



PROCESADORES DE LENGUAJE

Ingeniería Informática
Especialidad de computación
Tercer curso, segundo cuatrimestre



Departamento de Informática y Análisis Numérico
Escuela Politécnica Superior
Universidad de Córdoba
Curso académico 2017 - 2018

Relación de ejercicios nº 4: ANÁLISIS SINTÁCTICO ASCENDENTE

1. La siguiente gramática genera la tabla SLR que se indica

- $$P = \{$$
- 1) $E \rightarrow T$
 - 2) $E \rightarrow E + T$
 - 3) $T \rightarrow P$
 - 4) $T \rightarrow T * P$
 - 5) $P \rightarrow F$
 - 6) $P \rightarrow F \wedge P$
 - 7) $F \rightarrow (E)$
 - 8) $F \rightarrow \text{identificador}$
 - 9) $F \rightarrow \text{número}$
- $$\}$$

Estado	Acción								Ira			
	+	*	^	()	identificador	número	\$	E	T	P	F
0				d5		d6	d7		1	2	3	4
1	d8							ACEPTAR				
2	r1	d9	r1	r1	r1	r1	r1	r1				
3	r3	r3	r3	r3	r3	r3	r3	r3				
4	r5	r5	d10	r5	r5	r5	r5	r5				
5				d5		d6	d7		11	2	3	4
6	r8	r8	r8	r8	r8	r8	r8	r8				
7	r9	r9	r9	r9	r9	r9	r9	r9				
8				d5		d6	d7			12	3	4
9				d5		d6	d7				13	4
10				d5		d6	d7				14	4
11	d8				d15							
12	r2	d9	r2	r2	r2	r2	r2	r2				
13	r4	r4	r4	r4	r4	r4	r4	r4				
14	r6	r6	r6	r6	r6	r6	r6	r6				
15	r7	r7	r7	r7	r7	r7	r7	r7				

a. Analiza las expresiones:

- $a^2 + b$
- $a^{(2 + b)}$

b. Calcula las clausuras de la colección canónica de LR(0) - elementos:

- $I_0 = \text{clausura } \{ E' \rightarrow \cdot E \}$
- $I_1 = \text{clausura } \{ E \rightarrow E \cdot + T \}$
- $I_2 = \text{clausura } \{ E \rightarrow T \cdot , T \rightarrow T \cdot * P \}$
- $I_3 = \text{clausura } \{ T \rightarrow P \cdot \}$
- $I_4 = \text{clausura } \{ P \rightarrow F \cdot , P \rightarrow F \cdot ^ P \}$
- $I_5 = \text{clausura } \{ F \rightarrow (\cdot E) \}$
- $I_6 = \text{clausura } \{ F \rightarrow \text{identificador} \cdot \}$
- $I_7 = \text{clausura } \{ F \rightarrow \text{número} \cdot \}$
- $I_8 = \text{clausura } \{ E \rightarrow E + \cdot T \}$
- $I_9 = \text{clausura } \{ T \rightarrow T * \cdot P \}$
- $I_{10} = \text{clausura } \{ P \rightarrow F ^ \cdot P \}$
- $I_{11} = \text{clausura } \{ E \rightarrow E \cdot + T , F \rightarrow (E \cdot) \}$
- $I_{12} = \text{clausura } \{ E \rightarrow E + T \cdot , T \rightarrow T \cdot * P \}$
- $I_{13} = \text{clausura } \{ T \rightarrow T * P \cdot \}$
- $I_{14} = \text{clausura } \{ P \rightarrow F ^ P \cdot \}$
- $I_{15} = \text{clausura } \{ F \rightarrow (E) \cdot \}$

c. Utiliza la colección canónica de LR(0) - elementos para completar la tabla de análisis SLR con funciones de recuperación de errores para aplicar el método de “nivel de frase”.

d. Analiza la siguiente expresión errónea: $((a ++ b ** 2)$

2. La siguiente gramática permite generar asignaciones múltiples en el lenguaje C:

- $P = \{$
- 1) $S \rightarrow S A$
 - 2) $S \rightarrow A$
 - 3) $A \rightarrow \text{identificador} = L ;$
 - 4) $L \rightarrow \text{número}$
 - 5) $L \rightarrow \text{identificador} = L$
- $\}$

Por ejemplo:

$a = b = 1;$
 $c = 2;$

- a. Construye la colección canónica de LR(0)-elementos
- b. Dibuja el autómata que reconoce los prefijos viables.
- c. Construye la tabla de análisis sintáctico SLR
- d. Utiliza el método recuperación de errores de “nivel de frase” para completar la tabla SLR.
- e. Utiliza la tabla SLR para analizar la siguiente declaración errónea:
 $= a = = b 1 ;$

3. Considera la siguiente gramática de contexto libre

- $P = \{$
- 1) $S \rightarrow S D$

- 2) $S \rightarrow \varepsilon$
- 3) $D \rightarrow \text{enum } \textit{identificador} \{ L \};$
- 4) $L \rightarrow \textit{identificador}$
- 5) $L \rightarrow L, \textit{identificador}$

Esta gramática puede generar algunas “enumeraciones” del lenguaje de programación C:

enum color {blanco, negro, amarillo};

- a. Construye la colección canónica de LR(0)-elementos
- b. Dibuja el autómata que reconoce los prefijos viables.
- c. Construye la tabla de análisis sintáctico SLR
- d. Utiliza el método recuperación de errores de “nivel de frase” para completar la tabla SLR.
- e. Utiliza la tabla SLR para analizar la siguiente declaración errónea:

color { blanco amarillo ;

4. Dada la gramática

- $$P = \left\{ \begin{array}{l} S \rightarrow L \text{ punto } L \mid L \\ L \rightarrow L \text{ dígito } \mid \text{dígito} \end{array} \right\}$$

donde **punto** representa el punto decimal (“.”)

- a. Construye la tabla de análisis sintáctico LR(1) - canónico.
- b. Construye la tabla de análisis sintáctico LALR(1).
- c. Añade a la tabla de análisis LALR(1) funciones de recuperación de errores para poder aplicar el método de nivel de frase.
- d. Analiza la cadena errónea: **3..12.1**.

5. Dada la gramática

- $$P = \left\{ \begin{array}{l} S \rightarrow \textit{función } \textit{identificador} (L) : T \\ L \rightarrow T \textit{ identificador} , L \\ L \rightarrow T \textit{ identificador} \\ L \rightarrow \varepsilon \\ T \rightarrow \textit{real} \\ T \rightarrow \textit{carácter} \end{array} \right\}$$

- a. Construye la tabla de análisis sintáctico LALR(1).
- b. Analiza la sentencia:
función media (real x, real y) : real
- c. Dada la sentencia errónea:
carácter función error real valor ,) : real

- Utiliza el método de “modo de pánico” para analizar esta sentencia errónea.
- Utiliza el método de “nivel de fase” para analizar esta sentencia errónea.

6. Análisis sintáctico **ascendente** de prototipos de funciones en C:

- a. Diseña una gramática que permita generar los prototipos de las funciones de C que sólo utilizan los tipos **int**, **char** y punteros a **int** o **char**.
- b. Construye la tabla de análisis sintáctico LALR y analiza las siguientes sentencias:
 - **int tasa ();**
 - **char * letras (int , char **, char);**
- c. Utiliza el método de nivel de fase para analizar la siguiente sentencia errónea:
 - **int char consultar ((int , , ;**

7. Demuestra que una gramática LR no puede ser ambigua.

8. Demuestra que nunca se van a producir errores al consultar la tabla “**ir_a**” durante un análisis sintáctico LR.