bk=bk;
    pm->mat=(char**) malloc(bv*sizeof(char*));
    for (i=0; i<bv; i++)
    {   pm->mat[i]=(char*) malloc(bk*sizeof(char));
        for (j=0; j<bk; j++)
            pm->mat[i][j]=' ';
    }
    pm->raz_crt=0;
    pm->vrsta_operacije=LITERAL;
    return pm;
}

/* Ispisuje se matrica karaktera */
void ispisi(matrica* pm)
{   int i,j;
    for (i=0; i<pm->bv; i++)
    {   for (j=0; j<pm->bk; j++)
        {   printf("%c",pm->mat[i][j]);
            printf("\n");
        }
    }
}

/* Oslobadjamo strukturu matrice */
void oslobodi(matrica* pm)
{   int i,j;
    for (i=0; i<pm->bv; i++)
        free(pm->mat[i]);
    free(pm->mat);
    free(pm);
}

/* Pravimo "elementarnu" matricu na osnovu
niza karaktera */
matrica* napravi_matricu(char* lit)
{   matrica *pm;
    pm=(matrica*) malloc(sizeof(matrica));
    if (pm==NULL) exit(1);
    pm->bv=1;
    pm->bk=strlen(lit);
    pm->mat=(char**) malloc(sizeof(char*));
    pm->mat[0]=strdup(lit);
    pm->raz_crt=0;
    pm->vrsta_operacije=LIT;
    return pm;
}

/* Funkcija ubacuje blok matricu s u matricu d i to tako
da gornji levi ugao bude na poziciji x,y matrice d */
void ubaci_mat_u_mat(matrica* d, matrica* s, int x, int y)

```

```

{   int i,j;
    for (i=0; i<s->bv; i++)
        for(j=0; j<s->bk; j++)
            d->mat[x+i][y+j]=s->mat[i][j];
}

/* Gradimo novu matricu tako sto postavljamo matricu l,
   zatim operator i onda matricu d jedno pored drugoga */
matrica* matrica_pored_matrice(matrica* l, char op, matrica* d)
{   int bv,bk;
    int bordura,zagrade_levo,zagrade_desno,poz_operatora;
    matrica* pm;

    /* Ako matrice l i d sadrze razlomacku crtdu ostavljamo
       jedno prazno mesto izmedju razlomaka i operatora */
    bordura=(l->raz_crt==0 &&
              d->raz_crt==0)?0:1;

    /* Da li ima potrebe stavljati matricu l u zagrade ? */
    zagrade_levo=((op=='*') && (l->vrsta_operacije==ADITIVNA))?1:0;

    /* Da li ima potrebe stavljati matricu d u zagrade ? */
    zagrade_desno=((op=='*') && (d->vrsta_operacije==ADITIVNA)) ||
                  ((op=='-') && (d->vrsta_operacije==ADITIVNA))?1:0;

    /* Odredjujemo dimenzije nove matrice */
    bk=zagrade_levo + l->bk + zagrade_levo+
        bordura+1+bordura+
        zagrade_desno + d->bk + zagrade_desno;

    bv=max(l->raz_crt,d->raz_crt)+
        1+
        max(l->bv-l->raz_crt-1,d->bv-d->raz_crt-1) ;

    /* Alociramo prostor */
    pm=alociraj(bv,bk);
    /* Odredjujemo polozaj glavne razlomacke crte */
    pm->raz_crt=max(l->raz_crt,d->raz_crt);

    /* Postavljamo polja strukture */
    pm->vrsta_operacije=(op=='*')?MULTIPLIKATIVNA:ADITIVNA;

    /* Operator postavljamo na odgovarajuće mesto */
    poz_operatora=zagrade_levo+l->bk+zagrade_levo+bordura;
    pm->mat[pm->raz_crt][poz_operatora]=op;

    /* Ukoliko je potrebno staviti zagrade postavljamo ih */
    if (zagrade_levo)
    {   pm->mat[pm->raz_crt][0]='(';
        pm->mat[pm->raz_crt][1+l->bk]=')';
    }
    if (zagrade_desno)

```

```

    {   pm->mat[pm->raz_crtal
        [poz_operatora + 1 + bordura]='(';
        pm->mat[pm->raz_crtal
            [poz_operatora + 1 + bordura
                + zagrade_desno + d->bk]=')';
    }
    /* Ugradjujemo vec postojece matrice l i d na odgovarajuce pozicije */
    ubaci_mat_u_mat(pm,l,pm->raz_crtal->raz_crtal,
                    zagrade_levo);
    ubaci_mat_u_mat(pm,d,pm->raz_crtal->raz_crtal,
                    poz_operatoral+1+bordura+zagrade_desno);

    return pm;
}

/* Gradimo razlomak od dve matrice */
matrica* matrica_iznad_matrice(matrica* l, matrica* d)
{   int bv,bk,bordura;
    int i;
    matrica* pm;

    /* Ukoliko je ili brojilac ili imenilac razlomak,
       onda razlomacka crta mora biti duza */
    bordura=(l->raz_crtal==0 &&
              d->raz_crtal==0)?0:1;

    /* Odredjujemo broj vrsta i kolona nove matrice */
    bk=max(l->bk,d->bk)+2*bordura;
    bv=l->bv+1+d->bv;

    /* Alociramo memoriju */
    pm=alociraj(bv,bk);

    /* Ubacujemo brojilac */
    ubaci_mat_u_mat(pm,l,0,(bk-l->bk)/2);

    /* Gradimo razlomacku crtdu */
    for (i=0; i<bk; i++)
        pm->mat[l->bv][i]='-';

    /* Ubacujemo imenilac */
    ubaci_mat_u_mat(pm,d,l->bv+1,(bk-d->bk)/2);

    /* Postavljamo ostala polja strukture */
    pm->raz_crtal=l->bv;
    pm->vrsta_operacije=MULTIPLIKATIVNA;
    return pm;
}

main() {   yyparse(); }

```