Procesadores de lenguajes

Ejercicios del Tema 4

INTRODUCCIÓN AL ANÁLISIS ASCENDENTE

Ejercicio 4.1

Considere la siguiente gramática:

$$\langle D\rangle \rightarrow \langle T\rangle\,\langle L\rangle \text{pyc}$$
$$\langle T\rangle \rightarrow \text{float}$$
$$\langle T\rangle \rightarrow \text{int}$$
$$\langle L\rangle \rightarrow \langle V\rangle$$
$$\langle L\rangle \rightarrow \langle L\rangle \text{coma}\,\langle V\rangle$$
$$\langle V\rangle \rightarrow \text{id}$$
$$\langle V\rangle \rightarrow \text{id asig num}$$

La siguiente figura muestra la tabla de análisis SLR de la gramática anterior:

	float	**int**	**id**	**num**	**asig**	**coma**	**pyc**	**$**	**D**	**T**	**L**	**V**
0	d3	d4							1	2		
1								aceptar				
2			d7								5	6
3			r2									
4			r3									
5						d9	d8					
6						r4	r4					
7					d10	r6	r6					
8								r1				
9			d7									11
10				d12								
11						r5	r5					
12						r7	r7					

Realizar la traza del análisis ascendente de la cadena "**float a = 3, b;**".

Ejercicio 4.2

Calcular el autómata reconocedor de prefijos viables para la siguiente gramática:

$$\langle I \rangle \rightarrow \langle V \rangle \text{asig} \langle E \rangle \text{pyc}$$
$$\langle V \rangle \rightarrow \text{id}$$
$$\langle V \rangle \rightarrow \langle V \rangle \text{punto id}$$
$$\langle E \rangle \rightarrow \text{num}$$
$$\langle E \rangle \rightarrow \langle V \rangle$$

Ejercicio 4.3

Dada la siguiente tabla SLR:

	a	**o**	**t**	**f**	**$**	**C**	**D**	**B**
0			d4	d5		1	2	3
1	d6				aceptar			
2	r2	d7			r2			
3	r4	r4			r4			
4	r5	r5			r5			
5	r6	r6			r6			
6			d4	d5			8	3
7			d4	d5				9
8	r1	d7			r1			
9	r3	r3			r3			

(a) ¿Cual es el símbolo inicial?

(b) ¿Cuantas reglas tiene la gramática?

(c) Dibuje el Autómata de Prefijos Viables que genera esta tabla

ALGORITMO SLR

Ejercicio 4.4

Considere la siguiente gramática, que describe las proposiciones lógicas basadas en los valores *true* y *false* y los operadores de conjunción (^), disyunción (v) y negación (¬):

$E \rightarrow E$ **v** C

$E \rightarrow C$

$C \rightarrow C$ ^ L

$C \rightarrow L$

$L \rightarrow \neg L$

$L \rightarrow$ **true**

$L \rightarrow$ **false**

$L \rightarrow$ **(** E **)**

Construya la tabla de análisis SLR de la gramática anterior.

Ejercicio 4.5

La siguiente gramática, al igual que la del ejercicio anterior, permite describir un texto formado por un único párrafo. Construya la tabla de análisis SLR de la gramática planteada.

Párrafo → *ListaDeFrases* **FinDeLínea**

ListaDeFrases → *Frase*

ListaDeFrases → *ListaDeFrases* *Frase*

Frase → *ListaDeCláusulas* **punto**

ListaDeCláusulas → *Cláusula*

ListaDeCláusulas → *ListaDeCláusulas* **coma** *Cláusula*

Cláusula → *Palabra*

Cláusula → *Cláusula* **espacio** *Palabra*

Palabra → **letra**

Palabra → *Palabra* **letra**

Ejercicio 4.6

La siguiente gramática, al igual que la del ejercicio anterior, representa la sintaxis de la definición de tipos en un lenguaje funcional. Construya la tabla de análisis SLR de la gramática planteada.

Type → **type** **id** **eq** *Decl* **semicolon**

Decl → *Decl* **rel** *BaseDecl*

Decl → *BaseDecl*

BaseDecl → **id**

BaseDecl → *List*

BaseDecl → *Tuple*

List → **lbra** *Decl* **rbra**

Tuple → **lpar** *DeclList* **rpar**

DeclList → *DeclList* **comma** *Decl*

DeclList → *Decl*

Ejercicio 4.7

La siguiente gramática representa la sintaxis de la instrucción de asignación de un lenguaje basado en conjuntos:

Asig → **id** **igual** *Expr* **pyc**

Expr → *Base*

Expr → *Expr* **union** *Base*

Expr → *Expr* **interseccion** *Base*

Base → **llave_ab** *Elem* **llave_ce**

Base → **par_ab** *Expr* **par_ce**

Elem → **num**

Elem → *Elem* **coma** **num**

A continuación se muestra un ejemplo de una cadena de entrada para esta gramática:

A = { 1 , 2 } ∪ ({ 3 , 4, 5 } ∩ { 3 , 5 }) ;

(a) Describa los estados y transiciones del autómata reconocedor de prefijos viables.

(b) Construya la tabla de análisis SLR de la gramática planteada.

Ejercicio 4.8

La siguiente gramática representa la sintaxis de las expresiones de un lenguaje basado en notación prefija:

1: *Expr* → **par_ab** *Operador Lista* **par_ce**

2: *Operador* → **id**

3: *Operador* → **plus**

4: *Operador* → **minus**

5: *Operador* → **prod**

6: *Operador* → **div**

7: *Lista* → *Lista* **coma** *Param*

8: *Lista* → λ

9: *Param* → **num**

10: *Param* → **id**

11: *Param* → *Expr*

A continuación se muestra un ejemplo de una cadena de entrada para esta gramática:

(min, (+, a, 3), (*, 2, b))

(a) Describa los estados y transiciones del autómata reconocedor de prefijos viables.

(b) Construya la tabla de análisis SLR de la gramática planteada.

(c) Desarrolle la traza (contenido de la pila, estado del flujo de entrada y acción a realizar en cada paso) del analizador SLR para la cadena presentada como ejemplo.

Ejercicio 4.9

La siguiente gramática permite describir una versión reducida de la sintaxis de Prolog. Construya el autómata reconocedor de prefijos viables y la tabla de análisis SLR de la gramática planteada.

Programa → *Claúsula*

Programa → *Programa Claúsula*

Claúsula → *Predicado* **dot**

Claúsula → *Predicado* **imp** *Literales* **dot**

Predicado → **atom**

Predicado → **atom lparen** *Literales* **rparen**

Literales → *Literal*

Literales → *Literales* **comma** *Literal*

Literal → **var**

Literal → **const**

Literal → *Predicado*

Literal → *Lista*

Lista → **lbracket** *Literales* **rbracket**

Lista → **lbracket** *Literales* **tail** *Literal* **rbracket**

Ejercicio 4.10

La siguiente gramática describe la sintaxis de las expresiones regulares:

Expr → *Option*

Expr → *Expr* **or** *Option*

Option → *Base*

Option → *Option Base*

Base → **symbol**

Base → **lparen** *Expr* **rparen** *Oper*

Oper → λ

Oper → **star**

Oper → **plus**

Oper → **hook**

Construya el autómata reconocedor de prefijos viables y la tabla de análisis SLR de la gramática planteada.

Ejercicio 4.11

Considere la siguiente gramática:

$S \rightarrow M$ **a**

$S \rightarrow$ **b** M **c**

$S \rightarrow$ **d c**

$S \rightarrow$ **b d a**

$M \rightarrow$ **d**

Construya el autómata reconocedor de prefijos viables y la tabla de análisis SLR de la gramática planteada. ¿Se trata de una gramática SLR? Justifique la respuesta.

Ejercicio 4.12

La siguiente gramática describe una versión muy reducida de la instrucción SELECT del lenguaje SQL:

Select → **select** *FieldList* **from** *TableList WhereClause* **semicolon**

FieldList → *Field*

FieldList → *FieldList* **comma** *Field*

Field → *TableRef FieldRef*

TableRef → **id dot**

TableRef → λ

FieldRef → **id**

FieldRef → **star**

TableList → **id**

TableList → *TableList* **comma id**

WhereClause → λ

WhereClause → **where** *Field* **equal value**

Construya el autómata reconocedor de prefijos viables y la tabla de análisis SLR de la gramática planteada. ¿Se trata de una gramática SLR? Justifique la respuesta.

Ejercicio 4.13

A continuación se presenta la gramática (reducida) de un lenguaje funcional:

Definition → **id** *Arguments* **eq** *Expression*

Arguments → **lbracket** *Variables* **rbracket**

Arguments → λ

Variables → **id**

Variables → *Variables* **semicolon id**

Expression → **number**

Expression → *Call*

Call → **id** *Parameters*

Parameters → **lbracket** *ExpressionList* **rbracket**

Parameters → λ

ExpressionList → *Expression*

ExpressionList → *ExpressionList* **semicolon** *Expression*

Construya el autómata reconocedor de prefijos viables y la tabla de análisis SLR de la gramática planteada.

Ejercicio 4.14

La siguiente gramática representa la sintaxis simplificada del operador *new*:

NewOperator → **new id** *Constructor*

NewOperator → **new id** *Array*

NewOperator → **new type** *Array*

Constructor → **lparen rparen**

Constructor → **lparen** *Parameters* **rparen**

Parameters → **expr**

Parameters → *Parameters* **comma expr**

Array → *DimArray*

Array → *Array* **lbracket rbracket**

DimArray → **lbracket expr rbracket**

DimArray → *DimArray* **lbracket expr rbracket**

Construya el autómata reconocedor de prefijos viables y la tabla de análisis SLR de la gramática planteada.

Ejercicio 4.15

A continuación se presenta la gramática de un lenguaje de representación de árboles:

Arbol → **tree id lbrace** *ListaDeNodos* **rbrace**

ListaDeNodos → *Nodo*

ListaDeNodos → *ListaDeNodos* **comma** *Nodo*

Nodo → *NodoHoja*

Nodo → *NodoInterno*

NodoHoja → **lbracket id rbracket**

NodoInterno → **id lbrace** *ListaDeNodos* **rbrace**

Construya el autómata reconocedor de prefijos viables y la tabla de análisis SLR de la gramática planteada.

Ejercicio 4.16

El lenguaje PGN (*Portable Game Notation*) permite representar partidas de ajedrez. A continuación se presenta la gramática simplificada de este lenguaje:

Game → *TagList MovementList* **result**

TagList → *Tag*

TagList → *TagList Tag*

Tag → **lbracket id string rbracket**

MovementList → *Movement*

MovementList → *MovementList Movement*

Movement → *Index* **move**

Index → **number**

Index → λ

Construya el autómata reconocedor de prefijos viables y la tabla de análisis SLR de la gramática planteada.

Ejercicio 4.17

La siguiente gramática permite describir expresiones vectoriales formadas por dos operaciones: la suma de vectores y el producto de un escalar por un vector.

Expr → *Factor*

Expr → *Expr* **plus** *Factor*

Expr → *Expr* **minus** *Factor*

Factor → **vector**

Factor → **num** **prod** *Factor*

Factor → **lpar** *Expr* **rpar**

(a) Construya el autómata reconocedor de prefijos viables de la gramática anterior

(b) Construya la tabla de análisis SLR de la gramática planteada.

(c) Realice la traza para el siguiente ejemplo

```
3 * v1 + 2 * ( v2 - 5 * v3 )
```

Ejercicio 4.18

La siguiente gramática permite describir un circuito formado por resistencias unidas en serie o en paralelo (el token ***serie*** se representa mediante el símbolo **-**, el token ***paralelo*** mediante el símbolo **|** y el token ***resistencia*** mediante un identificador):

Circuito → *Circuito* **paralelo** *CircuitoSerie*

Circuito → *CircuitoSerie*

CircuitoSerie → *CircuitoSerie* **serie** *Base*

CircuitoSerie → *Base*

Base → **resistencia**

Base → **parab** *Circuito* **parce**

(a) Construya la tabla SLR de la gramática planteada.

(b) Desarrolle la traza (contenido de la pila, estado del flujo de entrada y acción a realizar en cada paso) del analizador SLR para la siguiente cadena:

```
R1 - ( R2 | R3 )
```

Ejercicio 4.19

La siguiente gramática permite describir la estructura de una clase:

Clase → **class id llaveab** *ListaComp* **llavece**

ListaComp → *ListaComp Comp*

ListaComp → *Comp*

Comp → *Tipo* **id pyc**

Comp → **id parab** *ListaTipo* **parce pyc**

Comp → *Tipo* **id parab** *ListaTipo* **parce pyc**

ListaTipo → *ListaTipo* **coma** *Tipo*

ListaTipo → *Tipo*

ListaTipo → λ

Tipo → **int**

Tipo → **char**

(a) Construya la tabla SLR de la gramática planteada.

(b) Desarrolle la traza del analizador SLR para la siguiente cadena:

class id llaveab char id parab int coma int parce llavece

Ejercicio 4.20

La siguiente gramática representa la sintaxis de las cláusulas import en Java:

Cláusula → **import** *CoP* **semicolon**

CoP → *Clase*

CoP → *Paquete*

Paquete → *Clase* **dot star**

Clase → **id**

Clase → *Clase* **dot id**

Construya el autómata reconocedor de prefijos viables y la tabla de análisis SLR de la gramática planteada.