

CONSTRUÇÃO DE COMPILADORES I [BCC328]

Segunda prova (2019–1)

03 de julho de 2019

Matrícula: _____

Nome: _____

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

Questão 1. Considere a linguagem de programação definida pela gramática a seguir:

```

0  S' → S $
1  S → S ; S
2  S → id := E
3  S → print ( L )

4  E → id
5  E → num
6  E → E + E
7  E → ( S , E )

8  L → E
9  L → L , E
    
```

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

	id	num	print	;	,	+	:=	()	\$	S	E	L
1	s4		s7								g2		
2				s3						a			
3	s4		s7								g5		
4						s6							
5				r1	r1					r1			
6	s20	s10					s8				g11		
7							s9						
8	s4		s7								g12		
9	s20	s10					s8				g15	g14	
10				r5	r5	r5		r5	r5				
11				r2	r2	s16			r2				
12				s3	s18								
13				r3	r3				r3				
14					s19			s13					
15				r8			r8						
16	s20	s10					s8				g17		
17				r6	r6	s16		r6	r6				
18	s20	s10					s8				g21		
19	s20	s10					s8				g23		
20				r4	r4	r4		r4	r4				
21							s22						
22				r7	r7	r7		r7	r7				
23				r9	s16			r9					

Determine se a cadeia de entrada

x := 2 ; print (x + 3 x)

(onde **x**, e **y** 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 2. Mostre que a gramática a seguir é LALR(1) mas não é SLR.

- 0 $S \rightarrow X \text{ \$}$
- 1 $X \rightarrow M \text{ a}$
- 2 $X \rightarrow \text{b} M \text{ c}$
- 3 $X \rightarrow \text{d} \text{ c}$
- 4 $X \rightarrow \text{b} \text{ d} \text{ a}$
- 5 $M \rightarrow \text{d}$

A seguir tem-se a tabela com os símbolos não terminais anuláveis, e os conjuntos *FIRST* e *FOLLOW*.

NT	NULLABLE	FIRST	FOLLOW
S	F	b d	
X	F	b d	\$
M	F	d	a c

Questão 3. Considere a seguinte gramática livre de contexto para uma linguagem de expressões de tipo:

- 0 $S \rightarrow T \text{ \$}$
- 1 $T \rightarrow T \text{ -> } T$
- 2 $T \rightarrow T * T$
- 3 $T \rightarrow \text{int}$

Esta linguagem têm um *tipo básico*, **int**, e tipos compostos. Um *tipo produto* é construído com o operador ***** e um *tipo função* é construído com o operador **->**.

O autômato de pilha para esta gramática é apresentada a seguir. Como há conflitos na tabela correspondente conclui-se que a gramática não é LR(1). Isto já era esperado, pois a gramática é ambígua.

```

state 1
S → .T $    ,?
T → .T -> T , $ -> *
T → .T * T , $ -> *
T → .int    , $ -> *
(1, T ) → 2
(1, int ) → 3

state 2
S → T . $    ,?
T → T . -> T , $ -> *
T → T . * T , $ -> *
(2, -> ) → 4
  
```

(2,*) \mapsto 5
accept on \$

state 3

$T \rightarrow \text{int} \cdot, \$ \rightarrow *$

reduction by rule $T \rightarrow \text{int}$ on

state 4

$T \rightarrow T \rightarrow \cdot T, \$ \rightarrow *$

$T \rightarrow \cdot T \rightarrow T, \$ \rightarrow *$

$T \rightarrow \cdot T * T, \$ \rightarrow *$

$T \rightarrow \cdot \text{int}, \$ \rightarrow *$

(4,T) \mapsto 6

(4,int) \mapsto 3

state 5

$T \rightarrow T * \cdot T, \$ \rightarrow *$

$T \rightarrow \cdot T \rightarrow T, \$ \rightarrow *$

$T \rightarrow \cdot T * T, \$ \rightarrow *$

$T \rightarrow \cdot \text{int}, \$ \rightarrow *$

(5,T) \mapsto 7

(5,int) \mapsto 3

state 6

$T \rightarrow T \rightarrow T \cdot, \$ \rightarrow *$

$T \rightarrow T \rightarrow \cdot T, \$ \rightarrow *$

$T \rightarrow T \rightarrow \cdot * T, \$ \rightarrow *$

(6, \rightarrow) \mapsto 4

(6,*) \mapsto 5

reduction by rule $T \rightarrow T \rightarrow T$ on

state 7

$T \rightarrow T * T \cdot, \$ \rightarrow *$

$T \rightarrow T \rightarrow \cdot T, \$ \rightarrow *$

$T \rightarrow T \rightarrow \cdot * T, \$ \rightarrow *$

(7, \rightarrow) \mapsto 4

(7,*) \mapsto 5

reduction by rule $T \rightarrow T * T$ on

Construa a tabela LR(1) a partir desta descrição dos estados do autônomo, **eliminando conflitos** de acordo com as seguintes regras de precedência e associatividade:

- * tem maior precedência que \rightarrow
- * é associativo à esquerda
- \rightarrow é associativo à direita