Autómatas de pila (pushdown automata)

Autómata de pila (a.p.): sexteta $(S, \Sigma, \Gamma, T, \iota, F)$, en que:

- S es un conjunto finito de *estados*.
- Σ es un alfabeto (*alfabeto de entrada*).
- Γ es un alfabeto (*símbolos de pila*, suponemos $\# \in \Gamma$).
- $T \subseteq S \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \times \mathrm{S} \times (\Gamma \cup \{\lambda\})$ (*transiciones*).
- $\iota \in S$ es el estado *inicial*.
- $F \subseteq S$ es el conjunto de *estados de aceptación*.

El autómata *ejecuta* una transición $(p, x, s; q, y)$ si estando en el estado p con x bajo la cabeza lectora y s en el tope de la pila, entonces lee x, saca s de la pila, apila y y cambia al estado q. (Si cualquiera de x, s o y es λ, entonces la cabeza no se mueve, no se saca símbolo alguno de la pila o no se apila símbolo alguno, respectivamente.)

El autómata *acepta* la cadena $x_1 x_2 \cdots x_n$ si y solo si partiendo del estado inicial (con x_1 bajo la cabeza lectora) puede alcanzar un estado de aceptación después de leer (mediante ejecuciones) la cadena completa. El **lenguaje aceptado** por un autómata de pila M es el conjunto $L(M) = \{w \in \Sigma^\star : M \text{ acepta a } w\}$.

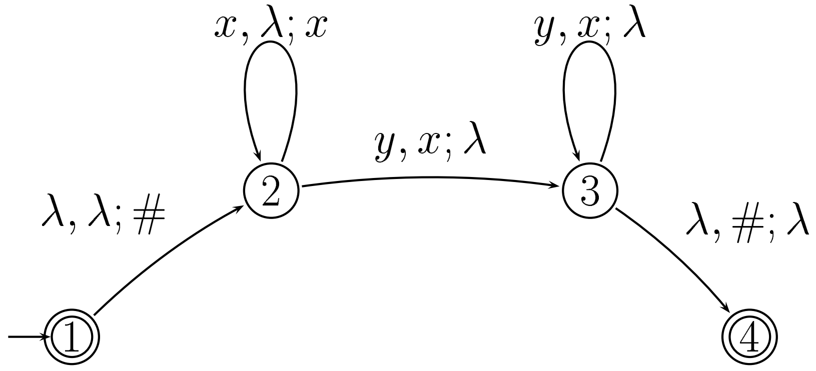
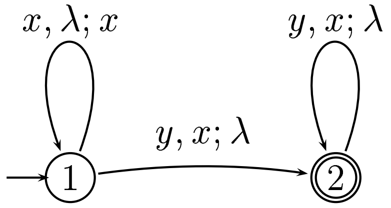


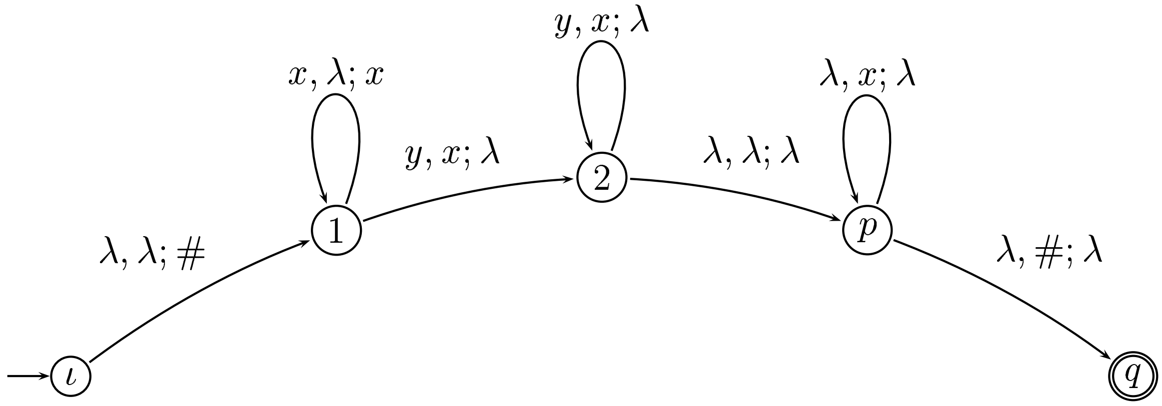
Figura 2.2: Diagrama de transición de autómata con $L(M) = \{x^n y^n : n \in \mathbb{N}\}$

Restringiendo las transiciones a la forma $(p, x, \lambda; q, \lambda)$, con $x \in \Sigma$, el autómata de pila se comporta como un autómata finito, de donde se sigue que *los lenguajes aceptados por los autómatas de pila incluyen a los lenguajes regulares*.

Teorema. Para cada autómata de pila que acepta cadenas sin vaciar la pila (inicialmente vacía), hay un autómata de pila que acepta el mismo lenguaje vaciando la pila antes de alcanzar un estado de aceptación.

Ejemplo. Los siguientes son diagramas de transición de autómatas de pila que aceptan el lenguaje $\{x^m y^n : m \geq n > 0\}$. El primero deja $m - n$ símbolos x en la pila. El segundo deja vacía la pila.





Decimos que una gramática $G = (V, T, S, R)$ es **libre de contexto** (g.l.c.) si para cada producción el lado izquierdo es un símbolo no terminal y el lado derecho es una cadena de $(V \cup T)^\star$.

Mediante $L(G)$ denotaremos al lenguaje generado por una gramática libre de contexto G y diremos que es un **lenguaje libre de contexto**.

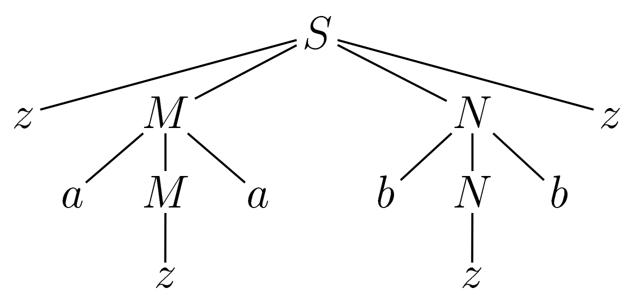
Ejemplo. Con $V = \{S, M, N\}$, $T = \{a, b, z\}$ y $R = \{S \to zMNz,\ M \to aMa,\ M \to z,\ N \to bNb,\ N \to z\}$, tenemos que:

$$L(G) = \{za^n za^n b^m zb^m z : m, n \in \mathbb{N}\}.$$

Ejemplos de derivaciones *por la izquierda (leftmost)* y *por la derecha (rightmost)*, respectivamente:

$$S \Rightarrow zMNz \Rightarrow zaMaNz \Rightarrow zazaNz \Rightarrow zazabNbz \Rightarrow zazabzbz$$
$$S \Rightarrow zMNz \Rightarrow zMbNbz \Rightarrow zMbzbz \Rightarrow zaMabzbz \Rightarrow zazabzbz$$

El *árbol sintáctico (parsing tree)* de las derivaciones anteriores:

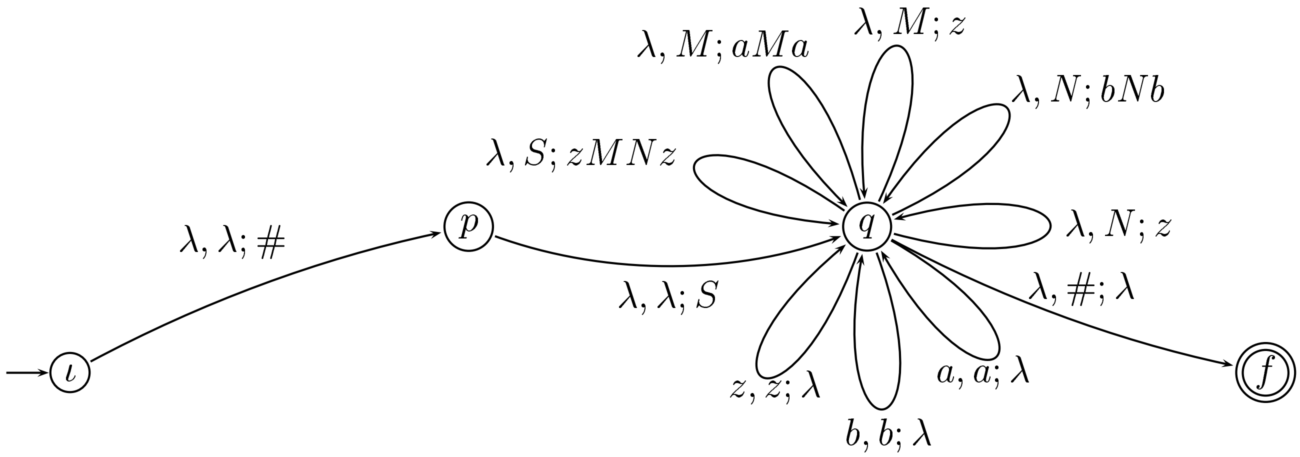


Es de notar que *toda cadena de un lenguaje libre de contexto puede generarse mediante una derivación por la izquierda.*

Teorema. $\{L(G) : G \text{ g.l.c. }\} = \{L(M) : M \text{ a.p. }\}$.

Teorema. Para cada gramática libre de contexto G hay un autómata de pila M tal que $L(G) = L(M)$.

Ejemplo. El siguiente es un diagrama de transición de un a.p. que reconoce el lenguaje generado por la g.l.c. del ejemplo anterior:



Dada $G = (V, T, S, R)$, constrúyase $M = (\mathcal{S}, \Sigma, \Gamma, \mathcal{T}, \iota, F)$ así:

1. $\Sigma = T$ (los símbolos terminales de G); $\Gamma = V \cup T \cup \{\#\}$ (con $\#$ un símbolo nuevo).
2. $\mathcal{S} = \{\iota, p, q, f\}$; $F = \{f\}$.
3. $\mathcal{T}$ con las siguientes transiciones:

$$
\begin{aligned}
&(\iota, \lambda, \lambda; p, \#),\\
&(p, \lambda, \lambda; q, S),\\
&(q, \lambda, N; q, w) \text{ para cada producción } N{\to}w,\\
&(q, x, x; q, \lambda) \text{ para cada } x \in \Sigma \text{ y}\\
&(q, \lambda, \#; f, \lambda).
\end{aligned}
$$

<u>Teorema</u>. Para cada autómata de pila M hay una gramática libre de contexto G tal que $L(G) = L(M)$.

Dado $M = (\mathcal{S}, \Sigma, \Gamma, \mathcal{T}, \iota, F)$, que podemos suponer acepta cadenas solo si deja vacía la pila, constrúyase $G = (V, T, S, R)$ así:

$V = \{S\} \cup \{\, \langle p, x, q \rangle : p, q \in \mathcal{S},\ x \in \Gamma \cup \{\lambda\} \,\}$.

$T = \Sigma$.

R con las siguientes producciones:

1. $S \to \langle \iota, \lambda, f \rangle$ para cada $f \in F$,
2. $\langle p, \lambda, p \rangle \to \lambda$ para cada $p \in \mathcal{S}$,
3. $\langle p, y, r \rangle \to x \langle q, z, r \rangle$ para cada $(p, x, y; q, z) \in \mathcal{T}$, con $y \neq \lambda$ y cada $r \in \mathcal{S}$,

$$\textcircled{p} \overset{\langle p,y,r\rangle}{\dashrightarrow} \textcircled{r} \qquad\qquad \textcircled{p} \xrightarrow{x,y;z} \textcircled{q} \overset{\langle q,z,r\rangle}{\dashrightarrow} \textcircled{r}$$

Sobre M, $\langle p, y, r \rangle$ significa la meta (deseo) de ir de p a r sacando y del tope y sin modificar el resto de la pila. Si dicha computación comienza cambiando a y por z, quedando en el estado q, entonces para lograr la meta inicial se debe cumplir la submeta $\langle q, z, r \rangle$.

4. $\langle p, y, r \rangle \to x \langle q, z, k \rangle \langle k, y, r \rangle$ para cada $(p, x, \lambda; q, z) \in \mathcal{T}$, cada $y \in \Gamma \cup \{\lambda\}$ y cada $k, r \in \mathcal{S}$.

$$\textcircled{p} \overset{\langle p,y,r\rangle}{\dashrightarrow} \textcircled{r} \qquad\qquad \textcircled{p} \xrightarrow{x,\lambda;z} \textcircled{q} \overset{\langle q,z,k\rangle}{\dashrightarrow} \textcircled{k} \overset{\langle k,y,r\rangle}{\dashrightarrow} \textcircled{r}$$

Si la computación apila a z sobre y, quedando en el estado q, entonces se logrará la meta inicial si se cumplen las submetas $\langle q, z, k \rangle$ y $\langle k, y, r \rangle$, para algún estado k.

Probemos $L(G) = L(M)$, lo que se sigue del siguiente lema:

<u>Lema</u>. $\langle p, y, r \rangle \overset{*}{\underset{G}{\Rightarrow}} w$ $(w \in T^\star)$ $\Longleftrightarrow$ M puede moverse del estado p al estado r por una ruta de tal suerte que al final w ha sido leída y y sacada del tope de la pila (sin modificar el resto de la pila).

Para demostrar este lema conviene la siguiente notación. Mediante $[p, \pi, w]$ indicaremos que cuando M se encuentra en el estado p durante una computación, el contenido de la pila es π y lo que resta de la palabra bajo análisis es w. A una tripleta tal se le llama *descripción instantánea* de M. Los pasos de una computación serán representados como se ilustra a continuación:

$[p, y\pi, xw] \xrightarrow{(p,x,y;q,z)} [q, z\pi, w]$ $\qquad$ $[p, y\pi, xw] \xrightarrow{(p,x,\lambda;q,z)} [q, zy\pi, w]$

$[p, \pi, w] \xrightarrow{*} [p', \pi', w']$ (aplicación de un número finito de transiciones)

$[p, \pi, w] \xrightarrow{n} [p', \pi', w']$ (aplicación de n transiciones)

Usando esta notación, el lema se puede reformular así:

Lema: $\langle p, y, r\rangle \xrightarrow[G]{*} w \iff [p, y\pi, w] \xrightarrow{*} [r, \pi, \lambda] \qquad (\pi \in \Gamma^\star)$.

Demostración:

$\Rightarrow$) P.D. $\langle p, y, r\rangle \xrightarrow[G]{*} w \implies [p, y\pi, w] \xrightarrow{*} [r, \pi, \lambda]$.

Inducción en la longitud de la derivación $\langle p, y, r\rangle \xrightarrow[G]{*} w$

Base: $\langle p, y, r\rangle \xrightarrow[G]{1} w \implies y = \lambda$, $p = r$ y $w = \lambda$,

$\therefore \langle p, \lambda, p\rangle \xrightarrow[G]{1} \lambda$ y $[p, \pi, \lambda] \xrightarrow{0} [p, \pi, \lambda]$

Paso Inductivo: Suponer a) (hip. ind.) y b) para demostrar c):

a) $\forall \langle p, y, r\rangle, w, \pi \quad \forall m \leq n$

$$\langle p, y, r\rangle \xrightarrow[G]{m} w \implies [p, y\pi, w] \xrightarrow{*} [r, \pi, \lambda]$$

b) $\langle p, y, r\rangle \xrightarrow[G]{n+1} xw$

c) P.D. $[p, y\pi, xw] \xrightarrow{*} [r, \pi, \lambda]$

La primera producción aplicada en la derivación de xw puede ser de tipo 3 o 4; respectivamente tenemos:

Tipo 3: $\langle p, y, r\rangle \xrightarrow[G]{1} x\langle q, z, r\rangle \xrightarrow[G]{n} xw \implies \exists (p, x, y; q, z) \in \mathcal{T}$,

$\therefore [p, y\pi, xw] \xrightarrow{1} [q, z\pi, w] \xrightarrow{*} [r, \pi, \lambda]$

Tipo 4: $\langle p, y, r\rangle \xrightarrow[G]{1} x\langle q, z, k\rangle\langle k, y, r\rangle \xrightarrow[G]{n} xw \implies (p, x, \lambda; q, z) \in \mathcal{T}$ y hay derivaciones $\langle q, z, k\rangle \xrightarrow[G]{n-j} w_1$ y $\langle k, y, r\rangle \xrightarrow[G]{j} w_2$, con $w = w_1w_2$,

$\therefore [p, y\pi, xw] \xrightarrow{1} [q, zy\pi, w_1w_2] \xrightarrow{*} [k, y\pi, w_2] \xrightarrow{*} [r, \pi, \lambda]$

$\Leftarrow$) P.D. $[p, y\pi, w] \overset{*}{\Longrightarrow} [r, \pi, \lambda] \quad \Longrightarrow \quad \langle p, y, r\rangle \underset{G}{\overset{*}{\Longrightarrow}} w$

Inducción sobre la longitud de $[p, y\pi, w] \overset{*}{\Longrightarrow} [r, \pi, \lambda]$

Base

$[p, y\pi, w] \overset{0}{\Longrightarrow} [r, \pi, \lambda] \quad \Longrightarrow \quad p = r,\ y = \lambda$ y $w = \lambda$,

$\therefore$ podemos aplicar $\langle p, \lambda, p\rangle \longrightarrow \lambda$.

Paso Inductivo Suponer a) (hip. ind.) y b) para demostrar c):

a) $\forall p, r, y, \pi, w \quad \forall m \leq n$

$$[p, y\pi, w] \overset{m}{\Longrightarrow} [r, \pi, \lambda] \quad \Longrightarrow \quad \langle p, y, r\rangle \underset{G}{\overset{*}{\Longrightarrow}} w$$

b) $[p, y\pi, w] \overset{n+1}{\Longrightarrow} [r, \pi, \lambda]$ (para algunas p, r, y, π, w)

c) P.D. $\langle p, y, r\rangle \underset{G}{\overset{*}{\Longrightarrow}} w$

La primera transición en la computación b) puede ser $(p, x, y; q, z)$, con $y \neq \lambda$, o $(p, x, \lambda; q, z)$; respectivamente tenemos:

Caso 1: $[p, y\pi, w] \overset{(p,x,y;q,z)}{\Longrightarrow} [q, z\pi, w_1] \overset{n}{\Longrightarrow} [r, \pi, \lambda]$ con $w = xw_1$ y tenemos:

$\langle p, y, r\rangle \underset{G}{\overset{1}{\Longrightarrow}} x\langle q, z, r\rangle \underset{G}{\overset{*}{\Longrightarrow}} xw_1$,

$\therefore \langle p, y, r\rangle \underset{G}{\overset{*}{\Longrightarrow}} w$.

Caso 2: $[p, y\pi, xw'] \overset{(p,x,\lambda;q,z)}{\Longrightarrow} [q, zy\pi, w'] \overset{n}{\Longrightarrow} [r, \pi, \lambda]$,

en que $w' = w_1w_2$, con $[q, zy\pi, w_1w_2] \overset{j}{\Longrightarrow} [k, y\pi, w_2] \overset{n-j}{\Longrightarrow} [r, \pi, \lambda]$;

Aplicando a) a estas dos subcomputaciones, (usando también

$[q, z\pi', w_1w_2] \overset{j}{\Longrightarrow} [k, \pi', w_2] \iff [q, z\pi', w_1] \overset{j}{\Longrightarrow} [k, \pi', \lambda]$

para la primera), obtenemos

$\langle q, z, k\rangle \underset{G}{\overset{*}{\Longrightarrow}} w_1$ y $\langle k, y, r\rangle \underset{G}{\overset{*}{\Longrightarrow}} w_2$;

entonces

$\langle p, y, r\rangle \underset{G}{\overset{1}{\Longrightarrow}} x\langle q, z, k\rangle\langle k, y, r\rangle \underset{G}{\overset{*}{\Longrightarrow}} xw_1\langle k, y, r\rangle \underset{G}{\overset{*}{\Longrightarrow}} xw_1w_2$,

$\therefore \langle p, y, r\rangle \underset{G}{\overset{*}{\Longrightarrow}} w$.

Forma normal de Chomsky

Sea G una g.l.c. Una *cadena–*λ de longitud n es una sucesión de producciones de la forma:

$$N_n \to N_{n-1}, \quad N_{n-1} \to N_{n-2}, \ldots, N_1 \to N_0, \quad N_0 \to \lambda$$

con $N_n, \ldots, N_0 \in V$. Diremos que N_n es el *origen* de la cadena.

Sea $\mathcal{U} \subseteq V$ el conjunto de orígenes de cadenas–λ. Supongamos $\lambda \notin L(G)$. Si se eliminan las producciones–λ (las de forma $N \to \lambda$) y se aplica el siguiente procedimiento, se obtendrá una gramática G' que genera el mismo lenguaje que G:

- Por cada producción $N \to w$, agréguense las producciones $N \to w'$, en que w' es cualquier cadena no vacía que se obtenga suprimiendo de w una o más ocurrencias de símbolos de $\mathcal{U}$.

Por ejemplo, a partir de:

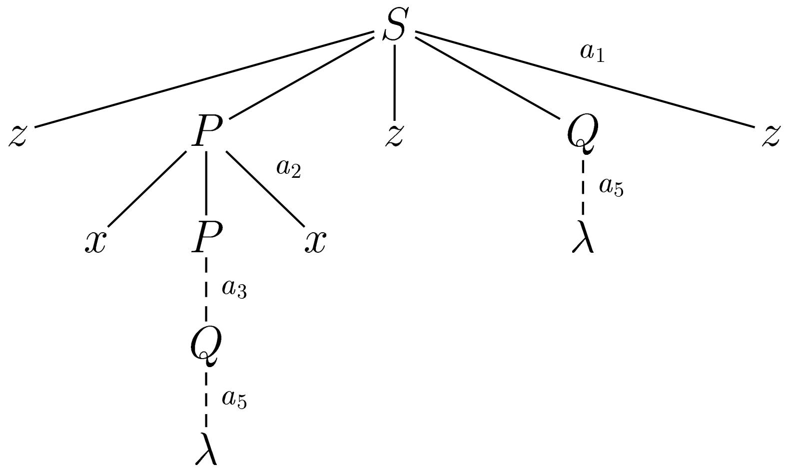
$$\begin{array}{ll} (a_1) & S \to zPzQz \\ (a_2) & P \to xPx \\ (a_3) & P \to Q \\ (a_4) & Q \to yPy \\ (a_5) & Q \to \lambda \end{array}$$

se obtiene:

$$\begin{array}{ll} (b_1) & S \to zPzQz \\ (b_1') & S \to zzQz \\ (b_1'') & S \to zPzz \\ (b_1''') & S \to zzz \\ (b_2) & P \to xPx \\ (b_2') & P \to xx \\ (b_3) & P \to Q \\ (b_4) & Q \to yPy \\ (b_4') & Q \to yy \end{array}$$

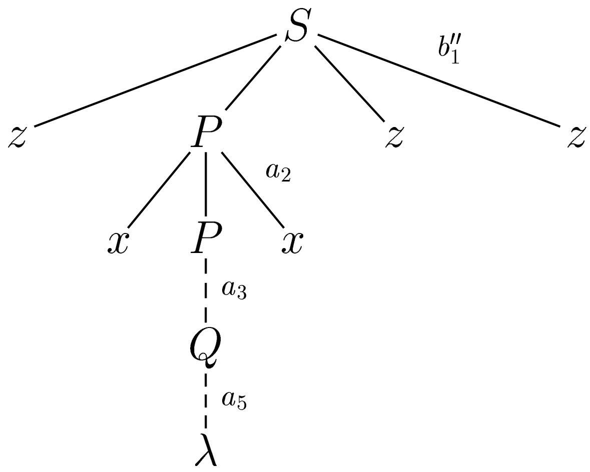
Eliminando las cadenas–λ que ocurran en una derivación en G, se obtiene una derivación en G' de la misma palabra, como se ilustra en las figuras siguientes:

Derivación de $zxxzz$ en que ocurren dos cadenas–λ



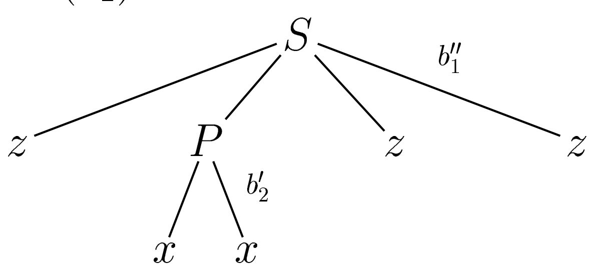
$$S \overset{a_1}{\Rightarrow} zPzQz \overset{a_2}{\Rightarrow} zxPxzQz \overset{a_3}{\Rightarrow} zxQxzQz \overset{a_5}{\Rightarrow} zxxzQz \overset{a_5}{\Rightarrow} zxxzz$$

Aplicando (b_1'') en vez de (a_1) se elimina la cadena–λ $Q \rightarrow \lambda$:



$$S \overset{b_1''}{\Rightarrow} zPzz \overset{a_2}{\Rightarrow} zxPxzz \overset{a_3}{\Rightarrow} zxQxzz \overset{a_5}{\Rightarrow} zxxzz$$

Aplicando (b_2') en vez de (a_2) se elimina la cadena–λ $P \rightarrow Q \rightarrow \lambda$:



$$S \overset{b_1''}{\Rightarrow} zPzz \overset{b_2'}{\Rightarrow} zxxzz$$

Lema. Para cualquier lenguaje l.c. que no tenga a λ hay alguna g.l.c. sin producciones–λ que lo genera.

Teorema. Si L es un lenguaje l.c. que no tiene a λ, entonces hay una g.l.c. G'' tal que $L(G'') = L$ y cada producción de G'' es de alguna de las dos formas $A \to a$ o $A \to BC$ (con $a \in T$ y $A, B, C \in V$).

Demostración. De acuerdo con el lema anterior, sea G' una g.l.c. sin producciones–λ que genera a L. A partir de G' constrúyase la gramática G'' así:

1. Por cada terminal $x \in G'$ agréguese un nuevo no terminal X y la producción $X \to x$. En las demás producciones sustitúyanse las ocurrencias de x por X.
2. Sustitúyase cada producción de la forma $N \to N_1 N_2 \cdots N_n$, con $n > 2$, por las $n-1$ siguientes: $N \to N_1 R_1$, $R_1 \to N_2 R_2$, …, $R_{n-2} \to N_{n-1} N_n$, en que cada R_k es un nuevo no terminal y no se usa en otras producciones.
3. Para eliminar las producciones que en su lado derecho tienen un no terminal solitario, considérese cada sucesión de producciones de la forma:

$$N_n \to N_{n-1}, \quad N_{n-1} \to N_{n-2}, \quad \ldots, \quad N_2 \to N_1$$

entonces introdúzcase la producción $N_n \to x$ si ya se tiene la producción $N_1 \to x$, e introdúzcase la producción $N_n \to AB$ si ya se tiene $N_1 \to AB$. Por último, suprímanse las producciones con solo un no terminal en su lado derecho.

Por ejemplo, a partir de:

$$\begin{array}{ll} (a_1) & S \to zMz \\ (a_2) & M \to N \\ (a_3) & M \to yMy \\ (a_4) & N \to x \end{array}$$

en el primer paso se obtiene:

$$\begin{array}{ll} (b_1) & S \to ZMZ \\ (b_2) & M \to N \\ (b_3) & M \to YMY \\ (b_4) & N \to X \\ (b_5) & X \to x \\ (b_6) & Y \to y \\ (b_7) & Z \to z \end{array}$$

después del segundo paso tendremos:

$$\begin{array}{ll} (c_1') & S \to ZR_1 \\ (c_1'') & R_1 \to MZ \\ (c_2) & M \to N \\ (c_3') & M \to YP_1 \\ (c_3'') & P_1 \to MY \\ (c_4) & N \to X \\ (c_5) & X \to x \\ (c_6) & Y \to y \\ (c_7) & Z \to z \end{array}$$

por último obtenemos:

$$\begin{array}{ll} (d_1') & S \to ZR_1 \\ (d_1'') & R_1 \to MZ \\ (d_2') & M \to x \\ (d_3') & M \to YP_1 \\ (d_3'') & P_1 \to MY \\ (d_4') & N \to x \\ (d_5) & X \to x \\ (d_6) & Y \to y \\ (d_7) & Z \to z \end{array}$$

Se dice que una gramática tiene **forma normal de Chomsky** si cumple con las restricciones presentadas en el Teorema anterior.

Si un lenguaje l.c. L incluye a λ, se puede obtener una g.l.c. *casi* en forma normal de Chomsky de la siguiente manera. Se construye una gramática G en forma normal de Chomsky para $L-\{\lambda\}$, con S como símbolo inicial; se modifica G introduciendo a S' como nuevo símbolo inicial; por cada regla cuyo lado izquierdo sea S se agrega una igual, pero con S' en dicho lugar; por último, se agrega la regla $S' \to \lambda$. Ejemplo:

Gramática para $L-\{\lambda\}$	Gramática para L
$$\begin{aligned} S &\to MN \\ N &\to MS \\ N &\to x \\ M &\to x \end{aligned}$$	$$\begin{aligned} S' &\to \lambda \\ S' &\to MN \\ S &\to MN \\ N &\to MS \\ N &\to x \\ M &\to x \end{aligned}$$

```
% Curso de Autómatas y Lenguajes Formales.
%
% Ejemplo de programa en Prolog para
% generar las palabras de la gramática
%
%   S -> zMNz
%   M -> aMa
%   M -> z
%   N -> bNb
%   N -> z
%

terminal(a).
terminal(b).
terminal(z).

noterminal(nS).
noterminal(nM).
noterminal(nN).

produccion(nS,[z,nM,nN,z]).
produccion(nM,[z]).
produccion(nM,[a,nM,a]).
produccion(nN,[z]).
produccion(nN,[b,nN,b]).

% reescribe(W,W1): Reescribiendo el primer símbolo no terminal
%                  que ocurra en W se obtiene W1. Ejemplos:
%
%                  ?- reescribe([nS],W).
%                  W = [z,nM,nN,z]
%
%                  ?- reescribe([z,nM,nN,z],W).
%                  W = [z,z,nN,z]
%                  W = [z,a,nM,a,nN,z]
%
%                  ?- reescribe([z,a],W).
%                  W = [z,a]
%
reescribe([],[]).
reescribe([X|Xs],W) :- produccion(X,W1), pega(W1,Xs,W).
reescribe([X|Xs],[X|W]) :- terminal(X), reescribe(Xs,W).

% pega(Xs,Ys,XsYs): XsYs es la concatenacion de Xs y Ys.
%                   Ejemplo:
%                   ?- pega([b,a],[b,nN,b],W).
%                   W = [b, a, b, nN, b]
%
pega([],Ys,Ys).
pega([X|Xs],Ys,[X|Zs]) :- pega(Xs,Ys,Zs).
```

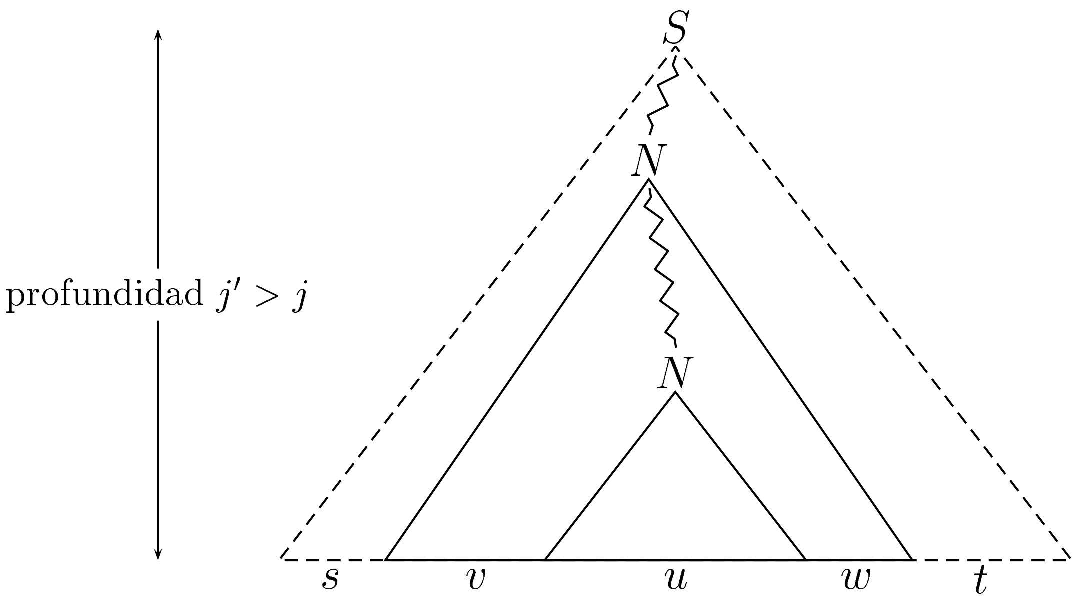
```
% genera(W,W1): Aplicando a W un número finito (mayor que 0)
%               de reescrituras se obtiene W1.
%
genera(W,W1) :- reescribe(W,W1).
genera(W,W2) :- genera(W,W1), reescribe(W1,W2).

% ?- genera([nS],W).
%
% W = [z,nM,nN,z]            <===
% W = [z,z,nN,z]
% W = [z,a,nM,a,nN,z]        <===
% W = [z,z,z,z]
% W = [z,z,b,nN,b,z]
% W = [z,a,z,a,nN,z]         <===
% W = [z,a,a,nM,a,a,nN,z]
% W = [z,z,z,z]
% W = [z,z,b,z,b,z]
% W = [z,z,b,b,nN,b,b,z]
% W = [z,a,z,a,z,z]
% W = [z,a,z,a,b,nN,b,z]     <===
% W = [z,a,a,z,a,a,nN,z]
% W = [z,a,a,a,nM,a,a,a,nN,z]
% W = [z,z,z,z]
% W = [z,z,b,z,b,z]
% W = [z,z,b,b,z,b,b,z]
% W = [z,z,b,b,b,nN,b,b,b,z]
% W = [z,a,z,a,z,z]
% W = [z,a,z,a,b,z,b,z]      <===
% W = [z,a,z,a,b,b,nN,b,b,z]
% W = [z,a,a,z,a,a,z,z]
% ...
%
% ?- genera([X],[z,a,z,a,b,z,b,z]).
% X = nS
```

```
% Curso de Autómatas y Lenguajes Formales.
%
% Ejemplo de programa en Prolog para
% generar las palabras de la gramática
%
%   S -> zMNz
%   M -> aMa
%   M -> z
%   N -> bNb
%   N -> z
%
% mediante "listas-diferencia".

nS(U,Y) :- z(U,V), nM(V,W), nN(W,X), z(X,Y).
nM(U,V) :- z(U,V).
nM(U,X) :- a(U,V), nM(V,W), a(W,X).
nN(U,V) :- z(U,V).
nN(U,X) :- b(U,V), nN(V,W), b(W,X).

z([z|X],X).
a([a|X],X).
b([b|X],X).

generable(W) :- nS(W,[]).

% ?- generable([z,a,z,a,b,z,b,z]).
% *** Yes
%
% ?- generable([a,a,b,b]).
% *** No
%
% ?- nM(W,[]).
% W = [z]
% W = [a, z, a]
% W = [a, a, z, a, a]
% W = [a, a, a, z, a, a, a]
% ...
% ?- nN(W,[]).
% W = [z]
% W = [b, z, b]
% W = [b, b, z, b, b]
% W = [b, b, b, z, b, b, b]
% ...
% ?- nS(W,[]).
% W = [z, z, z, z]
% W = [z, z, b, z, b, z]
% W = [z, z, b, b, z, b, b, z]
% W = [z, z, b, b, b, z, b, b, b, z]
% ...
```

Teorema 2.5. Sea L un lenguaje l.c. infinito. Entonces

$$\exists s,v,u,w,t \in L.((v \neq \lambda \vee w \neq \lambda) \wedge \forall n \in \mathbb{N}^+ . sv^n uw^n t \in L)$$

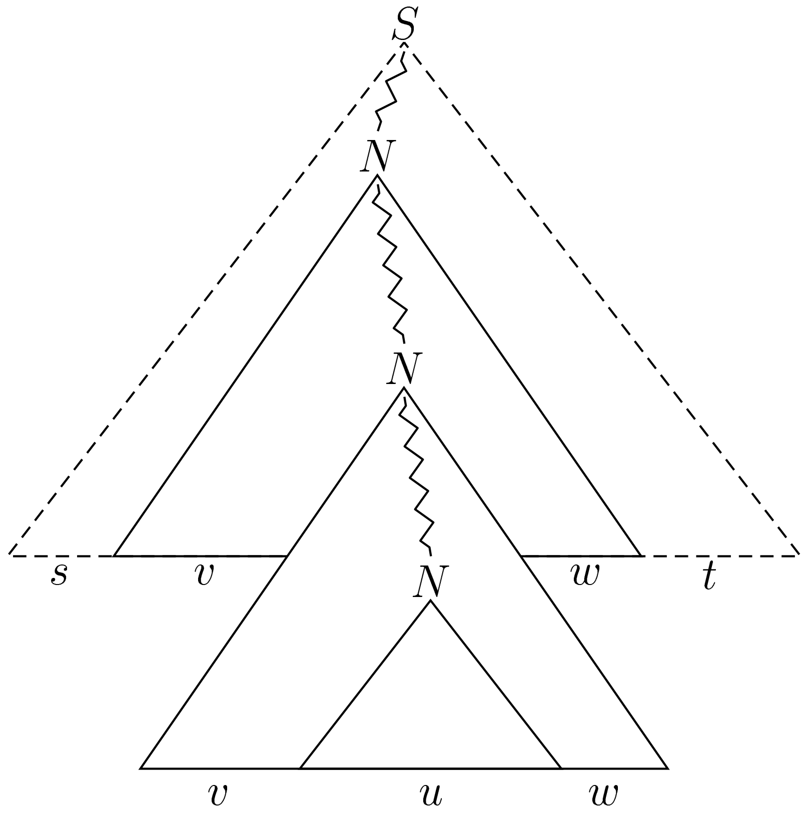
Demostración. Sea $G = (V, T, S, R)$ una gramática l.c. para L. Sea $m = \max\{|\beta| : (B \to \beta) \in R, \beta \in (V \cup T)^\star\}$. De aquí que si p es la profundidad (longitud de la trayectoria más larga) de un árbol sintáctico de una derivación en G, entonces el árbol tendrá cuando mucho m^p nodos terminales. Sea $j = |V|$ y tomemos cualquier palabra en L de longitud mayor que m^j; entonces todo árbol de derivación de dicha palabra deberá ser de profundidad mayor que j. Consideremos uno de estos árboles y alguna trayectoria en él de profundidad $j' > j$. En esta trayectoria necesariamente debe ocurrir más de una vez algún símbolo no terminal N. Fijémonos en la primera y en la última ocurrencia de N a lo largo de esa trayectoria. En la figura siguiente se ilustra que la palabra elegida puede factorizarse en la forma $svuwt$, correspondiente a la derivación $S \stackrel{*}{\Rightarrow} sNt \stackrel{*}{\Rightarrow} svNwt \stackrel{*}{\Rightarrow} svuwt$ (para la primera ocurrencia de N se tiene $N \stackrel{*}{\Rightarrow} vNw$ y para la segunda ocurrencia $N \stackrel{*}{\Rightarrow} u$):



$S \stackrel{*}{\Rightarrow} sNt \stackrel{*}{\Rightarrow} svNwt \stackrel{*}{\Rightarrow} svuwt$

Pudiera ser que tanto v como w fueran λ, caso en que se tendría la derivación $S \stackrel{*}{\Rightarrow} sNt \stackrel{*}{\Rightarrow} sNt \stackrel{*}{\Rightarrow} sut$, de la que se puede eliminar el segundo paso (correspondiente a la derivación de la segunda ocurrencia de N a partir de la primera ocurrencia: $N\stackrel{*}{\Rightarrow}N$), dando lugar a la derivación más corta $S\stackrel{*}{\Rightarrow}sNt\stackrel{*}{\Rightarrow}sut$. Con ésto se reduciría la longitud j' de la trayectoria elegida. Sin embargo, la profundidad del árbol correspondiente a la nueva derivación de la palabra elegida también deberá ser mayor que j, de donde se sigue que esta manera de acortar trayectorias solo se puede repetir un número finito de veces (si no fuera así, eventualmente se obtendría un árbol de profundidad no mayor que j). De acuerdo con lo anterior, podemos suponer que en el árbol de derivación considerado, para la trayectoria elegida se tiene que $v \neq \lambda$ ó $w \neq \lambda$.

Ahora bien, como se tiene $N\stackrel{*}{\Rightarrow}vNw$, componiendo esta derivación consigo misma tantas veces como se quiera, se obtienen las de forma $N\stackrel{*}{\Rightarrow}v^nNw^n$, con $n\in\mathbb{N}^+$. Como también se tienen $S\stackrel{*}{\Rightarrow}sNt$ y $N\stackrel{*}{\Rightarrow}u$, se puede construir $S\stackrel{*}{\Rightarrow}sNt\stackrel{*}{\Rightarrow}sv^nNw^nt\stackrel{*}{\Rightarrow}sv^nuw^nt$. Por lo tanto $sv^nuw^nt\in L$. La figura siguiente ilustra el caso $n = 2$:



$$S\stackrel{*}{\Rightarrow}sNt\stackrel{*}{\Rightarrow}svNwt\stackrel{*}{\Rightarrow}sv^2Nw^2t\stackrel{*}{\Rightarrow}sv^2uw^2t$$

<u>Teorema 2.6</u>. Hay un lenguaje libre de contexto para el que no hay autómata de pila determinístico del que sea el lenguaje de aceptación.

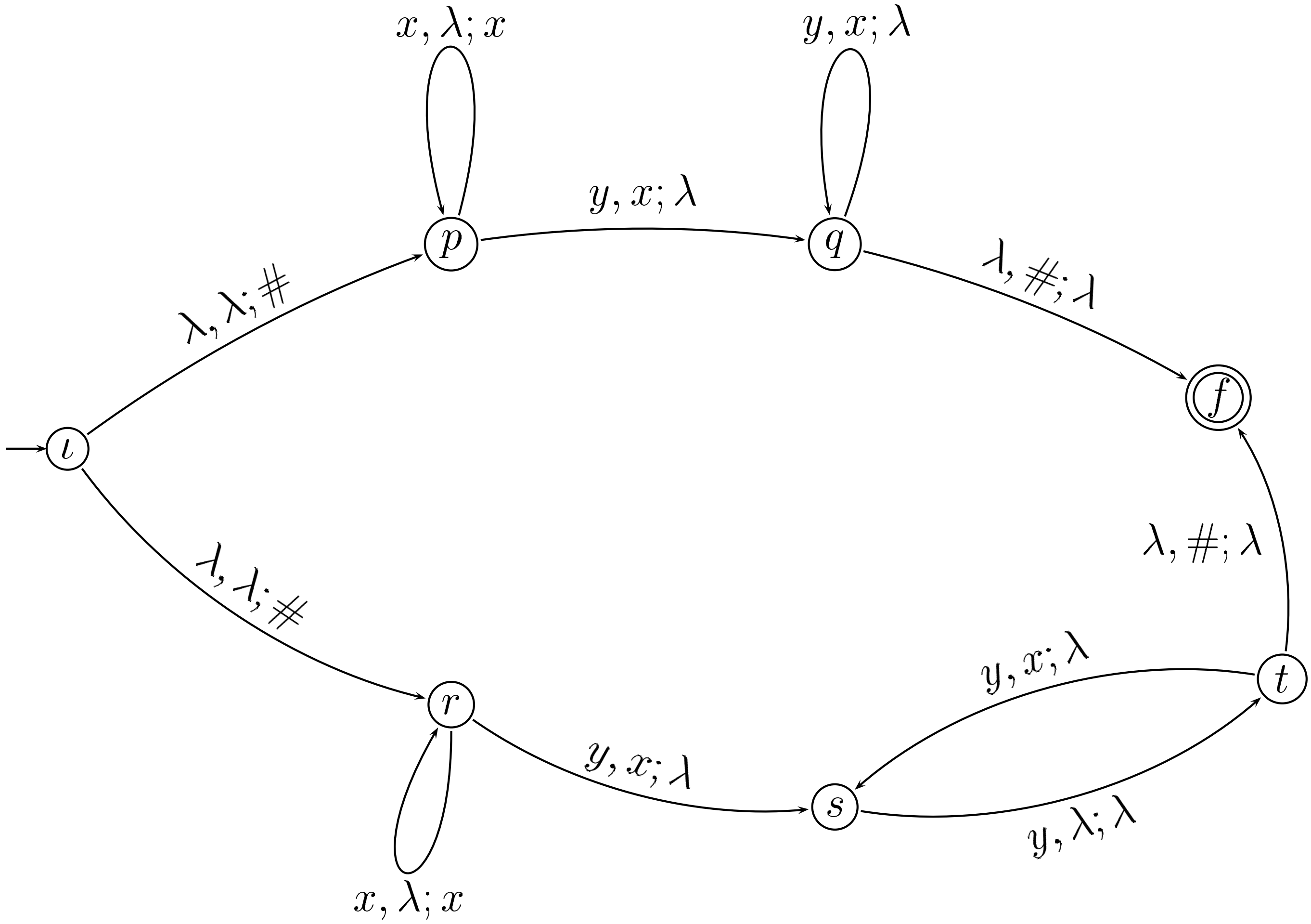
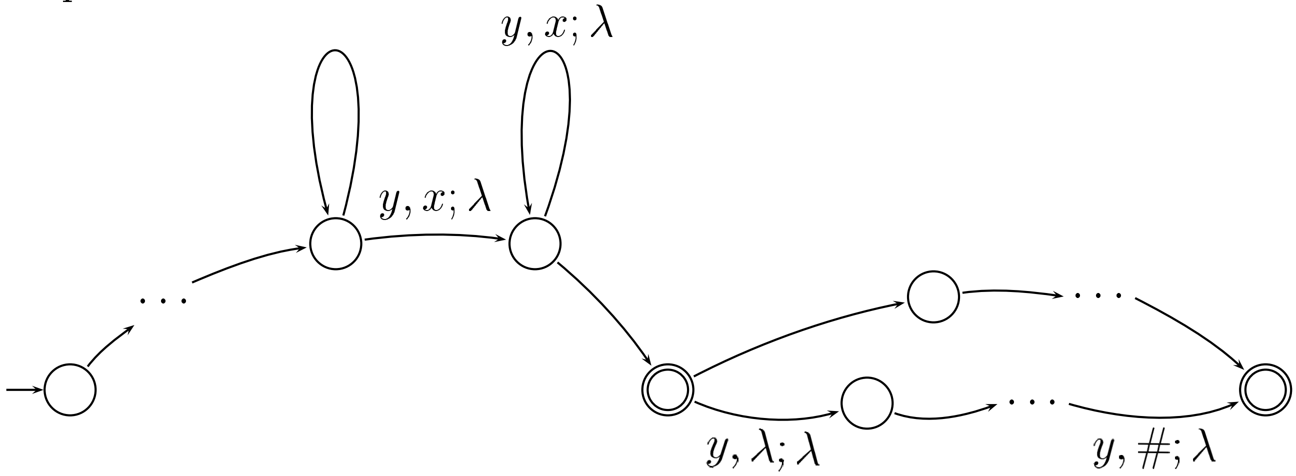


Figura 2.21: Diagrama de transición para autómata de pila M, con $L(M) = \{x^n y^n : n \in \mathbb{N}^+\} \cup \{x^n y^{2n} : n \in \mathbb{N}^+\}$

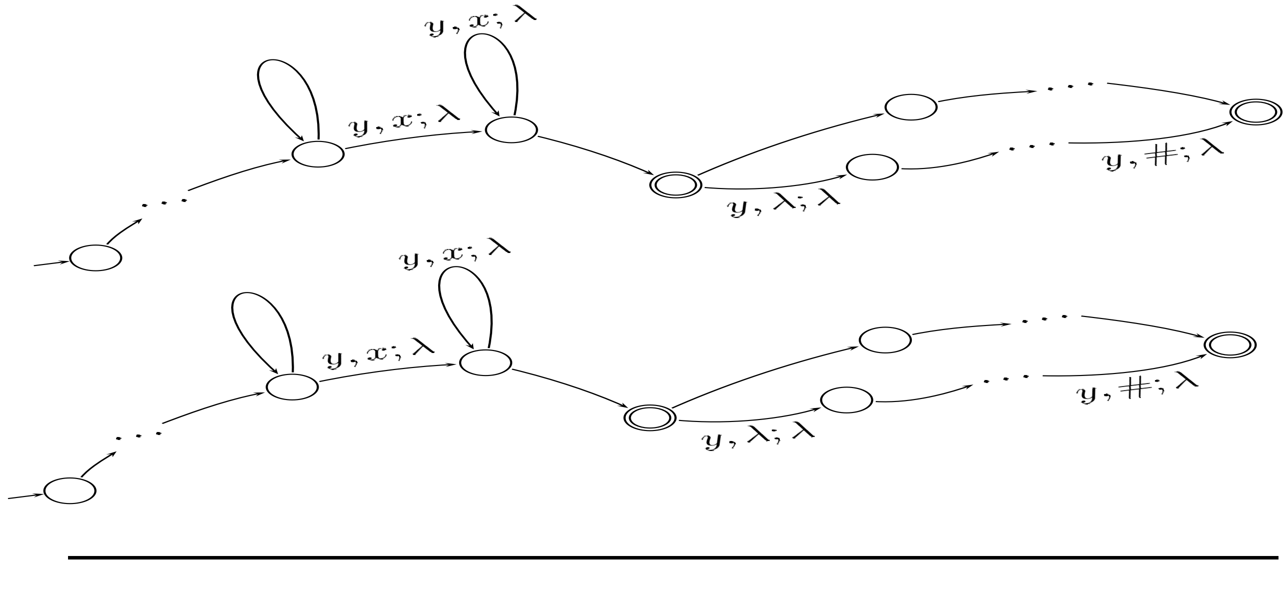
Ahora se demostrará, por reducción al absurdo, que no hay autómata de pila determinístico cuyo lenguaje de aceptación sea $L(M)$ (presentado en la figura anterior).

Esqueleto de diagrama de transición de autómata de pila determinista M que supuestamente aceptaría al lenguaje $\{x^n y^n : n \in \mathbb{N}^+\} \cup \{x^n y^{2n} : n \in \mathbb{N}^+\}$. Se destacan las transiciones que leen y de la entrada, el nodo inicial y los nodos de aceptación:

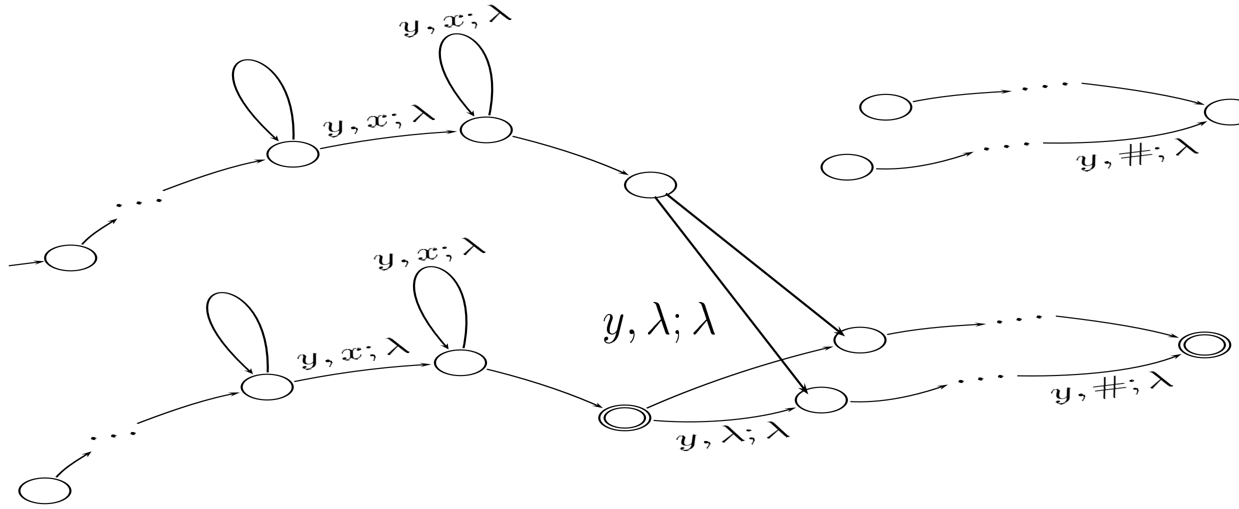


Para demostrar el Teorema 2.6, a partir del supuesto autómata M constrúyase uno nuevo, dando los pasos ilustrados en las tres figuras siguientes.

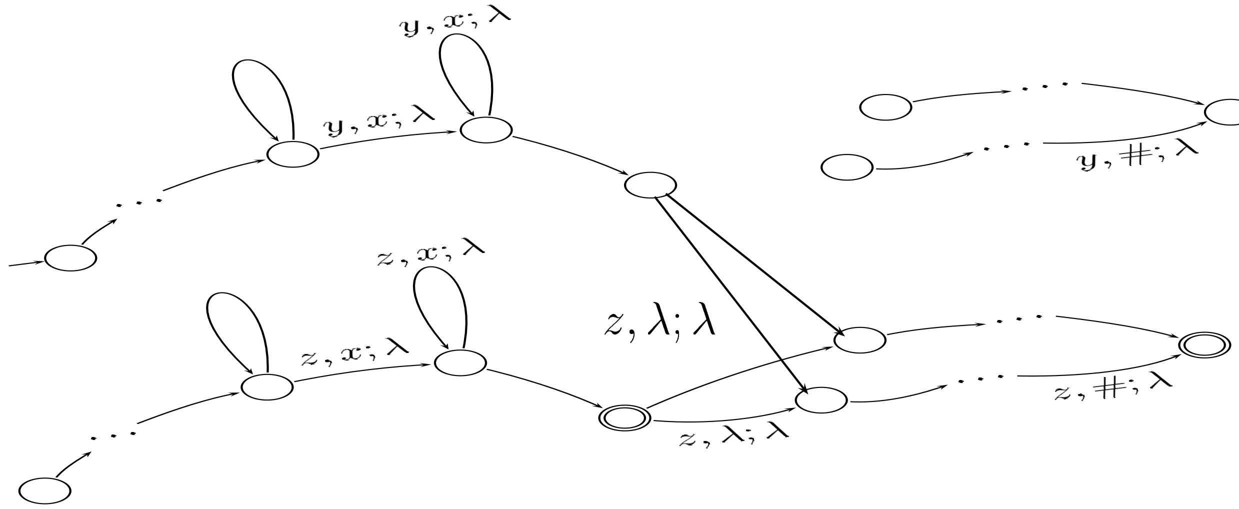
Tómense dos copias de M; llamémoslas M_1 y M_2, repectivamente:



A cada nodo de aceptación de M_1 quítesele tal característica. De M_2, al nodo inicial quítesele esta propiedad. A cada transición que se origina en algún viejo estado de aceptación de M_1 cámbiesele el estado de destino, para que sea el correspondiente estado en M_2:



Modifíquense las transiciones que leen una y de la entrada y que tienen como destino algún estado de M_2, para que en lugar de y lean z:



Tendríamos entonces un autómata de pila cuyo lenguaje de aceptación sería $\{x^n y^n z^n : n \in \mathbb{N}^+\}$, lo cual es imposible, pues este lenguaje ¡no es libre de contexto!. Por lo tanto, el supuesto autómata M no existe.

Analizadores sintácticos (*parsers*) $LL(k)$

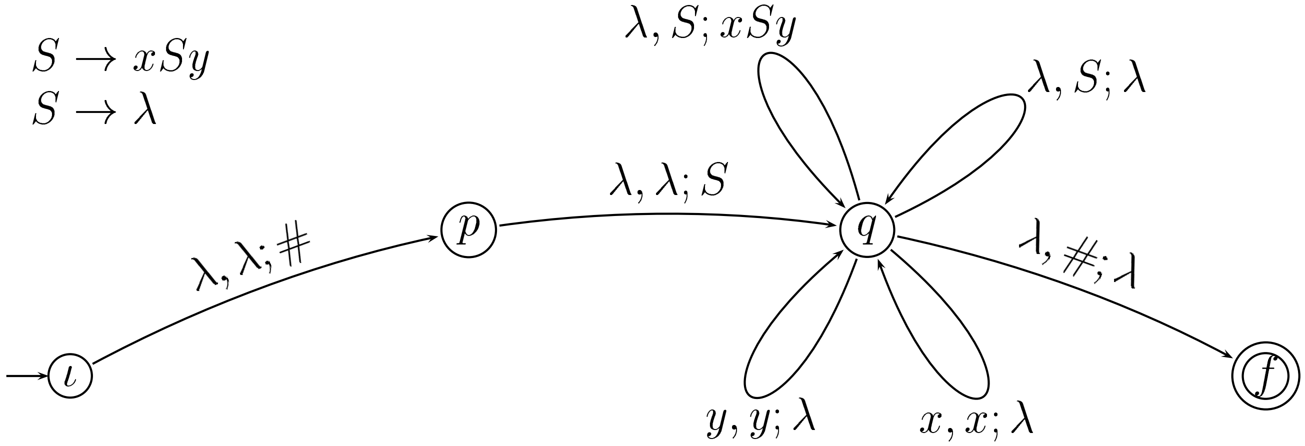


Figura 2.25: Una g.l.c. y diagrama de autómata de pila asociado a ella.

Una traducción del diagrama anterior a un fragmento de "programa" imperativo es la siguiente:

```
Estado := ι;
apila(#);
Estado := p;
apila(S);
Estado := q;
mientras tope_de_pila ≠ # haz
   caso tope_de_pila de
      S: (saca(S) y apila(xSy)) ó saca(S);
      x: saca(x), lee(x);
      y: saca(y), lee(y);
   fin caso
fin mientras;
saca(#);
Estado := f;
```

Aplicando el **principio de previsión** (*lookahead principle*), se puede eliminar el indeterminismo del programa anterior así:

```
Estado := ι;
apila(#);
Estado := p;
apila(S);
Estado := q;
lee(Símbolo);
mientras tope_de_pila ≠ # haz
   caso tope_de_pila de
      S: si Símbolo = x entonces saca(S) y apila(xSy),
            si_no saca(S);
      x: si Símbolo = x entonces saca(x) y lee(Símbolo),
            si_no sal_a_rutina_de_error;
      y: si Símbolo = y entonces saca(y) y lee(Símbolo),
            si_no sal_a_rutina_de_error;
   fin caso
fin mientras;
saca(#);
si Símbolo ≠ fin_de_cadena entonces sal_a_rutina_de_error;
Estado := f;
```

Sin embargo, para gramáticas como la siguiente:

$$S \to xSz \qquad S \to xyTyz \qquad T \to \lambda$$

no bastaría ver hacia adelante solo un símbolo, sería necesario observar dos. Si un analizador similar al anterior —determinista, que lee de izquierda a derecha la cadena bajo análisis y trata de producir para ésta una derivación *por la izquierda*— requiere adelantar la lectura de k símbolos, se dice que es un **analizador sintáctico** $LL(k)$. $\{x^n : n \in \mathbb{N}\} \cup \{x^n y^n : n \in \mathbb{N}\}$ es un lenguaje l.c.d. que no es reconocible por ningún analizador $LL(k)$.

A continuación tenemos una gramática y una tabla de análisis sintáctico $LL(1)$ para ella:

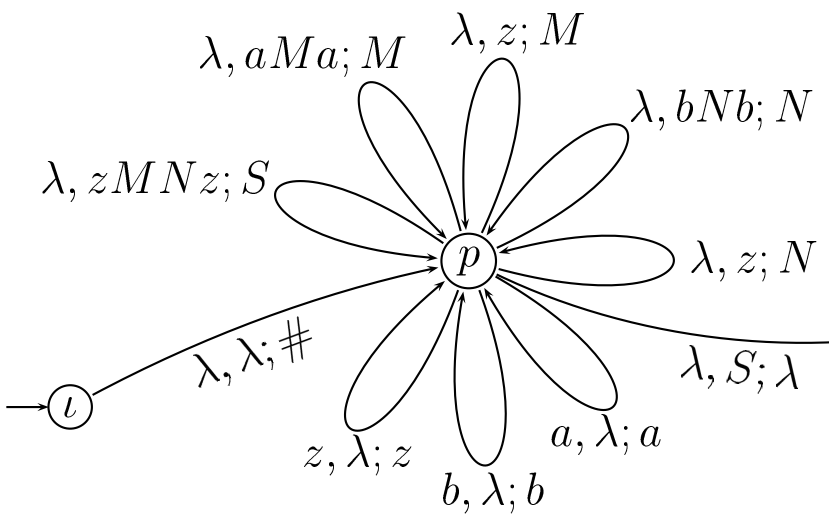
$$
\begin{aligned}
S &\to zMNz \\
M &\to aMa \\
M &\to z \\
N &\to bNb \\
N &\to z
\end{aligned}
$$

	a	b	z	EOS
S	`error`	`error`	$zMNz$	`error`
M	aMa	`error`	z	`error`
N	`error`	bNb	z	`error`

La siguiente es una rutina genérica para análisis sintáctico $LL(1)$:

```
apila(S);
lee(Símbolo);
mientras pila_no_vacía haz
   caso tope_de_pila de
      terminal:
        si tope_de_pila = Símbolo
          entonces desapila y lee(Símbolo),
          si_no sal_a_rutina_de_error;
      no_terminal:
        si tabla[tope_de_pila, Símbolo] ≠ error
          entonces cambia tope por tabla[tope_de_pila, Símbolo],
          si_no sal_a_rutina_de_error;
   fin caso
fin mientras;
si Símbolo ≠ fin_de_cadena entonces sal_a_rutina_de_error;
```

Analizadores sintácticos $LR(k)$



$S \to zMNz$
$M \to aMa$
$M \to z$
$N \to bNb$
$N \to z$

Figura 2.32: Otro diagrama de autómata de pila para la g.l.c. mostrada.

Estado	Pila	Resto de la entrada
ι	$\vert$	$zazabzbz$
p	$\vert\#$	$zazabzbz$
p	$\vert\#z$	$azabzbz$
p	$\vert\#za$	$zabzbz$
p	$\vert\#zaz$	$abzbz$
p	$\vert\#zaM$	$abzbz$
p	$\vert\#zaMa$	$bzbz$
p	$\vert\#zM$	$bzbz$
p	$\vert\#zMb$	zbz
p	$\vert\#zMbz$	bz
p	$\vert\#zMbN$	bz
p	$\vert\#zMbNb$	z
p	$\vert\#zMN$	z
p	$\vert\#zMNz$	
p	$\vert\#S$	
q	$\vert\#$	
f	$\vert$	

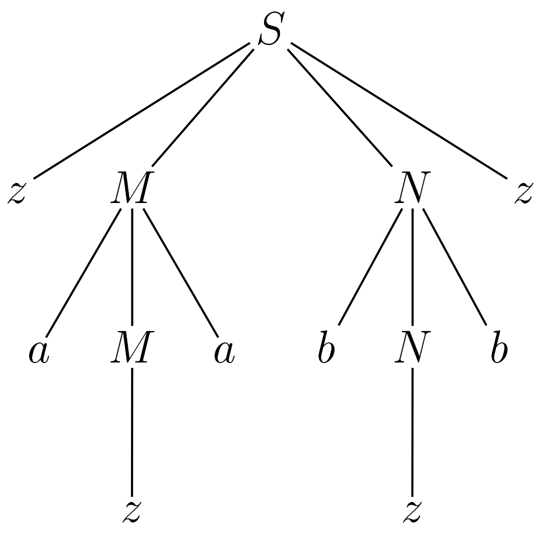


Figura 2.33: Análisis de la cadena $zazabzbz$ por el autómata de la Fig. 2.32.

	a	b	z	EOS	S	M	N
①			`shift` ②		⑭		
②	`shift` ③		`shift` ⑦			④	
③	`shift` ③		`shift` ⑦			⑧	
④		`shift` ⑤	`shift` ⑨				⑥
⑤		`shift` ⑤	`shift` ⑨				⑩
⑥			`shift` ⑪				
⑦	$M \to z$	$M \to z$	$M \to z$				
⑧	`shift` ⑫						
⑨		$N \to z$	$N \to z$				
⑩		`shift` ⑬					
⑪				$S \to zMNz$			
⑫	$M \to aMa$	$M \to aMa$	$M \to aMa$				
⑬		$N \to bNb$	$N \to bNb$				
⑭				`acepta`			

Figura 2.34: Tabla de análisis sintáctico $LR(1)$ para la gramática de la Figura 2.32.

Un algoritmo de análisis sintáctico $LR(1)$.

```
Ficha := 1;
apila(Ficha);
lee(Símbolo);
ElementoTabla := Tabla[Ficha,Símbolo]
mientras ElementoTabla ≠ acepta haz
  si ElementoTabla es un shift
    entonces
      principio
        apila(Símbolo);
        Ficha := ElementoTabla.Estado;
        apila(Ficha);
        lee(Símbolo)
      fin
    si_no
      si ElementoTabla es una reducción
        entonces
          principio
            desapila(lado derecho de ElementoTabla.Producción);
            Ficha := tope_de_pila;
            apila(lado izquierdo de ElementoTabla.Producción);
            Ficha := Tabla[Ficha,lado izquierdo de
                                       ElementoTabla.Producción];
            apila(Ficha)
          fin
      si_no
        si ElementoTabla está en blanco
          entonces sal_a_rutina_de_error;
  ElementoTabla := Tabla[Ficha,Símbolo]
fin mientras;
si Símbolo ≠ EOS entonces sal_a_rutina_de_error;
vacía_pila
```

Pila	Resto de la entrada
$\mid$	$zazabzbz$
$\mid$ ①	$zazabzbz$
$\mid$ ① z ②	$azabzbz$
$\mid$ ① z ② a ③	$zabzbz$
$\mid$ ① z ② a ③ z ⑦	$abzbz$
$\mid$ ① z ② a ③ M ⑧	$abzbz$
$\mid$ ① z ② a ③ M ⑧ a ⑫	$bzbz$
$\mid$ ① z ② M ④	$bzbz$
$\mid$ ① z ② M ④ b ⑤	zbz
$\mid$ ① z ② M ④ b ⑤ z ⑨	bz
$\mid$ ① z ② M ④ b ⑤ N ⑩	bz
$\mid$ ① z ② M ④ b ⑤ N ⑩ b ⑬	z
$\mid$ ① z ② M ④ N ⑥	z
$\mid$ ① z ② M ④ N ⑥ z ⑪	
$\mid$ ① S ⑭	
$\mid$	

Figura 2.36: Análisis de la cadena $zazabzbz$ con el algoritmo anterior y la tabla de la Figura 2.34.

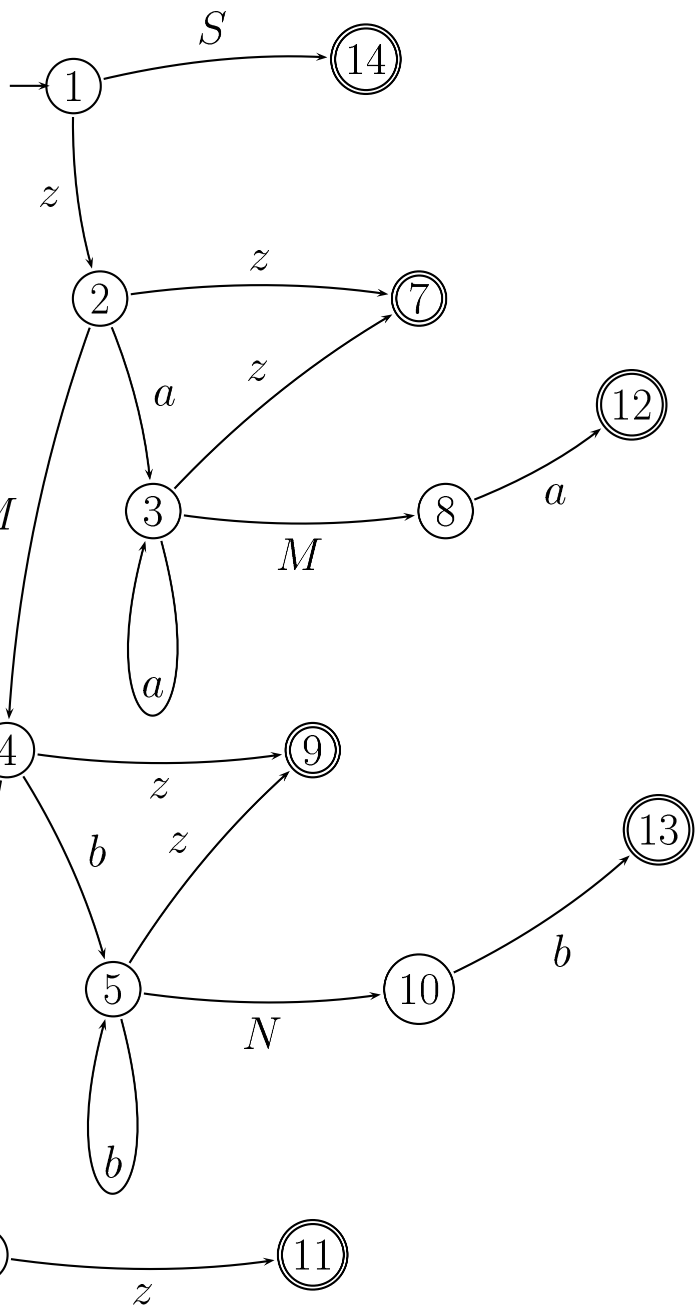
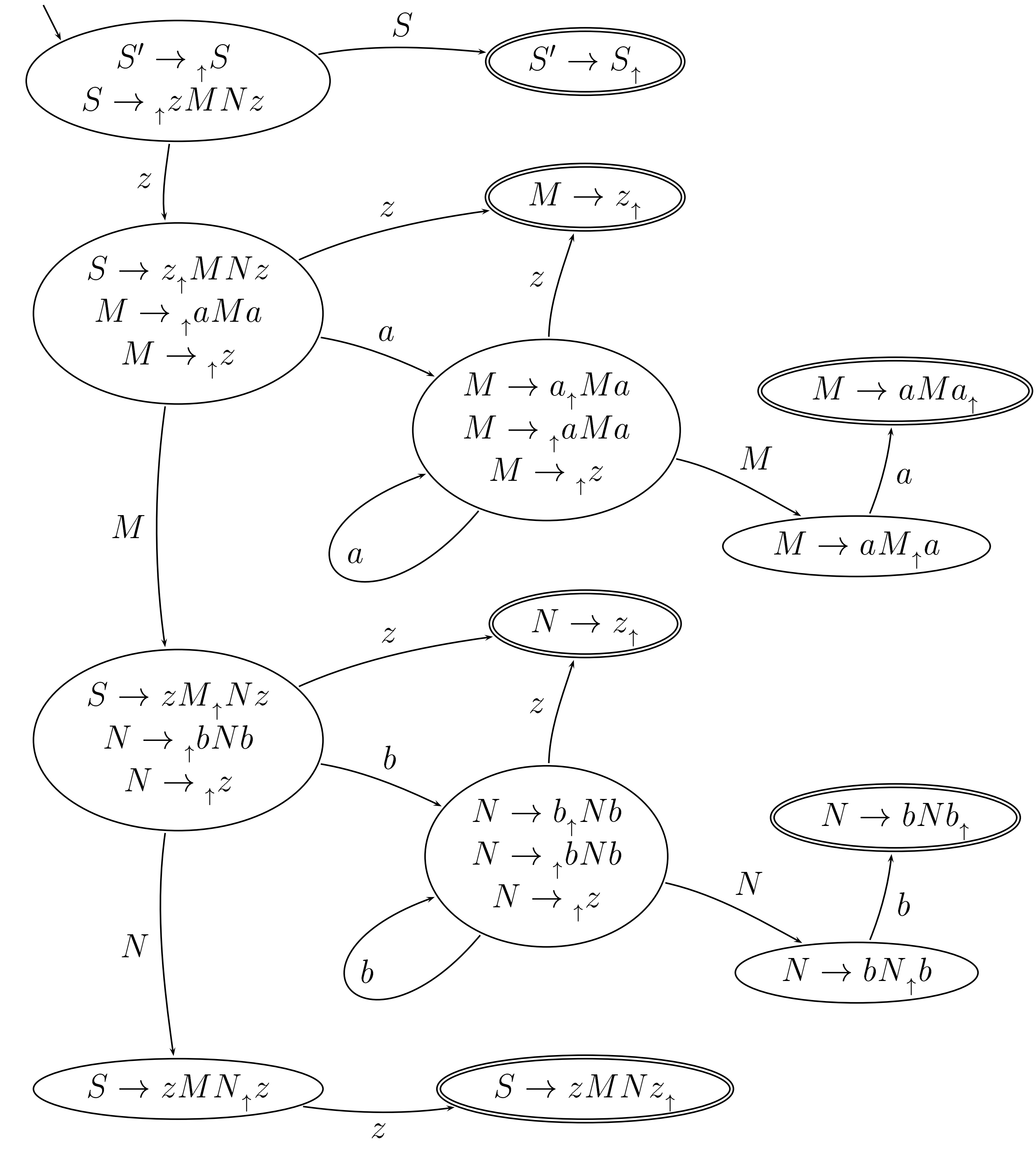


Figura 2.37: Autómata finito para la construcción de la tabla de la Figura 2.34.



Construcción del a.f. de la figura 2.37. Cada nodo consiste de un conjunto *cerrado* de producciones *marcadas*. Se marcan los lados derechos de las producciones mediante una ocurrencia del símbolo $\uparrow$. Si la marca aparece al principio, se dice que la producción es *inicial*; si aparece al final, se le llama *terminal*. Un conjunto de producciones es cerrado si tiene a todas las producciones iniciales cuyo lado izquierdo sea un símbolo no terminal que suceda a la marca de alguna producción en el conjunto. El nodo inicial es la cerradura de $\{S' \to {}_\uparrow S\}$, a partir de éste se construye recursivamente el diagrama, de la manera siguiente. Por cada subconjunto X de producciones en un nodo que tengan el mismo símbolo α enseguida de la marca, se agrega un arco de salida etiquetado con α, incidente en la cerradura del conjunto que se obtiene avanzando la marca una posición en cada producción en X. (En el presente diagrama cada X es singular). El método sirve para gramáticas no ambiguas y sin producciones λ.

Analizadores sintácticos $LR(k)$ versus $LL(k)$

$$
\begin{aligned}
&S \to X\\
&S \to Y\\
&S \to \lambda\\
&X \to xX\\
&X \to x\\
&Y \to xYy\\
&Y \to xy
\end{aligned}
$$

	x	y	EOS	S	X	Y
①	`shift` ②		$S \to \lambda$	⑨	⑧	⑦
②	`shift` ②	`shift` ⑥	$X \to x$		③	④
③			$X \to xX$			
④		`shift` ⑤				
⑤		$Y \to xYy$	$Y \to xYy$			
⑥		$Y \to xy$	$Y \to xy$			
⑦			$S \to Y$			
⑧			$S \to X$			
⑨			`acepta`			

Figura 2.38: Gramática y tabla de análisis sintáctico $LR(1)$ para el lenguaje $\{x^n : n \in \mathbb{N}\} \cup \{x^n y^n : n \in \mathbb{N}\}$.

Los analizadores $LR(k)$ corresponden a los lenguajes l.c.d.

Para el lenguaje $\{x^n y^n : n \in \mathbb{N}^+\} \cup \{x^n y^{2n} : n \in \mathbb{N}^+\}$ no hay analizador sintáctico $LR(k)$.
