Ex. Alternativo LFA 1

Prof. Leandro I. Pinto

Nome: _____

1) Construa Autômatos com Pilha **Determinísticos** para as seguintes Linguagens:

a) L = $\{a^n b^*(a+c)^m \mid m = 2n\}$

b) L =$\{a^i b^j c^k \mid i\ mod\ 2 = 0,\ j = \frac{i}{2}\ e\ k = 5\}$

c) $L = \{a^n (ba)^* b^n \mid n > 0\}$

d) $L = \{w | w \in \{a,b,c\}*, |w|_a mod\ 2 = 0, |w|_b mod\ 2 = 1\ e\ |w|_c = |w|_a + |w|_b\}$

2) Análise sintática é o processo de analisar uma sequencia de entrada a fim de verificar sua estrutura segundo uma gramática formal. Na computação, essa análise pode fazer parte de um compilador, junto com a análise léxica e semântica.
Um desenvolvedor está com dificuldade para definir as produções de uma gramática para o seu compilador, o qual compila código escrito numa linguagem chamada Portuga.
A linguagem Portuga possui somente os comandos 'se', 'enquanto', atribuição e 'escreva', o que pode ser visto na Figura abaixo.

```
/* Código exemplo em Portuga! */

a = 10;
b = 0;
enquanto a > 0 faça
  b = b*a;
  a = a + 1;
  se b > 200 entao
    escreva "b maior que 200\n";
  fim_se
fim_enquanto
```

A linguagem permite fazer vários se's e/ou enquanto's aninhados. Parte da gramática já está concluída como a seguir:

$Arquivo \rightarrow Arquivo\ Comandos \mid \varepsilon$
$Comandos \rightarrow\ Se \mid Enquanto \mid Attr \mid Escreva$
$Se \rightarrow ?$
$Enquanto \rightarrow ?$
$Attr \rightarrow ID = LogicExpr;$
$LogicExpr \rightarrow LogicExpr\ and\ ExpRel \mid LogicExpr\ or\ ExpRel \mid ExpRel$
$ExpRel \rightarrow ExpRel < Expr \mid ExpRel > Expr \mid Expr$
$Expr \rightarrow Expr + E_2 \mid E_2$
$E_2 \rightarrow E_2 * E_3 \mid E_3$
$E_3 \rightarrow ID \mid INTEGER$
$Escreva \rightarrow escreva\ STRING;$

Veja que o desenvolvedor não conseguiu criar as produções para implementar os comandos *Se* e *Enquanto*. Então, crie essas produções, incluindo novas produções se necessário.

3) Considere a gramática

$$G = (V, T, P, S), \quad onde:$$
$$P = \{S \to Bi, \quad B \to bAC|aD, \quad A \to aAC|\varepsilon, \quad C \to cc, \quad D \to aDd|\varepsilon\}$$

Determine o conjunto de produções D segundo o algoritmo a seguir:

$D = \emptyset$
`para toda` $A \to \alpha B$, onde $B \in V$
`faça` $D = D \cup \{C \to \beta A \phi\}$

4) Nas expressões aritméticas, operadores de multiplicação tem precedência sobre operadores de soma. Uma GLC pode garantir que operadores de multiplicação sejam reduzidos antes dos operadores de soma. Num compilador, isso garante que a multiplicação será calculada antes da soma.
Construa a árvore de derivação para as seguintes expressões e diga se é possível reduzir a soma antes da multiplicação em algum caso:

a) $2 * 3 + 4 * 5 + 7$
b) $5 + 3 * 1 + a$
c) $i + j * k * m$

Responda:
d) A gramática para expressões lógicas ($LogicExpr$) é ambígua? Se sim, prove.

5) Assinale V para Verdadeiro e F para Falso
a) [ ] AP Determinístico tem o mesmo poder computacional de um AP Não-Determinístico;
b) [ ] Um AP pode processar uma linguagem regular;
c) [ ] Uma GLC pode ser transformada em um AP e vice-versa, pois são formalismos equivalentes;
d) [ ] Máquina de Turing é considerado o mais geral dispositivo de computação. Ela pode inclusive processar linguagens sensíveis ao contexto;

6) Mostre a hierarquia de Chomsky e apresente um exemplo de linguagem de cada classe, exceto para as linguagens do tipo 0.

GLC para AP
A partir de uma GLC $G = (V, T, P, S)$ sem recursão a esquerda, fazemos:

$$M = \left(T, \{q_0, q_1, q_f\}, \delta, q_0, \{q_f\}, V \cup T\right), \; onde:$$

$\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, S)\}$
$\delta(q_1, \varepsilon, A) = \{(q_1, \alpha)|A \to \alpha \in P\}, \; para\; toda\; variável\; A$
$\delta(q_1, a, a) = \{(q_1, \varepsilon)\}, \; para\; todo\; terminal\; a\; de\; T$
$\delta(q_1, ?, ?) = \{(q_f, \varepsilon)\}$