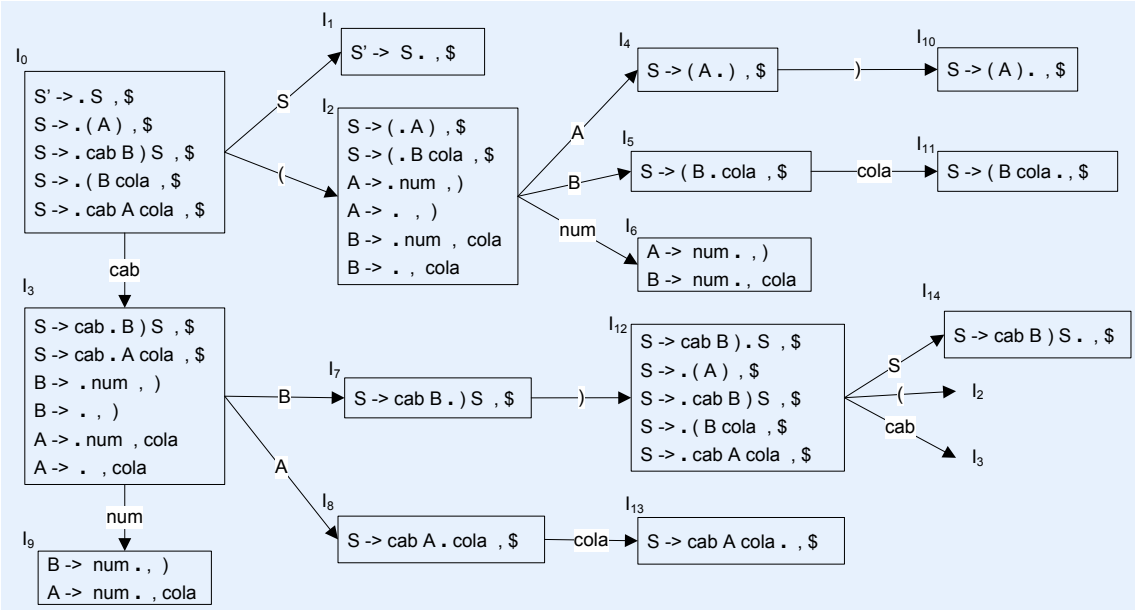
Procesadores de Lenguajes *1er parcial* *26/1/09*

1.- Dada la gramática:

S → **(** A **)**
| **cab** B **)** S
| **(** B **cola**
| **cab** A **cola**
A → **num** | ε
B → **num** | ε

a) *(2,5 pto.)* Construye la colección canónica de conjuntos de elementos LR(1).



b) *(0,25 pto.)* ¿Es una gramática LR(1)? Justifica tu respuesta.

Si que es LR(1) porque en ningún estado aparecenconflictos.

c) *(1 pto.)* Realiza de traza de análisis LR(1) para la cadena: "cab num) (num)".

[0, cab num) (num) $,] |- *[0 cab 3, num) (num) $,]* |-
[0 cab 3 num 9 ,) (num) $,] |- *[0 cab 3 B 7,) (num) $, 7]* |-
[0 cab 3 B 7) 12, (num) $, 7] |- *[0 cab 3 B 7) 12 (2, num) $, 7]* |-
[0 cab 3 B 7) 12 (2 num 6,) $, 7] |- *[0 cab 3 B 7) 12 (2 A 4,) $, 7-5]* |-
[0 cab 3 B 7) 12 (2 A 4) 10, $, 7-5] |- *[0 cab 3 B 7) 12 S 14, $, 7-5-1]* |-
[0 S 1, $, 7-5-1-2] |- *Aceptación*

d) *(1,5 pto.)* Construye la tabla de análisis LALR(1).

Se pueden fusionar los estados I_6 e I_9 con lo que se obtiene la tabla de análisis LALR(1) siguiente:

	(	)	cab	cola	num	$	S	A	B
0	d2		d3				1		
1						Acept			
2		r6		r8	d6-9			4	5
3		r8		r6	d6-9			8	7
4		d10							
5				d11					
6-9		r5 / r7		r7 / r5					
7		d12							
8				d13					
10						r1			
11						r3			
12	d2		d3				14		
13						r4			
14						r2			

e) *(0,25 pto.)* ¿Es una gramática LALR(1)? Justifica tu respuesta.

No es una gramática LALR(1) porque en el estado 6-9 hay dos conflictos reducción/reducción con los símbolos ")" y "cola"

2.- *(1 pto.)*

a) Dado el fragmento del ETDS de una calculadora, donde "num" representa un número entero o real:

$E \rightarrow E_1 + E_2$ *{ E.valor = E_1.valor + E_2.valor }*
$E \rightarrow$ **num** *{ E.valor = num.valor }*

¿Qué módulo del procesador del lenguaje proporciona valor al atributo num.valor?

El analizador léxico.

Indica si las siguientes afirmaciones son ciertas o falsas y justifica tu respuesta:

b) Puede existir una gramática SLR(1) que no sea LALR(1) pero no puede existir una gramática LL(1) que no sea LALR(1).

Es falsa ya que toda gramática SLR(1) es LALR(1) y hay gramáticas LL(1) que no son LALR(1)).

c) Sea una gramática independiente del contexto G. Si

$$S \Rightarrow^* \beta\ A\ a\ \omega \Rightarrow \beta\ \alpha\ a\ \omega$$

es una derivación más a derechas para G, entonces en su autómata LR(1) existirá un estado s, al que se llegará tras desplazar "$\beta\ \alpha$", que contendrá el elemento LR(1) [A-> α **.** , a]

Es cierta.
Viendo la derivación a derechas $S =>* \beta A a \omega => \beta \alpha a \omega$, y por la defición de elemento LR(1) válido para un prefijo viable, sabemos que el elemento [A-> α . , a] es válido para el prefijo viable "$\beta \alpha$". El estado "s" representa al estado que contiene a todos los elementos LR(1) válidos para el prefijo viable "$\beta \alpha$".
Esto mismo se puede justificar de la siguiente manera. El elemento LR(1) [A-> α . , a] indica que en la cima de la pila está la cadena α, y que en la forma sentencial a derechas esta cadena α debe ir seguida del símbolo "a". Efectivamente, la forma sentencial a derechas tiene como pivote a la producción A-> α, y está va seguida del símbolo "a". Además, para poder aplicar el pivote se habrán ido desplazando símbolos hasta tener en la cima de la pila el prefijo viable "$\beta \alpha$". En ese momento estaremos en un estado "s" en el que se deberá aplicar la reducción A-> α .

3.- Dada la gramática independiente del contexto:

E → **cte**
| **ct_octal**
| **ct_hexa**
| **(** OP LISTA **)**

OP → **+**
| *****
| **-**

LISTA → LISTA E
| E

Se pide:

a) *(1 pto.)* Un ETDS que compruebe que todos los operandos son compatibles con los operadores. Se considera que el operador suma y producto son compatibles con constantes enteras en base decimal, octal y hexadecimal, y que el operador resta sólo lo es con constantes en base decimal.

b) *(1 pto.)* Un ETDS que compruebe que el número de operandos en el caso de la suma o el producto es dos o mayor que dos, y en el caso de la resta exactamente dos.

c) *(0,75 pto.)* Escribir una especificación FLEX que reconozca los siguientes tokens:
- Una constante entera o decimal (por ejemplo: 220, 2.31) que no puede empezar por un cero ni un punto.
- Una constante entera en base octal. Siempre empezará por un cero e irá seguida de cualquier dígito octal. Ej. 021.
- Una constante entera en base hexadecimal. Empezará por 0x e irá seguida de dígitos hexadecimales. Ej. 0x3a.
- El resto de tokens de la gramática anterior.

La especificación debe eliminar blancos y tabuladores.

d) *(0,75 pto.)* Una especificación BISON para la anterior gramática que implemente el ETDS del apartado b).