Linguagens Formais e Autómatos

Teste 04 Janeiro 2001 14:00

Parte teórica - Duração: 15 min

Nome_____ **Número** ☐☐☐☐☐☐

Atenção:

Responda às perguntas na folha do enunciado
Indique o seu número e nome
A prova é sem consulta
Cada resposta errada terá uma cotação negativa de 2/3 do seu valor

Parte teórica 8 valores

Todas as perguntas tem a mesma cotação

1. As linguagens do tipo 3 na hierarquia de Chomsky:
   - só podem ser reconhecidas por expressões regulares ..... ☐
   - podem ser reconhecidas por expressões regulares ..... ☒
   - nunca podem ser reconhecidas por expressões regulares ..... ☐
   - são linguagens com apenas 3 símbolos ..... ☐

2. Uma gramática independente do contexto que possua autocontenção é:
   - uma linguagem regular..... ☐
   - uma linguagem independente do contexto ..... ☒
   - uma linguagem complexa ..... ☐
   - uma linguagem irregular..... ☐

3. Um compilador:
   - compila uma linguagem gerando os tokens ..... ☐
   - junta um programa fonte às respectivas bibliotecas ..... ☐
   - traduz um programa escrito em código fonte em código objecto ..... ☒
   - nenhuma das anteriores..... ☐

4. Um autómato finito diz-se determinístico se:
   - nenhum estado tem transições para o estado inicial ..... ☐
   - só tem um estado final ..... ☐
   - em cada estado um símbolo do alfabeto permite uma única transição ..... ☒
   - num estado cada símbolo do alfabeto permite pelo menos uma transição..... ☐

5. A afirmação "produções alternativas do mesmo não terminal começam por terminais distintos" é:
   - condição suficiente para uma gramática ser LL(1) ..... ☐
   - condição necessária para uma gramática ser LL(1) ..... ☒
   - condição necessária e suficiente para uma gramática ser LL(1)..... ☐
   - nenhuma das anteriores..... ☐

6. Numa gramática LL(2), o número 2 significa:
   - do tipo 2 ❑
   - utilização de 2 símbolos em avanço ☒
   - leitura de 2 símbolos da frase de cada vez ❑
   - nenhuma das anteriores ❑

7. Uma gramática diz-se recursiva à esquerda se
   - o mesmo terminal aparece em duas produções alternativas ❑
   - existe uma variável A tal que existe uma derivação $A \Rightarrow^{+} \alpha A$ ❑
   - duas produções alternativas começam pelo mesmo terminal ❑
   - existe uma variável A tal que existe uma derivação $A \Rightarrow^{+} A\alpha$ ☒

8. Utilizando apenas um inteiro para identificar o tipo dos identificadores podemos ter:
   - um número potencialmente ilimitado de tipos ❑
   - apenas um número limitado de tipos ☒
   - tipos definidos pelo utilizador ❑
   - apenas 8 tipos diferentes ❑

9. A inserção de acções semânticas numa gramática independente do contexto permite:
   - satisfazer os requisitos de linguagens dependentes do contexto ☒
   - a execução do código gerado ❑
   - a realização de um compilador de duas passagens ❑
   - nenhuma das anteriores ❑

10. Na geração de código para uma ocorrência aplicada de uma variável, numa linguagem declarativa (por exemplo o C), temos que:
    - efectuar, se necessário, uma conversão de inteiro para real ☒
    - efectuar, comparação de tipos em termos de compatibilidade ❑
    - gerar uma etiqueta correspondente à variável em causa ❑
    - inserir a variável na tabela de símbolos ❑

FIM

Atenção:

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

Grupo I

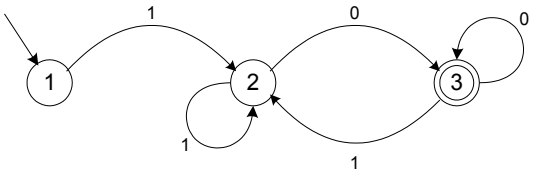
Cotação a) 2.0; b) 2.0

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

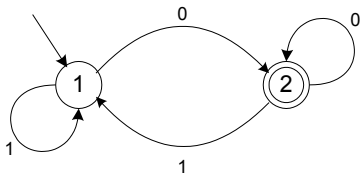
a) Para o alfabeto $\Sigma = \{0,1\}$, $L(A) = \{u \in \Sigma^* : u \text{ é um número par}\}$

Expressão regular: $1(1|0)^*0$ **ou menos correcto** $(1|0)^*0$

Autómato:



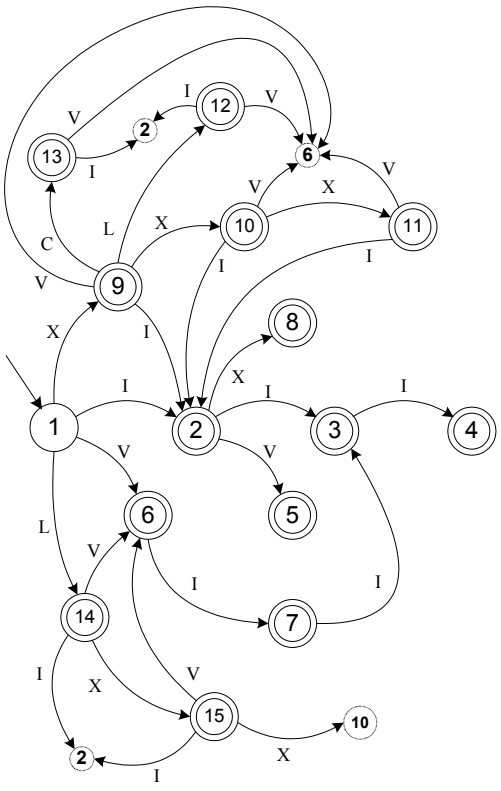
ou



b) Para o alfabeto $\Sigma = \{I, V, X, L, C\}$, $L(A) = \{u \in \Sigma^* : u \text{ é um núemro romano} < 100\}$

Expressão regular: $(X(X\{0,2\}|L|C)?\,|LX\{0,3\})?\,(I(I\{0,2\}|V|X)?\,|VI\{0,3\})$

Autómato:



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

Considere o seguinte conjunto:

$$L = \{a^n b^n c^m : 2m = n, n \geq 0\}$$

a) Represente-o sob forma gramatical.

Como **m** e **n** têm de ser inteiros, temos para m=0, n=0; para m=1, n=2; para m=2, n=4; etc..

S' → S | ε
S → aaSBA | aabbA
AB → BA
bB → bbb
bA → ba
aA → aa

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 **aabba**

S' $\Rightarrow^1$ S $\Rightarrow^2$ aabbA $\Rightarrow^3$ aabba

Se fosse pedida a frase **aaaabbbbaa**, seria:

S' $\Rightarrow^1$ S $\Rightarrow^2$ aaSBA $\Rightarrow^3$ aaaabbABA $\Rightarrow^4$ aaaabbBAA $\Rightarrow^5$ aaaabbbbAA $\Rightarrow^6$ aaaabbbbaA $\Rightarrow^7$ aaaabbbbaa

Grupo III a): 2.5 b): 1.5

Suponha uma gramática com as seguintes produções:

S→ aAB | bBA
A→ aA | C
B → bB | C
C → (C) | ε

a) Escreva um parser em descida recursiva que corresponda à gramática.

Este parser utiliza as variáveis globais token e nerros, e a função yylex() que devolve o próximo token.

```
enum{A,B,A_PAR,F_PAR,FIM};

int token,nerros=0;

void s();
void a();
void b();
void c();

void erro(char *s) {
      nerros++;
      printf("Erro %d: %s\n",nerros,s);
}

void s() { /* S->aAB | bBA */
  if (token==A) {
      token=yylex();
      a();
      b();
  }
  else
      if (token==B) {
         token=yylex();
         b();
         a();
      }
      else
         erro("não é 'a' nem 'b'");
}

void a() { /* A->aA | C  */
  if (token == A)
  {
     token=yylex();
     a();
  }
  else c();
}
```

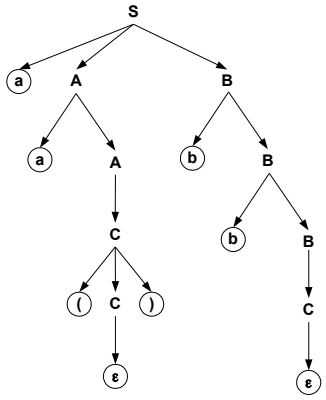
```
void b() { /* B->bB | C  */
  if (token == B)
  {
     token=yylex();
     b();
  else     c();
  }
}

void c() { /* C-> (C) | vazio */
  if (token == A_PAR) {
      token=yylex();
      c();
      if (token == F_PAR)
          token=yylex();
      else
          erro("falta ')'");
  }
      /* vazio, não dá erro */
}

void main() {
  token=yylex(); /* pega o 1º token */
  s();           /* produção inicial */

  if (nerros==0 && token==FIM)
       printf("Expressão válida \n");
  else
       printf("Expressão inválida \n");
}
```

b) Diga se a frase *aa()bb* pertence à gramática, justificando através da apresentação de uma árvore de derivação.



A frase pertence à gramática, pois todos os símbolos da frase foram consumidos e nas folhas da árvore só existem terminais.

Grupo IV

Cotação 7.0

Suponha um simulador de uma máquina de venda automática que dispõe de um conjunto de produtos e aceita moedas em euros (€0.01, €0.02, €0.05, €0.10, €0.20, €0.50, €1.00, €2.00). O objectivo é seleccionar um produto, introduzir o respectivo valor, receber o troco (se existir) e receber o produto. Considere os seguintes produtos: café (€0.25), chá (€0.25), chocolate (€0.40), copo (€0.05) e leite (€0.30). O formato de entrada de dados deve obedecer à seguinte regra: <produto>, <$moeda_1$>, ... <$moeda_n$>. O formato de saída deve obedecer à seguinte regra: <produto>, <$moeda_1$>, ... <$moeda_n$> | "dinheiro insuficiente".

Exemplo:

Entrada - café, €0.01, €0.10, €0.05, €0.20
Saída – café, €0.10, €0.01

Defina a gramática de modo a que a máquina funcione ininterruptamente e implemente-a utilizando *Flex* e *Bison*.

Gramática:

S → PS | ε
P → B ',' MR
R → ',' MR | ε
B → café | chá | chocolate | copo | leite
M → €0.01 | €0.02 | €0.05 | €0.10 | €0.20 | €0.50 | €1.00 | €2.00

Flex: Para execução no LINUX, símbolo '€' foi substituído por 'e'

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

%%

cafe        {return CAFE;}
cha         {return CHA;}
chocolate   {return CHOCOLATE;}
leite       {return LEITE;}
copo        {return COPO;}
e0?\.01      {return M_1_CENTIMO;}
e0?\.02      {return M_2_CENTIMOS;}
e0?\.05      {return M_5_CENTIMOS;}
e0?\.10?      {return M_10_CENTIMOS;}
e0?\.20?      {return M_20_CENTIMOS;}
e0?\.50?      {return M_50_CENTIMOS;}
e1(\.0{0,2})?      {return M_1_EURO;}
e2(\.0{0,2})?      {return M_2_EUROS;}
\n |
\,           {return(yytext[0]);}
[ \t]       ;
<<EOF>>     {return 0;}
.           {printf("Erro lexico '%s'\n",yytext);n_erros++;}

%%

int yywrap() {
    return(1);
}
```

Bison: Para execução no LINUX, símbolo '€' foi substituído por 'e'

```
%{
    #define YYSTYPE double
    int n_erros=0;
    char msg[1024];
    char *troco(double, double);

%}

%token CAFE CHA CHOCOLATE LEITE COPO
%token M_1_CENTIMO M_2_CENTIMOS M_5_CENTIMOS M_10_CENTIMOS M_20_CENTIMOS M_50_CENTIMOS
%token M_1_EURO M_2_EUROS

%%

maquina: maquina pedido
    | /* vazio */
    ;

pedido:  bebida ',' moeda resto '\n' {printf("%s\n",troco($1,$3+$4));}
    | error '\n'                     {yyerrok;}
    ;

resto: ',' moeda resto  {$$=$2+$3;}
    | /* vazio */       {$$=0.0;}
    ;

bebida: CAFE            {printf("cafe");$$=.25;}
      | CHA             {printf("cha");$$=.25;}
      | CHOCOLATE       {printf("chocolate");$$=.40;}
      | LEITE           {printf("leite");$$=.30;}
      | COPO            {printf("copo");$$=.05;}
      ;

moeda: M_1_CENTIMO      {$$=.01;}
     | M_2_CENTIMOS     {$$=.02;}
     | M_5_CENTIMOS     {$$=.05;}
     | M_10_CENTIMOS    {$$=.10;}
     | M_20_CENTIMOS    {$$=.20;}
     | M_50_CENTIMOS    {$$=.50;}

     | M_1_EURO         {$$=1.0;}
```

```
      | M_2_EUROS        {$$=2.0;}
      ;

%%

char *troco(double preco, double dinheiro)
{
  double troco;

  troco=dinheiro-preco;
  msg[0]=0;
  printf("(troco:%f)",troco,troco);
  if (troco<0)
    strcat(msg,",Dinheiro insuficiente");
  else
    if (troco==0)
      strcat(msg,",Não há troco");
    else
      while(troco>0.0)
       if (troco>=2.0) {
          strcat(msg,",e2.00");
          troco-=2.0;
        }
        else
          if (troco>=1.0) {
            strcat(msg,",e1.00");
            troco-=1.0;
          }
          else
            if (troco>=.5) {
              strcat(msg,",e0.50");
              troco-=.5;
            }
            else
              if (troco>=.2) {
                strcat(msg,",e0.20");
                troco-=.2;
              }
              else
                if (troco>=.1) {
                  strcat(msg,",e0.10");
                  troco-=.1;
                }
                else
                  if (troco>=.05) {
                    strcat(msg,",e0.05");
                    troco-=.05;
                  }
                  else
                    if (troco>=.02) {
                      strcat(msg,",e0.02");
                      troco-=.02;
                    }
                    else {
                        strcat(msg,",e0.01");
                        troco-=.01;
                      }
  return(msg);
}

int main() {

    printf("máquina de venda de produtos\n");
    yyparse();
}

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

FIM