CI3725 - Traductores e Interpretadores
Abril-Julio 2006
Prof. Ernesto Hernández-Novich

Laboratorio Semana IX

Gramáticas LR

Denominamos **asa** (*handle*) de una palabra a aquella subpalabra que corresponde con la parte derecha de alguna producción de manera que su reemplazo por la parte izquierda correspondiente representa un paso "en reversa" a través de una derivación más derecha, i.e. si usamos solamente derivaciones derechas tendremos

$$S \Rightarrow^* \delta A\omega \Rightarrow \delta\beta\omega$$

y diremos que β es un asa para $\gamma = \delta A\omega$. Llamaremos **prefijos viables** de una forma sentencial $\delta\beta\omega$ a los prefijos de δ. Si la gramática **no** es ambigua entonces existe exactamente un asa para cada forma sentencial.

Los **items** $LR(0)$ de una gramática libre de contexto $G = (N, \Sigma, P, S)$ son tripletas de la forma (A, α, β) tales que $A \to \alpha\beta \in P$ y se denotan $A \to \alpha \cdot \beta$. Para cada gramática puede construirse el conjunto de items $LR(0)$ aplicando el algoritmo de clausura de irems, definido como

$$clausura(I) =_{fix} I \cup \{B \to \cdot\gamma | A \to \alpha \cdot B\beta \in clausura(I) \wedge B \in N\}$$

Utilizando la clausura de los conjuntos de items $LR(0)$ puede construirse el autómata finito reconocedor de los prefijos viables de G definido como

$$M \quad = \quad (Q, N \cup \Sigma, \delta, I_0, Q - \{\emptyset\})$$

donde

$$\begin{aligned} Q &= items(G) \\ I_0 &= clausura(\{S \to \cdot\alpha | S \to \alpha \in P\}) \\ \delta(I, X) &= clausura(\{A \to \alpha X \cdot \beta | A \to \alpha \cdot X\beta \in I \wedge X \in N \cup \Sigma\}) \end{aligned}$$

Diremos que una gramática G es $LR(0)$ si:

1. Su símbolo inicial no aparece a la derecha en ninguna producción.
2. Todos los estados accesibles de su autómata de prefijos viables que tienen items terminales[1], tienen únicamente un elemento (**un** item).

Las configuraciones inválidas serían de la forma

$$\begin{aligned} I_i \quad : \quad & A \to \alpha \cdot \\ & B \to \beta \cdot \mathbf{a}\gamma \end{aligned}$$

que corresponde a un conflicto *shift-reduce*, o bien de la forma

$$\begin{aligned} I_i \quad : \quad & A \to \alpha \cdot \\ & B \to \beta \cdot \end{aligned}$$

que corresponde a un conflicto *reduce-reduce*.

[1] Un item terminal es de la forma $A \to \alpha\cdot$.

Ejemplo 1

Determinemos si la gramática $G = (\{S\}, \{(,), \mathbf{a}\}, P, S)$ es $LR(0)$ siendo P definida como

$$\begin{aligned} S &\rightarrow (S) \\ S &\rightarrow SS \\ S &\rightarrow \mathbf{a} \end{aligned}$$

Puesto que el símbolo inicial de la gramática aparece al lado derecho de la segunda producción, *"aumentamos"* la gramática con un nuevo símbolo inicial S' y calculamos los conjuntos de items.

La gramática aumentada queda como

$$\begin{aligned} S' &\rightarrow S \\ S &\rightarrow (S) \\ S &\rightarrow SS \\ S &\rightarrow \mathbf{a} \end{aligned}$$

y calculamos el conjunto de ítems I_0. Inicialmente se incluye en I_0 el item $S' \rightarrow \cdot S$ y luego calculamos su clausura, que de acuerdo con el algoritmo definido previamente implica agregar todos los items construidos a partir de las producciones para S con un punto a la izquierda, quedando

$$\begin{aligned} I_0 \quad : \quad & S' \rightarrow \cdot S \\ & S \rightarrow \cdot (S) \\ & S \rightarrow \cdot SS \\ & S \rightarrow \cdot \mathbf{a} \end{aligned}$$

Notamos que en I_0 todos los **no-terminales** que están a la **derecha** de un punto, ya están incluídos en la clausura, por lo tanto culmina el cálculo de $clausura(I_0)$. Para obtener los estados alcanzables del autómata de prefijos viables, basta con considerar para **todos** los items de I_0 el símbolo de la gramática a su derecha y "mover el punto"; eso genera un nuevo item al cual hemos de calcular su clausura. Consideremos el primer item de I_0 en el cual podemos "mover el punto" usando S entonces podemos construir el nuevo conjunto I_1 formado por todos los items de I_0 que pueden "mover el punto" usando S, que inicialmente nos queda

$$\begin{aligned} I_1 \quad : \quad & S' \rightarrow S \cdot \\ & S \rightarrow S \cdot S \end{aligned}$$

pero luego notamos que tenemos el no-terminal S a la derecha del punto, por lo que tenemos que calcular su clausura, quedando

$$\begin{aligned} I_1 \quad : \quad & S' \rightarrow S \cdot \\ & S \rightarrow S \cdot S \\ & S \rightarrow \cdot (S) \\ & S \rightarrow \cdot SS \\ & S \rightarrow \cdot \mathbf{a} \end{aligned}$$

Prosiguiendo de la misma manera nos quedará el conjunto de items $LR(0)$ definitivos como

$$\begin{aligned} I_0 \quad : \quad & S' \rightarrow \cdot S \\ & S \rightarrow \cdot (S) \\ & S \rightarrow \cdot SS \\ & S \rightarrow \cdot \mathbf{a} \\ I_1 \quad : \quad & S' \rightarrow S \cdot \end{aligned}$$

$$
\begin{array}{rcl}
 & & S \to S \cdot S \\
 & & S \to \cdot(S) \\
 & & S \to \cdot SS \\
 & & S \to \cdot \mathbf{a} \\
I_2 & : & S \to (\cdot S) \\
 & & S \to \cdot(S) \\
 & & S \to \cdot SS \\
 & & S \to \cdot \mathbf{a} \\
I_3 & : & S \to S \cdot S \\
 & & S \to \cdot(S) \\
 & & S \to \cdot SS \\
 & & S \to \cdot \mathbf{a} \\
I_4 & : & S \to \mathbf{a} \cdot \\
I_5 & : & S \to (S \cdot) \\
 & & S \to S \cdot S \\
 & & S \to \cdot(S) \\
 & & S \to \cdot SS \\
 & & S \to \cdot \mathbf{a} \\
I_6 & : & S \to SS \cdot \\
 & & S \to S \cdot S \\
 & & S \to \cdot(S) \\
 & & S \to \cdot SS \\
 & & S \to \cdot \mathbf{a} \\
I_7 & : & S \to (S)\cdot
\end{array}
$$

Nótese que el estado I_1 es accesible y contiene un item terminal ($S' \to S\cdot$) pero acompañado de otros items. Lo mismo puede decirse de I_6. Así que esta gramática **no** es $LR(0)$.

Ejemplo 2

Determinemos si la gramática

$$
\begin{array}{rcl}
S & \to & E \\
E & \to & E{+}T \\
E & \to & T \\
T & \to & T{*}F \\
T & \to & F \\
F & \to & (E) \\
F & \to & \mathbf{id}
\end{array}
$$

es $LR(0)$ y calculemos su autómata de prefijos viables. El conjunto de clausura de items nos queda

$$
\begin{array}{rcl}
I_0 & : & S \to \cdot E \\
 & & E \to \cdot E{+}T \\
 & & E \to \cdot T \\
 & & T \to \cdot T{*}F
\end{array}
$$

$$
\begin{array}{rcl}
 & & T \to \cdot F \\
 & & F \to \cdot(E) \\
 & & F \to \cdot \mathbf{id} \\
I_1 & : & S \to E \cdot \\
 & & E \to E \cdot + T \\
I_2 & : & E \to T \cdot \\
 & & T \to T \cdot * F \\
I_3 & : & T \to F \cdot \\
 & & \\
I_4 & : & F \to (\cdot E) \\
 & & E \to \cdot E + T \\
 & & E \to \cdot T \\
 & & T \to \cdot T * F \\
 & & T \to \cdot F \\
 & & F \to \cdot(E) \\
 & & F \to \cdot \mathbf{id} \\
I_5 & : & F \to \mathbf{id} \cdot \\
I_6 & : & E \to E + \cdot T \\
 & & T \to \cdot T * F \\
 & & T \to \cdot F \\
 & & F \to \cdot(E) \\
 & & F \to \cdot \mathbf{id} \\
 & & \\
I_7 & : & T \to T * \cdot F \\
 & & F \to \cdot(E) \\
 & & F \to \cdot \mathbf{id} \\
I_8 & : & F \to (E) \cdot \\
 & & E \to E \cdot + T \\
I_9 & : & E \to E + T \cdot \\
 & & T \to T \cdot * F \\
I_{10} & : & T \to T * F \cdot \\
I_{11} & : & F \to (E) \cdot
\end{array}
$$

y podemos construir el autómata de prefijos viables estableciendo las transiciones entre los I_i notando el símbolo de la gramática que lleva de uno a otro. Observando I_0 notaremos que hay una transición hasta I_1 con el símbolo E, una transición hasta I_2con el símbolo T, una transición hasta I_3 con el símbolo F, una transición hasta I_4con el símbolo (y una transición hasta I_5 con el símbolo **id**. Repitiendo el procedimiento para todos los conjuntos de items, nos quedaría el autómata

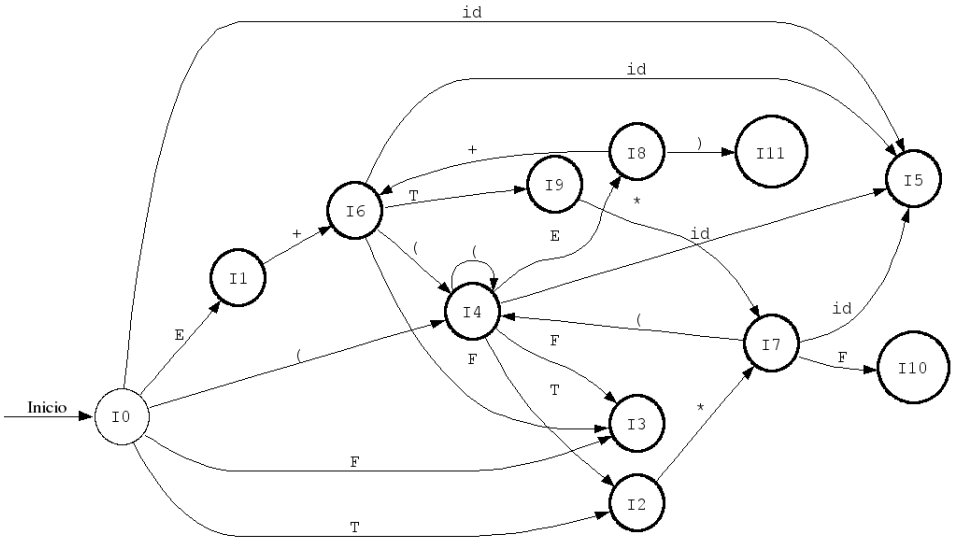

Figura 1: Autómata de Prefijos Viables para la Gramática del Ejemplo 2

En el conjunto de items podemos observar que I_1, I_2 e I_9 son inválidos (todos con conflicto *shift-reduce*) por tanto la gramática **no** es $LR(0)$.

Funcionamiento de un Parser LR por Tabla de Parsing

Un Parser LR es una máquina de estados que consta de:

- Una pila que contiene estados. El estado actual de la máquina parser LR es aquel que se encuentra en el tope de la pila.
- Un apuntador al siguiente elemento de la entrada. Al comenzar la ejecución de la máquina, apunta al primer caracter de la entrada.
- Un mecanismo de emisión para presentar por la salida los números asociados a las producciones de la gramática que han sido utilizadas. Al terminar la ejecución del parser, la salida tendrá (en orden inverso) las producciones necesarias para conseguir una derivación más derecha a partir del símbolo inicial de la gramática, de la cadena suministrada en la entrada.
- Una tabla de *acciones* y una tabla de *goto* que controla el funcionamiento de la máquina de estados. La tabla de *acciones* ($estado \times \Sigma$) contiene valores que pueden ser $shift\,\mathbf{n}$, $reduce\,\mathbf{n}$, *accept* o *error*. La tabla de *goto* ($estado \times N$) contiene valores que corresponden a estados de la máquina.

¿Cómo funciona? La ejecución de una máquina de parser LR es como sigue:

1. Al comenzar el procesamiento de una cadena de entrada, la pila contiene el estado inicial y se apunta al primer caracter de la entrada.
2. Se observa el estado en el tope de la pila (estado actual) y el símbolo terminal que están en la entrada. Se busca en la tabla de *acciones* esa combinación, y se opera según el valor contenido en la tabla, a saber:
   - Si la acción es $shift\,\mathbf{n}$, se consume el símbolo terminal de la entrada avanzando el apuntador al siguiente símbolo, y se coloca el estado **n** al tope de la pila.

- Si la acción es *reduce* **n**, se emite el número **n** en la salida. Se sacan de la pila tantos estados como número de símbolos (terminales o no-terminales) presentes en el lado derecho de la **n**-ésima producción de la gramática. Se observa el estado que quedó en el tope de la pila y el no-terminal que corresponde a la **n**-ésima producción de la gramática, se busca en la tabla de *goto* esa combinación, y se coloca el valor obtenido en el tope de la pila.
- Si la acción es *accept*, el parser termina su ejecución indicando la aceptación de la palabra contenida en la entrada. La secuencia de estados emitida por la salida hasta ese punto corresponde a las producciones utilizadas (en orden inverso) en una derivación más derecha.
- Si la acción es *error*, el parser termina su ejecución indicando la falla.

Construcción de la Tabla de Parsing LR para una Gramática

Comenzaremos por considerar la construcción de tablas de parsing LR para gramáticas $LR(0)$. El algoritmo de construcción consiste en:

1. Aumentar la gramática con un nuevo símbolo inicial S' tal que se agregue la producción $S' \rightarrow S\$$ donde S es el símbolo inicial original de la gramática y \$ es el terminal que representa el final de la entrada.
2. Enumerar contando desde **cero** cada producción de la gramática aumentada comenzando por la nueva producción inicial.
3. Calcular el autómata de prefijos viables a partir de los items $LR(0)$ de la gramática. Cada conjunto de items I_i servirá para construir el i-ésimo estado de la máquina parser LR.
4. Construir la tabla de *goto* usando una columna por cada símbolo **no-terminal** de la gramática original y una fila por cada estado. Si en el autómata de prefijos viables hay una transición desde I_i hasta I_j con el símbolo no-terminal **X** entonces, $goto[i, \mathbf{X}] \leftarrow j$.
5. Construir la tabla de *acciones* usando una columna por cada símbolo **terminal** de la gramática aumentada (incluyendo el \$) y una fila por cada estado.

   a) Si en el autómata de prefijos viables hay una transición desde I_i hasta I_j con el símbolo terminal **x**, entonces $acciones[i, \mathbf{x}] \leftarrow shift\,\mathbf{j}$.

   b) Si en el autómata de prefijos viables el i-ésimo conjunto contiene un item de la forma $A \rightarrow \alpha\cdot$ y $A \rightarrow \alpha$ es la n-ésima producción de la gramática ($n > 0$), entonces $\forall x \Rightarrow acciones[i, \mathbf{x}] \leftarrow reduce\,\mathbf{n}$, i.e. llenar la i-ésima fila de la tabla.

   c) Si en el autómata de prefijos viables el i-ésimo conjunto contiene un item de la forma $S' \rightarrow S\$\cdot$, entonces $acciones[i, \$] \leftarrow accept$.

   d) El resto de la tabla de *acciones* se llena con *error*.

Ejemplo 3

Construyamos la tabla de parsing LR(0) para la gramática

$$\begin{aligned} E &\rightarrow E{*}B \\ E &\rightarrow E{+}B \\ E &\rightarrow B \\ B &\rightarrow \mathbf{0} \\ B &\rightarrow \mathbf{1} \end{aligned}$$

Comenzamos por aumentar la gramática con un nuevo símbolo no-terminal inicial S y agregando un nuevo símbolo terminal \$ para representar el final de la entrada. Enseguida enumeramos las producciones según

describe el algoritmo, para tener

$$
\begin{array}{ll}
(0) & S \to E\$ \\
(1) & E \to E{*}B \\
(2) & E \to E{+}B \\
(3) & E \to B \\
(4) & B \to \mathbf{0} \\
(5) & B \to \mathbf{1}
\end{array}
$$

Construimos el autómata de prefijos viables calculando los conjuntos de items $LR(0)$ de la gramática, que nos quedan

$$
\begin{array}{lcl}
I_0 & : & S \to \cdot E\$ \\
 & & E \to \cdot E{*}B \\
 & & E \to \cdot E{+}B \\
 & & E \to \cdot B \\
 & & B \to \cdot \mathbf{0} \\
 & & B \to \cdot \mathbf{1} \\
I_1 & : & S \to E \cdot \$ \\
 & & E \to E \cdot {*}B \\
 & & E \to E \cdot {+}B \\
I_2 & : & E \to B \cdot \\
I_3 & : & B \to \mathbf{0} \cdot \\
I_4 & : & B \to \mathbf{1} \cdot \\
I_5 & : & E \to E{*} \cdot B \\
 & & B \to \cdot \mathbf{0} \\
 & & B \to \cdot \mathbf{1} \\
I_6 & : & E \to E{+} \cdot B \\
 & & B \to \cdot \mathbf{0} \\
 & & B \to \cdot \mathbf{1} \\
I_7 & : & E \to E{*}B \cdot \\
I_8 & : & E \to E{+}B \cdot \\
I_9 & : & S \to E\$\cdot
\end{array}
$$

siendo el autómata resultante de la forma

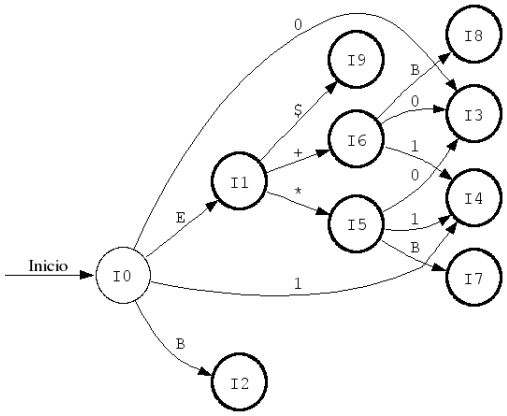

Figura 2: Autómata de Prefijos Viables para el Ejemplo 3

Notamos que la gramática es $LR(0)$ pues todos los conjuntos de items que contienen un elemento de la forma $A \to \alpha\cdot$ contienen únicamente ese elemento. Ahora podemos construir la tabla de parsing LR según el algoritmo. Observamos que:

- En el estado 0 notamos la transición $I_0 \to^{\mathbf{0}} I_3$ por lo tanto $acciones[0, \mathbf{0}] \leftarrow shift\,\mathbf{3}$[2]. Así mismo, observamos que hay una transición $I_0 \to^{\mathbf{1}} I_4$ por lo tanto $acciones[0, \mathbf{1}] \leftarrow shift\,\mathbf{4}$. Finalmente, las transiciones $I_0 \to^{E} I_1$ e $I_0 \to^{B} I_2$ dan lugar a $goto[0, E] \leftarrow 1$ y $goto[0, B] \leftarrow 2$ respectivamente.
- En el estado 1 notamos las transiciones $I_1 \to^{*} I_5$, $I_1 \to^{+} I_6$ e $I_1 \to^{\$} I_9$ que dan lugar a $acciones[1, *] \leftarrow shift\,\mathbf{5}$, $acciones[1, +] \leftarrow shift\,\mathbf{6}$ y $acciones[1, \$] \leftarrow shift\,\mathbf{9}$ respectivamente. No hay contribución a la tabla de *goto*.
- El estado 2 es un estado con un item final (tiene el punto a la derecha) y corresponde a la producción número 3 de la gramática, por lo tanto $\forall x \Rightarrow acciones[2, x] \leftarrow reduce\,\mathbf{3}$. No hay contribución a la tabla de *goto*.
- El estado 3 es un estado con un item final que corresponde a la producción número 4 de la gramática, por lo tanto $\forall x \Rightarrow acciones[3, x] \leftarrow reduce\,\mathbf{4}$. No hay contribución a la tabla de *goto*.
- El estado 4 es un estado con un item final que corresponde a la producción número 5 de la gramática, por lo tanto $\forall x \Rightarrow acciones[4, x] \leftarrow reduce\,\mathbf{5}$. No hay contribución a la tabla de *goto*.
- En el estado 5 notamos las transiciones $I_5 \to^{\mathbf{0}} I_3$ e $I_5 \to^{\mathbf{1}} I_4$ que dan lugar a $acciones[5, \mathbf{0}] \leftarrow shift\,\mathbf{3}$ y $acciones[5, \mathbf{1}] \leftarrow shift\,\mathbf{4}$ respectivamente. Finalmente, la transición $I_5 \to^{B} I_7$ da lugar a $goto[5, B] \leftarrow 7$.
- En el estado 6 notamos las transiciones $I_6 \to^{\mathbf{0}} I_3$ e $I_6 \to^{\mathbf{1}} I_4$ que dan lugar a $acciones[6, \mathbf{0}] \leftarrow shift\,\mathbf{3}$ y $acciones[6, \mathbf{1}] \leftarrow shift\,\mathbf{4}$ respectivamente. Finalmente, la transición $I_6 \to^{B} I_8$ da lugar a $goto[6, B] \leftarrow 8$.
- El estado 7 es un estado con un item final que corresponde a la producción número 1 de la gramática, por lo tanto $\forall x \Rightarrow acciones[7, x] \leftarrow reduce\,\mathbf{1}$. No hay contribución a la tabla de *goto*.
- El estado 8 es un estado con un item final que corresponde a la producción número 2 de la gramática, por lo tanto $\forall x \Rightarrow acciones[8, x] \leftarrow reduce\,\mathbf{2}$. No hay contribución a la tabla de *goto*.
- El estado 9 contiene el item $S \to E\$\cdot$ por lo tanto $acciones[9, \$] \leftarrow accept$.

Así, llegaremos a construir la tabla de parsing que queda como sigue (los espacios vacíos deben contener *error*).

[2] ¡No confundirse! El primer cero corresponde al **estado** y el segundo cero corresponde al símbolo **terminal** en la entrada.

	$*$	$+$	**0**	**1**	\$	E	B
0			$shift$ **3**	$shift$ **4**		1	2
1	$shift$ **5**	$shift$ **6**			$shift$ **9**		
2	$reduce$ **3**	$reduce$ **3**	$reduce$ **3**	$reduce$ **3**	$reduce$ **3**		
3	$reduce$ **4**	$reduce$ **4**	$reduce$ **4**	$reduce$ **4**	$reduce$ **4**		
4	$reduce$ **5**	$reduce$ **5**	$reduce$ **5**	$reduce$ **5**	$reduce$ **5**		
5			$shift$ **3**	$shift$ **4**			7
6			$shift$ **3**	$shift$ **4**			8
7	$reduce$ **1**	$reduce$ **1**	$reduce$ **1**	$reduce$ **1**	$reduce$ **1**		
8	$reduce$ **2**	$reduce$ **2**	$reduce$ **2**	$reduce$ **2**	$reduce$ **2**		
9					$accept$		

Cuadro 1: Tabla de Parsing LR para Ejemplo 3

Ilustraremos el funcionamiento del Parser LR utilizando la tabla para hacer el análisis de la cadena $\mathbf{0}+\mathbf{1}\$$ que pertenece al lenguaje. En la tabla, la pila crece hacia la **derecha** y cada grupo de filas representa un cambio en la máquina de estados (una fila para un $shift$ o $accept$, y dos filas para un $reduce$).

Estado	Entrada	Salida	Pila	Acción	Detalles
0	$\mathbf{0}+\mathbf{1}\$$		0	$shift$ **3**	Consumir y **push 3**.
3	$+\mathbf{1}\$$		0,3	$reduce$ **4**	Usar regla $B \to \mathbf{0}$
	$+\mathbf{1}\$$	4	0,2		Emitir "4", **pop** y $goto[0, B]$
2	$+\mathbf{1}\$$	4	0,2	$reduce$ **3**	Usar regla $E \to B$
	$+\mathbf{1}\$$	4,3	0,1		Emitir "3", **pop** y $goto[0, E]$
1	$+\mathbf{1}\$$	4,3	0,1	$shift$ **6**	Consumir y **push 6**.
6	$\mathbf{1}\$$	4,3	0,1,6	$shift$ **4**	Consumir y **push 4**.
4	\$	4,3	0,1,6,4	$reduce$ **5**	Usar regla $B \to \mathbf{1}$
	\$	4,3,5	0,1,6,8		Emitir "5", **pop** y $goto[6, B]$
8	\$	4,3,5	0,1,6,8	$reduce$ **2**	Usar regla $E \to E{+}B$
	\$	4,3,5,2	0,1		Emitir "2", tres **pop** y $goto[0, E]$
1	\$	4,3,5,2	0,1	$shift$ **9**	Consumir (queda igual) y **push 9**.
9	\$	4,3,5,2	0,1,9	$accept$	Fin.

Cuadro 2: Ejecución del Parser LR del Ejemplo 3

Nótese que lo emitido por la salida representa, en orden inverso, las producciones que deben utilizarse para obtener una derivación más derecha en la gramática **original** para la cadena suministrada en la entrada. Concretamente

$$\begin{aligned} E &\Rightarrow^2 E{+}B \\ &\Rightarrow^5 E{+}\mathbf{1} \\ &\Rightarrow^3 B{+}\mathbf{1} \\ &\Rightarrow^4 \mathbf{0}+\mathbf{1} \end{aligned}$$

Ejemplo 4

Construyamos la tabla de parsing LR(0) para la gramática

$$\begin{aligned} S &\to BB \\ B &\to \mathbf{a}B \\ B &\to \mathbf{c} \end{aligned}$$

Comenzamos por aumentar la gramática con un nuevo símbolo inicial T y agregando un nuevo símbolo terminal \$ para representar el fin de la entrada. En seguida enumeramos las producciones según indica el algoritmo para tener

$$
\begin{array}{ll}
(0) & T \to S\$ \\
(1) & S \to BB \\
(2) & B \to \mathbf{a}B \\
(3) & B \to \mathbf{c}
\end{array}
$$

Construimos el autómata de prefijos viables calculando los conjuntos de items $LR(0)$ para la gramática que nos quedan

$$
\begin{array}{lcl}
I_0 & : & T \to \cdot S\$ \\
 & & S \to \cdot BB \\
 & & B \to \cdot \mathbf{a}B \\
 & & B \to \cdot \mathbf{c} \\
I_1 & : & T \to S \cdot \$ \\
I_2 & : & S \to B \cdot B \\
 & & B \to \cdot \mathbf{a}B \\
 & & B \to \cdot \mathbf{c} \\
I_3 & : & S \to BB \cdot \\
I_4 & : & B \to \mathbf{a} \cdot B \\
 & & B \to \cdot \mathbf{a}B \\
 & & B \to \cdot \mathbf{c} \\
I_5 & : & B \to \mathbf{c} \cdot \\
I_6 & : & B \to \mathbf{a}B \cdot \\
I_7 & : & T \to S\$\cdot
\end{array}
$$

siendo el autómata resultante de la forma

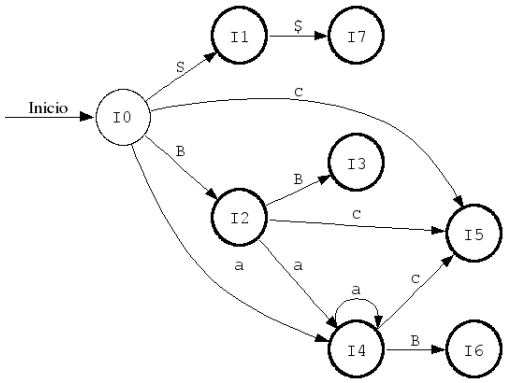

Figura 3: Autómata de Prefijos Viables para el Ejemplo 4

Notamos que la gramática es $LR(0)$ pues todos los conjuntos de items que contienen un elemento de la forma $A \to \alpha\cdot$ contienen únicamente ese elemento. Ahora podemos construir la tabla de parsing LR según el algoritmo. Nos queda

	a	**c**	$	S	B
0	$shift$ **4**	$shift$ **5**		1	2
1			$shift$ **7**		
2	$shift$ **4**	$shift$ **5**			3
3	$reduce$ **1**	$reduce$ **1**	$reduce$ **1**		
4	$shift$ **4**	$shift$ **5**			6
5	$reduce$ **3**	$reduce$ **3**	$reduce$ **3**		
6	$reduce$ **2**	$reduce$ **2**	$reduce$ **2**		
7			$accept$		

Cuadro 3: Tabla de Parsing LR para Ejemplo 4

Ejercicio 5

Construyamos la tabla de parsing LR(0) para la gramática

$$\begin{aligned} E &\rightarrow (L) \\ E &\rightarrow () \\ E &\rightarrow \mathbf{id} \\ L &\rightarrow L,E \\ L &\rightarrow E \end{aligned}$$

Comenzamos por aumentar la gramática con un nuevo símbolo inicial S y agregando un nuevo símbolo terminal \$ para representar el fin de la entrada. En seguida enumeramos las producciones según indica el algoritmo para tener

$$\begin{aligned} &(0) \quad S \rightarrow E\$ \\ &(1) \quad E \rightarrow (L) \\ &(2) \quad E \rightarrow () \\ &(3) \quad E \rightarrow \mathbf{id} \\ &(4) \quad L \rightarrow L,E \\ &(5) \quad L \rightarrow E \end{aligned}$$

Construimos el autómata de prefijos viables calculando los conjuntos de items $LR(0)$ para la gramática que nos quedan

$$\begin{aligned} I_0 \ : \ & S \rightarrow \cdot E\$ \\ & E \rightarrow \cdot(L) \\ & E \rightarrow \cdot() \\ & E \rightarrow \cdot\mathbf{id} \\ I_1 \ : \ & S \rightarrow E \cdot \$ \\ I_2 \ : \ & E \rightarrow (\cdot L) \\ & E \rightarrow (\cdot) \\ & L \rightarrow \cdot L,E \\ & L \rightarrow \cdot E \\ & E \rightarrow \cdot(L) \\ & E \rightarrow \cdot() \\ & E \rightarrow \cdot\mathbf{id} \end{aligned}$$

$$
\begin{array}{rcl}
I_3 & : & E \to \mathbf{id}\,\cdot \\
I_4 & : & E \to (L\,\cdot\,) \\
 & & L \to L\,\cdot\,,E \\
I_5 & : & E \to ()\,\cdot \\
I_6 & : & L \to E\,\cdot \\
I_7 & : & E \to (L)\,\cdot \\
I_8 & : & L \to L,\cdot\, E \\
 & & E \to \cdot(L) \\
 & & E \to \cdot() \\
 & & E \to \cdot\mathbf{id} \\
I_9 & : & E \to L,E\,\cdot \\
I_{10} & : & S \to E\$\cdot
\end{array}
$$

siendo el autómata resultante de la forma

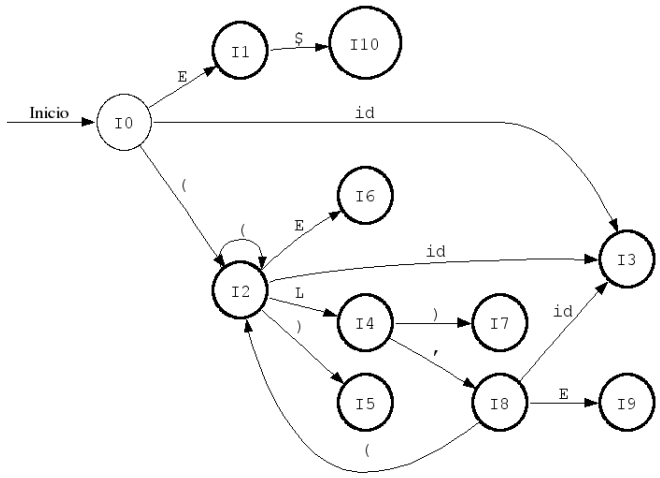

Figura 4: Autómata de Prefijos Viables para el Ejemplo 5

Notamos que la gramática es $LR(0)$ pues todos los conjuntos de items que contienen un elemento de la forma $A \to \alpha\cdot$ contienen únicamente ese elemento. Ahora podemos construir la tabla de parsing LR según el algoritmo. Nos queda

	(	)	,	**id**	$	E	L
0	$shift$ **2**			$shift$ **3**		1	
1					$shift$ **10**		
2	$shift$ **2**	$shift$ **5**		$shift$ **3**		6	4
3	$reduce$ **3**	$reduce$ **3**	$reduce$ **3**	$reduce$ **3**	$reduce$ **3**		
4		$shift$ **7**	$shift$ **8**				
5	$reduce$ **2**	$reduce$ **2**	$reduce$ **2**	$reduce$ **2**	$reduce$ **2**		
6	$reduce$ **5**	$reduce$ **5**	$reduce$ **5**	$reduce$ **5**	$reduce$ **5**		
7	$reduce$ **1**	$reduce$ **1**	$reduce$ **1**	$reduce$ **1**	$reduce$ **1**		
8	$shift$ **2**			$shift$ **3**		9	
9	$reduce$ **4**	$reduce$ **4**	$reduce$ **4**	$reduce$ **4**	$reduce$ **4**		
10					$accept$		

Cuadro 4: Tabla de Parsing LR para Ejemplo 5