Linguagens Formais e Autómatos

Exame 07 Fevereiro 2001 09:00

Parte prática - Duração: 01h30m

Atenção:

Indique em cada folha o seu número e nome
A prova é com consulta e tem a duração de 01h30m

Grupo I

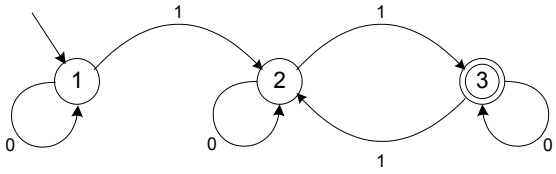
Cotação a) 2.0; b) 3.0

Para cada uma das alíneas seguintes represente a linguagem dada usando uma expressão regular e um autómato finito determinístico:

a) Para o alfabeto $\Sigma = \{0,1\}$, $L(A) = \{u \in \Sigma^* : u \text{ tem um número par de 1's}\}$

Expressão regular: $(0^*10^*1)^+0^*$

Autómato:

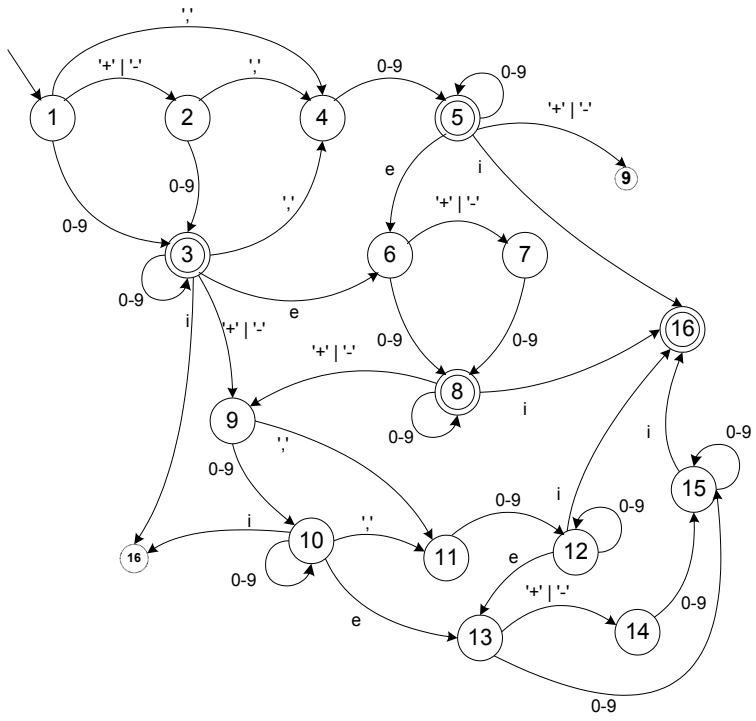


b) Para o alfabeto $\Sigma = \{-, +, 0,1,2,3,4,5,6,7,8,9, e, i, .\}$,
$L(A) = \{u \in \Sigma^* : u \text{ é um número imaginário}\}$

Expressão regular:

$$[-+]?\ ([0\text{-}9]^*".")?\ [0\text{-}9]^+([eE][-+]?\ [0\text{-}9]^+)?\ (i|[-+]?\ ([0\text{-}9]^*".")?\ [0\text{-}9]^+([eE][-+]?\ [0\text{-}9]^+)?\ i)?$$

Autómato:



Grupo II Cotação a) 2.0; b) 2.0; c) 1.0

Considere o seguinte conjunto:

$$L = \{b^m a^{2n}\, b^n a^{2n} b^m : m, n \geq 1\}$$

a) Represente-o sob forma gramatical.

Como **m** e **n** não têm nenhuma ligação entre eles os três elementos do meio podem ser construídos independentemente dos dois das pontas.

S → bSb | bAb
A → aaABC | aabC
CB → BC
bB → bb
bC → baa
aC → aaa

b) A gramática que obteve é LL(1)? Justifique.

A gramática indicada não é do tipo 3 porque possui autocontenção. Também não é do tipo 2 porque a condição imposta só pode ser satisfeita com dependência de contexto. Por estes motivos a gramática é de tipo 1.

Uma gramática do tipo 1 não pode ser LL(1)

c) Apresente uma sequência de derivações a realizar, para obter a expressão **baaaabbaaaab**

$S \Rightarrow^1 bAb \Rightarrow^2 baaABCb \Rightarrow^3 baaaabCBCb \Rightarrow^4 baaaabBCCb \Rightarrow^5 baaaabbCCb \Rightarrow^6 baaaabbaaCb \Rightarrow^7 baaaabbaaaab$

Grupo III Cotação 7.0

Suponha uma gramática que reconhece endereços URL e implemente-a utilizando *Flex* e *Bison*. Para além disso, o reconhecimento de um endereço deve ser seguido pela indicação dos seus componentes.

Exemplo 1: para http://www.dei.isep.ipp.pt/nova/index.html, a resposta deverá ser:

- endereço válido
- protocolo: http
- máquina: www.dei.isep.ipp.pt
- caminho: nova
- página: index.html

Exemplo 2: para mailto:dei@isep.ipp.pt, a resposta deverá ser:

- endereço válido
- protocolo: mailto
- utilizador: dei
- domínio: isep.ipp.pt

Considere como válidos os seguintes protocolos: http; https; ftp; mailto. Lembre-se que quer o caminho, quer a página podem não existir, caso em que a sua indicação deve ser ignorada e que a máquina pode ser substituída pelo respectivo endereço IP. Considere ainda que os nomes das máquinas e dos ficheiros apenas são constituídos pelos caracteres de ***a*** a ***Z***.

Gramática:

S → US | ε
U → P ':' '/' '/' M C | mailto ':' L '@' D
P → http | https | ftp
M → I | N
I → int '.' int '.' int '.' int
N → N '.' pal | pal
C → '/' R | ε
R → pal '/' R | pal | F
F → pal '.' pal | ε
L → L pal | L '.' | pal
D → D '.' pal | pal

U indica o url, podendo ser dividido em 2 tipos:

- o primeiro (ftp,http,https) é constituído por protocolo (P) "://" máquina (M, ip (I) ou nome (N)) e caminho (C) sendo que o caminho contém a página a aceder (F);
- o segundo (email) é constituído pela palavra reservada "mailto" seguido de ":" utilizador (L) e domínio de email (D)

pal é uma palavra constituida por letras de a **A** a **Z** e **int** é um número inteiro

Flex:

```
%{
    #include"url.tab.h"
    extern int n_erros;
%}


%%

http        {return HTTP;}
https       {return HTTPS;}
ftp         {return FTP;}
mailto      {return MAILTO;}
[a-zA-Z]+   {strncpy(yylval.pal,yytext);return PAL;}
[0-9]+      {strncpy(yylval.pal,yytext);return NUM;}
\/ |
\. |
\n |
@ |
:           {return(yytext[0]);}
[ \t]       ;
<<EOF>>     {return 0;}
.           {printf("Erro lexico '%s'\n",yytext);n_erros++;}

%%

int yywrap() {

    return(1);

}
```

Bison:

```
%{
    int n_erros=0;
    char protocolo[10];


    char maquina_dominio[1024];
    char caminho[1024];
    char pagina_utilizador[1024];


%}

%union {
    char pal[255];
}

%token HTTP HTTPS FTP MAILTO
%token <pal> PAL NUM

%%

urls: urls {protocolo[0]='\0';
            maquina_dominio[0]='\0';
            caminho[0]='\0';
            pagina_utilizador[0]='\0';}
      url
    | /* vazio */
    ;

url:  url1 ':' '/' '/' maquina caminho '\n'   {
               printf("url valido\n");
               printf("protocolo: %s\n", protocolo);
               printf("maquina: %s\n", maquina_dominio);
               if(strlen(caminho)>0) printf("caminho: %s\n", caminho);
               if(strlen(pagina_utilizador)>0) printf("pagina: %s\n", pagina_utilizador);
              }
    | MAILTO {strcpy(protocolo,"mailto");}
      ':' utilizador '@' dominio '\n'         {
               printf("url valido\n");
               printf("protocolo: %s\n", protocolo);
               printf("utilizador: %s\n", pagina_utilizador);
               if(strlen(maquina_dominio)>0) printf("dominio: %s\n", maquina_dominio);
              }
```

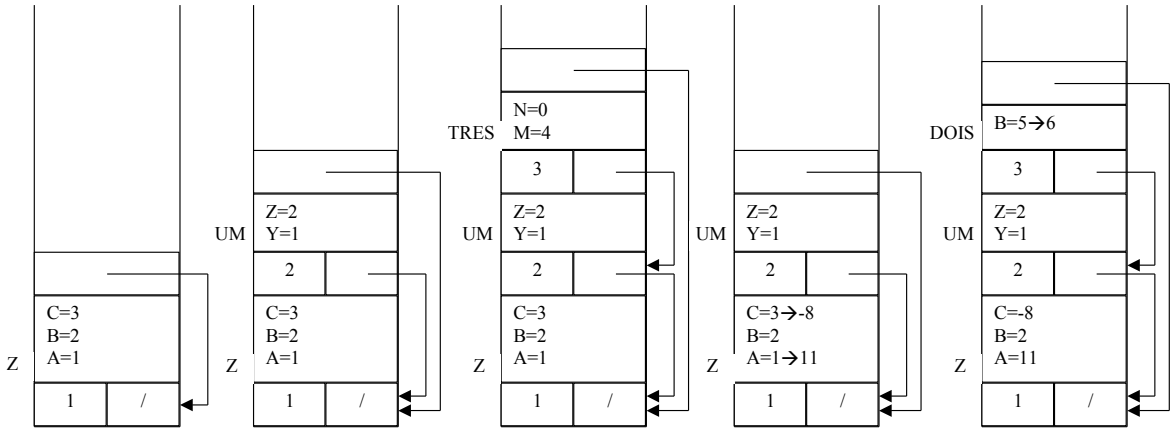
```
    | error '\n'                                    {printf("url invalido\n");yyerrok;}
    ;

url1: HTTP      {strcpy(protocolo,"http");}
    | HTTPS     {strcpy(protocolo,"https");}
    | FTP       {strcpy(protocolo,"ftp");}
    ;

maquina: ip
    | maquina1
    ;

ip: NUM '.' NUM '.' NUM '.' NUM    {
                strcat(maquina_dominio,$1);strcat(maquina_dominio,".");
                strcat(maquina_dominio,$3);strcat(maquina_dominio,".");
                strcat(maquina_dominio,$5);strcat(maquina_dominio,".");
                strcat(maquina_dominio,$7);
               }

maquina1: maquina1 '.' PAL      {strcat(maquina_dominio,".");strcat(maquina_dominio,$3);}
    | PAL                       {strcat(maquina_dominio,$1);}
    ;

caminho: '/' {strcat(caminho,"/");} caminho1
    | /* vazio */
    ;

caminho1: PAL  '/' {strcat(caminho,$1);strcat(caminho,"/");} caminho1
    | PAL                   {strcat(caminho,$1);}
    | pagina
    ;

pagina: PAL '.' PAL         {strcat(pagina_utilizador,$1);
                             strcat(pagina_utilizador,".");
                             strcat(pagina_utilizador,$3);
                            }
    | /* vazio */

utilizador: utilizador PAL    {strcat(pagina_utilizador,$2);}
    | utilizador '.'          {strcat(pagina_utilizador,".");}
    | PAL                     {strcat(pagina_utilizador,$1);}
    ;

dominio: dominio '.' PAL      {strcat(maquina_dominio,".");
                               strcat(maquina_dominio,$3);
                              }
    | PAL                     {strcat(maquina_dominio,$1);}
    ;

%%

int main() {

    printf("teste de url's\n");
    yyparse();

}

int yyerror(char *s) {
    printf("Erro sintactico: %s\n",s);
    n_erros++;
}
```

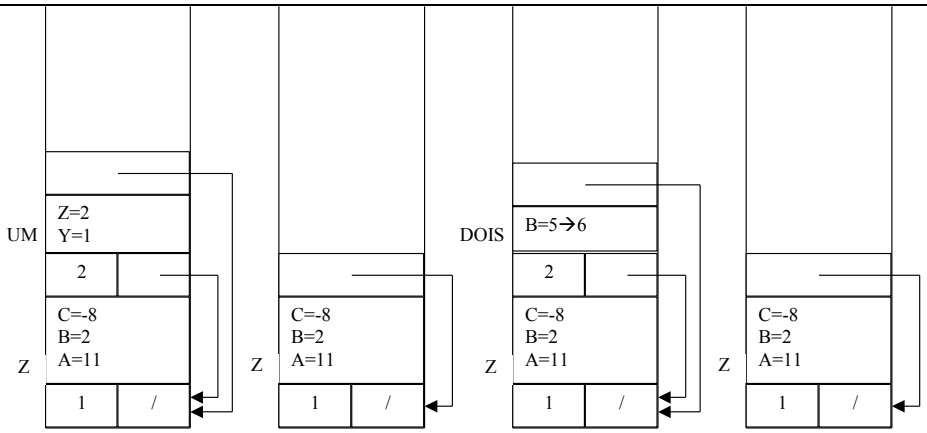
Grupo IV

Cotação 3.0

Para o programa esboçado, desenhe a stack de execução (indicando os valores das variáveis) nas sucessivas fases de execução indicando em cada uma as ligações estáticas e dinâmicas entre as várias frames presentes.

```
programa Z;
variável A=1, B=2, C=3: inteiro;

função UM;
variável Y=1, Z=2: inteiro;

      função TRES;
      variável M=4, N=0: inteiro
      início
            TRES = M * B + Y;
      fim;

início
      A = TRES + Z;
      C = C - A;
      chamar DOIS;
fim;

função DOIS;
variável B=5: inteiro;
início
      B = B + 1;
fim;

início
      chamar UM;
      chamar DOIS;
fim
```




UM
Z=2
Y=1
2
C=-8
B=2
A=11
Z
1
/
Z
C=-8
B=2
A=11
1
/
DOIS
B=5→6
2
C=-8
B=2
A=11
Z
1
/
Z
C=-8
B=2
A=11
1
/

FIM