CONSTRUÇÃO DE COMPILADORES I [BCC328]

Segunda prova (2017–2)
20 de novembro de 2017

Matrícula: _____

Nome: _____

Departamento de Computação
Universidade Federal de Ouro Preto
Prof. José Romildo Malaquias

Questão 1. Construa a tabela LR(1) para a seguinte gramática, e determine se a gramática é LR(1) ou não, justificando sua resposta.

$$P \to S \ \texttt{\$}$$

$$S \to X \ \texttt{a}$$
$$S \to \texttt{b} \ X \ \texttt{c}$$
$$S \to Y \ \texttt{c}$$
$$S \to \texttt{b} \ Y \ \texttt{a}$$

$$X \to \texttt{d}$$

$$Y \to \texttt{d}$$

Questão 2. Considere a linguagem de programação definida pela gramática seguinte:

$$0 \quad S \to E \ \texttt{\$+}$$

$$1 \quad E \to E \ \texttt{+} \ T$$
$$2 \quad E \to T$$

$$3 \quad T \to T \ \texttt{*} \ F$$
$$4 \quad T \to F$$

$$5 \quad F \to \texttt{(} \ E \ \texttt{)}$$
$$6 \quad F \to \texttt{id}$$

A tabela de análise sintática LR(1) para esta gramática é mostrada a seguir.

	id	+	*	(	)	$	E	T	F
0	s5			s4			g1	g2	g3
1		s6		s4		a			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			g8	g2	g3
5		r6	r6		r6	r6			
6	s5			s4				g9	g3
7	s5			s4					g10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Determine se a cadeia de entrada

$$\texttt{(x + y) z *}$$

(onde `x`, `y` e `z` são *tokens* do tipo `id`) pertence à linguagem, verificando se a cadeia é aceita pelo analisador sintático LR(1) que utiliza esta tabela. Para tanto simule a execução do autômato de pilha do analisador mostrando os seus movimentos processar a cadeia de entrada. Em cada etapa informe a configuração do autômato:

- estado atual
- conteúdo da pilha, incluindo o estado salvo do autômato e o símbolo gramatical em cada posição da pilha
- posição do cursor na cadeia de entrada
- próxima ação (*shift*, *reduce*+*goto*, *accept* ou *error*) a ser realizada

Questão 3. A gramática livre de contexto a seguir define expressões regulares usando os símbolos 0 e 1:

$$
\begin{array}{ll}
0 & S \to E \ \ \$ \\
1 & E \to E \ \ | \ \ E \\
2 & E \to E \ \ E \\
3 & E \to E \ \ * \\
4 & E \to \mathbf{0} \\
5 & E \to \mathbf{1} \\
6 & E \to \varepsilon
\end{array}
$$

Os estados do autômato de pilha para o analisador sintático ascendente LR(1) para esta gramática são indicados a seguir.

```
STATE 1
S ->  . E   "$" , ?
E ->  . E   "|"   E , $   |   epsilon   1   0   *
E ->  . E   E , $   |   epsilon   1   0   *
E ->  . E   "*" , $   |   epsilon   1   0   *
E ->  . "0" , $   |   epsilon   1   0   *
E ->  . "1" , $   |   epsilon   1   0   *
E ->  . "epsilon" , $   |   epsilon   1   0   *
( 1 , E ) ---> 2
( 1 , "0" ) ---> 3
( 1 , "1" ) ---> 4
( 1 , "epsilon" ) ---> 5

STATE 2
S -> E . "$" , ?
E -> E . "|"   E , $   |   epsilon   1   0   *
E -> E . E , $   |   epsilon   1   0   *
E -> E . "*" , $   |   epsilon   1   0   *
E ->  . E   "|"   E , $   |   epsilon   1   0   *
E ->  . E   E , $   |   epsilon   1   0   *
E ->  . E   "*" , $   |   epsilon   1   0   *
E ->  . "0" , $   |   epsilon   1   0   *
E ->  . "1" , $   |   epsilon   1   0   *
E ->  . "epsilon" , $   |   epsilon   1   0   *
( 2 , "|" ) ---> 6
( 2 , E ) ---> 7
( 2 , "*" ) ---> 8
( 2 , "0" ) ---> 3
( 2 , "1" ) ---> 4
( 2 , "epsilon" ) ---> 5
accept on $

STATE 3
E -> "0" .  , $   |   epsilon   1   0   *
reduction by rule   E   -->   "0"   on   ["$","|","epsilon","1","0","*"]

STATE 4
E -> "1" .  , $   |   epsilon   1   0   *
reduction by rule   E   -->   "1"   on   ["$","|","epsilon","1","0","*"]

STATE 5
E -> "epsilon" .  , $   |   epsilon   1   0   *
reduction by rule   E   -->   "epsilon"   on   ["$","|","epsilon","1","0","*"]

STATE 6
E -> E   "|" . E , $   |   epsilon   1   0   *
E ->  . E   "|"   E , $   |   epsilon   1   0   *
E ->  . E   E , $   |   epsilon   1   0   *
E ->  . E   "*" , $   |   epsilon   1   0   *
E ->  . "0" , $   |   epsilon   1   0   *
E ->  . "1" , $   |   epsilon   1   0   *
E ->  . "epsilon" , $   |   epsilon   1   0   *
( 6 , E ) ---> 9
( 6 , "0" ) ---> 3
( 6 , "1" ) ---> 4
```

```
( 6 , "epsilon" ) ---> 5

STATE 7
E -> E   E .  , $   |   epsilon   1   0   *
E -> E . "|"   E , $   |   epsilon   1   0   *
E -> E . E , $   |   epsilon   1   0   *
E -> E . "*" , $   |   epsilon   1   0   *
E ->  . E   "|"   E , $   |   epsilon   1   0   *
E ->  . E   E , $   |   epsilon   1   0   *
E ->  . E   "*" , $   |   epsilon   1   0   *
E ->  . "0" , $   |   epsilon   1   0   *
E ->  . "1" , $   |   epsilon   1   0   *
E ->  . "epsilon" , $   |   epsilon   1   0   *
( 7 , "|" ) ---> 6
( 7 , E ) ---> 7
( 7 , "*" ) ---> 8
( 7 , "0" ) ---> 3
( 7 , "1" ) ---> 4
( 7 , "epsilon" ) ---> 5
reduction by rule   E   -->   E E   on   ["$","|","epsilon","1","0","*"]

STATE 8
E -> E   "*" .  , $   |   epsilon   1   0   *
reduction by rule   E   -->   E "*"   on   ["$","|","epsilon","1","0","*"]

STATE 9
E -> E   "|"   E .  , $   |   epsilon   1   0   *
E -> E . "|"   E , $   |   epsilon   1   0   *
E -> E . E , $   |   epsilon   1   0   *
E -> E . "*" , $   |   epsilon   1   0   *
E ->  . E   "|"   E , $   |   epsilon   1   0   *
E ->  . E   E , $   |   epsilon   1   0   *
E ->  . E   "*" , $   |   epsilon   1   0   *
E ->  . "0" , $   |   epsilon   1   0   *
E ->  . "1" , $   |   epsilon   1   0   *
E ->  . "epsilon" , $   |   epsilon   1   0   *
( 9 , "|" ) ---> 6
( 9 , E ) ---> 7
( 9 , "*" ) ---> 8
( 9 , "0" ) ---> 3
( 9 , "1" ) ---> 4
( 9 , "epsilon" ) ---> 5
reduction by rule   E   -->   E "|" E   on   ["$","|","epsilon","1","0","*"]
```

A tabela LR(1) obtida a partir destes estados é dada a seguir.

	0	**1**	ε	**|**	*****	**$**	*E*
1	s3	s4	s5				g2
2	s3	s4	s5	s6	s8	a	g7
3	r4	r4	r4	r4	r4	r4	
4	r5	r5	r5	r5	r5	r5	
5	r6	r6	r6	r6	r6	r6	
6	s3	s4	s5				g9
7	s3/r2	s4/r2	s5/r2	s6/r2	s8/r2	r2	g7
8	r3	r3	r3	r3	r3	r3	
9	s3/r1	s4/r1	s5/r1	s6/r1	s8/r1	r1	g7

Observe que há vários conflitos nesta tabela.

Explique o significado de cada ação conflitante encontrada no estado 7 quando o próximo símbolo terminal na cadeia de entrada é o ‘ **0** ’.