



Universidad Nacional Autónoma de México

FACULTAD DE CIENCIAS

TAREA 04

Compiladores

Bonilla Reyes Dafne
Castañón Maldonado Carlos Emilio
García Ponce José Camilo
Velasco García Jorge Daniel

Profesora: Lourdes del Carmen González Huesca

Ayudante: Fausto David Hernández Jasso
Ayudante: Juan Alfonso Garduño Solís
Ayudante: Juan Pablo Yamamoto Zazueta



1. (2 pts) Considera la siguiente gramática donde E es el símbolo inicial:

$$\begin{aligned} E &\rightarrow Aa \\ A &\rightarrow BC|BCf \\ B &\rightarrow b \\ C &\rightarrow c \end{aligned}$$

- a) Construye los conjuntos canónicos de ítems $\text{LR}(0)$.
- b) Con estos conjuntos construye el autómata finito mostrando las transiciones con la función GoTo .
- c) Muestra la tabla de parsing que se genera usando el autómata anterior.

Conjuntos canónicos de ítems $\text{LR}(0)$

Para obtener los conjuntos canónicos de ítems $\text{LR}(0)$ primero obtenemos la gramática aumentada y el primer I_0 , que serán los siguientes:

1) $E \rightarrow E'$	$E \rightarrow \bullet E'$
2) $E \rightarrow Aa$	$E \rightarrow \bullet Aa$
3) $A \rightarrow BC$	$A \rightarrow \bullet BC$
4) $A \rightarrow BCf$	$A \rightarrow \bullet BCf$
5) $B \rightarrow b$	$B \rightarrow \bullet b$
6) $C \rightarrow c$	$C \rightarrow \bullet c$

$$\Rightarrow I_0 = \text{Closure}(\{E' \rightarrow \bullet E\})$$

De esta forma, tendremos los siguientes conjuntos:

Estado 0	
$E' \rightarrow \bullet E$	$\text{GoTo}(E) = I_3$
$E \rightarrow \bullet Aa$	$\text{GoTo}(A) = I_1$
$A \rightarrow \bullet BC$	$\text{GoTo}(B) = I_2$
$A \rightarrow \bullet BCf$	
$B \rightarrow \bullet b$	$\text{GoTo}(b) = I_4$

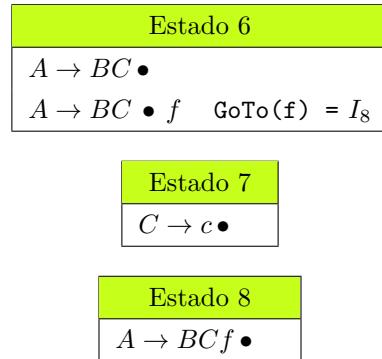
Estado 1	
$E \rightarrow A \bullet a$	$\text{GoTo}(a) = I_5$

Estado 2	
$A \rightarrow B \bullet C$	$\text{GoTo}(C) = I_6$
$A \rightarrow B \bullet Cf$	
$C \rightarrow \bullet c$	$\text{GoTo}(c) = I_7$

Estado 3	
$E' \rightarrow E \bullet$	

Estado 4	
$B \rightarrow b \bullet$	

Estado 5	
$E \rightarrow Aa \bullet$	



Autómata Finito con Transiciones GoTo

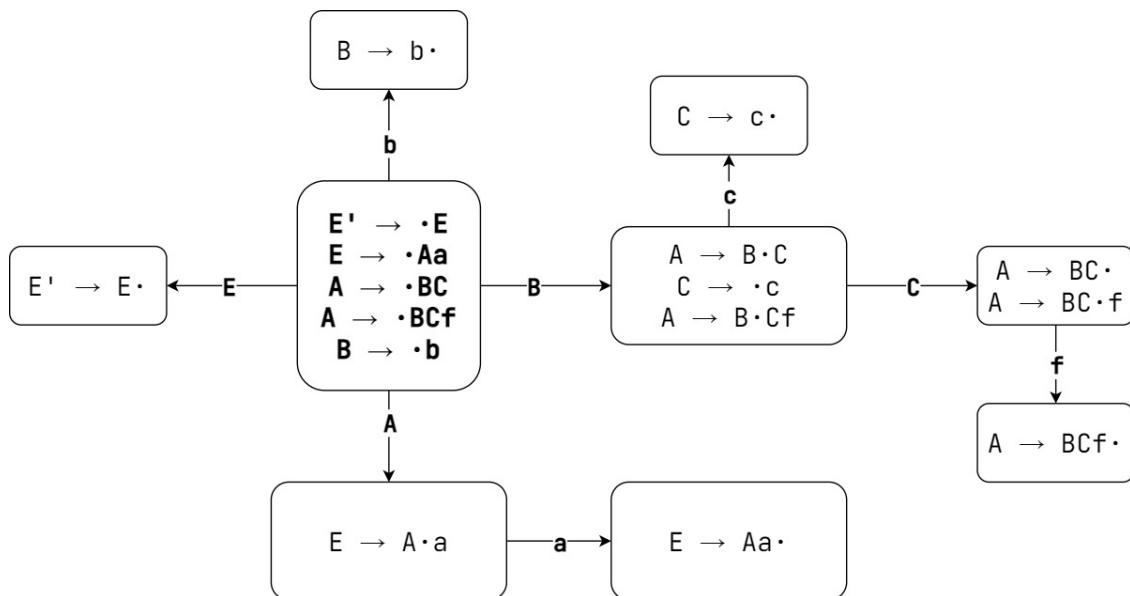


Tabla de Parsing

Estado	Acción					GoTo			
	a	b	c	f	\$	A	B	C	E
0		s4				1	2		3
1	s5								
2			s7						6
3					acc				
4	r5	r5	r5	r5	r5	r5			
5	r2	r2	r2	r2	r2	r2			
6	r3	r3	r3	r3 & s8		r3			
7	r6	r6	r6	r6	r6	r6			
8	r4	r4	r4	r4	r4	r4			

Notemos que en el estado 6 existe un conflicto con f , ya que $r3$ y $s8$ debido a que ambos deben estar ahí. Por lo tanto, podemos decir que la gramática **no es LR(0)**.



2. Considera la gramática para tipos, donde $- >$, \times e `int` son símbolos terminales:

$$T \rightarrow T - > T \mid T \times T \mid \text{int}$$

a) Calcula el resultado de $Follow(T)$ considerando un nuevo símbolo inicial.

Ya que queremos calcular el resultado de $Follow(T)$ considerando un nuevo símbolo inicial:

$$\begin{aligned} S &\rightarrow T \\ T &\rightarrow T - > T \\ T &\rightarrow T \times T \\ T &\rightarrow \text{int} \end{aligned}$$

Ahora, procedemos a calcular $FIRST$:

► $FIRST(S)$:

$$S \rightarrow T \Rightarrow FIRST(T)$$

► $FIRST(T)$:

$$\begin{aligned} T \rightarrow T - > &\Rightarrow FIRST(T) = FIRST(T) \\ T \rightarrow T \times T &\Rightarrow FIRST(T) = FIRST(T) \\ T \rightarrow \text{int} &\Rightarrow FIRST(T) = FIRST(\text{int}) = \{\text{int}\} \end{aligned}$$

► $FIRST(S) := \{\text{int}\}$

Conjunto de $FIRST$:

$$\begin{aligned} FIRST(S) &= \{\text{int}\} \\ FIRST(T) &= \{\text{int}\} \end{aligned}$$

Con todo lo anterior, ahora procedemos a calcular $FOLLOW$.

► $FOLLOW(S)$:

$$FOLLOW(S) = \{\#\}$$

► $FOLLOW(T)$:

$$\begin{aligned} S &\rightarrow T \\ A &\rightarrow \alpha B \end{aligned}$$

$$\Rightarrow FOLLOW(T) = FOLLOW(S) = \{\#\}$$

$$\begin{aligned} T &\rightarrow T - > T \\ A &\rightarrow \alpha B \beta \end{aligned}$$

$$\Rightarrow FOLLOW(T) = FIRST(- >) = \{- >\}$$

$$\begin{aligned} T &\rightarrow T \times T \\ A &\rightarrow \alpha B \beta \end{aligned}$$



$$\Rightarrow FOLLOW(T) = FIRST(\times) = \{\times\}$$

Conjunto de *FOLLOW*:

$$\begin{aligned} FOLLOW(S) &= \{\#\} \\ FOLLOW(T) &= \{\#, -, \times\} \end{aligned}$$

- b) Construye la tabla de parsing **SLR** para la gramática.

Recordando el conjunto de *FOLLOW*:

$$\begin{aligned} FOLLOW(S) &= \{\#\} \\ FOLLOW(T) &= \{\#, -, \times\} \end{aligned}$$

A su vez que nuestras reglas:

- 1) $S \rightarrow T$
- 2) $T \rightarrow T - > T$
- 3) $T \rightarrow T \times T$
- 4) $T \rightarrow int$

Y estos serian los estados:

Estado 0	
$S \rightarrow \bullet T$	GoTo(T) = I_1
$T \rightarrow \bullet T - > T$	
$T \rightarrow \bullet T \times T$	
$T \rightarrow \bullet int$	GoTo(int) = I_2

Estado 1	
$S \rightarrow T \bullet$	
$T \rightarrow T \bullet - > T$	GoTo(->) = I_3
$T \rightarrow T \bullet \times T$	GoTo(\times) = I_4

Estado 2	
$T \rightarrow int \bullet$	

Estado 3	
$T \rightarrow T - > \bullet T$	GoTo(T) = I_5
$T \rightarrow \bullet T - > T$	
$T \rightarrow \bullet T \times T$	
$T \rightarrow \bullet int$	GoTo(int) = I_2

Estado 4	
$T \rightarrow T \times \bullet T$	GoTo(T) = I_6
$T \rightarrow \bullet T - > T$	
$T \rightarrow \bullet T \times T$	
$T \rightarrow \bullet int$	GoTo(int) = I_2



Estado 5	
$T \rightarrow T - > T \bullet$	
$T \rightarrow T \bullet - > T$	GoTo($->$) = I_3
$T \rightarrow T \bullet \times T$	GoTo(\times) = I_4

Estado 6	
$T \rightarrow T \times T \bullet$	
$T \rightarrow T \bullet - > T$	GoTo($->$) = I_3
$T \rightarrow T \bullet \times T$	GoTo(\times) = I_4

Con esto en mente, procedemos a construir la tabla de parsing :

Estado	Acción				GoTo
	$- >$	\times	int	\$	
0			$s2$		1
1	$s3$	$s4$		acc	
2	$r4$	$r4$		$r4$	
3			$s2$		5
4			$s2$		6
5	$r2 \& s3$	$r2 \& s4$		$r2$	
6	$r3 \& s3$	$r3 \& s4$		$r3$	

Notese como es que existen conflictos en 5 y en 6 , por parte de las r con $r2$ y $r3$, mientras que los conflictos con las s , son por parte de $s3$ y de $s4$, lo cual hasta cierto punto es normal debido a todo lo hecho anteriormente.

- c) Elimina los conflictos existentes usando reglas de precedencia:

- ★ \times tiene mayor precedencia que $- >$
- ★ \times se asocia a la izquierda
- ★ $- >$ se asocia a la derecha

Veamos como resolver cada conflicto con las reglas de precedencia

Primero, en $M[5, - >]$, elegimos hacer shift a 3, debido a que $- >$ asocia a la derecha por lo tanto primero se reduce lo que esta a la derecha (los $- >$ más al final de la cadena)

Luego, en $M[5, \times]$, elegimos hacer shift a 4, debido a que como \times tiene mayor precedencia que $- >$ por lo tanto primero se debe reducir a \times antes que a $- >$

Después, en $M[6, - >]$, elegimos hacer reduce con la producción 3, debido a que como \times tiene mayor precedencia que $- >$ por lo tanto primero se debe reducir a \times antes que a $- >$

Y por ultimo, en $M[6, \times]$, elegimos hacer reduce con la producción 3, debido a que \times asocia a la izquierda por lo tanto primero se reduce lo que esta a la izquierda (los \times más al inicio de la cadena)

Obteniendo una tabla así

Estado	Acción				GoTo
	$- >$	\times	int	\$	
0			$s2$		1
1	$s3$	$s4$		acc	
2	$r4$	$r4$		$r4$	
3			$s2$		5
4			$s2$		6
5	$s3$	$s4$		$r2$	
6	$r3$	$r3$		$r3$	



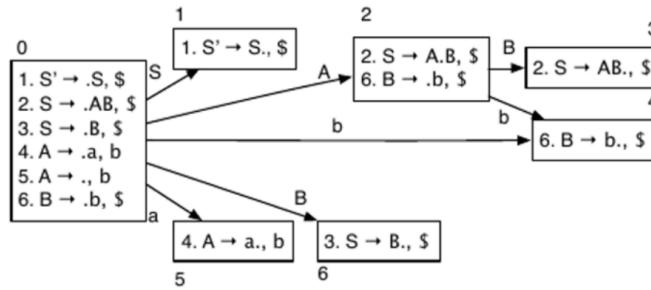
Otra forma de resolver estos conflictos usando la precedencia puede ser, ajustando la gramática para que ya incluya lo que queremos, aunque cambiando la gramática es un poco más laborioso, ya que necesitamos calcular los estados de nuevo

Pero la gramática quedaría algo así:

- 1) $S \rightarrow T$
- 2) $T \rightarrow M - > T$
- 3) $T \rightarrow M$
- 4) $M \rightarrow M \times D$
- 5) $M \rightarrow D$
- 6) $D \rightarrow int$



3. Considera el siguiente autómata LR(1):



- a) Proporciona la tabla completa con las acciones y las transiciones entre estados.

Estado	Acción			GoTo		
	a	b	\$	A	B	S
0	s5	r5 & s4		2	6	1
1			acc			
2		s4			3	
3			r2			
4			r6			
5		r4				
6			r3			

- b) Hay un estado que tiene un conflicto, ¿cuál es el estado? Describe el tipo de conflicto, además de explicarlo con sus propias palabras y proveer un mecanismo para resolverlo.

El estado que genera el conflicto es el inicial, debido a que se tiene un conflicto de tipo desplazamiento-reducción. Esto quiere decir que específicamente en la casilla $M[0, b]$ se tiene una acción de desplazamiento y una de reducción.

Esto debido a que tenemos que decidir si hacer reduce con la regla $A \rightarrow \epsilon$ o hacer shift al estado 4 (que sería leer la b para luego hacer reduce con $B \rightarrow b$), todo esto es generado ya que la gramática es ambigua, podemos hacer dos derivaciones para la cadena "b", $S \rightarrow AB \rightarrow \epsilon B \rightarrow B$ y $S \rightarrow B \rightarrow b$.

Por lo tanto, una solución consiste en eliminar la ambigüedad que existe inicialmente en la gramática que tiene el autómata, eliminando la producción $A \rightarrow \epsilon$ de la gramática y luego de esto la gramática nos queda igual solo que sin esta producción, por lo tanto el autómata queda similar solo que el estado 0 ya que no tiene el inciso 5 y la tabla de parsing quedaría igual pero sin el conflicto (solo se mantendría el shift al estado 4 y el reduce ya no aparece).



4. Calcula la tabla de parsing **LALR** para la gramática

$$E \rightarrow (L)|a \quad L \rightarrow L, E \mid E$$

además de ejecutar el análisis para la cadena ((a,a),a,(a))

Ejercicio 4

Calcula la tabla de parsing LALR para la gramática

$$E \rightarrow (L) \mid a \quad L \rightarrow L, E \mid E$$

además de ejecutar el análisis para la cadena $((a,a),a,(a))$

gramática aumentada

$$(0) \quad S \rightarrow E \quad (3) \quad L \rightarrow L, E$$

$$(1) \quad E \rightarrow (L) \quad (4) \quad L \rightarrow E$$

$$(2) \quad E \rightarrow a$$

estados

$$\left[\begin{array}{l} I_0: S \rightarrow \bullet E, \$ \\ E \rightarrow \bullet (L), \$ \xrightarrow{\text{First}(E\$)=\$} \\ E \rightarrow \bullet a, \$ \end{array} \right]$$

$$\left[I_1: S \rightarrow E \bullet, \$ \xrightarrow{\text{heredado}} \right]$$

$I_2: E \rightarrow (\circ L), \$ \rightarrow \text{heredado}$

$L \rightarrow \circ L, E,) \rightarrow \text{First}(,)\$) =)$

$L \rightarrow \circ E,)$

$L \rightarrow \circ L, E, ; \rightarrow \text{First}(;E)) = ;$

$L \rightarrow \circ E, ;$

$E \rightarrow \circ (L),) \rightarrow \text{First}(\varepsilon)) =)$

$E \rightarrow \circ a,)$

$E \rightarrow \circ (L), ; \rightarrow \text{First}(\varepsilon;) = ;$

$E \rightarrow \circ a, ;$

$I_2: E \rightarrow (\circ L), \$$

$L \rightarrow \circ L, E,) / ;$

$L \rightarrow \circ E,) / ;$

$E \rightarrow \circ (L),) / ;$

$E \rightarrow \circ a,) / ;$

$[I_3 : E \rightarrow a \circ, \$ \rightarrow \text{heredado}]$

$[I_4 : E \rightarrow (L \circ), \$ \rightarrow \text{heredado}]$
 $L \rightarrow L \circ, E,) / , \rightarrow \text{heredado}$

$[I_5 : L \rightarrow E \circ,) / , \rightarrow \text{heredado}]$

$I_6 : E \rightarrow (\circ L),) / , \rightarrow \text{heredado}$

$L \rightarrow \circ L, E,) \rightarrow \text{First}()) / \text{First}(,) =)$
 $L \rightarrow \circ E,)$

$L \rightarrow \circ L, E, , \rightarrow \text{First}(, E) = ,$
 $L \rightarrow \circ E, ,$

$E \rightarrow \circ (L),) \rightarrow \text{First}(E) =)$
 $E \rightarrow \circ a,)$

$E \rightarrow \circ (L), , \rightarrow \text{First}(E ,) = ,$
 $E \rightarrow \circ a, ,$

$$\left[\begin{array}{l} I_6 : E \rightarrow (\circ L),) / g \\ L \rightarrow \circ L, E,) / g \\ L \rightarrow \circ E,) / g \\ E \rightarrow \circ (L),) / g \\ E \rightarrow \circ a,) / g \end{array} \right]$$

$$\left[I_7 : E \rightarrow a \circ,) / g \rightarrow \text{heredado} \right]$$

$$\left[I_8 : E \rightarrow (L) \circ, \$ \rightarrow \text{heredado} \right]$$

$$\left[\begin{array}{l} I_9 : L \rightarrow L, \circ E,) / g \rightarrow \text{heredado} \\ E \rightarrow \circ (L),) / g \rightarrow \text{First}(E) / \text{First}(E_g) =) / g \\ E \rightarrow \circ a,) / g \end{array} \right]$$

$$\left[\begin{array}{l} I_{10} : E \rightarrow (L \circ),) / g \rightarrow \text{heredado} \\ L \rightarrow L \circ, E,) / g \rightarrow \text{heredado} \end{array} \right]$$

$$\left[I_{11} : L \rightarrow L, E \circ,) / g \rightarrow \text{heredado} \right]$$

$[I_{12} : E \rightarrow (L)^\circ,)/g \rightarrow \text{heredado}]$

automata

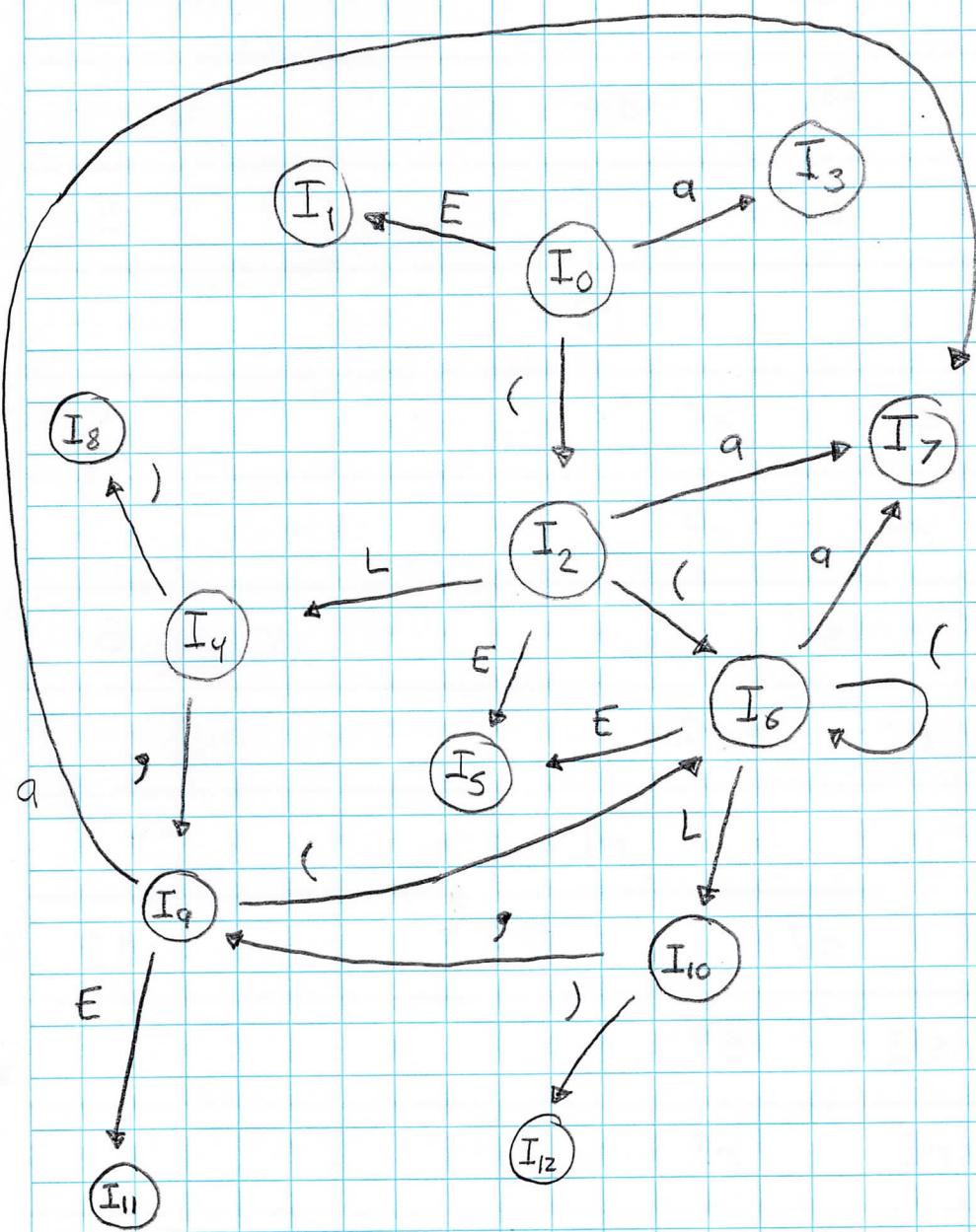


tabla LR(1)

estado	()	a	,	\$	S _i	GoTo L	E
0	s ₂	s ₃						1
1				acc				
2	s ₆	s ₇				4	5	
3				r ₂				
4	s ₈	s ₉						
5	r ₄	r ₄						
6	s ₆	s ₇				10	5	
7		r ₂	r ₂					
8				r ₁				
9	s ₆	s ₇					11	
10		s ₁₂	s ₉					
11		r ₃	r ₃					
12		r ₁	r ₁					

buscar nucleos iguales y combinar estados

2 y 6 3 y 7 4 y 10 8 y 12

estado original	estado	GoTo					
		()	a	,	\$	S	L
0 =	0	s2	s3				1
1 =	1				acc		
2 y 6 =	2	s2	s3			4	5
3 y 7 =	3		r2	r2	r2		
4 y 10 =	4		s6	s7			
5 =	5		r4	r4			
8 y 12 =	6		r1	r1	r1		
9 =	7	s2	s3				8
11 =	8		r3	r3			

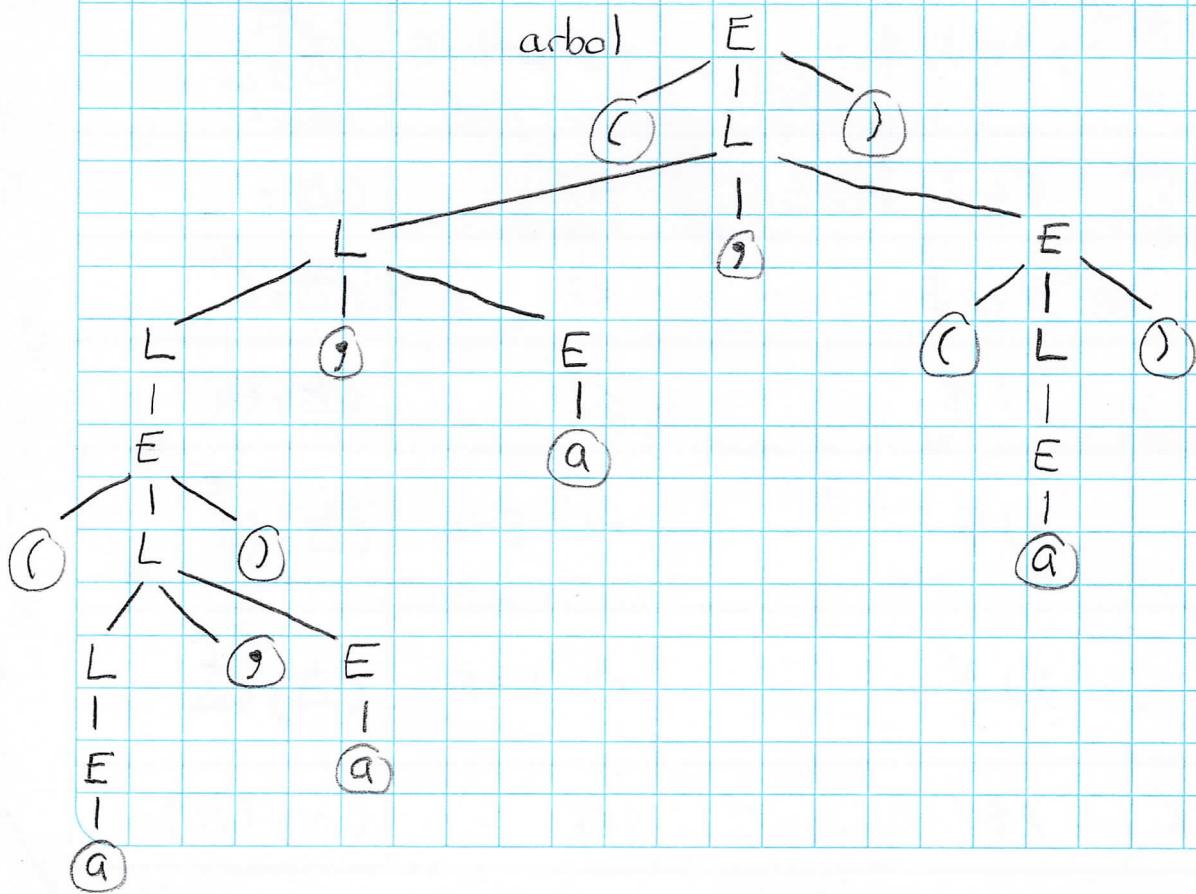
procesar cadena

((a, a), a, (a))

paso	pila	entrada	accion	simbolos
1	\$0	((a, a), a, (a))\$		
2	\$02	(a, a), a, (a))\$	S2	(
3	\$022	a, a), a, (a))\$	S2	((
4	\$0223	, a), a, (a))\$	S3	((a
5	\$0225	, a), a, (a))\$	r2 E → a	((a ^E
6	\$0224	, a), a, (a))\$	r4 L → E	((E ^L
7	\$02247	a), a, (a))\$	S7	((L,
8	\$022473), a, (a))\$	S3	((L, a
9	\$022478), a, (a))\$	r2 E → a	((L, a ^E
10	\$0224), a, (a))\$	r3 L → E	((L ^L , E
11	\$02246), a, (a))\$	S6	((L)

paso	pila	entrada	accion	símbolos
12	\$025	, a, (a)) \$	r1 E → (L)	((Δ)
13	\$024	, a, (a)) \$	r4 L → E	(Δ ^L)
14	\$0247	a, (a)) \$	s7	(Δ,
15	\$02473	, (a)) \$	s3	(Δ, a
16	\$02478	, (a)) \$	r2 E → a	(Δ, a ^E
17	\$024	, (a)) \$	r3 L → L, E	((Δ ^L , Δ ^E)
18	\$0247-	(a)) \$	s7	(Δ,
19	\$02472	a)) \$	s2	(Δ, (
20	\$024723)) \$	s3	(Δ, (a
21	\$024725)) \$	r2 E → a	(Δ, (a ^E
22	\$024724)) \$	r4 L → E	(Δ, (Δ ^L)
23	\$0247246) \$	s6	(Δ, (Δ)

paso	pila	entrada	accion	símbolos
24	\$02478) \$	r1 E → (L)	(Δ, (Δ))
25	\$024) \$	r3 L → L, E	(Δ, Δ)
26	\$0246	\$	s6	(Δ)
27	\$01	\$	r1 E → (L)	(Δ)
28	\$01	\$	accept	E





5. (**Hasta 1.5pts. extra**) Realiza una tabla comparativa entre los estilos de parsing **LL**, **LR0**, **SLR** y **LR1**.

Incluye características o descripciones breves de las coincidencias o diferencias entre ellos.

No olvides agregar las referencias, libros o páginas web consultadas.

Parser	Descripción	Similitudes	Ventajas	Desventajas
LL	Revisa la cadena de izquierda a derecha. Genera una derivación por izquierda, generando un árbol de arriba hacia abajo	No es tan similar a los otros 3, debido a que construye el árbol de arriba hacia abajo	Fácil de entender y usar sobre gramáticas no tan complejas	Limitado en que gramáticas puede manejar, como no soporta gramáticas con recursión izquierda
LR(0)	Revisa la cadena de izquierda a derecha. Genera una derivación por derecha, generando un árbol de abajo hacia arriba, tiene cero símbolos como lookahead	Es similar a LR(1), solo que sin lookahead, por lo tanto los estados son diferentes, tiene la misma forma para generar los estados que SLR, su diferencia es que en LR(0) los reduce se ponen en toda la fila del estado	Puede manejar gramáticas un poco más complejas que LL, como con recursión izquierda, no es tan complicado de entender	Como no usamos un lookahead pueden surgir varios shift/reduce conflicts en ciertas gramáticas
SLR	Revisa la cadena de izquierda a derecha. Genera una derivación por derecha, generando un árbol de abajo hacia arriba	Tiene la misma forma para generar los estados que LR(0), su diferencia es que en SLR los reduce se ponen en los símbolos del Follow de la parte derecha de la producción	Maneja mejor los conflictos que LR(0) al usar el Follow	Gramáticas más complejas o con algunas ambigüedades pueden causar varios problemas
LR(1)	Revisa la cadena de izquierda a derecha. Genera una derivación por derecha, generando un árbol de abajo hacia arriba, tiene un símbolo como lookahead	Es similar a LR(0), solo que usando un lookahead, por lo tanto los estados son diferentes y además se usará el lookahead para poner los reduce	Gracias a los lookahead puede manejar errores que producían los otros y por lo tanto manejar gramáticas más complicadas	Es más complicado de entender y puede ser algo laborioso

Fuentes:

Compilers: Principles, Techniques, and Tools (El libro del dragón)

Grupo de Google