Universidad Nacional Autónoma de México

Facultad de Ciencias

Compiladores

Tarea 4:

Autores:

Escamilla Soto Cristopher Alejandro
Montiel Manriquez Ricardo

06 de Mayo del 2022

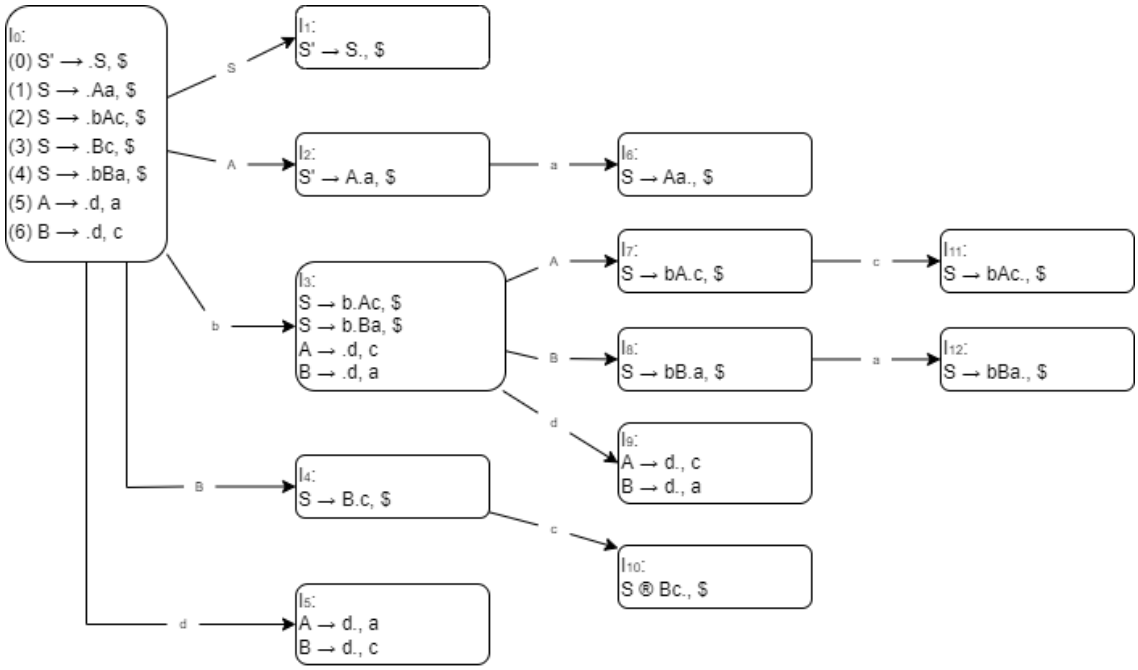
1. Demuestra que la siguiente gramática pertenece a la clase **LR(1)** pero no a la clase **LALR** mediante la construcción de la tabla de parsing.

$$E \to Aa \mid bAc \mid Bc \mid bBa \qquad A \to d \qquad B \to d$$

Agregaremos un regla gramatical como elemento inicial a las reglas anteriores:

$$S' \to S$$

Obtenemos el siguiente diagrama de estados:



Ahora podemos hacer la tabla de análisis sintáctico de **LR(1)** en base al diagrama de estados anterior:

State	Action					Goto		
	a	b	c	d	$	S	A	B
0		s3		s4		1	2	4
1					acc			
2	s6							
3				s9			7	8
4			s10					
5	r5		r6					
6								
7			s11					
8	s10							
9	r6		r5					
10					r3			
11					r3			
12					r3			

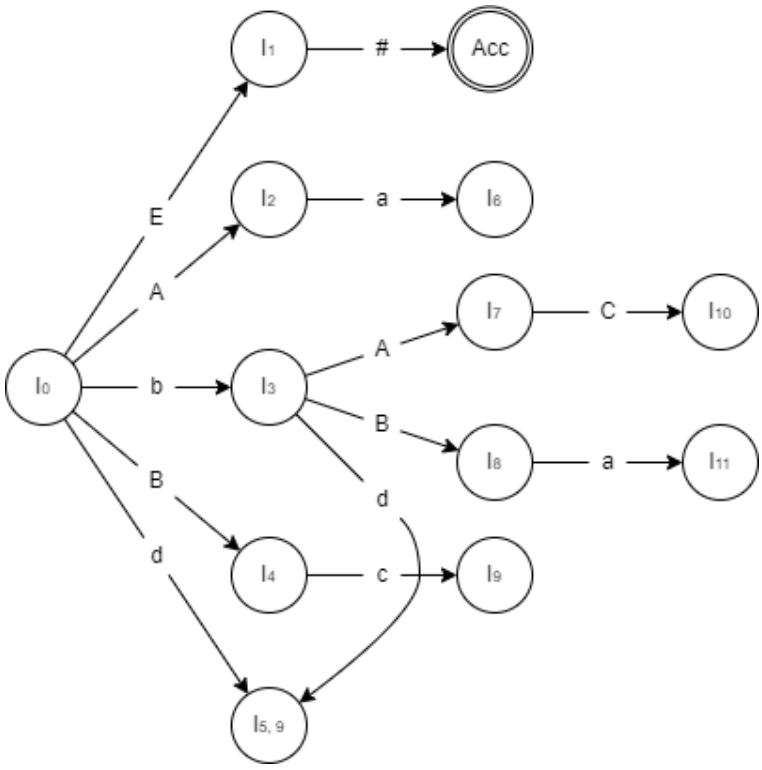
Como podemos ver no tenemos ninguna inconsistencia en la tabla, por lo que la gramática es del tipo **SLR(1)**

Pero es diferente al obtener la tabla de análisis mediante la fusión de estados de **LALR(1)** porque terminaremos fusionando los estados I_5 e I_9 y obtendríamos el siguiente estado:

$$\begin{aligned} I_{5+9} : A &\to d., \, a/c \\ B &\to d., \, a/c \end{aligned}$$

Esto es básicamente un conflicto de reducción, por lo que la gramática dada no seria del tipo **LALR(1)**

Su diagrama de estados seria de la siguiente forma:



Además analiza una cadena no trivial y de longitud al menos 3, mostrando la secuencia de acciones del autómata.

Cadena a analizar: **bda**

- **LR1:**

Stack	Symbol	Input	Action
0		bda#	s3
0, 3	b	da#	s9
0, 3, 9	b, d	a#	r6 $\Rightarrow$ B $\rightarrow$ d
0, 3	b, B	a#	s8
0, 3, 8	b, B	a#	s12
0, 3, 8, 12	b, B, a	#	r4 $\Rightarrow$ E $\rightarrow$ bBa
0	E	#	s1
0, 1	E	#	Accept

- **LALR:**

Stack	Symbol	Input	Action
0		bda#	s3
0, 3	b	da#	s(5, 9)
0, 3, (5, 9)	b, d	a#	r5 $\Rightarrow$ A $\rightarrow$ d
0, 3	b, A	a#	s7
0, 3, 7	b, A	a#	(No hay forma de continuar)
0, 3, (5, 9)	b, d	a#	r6 $\Rightarrow$ B $\rightarrow$ d
0, 3	b, B	a#	s8
0, 3, 8	b, B	a#	s11
0, 3, 8, 11	b, B, a	#	r4 $\Rightarrow$ E $\rightarrow$ bBa
0	E	#	s1
0, 11	E	#	Accept

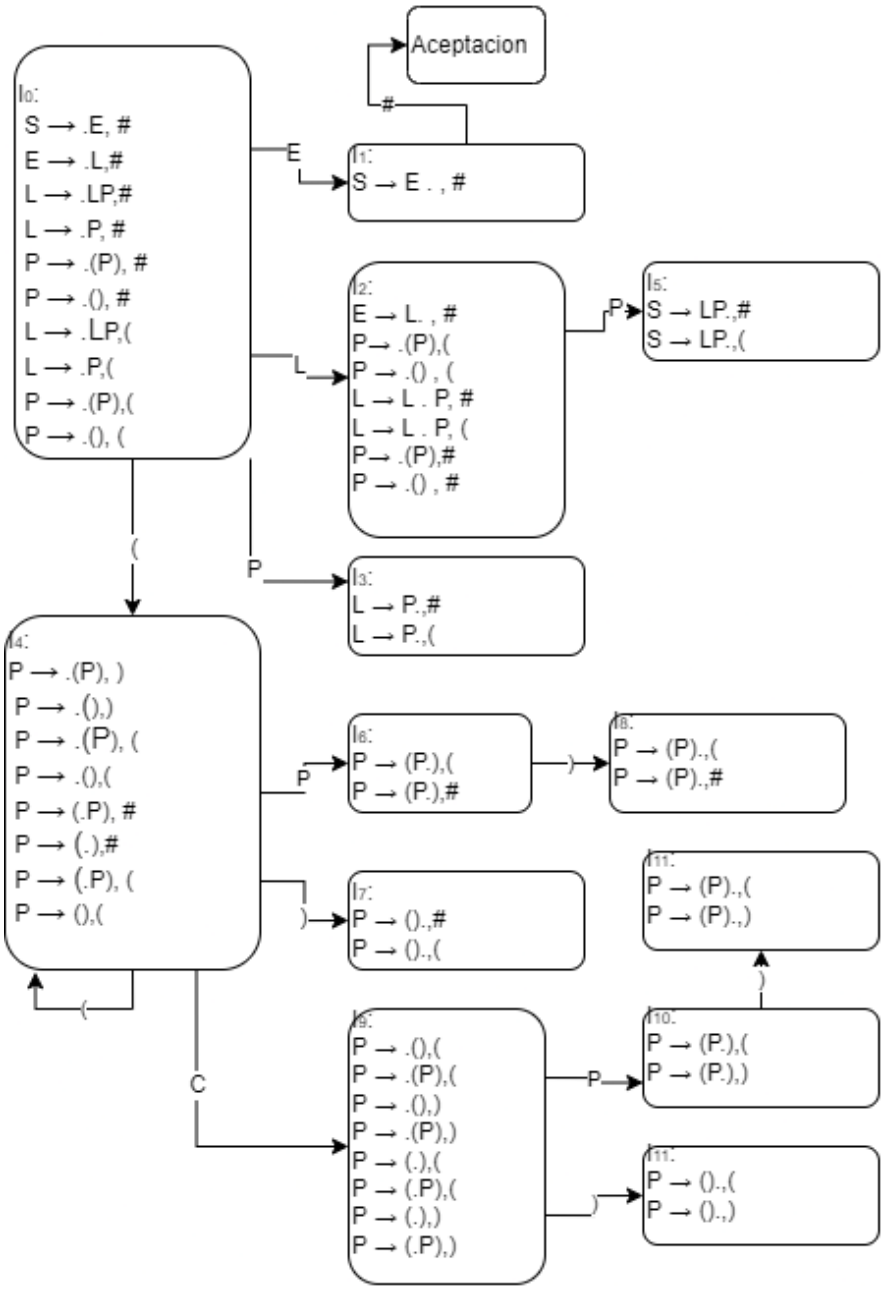
2. Da una propuesta de manejo de errores a través de subrutinas para las entradas vacías en la tabla de parsing **LR(1)** para la siguiente gramática que genera el lenguaje de paréntesis balanceados:

$$E \rightarrow L \mid \qquad L \rightarrow LP \mid P \qquad P \rightarrow (P) \mid ()$$

Demuestra su funcionamiento analizando la cadena ())(

Gramática extendida

S $\rightarrow$ E (R1)
E $\rightarrow$ L (R2)
L $\rightarrow$ LP (R3)
L $\rightarrow$ P (R4)
P $\rightarrow$ (P) (R5)
P $\rightarrow$ () (R6)


Aceptacion
#
I0:
S → .E, #
E → .L,#
L → .LP,#
L → .P, #
P → .(P), #
P → .(), #
L → .LP,(
L → .P,(
P → .(P),(
P → .(), (
E
I1:
S → E . , #
L
I2:
E → L. , #
P→ .(P),(
P → .() , (
L → L . P, #
L → L . P, (
P→ .(P),#
P → .() , #
P
I5:
S → LP.,#
S → LP.,(
(
P
I3:
L → P.,#
L → P.,(
I4:
P → .(P),)
P → .(),)
P → .(P), (
P → .(),(
P → (.P), #
P → (.),#
P → (.P), (
P → (),(
P
I6:
P → (P.),(
P → (P.),#
)
I8:
P → (P).,(
P → (P).,#
)
I7:
P → ().,#
P → ().,(
(
C
I9:
P → .(),(
P → .(P),(
P → .(),)
P → .(P),)
P → (.),(
P → (.P),(
P → (.),)
P → (.P),)
P
I10:
P → (P.),(
P → (P.),)
)
I11:
P → (P).,(
P → (P).,)
)
I11:
P → ().,(
P → ().,)

	Accion			Goto		
Pila	(	)	#	E	L	P
0	S4	E1	E2	1	2	3
1	E3	E3	ACEPTACION			
2	S4	E1	R1			5
3	R3	R3	R3			
4	S9	S7	E2			6
5	R2	R2	R2			
6	E4	S8	E2			
7	R5	R5	R5			
8	R4	R4	R4			
9	S9	S11	E2			10
10	E4	S12	E2			
11	R5	R5	R5			
12	R4	R4	R4			

Pila	Entrada	Accion	Simbolo
O	())(	S4	
O4	))(	S7	(
O47	)(	R5 P → ()	()
O	)(	S3	P
O3	)(	R3 L→ P	P
O	)(	S2	L
O2	)(	E1 ”(”	L
O24	)(	S7	L(
O247	(	R5 P → LP	L()
O2	(	S5	LP
O25	(	R2 l → LP	LP
O	(	S2	L
O2	(	S4	L
O29	#	E2 ”)”	L(
O247	#	R5 P → ()	L()
O2	#	S5	LP
O25	#	R2 L → LP	LP
O	#	S2	L
O2	#	R1 E → L	L
O	#	S1	E
O1	#	ACEPTACION	E

Los Errores son los siguientes:

E1: Ocurre en el estado I_2 cuando se espera que la cadena termine o empiece nuevamente, en el caso donde solamente recibamos un paréntesis derecho ')' lo agregamos a la pila I_4

E2: Ocurre en el estado I_4, donde se espera un paréntesis que abre '(' o cierra ')', si tenemos un antecedente donde tenemos uno que abre '(' podemos esperar un que cierra ')' y en caso de que tengamos uno que cierra '(' podemos seguir cerrando los paréntesis, entonces lo agregamos a I_7

E3: Ocurre en el estado I_6 e I_{10} cuando tenemos un antecedente P, entonces los paréntesis ya están balanceados por lo cual solo eliminamos el estado de la pila.

E4: Ocurre en el estado I_1 cuando ya tenemos como antecesor a E, solo eliminamos la pila y seguimos en I_1 para la aceptación de la cadena.

3. Argumenta detalladamente si los siguientes enunciados son verdaderos o falsos, también puedes dar contraejemplos:

   - Toda gramática que sea **SLR(1)** es no ambigua pero no toda gramática no ambigua es **SLR(1)**.

     Consideremos la siguiente gramatica con las siguientes producciones: [1]

     $$S \to L = R \mid R$$
     $$L \to *R \mid id \; R \to L$$

     Pensemos en L y R como el valor l y el valor r, y $*$ como un operador que indica "contenido de".

     $\mathbf{I}_0 = \{[S' \to \cdot S]$
     $[S \to \cdot L = R]$
     $[S \to \cdot R]$
     $[L \to \cdot * R]$
     $[L \to \cdot id]$
     $[R \to \cdot L]\}$

     $\mathbf{I}_1 = \{[S' \to S\cdot]\}$

[1] `https://n9.cl/6mrbs`

$\mathbf{I}_2 = \{[S \to L\cdot = R]$

$[R \to L\cdot]\}$

$\mathbf{I}_3 = \{[S \to R\cdot]\}$

$\mathbf{I}_4 = \{[L \to *\cdot R]\}$

$[R \to \cdot L]$

$[L \to \cdot * R]$

$[L \to \cdot id]\}$

$\mathbf{I}_5 = \{[L \to id\cdot]\}$

$\mathbf{I}_6 = \{[S \to L = \cdot R]\}$

$[R \to \cdot L]$

$[L \to \cdot * R]$

$[L \to \cdot id]\}$

$\mathbf{I}_7 = \{[L \to *R\cdot]\}$

$\mathbf{I}_8 = \{[R \to L\cdot]\}$

$\mathbf{I}_9 = \{[S \to L = R\cdot]\}$

Consideremos el conjunto de elementos de I_2. El primer elemento de este conjunto hace ACTION[2, =] que es 'shift 6'. Como FOLLOW(R) contiene = (para ver por que, consideramos la derivacion $S \to L \to R \to *R = R$, el segundo elemento establece ACTION[2, =]), para hacer "reduce $R \to L$". Dado que hay un 'shift' como un 'reduce' en n ACTION[2; =], el estado 2 tiene un conflicto de 'shift/reduce' en el símbolo de entrada '='.

La gramática dada previamente no es ambigua, mas bien este conflicto de 'shift/-reduce' surge del hecho de que el método de construcción del analizador SLR no es lo suficientemente poderoso para recordar suficiente contexto izquierdo como para decidir qué acción debe tomar el analizador en la entrada '=', habiendo visto una cadena reducible para L.

- Toda gramática que tenga una tabla de parsing **SLR(1)** sin conflictos es también **LR(1)**.
- Existen gramáticas sin ambigüedad para las que cualquier método **LR** producirá una tabla de parsing con conflictos.