

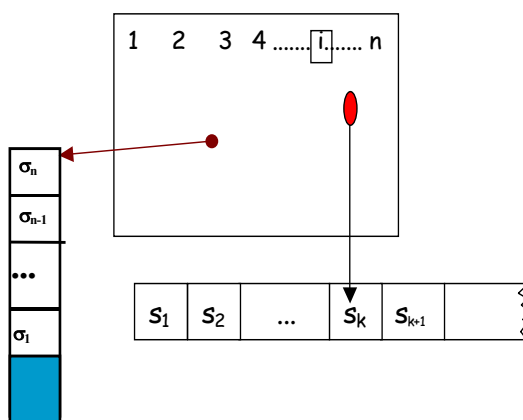
Capítulo 3

Autómatas de Pila y Gramáticas Independientes del Contexto

3.1 Introducción

En el Capítulo 2 se vieron los autómatas de estados finitos y su relación con las expresiones regulares. En este capítulo se verán los autómatas de pila y su relación con las gramáticas independientes del contexto. Los autómatas de pila tienen memoria representada en forma de una pila lo que les permite reconocer una clase de lenguajes más grande: los lenguajes independientes del contexto.

Un autómata de pila es una máquina con un número finito de estados, una cinta de entrada y una pila. El autómata lee cadenas de símbolos de longitud finita de la cinta de entrada que es infinita¹. El comportamiento de la máquina está determinado por el estado en que se encuentra, el símbolo o símbolos en el tope de la pila y el o los símbolos en la cinta de entrada. Para cada estado y dependiendo del tope de la pila, al leer uno o varios símbolos de la cinta de entrada cambia de estado, empila o desempila de la pila y avanza en la cinta de entrada. En la máquina de la figura de abajo, en el tiempo k está leyendo s_k y está en el estado i y la pila tiene la el símbolo σ_n en el tope. La cabeza lector únicamente puede avanzar.



Estas máquinas también pueden usarse para reconocer lenguajes. Es decir, para leer cadenas (secuencias de símbolos) y determinar si pertenecen o no a un lenguaje dado. También sirven para describir el comportamiento de ciertas máquinas.

¹ Es decir, las cadenas son de longitud finita, digamos n , y n puede ser tan grande como se quiera.

3.2 Autómatas de pila como reconocedores de lenguajes

La definición formal de un autómata de pila está dada en la Definición 3.1.

Definición 3.1. Un autómata de pila es una séxtupla $M=(Q, \Sigma, \Gamma, q_1, F, \Delta)$ donde:

- Q es un conjunto finito de estados
- Σ es un alfabeto de entrada finito
- Γ es un alfabeto de pila finito
- $q_1 \in Q$ es el estado inicial
- $F \subseteq Q$ es el conjunto de estados finales
- $\Delta \subseteq (Q \times \Sigma^* \times \Gamma^*) \times (Q \times \Gamma^*)$ es la relación de transición de estados


¿Cómo se usa un autómata de pilas para reconocer cadenas? El autómata comienza en el estado inicial con una secuencia de símbolos por leer. Inicialmente la pila está vacía. En cada instante lee símbolos de la cinta de entrada. Dependiendo de lo que lee, del tope de la pila y del estado en el que se encuentra, cambia de estado avanza en la cinta de entrada, desempila lo que reconoció en la pila y empile símbolos en la pila de acuerdo con la relación de transición de estados. Si puede aplicar sucesivamente estas transformaciones y llegar a un punto en el cual ha leído toda la cadena, está en un estado final y la pila está vacía, entonces se dice que el autómata acepta la cadena.

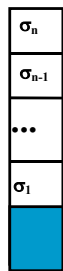
Las transiciones se interpretan así: si $((q, \beta, \phi), (p, \rho)) \in \Delta$ esto se lee así: "si está en el estado q , leyendo una cadena con prefijo β y con ϕ en el tope de la pila, avance sobre β , desempile ϕ , empile ρ y pase al estado p ".

- Al decir que ϕ está en el tope quiere decir que si: $\phi = \sigma_1 \sigma_2 \dots \sigma_n$ entonces la pila sería:



- Cuando ϕ es la cadena vacía, λ , se está diciendo que no importa qué está en la pila. NO QUIERE DECIR QUE LA PILA ESTÁ VACÍA. De hecho, la notación de transiciones no permite determinar cuando la pila está vacía. Para esto sería necesario usar un símbolo en el alfabeto de la pila que se use para marcar la base de la pila.

- Si $\rho = \sigma_1\sigma_2...\sigma_n$ al decir que se empila ρ sobre una pila: , la pila quedaría:



De manera análoga, si $\varphi = \lambda$ entonces se está indicando que no se debe empilar nada, NO QUIERE DECIR QUE EMPILE λ .

Si se mira la pila como una cadena, al decir que φ está en el tope de la pila, estamos diciendo que la pila es de la forma: $\mu\varphi$ para algún μ . Análogamente, al empilar ρ sobre una pila μ , la pila quedaría $\mu\rho$.

La transición descrita arriba, es la transición general. Hay transiciones más específicas descritas a continuación:

- $((q, \beta, \lambda), (p, \rho))$ si está en el estado q y la cadena que está leyendo tiene como prefijo β entonces (sin importar lo que está en el tope de la pila), lea β , empile ρ y pase al estado p .
- $((q, \beta, \rho), (p, \lambda))$ si está en el estado q y la cadena que está leyendo tiene como prefijo β y en el tope de la pila está ρ , entonces lea β , desempile ρ y pase al estado p (sin empilar nada).
- $((q, \beta, \lambda), (p, \lambda))$ si está en el estado q y la cadena que está leyendo tiene como prefijo β sin importar la pila, lea β , pase a p y no empile nada.
- $((q, \beta, \rho), (p, \rho\mu))$ si está en el estado q , lee β y ρ está en el tope de la pila, pase a p empilando μ sobre ρ .
- $((q, \beta, \rho), (p, \rho))$ si está en el estado q , lee β y ρ está en el tope de la pila, pase a p dejando ρ en el tope de la pila.

Note que en las últimas dos transiciones debemos volver a empilar ρ ya que al reconocerlo, se desempila. Si en cualquiera de las transiciones mencionadas arriba, β es λ , esto indica que se toma la acción correspondiente sin avanzar sobre la cinta de entrada.

El estado general de un autómata de pilas está dado por: el estado en que se encuentra, lo que le falta por leer y la pila. Formalmente, esto se define como configuración:

Definición 3.2. La configuración de un autómata de pilas es una tripleta, (q, ρ, ω) , con $q \in Q$, $\rho \in \Gamma^*$, $\omega \in \Sigma^*$ y donde: q es el estado en el que se encuentra, ρ es la pila y ω es lo que le falta por leer.

Interesa definir como pasar de una configuración a otra.

Definición 3.3. Dadas dos configuraciones: (q', ρ', ω') y (q, ρ, ω) si:

- $\omega = \beta\omega'$
- $\rho = \varphi\mu, \rho' = \varphi\eta$ y
- $((q, \beta, \mu), (q', \eta)) \in \Delta$

Se dice que (q', ω', ρ') es alcanzable en un paso a partir de (q, ω, ρ) y escribimos:

$$(q, \omega, \rho) \Rightarrow (q', \omega', \rho').$$

Es decir: Para todo $\omega' \in \Sigma^*, \varphi \in \Gamma^*, ((q, \beta, \mu), (q', \eta)) \in \Delta \equiv (q, \beta\omega', \varphi\mu) \Rightarrow (q', \omega', \varphi\eta)$

La clausura transitiva de esta relación da el concepto de “ser alcanzable desde” entre configuraciones.

Definición 3.4. Dadas dos configuraciones C y K decimos que K es alcanzable desde C y escribimos $C \Rightarrow^* K$ si ocurre alguna de las siguientes condiciones:

- $C = K$
- $C \Rightarrow K$
- Existe una configuración C' tal que $C \Rightarrow C'$ y $C' \Rightarrow^* K$

Con la definición de alcanzable, se define cuando un autómata acepta una cadena. Informalmente, una cadena es aceptada por un autómata de pilas si comienza con la cadena por leer en el estado inicial con la pila vacía y puede llegar a una configuración con un estado final, sin nada por leer y con la pila vacía.

Definición 3.5. Dado $M=(Q, \Sigma, \Gamma, q_i, F, \Delta)$ y $\omega \in \Sigma^*$, se dice que M acepta a ω , si existe un estado final f ($f \in F$) tal que $(q_i, \omega, \lambda) \Rightarrow^* (f, \lambda, \lambda)$.

Note que no es suficiente terminar en un estado final; hay que terminar en un estado final habiendo leído todo y con la pila vacía. Análogamente a que con los autómatas finitos, podemos definir el lenguaje reconocido por un autómata de pilas, como el conjunto de todas las cadenas que son aceptadas por dicho autómata.

Con el concepto de aceptar una cadena, se define el lenguaje aceptado por un autómata.

Definición 3.6. Dado $M=(Q, \Sigma, \Gamma, q_i, F, \Delta)$ y definimos el lenguaje aceptado por M , denotado como $L(M)$ así:
 $L(M) = \{\omega: \Sigma^* \mid \exists f \in F \text{ tal que } (q_i, \omega, \lambda) \Rightarrow^* (f, \lambda, \lambda)\}$

Los autómatas de pilas tienen más poder computacional que los autómatas de estados finitos. Estos pueden reconocer lenguajes que no pueden ser reconocidos por los autómatas finitos. Por ejemplo pueden reconocer el lenguaje : $\{a^n b^n: n \geq 0\}$.

Hay autómatas determinísticos y no-determinísticos.

Definición 3.7. Un *autómata de pila determinístico* es un *autómata de pila* para toda configuración (q, ρ, ω) existe a lo más una configuración (q', ρ', ω') tal que $(q, \rho, \omega) \Rightarrow (q', \rho', \omega')$.

A diferencia de lo que ocurre con los autómatas regulares, los autómatas de pila no determinísticos no tienen el mismo poder computacional que los determinísticos. Los ejemplos mostrados en este capítulo usan tanto autómatas determinísticos como no determinísticos.

Ejemplo 3.1 El autómata determinístico $M = (\{1, 2\}, \{a, b\}, \{a\}, 1, \{1, 2\}, \{((1, a, \lambda), (1, a)), ((1, b, a), (2, \lambda)), ((2, b, a), (2, \lambda))\})$ reconoce $\{a^n b^n : n \geq 0\}$.

A continuación se muestra la ejecución de la máquina para algunas cadenas de $\{a, b\}^*$.

aaabbb

Estado	Pila	Cadena	Acción
1	λ	aaabbb	aplicar $((1, a, \lambda), (1, a))$
1	a	aabbb	aplicar $((1, a, \lambda), (1, a))$
1	aa	abbb	aplicar $((1, a, \lambda), (1, a))$
1	aaa	bbb	aplicar $((1, b, a), (2, \lambda))$
2	aa	bb	aplicar $((2, b, a), (2, \lambda))$
2	a	b	aplicar $((2, b, a), (2, \lambda))$
2	λ	λ	acepta: 2 es final

ab

Estado	Pila	Cadena	Acción
1	λ	ab	aplicar $((1, a, \lambda), (1, a))$
1	a	b	aplicar $((1, a, \lambda), (1, a))$
2	λ	λ	Acepta: 2 es final

λ

Estado	Pila	Cadena	Acción
1	λ	λ	Acepta: 1 es final

bbb

Estado	Pila	Cadena	Acción
1	λ	bbb	Falla: no hay regla que aplicar y queda por leer bbb

aaabbbb

Estado	Pila	Cadena	Acción
1	λ	aaabbbb	aplicar $((1, a, \lambda), (1, a))$
1	a	aabbbb	aplicar $((1, a, \lambda), (1, a))$
1	aa	abbbb	aplicar $((1, a, \lambda), (1, a))$
1	aaa	bbbb	aplicar $((1, b, a), (2, \lambda))$
2	aa	bb	aplicar $((2, b, a), (2, \lambda))$
2	a	b	aplicar $((2, b, a), (2, \lambda))$
2	λ	b	falla: no hay regla que aplicar y falta por leer b

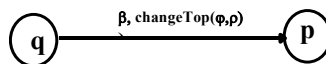
aaabbb

Estado	Pila	Cadena	Acción
1	λ	aaabbb	aplicar((1,a, λ),(1,a))
1	a	aabbb	aplicar((1,a, λ),(1,a))
1	aa	abbb	aplicar((1,a, λ),(1,a))
1	aaa	bbb	aplicar((1,b,a),(2, λ))
2	aa	bb	aplicar((2,b,a),(2, λ))
2	a	λ	Falla: pues leyó todo, pero la pila no está vacía.

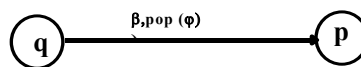
La notación mostrada para describir autómatas puede resultar un poco engorrosa. Explicamos una notación diagramática para describir estos autómatas,

Los estados, el estado inicial y los estados finales se representan igual que en los autómatas de estados finitos. Lo que varía son las transiciones. Estas se denotan como se muestra abajo.

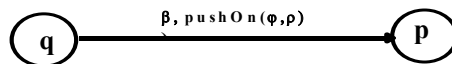
- $((q, \beta, \varphi), (p, \rho)) \quad \varphi \neq \lambda, \rho \neq \lambda$ (Si reconoce φ en el tope, desempile φ y empile ρ)



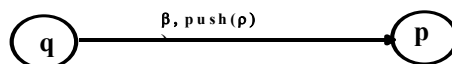
- $((q, \beta, \varphi), (p, \lambda)) \quad \varphi \neq \lambda$ (Si reconoce φ en el tope, desempile φ)



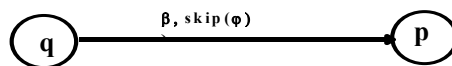
- $((q, \beta, \varphi), (p, \rho\rho)) \quad \varphi \neq \lambda, \rho \neq \lambda$ (Si reconoce φ en el tope empile ρ sobre φ)



- $((q, \beta, \lambda), (p, \rho)) \quad \rho \neq \lambda$ (empile ρ sin importar que hay en la pila)



- $((q, \beta, \varphi), (p, \varphi)) \quad \varphi \neq \lambda$ (Si reconoce φ en el tope déjelo en el tope)



- $((q, \beta, \lambda), (p, \lambda))$ (Ignore la pila)



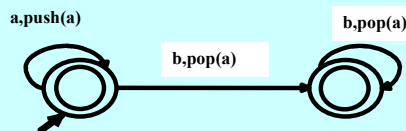
Estas transiciones las abreviaremos como se muestra abajo, cuando se quiera hacer referencia a ellas en el texto y no en un dibujo de un autómata.

- $q \xrightarrow{\beta, \text{changeTop}(\varphi, \rho)} p$
- $q \xrightarrow{\beta, \text{pop}(\varphi)} p$
- $q \xrightarrow{\beta, \text{pushOn}(\varphi, \rho)} p$
- $q \xrightarrow{\beta, \text{push}(\rho)} p$
- $q \xrightarrow{\beta, \text{skip}(\varphi)} p$
- $q \xrightarrow{\beta, \text{ignore}} p$

EJERCICIOS

1. Verifique que todas las transiciones más comunes descritas arriba pueden ser descritas por medio de las instrucciones **changeTop**, **pop**, **pushOn**, **push**, **skip**, e **ignore**,
2. Verifique que las instrucciones **ignore** y **pushOn** son innecesarias podrían simularse con otras instrucciones. Ayuda: Debe reemplazar cada instrucción por $|\Gamma|$ instrucciones.
3. ¿Se puede Podríamos hacer modelar todo con una sola instrucción eliminando las restricciones: $\varphi \neq \lambda$, $\rho \neq \lambda$? Justifique su respuesta.

Ejemplo 3.1a El ejemplo mostrado arriba: $\{a^n b^n : n \geq 0\}$ y $M = (\{1, 2\}, \{a, b\}, \{a\}, 1, \{1, 2\}, \{((1, a, \lambda), (1, a)), ((1, b, a), (2, \lambda)), ((2, b, a), (2, \lambda))\})$ se dibujaría así:

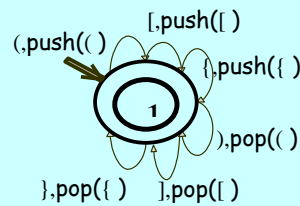


A continuación se muestran más ejemplos:

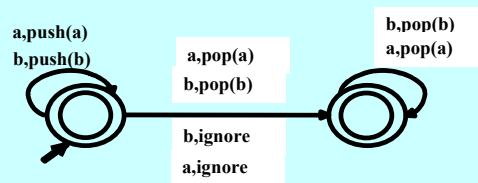
Ejemplo 3.2 Para el lenguaje $\{a^{3n} b^n : n \geq 0\}$ el autómata determinístico $M = (\{1, 2\}, \{a, b\}, \{a\}, 1, \{1, 2\}, \{((1, a, \lambda), (1, a)), ((1, b, aaa), (2, \lambda)), ((2, b, aaa), (2, \lambda))\})$ reconoce el lenguaje y se dibujaría así:



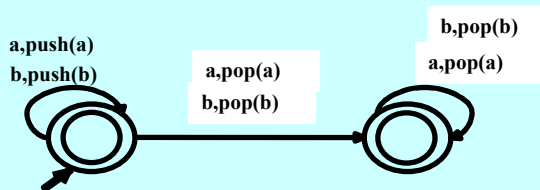
Ejemplo 3.3 El siguiente autómata determinístico reconoce las cadenas sobre $\{ \{, \}, [,], (,) \}$ en los que los paréntesis están balanceados.



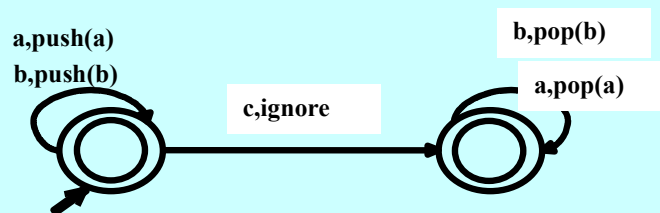
Ejemplo 3.4 El siguiente autómata no-determinístico reconoce las cadenas sobre $\{a, b\}$ que son palíndromos.



Ejemplo 3.5 El siguiente autómata no-determinístico reconoce las cadenas sobre $\{a, b\}$ que son palíndromos de longitud par.



Ejemplo 3.6 El siguiente autómata determinístico reconoce el lenguaje $\{\omega c \omega^R : \omega \in \{a, b\}^*\}$



EJERCICIOS

1. Muestre el seguimiento de los autómatas de arriba para una cadena del lenguaje de longitud 4 y una cadena que no está en el lenguaje de longitud 5.
2. Describa autómatas de pilas (determinísticos o no-determinísticos) para reconocer los siguientes lenguajes:
 - a. $\{a^n b^n : n > 0\}$
 - b. $\{a^n b^m : n \geq m \geq 0\}$

- c. $\{ a^n b^m : n \geq m > 0 \}$
 - d. $\{ a^n b^m : n > m \geq 0 \}$
 - e. $\{ a^n b^m : n > m > 0 \}$
 - f. $\{ a^n b^m : m \geq n \geq 0 \}$
 - g. $\{ a^n b^m : m \geq n > 0 \}$
 - h. $\{ a^n b^m : m > n \geq 0 \}$
 - i. $\{ a^n b^m : m > n > 0 \}$
 - j. $\{ a^n b^n : n \geq 0 \} \cup \{ a^n b^m : n \geq m \geq 0 \} \cup \{ a^n b^m : m \geq n \geq 0 \}$
3. *** Enriquezca la definición de autómeta de pilas para que de respuestas:
- a. en los estados
 - b. en las transiciones
4. Describa un autómeta de pila que reconozca el lenguaje de las palabras palíndromes sobre el alfabeto: $\{a, A, b, B\}$ donde no se toman en cuenta las diferencias por mayúscula y minúscula. Ejemplos de palabras en este lenguaje serían: aaAAbBbaaaa aaAA abbBbA
5. Describa un autómeta de pilas que reconozca el siguiente lenguaje:
- $$L = \{a\} \cup \{ (\omega) : \omega \in L \} \cup \{ \omega + \omega : \omega \in L \}$$
6. Describa un autómeta de pilas que reconozca el siguiente lenguaje:
- $$L = \{a\} \cup \{ f(\omega_1, \dots, \omega_n) : n > 0, 0 \leq i \leq n, \omega_i \in L \}.$$
- Ejemplos de palabras en este lenguaje a f(a) f(a,a,a) f(f(a,a),f(a),a)
7. Modifique el autómeta que reconoce paréntesis balanceados para que dentro de los paréntesis siempre deba haber o secuencia de a's separadas por comas o secuencias de expresiones de paréntesis y a's separadas por comas.
- Ejemplos de palabras en este lenguaje: a (a,a,a) ([a,{(a,a),a}],a)

3.3 Autómatas y Gramáticas

Los autómatas pila reconocen la misma clase de lenguajes generados por las gramáticas independientes del contexto. En esta sección veremos algoritmos como dada una gramática, podemos construir un autómeta de pilas que reconozca el lenguaje generado por la gramática. También veremos como construir una gramática que genere el lenguaje reconocido por un autómeta dado.

Se recomienda repasar las definiciones de gramáticas dadas en el Capítulo 1 antes de continuar leyendo esta sección.

3.3.1 De Autómatas a Gramáticas

Primero se da la definición de un autómata de pilas simplificado. En este tipo de autómata de pilas simplificado es aquel en que las instrucciones pueden ser solamente de una de las siguientes formas:

$$p \xrightarrow{a, \text{push}(c)} r$$

$$p \xrightarrow{a, \text{pop}(c)} r$$

Con $a \in \Sigma \cup \{ \lambda \}$ y $c \in \Gamma$

Formalmente esto se define así:

Definición 3.8. Un autómata de pila simplificado es una séxtupla $M=(Q, \Sigma, \Gamma, q_i, q_f, \Delta)$ donde :

- Q es un conjunto finito de estados
- Σ es un alfabeto de entrada finito
- Γ es un alfabeto de pila
- $q_i \in Q$ es el estado inicial
- $q_f \in Q$ es el estado final
- $\Delta \subseteq (Q \times (\Sigma \cup \{ \lambda \}) \times (\Gamma \cup \{ \lambda \})) \times (Q \times (\Gamma \cup \{ \lambda \}))$ es la relación de transición de estados, y se cumple que:
 $((q, a, b), (p, c)) \in \Delta \Rightarrow (b = \lambda \equiv c \neq \lambda)$

Note que los autómatas simplificados no necesariamente son determinísticos. La clase de los lenguajes reconocidos por los autómatas de pila simplificados es exactamente igual a la clase de lenguajes reconocidos por los autómatas de pila. Para demostrar esto habría que demostrar que dado un autómata de pilas es posible construir un autómata de pilas simplificado y viceversa. Es fácil ver que todo autómata simplificado es un autómata de pilas. La recíproca se deja como ejercicio. Ayuda: hay que agregar estados.

Dado un autómata de pilas simplificado (no necesariamente determinístico): $M=(Q, \Sigma, \Gamma, s, f, \Delta)$ se puede construir una gramática $G=(N, T, S, P)$ tal que $L(M)=L(G)$ de la siguiente forma:

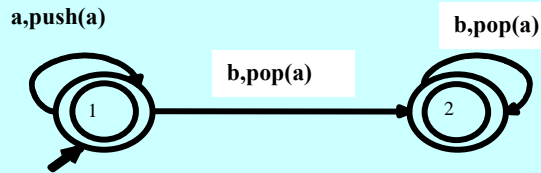
1. Hay un nóterminal por cada elemento en $Q \times Q$ para facilidad de expresión se definen así: $N = \{A_{pq} \mid p \in Q, q \in Q\}$
2. Los terminales son el alfabeto de entrada: $T=\Sigma$
3. El símbolo distinguido, S es: A_{sf}
4. La lista de producciones se construye así:
 - a. Por cada $q \in Q$ agregue la producción: $A_{qq} \rightarrow \lambda$:
 - b. Por cada $p \in Q, q \in Q, r \in Q$ agregue la producción: $A_{pq} \rightarrow A_{pr} A_{rq}$
 - c. Si se tiene que $p \xrightarrow{a, \text{push}(c)} r$ y además que $t \xrightarrow{b, \text{pop}(c)} q$ agregue la producción: $A_{pq} \rightarrow a A_{rt} b$

Es decir:

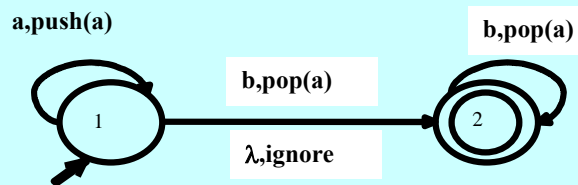
$$P = \{A_{pq} \rightarrow aA_r b : a, b \in \Sigma \cup \{\lambda\}, ((p, a, \lambda), (r, c)) \in \Delta, ((t, b, c), (q, \lambda)) \in \Delta\} \cup \\ \{A_{pq} \rightarrow A_{pr} A_{rq} : p \in Q, q \in Q, r \in Q\} \cup \\ \{A_{qq} \rightarrow \lambda : q \in Q\}$$

Es posible demostrar que para cualquier autómata M , al generar la gramática con estas reglas, $L(G) = L(M)$. Esto, sin embargo, queda fuera del propósito de estas notas.

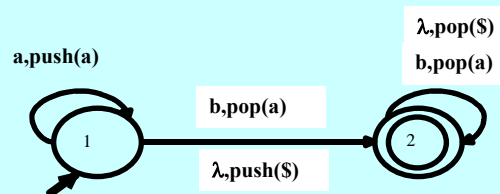
Ejemplo 3.7 Recuerde el autómata de arriba que reconoce $\{a^n b^n : n \geq 0\}$



Aún cuando no las reglas sí son de la forma correcta, se tienen 2 estados finales. Se arregla el autómata así:



El problema es que toda transición debe empujar o desempilar. Entonces se hace lo siguiente: Se agrega un símbolo al alfabeto de la pila, que es empujado al pasar del estado 1 al 2. Este debe ser desempilado.



Dadas las reglas, se pueden construir fácilmente los conjuntos de terminales, de los no-terminales y se puede determinar cuál es el símbolo distinguido.

- $N = \{A_{11}, A_{12}, A_{21}, A_{22}\}$
- $T = \{a, b\}$
- $S = A_{12}$

Ahora se empiezan a aplicar las reglas para obtener las producciones. De la regla (a) se obtienen las siguientes reglas:

- $A_{11} \rightarrow \lambda$

- $A_{22} \rightarrow \lambda$

Aplicando la regla (b), se agregan las siguientes producciones:

p	q	r	Produccion agregada
			$A_{pq} \rightarrow A_{pr} A_{rq}$
1	1	1	$A_{11} \rightarrow A_{11} A_{11}$
1	1	2	$A_{11} \rightarrow A_{12} A_{21}$
1	2	1	$A_{12} \rightarrow A_{11} A_{12}$
1	2	2	$A_{12} \rightarrow A_{12} A_{22}$
2	1	1	$A_{21} \rightarrow A_{21} A_{11}$
2	1	2	$A_{21} \rightarrow A_{22} A_{21}$
2	2	1	$A_{22} \rightarrow A_{21} A_{12}$
2	2	2	$A_{22} \rightarrow A_{22} A_{22}$

Ahora la regla 3:

$[1] \xrightarrow{a, \text{push}(a)} [1]$ y además que $[1] \xrightarrow{b, \text{pop}(a)} [2]$, entonces se agrega $A_{12} \rightarrow aA_{11}b$

$[1] \xrightarrow{a, \text{push}(a)} [1]$ y además que $[2] \xrightarrow{b, \text{pop}(a)} [2]$, entonces se agrega: $A_{12} \rightarrow aA_{12}b$

$[1] \xrightarrow{\lambda, \text{push}(\$)} [2]$ y además que $[2] \xrightarrow{\lambda, \text{pop}(\$)} [2]$, entonces se agrega: $A_{12} \rightarrow A_{22}$

Finalmente, la producciones serían las siguientes:

$A_{11} \rightarrow \lambda$	$A_{12} \rightarrow aA_{11}b$
$A_{11} \rightarrow A_{11} A_{11}$	$A_{12} \rightarrow aA_{12}b$
$A_{11} \rightarrow A_{12} A_{21}$	$A_{21} \rightarrow A_{21} A_{11}$
$A_{12} \rightarrow A_{11} A_{12}$	$A_{21} \rightarrow A_{22} A_{21}$
$A_{12} \rightarrow A_{12} A_{22}$	$A_{22} \rightarrow A_{21} A_{12}$
$A_{12} \rightarrow A_{22}$	$A_{22} \rightarrow A_{22} A_{22}$
	$A_{22} \rightarrow \lambda$

Por inspección podemos simplificar esta gramática.

Se ve que el símbolo A_{21} es inútil, ya que todas las reglas que lo tienen en la parte izquierda lo tienen también en la parte derecha. Esto hace que una vez se genera este símbolo, nunca puede eliminarse. Por lo tanto, se eliminan todas la reglas que tienen este símbolo dejando la gramática así:

$$A_{11} \rightarrow \lambda$$

$$A_{11} \rightarrow A_{11} A_{11}$$

$$A_{12} \rightarrow A_{11} A_{12}$$

$$A_{12} \rightarrow A_{12} A_{22}$$

$$A_{12} \rightarrow A_{22}$$

$$A_{12} \rightarrow aA_{11}b$$

$$A_{12} \rightarrow aA_{12}b$$

$$A_{22} \rightarrow A_{22} A_{22}$$

$$A_{22} \rightarrow \lambda$$

Ahora nos damos cuenta que los símbolos A_{22} y A_{11} sólo puede generar λ . Podemos entonces cambiar estos símbolos por λ cuando aparecen en la parte derecha de una regla y eliminar las reglas que los tienen en la parte izquierda dejando la gramática así:

$$A_{12} \rightarrow A_{12}$$

$$A_{12} \rightarrow A_{12}$$

$$A_{12} \rightarrow \lambda$$

$$A_{12} \rightarrow ab$$

$$A_{12} \rightarrow aA_{12}b$$

$$A_{22} \rightarrow \lambda$$

$$A_{22} \rightarrow \lambda$$

Las primeras dos reglas son inútiles y la cuarta y la quinta nunca se usarán debido a que A_{22} nunca aparece en la parte derecha de una regla. Entonces, la gramática queda:

$$A_{12} \rightarrow \lambda$$

$$A_{12} \rightarrow ab$$

$$A_{12} \rightarrow aA_{12}b$$

Ahora se mostrara que si una palabra está en el lenguaje, entonces es generada por la gramática. Faltaría demostrar que si no está en el lenguaje reconocido por el autómata entonces no es generada por la gramática. Esta demostración se omite.

La demostración se hace inductivamente. Primero se muestra que se puede generar λ y que se puede generar ab ; luego se muestra que si se puede generar $a^k b^k$ entonces se puede generar $a^{k+1} b^{k+1}$.

Recuerde que demostrar que la gramática genera una cadena, ω , es demostrar que $S \Rightarrow^* \omega$. Como el símbolo distinguido de esta gramática es A_{12} , en este caso se debe demostrar que $A_{12} \Rightarrow^* \omega^2$.

1. $A_{12} \Rightarrow^* \lambda$

A_{12}
 \Rightarrow aplicamos $A_{12} \rightarrow \lambda$
 λ

2. $A_{12} \Rightarrow^* ab$

A_{12}
 \Rightarrow aplicamos $A_{12} \rightarrow ab$
 ab

3. Si $A_{12} \Rightarrow^* a^k b^k$ entonces $A_{12} \Rightarrow^* a^{k+1} b^{k+1}$.

Hipótesis de inducción: $A_{12} \Rightarrow^* a^k b^k$

A_{12}
 \Rightarrow aplicamos $A_{12} \rightarrow aA_{12}b$
 $a A_{12} b$
 \Rightarrow^* hipótesis de inducción
 $aa^k b^k b$
 $=$ notación de cadenas
 $a^{k+1} b^{k+1}$

3.3.2 De Gramáticas a Autómatas

En esta sección se muestra como construir autómatas no determinísticos que reconocen el lenguaje generado por una gramática dada. Se construyen dos tipos de autómatas uno para hacer análisis descendente y otro para hacer análisis ascendente. En el análisis descendente comenzamos con el símbolo distinguido para generar las cadenas del lenguaje, en el análisis ascendente se parte de la cadena para llegar al símbolo distinguido. Viéndolo en términos del árbol de sintaxis en el análisis descendente comenzamos por la raíz y llegamos a las hojas. En el ascendente, comenzamos por las hojas para llegar a la raíz.

El proceso de construcción de los autómatas se hará a través de un ejemplo.

La siguiente gramática reconoce el lenguaje $\{a^n b^n : n \geq 0\}$: $(\{A\}, \{a, b\}, A, \{A \rightarrow \lambda, A \rightarrow aAb\})$. El conjunto de terminales es $\{a, b\}$, el de no-terminales en $\{A\}$, el símbolo distinguido es A y las producciones son las dos producciones de arriba.

² Note que en este caso el símbolo \Rightarrow no se está usando como implicación lógica; se usa como el concepto de “ser derivable en un paso”, visto en el Capítulo 1.

Autómata descendente:

El autómata descendente correspondiente a esta gramática se construye así:

- El autómata tiene tres estados: **s**, **q** y **f**, donde **s** es el estado inicial y **f** es el único estado final. En el estado **q**, se realizaran todas las acciones necesarias para reconocer el lenguaje.
- El alfabeto de entrada es el conjunto de símbolos terminales.
- El alfabeto de pila es la unión de los terminales y los no terminales agregando un símbolo nuevo $\$$ ³. Este símbolo se usará para marcar la base de la pila.
- Ahora las transiciones: la idea es que este autómata comience con el símbolo distinguido en la pila y aplique reglas de producción para reconocer las cadenas del lenguaje. Por o tanto:

- para comenzar ponemos el símbolo distinguido en la pila. Agregamos una transición del estado inicial a **q**, y se empila $\$$ y el símbolo distinguido:

$$s \xrightarrow{\lambda, \text{push}(\$A)} q$$

- Como el autómata acepta con la pila vacía, lo último que se debe hacer es desempilar la marca de base de pila. Agregamos, entonces, una transición de **q** el estado interno al estado final **f** donde se desempila el símbolo $\$$.

$$q \xrightarrow{\lambda, \text{pop}(\$)} f$$

- Ahora por cada regla de producción: $L \rightarrow \lambda$ agregar la regla: $q \xrightarrow{\lambda, \text{pop}(L)} q$ En este caso particular se agregaría la siguiente transición.

$$q \xrightarrow{\lambda, \text{pop}(A)} q$$

- Por cada regla de producción: $L \rightarrow \omega$ con $\omega \neq \lambda$ agregar la regla: $q \xrightarrow{\lambda, \text{ChangeTop}(L, \omega^R)} q$ En este caso particular se agregaría la siguiente transición:

$$q \xrightarrow{\lambda, \text{changeTop}(A, bAa)} q$$
⁴

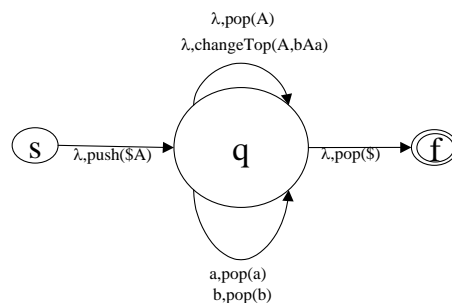
- Finalmente, la parte de reconocer: por cada símbolo **t** de los terminales, Σ , agregamos la siguiente transición: $q \xrightarrow{t, \text{pop}(t)} q$. En este caso particular se agregarían las siguientes transiciones.

$$q \xrightarrow{a, \text{pop}(a)} q \text{ y } q \xrightarrow{b, \text{pop}(b)} q$$

³ que es un símbolo que no existe en los terminales ni en los no terminales

⁴ Note que la regla era $A \rightarrow aAb$ y empilamos bAa porque se empila el reverso de la parte derecha de la regla.

El autómata quedaría así:



Su funcionamiento para dos ejemplos (una cadena en el lenguaje y una que no está en el lenguaje se define así)

aaabbb

Estado	Pila	Cadena	Acción
s	λ	aaabbb	λ, push(\$A) → q
q	\$A	aaabbb	λ, changeTop(A, bAa) → q
q	\$bAa	aaabbb	a, pop(a) → q
q	\$bA	aabbb	λ, changeTop(A, bAa) → q
q	\$bbAa	aabbb	a, pop(a) → q
q	\$bbA	abbb	λ, changeTop(A, bAa) → q
q	\$bbbAa	abbb	a, pop(a) → q
q	\$bbbA	bbb	λ, pop(A) → q
q	\$bbb	bbb	b, pop(b) → q
q	\$bb	bb	b, pop(b) → q
q	\$b	b	b, pop(b) → q
q	\$	λ	λ, pop(\$) → f
f	λ	λ	ACEPTA

aaabb

Estado	Pila	Cadena	Acción
s	λ	aaabb	λ, push(\$A) → q
q	\$A	aaabb	λ, changeTop(A, bAa) → q
q	\$bAa	aaabb	a, pop(a) → q
q	\$bA	aabb	λ, changeTop(A, bAa) → q
q	\$bbAa	aabb	a, pop(a) → q
q	\$bbA	abb	λ, changeTop(A, bAa) → q
q	\$bbbAa	abb	a, pop(a) → q
q	\$bbbA	bb	λ, pop(A) → q
q	\$bbb	bb	b, pop(b) → q
q	\$bb	b	b, pop(b) → q
q	\$b	λ	Falla: no hay reglas aplicables

El proceso descrito arriba para construir el autómata descendente a partir de una gramática independiente del contexto se define formalmente a continuación.

Dada una gramática $G=(N, \Sigma, S, P)$ construimos el autómata que reconoce $L(G)$ así:

$M = (\{s, q, f\}, \Sigma, N \cup \Sigma \cup \{\$, s, f\}, \Delta)$ con:

$$\Delta = \{(s, \lambda, \lambda), (q, \$S)\} \cup \{(q, \lambda, \$), (f, \lambda)\} \cup \{(q, \sigma, \sigma), (q, \lambda): \sigma \in \Sigma\} \cup \{(q, \lambda, A), (q, \omega^R): A \rightarrow \omega \in \Sigma\}.$$

El autómata generado reconoce la palabras en $L(G)$ usando análisis descendente.

Autómatas ascendentes

El autómata descendente correspondiente a esta gramática se construye así:

- El autómata también tiene tres estados: **s**, **q** y **f**, donde **s** es el estado inicial y **f** es el único estado final. En el estado **q**, se realizaran todas las acciones necesarias para reconocer el lenguaje.
- El alfabeto de entrada es el conjunto de símbolos terminales.
- El alfabeto de pila es la unión de los terminales y los no terminales agregando un símbolo nuevo $\$$ ⁵. Este símbolo se usará para marcar la base de la pila.
- Ahora las transiciones: la idea es que de este autómata comience trate de obtener el símbolo distinguido. Por lo tanto:
 - Si en la pila está $\$S$ en el tope donde **S** es símbolo distinguido, pasamos a **f** y aceptamos (siempre y cuando, claro, se haya terminado de leer la cadena). Agregamos, entonces, una transición de **q** el estado interno al estado final **f** donde se desempila $\$S$.

$$q \xrightarrow{\lambda, \text{pop}(\$S)} f$$

- Para comenzar ponemos el símbolo $\$$ en la pila. Agregamos una transición del estado inicial a **q**, y se empila $\$$:

$$s \xrightarrow{\lambda, \text{push}(\$)} q$$

- Ahora por cada regla de producción: $L \rightarrow \lambda$ agregar la regla: $q \xrightarrow{\lambda, \text{push}(L)} q$: Lo que esto quiere decir es que λ puede ser **L**. En este caso particular se agregaría la siguiente transición.

$$q \xrightarrow{\lambda, \text{push}(A)} q$$

- Por cada regla de producción: $L \rightarrow \omega$ con $\omega \neq \lambda$ agregar la regla: $q \xrightarrow{\lambda, \text{ChangeTop}(\omega, L)} q$ En este caso particular se agregaría la siguiente transición:

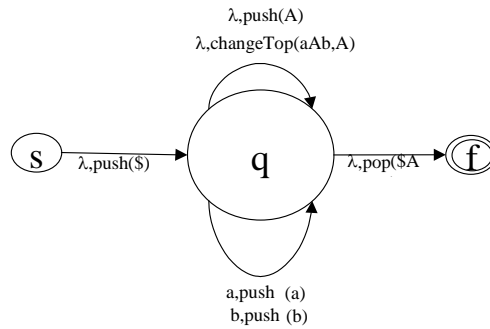
$$q \xrightarrow{\lambda, \text{changeTop}(aAb, A)} q$$

⁵ que es un símbolo que no existe en los terminales ni en los no terminales

- Finalmente, la parte de leer: por cada símbolo t de los terminales, Σ , agregamos la siguiente transición: $q \xrightarrow{t, \text{push}(t)} q$. En este caso particular se agregarían las siguientes transiciones.

$$q \xrightarrow{a, \text{push}(a)} q \text{ y } q \xrightarrow{b, \text{push}(b)} q$$

El autómata quedaría así:



Su funcionamiento para los dos mismos ejemplos de la sección anterior se muestra a continuación.

aaabbb

Estado	Pila	Cadena	Acción
s	λ	aaabbb	$\lambda, \text{push}(\$) \rightarrow q$
q	\$	aaabbb	$a, \text{push}(a) \rightarrow q$
q	\$a	aabbb	$a, \text{push}(a) \rightarrow q$
q	\$aa	abbb	$a, \text{push}(a) \rightarrow q$
q	\$aaa	bbb	$\lambda, \text{push}(A) \rightarrow q$
q	\$aaaA	bbb	$b, \text{push}(b) \rightarrow q$
q	\$aaaAb	bb	$\lambda, \text{changeTop}(aAb, A) \rightarrow q$
q	\$aaA	bb	$b, \text{push}(b) \rightarrow q$
q	\$aaAb	b	$\lambda, \text{changeTop}(aAb, A) \rightarrow q$
q	\$aA	b	$b, \text{push}(b) \rightarrow q$
q	\$aAb	b	$\lambda, \text{changeTop}(aAb, A) \rightarrow q$
q	\$A	λ	$\lambda, \text{pop}(\$A) \rightarrow f$
f	λ	λ	ACEPTA

aaabb

Estado	Pila	Cadena	Acción
s	λ	aaabb	$\lambda, \text{push}(\$) \rightarrow q$
q	\$	aaabb	$a, \text{push}(a) \rightarrow q$
q	\$a	aabb	$a, \text{push}(a) \rightarrow q$
q	\$aa	abb	$a, \text{push}(a) \rightarrow q$
q	\$aaa	bb	$\lambda, \text{push}(A) \rightarrow q$
q	\$aaaA	bb	$b, \text{push}(b) \rightarrow q$
q	\$aaaAb	b	$\lambda, \text{changeTop}(aAb, A) \rightarrow q$
q	\$aaA	b	$b, \text{push}(b) \rightarrow q$
q	\$aaAb	λ	$\lambda, \text{changeTop}(aAb, A) \rightarrow q$
q	\$aA	λ	Falla: no hay regla aplicable

Formalmente el proceso es el siguiente:

Dada una gramática $G=(N, \Sigma, S, P)$ se construye M para que $L(M)=L(G)$ así:

$M = (\{s, q, f\}, \Sigma, N \cup \Sigma \cup \{\$, s, \{f\}, \Delta)$ con

$\Delta = \{(s, \lambda, \lambda), (q, \$)\} \cup \{(q, \lambda, \$S), (f, \lambda)\} \cup \{(q, \sigma, \lambda), (q, \sigma): \sigma \in \Sigma\} \cup \{(q, \lambda, \omega), (q, A): A \rightarrow \omega \in \Sigma\}$

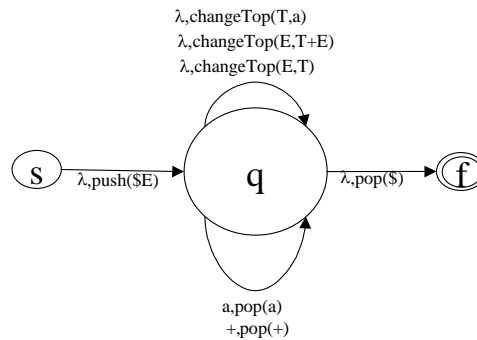
Este además analiza la cadena en forma ascendente.

A continuación se muestra otro ejemplo.

La siguiente gramática genera el lenguaje de las expresiones de sumas de a's con asociando a la izquierda:

- $E \rightarrow E + T$
- $E \rightarrow T$
- $T \rightarrow a$

El autómata descendente es:

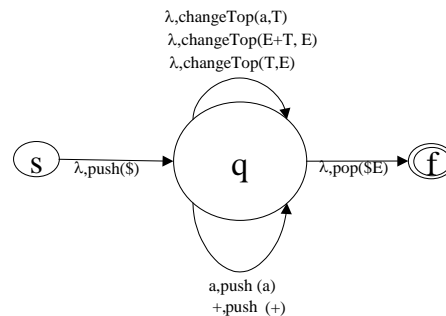


El seguimiento para la cadena a+a+a se muestra a continuación:

Estado	Pila	Cadena	Acción
s	λ	a+a+a	$\lambda, \text{push}(\$E) \rightarrow q$
q	$\$E$	a+a+a	$\lambda, \text{changeTop}(E, T+E) \rightarrow q$
q	$\$T+E$	a+a+a	$\lambda, \text{changeTop}(E, T+E) \rightarrow q$
q	$\$T+T+E$	a+a+a	$\lambda, \text{changeTop}(E, T) \rightarrow q$
q	$\$T+T+T$	a+a+a	$\lambda, \text{changeTop}(T, a) \rightarrow q$
q	$\$T+T+a$	a+a+a	$a, \text{pop}(a) \rightarrow q$
q	$\$T+T+$	+a+a	$+, \text{pop}(+) \rightarrow q$
q	$\$T+T$	a+a	$\lambda, \text{changeTop}(T, a) \rightarrow q$
q	$\$T+a$	a+a	$a, \text{pop}(a) \rightarrow q$
q	$\$T+$	+a	$+, \text{pop}(+) \rightarrow q$
q	$\$T$	a	$\lambda, \text{changeTop}(T, a) \rightarrow q$
q	$\$a$	a	$\lambda, \text{pop}(a) \rightarrow q$
q	$\$$	λ	$\lambda, \text{pop}(\$) \rightarrow f$
f	λ	λ	ACEPTA

Note que la derivación sugerida por este seguimiento es: $E \Rightarrow E+T \Rightarrow E+T+T \Rightarrow T+T+T \Rightarrow a+T+T \Rightarrow a+a+T \Rightarrow a+a+a$

El autómata ascendente se muestra a continuación:



El seguimiento para la cadena a+a+a es:

Estado	Pila	Cadena	Acción
s	λ	a+a+a	$\lambda, \text{push}(\$) \rightarrow q$
q	\$	a+a+a	$a, \text{push}(a) \rightarrow q$
q	\$a	+a+a	$\lambda, \text{changeTop}(a, T) \rightarrow q$
q	\$T	+a+a	$\lambda, \text{changeTop}(T, E) \rightarrow q$
q	\$ E	+a+a	$+, \text{push}(+) \rightarrow q$
q	\$E+	a+a	$a, \text{push}(a) \rightarrow q$
q	\$E+a	+a	$\lambda, \text{changeTop}(a, T) \rightarrow q$
q	\$E+T	+a	$\lambda, \text{changeTop}(E+T, E) \rightarrow q$
q	\$E	+a	$+, \text{push}(+) \rightarrow q$
q	\$E+	a	$a, \text{push}(a) \rightarrow q$
q	\$E+a	λ	$\lambda, \text{changeTop}(a, T) \rightarrow q$
q	\$E+T	λ	$\lambda, \text{changeTop}(E+T, E) \rightarrow q$
q	\$E	λ	$\lambda, \text{pop}(\$E) \rightarrow f$
f	λ	λ	ACEPTA

En este caso, la derivación correspondiente es: $E \Rightarrow E+T \Rightarrow E+a \Rightarrow E+T+a \Rightarrow E+a+a \Rightarrow T+a+a \Rightarrow a+a+a$

Cuando se está haciendo seguimiento de autómatas ascendentes a veces se usa otra notación. En lugar de decir a, push(a), se dice shift. Esto causa que el primer símbolo de la cadena de entrada pase a la pila. En lugar de decir **changeTop(E+T,E)** , se dice, reduce $E \rightarrow E+T$. No se empila \$ al principio; en lugar de esto se le agrega \$ al final de la cadena. Al estar E en el tope de la pila y \$ en la entrada, se acepta con la instrucción ACCEPT. A veces ni se dibuja el autómata.

Ejemplo 3.8 La siguiente gramática genera el lenguaje de expresiones de la forma a o f(exp, ..., exp)

- $E \rightarrow a$
- $E \rightarrow f(L)$
- $L \rightarrow E$
- $L \rightarrow L, E$

El seguimiento para f(f(a,a),a) sería:

Pila	Cadena	Acción
	f(f(a,a),a)\$	shift
f	(f(a,a),a)\$	shift
f(f(a,a),a)\$	shift
f(f	(a,a),a)\$	shift
f(f(a,a),a)\$	shift
f(f(a	,a),a)\$	reduce E→a
f(f(E	,a),a)\$	reduce L→E
f(f(L	,a),a)\$	shift
f(f(L,	a),a)\$	shift
f(f(L,a),a)\$	reduce E→a
f(f(L,E),a)\$	reduce L→L,E
f(f(L),a)\$	shift
f(f(L)	,a)\$	reduce E→f(L)
f(E	,a)\$	reduce L→E
f(L	,a)\$	shift
f(L,	a)\$	shift
f(L,a)\$	reduce E→a
f(L,E)\$	reduce L→L,E
f(L)\$	shift
f(L)	\$	reduce E→f(L)
E	\$	ACCEPT

EJERCICIOS

1. Construya los autómatas descendentes y ascendentes para las siguientes gramáticas
2. Sin construir el autómata ascendente muestre el seguimiento del reconocimiento de la cadena $a+a^*(a+a)$ para la gramática: (El símbolo distinguido es E u los terminales son a,+,.)

$E \rightarrow E + T$

$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow F$

$F \rightarrow a$

$F \rightarrow (E)$