1. Dada la siguiente gramática:

S -> X Y

X -> X b Z | Z b Z | b Z

Y -> Y a Z | Z a Z | a Z

Z -> c

a) (0,5 pto.) ¿Es una gramática LL(1)? Justificad vuestra respuesta.

No, porque tiene recursividad a izquierda inmediata (X-> X b Z)

b) (0,5 pto.) Transformad la gramática anterior en otra equivalente que sea LL(1)

S-> X Y

X -> Z b Z X' | b Z X'

X' -> b Z X' | ε

Y -> Z a Z Y' | a Z Y'

Y' -> a Z Y' | ε

Z -> c

c) (0,5 pto.) Construid la tabla de análisis LL(1) para la gramática obtenida en el apartado b.

	a	***b***	***c***	***$***
S		*XY, 1*	*XY, 1*	
X		*bZX', 3*	*ZbZX', 2*	
X'	***ε, 5***	*bZX', 4*	***ε, 5***	
Y	*aZY', 7*		*ZaZY', 6*	
Y'	*aZY', 8*			***ε, 9***
Z			*c, 10*	
a	*sacar*			
b		*sacar*		
c			*sacar*	
$				*aceptar*

d) (0,5 pto.) Realizad la traza de análisis LL(1) para la cadena: "cbcbcac" usando la tabla del apartado anterior.

(S$, cbcbcac$,) |- (XY$, cbcbcac$, 1)
|- (ZbZX'Y$, cbcbcac$, 1-2) |- (cbZX'Y$, cbcbcac$, 1-2-10)
|- (bZX'Y$, bcbcac$, 1-2-10) |- (ZX'Y$, cbcac$, 1-2-10)
|- (cX'Y$, cbcac$, 1-2-10-10) |- (X'Y$, bcac$, 1-2-10-10)
|- (bZX'Y$, bcac$, 1-2-10-10-4) |- (ZX'Y$, cac$, 1-2-10-10-4)
|- (cX'Y$, cac$, 1-2-10-10-4-10) |- (X'Y$, ac$, 1-2-10-10-4-10)
|- (Y$, ac$, 1-2-10-10-4-10-5) |- (aZY'$, ac$, 1-2-10-10-4-10-5-7)
|- (ZY'$, c$, 1-2-10-10-4-10-5-7) |- (cY'$, c$, 1-2-10-10-4-10-5-7-10)
|- (Y'$, $, 1-2-10-10-4-10-5-7-10) |- ($, $, 1-2-10-10-4-10-5-7-10-9)
|- aceptar

2. Dada la siguiente gramática:

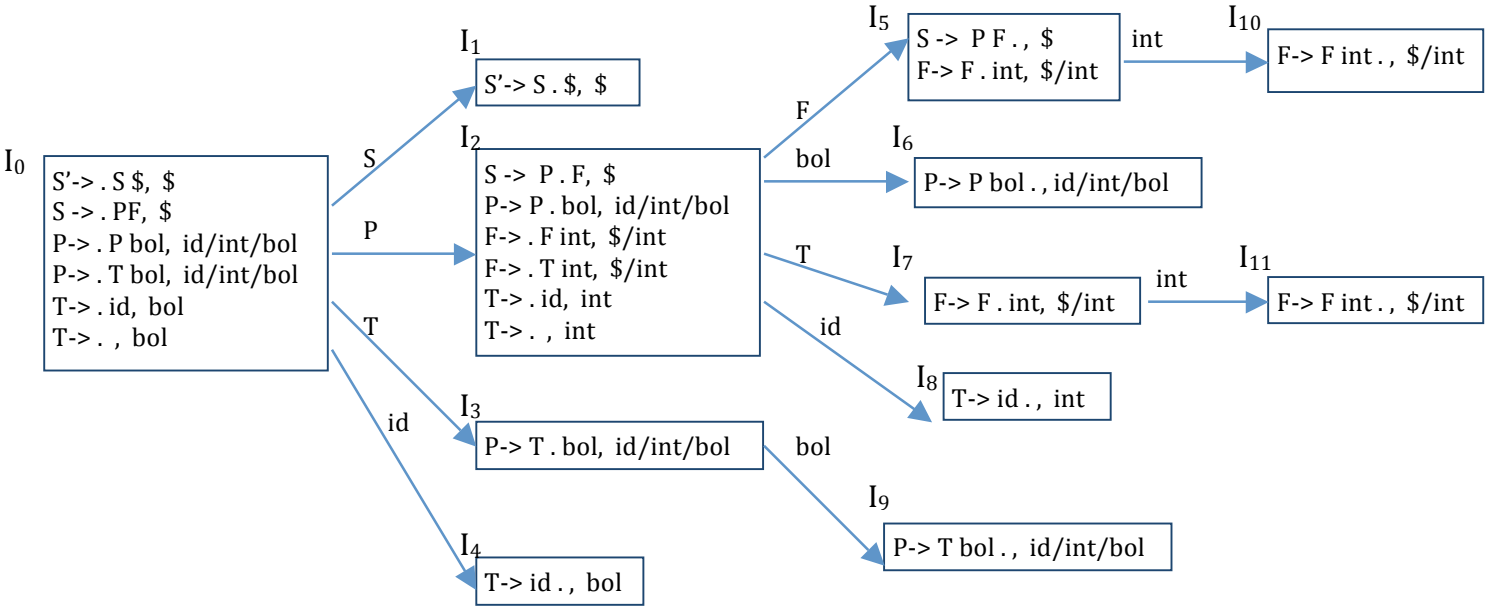
S -> P F

F -> F int | T int

P -> P bol | T bol

T -> id | **ε**

a) (1,5 pto.) Construid la colección canónica de conjuntos de elementos LR(1).



b) (1 pto.) Construid la tabla de análisis LALR(1).

	int	***bol***	***id***	***$***	***S***	***F***	***P***	***T***
0		*r-7*	*d4-8*		*1*		*2*	*3*
1				*aceptar*				
2	*r-7*	*d6*	*d4-8*			*5*		*7*
3		*d9*						
4-8	*r-6*	*r-6*						
5	*d10*			*r1*				
6	*r4*	*r4*	*r4*					
7	*d11*							
9	*r5*	*r5*	*r5*					
10	*r2*			*r2*				
11	*r3*			*r3*				

c) (0,5 pto.) ¿Es una gramática LALR(1)? ¿Es una gramática LR(1)? Justificad vuestras respuestas.

Es LALR(1) porque no hay conflictos en la tabla de análisis LALR(1). Es LR(1) porque toda gramática LALR(1) es LR(1).

d) (0,5 pto.) Realizad la traza de análisis LALR(1) para la cadena "id bol int"

(0, id bol int $,) |- (0 id 4-8, bol int $,) |- (0 T 3, bol int $, 6) |-
(0 T 3 bol 9, int $, 6) |- (0 P 2, int $, 6-5) |-(0 P 2 T 7, int $, 6-5-7)|-
(0 P 2 T 7 int 11, $, 6-5-7) |- (0 P 2 F 5, $, 6-5-7-3) |-(0 S 1, $, 6-5-7-3-1) |-
aceptar

Procesadores de Lenguajes 1er parcial 21/1/11

3.- La siguiente gramática genera expresiones booleanas bien sea mediante expresiones booleanas, igualdad entre expresiones aritméticas o conversión de tipos.

$S \rightarrow E$

$E \rightarrow E$ **and** E $\qquad$ $E \rightarrow$ **(** E **)**

$E \rightarrow E$ **or** E $\qquad$ $E \rightarrow E + E$

$E \rightarrow$ **not** E $\qquad$ $E \rightarrow E$ ***** E

$E \rightarrow E = E$ $\qquad$ $E \rightarrow$ **num**

a) Diseñad un ETDS que compruebe que los operadores **and**, **not** y **or** tienen como operandos expresiones lógicas y que los operadores = y + tienen operandos aritméticos.

Solución:

$S \rightarrow E$ { **print**("El tipo de la expresión es:" E.t }

$E \rightarrow E_1$ **and** E_2 { **if** E_1.t = E_2.t = t_logico **then** E.t = t_logico
else E.t = t_error; **print** ("Tipos incompatibles con el operador"}

| E_1 **or** E_2 { **if** E_1.t = E_2.t = t_logico **then** E.t = t_logico
else E.t = t_error; **print** ("Tipos incompatibles con el operador"}

| **not** E_1 { **if** E_1.t = t_logico **then** E.t = t_logico
else E.t = t_error; **print** ("Tipo incompatible con el operador"}

| **(** E_1 **)** { E.t = E_1.t }

| $E_1 = E_2$ { **if** E_1.t = E_2.t = t_aritmético **then** E.t = t_logico
else E.t = t_error; **print** ("Tipos incompatibles con el operador"}

| $E_1 + E_2$ { **if** E_1.t = E_2.t = t_aritmético **then** E.t = t_aritmético
else E.t = t_error; **print** ("Tipos incompatibles con el operador"}

| E_1 ***** E_2 { **if** E_1.t = E_2.t = t_aritmético **then** E.t = t_aritmético
else E.t = t_error; **print** ("Tipos incompatibles con el operador"}

| **num** { E.t = t_aritmético }

b) Escribid otro ETDS que calcule, en S, el valor lógico (verdadero o falso) de cualquier sentencia perteneciente al lenguaje generado por esta gramática.

Solución:

$S \rightarrow E$	{	**if** E.v = 0 **then print**("Falso")
		else print("Verdadero") **}**
$E \rightarrow E_1$ **and** E_2	{	$E.v = E_1.v * E_2.v$ **}**
| E_1 **or** E_2	{	**if** $E_1.v = 0 = E_2.v$ **then** E.v = 0 **else** E.v = 1 **}**
| **not** E_1	{	**if** $E_1.v = 0$ **then** E.v = 1 **else** E.v = 0 **}**
| **(** E_1 **)**	{	$E.v = E_1.v$ **}**
| $E_1 = E_2$	{	**if** $E_1.v = E_2.v$ **then** E.v = 1 **else** E.v = 0 **}**
| $E_1 + E_2$	{	$E.v = E_1.v + E_2.v$ **}**
| E_1 * E_2	{	$E.v = E_1.v * E_2.v$ **}**
| **num**	{	E.v = num.v **}**

Procesadores de Lenguajes (1º parcial)

21 de enero de 2011

4. Cuestiones teóricas (contestad brevemente):

   a) **(0.5 ptos.)** Construid una *expresión regular* (o un programa FLEX, si se quiere) para las constantes numéricas, que permita, por ejemplo: `3.14` `3.` `3` `0.14`

   ```
   digito  0 | 1 | ... | 9
   cte     (digito)+ ( ‘‘.’’ digito* ) ?
   ```

   b) **(0.5 ptos.)** Dada una *derivación a izquierdas* $S \stackrel{*}{\Rightarrow} wA\alpha$ y siendo a el símbolo actual de la cadena de entrada. ¿Qué parte de la forma sentencial $wA\alpha$ puede estar en la pila de un analizador LL(1) y qué condición se debe cumplir para que la siguiente acción sea derivar $A \to \beta$?

   $A\alpha$

   $a \in$ PRIMEROS (β SIGUIENTES(A)).

   c) **(0.5 ptos.)** Dada la gramática $\{ S \to a\ S\ b \mid a\ b \}$, indicad cual será el *pivote* de las siguientes formas sentenciales: $aaabbb$ y $aaSbb$

   `a a a b b b` `a a S b b`

   d) **(1 ptos.)** Para la misma gramática, describir todos sus prefijos viables. Para el prefijo viable a obtened todos sus ítem válidos LR(0).

   $S \mid a^+(b \mid S\ b?)$

   $\{[S \to a\ .\ S\ b], [S \to a\ .\ b], [S \to .\ a\ S\ b], [S \to .\ a\ b],\}$