

Introducción

El *Autómata con Pila* (AP), en inglés *Push Down Automata*, es básicamente un autómata finito al que se le ha incorporado una memoria de pila. La memoria de pila o memoria LIFO (*Last In First Out*) incrementa la capacidad de resolver problemas del autómata finito convencional, al incorporarle la posibilidad de memorizar total o parcialmente la cadena leída y cualquier otra marca que ayude al procesamiento de la misma. Así, el AP dispone de registros de memoria que puede usar ventajosamente, aun a pesar de las limitaciones propias de las memorias LIFO. Como ya fue anticipado en el Capítulo 1, los autómatas con pila son capaces de reconocer lenguajes generados por gramáticas menos restringidas que las regulares: las gramáticas independientes de contexto o tipo 2. El hecho de que todos los lenguajes de computación respondan a este tipo de gramáticas hace que los AP sean muy estudiados y estén ampliamente difundidos como elementos centrales en los compiladores y en otras numerosas aplicaciones.

Una variante muy interesante del autómata con pila convencional es el *Autómata con Pila Bidireccional* (APB), que surge de incorporar memoria de pila en un AFDB como los estudiados en el Capítulo 3. A diferencia de lo que ocurre con éstos, los APB aumentan la capacidad de reconocimiento de los AP y pueden operar algunos lenguajes provenientes de gramáticas tipo 1. Además, la bidireccionalidad permite en muchos casos simplificar notablemente los diseños. Otro atractivo de los APB es la existencia de algoritmos muy eficientes para su simulación en computadoras, como es el de Cook.¹

Autómatas con pila deterministas y no deterministas

Agregarle una memoria de pila a un autómata finito convencional implica incorporarle dos nuevos componentes: el alfabeto de símbolos de la pila y, entre ellos, un símbolo especial destinado a servir de referencia, llamado marca de fondo de pila. Este último es necesario debido a la naturaleza de los accesos LIFO y con el fin de permitir comprobar si la pila ya está descargada.

Definición del AP

Tal cual lo anticipado, el autómata con pila es un autómata finito al que le fue incorporado una memoria LIFO. Por tal motivo, mantiene su alfabeto de entrada, conjunto de estados, estado inicial, estados de aceptación y función de transición. Luego a la definición se incorporan el alfabeto de pila y la marca de referencia, por lo que el AP es una séptupla. Se define:

$$AP = (\Sigma_E, \Gamma, Q, q_0, \#, A, f)$$

donde:

- Σ_E : Alfabeto de símbolos de entrada
- Γ : Alfabeto de símbolos de la pila
- Q : Conjunto finito y no vacío, de estados posibles
- q_0 : Estado inicial de operación, $q_0 \in Q$
- A : Conjunto de estados de aceptación, $A \subseteq Q$
- $\#$: Símbolo de referencia de pila, $\# \in \Gamma$ y $\# \notin \Sigma_E$
- f : Función de transición

Es indudable que, al incorporar una memoria de pila, es necesario prever su reconocimiento y operación por parte del autómata y esto debe ser contemplado en la función de transición. En el caso de un AP no determinista (APND), se trata de una relación, que toma la forma:

$$f: Q \times (\Sigma_E \cup \{\lambda\}) \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma^*)$$

A partir de la observación de esta formulación, se deduce que:

¹ "Lenguajes, gramáticas y autómatas", Rafael Cases Muñoz (Ref. Co071, Pg. 219).

Unidad 5: Autómatas con Pila

- a) El comportamiento del autómata en un intervalo de tiempo dado queda determinado por tres argumentos, que son: *i)* el estado actual, *ii)* el símbolo de entrada y *iii)* el símbolo leído de tope de la pila.
- b) A partir de estos tres argumentos, queda definido: *i)* el próximo estado del autómata y *ii)* la acción sobre la pila. Esta acción consiste en la grabación de varios símbolos, la de uno solo o de ninguno (λ), y esto último corresponde al caso en que la pila es descargada, es decir, leída y no grabada. Además, el autómata moverá el cabezal sobre la cinta de entrada a la próxima posición de lectura.
- c) Las condiciones que hacen no determinista al AP definido son: *i)* que en un cierto intervalo de tiempo el APND puede operar sin leer la cadena de entrada (transición λ), *ii)* que en algunos casos haya varias opciones de próximo estado y carga de pila, las que quedan agrupadas en un elemento del conjunto potencia $P(Q \times \Gamma^*)$ y *iii)* una combinación de ambas condiciones.

En el caso del AP determinista (APD), la literatura presenta dos opciones para la definición de la función de transición. La primera surge al eliminar la transición λ y admitir una sola condición de próximo estado y acción sobre la pila. La función queda definida entonces como:

$$f: Q \times \Sigma_E \times \Gamma \rightarrow Q \times \Gamma^*$$

La segunda definición posible de la función de transición del APD es la siguiente:

$$f: Q \times (\Sigma_E \cup \{\lambda\}) \times \Gamma \rightarrow Q \times \Gamma^*$$

con la condición que si $f(q, \lambda, b)$ está definida para $q \in Q$ y $b \in \Gamma$, entonces necesariamente no debe estarlo $f(q, a, b)$ para todo $a \in \Sigma_E$.

Representación

Como pudo observarse, la función de transición de los APD y APND tiene tres argumentos y esto imposibilita su representación mediante una tabla. Para casos en los que la cantidad de estados y la cantidad de símbolos de pila es poco numerosa, puede recurrirse a una tabla en la que cada fila corresponda a un elemento de $Q \times \Gamma$ y cada columna a un elemento del alfabeto de entrada Σ_E o λ . Además, para todos los casos sigue siendo válida la representación mediante grafos y ésta es la opción más utilizada. En estos casos, los arcos dirigidos son rotulados con una etiqueta $a, b / \alpha$, donde $a \in \Sigma_E$ representa el símbolo de entrada leído, $b \in \Gamma$ el símbolo leído del tope de pila y $\alpha \in \Gamma^*$ representa la cadena grabada sobre la pila, de tal forma que el último símbolo de misma quede en la cima de la pila. Las dos formas de representación, tabla y grafo serán utilizadas en los ejemplos que se presentan en este capítulo.

Concepto de configuración o descripción instantánea

Al igual que en los AF, es necesario definir la condición en la que se encuentra un autómata con pila en un intervalo de tiempo t dado y, para ello, se recurre a la descripción instantánea o configuración:

$$K_t = (q, \beta, \delta)$$

donde q representa el estado en que se encuentra el AP, β la subcadena de entrada pendiente de ser leída y δ el contenido de la pila. A partir de esta definición y ante una cadena de entrada α , tal que $\alpha \in \Sigma_E^*$, se puede reconocer la *configuración inicial* como:

$$K_0 = (q_0, \alpha, \#)$$

De igual forma, se define la *configuración final* como:

$$K_n = (q, \lambda, \delta)$$

donde la cadena debe haber sido completamente leída. En este caso, las condiciones a ser cumplidas por el estado q y el contenido de la pila δ son diversas y se especifican en el siguiente apartado.

Unidad 5: Autómatas con Pila

El tránsito de una configuración a otra es también aquí denominado *movimiento*, es decir que si existe la transición $f(p, a, b) = (q, bb)$, el movimiento del estado p al q , leyendo el símbolo de entrada a , extrayendo de la cima de la pila el símbolo de pila b y almacenando en la pila la cadena bb , queda representado como:

$$(p, a\beta, \delta b) \vdash (q, \beta, \delta bb)$$

y el movimiento desde la configuración inicial a la final es representado:

$$(q_0, \alpha, \#) \vdash^* (q, \lambda, \delta)$$

donde \vdash^* representa una cantidad finita de movimientos.

Por último, y tal como ocurrió con los AFD y APND, el comportamiento de los autómatas con pila ante una cierta cadena de entrada queda convenientemente representado por los árboles de configuraciones o descripciones instantáneas.

Aceptación de lenguajes

Se admite que hay varias formas de aceptación de lenguajes por parte de los autómatas con pila. En todos los casos, las sentencias de los lenguajes deben conducir al autómata con pila desde su configuración inicial hasta su configuración final, y la diferencia está en las condiciones que definen la configuración final de aceptación, que son las siguientes:

a) Aceptación por vaciado de pila

$$L = \{\alpha / (q_0, \alpha, \#) \vdash^* (q, \lambda, \#) \text{ con } q_0, q \in Q, \alpha \in \Sigma_E^*, \# \in \Gamma\}$$

b) Aceptación por estado de aceptación

$$L = \{\alpha / (q_0, \alpha, \#) \vdash^* (q, \lambda, \delta) \text{ con } q_0 \in Q, q \in A, \alpha \in \Sigma_E^*, \delta \in \Gamma^+, \# \in \Gamma\}$$

c) Aceptación por vaciado de pila y estado de aceptación

$$L = \{\alpha / (q_0, \alpha, \#) \vdash^* (q, \lambda, \#) \text{ con } q_0, q \in Q, q \in A, \alpha \in \Sigma_E^*, \# \in \Gamma\}$$

Nótese que en todos los casos la sentencia ha sido completamente leída y la exigencia o no de pila vacía distinguirá dos tipos característicos de problemas que serán estudiados a través de los ejemplos presentados en lo que sigue.

No equivalencia de los APD y APND

Los Autómatas con Pila No Deterministas (APND) no tienen necesariamente Autómatas con Pila Deterministas (APD) equivalentes que acepten los mismos lenguajes. Es muy importante notar que esta falta de equivalencia no significa que no pueda existir un autómata con pila determinista que acepte una cadena específica del lenguaje, pero se trataría de una equivalencia muy restringida que carecería de generalidad. En los Ejemplos 5.1 y 5.2, se ilustra este problema.

AP deterministas y transiciones λ

Al estudiarse los autómatas finitos en el Capítulo 3, se dijo que, por tratarse de máquinas abstractas, se prescindía de la técnica de implementación del medio de entrada y se utilizaba el concepto genérico de una *cinta de entrada* y su correspondiente cabezal. Es decir que se admite que la máquina es capaz de reconocer el momento en el que ha completado la lectura de una cadena, sin necesidad de precisar el modo en el que lo hace.

El autómata con pila es una evolución del autómata finito y el razonamiento anterior, en cuanto al reconocimiento de que una cadena de entrada ha sido completamente leída, es igualmente válido. Sin embargo, el autómata con pila utiliza también en su operación un tercer argumento, que es el símbolo presente en el tope de la pila (memoria LIFO), y dos de las condiciones de aceptación de sentencias requieren que la pila se encuentre tal como estaba al comenzar la máquina a operar (ver Aceptación de lenguajes). Esta situación se identifica como de *pila vacía* y, para ello, el AP incluye en su definición un símbolo de su alfabeto de pila que es explícitamente declarado *marca del fondo de la pila* ($\#$) y que permite identificar esta condición.

Por este motivo, una vez que el AP ha completado la lectura de una sentencia, debe verificarse que la pila se encuentra vacía y la única forma es leerla y comprobar que el símbolo

Unidad 5: Autómatas con Pila

leído es precisamente la *marca de fondo de pila* $\#$. Por esta razón, en los ejemplos presentados en este capítulo, cuando los autómatas han completado la lectura de la cadena de entrada, realizan un movimiento adicional que está destinado a verificar la condición de la pila y, en caso de estar vacía, transitar al estado de aceptación. Aquí, la cadena de entrada ya ha sido leída y esta lectura adicional de la pila se representa como una transición λ (sin leer en su entrada).

Es importante advertir que, pese a la transición λ , esto por sí solo no define al AP como no determinista (ver Aceptación de lenguajes punto c). En efecto, si se completó la lectura de la sentencia de entrada con el autómata en un cierto estado q , el AP será determinista siempre que:

$$|f(q, \lambda, \#)| = 1 \text{ y } f(q, a, \#) = \emptyset \text{ para todo símbolo } a \in \Sigma_E$$

y será considerado no determinista cuando:

$$f(q, \lambda, \#) \neq \emptyset \text{ y } f(q, a, \#) \neq \emptyset \text{ para algún símbolo } a \in \Sigma_E.$$

Se sugiere comprobar que los AP de los próximos ejemplos 5.2, 5.5 y 5.6 son no deterministas, mientras que los AP de los ejemplos 5.1, 5.3 y 5.4 son deterministas, en este caso, a pesar de la transición λ .

Ejemplo 5.1

Se presenta como primer ejemplo el caso del reconocimiento de un palíndromo de largo impar, donde el carácter central es un separador que no está presente en el prefijo que lo antecede. Es decir que la cadena a ser aceptada toma la forma general $\beta = \mathbf{ZcZ}^{-1}$, donde $\mathbf{Z} \in \{\mathbf{a}, \mathbf{b}\}^*$. Se pide: a) proponer el grafo de un autómata que acepte este tipo de cadenas, b) representarlo formalmente y c) comprobar el comportamiento del autómata ante la cadena de entrada $\alpha = \mathbf{abbcbbba}$.

a) Grafo del autómata con pila

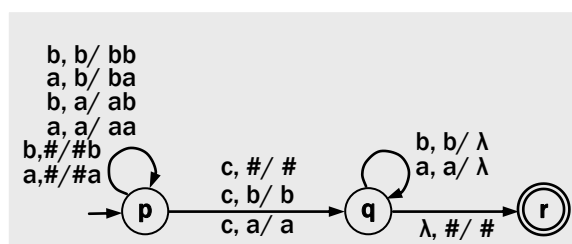


Figura 5.1: Grafo del AP del Ejemplo 5.1.

Se aprovecha este primer ejemplo para reiterar la convención sobre los rótulos que identifican los arcos del grafo: $\mathbf{a, \#/ \#a}$ significa que el símbolo de entrada es \mathbf{a} , el símbolo leído del tope de pila es $\mathbf{\#}$ y se graba sobre la pila la cadena $\mathbf{\#a}$. Esto implica reponer la $\mathbf{\#}$ e incorporar la \mathbf{a} leída de la entrada. Nótese que también pudo solo haberse repuesto el símbolo leído, en este caso $\mathbf{\#}$ o no haber grabado nada (λ). También debe notarse que en la convención adoptada la marca de la pila es el primer carácter de la izquierda y crece hacia la derecha.

b) La definición formal es: $AP = (\Sigma_E, \Gamma, Q, q_0, \#, A, f)$ donde:

$$\Sigma_E = \{ a, b, c \}$$

$$\Gamma = \{ \#, a, b \}$$

$$Q = \{ p, q, r \}$$

$$q_0 = p$$

$$\# = \text{pila vacía.}$$

$$A = \{ r \}$$

Unidad 5: Autómatas con Pila

$f:$	a	b	c	λ
p, #	p, #a	p, #b	q, #	
p, a	p, aa	p, ab	q, a	
p, b	p, ba	p, bb	q, b	
q, #				r, #
q, a	q, λ			
q, b		q, λ		

Tabla 5.1: Función f del AP del Ejemplo 5.1

Nótese que no se han incluido condiciones de no aceptación de sentencias que no pertenecen al lenguaje, por lo cual la función f es parcial, asumiéndose que cualquier situación no prevista hará que el autómata no pueda seguir operando (*cancela*) y rechace en consecuencia la cadena de entrada.

c) Representación de la aceptación de $\alpha = \text{abbcbbba}$

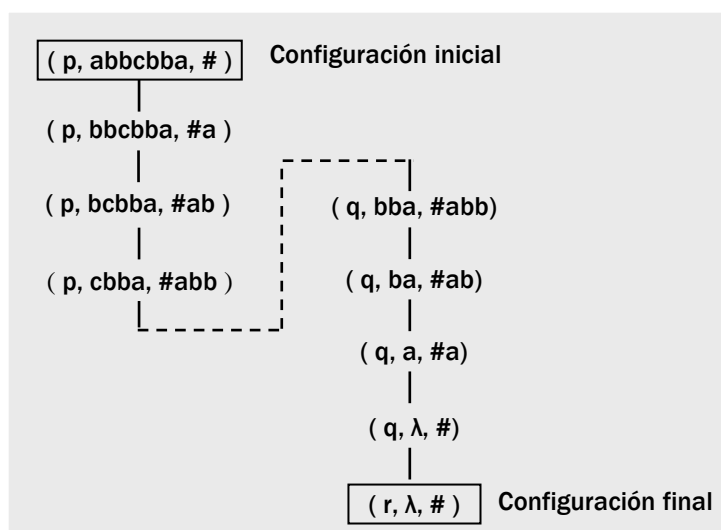


Figura 5.2: Árbol de descripciones instantáneas de $\alpha = \text{abbcbbba}$.

Obsérvese que, mientras el AP se encuentra en el estado p , la pila está destinada a almacenar una imagen invertida de la cadena de entrada. El AP se mantiene en esta condición hasta que lee el símbolo c , con el cual transita al estado q , y a partir de allí verifica que la secuencia de los siguientes símbolos de entrada sean los mismos encontrados en la pila. En estos casos, la pila funciona como un almacén de la cadena de entrada y para la operación de este APD es indispensable el separador c . Si a través de esta verificación se completa la lectura de la cadena y la misma responde a la forma general \mathbf{ZcZ}^{-1} , el AP tendrá la pila vacía ($\#$ en el tope de pila) y la entrada es aceptada. Si esto no ocurre el autómata cancelará su funcionamiento, que es la condición de error.

La necesidad del separador c implica una severa limitación ya que en muchos lenguajes esta condición puede no ser posible o no ser conveniente. Cabe preguntarse qué ocurre si los palíndromos no disponen de ese separador, es decir que tienen la forma general \mathbf{ZZ}^{-1} , y este interrogante da lugar al siguiente ejemplo.

Ejemplo 5.2

Se presenta aquí el caso del reconocimiento de un palíndromo de largo par, donde no hay carácter central, lo que implica que la cadena a ser aceptada toma la forma general $\beta = \mathbf{ZZ}^{-1}$, donde $\mathbf{Z} \in \{a, b\}^*$. Al igual que en el ejemplo anterior se pide: a) proponer el grafo de un autómata que acepte este tipo de cadenas, b) representarlo formalmente y c) comprobar el comportamiento del autómata ante la cadena de entrada $\alpha = \text{abbbba}$.

a) Grafo del autómata con pila

Unidad 5: Autómatas con Pila

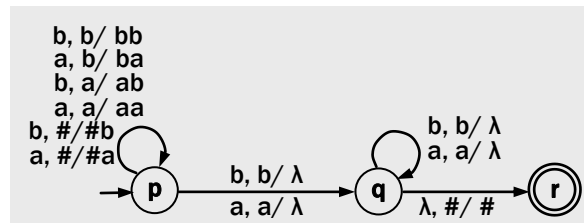


Figura 5.3: Grafo del AP del Ejemplo 5.2.

b) La definición formal es $AP = (\Sigma_E, \Gamma, Q, q_0, \#, A, f)$ donde:

$\Sigma_E = \{ a, b \}$

$\Gamma = \{ \#, a, b \}$

$Q = \{ p, q, r \}$

$q_0 = p$

$\#$ = pila vacía.

$A = \{ r \}$

$f:$	a	b	λ
p, #	p, #a	p, #b	
p, a	p, aa q, λ	p, ab	
p, b	p, ba	p, bb q, λ	
q, #			r, #
q, a	q, λ		
q, b		q, λ	

Tabla 5.2: Función f del AP del Ejemplo 5.2

En este caso, ante cada lectura la ausencia de un separador obliga al autómata tanto a cargar la pila como a verificarla contra el siguiente símbolo de entrada. Esto es debido a la imposibilidad que tiene el autómata de reconocer en cada momento si ya llegó a la mitad del palíndromo, lo que lo convierte en un APND. Este es uno de los casos en que no hay un APD equivalente que cumpla la misma función.

c) Representación de la aceptación de $\alpha = \text{abbbba}$

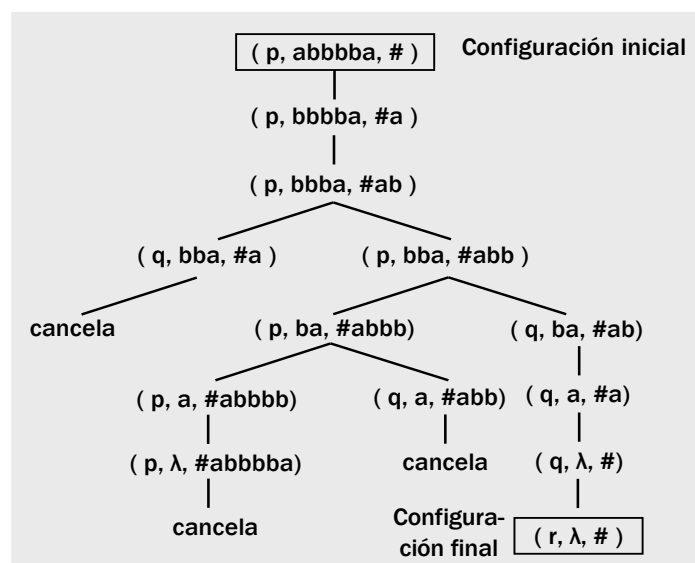


Figura 5.4: Árbol de descripciones instantáneas de $\alpha = \text{abbbba}$.

El árbol de descripciones instantáneas muestra lo anticipado, en el sentido que en el estado p , y teniendo el símbolo b tanto en la entrada como en la cima de la pila, el AP debe explorar dos

Unidad 5: Autómatas con Pila

opciones: cargar la entrada en la pila manteniéndose en el estado **p** y contrastar la entrada con el contenido de la pila, desapilar y pasar a **q**.

Ejemplo 5.3

En los ejemplos anteriores, la pila fue utilizada para poder comparar entre sí dos partes de la cadena de entrada, un prefijo contra un sufijo. Una variante se presenta cuando la pila es utilizada para comprobar que las cantidades totales de ciertos símbolos sean las mismas y esto es lo propuesto en este ejemplo. Se busca verificar que la cadena responda a la forma general $\beta = a^m b^n c^j d^k$, donde $m+n=j+k$ y $|\beta| > 0$. Similarmente a los casos anteriores se pide: a) Proponer el grafo de un autómata que acepte este tipo de cadenas y b) representarlo formalmente.

En este caso, se utiliza un único símbolo auxiliar que es almacenado en la pila por cada **a** o **b** que se lee de la entrada (los cuenta) y se descarga por cada **c** o **d** (los descuenta), esperándose completar la lectura de la entrada con la pila vacía.

a) Grafo del autómata con pila:

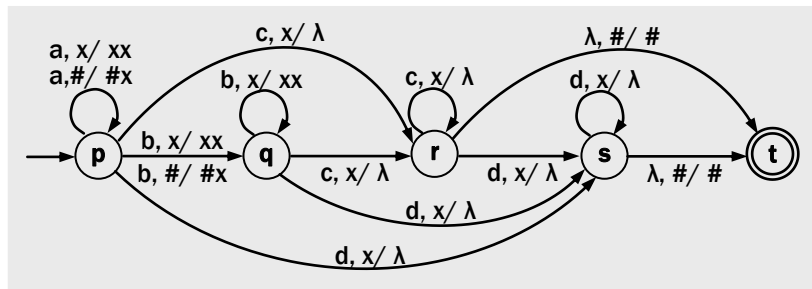


Figura 5.5: Grafo del AP del Ejemplo 5.3 (no se incluye estado de error).

b) La definición formal es $AP = (\Sigma_E, \Gamma, Q, q_0, \#, A, f)$, donde los siete componentes son:

$$\Sigma_E = \{ a, b, c, d \}$$

$$\Gamma = \{ x, \# \}$$

$$Q = \{ p, q, r, s, t \}$$

$$q_0 = p$$

$$\# = \text{pila vacía.}$$

$$A = \{ t \}$$

f:	a	b	c	d	λ
p, #	p, #x	q, #x			
p, x	p, xx	q, xx	r, λ	s, λ	
q, x		q, xx	r, λ	s, λ	
r, x			r, λ	s, λ	
r, #					t, #
s, x				s, λ	
s, #					t, #

Tabla 5.3: Función **f** del AP del Ejemplo 5.3.

Como puede apreciarse, es necesario asegurar que los símbolos estén ordenados y que las cantidades de cada uno cumplan la relación $m+n = j+k$. Para lo primero, se utiliza la secuencia de estados p, q, r, s y lo segundo está a cargo de la pila.

Es necesario destacar que en esta solución tampoco se han representado las posibles condiciones de error, cuya identificación y justificación ha quedado reservada al lector como ejercicio.

Unidad 5: Autómatas con Pila

Ejemplo 5.4

Una alternativa a la solución anterior es utilizar la pila para asegurar la correcta secuencia de los símbolos de entrada **a** y **b**. Esto no ofrece una ventaja significativa, ya que en el AP solo se ahorra un estado, pero muestra una variante interesante en la utilización de la memoria LIFO. La idea que aquí se propone es validar los símbolos leídos según el carácter que se encuentra en el tope de la pila. Como se comprueba fácilmente, esto no es aplicable a la secuencia de los símbolos **c** y **d**, que debe estar asegurada a través de diferentes estados del autómata.

Al igual que en el ejemplo anterior, se requiere: a) Proponer el grafo de un autómata que acepte este tipo de cadenas ($\beta = a^m b^n c^j d^k$, donde $m+n = j+k$ y $|\beta| > 0$) y b) representarlo formalmente.

a) Grafo del autómata con pila:

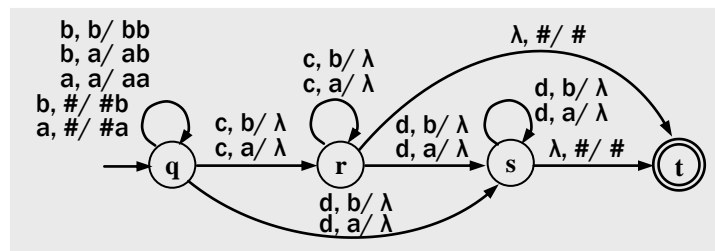


Figura 5.6: Grafo del AP del Ejemplo 5.4.

b) Su definición formal es $AP = (\Sigma_E, \Gamma, Q, q_0, \#, A, f)$, donde los siete componentes son:

$$\Sigma_E = \{ a, b, c, d \}$$

$$\Gamma = \{ \#, a, b \}$$

$$Q = \{ q, r, s, t \}$$

$$q_0 = q$$

$$\# = \text{pila vacía.}$$

$$A = \{ t \}$$

f:	a	b	c	d	λ
q, #	q, #a	q, #b			
q, a	q, aa	q, ab	r, λ	s, λ	
q, b		q, bb	r, λ	s, λ	
r, a			r, λ	s, λ	
r, b			r, λ	s, λ	
r, #					t, #
s, a				s, λ	
s, b				s, λ	
s, #					t, #

Tabla 5.4 Función f del AP del Ejemplo 5.4.

También aquí se ha dejado al lector la identificación y justificación de las posibles condiciones de error, similares a las del ejemplo anterior.

Ejemplo 5.5

En los dos últimos ejemplos, se buscó confirmar que la cantidad de ciertos símbolos en un prefijo y sufijo de la cadena de entrada era la misma. En este ejemplo, se presenta una variante, que consiste en proponer un AP que acepte cadenas tales como $\beta = a^m b^n c^j d^k$, donde $m+j=n+k$ y $|\beta| > 0$. Es decir, que las cantidades a verificar corresponden a símbolos alternados y no agrupados en el prefijo-sufijo. Se pide: a) representar el grafo de un autómata que acepte este tipo de cadenas, b) definirlo formalmente y c) representar la aceptación de la cadena $\alpha = abbbcccd$.

a) Grafo del autómata con pila:

Unidad 5: Autómatas con Pila

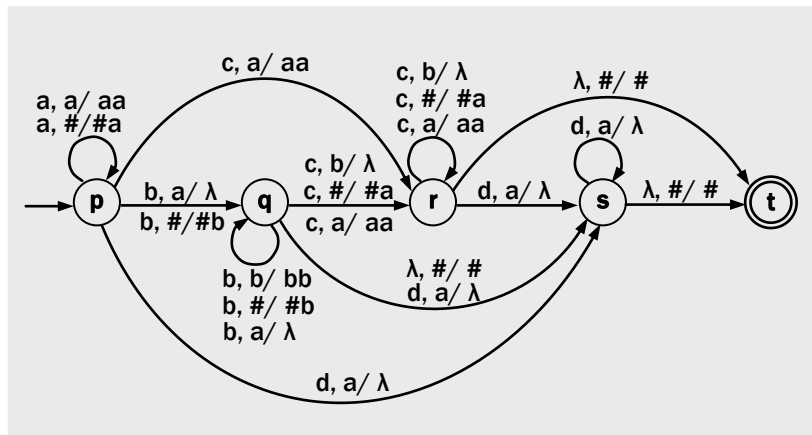


Figura 5.7: Grafo del AP del Ejemplo 5.5 (no se incluye estado de error).

b) Definición formal del AP = $(\Sigma_E, \Gamma, Q, q_0, \#, A, f)$, donde los siete componentes son:

f :	a	b	c	d	λ
p, #	p, #a	q, #b			
p, a	p, aa	q, λ	r, aa	s, λ	
q, #		q, #b	r, #a		s, #
q, a		q, λ	r, aa	q, λ	
q, b		q, bb	r, λ		
r, #			r, #a		t, #
r, a			r, aa	s, λ	
r, b			r, λ		
s, #					t, #
s, a				s, λ	

Tabla 5.5: Función f del AP del Ejemplo 5.5.

c) Representación de la aceptación de $\alpha = \text{abbbcccd}$

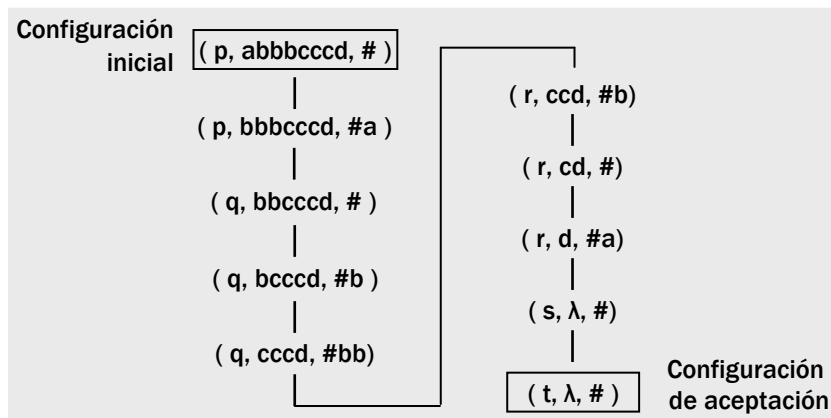


Figura 5.8: Árbol de descripciones instantáneas de $\alpha = \text{abbbcccd}$.

Aquí se han utilizado dos símbolos excluyentes que se acumulan con diferente significado en la pila a medida que es leída la cadena de entrada. El carácter **a** representa el exceso de entradas **a** o **c** sobre las entradas **b** o **d**, mientras que se utiliza el carácter **b** para indicar lo contrario. Obviamente que al completarse la lectura de una cadena correcta la pila debe haber quedado vacía.

Con esta solución propuesta, recae sobre la pila la responsabilidad de asegurar el requerido equilibrio en la cantidad de caracteres de diferente tipo y el orden en la secuencia de entrada es controlado a través de cambios de estado. También aquí se ha dejado al lector la tarea de identificar las posibles condiciones de error.

Una alternativa al árbol de descripciones instantáneas en la representación del proceso de aceptación de cierta cadena por parte de un AP es la llamada *tabla operativa*. Esta tabla tiene una

Unidad 5: Autómatas con Pila

fila por cada intervalo de tiempo y muestra en sus columnas el estado, la cadena a ser leída y el contenido de la pila, es decir, los componentes de la configuración o descripción instantánea. Ofrece la ventaja de ser una representación más compacta que el árbol de configuraciones, pero tiene una gran limitación al no facilitar la representación de caminos alternativos. Por esta razón, la tabla operativa es poco recomendable para mostrar el comportamiento de autómatas no deterministas.

En la Tabla 5.6 se presenta la secuencia operativa del proceso de aceptación de la cadena mostrado en el árbol de configuraciones de la Figura 5.8. Estas representaciones son denominadas “tablas operativas”.

N°	Estado	Cadena a leer	Contenido de pila
1	p	abbbcccd	#
2	p	bbbcccd	#a
3	q	bbcccd	#
4	q	bcccd	#b
5	q	cccd	#bb
6	r	ccd	#b
7	r	cd	#
8	r	d	#a
9	s	λ	#
10	t	λ	#

Tabla 5.6: Tabla operativa de la aceptación de la cadena $\alpha = \text{abbbcccd}$.

Ejemplo 5.6

En este ejemplo, se presenta una última variante en la utilización de la memoria de pila. El objetivo es validar una clave que debe contener los cinco dígitos impares ordenados en forma creciente, sin perjuicio que los mismos dígitos estén además presentes en otras posiciones. Ejemplos de claves correctas son 291437567819 y 5143850729, mientras que 5143860729 es una clave incorrecta por la falta del dígito cinco. Se pide: a) representar el grafo de un autómata para validar estas claves, b) definirlo formalmente y c) representar la aceptación de la clave 91325879 con la tabla operativa. Se dejan nuevamente al lector la identificación de las posibles condiciones de error, que corresponden a la lectura completa de la cadena de entrada sin haber detectado la secuencia 1-3-5-7-9.

a) Grafo del autómata con pila

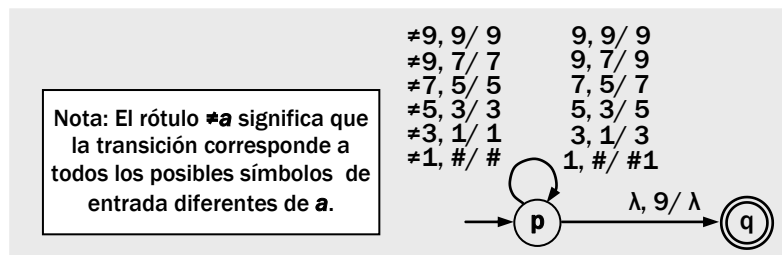


Figura 5.9: Grafo del AP del Ejemplo 5.6 (no incluye estado de error).

b) La definición formal del AP tendrá como componentes a:

$$\Sigma_E = \{0, 1, 2, 3, \dots, 9\}, \quad q_0 = p, \quad Q = \{p, q\}$$

$$\Gamma = \{\#, 1, 3, 5, 7, 9\}, \quad \# = \text{pila vacía}, \quad A = \{q\}$$

Unidad 5: Autómatas con Pila

$f:$	1	3	5	7	9	λ	Pares
p, #	p, #1	p, #	p, #	p, #	p, #		p, #
p, 1	p, 1	p, 3	p, 1	p, 1	p, 1		p, 1
p, 3	p, 3	p, 3	p, 5	p, 3	p, 3		p, 3
p, 5	p, 5	p, 5	p, 5	p, 7	p, 5		p, 5
p, 7	p, 7	p, 7	p, 7	p, 7	p, 9		p, 7
p, 9	p, 9	p, 9	p, 9	p, 9	p, 9	q, λ	p, 9

Tabla 5.7: Función f del AP del Ejemplo 5.6.

c) Tabla operativa de la validación de la cadena **91325879**:

No	Estado	Cadena a leer	Contenido de pila
1	p	91325879	#
2	p	1325879	#
3	p	325879	#1
4	p	25879	#3
5	p	5879	#3
6	p	879	#5
7	p	79	#5
8	p	9	#7
9	p	λ	#9
10	q	λ	#

Tabla 5.8: Tabla operativa del Ejemplo 5.6.

Llegado a este punto se ha completado, a través de sus conceptos fundamentales y numerosos ejemplos, una presentación general de los autómatas con pila determinista y no determinista. Se pudo comprobar que se trata de un autómata finito dotado de una memoria LIFO y que este nuevo recurso amplía enormemente el alcance de los problemas que este autómata puede resolver. En el apartado siguiente se muestra una de las principales aplicaciones del autómata con pila, que es la comprobación que las sentencias de los lenguajes de programación responden a las exigencias de sus gramáticas.

Autómatas con pila asociados a una gramática

Los autómatas con pila destinados a validar ciertos lenguajes fueron definidos hasta aquí a través de un proceso de prueba y error, inspeccionando la forma general de las cadenas del lenguaje y aplicando intuición y experiencia. Pudo comprobarse en los ejemplos anteriores que se trata de un procedimiento efectivo pero nada sistemático, que carece de generalidad. Además, nadie puede asegurar que han sido considerados todos los posibles tipos de sentencias que forman parte de cierto lenguaje. Lo expuesto puede ilustrarse con el esquema presentado en la Figura 5.10.

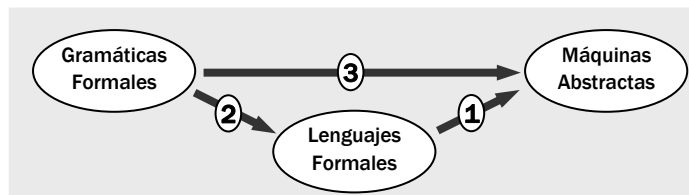


Figura 5.10: Vínculo entre gramáticas, lenguajes y máquinas.

Lo realizado fue definir las máquinas reconocedoras a partir de una inspección de las sentencias de un lenguaje, lo que es representado por la línea de flujo “1”. Sin embargo, como todo lenguaje formal responde a una gramática (línea de flujo “2”), lo que se propone es definir las máquinas abstractas a partir de sus reglas de producción (línea de flujo “3”). Esta idea ya fue puesta en práctica cuando se definieron autómatas finitos a partir de las reglas de reescritura de gramáticas regulares y lo que se hace ahora es extenderla para la definición de autómatas con pila a partir de gramáticas independientes de contexto. Este enfoque permite superar la ya señalada falta de generalidad y asegura que la máquina propuesta, reconocerá todas las posibles sentencias de un lenguaje especificado por una gramática.

En el caso de las gramáticas independientes del contexto, estos autómatas llevan el nombre de *analizadores sintácticos* y cobran una especial importancia por tratarse de las gramáticas de los lenguajes de programación. Los analizadores sintácticos fueron intensivamente estudiados a partir de 1970, coincidiendo con la inquietud por proponer, formalizar e implementar nuevos lenguajes y sus compiladores, entre ellos el Pascal (1970), Prolog (1970), C (1971), SmallTalk (1971), PL/M (1972), Modula-2 (1978), ADA (1979) y C++ (1983), por citar los más conocidos. A partir de este enorme esfuerzo de investigación y desarrollo, se detectaron numerosos patrones de funcionamiento de los analizadores sintácticos, que condujeron a la creación de programas generadores de analizadores. Puede citarse como ejemplo el caso de *Yacc* (*Yet Another Compiler-Compiler*) que genera analizadores sintácticos en lenguaje de programación C, basados en una gramática del lenguaje escrita en notación BNF. Otro ejemplo es el generador de analizadores sintácticos de código abierto *JavaCC* (*Java Compiler-Compiler*) que entrega sus resultados en el lenguaje de programación Java. La denominación de “Compiler Compiler” proviene del hecho que estos generadores tienen por finalidad la construcción de analizadores sintácticos, los que a su vez forman parte de la *etapa de análisis* de los compiladores. Esta etapa incluye las fases de *análisis léxico*, destinado a identificar los componentes del lenguaje, *análisis sintáctico*, que verifica que la estructura responda a las reglas de reescritura de la gramática, y finalmente la fase de *análisis semántico* que se ocupa del sentido y contenido lógico de las sentencias. Habiendo llegado a este punto se recomienda al lector remitirse al Apéndice A con el fin de familiarizarse con los conceptos de compiladores.

En una primera clasificación, los analizadores sintácticos pueden agruparse en *descendentes* y *ascendentes*. Dentro de estos grupos se distinguen familias de analizadores que responden a diferentes criterios, entre ellos las formas normales en las que deben estar expresadas las producciones de las gramáticas. El determinismo de los analizadores y la utilización de procedimientos recursivos, son otros criterios de clasificación.

Analizadores Sintácticos Descendentes (ASD)

El *Analizador Sintáctico Descendente* (ASD), en inglés *Top-Down Parser*, comienza a operar a partir del axioma de la gramática y procura desarrollar por izquierda el árbol de derivación sintáctica a medida que la sentencia es leída. Si este proceso puede ser continuado hasta

Unidad 5: Autómatas con Pila

completarse la lectura de la cadena significa que la misma responde a las producciones de la gramática y, por lo tanto, es aceptada.

A. ASD para gramáticas en Forma Normal de Greibach (FNG)

Dada la gramática $G = (\Sigma_T, \Sigma_N, S, P)$ cuyas reglas de producción están en forma normal de Greibach, se define el siguiente autómata con pila:

$$AP = (\Sigma_T, \Sigma_N \cup \{\#\}, \{p, q, r\}, p, \#, \{r\}, f)$$

y la función de transición se define a partir de las reglas de reescritura según la manera en que estas reglas están estructuradas:

$$\left. \begin{array}{ll} - A := a\eta & \Rightarrow f(q, a, A) = (q, \eta) \\ - A := \lambda & \Rightarrow f(q, \lambda, A) = (q, \lambda) \\ - A := a & \Rightarrow f(q, a, A) = (q, \lambda) \end{array} \right\} \begin{array}{l} A \in \Sigma_N \\ a \in \Sigma_T \\ \eta \in \Sigma_N^* \end{array}$$

a las que se agregan $f(p, \lambda, \#) = (q, \#S)$

$$f(q, \lambda, \#) = (r, \#)$$

Esta definición de la función de transición implica que para toda forma sentencial que pueda ser derivada por izquierda, tal como $S \rightarrow^* \beta\alpha$, donde $\beta \in \Sigma_T^*$, $\alpha \in \Sigma_N^*$ será posible el movimiento $(q, \beta, \#) \vdash^* (q, \lambda, \alpha)$.

Es decir que si la gramática permite derivar por izquierda la forma sentencial $\beta\alpha$, significa que a partir de la lectura de la cadena β el autómata podrá pasar a una configuración en la que β haya sido leída y α esté cargada en la pila. Esta afirmación no es obvia y su verificación será objeto del Ejemplo 5.8 que se presenta más adelante.

El grafo de este autómata, de solo tres estados, aceptará las sentencias desarrolladas a partir de la gramática G por vaciado de pila y estado de aceptación, tal como fue definido al tratar la aceptación de lenguajes por parte del AP . El grafo de este analizador sintáctico descendente toma la forma:

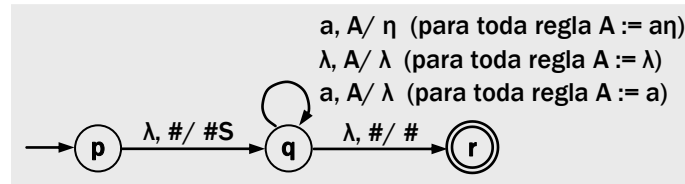


Figura 5.11: analizador sintáctico descendente para FNG.

Ejemplo 5.7

Determine el analizador sintáctico que corresponde a la gramática cuya definición y producciones se muestran a continuación. Compruebe luego la aceptación de la cadena $\delta = aab$. La gramática G está expresada en forma normal de Greibach:

$$G = (\{a, b\}, \{A, B, C, D\}, A, P)$$

$$P = \{A := bBC \mid aB \mid \lambda, B := aD \mid aC \mid a, C := b, D := bD \mid bC\}$$

La definición formal del analizador sintáctico o autómata con pila es:

$$AP = (\Sigma_E, \Gamma, Q, q_0, \#, A, f)$$

$$AP = (\{a, b\}, \{\#, A, B, C, D\}, \{p, q, r\}, p, \#, \{r\}, f)$$

La función de transición es presentada en la Tabla 5.9, el grafo del analizador sintáctico en la Figura 5.12 y el árbol de configuraciones o descripciones instantáneas del proceso de aceptación

Unidad 5: Autómatas con Pila

de la cadena propuesta en la Figura 5.13, a continuación. Se sugiere al lector comprobar la relación entre estas tres representaciones.

f:	a	b	λ
p, #			q, #A
q, A	q, B	q, BC	q, λ
q, B	q, C q, D q, λ		
q, C		q, λ	
q, D		q, D q, C	
q, #			r, #

Tabla 5.9: Función de transición del analizador sintáctico descendente.

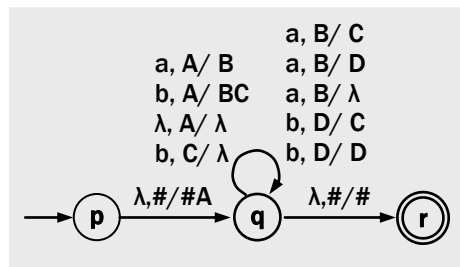


Figura 5.12: Grafo del analizador sintáctico descendente.

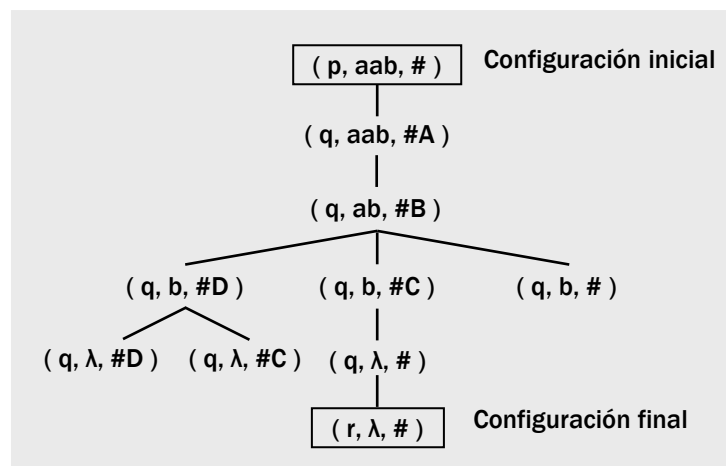


Figura 5.13: Árbol de configuraciones para β =aab del Ejemplo 5.7.

Ejemplo 5.8

Comprobar que, si la gramática del ejercicio anterior permite derivar por izquierda la forma sentencial $\beta\alpha$, el autómata podrá, a partir de la lectura de la cadena β , llegar a una configuración en la que β ha sido leída y α esté cargada en la pila.

Para mostrarlo, se comienza por derivar una sentencia por izquierda, por ejemplo la que sigue:

$A \rightarrow bBC \rightarrow baDC \rightarrow babDC \rightarrow babb\overset{\alpha}{\boxed{CC}} \rightarrow babbbC \rightarrow babbbb$

Por su parte, la representación de la aceptación de la misma sentencia $\delta = babbbb$ por parte del analizador sintáctico toma la forma que se muestra en la Figura 5.14.

Considérese, por ejemplo, la cuarta forma sentencial obtenida en el proceso de derivación. En este caso, el valor del prefijo β es **babb** y el sufijo α contiene **CC** (recuadrado con línea punteada). Puede observarse que cuando el analizador sintáctico leyó el prefijo **babb**, es decir, la cadena que falta leer es **bb**, el contenido de la pila es α =**CC** (también recuadrado). Esto confirma

Unidad 5: Autómatas con Pila

lo anticipado en la presentación del analizador. Se sugiere comprobar que puede establecerse en todos los casos esta misma relación, entre las formas sentenciales de la derivación de la cadena **babbbb** y las descripciones instantáneas del comportamiento del autómata.

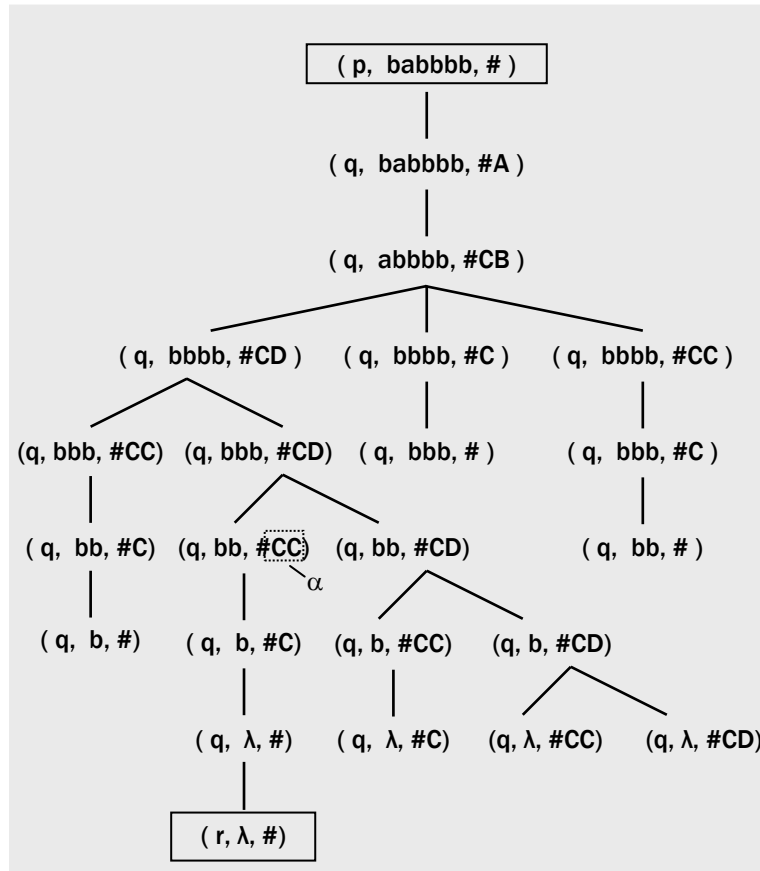


Figura 5.14: Árbol de descripciones instantáneas para $\delta = \text{babbbb}$.

B. ASD para gramáticas independientes del contexto en forma general

Se presenta ahora un analizador sintáctico que no requiere la previa conversión de la gramática a una forma normal, es decir que admite que las reglas de reescritura tengan la forma general $A := \alpha$, siendo $A \in \Sigma_N$ un símbolo no terminal y $\alpha \in (\Sigma_T \cup \Sigma_N)^*$.

Dada una gramática definida como $G = (\Sigma_T, \Sigma_N, S, P)$ el autómata con pila no determinista se define como:

$$AP = (\Sigma_T, \Sigma_T \cup \Sigma_N \cup \{\#\}, \{p, q, r\}, p, \#, \{r\}, f)$$

Hasta aquí puede comprobarse una primera diferencia con respecto al analizador descendente anterior y es el alfabeto de pila Γ , que incluye ahora tanto los símbolos terminales como los no terminales.

La otra diferencia está naturalmente en la función de transición, que es definida de la siguiente manera:

Se incorporan: $f(p, \lambda, \#) = (q, \#S)$ donde $S \in \Sigma_N$ es el axioma

$f(q, a, a) = (q, \lambda)$ por cada $a \in \Sigma_T$

$f(q, \lambda, A) = (q, \alpha)$ por cada regla $A := \alpha$

y por último: $f(q, \lambda, \#) = (r, \#)$.

Este AP puede representarse según se muestra a continuación:

Unidad 5: Autómatas con Pila

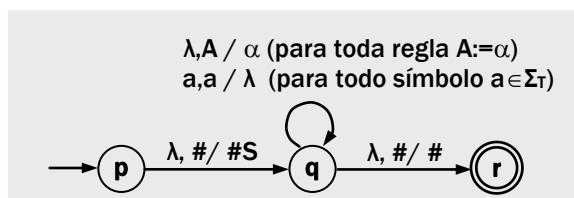


Figura 5.15: Analizador sintáctico descendente para GIC.

Puede comprobarse que el símbolo en la cima de pila es el que conduce el proceso de validación. En efecto, cada vez que ese símbolo es un no terminal **A**, se reemplaza en la pila por la cadena α , y cada vez que en el tope de pila hay un terminal se espera encontrar el mismo símbolo en la cadena de entrada, y es cancelado. Se prosigue así hasta que la sentencia de entrada es completamente leída y en esa condición la pila debería estar vacía. Aquí debe observarse que pueden haber múltiples reglas de reescritura con el mismo símbolo no terminal en el primer miembro, y esto hace que se trate nuevamente de un APND.

Los dos analizadores sintácticos descendentes hasta aquí estudiados, son esencialmente no deterministas y suelen ser llamados en general, LL básicos. El acrónimo LL proviene del inglés *Left-Left*, donde la primera **L** se refiere a que la cadena es leída de izquierda a derecha y la segunda **L** a que el proceso que desarrolla corresponde al de una derivación por izquierda. El prestigioso Niklaus Wirth, precursor de la ciencia de la computación y autor de lenguajes tales como Pascal y Modula, dedicó especial atención a los analizadores sintácticos LL y propuso numerosas variantes tendientes a su optimización a través de procesos de preanálisis, generando los analizadores predictivos llamados LL(k). El recurso del preanálisis se presenta más adelante y posibilita disponer de analizadores LL deterministas.

Ejemplo 5.9

Empleando la misma gramática del Ejemplo 5.7 defina el AP y compruebe la aceptación de la cadena $\delta = aab$.

$$G = (\{a, b\}, \{A, B, C, D\}, A, P)$$

$$P = \{A := bBC \mid aB \mid \lambda, B := aD \mid aC \mid a, C := b, D := bD \mid bC\}$$

La definición formal del autómata con pila es:

$$AP = (\{a, b\}, \{\#, A, B, C, D, a, b\}, \{p, q, r\}, p, \#, \{r\}, f)$$

y en base a la definición del analizador sintáctico se construye la tabla de transición 5.10, mostrada a continuación:

f:	a	b	λ
p, #			q, #A
q, A			q, bBC q, aB q, λ
q, B			q, aD q, aC q, a
q, C			q, b
q, D			q, bD q, bC
q, a	q, λ		
q, b		q, λ	
q, #			r, #

Tabla 5.10: Relación de transición del ap del Ejemplo 5.9.

Las múltiples definiciones de la relación de transición que se presentan en las celdas correspondientes a las transiciones λ , dan claramente al autómata generado su característica de

Unidad 5: Autómatas con Pila

no determinista. Esto hará que existan varios caminos distintos desde una configuración inicial hasta una final, para el tratamiento de una cadena dada, como se comprobará a continuación.

En la Figura 5.16 se presenta el grafo correspondiente:

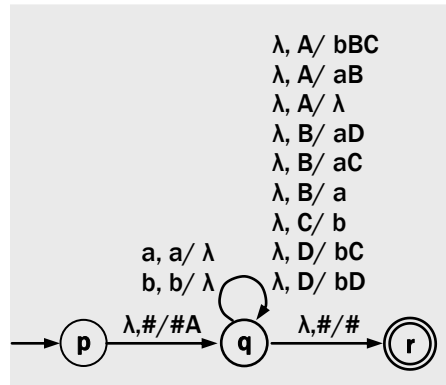


Figura 5.16: Grafo del AP del Ejemplo 5.9.

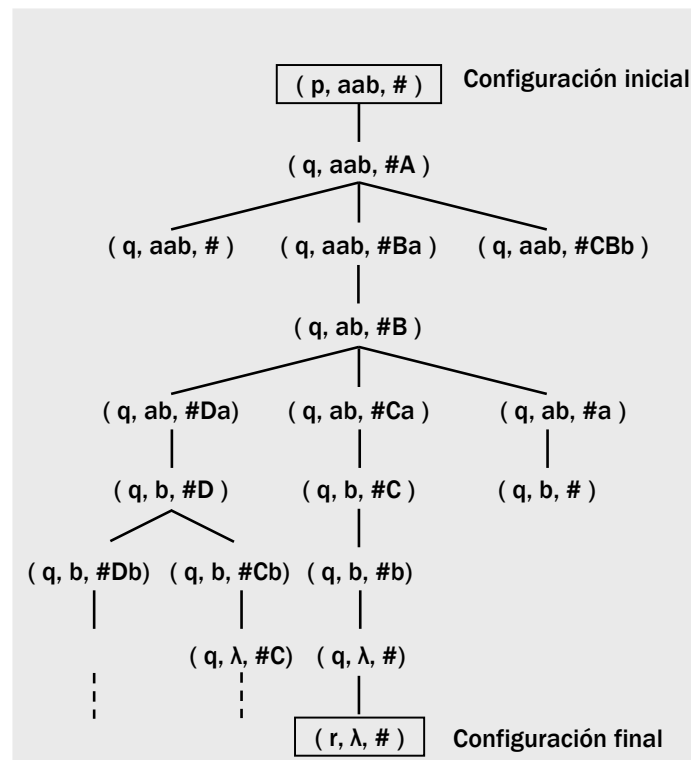


Figura 5.17: Árbol de configuraciones para $\delta = aab$ del Ejemplo 5.9.

El árbol de descripciones instantáneas mostrado en la Figura 5.17, corresponde al proceso efectuado por el autómata definido sobre la cadena $\delta = aab$; puede verse su frondosa ramificación provocada por el no determinismo.

Analizadores Sintácticos Ascendentes (ASA)

Los *Analizadores Sintácticos Ascendentes* (ASA) comprueban la validez de la sentencia operando de abajo hacia arriba. Es decir, recorren el árbol de derivación sintáctica desde las hojas hacia su raíz, que es el axioma de la gramática. En inglés, son denominados *Bottom-Up Parsers* y, en general, estos analizadores disponen de mayor poder de reconocimiento que los analizadores descendentes. Hay numerosas propuestas que responden a esta concepción ascendente y aquí se van a considerar los denominados LR básicos, que son no deterministas, y cuyo nombre proviene de las cadenas que son leídas de izquierda a derecha (*Left*) y se sigue un proceso de reducción por izquierda. Debe recordarse aquí que la reducción es el proceso por el cual se transita desde la sentencia al axioma de la gramática, aplicando las reglas de producción en orden inverso, es decir que se opera en sentido opuesto al de la derivación. Además, puede comprobarse que a una

Unidad 5: Autómatas con Pila

reducción por izquierda le corresponde una derivación por derecha (*Right*) y de allí el segundo carácter de la denominación LR.

Al igual que en los analizadores descendentes, la formulación básica de los ascendentes es esencialmente no determinista. Esto se origina en la búsqueda del camino hacia la aceptación de la cadena y es conocido como *el problema del retroceso*: al existir varios caminos a seguir, se debe probar primero uno de ellos y si no se consigue el objetivo (aceptación) deberá volverse e intentarlo por otro. Los analizadores más avanzados recurren a algoritmos que simulan y resuelven de manera determinista un problema esencialmente no determinista, lo que permite implementar los analizadores LL y LR sin la necesidad de costosas búsquedas en árboles con factores de ramificación elevado. Estas soluciones son sumamente imaginativas e interesantes, pero por requerir un extenso tratamiento y no ser nuestro objetivo entrar aquí en detalles de implementación, quedan fuera del alcance previsto para este libro; se mostrarán sin embargo, algunos ejemplos simples de analizadores sintácticos con preanálisis en el punto 4 de este capítulo.

Volviendo específicamente a los *analizadores sintácticos ascendentes*, en lo sucesivo, se trabajará con el LR no determinista básico y se lo utilizará en ejemplos y ejercicios. Así, dada una gramática independiente de contexto $G = (\Sigma_T, \Sigma_N, S, P)$, se presenta un analizador sintáctico ascendente que es implementado con un autómata con pila no determinista con los siguientes componentes:

$$AP = (\Sigma_T, \Sigma_T \cup \Sigma_N \cup \{\#\}, \{p, q\}, p, \#, \{q\}, f)$$

Su función de transición es definida de la siguiente forma:

$$f(p, a, b) = (p, ba) \quad \text{para todo } a \in \Sigma_T \text{ y } b \in (\Sigma_T \cup \Sigma_N)$$

$$f(p, \lambda, \alpha) = (p, A) \quad \text{por cada regla } A := \alpha \text{ en } P$$

y por último: $f(p, \lambda, \#S) = (q, \#)$ donde S = axioma

Con respecto a la operación de este autómata, deben observarse diversos aspectos:

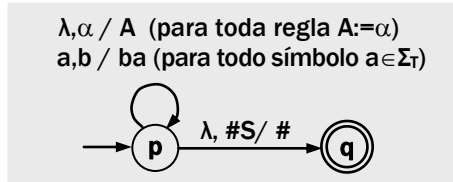


Figura 5.18: Analizador sintáctico ascendente para GIC.

- En cada intervalo de tiempo, se presenta la opción de desplazar un nuevo símbolo a de la entrada a la pila, o reducir una cadena α ya almacenada en la pila. Esto motiva también que se llame a estos analizadores *por reducción y desplazamiento*.
- Para poder hacer la reducción se debe reconocer la cadena α en la pila, lo que implica identificar anticipadamente varios símbolos en la pila; esto plantea una contradicción con la concepción de la memoria tipo lifo en la que solo se hace visible el símbolo que se encuentra en su tope. El problema se resuelve apelando al no determinismo del ap, con lo cual se puede asumir en la práctica a la pila como una estructura híbrida que opera como tal, pero admite la observación de todo su contenido.
- En la reducción de la cadena α , seguramente se presentarán diferentes opciones ya que es probable que numerosas producciones tengan a α en su segundo miembro. Por su lado, en el desplazamiento desde la entrada hacia la pila, también se presenta siempre la opción de seguir apilando símbolos de entrada o efectuar una reducción.
- La cadena será aceptada si a través de su completa lectura pudo conducirse la reducción hasta dejar en la pila solo el axioma de la gramática.

Ejemplo 5.10

Definir un analizador sintáctico ascendente para la gramática del Ejemplo 5.7:

Unidad 5: Autómatas con Pila

$G = (\{a, b\}, \{A, B, C, D\}, A, \{A:=bBC \mid aB \mid \lambda, B:=aD \mid aC \mid a, C:=b, D:=bD \mid bC\})$

y comprobar que la sentencia $\delta = \mathbf{babbbb}$ pertenece al lenguaje generado por ella. Para ello, se presenta la función de transición en la Tabla 5.11 y el grafo del AP en la Figura 5.19.

f	a	b	λ
p, bBC			p,A
p, aB			p,A
p, λ			p,A
p, aD			p,B
p, aC			p,B
p, a			p,B
p, b			p,C
p, bC			p,D
p, bD			p,D
p, x	p, xa	p, xb	
p, A			q, λ

Tabla 5.11: Función de transición f del AP del Ejemplo 5.10.

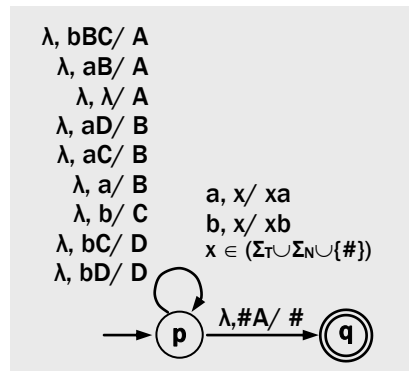


Figura 5.19: Grafo del AP del ejemplo 5.10.

El árbol de descripciones instantáneas que corresponde a la validación de la cadena $\delta = \mathbf{babbbb}$ es mostrado en la Figura 5.20, donde puede verse la gran cantidad de ramificaciones provocadas por el no determinismo del autómata diseñado con la técnica de análisis sintáctico ascendente.

Unidad 5: Autómatas con Pila

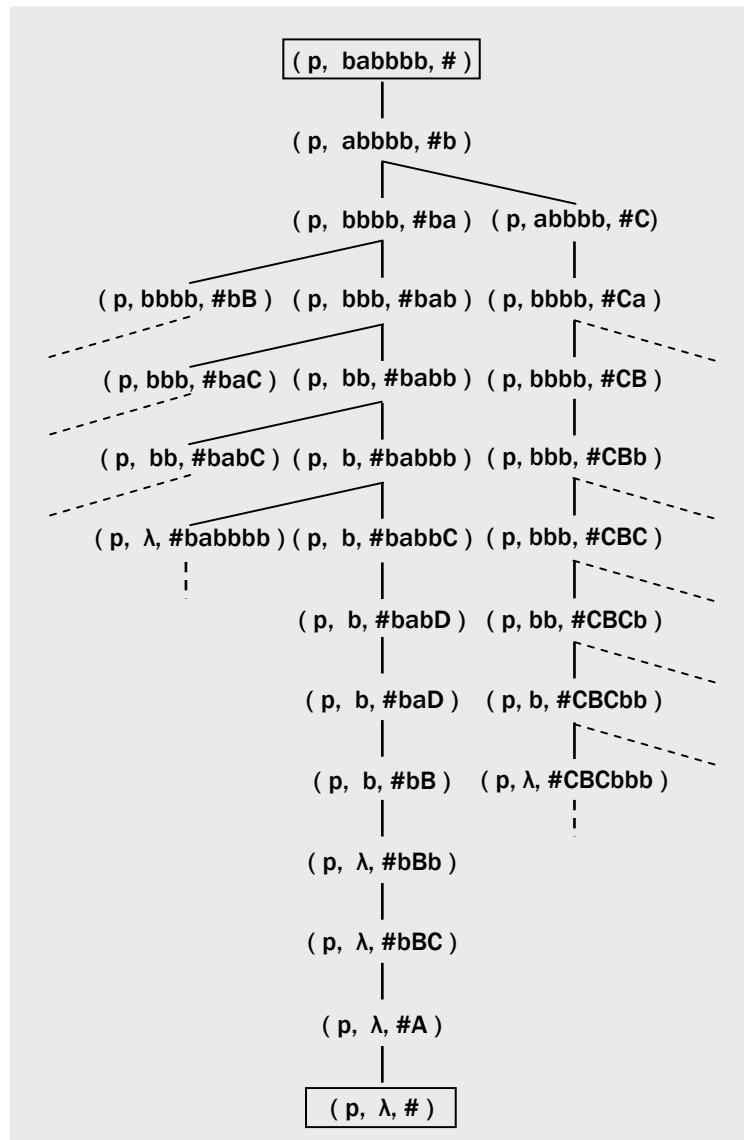


Figura 5.20: Árbol de configuraciones para $\delta = \text{babbbb}$.

En el árbol de la figura anterior, se muestra la secuencia de movimientos desde la configuración inicial hasta la configuración final de aceptación, y también se muestran, con fines ejemplificativos, algunas de las ramificaciones que no conducen a aceptar la cadena. Como puede comprobarse, primero se traslada la cadena de entrada a la pila, hasta un punto en el que sea posible comenzar con la reducción hacia el axioma, y luego se concreta la reducción hasta que el axioma queda en la pila. Como ya se señaló, también puede comprobarse que se trata de un árbol con una enorme cantidad de alternativas, cuya exploración demanda un considerable esfuerzo.

Para facilitar la interpretación de la operación realizada por el AP se muestra a continuación la derivación por derecha generada de la sentencia y su árbol de derivación sintáctico.

$$A \rightarrow bBC \rightarrow bBb \rightarrow baDb \rightarrow babDb \rightarrow babbCb \rightarrow babbbb$$

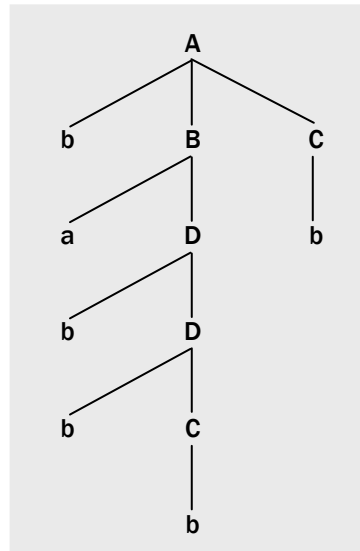


Figura 5.21: Árbol de derivación de la sentencia $\delta = \text{babbbb}$.

Se recomienda al lector seguir la actividad del analizador sintáctico ascendente sobre el árbol de descripciones instantáneas y la progresiva reducción de la sentencia, utilizando las producciones, hasta alcanzar el axioma. Como ya fue anticipado, esto es la demostración concluyente de que la sentencia fue generada a partir de las reglas de producción de la gramática y, por lo tanto, pertenece al lenguaje generado por ella.

Ejemplo 5.11

En el Ejemplo 5.1, se propuso un autómata con pila capaz de reconocer un lenguaje de palíndromos de largo impar y, en el Ejemplo 5.2, se propuso otro AP para palíndromos de largo par. A partir de estos ejemplos, se comprobó que el primer problema podía ser resuelto por un AP determinista y el segundo requería de un AP no determinista. Ambos autómatas fueron definidos a partir de un reconocimiento de las particularidades de las sentencias correspondientes a cada caso.

En este ejemplo, se utilizarán los conceptos aprendidos sobre analizadores sintácticos y se propondrán máquinas capaces de reconocer similares lenguajes, solo que se lo hará a partir de las reglas de reescritura de la gramática que los genera.

Naturalmente, el primer paso es conocer las producciones de la gramática, que para el caso estudiado pueden ser determinadas con facilidad ya que se trata de un lenguaje muy simple:

$$L = \{ \alpha c \alpha^{-1} \mid \alpha \in \{a, b\}^+ \} \cup \{ \alpha \alpha^{-1} \mid \alpha \in \{a, b\}^+ \}$$

La gramática buscada toma la forma:

$$G = \{ \{a, b, c\}, \{S\}, S, P \}$$

donde:

$$P = \{ S := aSa \mid bSb \mid aa \mid bb \mid aca \mid bcb \}$$

Se realizan algunas derivaciones muy simples para confirmar que desde el axioma pueden alcanzarse sentencias del tipo de las consideradas:

$$S \rightarrow aSa \rightarrow abSba \rightarrow abbcba$$

$$S \rightarrow aSa \rightarrow abSba \rightarrow abbbba$$

$$S \rightarrow bSb \rightarrow bbSbb \rightarrow bbaSabb \rightarrow bbaacaabb$$

A. Analizador sintáctico descendente

En primer lugar, se implementa un analizador sintáctico descendente que opera con la forma general de las gramáticas tipo 2 de Chomsky. El autómata queda definido como:

$$AP = (\{a, b, c\}, \{a, b, c, S, \#\}, \{p, q, r\}, p, \#, \{r\}, f)$$

Unidad 5: Autómatas con Pila

El grafo es mostrado en la siguiente figura y se deja al lector la presentación de la función de transición en una tabla.

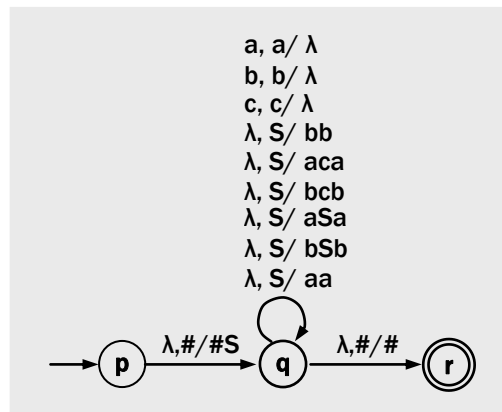


Figura 5.22: Grafo del analizador sintáctico descendente.

Como se ve, la definición del autómata es una tarea totalmente sistemática que responde a un patrón estándar, donde lo único específico son las transiciones que dependen de las reglas de reescritura. En este caso, el árbol de descripciones instantáneas que representa la aceptación de la sentencia $\delta = \text{abbcbbba}$ se muestra en la Figura 5.23.

En la figura puede observarse que cada vez que el axioma S queda al tope de pila se presenta en el árbol un factor de ramificación igual a la cantidad de reglas de producción que tiene a S en el primer miembro. En este caso, esto ocurre tres veces y eso lleva a que el árbol de descripciones instantáneas presente un total de 18 opciones que deben ser exploradas. Nótese que en el árbol de la Figura 5.23 se representan solo algunas de las posibles opciones y con líneas de trazos se indica que hay otras. Aquí cabe comparar el árbol de la Figura 5.23 con el de la Figura 5.2, que resuelve un problema similar sin ramificaciones por tratarse de un AP determinista, o el de la Figura 5.4 que corresponde a la aceptación de una cadena similar y presenta solo cuatro opciones a ser exploradas.

Esto es una demostración de que toda ventaja tiene siempre algún costo, y en este caso la disponibilidad de un analizador sistemático y general paga el precio de un mayor esfuerzo para confirmar la aceptación o no de una sentencia. Se invita al lector a reflexionar sobre esto.

Unidad 5: Autómatas con Pila

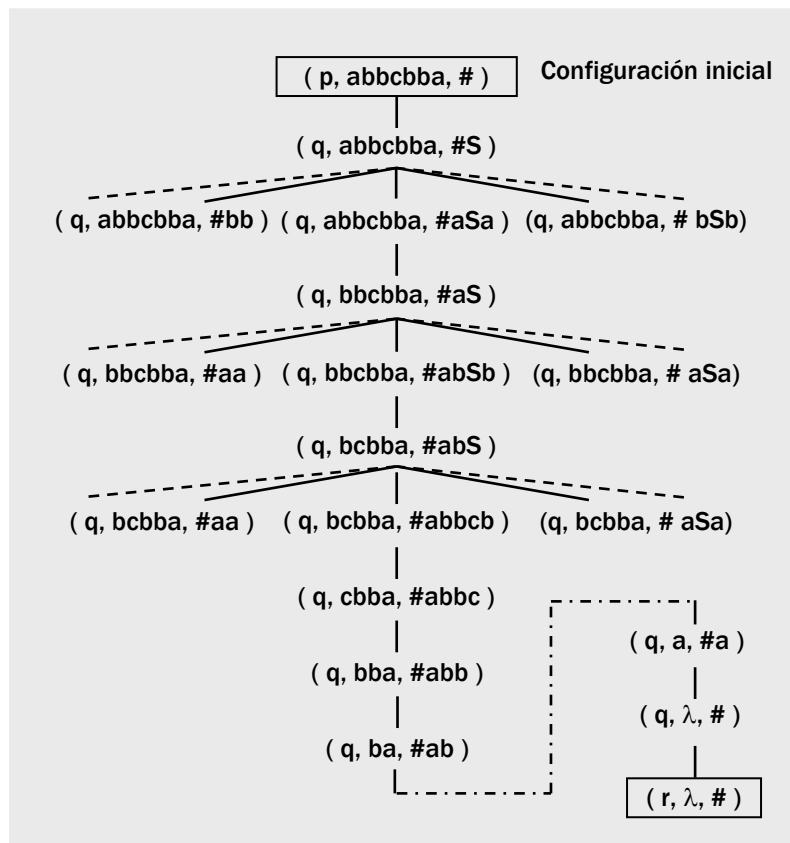


Figura 5.23: Árbol de configuraciones para $\delta = \text{abbcbbba}$.

B. Analizador sintáctico ascendente

El mismo problema será resuelto mediante un analizador sintáctico de tipo ascendente, mostrando su definición, el grafo en la Figura 5.24 y nuevamente se deja al lector la presentación de la función de transición en una tabla.

El autómata queda definido como:

$$AP = (\{a, b, c\}, \{a, b, c, S, \#\}, \{p, q\}, p, \#, \{q\}, f)$$

Aquí también la definición del autómata es una tarea sistemática con un patrón estándar, que queda orientado a resolver cierto problema específico a partir de sus transiciones que surgen de las reglas de producción de la gramática. El árbol de descripciones instantáneas representado en la Figura 5.25 corresponde a la aceptación de la misma sentencia $\delta = \text{abbcbbba}$.

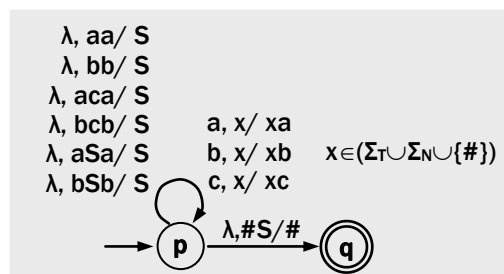


Figura 5.24: Grafo del analizador sintáctico ascendente.

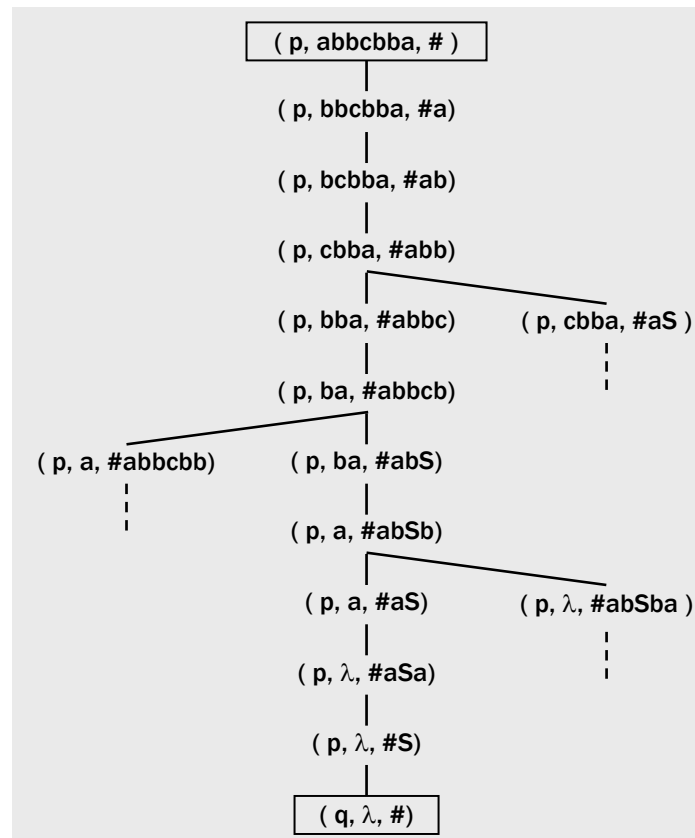


Figura 5.25: Árbol de configuraciones para $\delta = abbcbbba$.

En este caso, el árbol de descripciones instantáneas presenta menos ramificaciones que el árbol que corresponde al analizador descendente, pero ésta no es una regla general. Esto depende de la variedad de las reglas de producción de la gramática, lo que hace que uno u otro árbol implique un mayor esfuerzo para la aceptación de una misma sentencia. Como se recordará, en los procesos de reconocimiento representados en los árboles de las Figuras 5.14 y 5.20, ocurrió precisamente lo contrario. Si bien no hay conclusiones definitivas, como ya se dijo, los analizadores ascendentes tienen generalmente mayor capacidad de reconocimiento que los descendentes y como contrapartida los procesos asociados suelen ser más complejos.

Analizadores sintácticos con preanálisis

Tal como fue anticipado y confirmado con los ejemplos precedentes, los *analizadores* básicos, ya sean descendentes o ascendentes, son Autómatas con Pila No Deterministas (APND) donde el proceso de validación de cadenas implica resolver un problema en un espacio de estados que queda representado por un árbol con elevado factor de ramificación, lo que inevitablemente impacta en la eficiencia del proceso.

Para superar esta limitación se utiliza la técnica llamada **preanálisis**, que implica conocer anticipadamente los próximos k símbolos a ser leídos por el autómata. En base a este conocimiento, el AP selecciona su próximo movimiento, pudiendo predecir cuál de todas las posibles producciones aplicables es la que conducirá a la aceptación (esto es, elegir la rama correcta en el árbol de configuraciones).

Así se pueden sortear la mayoría de los inconvenientes que presentan los analizadores sintácticos básicos ascendentes y descendentes, dando lugar a los denominados analizadores sintácticos *predictivos*, de los cuales los analizadores por descenso recursivo, LL(k) y LR(k), son los más conocidos. A partir de esta idea, se plantea una jerarquía para los analizadores sintácticos, cuya característica distintiva es el número de símbolos de preanálisis que puede conocer.

Para ambos tipos de analizadores se han desarrollado numerosas y variadas propuestas para sus implementaciones. Para los analizadores descendentes, el proceso por descenso recursivo genera un autómata finito para cada no terminal de la gramática, en el cual se transita

Unidad 5: Autómatas con Pila

desde el estado inicial hasta el de aceptación siguiendo un camino etiquetado con los símbolos del lado derecho de la producción de ese no terminal. Si la transición de un estado a otro se indica con un terminal y ese terminal es el símbolo de preanálisis, se efectiviza la transición; si la transición hacia el siguiente estado está indicada con un símbolo no terminal, entonces “se ejecuta” el AF definido para el mismo (ver Ejemplo 5.12). Otro analizador descendente (no recursivo) es el denominado $LL(k)$ en el cual se adiciona una tabla de decisión al AP que permite seleccionar la alternativa a utilizar en caso de haber más de una (ver Ejemplos 5.12 y 5.13). Para que estos analizadores puedan construirse, la gramática independiente del contexto debe ser no ambigua, no debe tener recursión por izquierda y sus producciones deben estar factorizadas por izquierda; con esto se intenta lograr que con un solo símbolo de preanálisis se sepa qué regla utilizar.

En el caso de los analizadores $LR(k)$, se mencionan, a continuación, algunos de los más importantes:

- $LR(0)$: No utiliza preanálisis, por lo que es apropiado cuando las producciones de las gramáticas no generan transiciones alternativas. Es el más fácil de implementar y, a la vez, el que dispone de menor poder de reconocimiento.
- $SLR(1)$: Utiliza un solo símbolo de preanálisis y su nombre (del inglés) significa *Simple LR*.
- $LR(1)$: Es un analizador ascendente LR clásico que utiliza un símbolo de preanálisis. La implementación de su AP conduce a estructuras de datos muy grandes y ha justificado el desarrollo de generadores automáticos.
- $LALR(1)$: Su nombre proviene del inglés *Look-Ahead LR*, en el que se combina la potencia del $LR(1)$ con la simplicidad y eficiencia del $SLR(1)$. Éste es el método que utiliza YACC para la generación de analizadores sintácticos en C.

Todas estas versiones de LR anexan una tabla al AP que guía su funcionamiento cuando hay alternativas.

En su forma más elemental, el preanálisis implica conocer anticipadamente el siguiente símbolo al actualmente leído por el autómata ($k = 1$). Con el objetivo de esclarecer las ideas en torno a este tema se hará en este apartado una introducción a los analizadores $LL(1)$. Para ello, se han seleccionado algunos ejemplos simples que se desarrollan a continuación.

Ejemplo 5.12

Sea $L_{(5.12)} = \{a^n b^n \mid n \geq 0\}$ el sencillo lenguaje de cadenas de $\{a, b\}^*$ que tienen un prefijo formado solo por símbolos **a**, al cual le continúa un sufijo formado solo por símbolos **b**, de igual largo. Una gramática independiente del contexto para este lenguaje es:

$$G_{(5.12)} = (\{a, b\}, \{S\}, S, \{S := aSb \mid \lambda\})$$

donde la primera producción se utiliza en una derivación desde el axioma **S**, para hacer crecer los prefijos y sufijos hasta el largo requerido (aplicándola **n** veces), y la segunda sirve, a la vez, para terminar este ciclo de aplicación de la primera regla, como para obtener la cadena vacía en el caso $n=0$.

El autómata con pila que reconoce el lenguaje generado por la gramática, según procedimiento presentado al estudiar el analizador sintáctico descendente, se muestra a continuación:

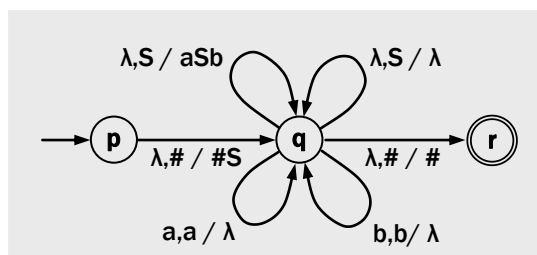


Figura 5.26: Grafo del AP del Ejemplo 5.12.

Unidad 5: Autómatas con Pila

Nótese que en el grafo solo se muestran las transiciones que se diseñaron para la aceptación de las cadenas del lenguaje $L_{(5.12)}$; luego toda situación no prevista que pudiera presentarse para una cadena en particular, llevará al autómata a rechazarla.

$$AP = (\{a, b\}, \{a, b, S, \#\}, \{p, q, r\}, p, \#, \{r\}, f)$$

$f:$	a	b	λ
p, #			q, #S
q, S			q, aSb q, λ
q, a	q, λ		
q, b		q, λ	
q, #			r, #

Tabla 5.12: Definición formal del AP del Ejemplo 5.12.

También en la tabla de la función f de su definición formal, solo se muestran aquellas filas correspondientes a combinaciones de estado y símbolo de pila que producen transiciones válidas para algún símbolo de entrada o λ . En particular, puede verse en ella el no determinismo de este autómata, cuando se encuentra en el estado q estando el axioma S en la cima de la pila: sin leer en su entrada (transición λ), podrá optar por reemplazar el símbolo S en la pila por la cadena aSb o sencillamente extraerlo.

Este no determinismo del AP hace que el proceso de análisis de una cadena, tenga que explorar posiblemente más de una rama del árbol de descripciones instantáneas para aceptarla. Al querer confeccionar un programa de cómputo que lo implemente, éste podrá en su código:

- Desarrollar una estrategia de vuelta atrás cronológica (*backtracking*) cuando encuentre la situación de no determinismo, para reemplazar en la pila S por la cadena aSb y continuar analizando la cadena que resta por leer, para retornar a este punto en caso de no aceptación e intentarlo nuevamente por el otro camino, eliminando S de la pila y efectuando nuevamente el proceso de la cadena por leer (seguir la otra rama posible del árbol de configuraciones).²
- Aplicar la técnica de preanálisis, esto es, “espíar” el siguiente símbolo de la cadena que resta por leer en la entrada ¡sin leerlo!, para poder determinar cuál regla es conveniente aplicar, anticipando de esta forma la rama correcta del árbol de descripciones instantáneas a seguir. Si aún hay un símbolo a por leer, conviene reemplazar S en la pila por aSb , pero si el siguiente símbolo a leer es b (lo que dice que se terminó el prefijo de aes), entonces conviene eliminar S de la pila sin reemplazarlo.

Interesa ahora diseñar un algoritmo de análisis sintáctico para este lenguaje con la segunda propuesta, esto es, utilizando preanálisis para obtener un algoritmo predictivo. Usaremos primero el algoritmo de análisis sintáctico por descenso recursivo y luego, un algoritmo descendente no recursivo **LL(1)**.

Para el análisis sintáctico por descenso recursivo, se genera un *AFND* para cada símbolo no terminal; en este caso solo lo tenemos a S (ver Figura 5.27). Por la primera producción $S := aSb$, se crean los estados S , p , q y r con las transiciones adecuadas para cada símbolo del lado derecho de la producción. Por la producción $S := \lambda$, se agrega la transición λ desde el estado inicial S al estado r de aceptación.

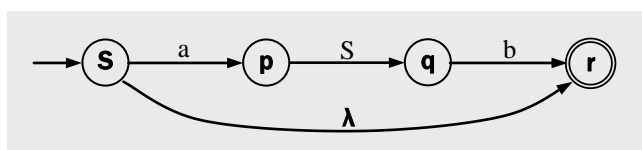


Figura 5.27: AFND para S en algoritmo de descenso recursivo del Ejemplo 5.12.

² Un algoritmo de vuelta atrás (retroceso) denominado **KPM**, por *Knuth Parsing Machine*, codificado en lenguaje Pascal puede verse en el Apéndice A del libro de Sanchis Llorca y Galán Pascual: *Compiladores*, Editorial Parainfo, Madrid, España, 1988.

Unidad 5: Autómatas con Pila

La transición λ de la Figura 5.27 se “ejecutará” si el símbolo de preanálisis es **b**. Si por el contrario, el símbolo de preanálisis es **a**, el autómata lo leerá y pasará al estado **p** y deberá “re-ejecutarse” para poder pasar al estado **q** (eso indica la etiqueta **S** de la transición de **p** a **q**; de aquí la denominación *recursivo* del algoritmo). Si puede llegar desde el estado **S** hasta el estado **r** en la re-ejecución, al retornar a la ejecución original el nuevo estado será **q** y, si estando en este estado puede leer un símbolo **b** en su entrada, pasará al estado **r** aceptando la cadena de entrada.

Este procedimiento puede presentarse conceptualmente en pseudocódigo como se muestra a continuación en la Figura 5.28; la variable *preanálisis* siempre contiene el siguiente símbolo de la cadena de entrada a ser leído y se ha supuesto que luego de la cadena de entrada a procesar, se cuenta con un símbolo **\$** de fin de cadena.

Para el caso de que éste procedimiento de la Figura 5.28 se ejecute teniendo una cadena de **{a, b}*** para analizar en su entrada (terminada con un símbolo **\$**), el mismo podrá:

- Rechazar la cadena al no poder leer un símbolo esperado (Figura 5.28-3). Nótese que el llamado a leer un símbolo **a** luego de saber por el preanálisis que efectivamente existe una **a** en la entrada, no producirá nunca un error; pero no puede saberse si luego de re-ejecutar el procedimiento **S** y que éste termine, se tendrá un símbolo **b** a leer.
- Rechazar la cadena de entrada por contener un símbolo no perteneciente al alfabeto de entrada (Figura 5.28-2) o por terminación anticipada.
- Aceptar la cadena al detectar el símbolo de fin de cadena (Figura 5.28-1) luego de una cadena del formato requerido.

```
Procedimiento Descenso-Recursivo
  aceptada = "No"
  Si (preanálisis == '$')
    aceptada = "Si"
  Sino
    Ejecutar S
  Fin-Si
  Imprimir valor de aceptada
Fin-Procedimiento-Descenso-Recursivo

Procedimiento S
  Si (preanálisis == 'a')
    Ejecutar Leer-Símbolo('a')
    Ejecutar S
    Ejecutar Leer-Símbolo('b')
    Si (preanálisis == '$')
      aceptada = "Si"
    Fin-Si
  Sino
    Si (preanálisis == 'b')
      /* transición  $\lambda$  */
    Sino
      Informar Error y Rechazar
    Fin-Si
  Fin-Si
Fin-Procedimiento-S.

Procedimiento Leer-Símbolo(símbolo esperado)
  Si (preanálisis == símbolo esperado)
    Leer el siguiente símbolo de entrada
  Sino
    Informar Error y Rechazar
  Fin-Si
Fin-Procedimiento-Leer-Símbolo.
```

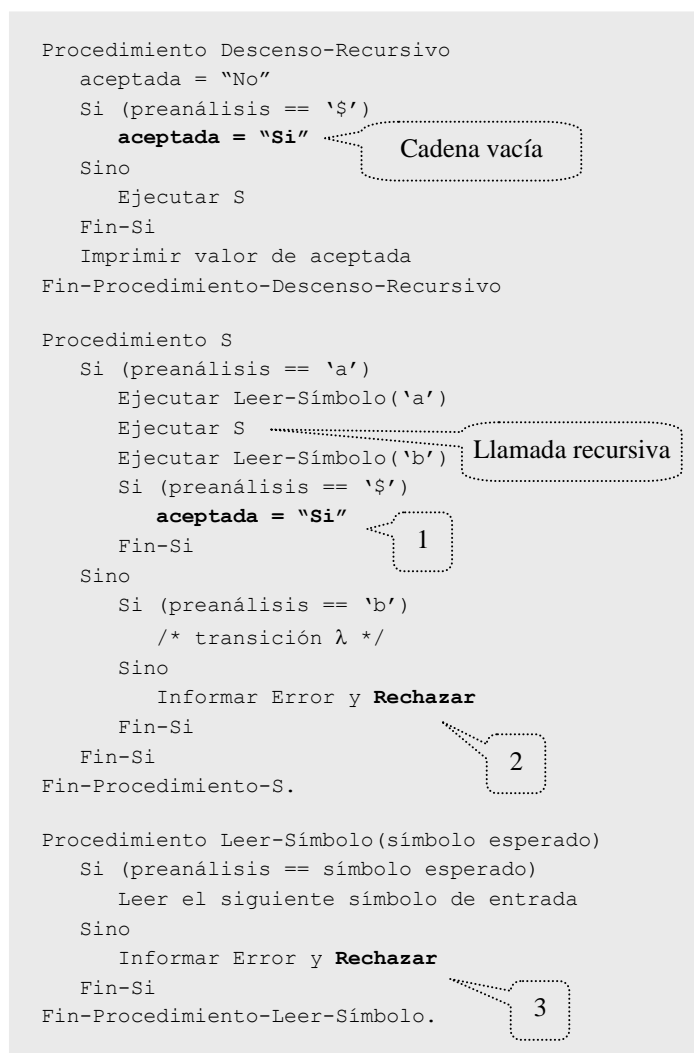


Figura 5.28: Procedimiento de análisis sintáctico predictivo por descenso recursivo para el Ejemplo 5.12.

Unidad 5: Autómatas con Pila

En la Tabla 5.13, se muestra la secuencia de acciones producidas por el algoritmo de la Figura 5.28, al operar sobre algunas cadenas de entrada.

Procedimiento	Preanálisis	Cadena a Leer	Acciones
DR	a	aabb\$	aceptada = "No"
DR	a	aabb\$	Ejecutar S
DR - S	a	aabb\$. Ejecutar Leer-Símbolo('a')
DR - S - LS	a	abb\$. . Leer símbolo de entrada
DR - S	a	abb\$. Ejecutar S
DR - S - S	a	abb\$. . Ejecutar Leer-Símbolo('a')
DR - S - S - LS	b	bb\$. . . Leer símbolo de entrada
DR - S - S	b	bb\$. . Ejecutar S
DR - S - S - S	b	bb\$. . . /* transición λ */
DR - S - S	b	bb\$. . Ejecutar Leer-Símbolo('b')
DR - S - S - LS	b	b\$. . . Leer símbolo de entrada
DR - S	b	b\$. Ejecutar Leer-Símbolo('b')
DR - S	\$	\$. . Leer símbolo de entrada
DR - S	\$	\$	aceptada = "Si"
DR	\$	\$	Imprimir aceptada \Rightarrow SI
Procedimiento	Preanálisis	Cadena a Leer	Acciones
DR	a	a\$	aceptada = "No"
DR	a	a\$	Ejecutar S
DR - S	a	a\$. Ejecutar Leer-Símbolo('a')
DR - S - LS	\$	\$. . Leer símbolo de entrada
DR - S	\$	\$. Ejecutar S
DR - S - S	\$	\$. . Error y Rechazar \Rightarrow NO
Procedimiento	Preanálisis	Cadena a Leer	Acciones
DR	b	b\$	aceptada = "No"
DR	b	b\$	Ejecutar S
DR - S	b	b\$. /* transición λ */
DR	b	b\$	Imprimir aceptada \Rightarrow NO

Tabla 5.13: Proceso de cadenas **aabb**, **a** y **b** con algoritmo de descenso recursivo.

Se pasará ahora a diseñar un analizador sintáctico predictivo no recursivo **LL(1)**, donde el número entre paréntesis indica el número de símbolos de preanálisis tenidos en cuenta. Para ello, debe construirse la tabla de decisión **M** que, asociada al AP de la Figura 5.26, permita su funcionamiento en forma determinista. Obviaremos aquí el procedimiento de construcción de **M**, pero apelando a los comentarios anteriores sabemos que si el símbolo de preanálisis es **a** debe usarse **S:=aSb** y si es **b** entonces lo aconsejado es usar **S:= λ** .

Así se construye la tabla de análisis sintáctico **M(Σ_N, Σ_T)**; esta tabla tiene como encabezados de sus filas a los no terminales de la gramática y de sus columnas a los símbolos terminales de la gramática y al símbolo **\$** de fin de cadena. En cada celda, se colocará una o ninguna producción, de acuerdo al anterior criterio.

M(Σ_N, Σ_T)	a	b	\$
S	S := aSb	S := λ	S := λ

Tabla 5.14: Tabla de análisis sintáctico LL(1) del Ejemplo 5.12.

Con esta tabla asociada al autómata con pila, el procedimiento de análisis de una cadena de entrada, puede verse en la Figura 5.29. El algoritmo se ha desarrollado en pseudocódigo y utiliza un procedimiento *Leer-Símbolos* idéntico al de la Figura 5.28.

Unidad 5: Autómatas con Pila

```
Procedimiento LL
  Crear pila P y Poner(#)
  Si (preanálisis ≠ '$')
    Poner(S)
  Mientras (preanálisis ≠ '$')
    Según sea cimaDePila:
      a|b) Si (cimaDePila == preanálisis)
        Leer-Símbolo(preanálisis)
        Sacar()
      Sino
        Informar Error y Rechazar
    Fin-Si
  S) Si (M[cimaDePila, preanálisis] no está vacía)
    Sacar()
    Poner(lado derecho de la producción de la celda)
  Sino
    Informar Error y Rechazar
  Fin-Si
  #) Informar Error y Rechazar
Fin-Mientras
Fin-Si
Si (cimaDePila == '#')
  Aceptar
Sino
  Informar Error y Rechazar
Fin-Si
Fin-Procedimiento-LL.
```

Figura 5.29: Algoritmo del analizador sintáctico LL(1) del Ejemplo 5.12.

Se ha utilizado una estructura de datos pila (**P**), una variable *cimaDePila* que siempre contiene el próximo símbolo a sacar de la pila y dos funciones usuales:

- Poner(α)**: es una función que inserta en la pila los símbolos de la cadena α (de terminales y no terminales), en orden inverso al presentado, de tal forma que el primer símbolo de la cadena quede en la cima de la pila.
- Sacar()**: es la función de extracción de un símbolo de la pila, que retira el símbolo en la cima de la pila.

Unidad 5: Autómatas con Pila

Preanálisis	Cadena a Leer	Pila	Cima de Pila	Acciones
a	aabb\$	λ	λ	Se crea la pila
a	aabb\$	#	λ	Poner(#)
a	aabb\$	S#	S	Poner(S)
a	aabb\$	S#	S	Sacar() y Poner(aSb)
a	aabb\$	aSb#	a	Leer(a) y Sacar()
a	abb\$	Sb#	S	Sacar() y Poner(aSb)
a	abb\$	aSbb#	a	Leer(a) y Sacar()
b	bb\$	Sbb#	S	Sacar() y Poner(λ)
b	bb\$	bb#	b	Leer(b) y Sacar()
b	b\$	b#	b	Leer(b) y Sacar()
\$	\$	#	#	Aceptar
Preanálisis	Cadena a Leer	Pila	Cima de Pila	Acciones
a	a\$	λ	λ	Se crea la pila
a	a\$	#	λ	Poner(#)
a	a\$	S#	S	Poner(S)
a	a\$	Sb#	S	Sacar() y Poner(aSb)
a	a\$	aSb#	a	Leer(a) y Sacar()
\$	\$	Sb#	S	Rechazar
Preanálisis	Cadena a Leer	Pila	Cima de Pila	Acciones
b	b\$	λ	λ	Se crea la pila
b	b\$	#	λ	Poner(#)
b	b\$	S#	S	Poner(S)
b	b\$	Sb#	S	Sacar() y Poner(aSb)
b	b\$	aSb#	a	Rechazar

Tabla 5.15: Proceso de cadenas **aabb**, **a** y **b** con algoritmo LL(1).

La tercera función usual en el manejo de estructuras LIFO, que verifica si la pila está vacía, se implementa como en el autómata con pila almacenando como primer símbolo un **#**, a modo de marca del fondo de pila.

El algoritmo al ser ejecutado con una cadena de símbolos del alfabeto de entrada **{a, b}**, terminada con el símbolo de fin de cadena **\$**, funciona exactamente como indica el **AP** que implementa la gramática del lenguaje $L_{(5.12)}$ y es guiado por la tabla **M**, para seleccionar las producciones que conviene utilizar en cada paso.

Ejemplo 5.13

Recuérdese ahora, la pequeña gramática de expresiones aritméticas propuesta en el Ejemplo 31 del Capítulo 2:

$$G = (\{ \text{num}, +, *, (,) \}, \{ E \}, E, \{ E := E + E \mid E * E \mid (E) \mid \text{num} \})$$

Como se discutió en su oportunidad, esta gramática es ambigua y tiene recursión por izquierda en el símbolo no terminal **E**. Se quiere desarrollar un algoritmo de análisis sintáctico descendente para ella, pero como se discutió oportunamente, una gramática con las características de **G**, no puede ser tratada por los algoritmos de descenso recursivo o **LL(1)**. Por ello, se inicia el estudio quitando la recursión por izquierda, para lo cual creamos un nuevo no terminal **X** y las nuevas producciones:

$$E := (E)X \mid \text{num}X; \quad X := +EX \mid *EX \mid \lambda$$

quedando la gramática en el formato:

$$G' = (\{ \text{num}, +, *, (,) \}, \{ E, X \}, E, \{ E := (E)X \mid \text{num}X, X := +EX \mid *EX \mid \lambda \})$$

Con la esperanza de que esta versión equivalente no sea ambigua, se intenta construir un analizador sintáctico **LL(1)** (predictivo no recursivo). Obviando nuevamente el procedimiento de generación de la tabla **M**, que está fuera del alcance de este libro, se obtiene la Tabla 5.16, donde se detectan celdas con más de una producción. Esto indica que la gramática no es **LL(1)** y que no puede armarse un analizador sintáctico descendente predictivo no recursivo para ella.

Unidad 5: Autómatas con Pila

Sin embargo, nótese que de cada par de reglas en las celdas $M(X,+)$ y $M(X,*)$, una de ellas es la regla no generativa $X:=\lambda$. Esta ambigüedad podría quitarse ya que, si se sabe que el símbolo de preanálisis es $+$ cuando en la cima de la pila se encuentra el no terminal X , no hay dudas de que aplicar la regla generativa $X:=+EX$ es lo adecuado, y lo mismo sucede en el caso de tener un símbolo de preanálisis $*$ respecto de la regla $X:=*EX$.

$M(\Sigma_N, \Sigma_T)$	num	+	*	()	\$
E	E:=numX			E:=(E)X		
X		X:=+EX X:= λ	X:=*EX X:= λ		X:= λ	X:= λ

Tabla 5.16: Tabla de análisis sintáctico no recursivo del Ejemplo 5.13.

Por ello, puede elegirse esta forma de quitar la ambigüedad y quedarse con la Tabla 5.17 como válida para los propósitos de análisis sintáctico de cadenas. Las celdas vacías indican como siempre, situaciones en las que habrá de anunciarse un error.

$M(\Sigma_N, \Sigma_T)$	num	+	*	()	\$
E	E:=numX			E:=(E)X		
X		X:=+EX	X:=*EX		X:=λ	X:=λ

Tabla 5.17: Tabla de análisis sintáctico (modificada) del Ejemplo 5.13.

////////////////////////////////////