INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE FÍSICA Y MATEMÁTICAS

Autómatas de pila y gramáticas independientes del contexto

Tesis que, para obtener el título de Licenciado en Física y Matemáticas, presenta

Magaly González Mota

Director: Eduardo Virueña Silva

México D. F., junio del 2007 .

Con cariño:

Para mis padres Juan de la Cruz y Ma. Elva por todo el apoyo que siempre he tenido

Para Jean Paul, Mauricio, Lizette e Ivan que me ayudarón para concluir este proyecto

Índice general

Introducción

En esta tesis se estudiara la relación entre los autómatas de pila y las gramáticas independientes del contexto. La forma de abordar el problema será tratar de encontrar un procedimiento que nos permita construir una gramática independiente del contexto, a apartir de un autómata de pila, sin perder de vista que ambas estructuras deberán reconocer o generar el mismo lenguaje. De la misma forma también se buscará un procedimiento para construir un autómata de pila a partir de una gramática independiente del contexto.

Primero será necesario dar unas definiciones básicas, que utilizaremos para estudiar o demostrar algunas propiedades de los autómatas de pila; ya en el segundo capítulo se definirá la estructura que motiva nuestro estudio así como sus características, para así, poder encontrar algoritmos que nos permitan realizar las construcciones necesarias.

Los dos últimos capítulos nos proporcionan diferentes formas de caracterizar las gramáticas independientes del contexto y su estructura, y las diferentes clases de autómatas de pila, con esto y los procesos de construcción se tendrán varias formas para estudiar los lenguajes independientes del contexto.

Capítulo 1

Preliminares

1.1. Conceptos básicos

Si nosotros observamos las siguientes líneas podemos observar algunas cosas en común:

Programas escritos en lenguajes de alto nivel.

Palabras o frases en español.

Los números usados por una calculadora

Lo primero que podemos observar es que cada una de ellas está compuesta por una secuencia de símbolos que pertenecen a algún conjunto finito; para el caso de la segunda línea el conjunto $\{a, b, c, ..., x, y, z\}$ junto con todos los símbolos de puntuación correspondientes, de manera análoga para la tercera línea, donde el conjunto son los dígitos $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$. Los programas escritos en lenguajes de alto nivel también están compuestos por un conjunto finito de palabras reservadas, identificadores y símbolos especiales (como fin de archivo, abrir archivo, retorno de carro etc.)
En general cada una de ellas está formada por una secuencia de elementos de un conjunto finito no vacío llamado *alfabeto* que denotaremos por Σ, a los elementos de dicho alfabeto los llamaremos *letras* y los denotaremos por σ, además supondremos que las letras se escriben con un solo carácter.

Definición 1 *Sea* Σ *un alfabeto, una* palabra *sobre el alfabeto es una* n-ada *de caracteres de* Σ*, con* $n \geq 0$.

En caso de que $n = 0$ *la palabra será llamada palabra* vacía *y la denotaremos mediante la letra* λ.

Definición 2 *Un conjunto de palabras lo llamaremos* lenguaje*. El lenguaje compuesto por todas las palabras sobre el alfabeto, se le conoce como* diccionario *y se denota por* Σ^*.

Observación 1 Σ^* *es a lo sumo numerable.*

Entre los elementos de un lenguaje es posible hacer operaciones, a continuación describimos algunas de ellas:

Definición 3 *Sea* Σ *un alfabeto, se define la* concatenación *de palabras como una operación:*

$$\cdot : \Sigma^* \times \Sigma^* \to \Sigma^*$$

$$\forall \omega, \varphi \in \Sigma^* : \omega \cdot \varphi = (\omega_1, \omega_2, \ldots, \omega_n \varphi_1, \varphi_2, \ldots, \varphi_n)$$

En la palabra resultante, a la palabra ω la llamaremos *prefijo* y a la palabra φ, la llamaremos *sufijo*

Observación 2 *Dados* $\lambda, \omega \in \Sigma^*$:

$$\lambda \cdot \omega = \omega \cdot \lambda = \omega$$

Definición 4 *Sea* Σ *un alfabeto y* $\omega = (\omega_1, \omega_2, ..., \omega_n)$ *una palabra sobre* Σ. *La* longitud *de una palabra de* Σ *es una función:* $|\cdot| : \Sigma^* \to \mathbb{N}$ *definida así:*

$$|\omega| = n$$

Observación 3

1. $|\lambda| = 0$
2. *si* $\varphi, \omega \in \Sigma$*:* $|\omega \cdot \varphi| = |\omega| + |\varphi|$

Demostración:

Procederemos por inducción sobre φ
Base: Si $\varphi = \lambda$ entonces

$$|\omega \cdot \varphi| = |\omega \cdot \lambda| = |\omega| + 0 = |\omega| + |\lambda| = |\omega| + |\varphi|$$

Hipótesis de inducción: Supongamos que para cualquier palabra α,$|\alpha| \leq n$ se tiene que:

$$|\omega \cdot \alpha| = |\omega| + |\alpha|$$

Ahora sea $\psi = \alpha \cdot \beta$ una palabra de longitud $n+1$ con $\alpha \in \Sigma^*$ y $\beta \in \Sigma$

$$|\varphi \cdot \psi| = |\varphi \cdot \alpha \cdot \beta| = |\varphi \cdot \alpha| + 1 = |\varphi| + n + 1 = |\varphi| + |\alpha \cdot \beta| = |\varphi| + |\psi|$$

□

Definición 5 *Se define la* concatenación *de dos conjuntos* $A, B \subset \Sigma^*$ *como:*

$$A \cdot B = \{\alpha \cdot \beta | \alpha \in A, \beta \in B\}$$

Notación 1 *Denotaremos* $\{\lambda\} = \boldsymbol{\lambda}$

Observación 4

1. *La concatenación de palabras es asociativa y tiene un idéntico.*
2. *La concatenación de conjuntos es asociativa tiene un idéntico, y se distribuye con respecto a la unión arbitraria.*

Demostración:

1. Por la observación (2) el idéntico en la concatenación de palabras es λ
2. Sean $\varphi\omega \in \Sigma^*$ y $\varphi = \alpha \cdot \beta$ tenemos que:

$$\varphi \cdot \omega = (\alpha \cdot \beta) \cdot \omega = (\alpha_1\alpha_2 \cdots \alpha_n \cdot \beta_1\beta_2 \cdots \beta_m) \cdot \omega_1\omega_2 \cdots \omega_r$$

$$= \alpha_1\alpha_2 \cdots \alpha_n \cdot \beta_1\beta_2 \cdots \beta_m \cdot \omega_1\omega_2 \cdots \omega_r$$

$$= \alpha_1\alpha_2 \cdots \alpha_n \cdot (\beta_1\beta_2 \cdots \beta_m \cdot \omega_1\omega_2 \cdots \omega_r) = \alpha \cdot \psi$$

con $\psi = \beta \cdot \omega$. Por lo tanto, la concatenación de palabras es asociativa.

3. Sean $A \subset \Sigma^*$ se tiene que:

$$A \cdot \boldsymbol{\lambda} = \{\alpha \cdot \lambda | \alpha \in A \ \ y \ \ \lambda \in \boldsymbol{\lambda}\} = A$$
$$\boldsymbol{\lambda} \cdot A = \{\lambda \cdot \alpha | \lambda \in \boldsymbol{\lambda} \ \ y \ \ \alpha \in A\} = A$$

Entonces $\boldsymbol{\lambda}$ es el idéntico en la concatenación de conjuntos.

4. Sean $A, B, C \subset \Sigma^*$ se tiene que:

$$\begin{aligned}(A \cdot B) \cdot C &= \{\psi\omega | \psi \in (A \cdot B), \omega \in C\} = \{(\alpha\beta)\omega | \alpha\beta \in A \cdot B, \omega \in C\} \\ &= \{\alpha\beta\omega | \alpha \in A, \beta \in B, \omega \in C\} = \{\alpha(\beta\omega) | \alpha \in A, (\beta\omega) \in B \cdot C\} \\ &= \{\alpha \cdot \varphi | \alpha \in A, \varphi \in (B \cdot C)\} = A \cdot (B \cdot C)\end{aligned}$$

5. La concatenación se distribuye con respecto a la unión arbitraria, es decir deseamos probar que:

$$A \cdot \bigcup_{i \in I} B_i = \bigcup_{i \in I} (A \cdot B_i)$$

Sea $\omega \in A \cdot \bigcup_{i \in I} B_i$ entonces existe $\varphi \in A$ y $\psi \in \bigcup_{i \in I} B_i$ tal que $\omega = \varphi\psi$ así para algún índice $i_0 \in I$ tenemos que $\psi \in B_{i_0}$ entonces

$$\varphi\psi \in A \cdot B_{i_0} \subseteq \bigcup_{i \in I} (A \cdot B_i)$$

como ω es arbitraria

$$\left(A \cdot \bigcup_{i \in I} B_i \right) \subset \bigcup_{i \in I} (A \cdot B_i)$$

ahora si $\omega \in \bigcup_{i \in I} (A \cdot B_i)$ entonces $\exists\, i_0 \in I$ tal que

$$\omega \in A \cdot B_{i_0}$$

entonces existen $\varphi \in A$ y $\psi \in B_{i_0}$ más aún $\psi \in \bigcup_{i \in I} B_i$ tal que $\omega = \varphi\psi$ así $\varphi\psi \in A \cdot \bigcup_{i \in I} B_i$
ahora como ω es arbitraria

$$\bigcup_{i \in I} (A \cdot B_i) \subseteq A \cdot \bigcup_{i \in I} B_i$$

por lo tanto

$$\bigcup_{i \in I} (A \cdot B_i) = A \cdot \bigcup_{i \in I} B_i$$

□

1.2. Expresiones regulares y Autómatas finitos

Definición 6 *Las* Expresiones regulares *sobre un alfabeto, se definen así:*

1. $\emptyset$ *es una expresión regular.*
2. $\{\lambda\}$ *es una expresión regular y se escribe:* $\boldsymbol{\lambda}$
3. $\forall \sigma \in \Sigma : \sigma$ *es expresión regular y se escribe:* $\boldsymbol{\sigma}$.

Ahora, si α *y* β *son expresiones regulares, entonces también lo serán:*

4. $\alpha \cup \beta$ *que escribiremos:* $\alpha + \beta$
5. $\alpha \cdot \beta$
6. α^*

Solo las expresiones que se obtienen por composición finita de las reglas anteriores son expresiones regulares.

Definición 7 *La* inversión *de una palabra se define mediante la siguiente función:*

$$.^R : \Sigma^* \to \Sigma^*$$

1. $\lambda^R := \lambda$
2. $\forall \sigma \in \Sigma, \forall \varphi \in \Sigma^*$*:* $(\sigma \cdot \varphi)^R = \varphi^R \cdot \sigma$
   es decir, la palabra escrita al revés
3. $\forall A \subset \Sigma^* \ A^R := \left\{\omega^R | \omega \in A\right\}$

Proposión 1 *Sea* α *una expresión regular. Entonces* α^R *también es una expresión regular*
Demostración:
Se procederá por inducción sobre la construcción de expresiones regulares

1. $\emptyset^R = \emptyset$
2. $\boldsymbol{\lambda}^R = \{\lambda\}^R = \left\{\lambda^R\right\} = \{\lambda\} = \boldsymbol{\lambda}$

3. Sea $\sigma \in \Sigma$ entonces por la definición (7)

$$\sigma^R = (\sigma\lambda)^R = \lambda^R\sigma = \lambda\sigma = \sigma$$

por lo tanto

$$\boldsymbol{\sigma}^R = \{\sigma\}^R = \{\sigma^R\} = \{\sigma\} = \boldsymbol{\sigma}$$

$(1), (2)$ y (3) son expresiones regulares.

4. Supongamos que α, β, son expresiones regulares y que α^R, β^R también lo son, entonces

$$(\alpha + \beta)^R = \{\omega^R | \omega \in \alpha + \beta\} = \{\omega^R | \omega \in \alpha\} \cup \{\omega^R | \omega \in \beta\} = \alpha^R \cup \beta^R$$

por lo tanto $(\alpha + \beta)^R$ es expresión regular.

5. Si φ, ψ son palabras queremos probar que $\psi^R\varphi^R = (\varphi\psi)^R$
Base: si $\varphi = \lambda$

$$(\varphi\psi)^R = (\lambda\psi)^R = \psi^R = \psi^R\lambda = \psi^R\lambda^R = \psi^R\varphi^R$$

Hipótesis de inducción: Sea $\alpha \in \Sigma^*$ con $|\alpha| = n$, $n \geq 1$ entonces

$$(\varphi\psi)^R = \{\beta^R | \beta \in \varphi\psi\} = \{\beta^R | \beta = a \cdot b, a \in \varphi, b \in \psi\} = \psi^R\varphi^R$$

Ahora para $\alpha, \sigma \in \Sigma^*$

$$((\alpha\sigma)\,\psi)^R = (\sigma\,(\alpha\psi))^R = (\alpha\psi)^R\,\sigma$$

por la hipótesis de inducción

$$(\alpha\psi)^R\,\sigma = \psi^R\alpha^R\sigma = \psi^R\,(\sigma\alpha)^R$$

que es una expresión regular.

6. $(\alpha^*)^R = \left(\alpha^R\right)^*$

base $(\alpha^0)^R = \boldsymbol{\lambda}^n = \boldsymbol{\lambda} = \left(\alpha^R\right)^0$

hipótesis de inducción $(\alpha^*)^R = \left(\alpha^R\right)^*$

Para $n+1$

$$\left(\alpha^{n+1}\right)^R = \left(\alpha^n \alpha\right)^R = \alpha^R \left(\alpha^n\right)^R = \alpha^R \left(\alpha^R\right)^n = \left(\alpha^R\right)^{n+1}$$

ahora

$$\left(\alpha^*\right)^R = \left(\bigcup_{n=0}^{\infty} \alpha^n\right)^R = \bigcup_{n=0}^{\infty} \left(\alpha^n\right)^R = \bigcup_{n=0}^{\infty} \left(\alpha^R\right)^n = \left(\alpha^n\right)^*$$

que es una expresión regular.

Por lo tanto, la inversión de una expresión regular es una expresión regular.

□

Definición 8 *Un* Autómata Finito Determinístico *(AFD) es una quintupla:* $M = (Q, \Sigma, q_0, f, F)$*, Donde*

Q *es un conjunto finito no vacío (los elementos de* Q *son llamados estados)*

Σ *es un alfabeto.*

$q_0 \in Q$*, es un estado al que llamaremos* estado inicial*.*

f *es una función* $Q \times \Sigma \to Q$ *que se llama* función de transición*.*

$F \subset Q$ *es un conjunto de estados a los cuales llamaremos* estados finales*.*

Un autómata puede ser representado mediante un grafo dirigido el cual se conoce como diagrama de transiciones, donde los vértices del mismo corresponden a los estados del autómata, en el caso del estado inicial este tendrá una flecha que apunta hacia el, y los estados finales se representarán mediante un círculo con línea doble; si existe una transición del estado q al p sobre la entrada a entonces existe un arco con etiqueta a que va del estado q al estado p en el diagrama de transición. El autómata acepta una cadena ω si la secuencia de transiciones correspondientes a los símbolos de ω conducen del estado inicial a un estado final.

Definición 9 *Sea* $M = (Q, \Sigma, q_0, f, F)$ *un AFD definimos la* iteración *de* f *sobre* Σ^* *así:*

$f^* : Q \times \Sigma \to Q$

$\forall q \in Q : f^*(q, \lambda) := q$

$\forall q \in Q \ , \forall \sigma \in \Sigma \ y \ \forall \varphi \in \Sigma^* \ : f^*(q, \sigma\varphi) := f^*(f(q, \sigma), \varphi)$

Definición 10 *Sea $M = (Q, \Sigma, q_0, f, F)$ un AFD el* lenguaje *de M se define como:*

$$L(M) := \{w \in \Sigma^* | f^*(q_0, w) \in F\}$$

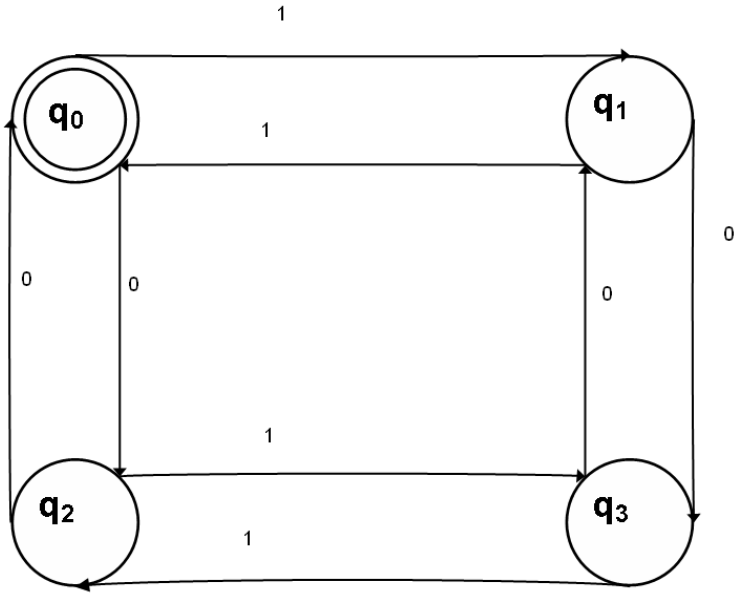

Figura 1.1: figura 1

Revisando el autómta de la figura (1.1) tenemos que:

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$

$F = \{q_0\}$

y la función de transición se describe como sigue:

f	0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Definición 11 *Un conjunto $A \in \Sigma^*$ se dice* regular *si existe un AFD M tal que $L(M) = A$*

Observación 5 *Sean $L(A)$ y $L(B)$ dos lenguajes sobre el alfabeto Σ, entonces $L(A) = L(B)$ si y sólo si $L(A) \subseteq L(B)$ y $L(B) \subseteq L(A)$.*

Observación 6 *Sean $L(A)$ y $L(B)$ dos lenguajes sobre el alfabeto Σ, entonces*

$$(L(A) \cdot L(B))^R = L(B)^R \cdot L(A)^R.$$

Definición 12 *Dos autómatas finitos se dicen* equivalentes *si reconocen el mismo lenguaje es decir: Dados $M_p = (Q, \Sigma, p_0, f, F), M_q = (Q, \Sigma, q_0, f, F)$ AFD*

$$M_P \equiv M_q \Leftrightarrow L(M_p) = L(M_q)$$

Observación 7 *Sea M un autómata finito, entonces existe una expresión regular r para la cual $L(r) = L(M)$*

Observación 8 *Un lenguaje es regular si y sólo si es aceptado por un autómata finito*
Ahora, si se modifica el modelo del autómata finito, para permitirle ninguna, una o más transiciones de un estado sobre el mismo símbolo de entrada, al nuevo modelo lo conoceremos como autómata finito no determinístico.

Definición 13 *Un* Autómata Finito No Determinístico (AFND) *es una quintupla $M = (Q, \Sigma, I, \mathfrak{R}, F)$. Donde*

Q es un conjunto de estados

Σ *es un alfabeto.*

$I \subset Q$ *es un conjunto de estados a los cuales llamaremos* estados iniciales.

$\mathfrak{R}$ *es una relación sobre* $Q \times \Sigma \times Q$ *que se llama* relación de transición.

$F \subset Q$ *es un conjunto de estados a los cuales llamaremos* estados finales.

Observación 9 *Si I se reduce a un solo estado y $\mathfrak{R}$ es tal que sus elementos pueden escribirse mediante una función como la descrita en las definiciones (8) y (9) entonces el autómata será determinístico.*

Definición 14 *Sea M un AFND y sea $\omega \in \Sigma^*$ un* camino *que empieza en q_0, inducido por $\omega = \sigma_1\sigma_2, \cdots, \sigma_n$ en un autómata es una $n+1$-ada de estados $(q_0, q_1, \cdots, q_n)$ tales que $\forall k \in [1, 2, \cdots, n]$, $(q_{k-1}, \sigma_k, q_k) \in \mathfrak{R}$. También diremos que el camino empieza en q_0 e induce λ es (q_0).*

Definición 15 *Una palabra ω se dice* reconocida *por un AFND M, si ω induce un camino que empieza en un estado inicial y termina en algún estado final.*

Definición 16 *El* lenguaje *del AFND M, es el conjunto de todas las palabras en Σ^* que son reconocidas por M*

$L(M) = \{\omega \in \Sigma^* | \omega \text{ induce un camino } (q_0, q_1, \cdots, q_n) \text{ tal que } q_o \in I \text{ y } q_n \in F\}$

Observación 10 *Dado un AFND $M = (Q, \Sigma, I, \mathfrak{R}, F)$ entonces existe un AFD M' tal que $L(M) = L(M')$.*

Ejemplo 1 *Sea $M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$ un AFND en el que :*

$\delta(q_0, 0) = \{q_0, q_1\}$

$\delta(q_0, 1) = \{q_1\}$

$\delta(q_1, 0) = \emptyset$

$\delta(q_1, 1) = \{q_0, q_1\}$

Podemos construir una AFD $M' = (Q, \{0,1\}, \delta', [q_0], F)$ *que acepte a* $L(M)$ *de la siguiente forma* Q *esta dado por todos los subconjuntos de* $\{q_0, q_1\}$*, a los cuales denotaremos así:*

$$[q_0], [q_1], [q_0, q_1], \emptyset$$

como $\delta(q_0, o) = \{q_o, q_1\}$
tenemos: $\delta'([q_0], 0) = [q_0, q_1]$ *de la misma forma:*

$\delta'([q_0], 1) = [q_1]$

$\delta'([q_1], 0) = \emptyset$

$\delta'([q_1], 1) = [q_0, q_1]$

$\delta'(\emptyset, 0) = \delta'(\emptyset, 1) = \emptyset$

El conjunto de estado finales es $\{[q_1], [q_0, q_1]\}$

1.3. Lenguajes y Gramáticas

El lenguaje es el medio de comunicación entre los seres humanos a través de signos orales y escritos que poseen un significado. Para que exista el lenguaje se requieren ciertos factores como la sintaxis que da estructura al lenguaje y la semántica que le da significado al lenguaje. Además de manera conjunta con los lenguajes tenemos la gramática que estudia los elementos de un lenguaje y sus combinaciones. Así como es importante podernos comunicar con otras personas, actualmente también es importante podernos comunicar con las computadoras; es decir establecer un lenguaje y una gramática para facilitar el uso de las mismas. Introduciremos de manera natural el concepto de gramática mediante una analogía con el lenguaje español y su gramática.

Ejemplo 2 *Suponga que tenemos la siguiente frase:*

La gata gris duerme en la cama diariamnte

Observamos que la frase se divide en dos partes escenciales sujeto *y* predicado*; el sujeto a su vez se divide en* artículo,sustantivo *y* adjetivo*; y el predicado se divide en* verbo, preposición, artículo *y* advervio*. Nosotros podemos decir*

que existe una variable llamada $< frase >$ *que genera 2 variables* $< sujeto >$ *y* $< predicado >$ *es decir:*

$$< frase > \longrightarrow < sujeto >< predicado >$$

(sustituimos la palabra produce por la flecha) a su vez las variables

$$< sujeto > \longrightarrow < articulo >< sustantivo >< adjetivo >$$

$$< predicado > \longrightarrow < verbo >< preposicion >< articulo >< advervio >$$

por último las variables

$$< articulo > \longrightarrow la \mid el$$

$$< sustantivo > \longrightarrow gata \mid cama$$

$$< adjetivo > \longrightarrow gris$$

$$< verbo > \longrightarrow duerme$$

$$< preposicion > \longrightarrow en$$

$$< advervio > \longrightarrow diariamente$$

a las variables la, el, gata, cama, gris, duerme, en, diariamente *las llamamos* símbolos terminales*, mientras que a las variables escritas entre* $< \ >$ *las conocemos como* símbolos no terminales*; el proceso que sustituye unas variables por otras se le conoce como* producción *y a la variables* $< frase >$ *se le conoce como* símbolo inicial.
De manera formal tenemos

Definición 17 *Una* Gramática *es una cuadrupla* $G = (N, T, P, S)$ *donde*

N *es un alfabeto a cuyos símbolos llamamos* no terminales.

T *es un alfabeto a cuyos símbolos llamamos* terminales.

P *es un subconjunto finito de* $(N \setminus T)^* \times N^* \to N^*$ *y a los elementos* (u, v)*de* P *los conocemos como* producciones *de* G.

S *es el* símbolo inicial.

En el párrafo previo a la definición se describe cuales son los conjuntos N, y T así como cual es el elemento S y las producciones correspondientes al ejemplo.

Notación 2 *A los símbolos no terminales se les representa mediante letras mayúsculas, mientras que los símbolos terminales serán representados mediante letras minúsculas.*

Definición 18 *Para $\omega, \omega' \in N^*$ escribimos $\omega \Longrightarrow \omega'$ si existen $x, y \in N^*$ y una producción $u \longrightarrow v$ en P tal que*

$$\omega = xuy$$

y

$$\omega' = xvy$$

Decimos que ω deriva ω'
Escribimos $\omega \overset{*}{\Longrightarrow} z$ si $\omega = z$ o existen $\omega_1, \omega_2, \cdots, \omega_n$ con $n \geq 2$ en N^* tales que

$$\omega = \omega_1, z = \omega_n \quad y \quad \omega_i \Longrightarrow \omega_{i+1}$$

sin pérdida de generalidad a esta transformación la llamaremos *derivación* en G y decimos que ω *deriva* a z.

Definición 19 *El lenguaje $L(G)$ generado por G es el conjunto de palabras en T, que puede ser derivado a partir de S*

$$L(G) = \left\{ \omega \in T^* | S \overset{*}{\Longrightarrow} \omega \right\}$$

Ahora, regresando al ejemplo anterior observamos que las producciones pueden generar frases como la siguiente

El cama gris duerme en la gata diariamente

La cual no es semánticamente aceptable, pues carece de significado, pero es aceptada por la sintaxis de la gramática, debido a este comportamiento será necesario hacer modificaciones para que el lenguaje sólo reconozca oraciones que posean significado (es decir analizar el contexto de la oración).

Definición 20 *Una gramática es* regular *si cada producción P es de la forma*

$$\alpha \longrightarrow x\beta \ \ con \ \ (x \in T^*, \alpha, \beta \in N \setminus T)$$

o de la forma

$$\alpha \longrightarrow y \ \ con \ \ (\alpha \in N \setminus T, y \in T^*)$$

Decimos que una gramática Ges independiente del contexto *si todas las producciones son de la forma*

$$\alpha \longrightarrow z \ \ con \ \ z \in (T \cup N)^*$$

Ejemplo 3 *Sea $G = (N, T, P, S)$ donde $T = \{a, b\}$ $N \setminus T = \{S, B\}$ y P consiste de las producciones*

$$S \longrightarrow aSb \ \mid \ \lambda$$

entonces $L(G) = \{a^n b^n | n \geq 1\}$ que es un lenguaje independiente del contexto.

Ejemplo 4 *Sea $G = (N, T, P, S)$ con $T = \{a, b\}$, $N \setminus T = \{S\}$ y P tenga las producciones*

$$S \longrightarrow aSa \mid bSb \mid a \mid b \mid \lambda$$

Tenemos que $L(G) = \left\{\omega \in T^* | \omega = \omega^R\right\}$ que es el lenguaje de los palíndromos, en el alfabeto T

Capítulo 2

Autómatas de Pila

En el capítulo anterior se menciona que las expresiones regulares tienen asociado un autómata finito, de manera análoga las gramáticas independientes del contexto también tienen asociado un autómata al cual conoceremos como *Autómata de Pila*. Debido a esta similitud con las gramáticas independientes del contexto necesitaremos emplear dos estructuras que son fundamentales en la definición y uso de los autómatas de pila.

La primera es la *cinta de entrada* que es un arreglo en donde será guardada la cadena de símbolos terminales que acepta el autómata, se guardará un símbolo por localidad del arreglo y se tendrá una marca en la siguiente localidad del último símbolo escrito en la cinta; la segunda es una *pila*, en la cual escribiremos los símbolos no terminales, para que estos a su vez puedan ser sustituidos por los símbolos terminales en la cinta de entrada. En la pila el símbolo que se encuentre más a la derecha será el *tope de la pila*.

En los autómatas de pila se producen dos tipos de movimientos: el primero introduce o saca un símbolo de la pila, y dependiendo de cual sea la acción realizada el tope de la pila avanzará o retrocederá un lugar. El segundo movimiento no afecta directamente el tope de la pila, pero revisa un símbolo de la cinta de entrada, y recorre un lugar a la derecha la última posición a la cual apuntabamos en la cinta.

Es posible definir un lenguaje para los autómatas de pila para lo cual existen dos maneras; la primera consiste en definir el lenguaje aceptado como el conjunto de todas las entradas para las cuales una sucesión de movimientos

provoca que el autómata de pila vacíe su pila.

La segunda manera consiste en designar algunos estados como estados finales y definimos el lenguaje aceptado como el conjunto de todas las entradas para las cuales alguna selección de movimientos provoca que el autómata alcance un estado final. Ambas formas son equivalentes en el sentido de que si un conjunto es aceptado mediante el vaciado de la pila por algún autómata, puede ser aceptado mediante el acceso de un estado final por otro autómata y viceversa.

Definición 21 *Un* Autómata de Pila (AP) *se define como una sextupla*

$$M = (Q, S, U, P, I, F)$$

donde:

Q *es un conjunto finito de estados*

S *es un alfabeto al que llamaremos* alfabeto de entrada.

U *es un alfabeto al que llamaremos* alfabeto de pila .

P *es el programa de* M

$I \subseteq Q$ *es el conjunto de estado iniciales*

$F \subseteq Q$ *es el conjunto de estados finales.*

Definición 22 *La forma de representar un estado válido en un autómata de pila es la siguiente:* (q_i, φ, σ) *a la cual llamaremos* configuración*, donde el estado en que nos encontramos o estado de control esta dado por* q_i*,* φ *es el prefijo que se encuentra en la cinta de entrada y ya fue revisada y* σ *es la cadena contenida en la pila.*
El programa P tiene una secuencia finita de instrucciones con las siguientes formas

$q]scan(s, q')$

La cual aplicada en la configuración (q_i, φ, σ), (con φ la cadena ya revisada en la cinta de entrada) escribe el símbolo s en la primera

casilla a la derecha después del último símbolo de φ; y nos lleva al estado q_i', esta transformación se representa de la siguiente forma:

$$(q_i, \varphi, \sigma) \xrightarrow{s} (q_i', \varphi s, \sigma)$$

En otras palabras esta instrucción revisa el símbolo que entra en la cinta y lo coloca inmediatamente después de la palabra que ya se encontraba guardada.

$q]write(u, q')$

La cual aplicada en la configuración (q_i, φ, σ) mueve el tope de la pila una posición a la derecha y escribe un símbolo u en esa nueva posición, pasando al estado q_i' dicho movimiento es representado de la siguiente forma:

$$(q_i, \varphi, \sigma) \xrightarrow{w} (q_i', \varphi, \sigma u)$$

Este movimiento introduce a la pila un nuevo símbolo.

$q]read(u, q')$

La cual aplicada en la configuración $(q_i, \varphi, \sigma' u)$ mueve el tope de la pila a la izquierda y entra en el estado q_i', dicho movimiento es representado de la siguiente forma:

$$(q_i, \varphi, \sigma) \xrightarrow{r} (q_i', \varphi, \sigma)$$

Este movimiento sacará un símbolo de la pila.

Notación 3 *Es posible representar una secuencia de movientos en un autómata de pila*

$$(q_0, \varphi_0, \sigma_0) \longrightarrow (q_1, \varphi_1, \sigma_1) \longrightarrow \cdots \longrightarrow (q_k, \varphi_k, \sigma_k)$$

donde cada movimiento es un movimiento read*,* write *o* scan *se denotará de la siguiente forma:*

$$(q_0, \varphi_0, \sigma_0) \Longrightarrow (q_k, \varphi_k, \sigma_k)$$

Un autómata de pila comienza su funcionamiento con el tope de la pila y de la cinta de entrada en la primera posición. La cadena que analiza pasa a través de una secuencia de movimientos, mientras la secuencia no sea rechazada; si en algún momento todos los símbolos de la cadena ya fueron revisados, la pila se encuentra vacía, y la última posición de la cinta de entrada corresponde a un estado final; entonces concluimos que la cadena es aceptada.

Definición 23 *Una configuración inicial válida para un autómata de pila es una configuración* (q, λ, λ) *en donde* q *es un estado inicial del autómata; y la configuración final es* (q', φ, λ)*, donde* q *es un estado final,* φ *la cadena escrita en la cinta de entrada, de donde decimos que una cadena es aceptada por el autómata sólo si* M *tiene la secuencia de movimientos:*

$$(q, \lambda, \lambda) \Longrightarrow (q', \varphi, \lambda)$$

Donde q *es el estado inicial,* q' *es un estado final; el conjunto de cadenas aceptadas será el lenguaje reconocido por el autómata.*

Ejemplo 5 *Sea* M_{cm} *un autómata de pila con* $S = \{a, b, c\}$ *y* $U = \{a, b\}$ *el alfabeto de la pila, el programa del autómata será el siguiente:*

1]scan (a,2)(b,3)(c,4)

2]write(a,1)

3]write(b,1)

4]scan (a,5)(b,6)

5]read (a,4)

6]read (b,4)

Donde el estado 1 es el estado inicial a menos que se especifique otra cosa; y el estado final es el 6.
En la figura a cada estado se le colocó una etiqueta la cual está relacionada con el nombre de la instrucción del programa que ejecuta.
El lenguaje reconocido por el autómata de pila es:

$$L_{cm} = \varphi c \varphi^R \varphi \in (a \cup b)^*$$

a este lenguaje lo conocemos como el lenguaje del reflejo con el centro marcado o lenguaje de los palíndromos con el centro marcado.

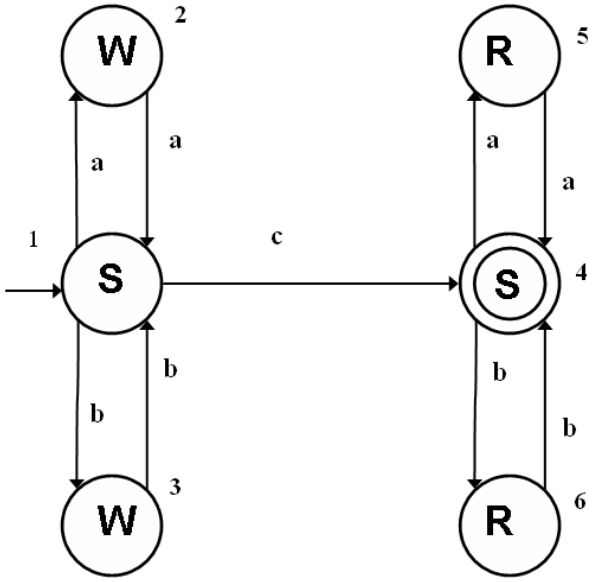

Figura 2.1:

Observamos que en este lenguaje la letra c está en el centro de la cadena y su sufijo es igual a la inversa de su prefijo, con lo cual aceptará cadenas como la siguiente $abcba$ mediante la siguiente secuencia de movimientos:

$(1, \lambda, \lambda) \overset{S}{\rightarrow} (2, a, \lambda) \overset{W}{\rightarrow} (1, a, a)$

$\overset{S}{\rightarrow} (3, ab, a) \overset{W}{\rightarrow} (1, ab, ab)$

$\overset{S}{\rightarrow} (4, abc, ab)$

$\overset{S}{\rightarrow} (6, abcb, ab) \overset{R}{\rightarrow} (4, abcb, a)$

$\overset{S}{\rightarrow} (5, abcba, a) \overset{R}{\rightarrow} (4, abcba, \lambda)$

La cual es una secuencia de movimientos aceptada por el autómata.
El lenguaje L_{cm} es generado por la gramática independiente del contexto:

$G_{cm} : \Sigma \to S$

$S \to aSa$

$S \to bSb$

$S \to c$

2.1. Movimientos propios y autómatas sin ciclos

En los autómatas de pila existen varias subclases entre las que se encuentran los autómatas *con buen comportamiento* los cuales sin pérdida de generalidad reconocen los mismos lenguajes reconocidos por toda la clase de los autómatas de pila; a esta subclase se le conoce como *Autómatas de pila propios* y a continuación se presenta una justificación de la equivalencia entre esta subclase y los autómatas de pila.

2.1.1. Autómatas de pila propios

Cuando se ejecuta el programa de un autómata de pila se pueden producir comportamientos que carecen de sentido e improductivos, en particular:

1. Revisar más alla del fin de la cadena en la cinta de entrada.

2. Intentar leer un símbolo como primer movimiento en la pila es decir, intentar mover el tope de la pila a la izquierda en el inicio de la pila.

3. Escribir un símbolo en la pila, y que el siguiente movimiento sea la lectura del mismo símbolo.

   $$\cdots (q, \varphi, \sigma) \xrightarrow{w} (q', \varphi, \sigma u) \xrightarrow{r} (q'', \varphi, \sigma)$$

   Observamos que el único efecto es movernos del estado q al estado q'', lo cual sólo puede ocurrir si el autómata tiene una instrucción de escritura

   $$q]write(u, q')$$

   para la cual la etiqueta del estado q' es la instrucción read.

4. Repetición infinita de movimientos de escritura (un ciclo).

Los primeros tres tipos se analizarán en esta sección, el último tipo de movimiento improductivo se analizará en la siguiente sección.

Definición 24 *En el programa de un autómata de pila M una instrucción de escritura es* impropia*, si el siguiente estado al que apunta está etiquetado con una instrucción de lectura. Un autómata se dice* propio *si no contiene movimientos impropios.*
Las instrucciones impropias pueden ser eliminadas sin que esto altere el lenguaje reconocido por el autómata, es decir, dado un autómata M es posible construir un autómata propio M' añadiendo y borrando instrucciones de la siguiente forma:

Paso 1: Sean q y q' dos estados para los cuales el autómata tiene la secuencia de movimientos

$$(q, \varphi, \sigma) \overset{k\, mov\, write}{\Longrightarrow} (q'', \varphi, \sigma\tau) \overset{k\, mov\, read}{\Longrightarrow} (q', \varphi, \sigma)$$

tenemos k movimientos de escritura seguidos del mismo número de movimientos de lectura, en otras palabras el contenido de la pila no se modifica del estado q al estado q' y además no se revisa ningún símbolo en la cinta de entrada.
Siempre que

$$p \,]\, mov\ (-, q)$$

con mov un movimiento permitido por el programa del autómata, sea una instrucción en M, añadimos la instrucción

$$p \,]\, mov\ (-, q')$$

En caso de que el estado q sea el estado inicial en M entonces q' será el estado inicial en M'. El procedimiento anterior puede realizarse para cada par de estados q y q' para los cuales M presenta una secuencia de movimientos impropios.

Paso 2: Borrar cada instrucción impropia. Las instrucciónes restantes son el programa de M'

Proposión 2 *Sea M un autómata de pila con movimientos impropios; es posible construir un autómata de pila M' propio tal que $L(M) = L(M')$.*

Demostración:
Debemos probar que $\omega \in L(M)$ si y sólo si $\omega \in L(M')$
Sea M un autómata de pila, supongamos que la siguiente secuencia de movimientos acepta a la cadena ω en M

$$(q_0, \lambda, \lambda) \Longrightarrow (q_k, \omega, \lambda) \tag{2.1}$$

en caso de que la secuencia contenga movimientos impropios, éstos deben ser removidos empleando el paso 1 o el paso 2 según sea necesario y obtenemos una secuencia de movimientos aceptados propios de M' para ω.

$$(q, \varphi, \sigma) \Longrightarrow (q'', \varphi, \sigma\tau) \Longrightarrow (q', \varphi, \sigma) \tag{2.2}$$

la subsecuencia más larga de (2.1) que se compone de k movimientos de escritura seguidos de k movimientos de lectura $k \geq 1$, que contiene el primer movimiento impropio, entonces al menos uno de los siguientes argumentos deberá ser cierto

1. q es el primer movimiento de la secuencia (2.1) (esto es $q = q_0$), o el movimiento que precede a la subsecuencia (2.2) no es un movimiento de escritura.

2. q' es el último estado de la secuencia (2.1)(esto es $q' = q_k$), o el movimiento que sigue a la subsecuencia (2.2) no es un movimiento de lectura.

Si q es el primer estado de la secuencia (2.1), entonces q es el estado inicial de M y, por construcción, q' es el estado inicial de M'. Por lo tanto, la subsecuencia

$$(q', \lambda, \lambda) \Longrightarrow (q_k, \omega, \lambda)$$

es una secuencia de movimientos para ω que no contiene movimientos impropios.
Si q no es el primer estado de la secuencia (2.1), la instrucción

$$p \,]\, mov\ (-, q)$$

ejecutada justo despúes de la subsecuencia (2.2) no puede ser impropia, y M' tiene por construcción la instrucción

$$p \,]\, mov\ (-, q')$$

por lo tanto, ω es aceptada por la nueva secuencia

$$(q_0, \lambda, \lambda) \Longrightarrow (p, -, -) \longrightarrow (q', \varphi, \sigma) \Longrightarrow (q_k, \omega, \lambda)$$

que no contiene movimientos impropios.
Repitiendo este proceso para cada movimiento impropio restante, se produce una secuencia de movimientos en M' para ω; así $\omega \in L(M')$ si $\omega \in L(M)$
Ahora, en M' no hay movimientos no propios, y dada una cadena ω en $L(M')$ también pertenecerá a $L(M)$ pues los movimientos impropios no alteran el contenido de la pila, ni el de la cinta de entrada por lo cual para toda cadena $\omega \in M'$ tenemos que $\omega \in M$, por lo tanto $\omega \in L(M)$ sólo si $\omega \in L(M')$

□

2.1.2. Autómatas de pila sin ciclos

El último tipo de movimiento impropio se analizará en esta sección.

Definición 25 *Un autómata de pila propio M se dice* sin ciclos *si el programa de éste no contiene ciclos de instrucciones de escritura:*

$$q_0]write(u_0, q_1)$$

$$q_1]write(u_1, q_2)$$

$$\vdots$$

$$q_n]write(u_n, q_n + 1)\ \cdots$$

Suponga que se tiene un autómata con un ciclo de instrucciones de escritura, estas imprimen los símbolos $u_0, u_1, \cdots, u_n \cdots$ de manera infinita, así los estados: $q_0, q_1, \cdots, q_n$ no pueden aparecer como una secuencia de movimientos aceptados.

Proposión 3 *Para algún autómata de pila determinístico M es posible construir un autómata de pila M' sin ciclos tal que $L(M) = L(M')$.*

2.2. Análisis sintáctico

Cuando trabajamos con un lenguaje de alto nivel en una computadora es necesario usar traductores que permitan a la máquina interpretar las instrucciones que el usuario programa; en los traductores de lenguajes se usan varios estados de procesamiento. Cuando las frases o instrucciones válidas del lenguaje son especificadas por una gramática de estructura de frases, el primer estado del proceso de traducción construye un árbol de derivación para la frase dada; una vez que está es clara tendrá asignado un único árbol de derivación para cada tipo sintáctico, de esta manera es posible asociar un significado a cada frase de acuerdo con la gramática de la misma. Al análisis anterior se le conoce con el nombre de *análisis de sintaxis.*

2.2.1. Árboles de derivación

Cuando tenemos una gramática independiente del contexto es muy útil presentar sus producciones mediante árboles de derivación; sus vértices están etiquetados con símbolos terminales o variables de la gramática.

Sea $G = (N, T, P, S)$ una gramática independiente del contexto, un árbol se llama de *derivación (o de análisis gramátical)* para G si:

1. Cada vértice tiene una etiqueta que es un símbolo de $N \cup T \cup \epsilon$.
2. La etiqueta de la raíz es S.
3. Si un vértice es interior y tiene etiqueta A, entonces A debe de estar en N.
4. Si el vértice n tiene etiqueta A y los vértices $n_1, n_2, n_3, \ldots, n_k$ son los hijos del vértice, de izquierda a derecha con etiquetas $x_1, x_2, x_3, \ldots, x_k$ respectivamente, entonces $A \longrightarrow x_1|x_2|x_3| \ldots |x_k$ debe ser una producción de P.
5. Si el vértice n tiene etiqueta ϵ, entonces es una hoja y es el único hijo de su padre.

Ejemplo 6 $G = (\{S, A\}, \{a, b\}, P, S)$ *en donde:*

$$S \longrightarrow aAS \mid a$$

$$A \longrightarrow SbA \mid SS \mid ba$$

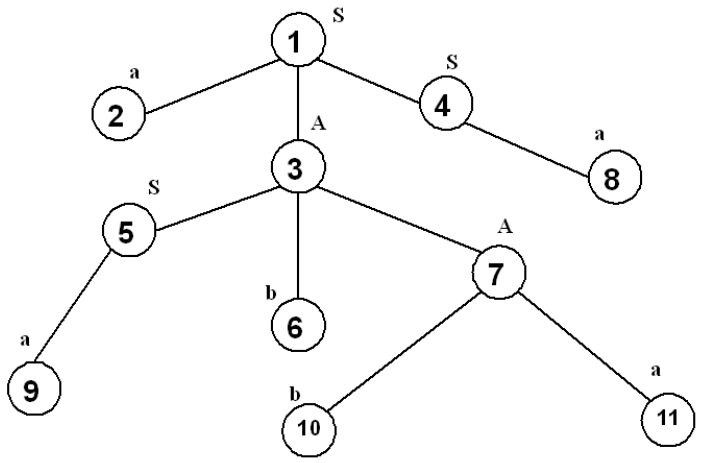

Figura 2.2:

Podemos extender el ordenamiento desde la izquierda de los hijos para producir un ordenamiento de izquierda a derecha de todas las hojas.

Un árbol de derivación es una descripción natural de una oración particular de la gramática G; si leemos las etiquetas de las hojas de izquierda a derecha se tendrá dicha oración y la cadena resultante será el producto del árbol de derivación.

Un subárbol de un árbol de derivación es un vértice particular con todos sus descendientes, las aristas que los conectan y sus etiquetas; la diferencia es que la raíz puede no ser el símbolo inicial de la gramática. Si en cada paso de una derivación se aplica una producción a la variable que se encuentra más a la izquierda la derivación se llama extrema izquierda, esto se aplica de manera análoga para la derecha.

Si $\omega \in L(G)$ tiene al menos un árbol de análisis gramátical particular, ω tiene una derivación izquierda y derecha únicas. Como puede existir más de un árbol de derivación para ω puede haber varias derivaciones izquierda y derecha.
Una gramática independiente del contexto G de la que alguna palabra tenga dos árboles de análisis gramátical se dice que es ambigua. No hay que perder de vista que es posible que una gramática produzca derivaciones que nos lleven a una cadena compuesta únicamente de símbolos terminales, pero que no necesariamente representan una oración con significado alguno, o no representan la instrucción que el programa necesita, como se observa en el siguiente ejemplo.

Ejemplo 7 *Consideremos la siguiente gramática:*

$$G_E : S \to A$$

$$S \longrightarrow if \;\; B \;\; then \;\; A \;\; else \;\; S$$
$$B \longrightarrow A = A$$
$$A \longrightarrow T$$
$$A \longrightarrow T + A$$
$$T \longrightarrow x|y|z$$

Ahora empleando la gramática anterior construyamos un árbol de derivación para la instrucción:

$$if \;\; x \;\; = \;\; y \;\; then \;\; x \;\; else \;\; x + y$$

trazaremos los pasos de la producción de la frase anterior

$$S \longrightarrow if\, B \;\; then \;\; A \;\; else \;\; S$$
$$\longrightarrow if\, A = A \;\; then \;\; A \;\; else \;\; S$$
$$\longrightarrow if\, T = A \;\; then \;\; A \;\; else \;\; S$$
$$\longrightarrow if\, x = A \;\; then \;\; A \;\; else \;\; S$$
$$\longrightarrow if\, x = T \;\; then \;\; A \;\; else \;\; S$$
$$\longrightarrow if\, x = y \;\; then \;\; A \;\; else \;\; S$$

$$\longrightarrow if\ x = y\ then\ T\ else\ S$$

$$\longrightarrow if\ x = y\ then\ z\ else\ S$$

$$\longrightarrow if\ x = y\ then\ z\ else\ A$$

$$\longrightarrow if\ x = y\ then\ z\ else\ T + A$$

$$\longrightarrow if\ x = y\ then\ z\ else\ x + A$$

$$\longrightarrow if\ x = y\ then\ z\ else\ x + T$$

$$\longrightarrow if\ x = y\ then\ z\ else\ x + y$$

Cada cadena en los pasos de derivación comienza con el símbolo S; pero observamos que para la producción

$$E \longrightarrow A$$

$$E \longrightarrow if\ \ x\ =\ y\ \ then\ \ x\ \ else\ \ x + y$$

tenemos dos candidatos, y será necesario elegir qué producción usar alternando con las diferentes posibilidades. Hasta agotar todas y llegar a la secuencia de movimientos que acepten la cadena correcta (esto es importante pues es posible obtener frases como la siguiente: if $x = y$ then $x = y$ else $x = y$, que es diferente a la frase que se va a revisar). A dicho método se le conoce con el nombre de *arriba a abajo*[1] porque la derivación comienza a construirse desde el nodo raíz al tope del árbol bajando a través de los niveles. Al proceso inverso se le conoce como *de abajo a arriba*[2].

2.3. Construcción de un AP analizador de sintaxis

Para diseñar un autómata de pila que realice las operaciones arriba a abajo del analizador de sintaxis se necesita $G = (N, T, P, \Sigma)$ una gramática independiente del contexto. A partir de una cadena aceptada por $L(G)$ será posible construir un autómata de pila.

[1] en ingles top-down
[2] en ingles buttom-up

Primero supongamos que :

$$\Sigma \Rightarrow \varphi_1 A_1 \beta_1 \ldots \Rightarrow \varphi_k A_k \beta_k \Rightarrow \omega$$

es la derivación más a la izquierda de ω donde:

$\varphi_i \in T^*$

$A_i \in N$ y $\beta_i \in (N \cup T)^* \ \ \forall i = 1 \ldots k$

Las sentencias de tal derivación son representadas por configuraciones de M específicamente la sentencia:

$$\varphi_i A_i \beta_i$$

está representada por la configuración

$$(q_R, \varphi_i, \beta_i^R A_i)$$

en la cual los símbolos de la cadena de entrada revisados hasta ahora es φ_i, la cinta contiene la inversa de $A_i \beta_i$ y q_R es un estado especial de M.
El comportamiento cuando M presenta una cinta de entrada que contiene alguna cadena ω será asumir, en sucesión, las configuraciones correspondientes de cualquier derivación izquierda de ω

$$(q_I, \lambda, \lambda) \Rightarrow (q_R, \lambda, \Sigma) \Rightarrow (q_R, \varphi_1, \beta_1^R A_1) \Rightarrow (q_R, \varphi_k, \beta_k^R A_k) \Rightarrow (q_R, \omega, \lambda)$$

Es posible construir un autómata de pila con un programa de instrucciones que aplique cada secuencia de movimientos

$$(q_R, \varphi_i, \beta_k^R A_i) \Rightarrow (q_R, \varphi_{i+1}, \beta_{i+1}^R A_{i+1})$$

en correspondencia a un paso de derivación de la gramática G. Siempre que sean producciones alternativas aplicables a la oración, el autómata de pila podría tener un conjunto de movimientos alternativos correspondiente. Así M en general será un autómata determinístico.
El programa de M tendrá dos operaciones básicas *expansión y correspondencia*. La expansión corresponde a aplicar la producción $A_i \rightarrow \psi$ de sentencias de la forma

$$\varphi_i A_i \beta_i$$

La correspondencia se empleará si el símbolo terminal en que comienza ψ puede ser asociado con el siguiente símbolo en la cinta de entrada.
El símbolo tope de la pila determina cual de estas dos operaciones ocurre, un símbolo no terminal indica expansión y un símbolo terminal indica correspondencia. El estado q_R es el estado al leer el símbolo tope de la pila.

1. Expansión: Reemplaza la letra no terminal A en el toppe de la pila por ψ^R, la inversa del lado derecho de alguna producción $A \to \psi$ en G, regresa al estado q_k, la expansión de acuerdo con la regla $A \longrightarrow \psi$ se implementa en el programa del autómata por las siguientes instrucciones:

$$q_R \ ] \ read \ (A, q_A)$$

$$q_A \ ] \ write \ (\psi^R, q_R)$$

Aquí se usa una instrucción de escritura generalizada abreviado la secuencia de movientos escribir que añaden a la cadena ψ^R en la pila.

2. Correspondencia: Si el símbolo del tope de la pila es una letra terminal t revisamos la siguiente letra s en la cinta de entrada; si $s = t$ la correspondencia es exitosa y podemos continuar; de otra forma la correspondencia falla y la operación se suspende, pues el contenido de la pila y el de la cinta de entrada no es el mismo. La correspondencia se implementa por las instrucciones

$$q_R \ ] \ read(t, q_t)$$

$$q_t \ ] \ scan(t, q_R)$$

para cada $t \in T$

3. El reconocimiento para el lenguaje de la gramática se completa añadiendo la instrucción inicial

$$q_I] write(\Sigma, q_R)$$

Con la cual iniciamos la pila con el símbolo Σ. Los estados iniciales y finales de M son:

$$I = \{q_I\}$$

$$F = \{q_R\}$$

Podemos incluir q_I en F si $\Sigma \longrightarrow \lambda$ es una producción de G.

Con base en las anteriores reglas se construye una tabla que las resume. Dada una gramática independiente del contexto $G = (N, T, P, \Sigma)$ para construir un autómata de pila $M = (Q, T, U, P, I, F)$ tal que $L(G) = L(M)$ sea

$U = N \cup T \cup |\Sigma|$

$Q = \{q_I, q_R\} \cup \{q_x \ni x \in U\}$

$I = \{q_I\}\, q_R \in F$

Las instrucciones de M son:

Para iniciar	$q_I]write\,(\Sigma, q_R)$
Para expandir Para cada producción $A \longrightarrow \psi \in G$,donde $A \in N \cup \{\Sigma\}$ y $\psi \in (N \cup T)^* - \boldsymbol{\lambda}$	 $q_R]read\,(A, q_A)$ $q_R]read\,(A, q_A)$
Para hacer corresponder Para cada símbolo $t \in T$	 $q_R]read\,(t, q_t)$ $q_t]scan\,(t, q_R)$
Para aceptar λ Si G tiene $\Sigma \longrightarrow \lambda$	 $q_I \in F$

Cuadro 2.1: Construcción de un analizador de AP

Note que las reglas anteriores no producen un autómata de pila propio, porque se ejecuta un movimiento de lectura inmediatamente despúes de un movimiento de escritura, en la expansión de un símbolo no terminal, por lo cual seá necesario eliminar movimientos improductivos para poder tener el autómata final.
A continuación se justifica la construcción de la tabla y la aplicación de sus reglas mediante el siguiente:

Teorema 1 *Para cualquier gramática independiente de contexto G es posible construir un autómata de pila tal que $L(M) = L(G)$. Además la secuencia de movimientos por M tiene una correspondencia uno a uno con la derivación izquierda de G.*
Demostración:
Sea G una gramática independiente del contexto y sea M un autómata de pila formado de G de acuerdo con la tabla anterior dado $\omega_0 = \Sigma$, y sea ω_i con $i = 1, 2, \ldots, k$ una secuencia de cadenas

$$\omega_i = \varphi_i A_i \beta_i$$

en la cual $\varphi_i \in T^*$, $A_i \in N$, $\beta_i \in (N \cup T)^*$
Así, en cada cadena ω_i A_i será el símbolo no terminal que está más a la izquierda. Se verá que $\Sigma \Longrightarrow \omega_1 \Longrightarrow \omega_2 \Longrightarrow \ldots \omega_k \Longrightarrow \omega$ es la derivación izquierda de ω en G si y sólo si

$$\begin{aligned}(q_R, \lambda, \Sigma) &\Longrightarrow \left(q_R, \varphi, \beta_1^R A_1\right) \\ &\Longrightarrow \left(q_R, \varphi, \beta_2^R A_2\right) \\ &\vdots \\ &\Longrightarrow \left(q_R, \varphi, \beta_K^R A_K\right) \\ &\Longrightarrow (q_R, \omega, \lambda)\end{aligned}$$

es una secuencia de movimientos de M donde

$$(q_I, \lambda, \lambda) \overset{\omega}{\Longrightarrow} (q_R, \lambda, \sigma)$$

Éste es el único movimiento inicial posible de A, y donde q_R es el estado final. Este resultado verificará que $\omega \in L(G)$ si y sólo si $\omega \in L(M)$. Además, la relación de la oración $\varphi A \beta$ con la configuración $(q_R, \varphi, \beta^R A)$ prueba que es necesaria una correspondencia uno a uno entre las derivaciones en G y las secuencias de movimientos en M.
Esta afimación se sigue por indución de la siguiente proposición:

1. La gramática permite el paso de derivación izquierda

   $$\varphi A \beta \Longrightarrow \varphi' A' \beta'$$

   si y sólo si

   $$(q_R, \varphi, \beta^R A) \Longrightarrow (q_R, \varphi', \beta'^R A')$$

   es una secuencia de M que contiene exactamente un movimiento generalizado de escritura.

2. La gramática G permite el paso de derivación izquierda

   $$\varphi A \beta \Longrightarrow \omega$$

   si y sólo si

   $$(q_R, \varphi, \beta^R A) \Longrightarrow (q_R, \omega, \lambda)$$

   es una secuencia de movimientos con exactamente un movimiento generalizado de escritura.

A continuación se proporciona la justificacion de las dos afirmaciones anteriores.

Primera:

Si $\varphi A\beta \Longrightarrow \varphi' A'\beta'$ entonces G tiene una producción $A \longrightarrow \psi$ tal que

$$\varphi' A'\beta' = \varphi\psi\beta$$

en este caso la producción cambia un símbolo terminal con lo cual se tendrá una expansión, con lo cual M tendrá instrucciones de lectura y de escritura tales que:

$$(q_R, \varphi, \beta^R A) \overset{q_R]read(A,q_A)}{\longrightarrow} (q_R, \varphi, \beta^R) \overset{q_A]write(\psi^R,q_R)}{\longrightarrow} (q_R, \varphi, \beta^R\psi^R)$$

son movimientos de M así:

$$(q_R, \varphi, \beta^R A) \Longrightarrow (q_R, \varphi, \beta^R\psi^R)$$

La cadena $\psi\beta$ contiene, por hipótesis, el símbolo no terminal izquierdo A' que es $\psi\beta = \varphi'' A'\beta'$, donde:

$$\varphi'' \in T^* \;\; y \;\; \beta' \in (N \cup T)^*$$

Suponga que $\psi'' = s_1 s_2 s_3, \ldots, s_m$ son todos símbolos terminales; la construcción de M tiene movimientos de lectura y revisión que permiten la siguiente secuencia:

$(q_R, \varphi, \beta'^R A' s_m s_{m-1}, \ldots, s_2 s_1) \overset{q_R]read(s_1,q_{s1})}{\longrightarrow}$

$(q_{s1}, \varphi, \beta'^R A' s_m s_{m-1}, \ldots, s_2) \overset{q_1]scan(s_1,q_R)}{\longrightarrow}$

$(q_R, \varphi s_1, \beta'^R A' s_m s_{m-1}, \ldots, s_2) \longrightarrow$

$\vdots$

$\overset{q_R]read(s_m,q_{sm})}{\longrightarrow}$

$(q_{sm}, \varphi s_1 s_2 \cdots s_{m-1}, \beta'^R A' s_m) \overset{q_{sm}]scan(s_m,q_R)}{\longrightarrow} (q_R, \varphi s_1 s_2 \cdots s_m, \beta'^R A')$

$(q_R, \varphi, \beta'^R A' \varphi''^R) = (q_R, \varphi', \beta'^R a')$

donde $\varphi' = \varphi\varphi'' = \varphi s_1 s_2, \cdots, s_m$

Suponemos que $(q_R, \varphi, {\beta'}^R A') \Longrightarrow (q_R, \varphi', {\beta'}^R A')$ es una secuencia de movimientos que contiene sólo un movimiento generalizado de escritura. Como el símbolo tope de la pila es no terminal, la primer secuencia de movimientos deberá ser un movimiento de lectura seguido por un movimiento generalizado de escritura

$$(q_R, \varphi, \beta^R A) \overset{q_R]read(A,q_A)}{\longrightarrow} (q_A, \varphi, \beta^R) \overset{q_A]write(\psi^R,q_R)}{\longrightarrow} (q_R, \varphi, \beta^R \psi^R)$$

donde $A \longrightarrow \psi$ es una producción de G. Los movimientos restantes pueden ser sólo movimientos de lectura y revisión que emparejen los símbolos terminales sucesivos en $\psi\beta$, con símbolos en la cinta de entrada:

$$(q_r, \varphi, \beta^R \psi^R) \Longrightarrow (q_R, \varphi', {\beta'}^R A')$$

Por lo tanto, $\varphi\psi\beta = \varphi' A' \beta'$ y como $A \longrightarrow \psi$ está en G tenemos:

$$\varphi\psi\beta \Longrightarrow \varphi' A' \beta'$$

es permitida por G.

Para la demostración de que $L(M) = L(G)$, es necesario mostrar que $\lambda \in L(M)$ si y sólo si $\lambda \in L(G)$. Del último renglón de la tabla de construcción se observa que $\lambda \in L(M)$ implica que $\lambda \in L(M)$, para q_I es el estado final de M sólo si $\Sigma \longrightarrow \lambda$ es una producción en G.

Segunda afirmación:
Necesidad:
Supongamos que $\varphi A\beta \Longrightarrow \omega$ es decir, existe una derivación

$$\varphi A\beta \longrightarrow \psi_1 \longrightarrow \psi_2 \cdots \longrightarrow \psi_n = \omega$$

sea

$$A \longrightarrow \varphi_i A_i \beta_i \;\; con : \begin{cases} A_i \in N \\ \varphi_i \in T^* \\ \beta_i \in (N \cup T)^* - \lambda \end{cases}$$

Una producción en G entonces el autómata permite la siguiente secuencia de pasos de derivación

$$(q_0, \lambda, \lambda) \Longrightarrow (q_I, \lambda, \Sigma) \Longrightarrow (q_R, \varphi, \beta^R A)$$

con q_0 y q_I estados iniciales.
Como el tope de la pila es un símbolo no terminal entonces se sustituye por una producción de G

$$(q_R, \varphi, \beta^R A) \overset{q_R]read(A,q_A)}{\longrightarrow} \longrightarrow (q_A, \varphi, \beta^R) \overset{q_A]write(\beta_1^R A_1 \varphi_1^R, q_R)}{\Longrightarrow}$$

$$\longrightarrow (q_R, \varphi, \beta^R \beta_1^R A_1 \varphi_1^R)$$

Ahora como φ_1^R en el tope de la pila es una cadena de símbolos terminales será necesario aplicar varios movimientos de revisión para sacarlos de la pila y escribirlos en la cinta de entrada.

$$\longrightarrow (q_R, \varphi, \beta^R \beta_1^R A_1 \varphi_1^R) \overset{scan}{\Longrightarrow} (q_R, \varphi\varphi_1, \beta^R \beta_1^R A_1)$$

nuevamente A_1 es un no terminal, por lo cual es necesario volver a expandir

$$(q_R, \varphi\varphi_1, \beta^R \beta_1^R A_1) \overset{q_R]read(A_1,q_{A1})}{\longrightarrow}$$

$$\longrightarrow (q_R, \varphi\varphi_1, \beta^R \beta_1^R) \overset{q_{A1}]write(\beta_2^R A_2 \varphi_2^R, q_R)}{\longrightarrow}$$

$$\longrightarrow (q_R, \varphi\varphi_1, \beta^R \beta_1^R \beta_2^R A_2 \varphi_2^R)$$

$$\overset{scan}{\Longrightarrow} (q_R, \varphi\varphi_1\varphi_2, \beta^R \beta_1^R \beta_2^R A_2) \overset{q_R]read(A_2,q_{A2})}{\longrightarrow}$$

$$\longrightarrow (q_R, \varphi\varphi_1\varphi_2, \beta^R \beta_1^R \beta_2^R) \overset{q_{A2}]write(\beta_3^R A_3 \varphi_3^R, q_R)}{\longrightarrow}$$

$$\longrightarrow (q_R, \varphi\varphi_1\varphi_2, \beta^R \beta_1^R \beta_2^R \beta_3^R A_3 \varphi_3^R)$$

$$\vdots$$

$$\longrightarrow (q_R, \varphi\varphi_1 \cdots \varphi_k, \lambda)$$

$$= (q_R, \psi_n, \lambda) = (q_R, \omega, \lambda)$$

Suficiencia:
Supongamos que $(q_R, \varphi, \beta^R A) \Longrightarrow (q_R, \omega, \lambda)$, observamos que en la parte izquierda de la derivación el símbolo en el tope de la pila es no terminal entonces dada una producción $A \longrightarrow \psi$ en la gramática que nos permite realizar una expansión.

$$(q_R, \varphi, \beta^R A) \overset{q_R]read(A,q_A)}{\longrightarrow}$$

$\longrightarrow (q_A, \varphi, \beta^R) \overset{q_A]write(\psi^R, q_R)}{\Longrightarrow}$

$\longrightarrow (q_R, \varphi, \beta^R\psi^R)$

Los movimientos restantes deberán hacer corresponer los símbolos terminales, e introducirlos en la cinta de entrada para lo cual se utilizarán movimientos read y scan.

$$(q_R, \varphi, \beta^R A) \Longrightarrow (q_R, \varphi\psi\beta, \lambda)$$

ahora sustituyendo con la parte izquierda de la producción:

$$(q_R, \varphi\psi\beta) \Longrightarrow (q_R, \varphi A\beta, \lambda)$$

por lo tanto $\varphi A\beta \Longrightarrow \omega$

□

2.4. Conjuntos travesía

La secuencia de movimientos desde que expandimos el símbolo no terminal A, hasta que se leen en la pila y se escriben en la cinta de entrada todos los símbolos terminales

$$(q, \varphi, \sigma) \Longrightarrow (q', \varphi\omega, \sigma)$$

con σ el contenido inicial de la pila y el estado q con una etiqueta de escritura que añadirá a la pila la expansión de A; es una secuencia de movimientos donde el tope de la pila regresa a su posición inicial se llama *travesía*; y la cadena ω sobre la cual el tope de la pila avanza se le conoce como *cadena observada por la travesía*

La secuencia correspondiente a la derivación única será

$$\varphi A\sigma^R \overset{*}{\Longrightarrow} \varphi\omega\sigma^R$$

en la gramática G esto es $A \overset{*}{\Longrightarrow} \omega$, es decir, la cadena observada por una travesía se deriva de un símbolo no terminal en la gramática correspondiente.

Definición 26 *Sea $M = (Q, S, U, P, I, F)$ un autómata de pila propio, una secuencia de movimientos de M*

$$(q, \varphi, \sigma) \overset{T}{\Longrightarrow} (q', \varphi\omega, \sigma)$$

se llama travesía *de ω del estado q al estado q' si*

1. $\sigma = \sigma'$
2. $\varphi\omega = \varphi'$
3. *Cada configuración* (q_i, φ, σ_i) *ocurre dentro de una secuencia de movimientos que satisfacen* $|\sigma_i| \geq |\sigma|$

Para representar una travesía escribimos:

$$(q, \varphi, \sigma) \overset{T}{\Longrightarrow} (q', \varphi\omega, \sigma)$$

el conjunto travesía $T(q, q')$ es el conjunto de todas las cadenas observadas por las travesías de q a q'.

$$T(q, q') = \left\{ \omega \in S^* | (q, \varphi, \sigma) \overset{T}{\Longrightarrow} (q', \varphi\omega, \sigma) \textit{ para algún } \varphi \in S^* \textit{ y algún } \sigma \in U^* \right\}$$

Una travesía sin movimientos

$$(q, \varphi, \sigma) \overset{T}{\Longrightarrow} (q, \varphi, \sigma)$$

que observa a la cadena vacía es llamada *travesía trivial* así $\lambda \in T(q, q')$ para cada $q \in Q$

2.5. Propiedades de los conjuntos travesía

Proposión 4 *Si* $(q, \varphi, \sigma) \overset{T}{\Longrightarrow} (q', \varphi\omega, \sigma)$ *es una travesía de* ω *para algún* $\varphi \in S^*$ *y algún* $\sigma \in U^*$*, entonces es una travesía de* ω *para cualquier* $\varphi \in S^*$ *y cualquier* $\sigma \in U^*$

En el caso de que esto no fuera cierto entonces los movimientos que leen la cadena ω mueven el tope de pila más a la izquierda de la posición inicial, o no leen todos los símbolos no terminales de la pila.

Si para alguna travesía de ω las cadenas φ, σ son vacías tenemos:

$$(q, \lambda, \lambda) \overset{T}{\Longrightarrow} (q', \omega, \lambda)$$

La cual es una secuencia aceptada de movimientos para ω sólo si $q \in I$ y $q' \in F$. Además una secuencia aceptada de movimientos para ω puede ser una travesía de la misma porque el tope de la pila no se mueve a la izquierda, lo cual solamente es posible cuando la pila esta vacía.

Proposión 5 *Un autómata de pila M reconoce la cadena ω si y sólo si esta se observa en alguna travesía de algún estado inicial a algún estado final.* $\omega \in L(M) \Leftrightarrow \omega \in T(q, q'), q \in I, q' \in F$

Demostración:
Necesidad:
Supongamos que $\omega \in L(M)$, es decir existe una secuencia de movimientos

$$(q_I, \lambda, \lambda) \Longrightarrow (q_f, \omega, \lambda)$$

Para q_I estado inicial y $q_f \in F$, que reconoce a la cadena ω, la cual pertenece a la travesía $T(q_I, q_f)$

Suficiencia:
Si $\omega \in T(q, q')$ con $q \in I$ y $q' \in F$, existe una travesía

$$(q, \varphi, \sigma) \Longrightarrow (q', \varphi\omega, \sigma)$$

y por la proposición anterior también es valido para $\varphi = \lambda$ y $\sigma = \lambda$
por lo tanto $(q, \lambda, \lambda) \Longrightarrow (q', \omega, \lambda)$ y $\omega \in L(M)$

□

Corolario 1

$$L(M) = \bigcup_{q \in I, q' \in F} T(q, q')$$

Proposión 6 *Sea M un autómata de pila propio sólo las travesías de λ por M son travesías triviales*

$$(q, \varphi, \sigma) \overset{T}{\Longrightarrow} (q, \varphi, \sigma)$$

donde

$$q \in Q, \varphi \in S^*, \sigma \in U^*$$

esto es $\lambda \in T(q, q') \Leftrightarrow q = q'$
Demostración:

La travesía que observa la cadena vacía λ no contiene movimientos de revisión, por lo que el contenido de la cinta de entrada no cambia; además, por hipótesis, el autómata es propio, por lo cual no se permite escribir y leer símbolos sin revisarlos, por lo tanto la única cadena permitida es λ,

□

2.6. Construcción de una gramática

Para construir una gramática a partir de un autómata de pila se relaciona el conjunto de los símbolos no terminales $N(q, q')$ con los conjuntos travesía $T(q, q')$, donde cada símbolo no terminal denota un elemento del correspondiente conjunto travesía.

Definición 27 *En una travesía* $(q, \varphi, \sigma) \overset{T}{\Longrightarrow} (q', \varphi\omega, \sigma)$*, de un autómata de pila, un movimiento de escritura en la travesía se llama* movimiento básico de escritura *si éste se ejecuta a partir de una configuración en la cual el contenido de la pila es* σ*; y un movimiento de lectura se llama* movimiento básico de lectura *si permite a la pila llegar a* σ*.*

Los movimientos básicos forman lo que se conoce como *pareja correspondiente* si el movimiento de escritura precede al movimiento de lectura y no intervienen otros movimientos base.
Los movimientos básicos de lectura siempre deberán ser correspondidos con movimentos básicos de escritura, y deben leer todos los símbolos escritos en la pila desde el inicio de la travesía hasta el final de la misma; de la misma forma, si un símbolo se escribe con un movimento básico de escritura, el movimiento que revise el símbolo estará seguido de un movimiento básico de lectura.

Proposión 7 *Una travesía que contiene un movimiento básico de lectura (escritura) también contiene su pareja correspondiente, un movimiento básico de escritura (lectura).*
Para demostrar la proposición es necesario recordar que una travesía regresa el tope de la pila a su posición inicial, lo cual sólo se logra si todos los símbolos escritos en la pila son leídos, es decir, existe el mismo número de movimientos de lectura que de escritura.
Es posible relacionar cadenas de un conjunto travesía, con cadenas pertenecientes a otros conjuntos travesía. En la siguiente proposición se expresan formalmente las cuatro posibles formas de una travesía, todas mutuamente excluyentes.

Proposión 8 *Sea* M *un autómata de pila y supongamos que*

$$(q, \varphi, \sigma) \overset{T}{\Longrightarrow} (q', \varphi\omega, \sigma)$$

es una travesía de ω en M, entonces exactamente una de las siguientes afirmaciones es cierta.

1. *La travesía sólo tiene un movimiento scan*

$$(q, \varphi, \sigma) \xrightarrow{scan} (q', \varphi\omega, \sigma)$$

2. *La travesía tiene más de un movimiento, y el último movimiento es scan.*

$$(q, \varphi, \sigma) \overset{T}{\Longrightarrow} (q'', \varphi\psi, \sigma) \xrightarrow{scan} (q', \varphi\psi s, \sigma)$$

   con $\omega = \psi s$ y $\psi \in T(q, q'')$

3. *La travesía tiene más de un movimiento, el último movimiento es de lectura y su pareja correpondiente (movimiento de escritura) no es el primer movimiento.*

$$(q, \varphi, \sigma) \overset{T}{\Longrightarrow} (q'', \varphi\psi, \sigma) \xrightarrow{write} (p, \varphi\psi, \sigma u) \overset{T}{\Longrightarrow} (p'.\varphi\psi\theta, \sigma u) \xrightarrow{read} (q'\varphi\psi\theta, \sigma)$$

   donde: $u = \psi\theta$, $\psi \in T(q, q'')$ y $\theta \in T(p, p')$

4. *La travesía tiene más de un movimiento, el último movimiento es de lectura y su pareja correspondiente (movimiento de escritura) es el primer movimiento.*

$$(q, \varphi, \sigma) \xrightarrow{write} (p, \varphi, \sigma u) \overset{T}{\Longrightarrow} (p', \varphi\theta, \sigma u) \xrightarrow{read} (q', \varphi\theta, \sigma)$$

   donde: $\omega = \theta$ y $\theta \in T(p, p')$

Demostración:

Observamos que cada caso es mutuamente excluyente con los demás. Para la primera afirmación sólo existe la posibilidad de relizar un movimiento y el movimiento de revisión es el único que no mueve el tope de la pila.
Para el caso en el que tenemos más de un movimiento, el último deberá ser de revisión o de lectura, pues es necesario regresar a su posición original el tope de la pila o mantenerlo en su lugar, si el último movimiento es de lectura, será un movimiento base y su pareja correspondiente puede ser el primer movimiento o no; para la segunda y tercera afirmación el primer movimiento será una travesía ,y en el caso de la tercera y cuarta afirmación

los movimientos entre la pareja correspondiente serán travesías que no contendrán movimientos base.

Para poder construir una gramática a partir de un autómata de pila, no se debe perder de vista que es necesario tener un conjunto de producciones que nos lleve de un estado inicial hacia algún estado final, estas producciones son de tal forma que para algúna cadena ω en S^* $\omega \in T(q, q')$ en M si y sólo si $N(q, q') \overset{*}{\Longrightarrow} \omega$, en G donde $T(q, q')$ es un conjunto travesía y $N(q, q')$ es el correspondiente símbolo no terminal.

□

En las afirmaciones de la proposición anterior, observamos que cada relación específica entre los conjuntos travesía corresponde a relaciones entre los conjuntos de cadenas derivables de los símbolos no terminales de la gramática.

1. $$(q, \varphi, \sigma) \overset{scan}{\longrightarrow} (q', \varphi s, \sigma)$$
   corresponde a $N(q, q') \Longrightarrow s$
2. $$\underbrace{(q, \varphi, \sigma) \overset{T}{\Longrightarrow} (q'', \varphi\psi, \sigma)}_{N(q,q'') \overset{*}{\Longrightarrow} \psi} \overset{scan}{\longrightarrow} (q', \varphi\psi s, \sigma)$$
   corresponde a $N(q, q') \Longrightarrow N(q, q'')s$
3. $$\underbrace{(q, \varphi, \sigma) \overset{T}{\Longrightarrow} (q'', \varphi\psi, \sigma)}_{N(q,q'') \overset{*}{\Longrightarrow} \psi} \overset{write}{\longrightarrow} \underbrace{(p, \varphi\psi, \sigma u) \overset{T}{\Longrightarrow} (p', \varphi\psi\theta, \sigma u)}_{N(p,p') \overset{*}{\Longrightarrow} \theta} \overset{read}{\longrightarrow} (q', \varphi\psi\theta, \sigma)$$
   corresponde a $N(q, q') \Longrightarrow N(q, q'')N(p, p')$
4. $$(q, \varphi, \sigma) \overset{write}{\longrightarrow} \underbrace{(p, \varphi, \sigma u) \overset{T}{\Longrightarrow} (p, \varphi\theta, \sigma u)}_{N(p,p') \overset{*}{\Longrightarrow} \theta} \overset{read}{\longrightarrow} (q'\varphi\theta, \sigma))$$
   corresponde a $N(q, q') \Longrightarrow N(p, p')$

□

2.7. Relación entre travesías y pasos de derivación

Como asociamos los símbolos no terminales con travesías es necesario conocer cuales son los estados (q, q') en los que inician y terminan éstas. La travesía comenzará en un estado inicial o después de un movimiento de escritura en la cerradura de la travesía, entonces:

$$B = I \cup \{q' \mid q]write(u, q') \text{ en } M \text{ para algún } q \in Q \text{ y algún } u \in U\}$$

El conjunto de los símbolos no terminales de G es

$$N = \{N(q, q') | q \in B, q' \in Q\}$$

Dependiendo del tipo de instrucciones de M se definen las reglas que relacionen las travesías con símbolos no terminales, las cuales se muestran en la tabla.

Observamos que la regla 5 relaciona las travesías de estados iniciales a esta-

Regla	**Si M tiene la(s) instrucción(es)**	**Entonces G tiene**
1	$q]scan(s, q')$ $q \in B$	$N(q, q') \longrightarrow s$
2	$q'']scan(s, q')$ $q \in B$	$N(q, q') \longrightarrow N(q, q'')s$
3	$q'']write(u, p)$ $p']read(u, q')$ $q \in B$	$N(q, q') \longrightarrow N(q, q'')N(p, p')$
4	$q]write(u, p)$ $p']read(u, q')$ $q \in B$	$N(q, q') \longrightarrow N(p, p')$
5	$q \in I, q' \in F$	$\Sigma \longrightarrow N(q, q')$
6	$I \cap F \neq \oslash$	$\Sigma \longrightarrow \lambda$

Cuadro 2.2: **Construcción de una gramática a partir de un AP**

dos finales en M como cadenas derivables de Σ en G y la regla 6 prueba que G genera a λ sólo si M acepta a λ.

Ejemplo 8 *Sea M el autómata de la figura(2.3).*

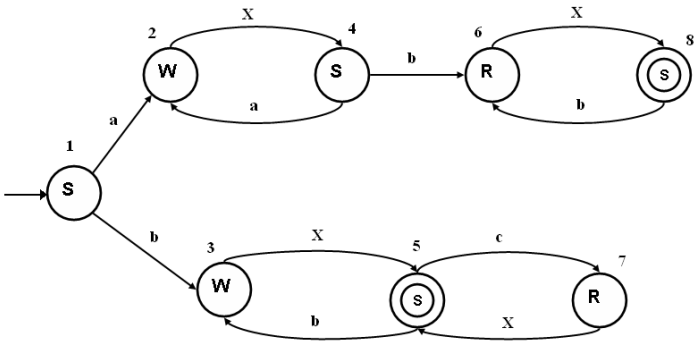

Figura 2.3:

$T(1,2):A$	$T(1,7):E$	$T(5,5):I$	$T(2,8)$	$T(8,6)$
$T(1,3):B$	$T(1,8):F$	$T(5,3):J$	$T(2,6)$	
$T(1,5):C$	$T(4,6):G$	$T(5,7):K$	$T(3,5)$	
$T(1,6):D$	$T(4,2):H$	$T(4,8):L$	$T(3,7)$	

El conjunto de travesías es :
con $B = \{1, 4, 5\}$
Nombramos con letras mayusculas aquellas travesías cuyo estado inicial pertenece a B y aplicando las reglas de la tabla (2.2) la gramática G será la siguiente:

Regla 1	Regla 2	Regla 3	Regla 5
$A \longrightarrow a$	$E \longrightarrow Cc$	$C \longrightarrow BK$	$\Sigma \longrightarrow F$
$B \longrightarrow b$	$G \longrightarrow Lb$	$F \longrightarrow AG$	$\Sigma \longrightarrow C$
$G \longrightarrow b$	$K \longrightarrow Ic$	$L \longrightarrow HG$	
$H \longrightarrow a$	$J \longrightarrow Ib$	$I \longrightarrow JK$	
$J \longrightarrow b$	$D \longrightarrow Fb$		
$K \longrightarrow c$			

Las reglas 4 y 6 no son aplicables. Despúes de eliminar las producciones $E \longrightarrow Cc$ y $C \longrightarrow BK$ por no ser útiles se tiene que $L(M) = L(G)$

El resultado que respalda la tabla (2.2) es el siguiente:

Teorema 2 *Para algún autómata de pila M es posible construir una gramática independiente del contexto G tal que $L(M) = L(G)$*
Demostración:
Para demostrar el teorema supondremos que M es propio; primero construiremos la derivación

$$N(q, q') \stackrel{*}{\Longrightarrow} \omega$$

para algúna travesía en el autómata, y después a partir de algúna derivación

$$N(q, q') \stackrel{*}{\Longrightarrow} \omega$$

en la gramática construiremos una travesía en el autómata, en ambos casos la demostración se hará por inducción sobre el número de movimientos en la travesía para la primera parte, y para la segunda parte sobre el número de pasos en la derivación empleando las reglas de la tabla (2.2).
Primera parte
Base: Suponemos que

$$(q, \varphi, \sigma) \stackrel{T}{\Longrightarrow} (q', \varphi\omega, \sigma)\ q \in B \tag{2.3}$$

es una travesía de ω para M. Si la travesía tiene sólo un movimiento éste deberá ser un movimiento de revisión

$$(q, \varphi, \sigma) \xrightarrow{scan} (q', \varphi s, \sigma)$$

con lo cual M tiene el movimiento

$$q]scan(s, q')$$

y de la tabla (2.2)

$$N(q, q') \longrightarrow s$$

y la derivación $N(q, q') \overset{*}{\Longrightarrow} \omega$ con $(\omega = s)$ consiste de un sólo paso.
Hipótesis de inducción:
Para $n \geq 1$ la derivación $N(p, p') \overset{*}{\Longrightarrow} \psi$ puede ser construida para algúna cadena aceptada por una travesía de p a p' en n o menos movimientos.
Paso de Inducción:
Ahora supongamos que la travesía tiene $n + 1$ movimientos, se tienen dos posibilidades dependiendo de cual sea el último movmiento de lectura o de revisión y ambas son mutuamente excluyentes.

Caso I : El último movimiento es *scan*

$$(q, \varphi, \sigma) \overset{T}{\Longrightarrow} (q'', \varphi\psi, \sigma) \xrightarrow{scan} (q', \varphi\psi s, \sigma) \tag{2.4}$$

con $q \in Q$ y$\omega = \psi s$

Así los primeros n movimientos constituyen una travesía que por inducción produce la dervivación $N(q, q'') \overset{*}{\Longrightarrow} \psi$ además el autómata tiene la instrucción

$$q'']scan(s, q')$$

y G tiene la producción $N(q, q') \longrightarrow N(q, q'')s$ así $N(q, q') \Longrightarrow N(q, q'')s$ forman la derivación $\psi s = \omega$

Caso II: El último movimiento es de lectura

$$(q, \varphi, \sigma) \Longrightarrow (p', \varphi\omega, \sigma u) \xrightarrow{read} (q', \varphi\omega, \sigma)$$

para algún $p' \in Q$ y $u \in U$ el movimiento de lectura deberá estar correspondido con un movimiento básico de escritura en la travesía

$$(q, \varphi, \sigma) \stackrel{T}{\Longrightarrow} (q'', \varphi\psi, \sigma) \stackrel{write}{\longrightarrow} (p, \varphi\psi, \sigma u) \stackrel{T}{\Longrightarrow} (p', \varphi\psi\theta, \sigma u) \stackrel{read}{\longrightarrow} (q'\varphi\psi\theta, \sigma) \tag{2.5}$$

donde $\omega = \psi\theta$ así el autómata tendrá los movimientos:

$$q'' \ ] \ write(u, p)$$

$$p' \ ] \ read(u, q')$$

la secuencia anterior tendrá dos travesía de las cuales ninguna tendra más de n movimientos

$$(q', \varphi, \sigma) \stackrel{T}{\Longrightarrow} (q'', \varphi\psi, \sigma)$$

$$(p, \varphi\psi, \sigma u) \stackrel{T}{\Longrightarrow} (p', \varphi\psi\theta, \sigma u)$$

la primera travesía deberá ser trivial así el movimiento de escritura es el primer movimiento en la secuencia (2.5); además como M es propio la segunda travesía tendrá al menos un movimiento de revisión con lo cual no será trivial. En caso de ser trivial $q'' = q$ y $\psi = \lambda$ y por la regla (4) G tendrá la producción

$$N(q, q') \longrightarrow N(p, p')$$

Aplicando la hipótesis de inducción la derivación $N(p, p') \stackrel{*}{\Longrightarrow} \theta$ puede ser construida y $N(q, q') \Longrightarrow N(p, p') \stackrel{*}{\Longrightarrow} \omega$ es una derivación en G de $\theta = \omega$ además si la primera travesía es no trivial por la regla (3) G tendrá una producción

$$N(q, q') \longrightarrow N(q, q'')N(p, p')$$

por hipótesis de inducción las derivaciones

$$N(q, q'') \stackrel{*}{\Longrightarrow} \psi \ y \ N(p, p') \stackrel{*}{\Longrightarrow} \theta$$

pueden ser construidas en G y en conjunto con

$$N(q, q') \Longrightarrow N(q, q'')N(p, p')$$

forman una derivación de $\psi\theta = \omega$.

Segunda parte
Base: Si la derivación de ω consiste de un sólo paso deberemos usar $N(q, q') \longrightarrow s$ con $q \in B$ y $q' \in Q$ donde $\omega = s$ y M deberá tener la instrucción

$$q \ ] \ scan(s, q')$$

y

$$(q, \varphi, \sigma) \xrightarrow{scan} (q'\varphi s, \sigma)$$

es una travesía de ω
Hipótesis de inducción:
Para $n \geq 1$ suponemos que si ψ se deriva de $N(p, p')$ en n o menos pasos, entonces M observa a ψ en una travesía de un estado p a un estado p'.
Paso de inducción:
Ahora supongamos que ω se deriva de $N(q, q')$ en $n + 1$ pasos. La primera producción no podrá ser $N(q, q') \longrightarrow s$ (sería trivial) con lo cual el primer paso de la derivación será la aplicación de algúna producción tomando en cuenta las tres reglas restantes (o tipos de travesías restantes de la trivial) de la tabla (2.2).

Caso I: Sea $N(q, q') \longrightarrow N(q, q'')s$ una producción de G donde $\omega = \psi s$ para algúna ψ tal que $N(q, q') \stackrel{*}{\Longrightarrow} \psi$; entonces M deberá tener la instrucción

$$q'' \ ] \ scan(s, q')$$

además por la hipótesis de inducción la derivación $N(q, q'') \stackrel{*}{\Longrightarrow} \psi$ tiene n pasos y

$$(q, \varphi, \sigma) \stackrel{T}{\Longrightarrow} (q'', \varphi\psi, \sigma) \xrightarrow{scan} (q', \varphi\psi s, \sigma)$$

es una travesía de $\psi s = \omega$ de q a q'

Caso II: Sea $N(q, q') \longrightarrow N(q, q'')N(p, p')$ una producción en G donde

$$\omega = \psi\theta$$

tal que

$$N(q, q'') \stackrel{*}{\Longrightarrow} \psi$$

$$N(p,p') \overset{*}{\Longrightarrow} \theta$$

entonces M deberá tener las instrucciones

$$q''\ ]\ write(u,p)$$

$$p'\ ]\ read(u,q')$$

para algún símbolo u en la pila.

Como las derivaciones $N(q,q'') \overset{*}{\Longrightarrow} \psi$ y $N(p,p') \overset{*}{\Longrightarrow} \theta$ no pueden tener más de n pasos es posible construir una travesía para ψ del estado q al estado q' y para θ del estado p al estado p' entonces:

$$(q,\varphi,\sigma) \overset{T}{\Longrightarrow} (q'',\varphi\psi,\sigma) \overset{write}{\longrightarrow} (p,\varphi\psi,\sigma u)$$

$$\overset{T}{\Longrightarrow} (p'\varphi\psi\theta,\sigma u \overset{read}{\longrightarrow} (q',\varphi\psi\theta,\sigma)$$

Es una travesía de $\psi\theta$ del estado q al estado q'

Caso III: Suponemos $N(q,q') \longrightarrow N(p,p')s$ es una producción en G donde $\omega = \theta$ y $N(p,p') \overset{*}{\Longrightarrow} \theta$

Si $N(q,q') \longrightarrow N(p,p')$ es una producción en G hacemos $q'' = q$ y $\psi = \lambda$ en el caso anterior y así :

$$(q,\varphi,\sigma) \overset{write}{\longrightarrow} (p,\varphi,\sigma u) \overset{T}{\Longrightarrow} (p',\varphi\theta,\sigma u) \overset{read}{\longrightarrow} (q',\varphi\theta,\sigma)$$

Es la travesía de ω

Además para algunos estados $q \in B$ y $q'' \in Q$ $\omega \in T(q,q')$ si y sólo si $N(q,q') \overset{*}{\Longrightarrow} \omega$ y como consecuencia

$$L(M) = \{\omega \in S^* | \omega \in T(q,q'), q \in I, q' \in F\}$$

$$= \left\{\omega \in S^* | N(q,q') \overset{*}{\Longrightarrow} \omega, q \in I, q' \in F\right\}$$

entonces $\Sigma \longrightarrow N(q,q')$ es una producción de G si y sólo si $q \in I$ y $q' \in F$. Por lo tanto

$$L(M) = \left\{\omega \in S^* | \Sigma \overset{*}{\Longrightarrow} \omega\right\} \ = \ L(G)$$

Construcción de la Travesía	**Pasos de derivación**
1-. Un movimiento *scan* $(q, \varphi, \sigma) \xrightarrow{scan} (q', \varphi s, \sigma)$	$N(q, q') \Longrightarrow s$
2-. Último movimiento *scan* $(q, \varphi, \sigma) \overset{T}{\Longrightarrow} (q'', \varphi\psi, \sigma$ $\xrightarrow{scan} (q', \varphi\psi s, \sigma)$	$N(q, q') \Longrightarrow N(q, q'')s$ donde $N(q, q'') \overset{*}{\Longrightarrow} \psi$
3-. Último mov.*read* y el mov. correspondiente *write* no es el primer mov. $(q, \varphi, \sigma) \overset{T}{\Longrightarrow} (q'', \varphi\psi, \sigma)$ $\xrightarrow{write} (p, \varphi\psi, \sigma u)$ $\overset{T}{\Longrightarrow} (p', \varphi\psi\theta, \sigma u$ $\xrightarrow{read} (q', \varphi\psi\theta, \sigma)$	$N(q, q') \Longrightarrow N(q, q'')N(p, p')$ donde $N(q, q'') \overset{*}{\Longrightarrow} \psi$ $N(p, p') \overset{*}{\Longrightarrow} \theta$
4-.Último mov.*read* y el mov. correspondiente *write* es el primer mov. $(q, \varphi, \sigma) \xrightarrow{write} (p, \varphi\psi, \sigma u)$ $\overset{T}{\Longrightarrow} (p', \varphi\theta, \sigma u)$ $\xrightarrow{read} (q', \varphi\theta, \sigma)$	$N(q, q') \Longrightarrow N(p, p')$ donde $N(p, p') \overset{*}{\Longrightarrow} \theta$

Cuadro 2.3: Correspondencia entre las travesías y los pasos de derivación

□

Es posible relacionar los árboles de derivación y las travesías tal como se muestra en la siguiente tabla: Además, la construcción es reversible, es decir, si construimos una derivación de la travesía, y posteriormente le construimos a dicha travesía una derivación, siempre obtendremos la travesía original. Así tenemos el siguiente resultado.

Corolario 2 *Si M es un autómata de pila propio y G es la gramática obtenida de M a partir de las reglas de la tabla (2.2), entonces para cada $\omega \in S^*$, la travesía por la cual M acepta a ω tiene una correspondencia uno a uno con la derivación más a la izquierda de ω en G.*

En el último corolario del capitulo 2 se establece que existe una correspondencia uno a uno entre las travesias de un autómata de pila (no deterministico), y las derivaciones izquierdas de la gramática asociada

Capítulo 3

Propiedades de los Autómatas de pila

3.1. Propiedades de los lenguajes independientes del contexto

Los lenguajes de las gramáticas independientes del contexto tienen propiedades de cerradura similares a las de los lenguajes de las expresiones regulares, las cuales se presentan a continuación.

Primero se estudiará la intersección de los lenguajes independientes del contexto L con el conjunto regular R.
Sea

$$M_f = (Q_f, T, P_f, I_f, F_f)$$

un autómata finito para R y

$$M_p = (Q_p, T, U, P_p, I_p, F_p)$$

un autómata de pila para L. Al realizar la intersección de estos dos autómatas obtenemos el autómata de pila

$$M = (Q, T, U, P, I, F)$$

donde $Q = Q_p \times Q_f$, $I = I_p \times I_f$, $F = F_p \times F_f$
El programa de M será el siguiente:

Si M_p tiene	**y M_f tiene**	**entonces M tiene**
$q]scan(s,q')$	$p \xrightarrow{s} q$	$(q,p)]scan(s,(q',p'))$
$q]write(u,q')$	$p \in Q_f$	$(q,p)]write(u,(q',p))$
$q]read(u,q')$	$p \in Q_f$	$(q,p)]read(u,(q',p'))$

Cuadro 3.1: Programa de $L(M_p) \cap L(M_f)$

Teorema 3 *El autómata M reconoce la secuencia de movimientos para ω*

$$((q,p).\lambda,\lambda) \overset{T}{\Longrightarrow} ((q',p'),\omega,\lambda)$$

con $(q,p) \in$ y $(q',p') \in F$ si y sólo si

$$(q,\lambda,\lambda) \overset{T}{\Longrightarrow} (q',\omega,\lambda)$$

está en M_p y

$$p \overset{\omega}{\Longrightarrow} p'$$

está en M_f

Demostración:

Necesidad

Supongamos que $((q,p),\lambda,\lambda) \overset{T}{\Longrightarrow} ((q',p'),\omega,\lambda)$ con $q,q' \in Q_p$ y $p,p' \in Q_f$ como tenemos cuatro tipos de travesías la demostración se hará por casos.

1. La travesía sólo tiene un movimiento

   $$((q,p),\lambda,\lambda) \xrightarrow{scan} ((q',p'),s,\lambda)$$

   Entonces M_p tiene $q]scan(s,q')$ y M_f tiene $p \longrightarrow p'$

2. La travesía tiene más de un movimiento y el último movimiento es de revisión

   $$((q,p),\lambda,\lambda) \overset{T}{\Longrightarrow} ((q'',p''),\varphi,\lambda) \xrightarrow{s} ((q',p'),\varphi s,\lambda)$$

   Entonces existe en M_p una secuencia

   $$(q,\lambda,\lambda) \overset{T}{\Longrightarrow} (q'',\varphi,\lambda) \xrightarrow{s} (q',\varphi s,\lambda)$$

   y en M_f tendremos $p \xrightarrow{\omega} p''$ y $p'' \longrightarrow p'$ entonces $p \longrightarrow p'$

3. La travesía tiene más de un movimiento, el último movimiento es de lectura y su pareja correspondiente el movimiento de escritura no es el primer movimiento.

$$((q,p),\lambda,\lambda) \overset{T}{\Longrightarrow} ((q'',p''),\varphi,\lambda) \overset{write}{\longrightarrow} ((s,r),\varphi,u)$$

$$\overset{T}{\Longrightarrow} ((s',r'),\varphi\psi,u) \overset{read}{\longrightarrow} ((q',p'),\varphi\psi,\lambda)$$

entonces existen $q, q', s, s', q'' \in Q_p$ tales que

$$(q,\lambda,\lambda) \overset{T}{\Longrightarrow} (q'',\varphi,\lambda) \overset{write}{\longrightarrow} (s,\varphi,s) \overset{T}{\Longrightarrow} (s',\varphi\psi,u) \overset{read}{\longrightarrow} (q',\varphi\psi,\lambda)$$

es decir

$$(q,\lambda,\lambda) \overset{T}{\Longrightarrow} (q',\omega,\lambda)$$

con $\omega = \varphi\psi$. Ahora M_f reconoce la cadena ω mediante las siguientes transiciones

$$p \xrightarrow{\omega_1} p'' \xrightarrow{\omega_2} r \xrightarrow{\omega_3} r' \xrightarrow{\omega_4} p'$$

donde $\omega_1\omega_2\omega_3\omega_4 = \omega$ por lo tanto $p \overset{\omega}{\Longrightarrow} p'$.

4. La travesía tiene más de un movimiento el último movimiento es de lectura y su pareja correspondiente el movimiento de escritura es el primer movimiento

$$((q,p),\lambda,\lambda) \overset{write}{\longrightarrow} ((q'',p''),\lambda,u) \overset{T}{\Longrightarrow} ((q''',p'''),\omega,u) \overset{read}{\longrightarrow} ((q',p'),\omega,\lambda)$$

y por la tabla (2.1) M_p tiene

$$(q,\lambda,\lambda) \overset{write}{\longrightarrow} (q'',\lambda,u) \overset{T}{\Longrightarrow} (q''',\omega,u) \overset{read}{\longrightarrow} (q',\omega,\lambda)$$

es decir $(q,\lambda,\lambda) \overset{T}{\Longrightarrow} (q',\omega,\lambda)$, y M_f tiene

$$p \xrightarrow{\omega_1} p'' \xrightarrow{\omega_2} p''' \xrightarrow{\omega_3} p'$$

entonces $p \overset{\omega}{\Longrightarrow} p'$ con $\omega = \omega_1\omega_2\omega_3$

Suficiencia
Supongamos $(q,\lambda,\lambda) \overset{T}{\Longrightarrow} (q',\omega,\lambda)$ y $p \overset{\omega}{\Longrightarrow} p'$, se tiene cuatro tipos de travesías, por lo cual se tendrá que analizar como aplicar la tabla (2.1) para cada uno de ellos.

1. Si M_p tiene
$$(q, \lambda, \lambda) \xrightarrow{scan} (q', \omega, \lambda)$$
y
$$p \xrightarrow{\omega} p'$$
por la tabla (2.1) tenemos
$$((q, p), \lambda, \lambda) \xrightarrow{scan} ((q', p')\omega, \lambda)$$
que es una travesía que consta de un sólo movimiento.

2. Si M_p tiene
$$(q, \lambda, \lambda) \xRightarrow{T} (q'', \varphi, \lambda) \xrightarrow{scan} (q', \varphi s, \lambda)$$
y M_f tiene
$$p \xrightarrow{\omega_1} p'' \xrightarrow{\omega_2} p' \; con \; \omega_1\omega_2 = \omega$$
por la tabla (2.1) M tendrá
$$((q, p), \lambda, \lambda) \xRightarrow{T} ((q'', p''), \varphi, \lambda) \xrightarrow{scan} ((q', p'), \varphi s, \lambda)$$
con $\omega = \varphi s$; es una travesía con más de un movimiento cuyo último movimiento es de revisión.

3. Si M_p tiene
$$(q, \lambda, \lambda) \xRightarrow{T} (q'', \varphi, \lambda) \xrightarrow{write} (s, \varphi, u) \xRightarrow{T} (s', \varphi\theta, u) \xrightarrow{read} (q', \varphi\theta, \lambda)$$
con $\omega = \varphi\theta$, y M_f tiene
$$p \xrightarrow{\omega_1} p'' \xrightarrow{\omega_2} r \xrightarrow{\omega_3} r' \xrightarrow{\omega_4} p'$$
entonces, por la tabla(2.1), M tendrá
$$((q, p), \lambda, \lambda) \xRightarrow{T} ((q'', p''), \varphi, \lambda) \xrightarrow{write} ((s, r), \varphi, u)$$
$$\xRightarrow{T} ((s', r'), \varphi\theta, u) \xrightarrow{read} ((q', p'), \varphi\theta, \lambda)$$
que es una travesía (con $\omega = \varphi\theta$) con más de un movimiento y la pareja correspondiente *write* no es el primer movimiento.

4. Si M_p tiene

$$(q, \lambda, \lambda) \overset{write}{\longrightarrow} (q'', \lambda, u) \overset{T}{\Longrightarrow} (q'''\omega, u) \overset{read}{\longrightarrow} (q', \omega, \lambda)$$

y M_f tiene

$$p \overset{\omega_1}{\longrightarrow} p'' \overset{\omega_2}{\longrightarrow} p''' \overset{\omega_3}{\longrightarrow} p'$$

con $\omega = \omega_1\omega_2\omega_3$ entonces por la tabla (2.1) M tendrá

$$((q, p), \lambda, \lambda) \overset{write}{\longrightarrow} ((q'', p''), \lambda, u) \overset{T}{\Longrightarrow} ((q''', p'''), \omega, u) \overset{read}{\longrightarrow} ((q', p'), \omega, \lambda)$$

que es una travesía con más de un movimiento y el primer movimiento es la pareja correspondiente *write*.

□

Puesto que el complemento de un conjunto regular es regular, y $L - R = L \cap R^c$ entonces $L - R$ será un lenguaje independiente del contexto. Para poder trabajar con la cerradura de los lenguajes independientes del contexto y las operaciones de conjuntos (unión, concatenación e inversa) utilizamos su relación con las gramáticas independientes del contexto.
Dadas dos gramáticas

$$G_1 = (T, N_1, P_1, \Sigma_1)$$
$$G_2 = (T, N_2, P_2, \Sigma_2)$$

dos gramáticas independientes del contexto con el mismo alfabeto terminal T, pero con conjuntos no terminales disjuntos, $N_1 \cap N_2 = \emptyset$ entonces la gramática para el lenguaje resultante de concatenar $L = L(G_1) \cdot L(G_2)$ se obtendrá usando nuevos símbolos terminales A_1, A_2 en lugar de Σ_1, Σ_2 y añadiendo la producción

$$\Sigma \longrightarrow A_1A_2$$

a la únion de P_1 y P_2, y para completar agregamos la producción $\Sigma \longrightarrow \lambda$ si G_1 tiene $\Sigma_1 \longrightarrow \lambda$ y $\Sigma_2 \longrightarrow \lambda$.

La clase de los lenguajes independientes del contexto no son cerradas bajo las operaciones de intersección y complemento, lo cual demostraremos por medio de un contraejemplo; sean

$$L_1 = \{a^n b^n c^m | m, n \geq 0\}$$

$$L_2 = \{a^m b^n c^n | m, n \geq 0\}$$

La intersección de L_1 y L_2 es

$$L_1 \cap \text{Ł}_2 = \{a^n b^n c^n | n \geq 0\} = L_{dm}$$

el lenguaje de doble correspondencia, el cual no es independiente del contexto.
Ahora, si el complemento de un lenguaje independiente del contexto fuera siempre independiente del contexto, podríamos demostrar la cerradura bajo intersección usando las leyes de De Morgan.

$$L_1 \cap L_2 = (L_1^c \cup L_2^c)^c$$

con lo cual el complemento no es independiente del contexto.
Para estudiar el traductor de estado finito de un lenguaje independiente del contexto es útil pensar en términos de un autómata de pila que genere lenguajes independientes del contexto, un *autómata de pila generador (APG)* es un autómata en el cual consideramos la cinta de entrada como cinta de salida (impresión) y reemplazamos cada movimiento *scan*

$$q]scan(s, q')$$

con un movimiento *print*

$$q]print(s, q')$$

que escribe el símbolo s en la cinta de salida. Claramente el APG genera una cadena ω (esto es, escribe ω en la cinta de salida) por la travesía

$$(q, \lambda, \lambda) \overset{T}{\Longrightarrow} (q', \omega, \lambda)$$

si y sólo si la máquina que representa un autómata reconoce ω por la misma secuencia de movimientos.
El autómata generador

$$M_g = (Q_g, S, U, P_g, I_g, F_g)$$

genera cadenas en L que son procesadas por un traductor secuencial generalizado.

$$M_t = (Q_t, S, R, P_t, I_t, F_t)$$

La combinación de M_g y M_t es un nuevo APG

$$M = (Q, R, U, P, I, F)$$

con

$$q = Q_g \times Q_t, \qquad F = f_g \times F_t, \qquad I = I_g \times I_t$$

que justamente generan las cadenas en R^* que son traducidas por M en cadenas en L. Esto es, M genera φ, por la travesía

$$((q, p), \lambda, \lambda) \overset{T}{\Longrightarrow} ((q', p'), \varphi, \lambda)$$

con $(q, p) \in I$, $(q', p') \in F$ si y sólo si M_g genera alguna cadena ω por la travesía

$$(q, \lambda, \lambda) \overset{T}{\Longrightarrow} (q', \omega, \lambda)$$

y φ es la traducción de ω para M

$$p \overset{\omega/\varphi}{\Longrightarrow} p'$$

el programa de M se especifica en la tabla (3.2) y se añade una instrucción adicional

$$q]null(q')$$

La cual nos coloca en el estado q' del autómata sin hacer movimientos en el tope de la pila. La instrucción *null* es análoga a la transición λ en un autómata de estado finito y puede ser reemplazada con el par de instrucciones

$$q]write(u, q'')$$

$$q'']read(u, q')$$

donde q'' es un nuevo estado auxiliar y u es un símbolo arbitrario en la pila (los movimientos impropios después pueden ser eliminados)

Teorema 4 *La clase de los lenguajes independientes del contexto con alfabeto terminal T es cerrada bajo las operaciones de unión, concatenación, cerradura e inversión (reversa), intersección y diferencia con un conjunto regular. Esto es si L_1 y L_2 son lenguajes independientes del contexto y R es un conjunto regular entonces*

$$L_1 \cup L_2 \qquad L_1^R \qquad L_1 \cdot L_2$$

$$L_1 \cap R \qquad L_1^* \qquad L_1 - R$$

son independientes del contexto, sin embargo

$$L_1 \cap L_2 \qquad L_1^* = T^* - L_1$$

no necesariamente son independientes del contexto.

Si M_g tiene	y M_t tiene	Entonces M tiene
$q]write(u, q')$	$p \in Q_t$	$(q,p)]write(u,(q',p))$
$q]read(u, q')$	$p \in Q_t$	$(q,p)]read(u,(q',p))$
$q]print(s, q')$	$p]scan(s, p')$	$(q,p)]null(q',p')$
$q \in Q_g$	$p]print(r, p')$	$(q,p)]print(r,(q,p'))$

Cuadro 3.2: Especificación del programa de un APG para la traducción de $L(M_g)$ por M

3.2. Autómatas de pila determinísticos

Los autómatas de pila determinísticos son una subclase muy importante de los autómatas de pila, por lo cual es necesario, conocer como son y de qué manera se comportan.

Definición 28 *Un* autómata de pila determinístico (APD) *es un autómata de pila* $M = (Q, S, U, P, q_I, F)$ *con un sólo estado inicial, y un programa en el cual sólo una instrucción es aplicable en una configuración dada.*

Un autómata determinístico acepta una cadena ω, si después de revisarla alcanzamos un estado final, sin importar el contenido de la pila; para mostrar que la condición de pila vacía es demasiado estricta tenemos el siguiente lenguaje:

$$L = a^* \cup \left\{a^k b^k | k \geq 1\right\}$$

Si decimos que una cadena es aceptada únicamente alcanzando un estado final, sin importar el contenido de la pila el autómata siguiente reconoce al lenguaje L.
Iniciaremos el autómata con un marcador final (la letra y) para que en caso de aparecer la letra b, el autómata podrá revisar si previamente se encuentra

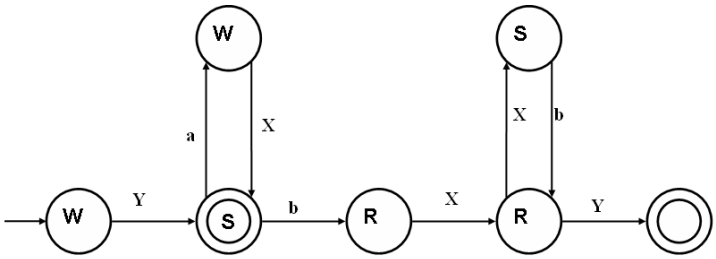

Figura 3.1: autómata para $a^* \cup \{a^k b^k \mid k \geq 1\}$

la letra a correspondiente a la revisión de b. Sin embargo el siguiente argumento muestra que ningún APD puede reconocer a L cuando la condición de pila vacía es la condición de aceptación.
Si un APD acepta a ω y a el prefijo de ω, entonces la secuencia de movimientos aceptada para el prefijo deberá ser la subsecuencia inicial de movimientos para ω

$$\underbrace{\omega_1\omega_2\cdots\omega_{r-1}\omega_r}_{prefijo\,de\,\omega}\,\omega_{r+1}\cdots\omega_n = \omega$$

Suponga que M es un APD de n estados que acepta cada cadena ω de L por la travesía

$$(q_I, \lambda, \lambda) \overset{T}{\Longrightarrow} (q', \omega, \lambda)\ q' \in F$$

Entonces para $p \geq n$ el comportamiento de M para $\omega = a^p b^p$ es:

$$(q_I, \lambda, \lambda) \overset{T}{\Longrightarrow} (q_1, a, \lambda) \overset{T}{\Longrightarrow} \cdots (q_p, a^p, \lambda) \overset{T}{\Longrightarrow} (q', a^p b^p, \lambda)$$

donde (q_i, a^i, λ) denota la configuración por la cual M acepta la cadena a^i. Entre las primeras $p+1$ configuraciones de esta secuencia algunos estados deberán repetirse, suponga por lo tanto que $q_i = q_j = q$, para algún $j \geq 1$.

Esto es suponemos que:

$$(q_I, \lambda, \lambda) \overset{T}{\Longrightarrow} (q, a^i, \lambda) \overset{T}{\Longrightarrow} (q, a^j, \lambda) \overset{T}{\Longrightarrow} (q_p, a^p, \lambda) \overset{T}{\Longrightarrow} \cdots \overset{T}{\Longrightarrow} (q', a^p b^p, \lambda)$$

Entonces debemos tener

$$(q_I, \lambda, \lambda) \overset{T}{\Longrightarrow} (q, a^i, \lambda) \overset{T}{\Longrightarrow} \cdots \overset{T}{\Longrightarrow} (q_p, a^{p-(j-i)}, \lambda) \overset{T}{\Longrightarrow} (q', a^{p(j-i)} b^p, \lambda)$$

Así M acepta la cadena $a^{p-(j-i)}b^p$, la cual no está en L.
Este ejemplo muestra que la condición de pila vacía es una condición que limita las capacidades de la definición del lenguaje de un APD.
En consecuencia adoptamos para el autómata la aceptación por el estado final.

Definición 29 *Una cadena ω es aceptada por un APD M, si M tiene una secuencia de movimientos*

$$(q_I, \lambda, \lambda) \Longrightarrow (q', \omega, \sigma)$$

donde $q' \in F$ y $\sigma \in U^*$, el lenguaje reconocido por M es el conjunto de las cadenas reconocidas por M.

Observamos que la definición anterior es muy completa ya que no permite que un autómata de pila (no determinístico) pueda aceptar lenguajes que no son independientes del contexto, lo cual se verifica convirtiendo algún autómata de pila M en un (posible) autómata determinístico M' que acepta el mismo conjunto de cadenas con la pila vacía, simplemente añadimos a M las instrucciones

$$q]read(u, q_A) \qquad u \in U$$

para cada estado final q de M, y la instrucción

$$q_A]read(u, q_A) \qquad u \in U$$

donde q_A es el nuevo estado final de M'. Estas instrucciones vacían la pila validando alguna secuencia de movimientos de M.
A continuación se muestra una modificación del autómata de la figura anterior (para vacíar su pila)
Usando la modificación descrita anteriormente podemos obtener de cualquier autómata determinístico de pila un autómata de pila equivalente que tenga

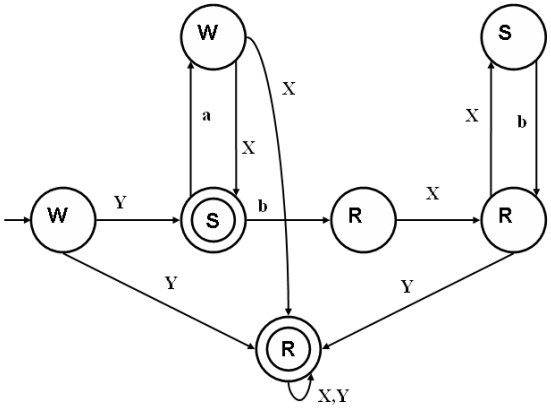

Figura 3.2: autómata con pila vacía para $a^* \cup \{a^k b^k \mid k \geq 1\}$

por condición de aceptación la pila vacía (aunque la máquina modificada no necesariamente será determinística). Por lo tanto, la clase de los lenguajes definidos por los APD es, a lo más, una subclase de los lenguajes independientes del contexto.

Definición 30 *Un lenguaje independiente del contexto es llamado determinístico, si es reconocido por algún autómata de pila determinístico.*

Ejemplo 9 *El lenguaje de correspondencia simple L_m, el lenguaje de los paréntesis L_p y el lenguaje de los paréntesis dobles L_{dp}, son lenguajes determinísticos.*

Los autómatas que construimos para estos lenguajes podrían ser arreglados para aceptar cadenas como lo pide la definición (29), teniendo que comenzar la pila del autómata con un nuevo símbolo y deteniéndonos en un estado final cuando este símbolo es leído.

Observación 11 *Todos los lenguajes regulares son determinísticos puesto que todos son reconocidos por algún autómata determinístico de estado fini-*

to, el cual es un autómata determinístico de pila.

Debido al comportamiento descrito anteriormente, surge la pregunta ¿Es la clase de lenguajes determinísticos una subclase propia de los lenguajes independientes del contexto?, si tomamos el lenguaje de la imagen reflejada, será posible mostrar este hecho; usamos el lenguaje no determinístico

$$L_{nd} = \left\{a^k b^k | k \geq 1\right\} \cup \left\{a^k b^{2k} | k \geq 1\right\}$$

el cual además es usado para estudiar las propiedades de cerradura de los lenguajes determinísticos. Cada cadena en L_{nd} tiene la forma $a^k b^m$ con $m = k$ o $m = 2k$.
Un autómata de pila no determinístico para L_{nd} se muestra en la figura (3.3)

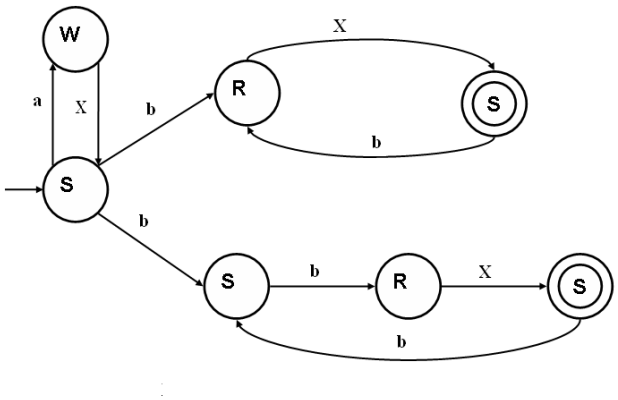

Figura 3.3: autómata de pila para L_{nd}

Intuitivamente el autómata no puede reconocer el lenguaje porque al terminar de revisar todas las $a's$ en ningún momento se tiene base para decidir si las $b's$ corresponden individualmente o en pares (el autómata no determinístico no tiene que tomar esa decisión, es capaz de tomar cualquier alternativa).

Teorema 5 *Los lenguajes determinísticos independientes del contexto son una subclase propia de todos los lenguajes independientes del contexto. En*

particular el lenguaje

$$L_{nd} = \left\{ a^k b^m | m = k \;\; o \;\; m = 2k \;\; k \geq 1 \right\}$$

es independiente del contexto pero no es determinístico.

Demostración:

Puesto que L_{nd} es reconocido por el autómata de la figura(3.3) es un lenguaje independiente del contexto. Ahora supongamos que M es un autómata de pila determinístico que reconoce L_{nd} a partir de dos copias de M podriamos construir el APD M' donde

1. La pila y el tope de la pila son compartidos por ambas copias de M.
2. El estado inicial de M' es el estado final de la copia 1 de M.
3. Los estados finales de M' son los estados finales de la copia 2 de M.

Colocamos el programa de M' para transferir el control de la copia 1 de M a la copia 2 de M inmediatamente después de cualquier movimiento que coloque la copia 1 en un estado final.

4. Cada instrucción etiquetada por un estado final en M

   $$q]mov(x, q'), \quad q \in F$$

   se convierte en la instrucción

   $$q_1]mov(x, q_2)$$

   en M', donde q_1 es el ejemplo de q en la copia 1 de M y q_2 es el ejemplo de q' en la copia 2.

5. Las otras instrucciones se conservan en ambas copias de M. Si M tiene

   $$q]mov(x, q') \quad q \notin F$$

   entonces M' tiene ambas

   $$q_1]mov(x, q_1^{'}) \qquad y \qquad q_2]mov(x, q_2^{'})$$

Por construcción M' es determinístico si M lo es, y afirmamos que

$$L(M') = \left\{a^k b^{2k} | k \geq 1\right\}$$

para ver esto observamos que cada cadena $\omega = a^k b^k$ guía a M' a un estado final de la copia 1, el cual no puede ser un estado final de M' por lo tanto ninguna cadena es aceptada por M'; sin embargo M' acepta cada cadena $a^k b^{2k} = a^k b^k b^k$ por la secuencia de movimientos

$$\underbrace{(q_I, \lambda, \lambda) \Longrightarrow (q', a^k b^k, \sigma_1)}_{en\ la\ copia\ 1} \Longrightarrow \underbrace{(q_2^{'}, a^k b^k b^k, \sigma_2)}_{en\ la\ copia\ 2} \quad q_2^{'} \in F'$$

la cual transfiere el control a la copia 2 inmediatamente después de la secuencia de movimientos para $a^k b^k$.
Ahora, si convertimos M' a M'' cambiando cada movimiento de revisión en la copia 2 de M

$$q_2](scan(b, q_2^{'})$$

a

$$q_2](scan(c, q_2^{'})$$

entonces la secuencia de movimientos reconocida para $a^k b^k b^k$ en M' se convierte en una secuencia de movimientos reconocidos para $a^k b^k c^k$ en M''. Así M'' es un APD que reconoce

$$L_{dm} = \left\{a^k b^k c^k | k \geq 1\right\}$$

el cual no es independiente del contexto. Esta contradicción concluye la demostración del teorema.

□

3.2.1. Propiedades de cerradura de los APD

Las propiedades de cerradura de estos lenguajes difieren significativamente de aquellas de la clase completa de los lenguajes independientes del contexto. En esta clase de lenguajes el complemento de un lenguaje determinístico es siempre un lenguaje determinístico. Para establecer este resultado hacemos uso de la idea de intercambiar los papeles de los estados finales y no finales en un autómata de pila para algún lenguaje determinístico arbitrario. La

construcción que se empleo para demostrar la cerradura de los lenguajes independientes del contexto bajo la intersección con un conjunto regular, se aplica a los lenguajes determinísticos. Si M_p es un APD para L y M_f es un autómata finito determinístico para R, entonces M es un APD para $L \cap R$. La construcción que se empleo para la cerradura bajo un traductor finito, no puede ser trasladada para los lenguajes determinísticos, incluso si restringimos la atención a traductores finitos.

Ejemplo 10 *El lenguaje* $L = \{a^k b^k | k \geq 1\} \cup \{a^k c 2k | k \geq 1\}$*, es determinístico pero la traducción de* L *por la máquina que traslada* $c's$ *en* $b's$ *es un lenguaje no determinístico* L_{nd}*. Así los lenguajes independientes del contexto no son cerrados bajo la clase de máquinas traductoras finitas.*
Para determinar la cerradura de estos lenguajes bajo otras operaciones elementales de conjuntos, usamos el hecho de que los siguientes lenguajes son determinísticos

$$L_1 = \{a^k b^k | k \geq 1\} \quad L_2 = \{a^k b^{2k} | k \geq 1\} \quad L_1 \cup c \cdot L_2 \quad L_1^R \cup c \cdot L_2^R$$

mientras que los lenguajes

$$L_{nd} = L_1 \cup Ł_2 \quad L_1 \cup L_2 \cdot c \quad c \cdot L_{nd}$$

son no determinísticos. Dado que L_{nd} es la unión de lenguajes determinísticos, pero no es en sí mismo determinístico; es decir, la cerradura no se conserva bajo la unión, similarmente el lenguaje

$$L_1^R \cup c \cdot L_2^R = \{b^k a^k \cup c b^{2k} a^k | k \geq 1\}$$

es determinístico, pero su inversa $L_1 \cup L_2 \cdot c$ no lo es.

Sea

$$L_4 = (c \cup \lambda) \cdot (L_1 \cup c \cdot L_2)$$

si la concatenación de conjuntos fuera determinística su intersección con algún conjunto regular sería determinística. Pero la intersección de L_4 y el conjunto regular ca^*b^* es

$$L_4 \cap ca^*b^* = c \cdot L_{nd}$$

lo cual no es determinístico. Finalmente

$$L_5 = c \cup L_1 \cup c \cdot L_2$$

es un lenguaje determinístico. Si su cerradura fuera determinística, entonces

$$L_5^* \cap ca^*b^* = c \cdot L_1 \cup c \cdot L_2 = c \cdot L_{nd}$$

sería también deterministica; pero no lo es. En resumen tenemos:

Teorema 6 *La clase de los lenguajes independientes del contexto determinísticos sobre un alfabeto T, es cerrada bajo operaciones de complemento, intersección con un conjunto regular y diferencia con un conjunto regular, y no es cerrada bajo las operaciones de unión, intersección, concatenación, inversa, cerradura de conjunto o traducción (determinística) por una máquina de estado finito.*
Esto es si L_1 y L_2 son lenguajes independientes del contexto, R es un conjunto regular y M una máquina traductora de estado finito, entonces

$$L_1^c = T^* - L_1 \qquad L_1 \cap R \qquad L_1 - R$$

son determinísticos, mientras que

$$L_1 \cup L_2 \qquad L_1 \cdot L_2 \qquad L_1^R$$
$$L_1 \cap L_2 \qquad L_1^* \qquad M(L_1)$$

no son determinísticos.

3.2.2. Complementos de los lenguajes determinísticos

Sea M un APD para L, deseamos diseñar un APD M_c que alcance el estado final sólo para cada cadena que no esté en L (esto es, tal que $L(M_c) = L^c$). De manera análoga con los autómatas finitos, desearíamos intercambiar los estados finales y no finales de M para obtener M_c. Pero encontramos una dificultad inmediata, en aplicar esta idea, para M no habría secuencias de movimientos para todas las cadenas en S^*; algunas cadenas llevarían a M a configuraciones muertas, desde las cuales el autómata no tiene instrucciones aplicables.

Sea Y el conjunto de las cadenas que no son observadas por secuencia alguna de movimientos de M, y sea M' la máquina obtenida de M intercambiando los estados finales y no finales. Puesto que M' puede no tener una secuencia de movimientos para alguna cadena en Y; M' no reconoce el complemento de $L(M)$.
Una configuración *muerta* (q, φ, σ) puede surgir de dos formas

1. Intentar leer una pila vacía; la etiqueta de q es una instrucción de lectura y $\sigma = \lambda$.

2. Efecto en un programa incompleto

   a) La etiqueta de q es una instrucción de revisión, sin un estado sucesor específico para el siguiente símbolo de entrada.

   b) La etiqueta de q es una instrucción de lectura, sin un estado sucesor para el símbolo tope de la pila.

   c) La etiqueta de q no tiene instrucción.

Podemos modificar un autómata de pila determinístico M tal que las configuraciones muertas no ocurran para ninguna cadena de entrada.

Para prevenir que M intente leer una pila vacía, ampliamos el alfabeto de la pila, incluyendo una nueva letra x que marcará la base de la pila. Añadimos a M un nuevo estado inicial q'_I e inicializamos la pila con la instrucción

$$q'_I]write(x, q_I)$$

entonces para cada instrucción de lectura

$$q]read(u, q')$$

añadimos la instrucción

$$q]read(x, q_R)$$

donde q_R es el nuevo estado (no final) auxiliar. Con estos cambios M tiene un programa incompleto, pero nunca tratará de leer la pila cuando esta se encuentre vacía.
Para completar el programa de M añadimos la instrucción

$$q_R]scan(s, q_R) \qquad s \in S$$

lo que provoca que M revise cualquier símbolo restante de entrada en un estado no final q_R. Además necesitamos añadir nuevas instrucciones

$$q]read(u, q_R) \qquad o \qquad q]scan(s, q_R)$$

para asegurar que cada estado con etiqueta de lectura o revisión en M tiene un estado sucesor en la pila para cada símbolo de entrada, y eliminando de

este modo las configuraciones muertas, además estos cambios no modifican el lenguaje reconocido por M.
La ausencia de configuraciones muertas no es suficiente para garantizar una secuencia de movimientos para cada cadena de entrada en S^*; la máquina podría entrar en un ciclo interminable de instrucciones de lectura y escritura; sin embargo si M es propio y libre de ciclos esto no ocurrirá. Por lo tanto, hacer estos cambios sobre un autómata de pila propio y libre de ciclos producirá un autómata determinístico que tiene una secuencia de movimientos para cada cadena de entrada posible.

Definición 31 *Un autómata determinístico de pila se llama* completo *si cada cadena $\omega \in S^*$ es observada por una secuencia de movimientos que comienza desde la configuración inicial (q_I, λ, λ).*

Al eliminar los movimientos impropios y los ciclos de un autómata de pila determinístico podemos asegurar que los APD completos pueden ser encontrados para cualquier lenguaje determinístico.

Proposión 9 *Cada lenguaje determinístico independiente del contexto es reconocido por algún autómata de pila determinístico completo.*

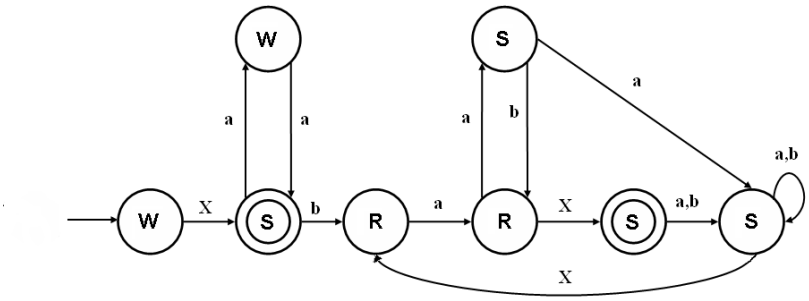

Figura 3.4: autómata con pila vacía para $a^* \cup \{a^k b^k \mid k \geq 1\}$

Ejemplo 11 *Un autómata de pila determinístico completo para*

$$L = a^* \cup \left\{a^k b^k | k \geq 1\right\}$$

se muestra en la figura (3.4). Puesto que el autómata de la figura ya usa y como marcador para el fin de la pila, la modificación de la pila vacía no es necesaria.

El problema restante es obtener un APD completo para un lenguaje determinístico arbitrario, esto es: entre la ejecución de dos movimientos de revisión, el autómata completo podría pasar a través de la secuencia de estados

$$(q_{p-1}, \varphi, \sigma_{p-1} \xrightarrow{s} (q_p, \varphi s, \sigma_p) \longrightarrow \cdots$$
$$\longrightarrow (q_r, \varphi s, \sigma_r) \xrightarrow{s} (q_{r+1}, \varphi st, \sigma_{r+1}) \longrightarrow \cdots$$

algunos de los cuales son finales y algunos no lo son. La cadena φs será reconocida por M si alguno de los estados $q_p \cdots q_r$ es un estado final de M. Si M' es el autómata obtenido de M por intercambio de estados finales y no finales, es probable que algunas cadenas sean reconocidas por ambos M y M', específicamente si q_p es un estado final de M y q_r no lo es, entonces el autómata reconocerá la cadena φs por la secuencia de movimientos

$$(q_I, \lambda, \lambda) \Longrightarrow (q_p, \varphi s, \sigma_p)$$

y M' reconocerá φs por la secuencia de movimientos

$$(q_I, \lambda, \lambda) \Longrightarrow (q_r, \varphi s, \sigma_r)$$

Por lo tanto M' no será el autómata deseado para reconocer $L(M)^c$.
Este problema se resuelve modificando M' así: cada estado final estará etiquetado con alguna instrucción de revisión.

Definición 32 *Sea M un autómata de pila determinístico completo, y sea Q_s el conjunto de estados de M que están etiquetados con la instrucción de revisión; si cada estado final de M pertenece a Q_s, entonces M está en* forma complementable.
Para poner un autómata determinístico en su forma complementable, construimos un autómata M' que consiste de dos copias de M llamadas M_1 y M_2, y deseamos que M' tenga el mismo comportamiento de M, excepto que

el control estará en M_1 o M_2, si el estado final de M ha sido visitado o no desde el último movimiento de revisión.
El programa de M' está especificado por la tabla (3.5), cada instrucción cuyo estado sucesor es un estado final en M, transfiere el control al estado correspondiente en la copia M_2; el control regresa a la copia M_1 con el siguiente movimiento de revisión cuyo estado sucesor es no final en M. El estado inicial de M' es el estado inicial de M_1 o M_2, dependiendo de que q_I sea final o no final en M.
Los estados finales de M' son los estados de M_2 que están etiquetados con la instrucción de revisión, y además, M' es un autómata determinístico complementable que reconoce a $L(M)$.
Ahora, suponemos que M es un APD en forma complementable; como M es completo cada cadena $\omega \in S^*$ es observada por una secuencia de movimientos única que termina en un estado en Q_s

$$(q_I, \lambda, \lambda) \longrightarrow (q', \omega, \sigma) \quad q' \in Q$$

si tomamos M_c que sea M con los estados finales

$$F_c = Q_s - F$$

entonces M_c es un APD complementable determinístico solo para las cadenas rechazadas por M, así tenemos la siguiente

Proposión 10 *Sea L un lenguaje independiente del contexto, éste es determinístico, si y sólo si L^c es determinístico.*

3.3. Autómatas numerables

Sería interesante saber que pasa con la capacidad de un autómata de pila cuando limitamos el tamaño de la pila del mismo.
Si M es un autómata de pila con alfabeto (de la pila) $U = \{u_1, u_2, \cdots, u_n\}$, es fácil obtener una máquina M' con un alfabeto de pila que tenga dos símbolos $\{0, 1\}$ que reconozca el mismo lenguaje. Simplemente adoptamos alguna codificación de los símbolos en U como secuencias binarias de m dígitos, donde $2^m \geq n$, siempre que M tiene una instrucción $write(u, q)$, M' tiene una cadena de m instrucciones de escritura que añaden el código de U a la pila. Donde M tiene una instrucción de lectura, se usa un árbol de altura m de

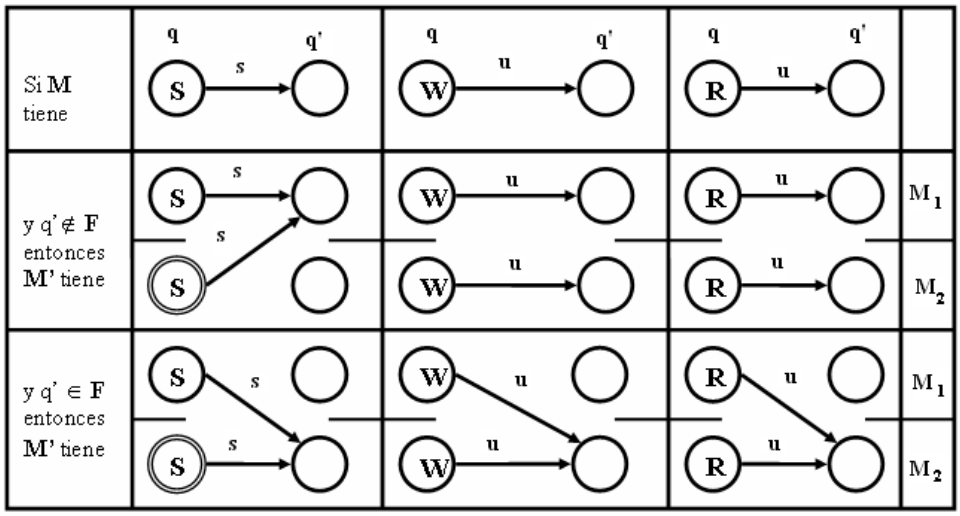

Figura 3.5: Conversión de un APD a su forma complementable

instrucciones de lectura en M' para descifrar m-símbolos de la pila (en orden inverso) y para dirigir la máquina al siguiente estado apropiado.
Así la clase de los autómatas de pila con dos símbolos en la pila, tiene toda la capacidad de la clase general de los autómatas de pila; sin embargo, el argumento anterior no dice nada acerca de los autómatas con un sólo símbolo en el alfabeto de la pila $U = \{\Delta\}$. En tal caso, el contenido de la cinta de entrada siempre es Δ^k, para $k \geq 1$, sin embargo, la información contenida en la pila está completamente determinada por el entero k, específicando la distancia entre la primera casilla de la pila y el tope de ésta, puesto que las instrucciones de escritura y lectura tienen el efecto de incrementar o decrementar el entero k en uno.

Definición 33 *Un* Autómata numerable *es un autómata de pila*

$$M = (Q, S, \{\Delta\}, P, I, F)$$

con un alfabeto de pila de un símbolo y configuración (q, φ, Δ^k) *y se abrevia* (q, φ, k) *donde* k *se llama* el contador*, en lugar de*

$$write(\Delta, q) \qquad y \qquad read(\Delta, q)$$

únicamente escribimos

$$up(q) \qquad y \qquad down(q)$$

Ejemplo 12 *La figura (3.6) muestra un autómata numerable* M_p *con alfabeto de pila* $\{(,)\}$*; esta máquina agrega uno a su contador por cada parétesis izquierdo revisado, y sustrae uno para cada parétesis derecho revisado.*
Si $l(\varphi)$ y $r(\varphi)$, son respectivamente los números de los paréntesis izquierdos y derechos en la cadena φ, la configuración de M_p después de revisar φ será

$$(1, \varphi, l(\varphi) - r(\varphi))$$

así M_p usa la pila para mantener la cuenta del exceso de paréntesis izquierdos sobre los derechos. Dado que la configuración válida es $(1, \varphi, 0)$, M_p reconoce únicamente cadenas con un número igual de paréntesis izquierdos y derechos. Más aún, ningún prefijo φ de una cadena reconocida podría tener $l(\varphi) < r(\varphi)$, porque ninguna configuración de un autómata numerable podría tener un

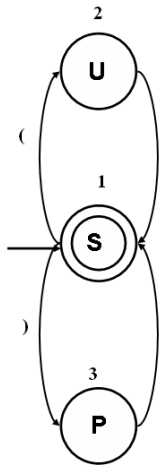

Figura 3.6:

contador negativo (no sería válida).
De lo anterior concluimos que M_p reconoce el lenguaje de los paréntesis balanceados. Una secuencia de movimientos válida para la cadena (()()) es:

$$(1, \lambda, 0) \xrightarrow{s} (2, (, 0) \xrightarrow{u} (1, (, 1)$$

$$\xrightarrow{s} (2, ((, 1) \xrightarrow{u} (1, ((, 2)$$

$$\xrightarrow{s} (3, (()), 2) \xrightarrow{d} (1, (()), 1)$$

$$\xrightarrow{s} (2, (()(, 1) \xrightarrow{u} (1, (()(, 2)$$

$$\xrightarrow{s} (3, (()()), 2) \xrightarrow{d} (1, (()()), 1)$$

$$\xrightarrow{s} (3, ((()())), 1) \xrightarrow{d} (1, ((()())), 0)$$

Puesto que la clase de los autómatas numerables es lo suficientemente poderosa para reconocer el lenguaje de los paréntesis balanceados, los autómatas numerables son más poderosos que los autómatas finitos.

Todas las cadenas en L_p tienen la propiedad de que cada paréntesis izquierdo está correspondida por una subsecuente aparición de un paréntesis derecho.

Aunque el autómata numerable es capaz de examinar la simple estructura del lenguaje de los paréntesis, el reconocimiento de algunos lenguajes independientes del contexto, requiere la correspondencia de varios tipos de símbolos y está más allá de las capacidades de los autómatas numerables; por ejemplo, el lenguaje de los paréntesis dobles L_{dp} deberá revisar que $l(\varphi) < r(\varphi)$ de forma independiente para los dos tipos de paréntesis y además también conservar el registro mostrando el orden en el cual los paréntesis izquierdos fueron revisados.
De lo anterior podemos concluir que ningún autómata numerable es capaz de mantener el registro necesario, para reconocer L_{dp}, sin embargo, sí es capaz de codificar suficiente información dentro de su contador para reconocer L_p.

Teorema 7 *Los autómatas numerables son intermedios en cuanto a alcances (poder) entre los autómatas finitos y los de pila. En partícular existe un autómata numerable que reconoce L_p, pero ninguno que reconzca L_{dp}.*

Demostración:

En el desarrollo del ejemplo anterior se construye un autómata numerable que reconoce L_p. Únicamente resta probar que ningún autómata numerable reconoce L_{dp}.
Sin pérdida de generalidad suponemos que M es propio, entonces para cada cadena $\omega = \{(, [\}$, denotemos $\hat{\omega}$ la cadena inversa de ω. Cada una de las cadenas $\omega\hat{\omega}$ es una cadena simétrica en L_{dp}, formada a través del uso de la siguientes reglas:

1. ( ) y [ ] están en L_{dp}
2. Si $\alpha \in L_{dp}$ entonces también lo están (α) y $[\alpha]$

Para cada entero $r \geq 1$, sea Lr el lenguaje

$$L_r = \{\omega\hat{\omega} \ \mid \omega \in \{(, [\} \ \ y \ \ |\omega| = r\}$$

cada cadena $\omega\hat{\omega}$ en L_m debe ser aceptada por al menos una secuencia de movimientos en M digamos

$$(q, \lambda, 0) \Longrightarrow (q_c, \omega, k_c) \Longrightarrow (q', \omega\hat{\omega}, 0) \qquad q \in I \ \ y \ \ q' \in F$$

donde (q_c, ω, k_c) es la configuración de M inmediatamente después de que esta revisa el último símbolo de ω. Denotemos por $k(\omega)$ el valor más pequeo

de k_c sobre todas las secuencias de movimientos de M que reconocen $\omega\hat{\omega}$. Primero, tenemos una cadena $\omega \in \{(,]\}^r$ para la cual $k(\omega) \geq \frac{2^r}{n}$.
Supongamos que $\omega_1\hat{\omega_1}$ y $\omega_2\hat{\omega}$ son dos cadenas diferentes en L_r entonces M tiene las secuencias de movimientos

$$(q_1, \lambda, 0) \Longrightarrow (q_{c1}, \omega_1, k_{c1}) \Longrightarrow (q_1', \omega_1\hat{\omega_1}, 0) \qquad q_1 \in I \;\; y \;\; q_1' \in F$$

$$(q_2, \lambda, 0) \Longrightarrow (q_{c2}, \omega_2, k_{c2}) \Longrightarrow (q_2', \omega_2\hat{\omega_2}, 0) \qquad q_2 \in I \;\; y \;\; q_2' \in F$$

que aceptan las cadenas. Observamos que no es posible tener cadenas de la forma $\omega_1\hat{\omega_2}$ y viceversa $\omega_2\hat{\omega_1}$, pues éstas no son cadenas de paréntesis balanceados. Así, cada cadena en $\{(,]\}^r$ debe llevarnos a una combinación distinta de q_c y k_c. Como tenemos 2^r elementos distintos en $\{(,]\}^r$ y debe existir al menos un elemento ω con $n \cdot k(\omega) \geq 2^r$.
Ahora, se muestra que si M es un autómata sin ciclos $k(\omega) \geq nr$. Sea ω como en el parrafo anterior, y consideremos que M reconoce una cadena $\omega\hat{\omega}$; dado que $|\omega| = r$, son hechos r-movimientos de revisión, durante el reconocimiento de ω por M; como M es propio no se tienen movimientos de escritura seguidos de movimientos de lectura; además, como M tiene n estados, no más de $n-1$ movimientos se pueden realizar, pues M es sin ciclos. Así $(n-1) \cdot r$ pueden ser realizados mientras se revisa ω por un autómata de pila sin ciclos.
Como la elección de r es arbitraria la relación

$$\frac{2^r}{n} \leq k(\omega) < n \cdot r \qquad \forall r \geq 1$$

Sin embargo, para r muy grande $\frac{2^r}{n} > n \cdot r$, así M no puede ser un autómata numerable que reconozca L_{dp}.

□

3.4. Ambigüedad

Encontramos que cada lenguaje finito, tiene asociada una gramática sin ambigüedad en la cual las derivaciones izquierdas podrían ser colocadas en correspondencia uno a uno con las secuencias reconocidas de un autómata finito determinístico. Sin embargo para los lenguajes independientes del contexto, no es tan simple; sería interesante saber si los lenguajes independientes del contexto están relacionados con los lenguajes determinísticos, o si los lenguajes no ambiguos deberán ser determinísticos, o si un lenguaje determinístico

bebe ser ambiguo.

En el último corolario del capítulo 2 se establece una correspondencia uno a uno entre las travesías en un autómata de pila no determinístico y las derivaciones izquierdas en la gramática asociada. No podemos concluir de forma directa que los lenguajes determinísticos no son ambiguos porque las secuencias de movimientos válidas de un APD no necesariamente son travesías. Para demostrar que los lenguajes determinísticos no son ambiguos mostramos que cualquier autómata de pila determinístico puede ser provisto de instrucciones que vacían la pila, permitiendo que éste acepte cada cadena por una travesía única.
Sea M un APD sin pérdida de generalidad suponemos que M está en forma complementable, y cada estado final de M solamente está etiquetado con la instrucción de revisión. Para cada cadena $\omega \in L(M)$ hay una única secuencia de movimientos válida para que M llegue a un estado con una instrucción de revisión. Antes de construir una gramática para $L(M)$, convertimos M a una máquina M' que acepte las cadenas de $L(M)$ con la pila vácia. Esto es fácil de hacer añadiendo las instrucciónes a M que se muestran en la figura (3.7).

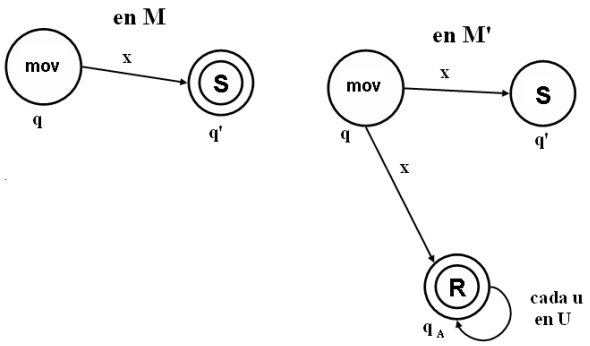

Figura 3.7:

Siempre que M tenga una instrucción cuyo estado sucesor sea el estado final, añadimos una instrucción duplicado de la cual el estado sucesor es q_A, el estado final de M', que será el estado donde M' vaciará su pila. Aunque el autómata de pila nuevo no es determinístico su secuencia de movimientos válidos para cualquier cadena sigue siendo única.
Una cadena ω que lleva M al estado final q' podría llevar a M' ya sea al estado q' o a el estado q_A.

$$(q_I, \lambda, \lambda) \Longrightarrow (q, \varphi, \sigma) \longrightarrow (q', \omega, \sigma) \xrightarrow{s} \cdots$$

o

$$(q_I, \lambda, \lambda) \Longrightarrow (q_A, \omega, \sigma) \Longrightarrow (q_A, \omega, \lambda)$$

ω sólo es reconocida en el último caso, puesto que en el primer caso la máquina está en un estado no final, y es necesario que el movimiento de revisión sea el siguiente.
Ahora, sea G la gramática independiente del contexto construida para M' (de acuerdo con sus travesías). Si G' fuera ambigua, habría al menos dos derivaciones izquierdas distintas de alguna cadena ω, lo cual implica que M acepta a ω por dos o más secuencias distintas de movimientos; de esta contradicción concluimos que

Corolario 3 *La clase de los lenguajes determinísticos independientes del contexto es una subclase propia de los lenguajes independientes del contexto no ambiguos.*

Capítulo 4

Lenguajes independientes del contexto

Caracterizar las oraciones válidas de un lenguaje, como el conjunto de cadenas, generadas por alguna gramática independiente del contexto, es un paso esencial al aplicar muchos métodos prácticos del anális sintáctico. En los capítulos anteriores hemos probado algunos teoremas básicos acerca de la estructura de los lenguajes independientes del contexto. Estos teoremas podrían ser usados para distinguir los lenguajes finitos de los lenguajes independientes del contexto.

Definición 34 *Sea G una gramática independiente del contexto, una producción $A \longrightarrow \alpha$ en la gramática se llama* regla de G. *El símbolo no terminal A es la parte izquierda de la regla, y la cadena α es la parte derecha de la regla.*

En una producción de la forma $A \longrightarrow x\beta$, la letra $x \in N \cup T$ se llama *manipuladora de la producción.* Una producción de la forma $A \longrightarrow B$ con A y B símbolos no terminales, se llama *producción no generativa.* Si G permite la derivación

$$A \stackrel{*}{\Longrightarrow} A\varphi$$

entonces la letra no terminal A se dice *recursiva izquierda.* Si G permite la derivación

$$A \stackrel{*}{\Longrightarrow} \varphi A$$

entonces A se dice *recursiva derecha*. Si G permite la derivación

$$A \stackrel{*}{\Longrightarrow} A$$

en uno o más pasos entonces se dice que A es un *no terminal cíclico*.

4.1. Transformación de gramáticas

Hasta este momento hemos caracterizado la equivalencia de gramáticas, si generan el mismo lenguaje; sin embargo, si las gramáticas se transforman en alguna otra forma, nos interesa preservar la *estructura* del lenguaje generado representado por los árboles de derivación; pues se puede tener el caso en que una gramática genere varias oraciones, por una sóla derivación, mientras que las mismas oraciones puedan ser generadas, por varias derivaciones en la otra gramática; por lo cual nuestra afirmación inicial; no es un buen argumento para la equivalencia de gramáticas.

4.1.1. Equivalencia de gramáticas

En el siguiente ejemplo se presentan dos gramáticas que generan el mismo lenguaje, sin embargo la forma en que derivan las cadenas reconocidas, y más aún su estrucctura es muy diferente.

Ejemplo 13 *Sean*

$$G_1 : \Sigma \longrightarrow S \qquad G_2 : \Sigma \longrightarrow S$$

$$S \longrightarrow S01 \qquad S \longrightarrow S0S$$

$$S \longrightarrow 1 \qquad S \longrightarrow 1$$

que generan el lenguaje

$$L = 1(01)^*$$

Ambas gramáticas generan el mismo lenguaje, lo único que las hace diferentes es que G_2 es ambigua y G_1 no lo es; de aqui podemos concluir que G_1 y G_2 no son estructuralmente equivalentes; en particular G_2 asocia muchos árboles de derivación distintos para oraciones en el lenguaje, mientras que G_1 asocia un árbol de derivación único.
Una definición de equivalencia apropiada para la transformación de gramáticas

debe igualar a las gramáticas G_1 y G_2, sólo si las mismas oraciones son ambiguas en cada gramática. Esto puede lograrse pidiendo que el árbol de derivación izquierdo de G_1 y G_2 tenga una correspondencia uno a uno para cada oración generada. Aunque este criterio podría ser estricto, para una forma de ambigüedad estructural es facílmente removido. Consideremos el siguiente ejemplo

Ejemplo 14

$$G_1 : \Sigma \longrightarrow S$$

$$S \longrightarrow S01$$

$$S \longrightarrow 1$$

y

$$G_1 : \Sigma \longrightarrow S$$

$$S \longrightarrow S$$

$$S \longrightarrow S01$$

$$S \longrightarrow 1$$

La gramática G_2 tiene todas las producciomes de G_1 más la regla $S \longrightarrow S$. Puesto que esta regla podría ser aplicada un número arbitrario de veces para cualquier oración que contenga a S cualquier cadena generada por G_2 tiene una infinidad de derivaciones distintas. En cambio G_1 genera cada cadena por una derivación única. Aunque G_2 es *muy* ambigua, la única diferencia entre la descripción estructural de cualquier oración de acuerdo a las dos gramáticas es el número de repeticiones de ciertas derivaciones en cada derivación.
Las derivaciones que contiene repeticiones de oraciones surgen sólo de las gramáticas que tienen letras no terminales cíclicas, como los símbolos no terminales cíclico son improductivos, y queremos removerlos; el criterior de equivalencia podría permitir remover la ambiguedad asociada. Por lo tanto se observa que la derivación izquierda contiene oraciones no repetidas, proporcionando la descripción estructural correcta de las oraciones generadas por la gramática.

Definición 35 *Una derivación izquierda permitida por una gramática independiente del contexto se llama* mínima*, si ningúna oración es repetida en la derivación.*

Definición 36 *Dos gramáticas independientes del contexto G_1 y G_2 se dicen* débilmente equivalentes *si $L(G_1) = L(G_2)$. Y se dicen* fuertemente equivalentes*, si son débilmente equivalentes y para cada cadena ω, su derivación mínima izquierda permitida por G_1, puede ser colocada en correspondencia uno a uno con la permitida por G_2.*

Observación 12 *Ambas definiciones (débilmente y fuertemente) son relaciones de equivalencia en la clase de todas las gramáticas independientes del contexto*

Para las gramáticas que contienen símbolos no terminales cíclicos, cada derivación izquierda es mínima; la equivalencia fuerte depende solo de la existencia de una correspondencia uno a uno entre las derivaciones izquierdas.

A continuación presentamos un ejemplo de equivalencia fuerte

Ejemplo 15 *Consideremos las gramáticas*

$$\begin{array}{lll} G_1: & G_2: & G_3: \\ \Sigma \longrightarrow A & \Sigma \longrightarrow A & \Sigma \longrightarrow A \\ A \longrightarrow a & A \longrightarrow a & A \longrightarrow a \\ A \longrightarrow aB & A \longrightarrow abcA & A \longrightarrow aB \\ B \longrightarrow bC & & B \longrightarrow bC \\ C \longrightarrow cA & & C \longrightarrow cA \\ & & A \longrightarrow abcA \end{array}$$

La gramática G_2, se obtiene reemplazando las reglas de G_1, usadas en la derivación

$$A \Longrightarrow aB \Longrightarrow abC \Longrightarrow abcA$$

con la regla

$$A \longrightarrow abcA$$

observamos que $L(G_1) = L(G_2)$. Una derivación única de la cadena ω en G_1 puede ser construida de una derivación de ω en G_2 reemplazando cada producción $A \longrightarrow abcA$ con aplicaciones de las reglas

$$A \longrightarrow ab \qquad B \longrightarrow bC \qquad C \longrightarrow cA$$

así, G_1 y G_2 son fuertemente equivalentes. Las producciones de la gramática G_3 son la unión de las producciónes de G_1 y G_2, por lo tanto $L(G_3) =$

$L(G_1)$; G_3 no puede ser fuertemente equivalente con G_1 o G_2, porque G_3 es ambigua (y no tiene símbolos no terminales cíclicos), mientras que G_1 y G_2 son ambiguas.
Existen transformaciones que describen las condiciones bajo las cuales una gramática transformada es equivalente a la original.

4.1.2. Sustitución y Expansión

Se dice que una gramática es transformada por *sustitución*, cuando para algún símbolo no terminal B, la parte derecha de las reglas de B es sustituida por producciones de B, en la parte derecha de alguna otra producción en la gramática independiente del contexto. Supongamos G es independiente del contexto y tiene la producción

$$A \longrightarrow \varphi B \psi \qquad B \in N, \quad \varphi, \psi \in (N \cup T)^*$$

y las reglas

$$B \longrightarrow \beta_1,\ B \longrightarrow \beta_2,\ \cdots, B \longrightarrow \beta_n$$

la transformación de G por sustitución para B, en la producción $A \longrightarrow \varphi B \psi$ en la gramática G' es obtenida de acuerdo con las siguientes reglas

R1 : $A \longrightarrow \varphi B \psi$ no está en G'

R2 : Todas las demás producciones de G están en G'

R3 : Cada producción $A \longrightarrow \varphi \beta_i \psi$ con $i = 1, \cdots, n$ están en G'

Ejemplo 16 *Considere las gramáticas*

$$\begin{array}{lcl} G_1: & \Sigma \longrightarrow S & \\ & S \longrightarrow TT & \\ & T \longrightarrow c & \\ & T \longrightarrow S & \\ & T \longrightarrow aSb & \end{array} \qquad \begin{array}{lcl} G_2: & \Sigma \longrightarrow S & \Sigma \longrightarrow ST \\ & T \longrightarrow S & S \longrightarrow cT \\ & T \longrightarrow c & S \longrightarrow aSbT \\ & T \longrightarrow aSb & \end{array}$$

La gramática G_2 resulta de sustituir la parte izquierda de T en la producción $S \longrightarrow TT$ de G_1. La producción $S \longrightarrow TT$ es omitida en G_2, pero las reglas de T en G_1 son conservadas. La gramática G_2 es fuertemente equivalente a

G_1.

Si la producción introducida por la sustitución es un duplicado de una producción en la gramática original, la transformación de la gramática necesita no ser fuertemente equivalente a la original. Por ejemplo, si la gramática G tiene las producciones

$$A \longrightarrow \varphi B\psi \qquad B \longrightarrow \beta \qquad A \longrightarrow \varphi\beta\psi$$

y sustituimos a B en la primera regla de A obtenemos la producción

$$A \longrightarrow \varphi\beta\psi \qquad B \longrightarrow \beta$$

en la gramática transformada G'. Mientras que G permite dos derivaciones de la oracion $\varphi\beta\psi$ desde A

$$A \Longrightarrow \varphi B\psi \Longrightarrow \varphi\beta\psi$$

$$A \longrightarrow \varphi\beta\psi$$

G' sólo permite la segunda derivación.

Proposión 11 *(**Sustitución**) Sea G una gramática independiente del contexto y G' una gramática construida a partir de G, sustituyendo algún símbolo no terminal B en la producción $A \longrightarrow \varphi B\psi$; si ninguna producción de G' introducida a través de la regla (3) de la sustitución, ha sido ya introducida a través de la regla (2), entonces G y G' son fuertemente equivalentes.*

Demostración
Para establecer la equivalencia fuerte de G y G', es necesario que para cada cadena ω se establecer una correspondencia uno a uno, entre la derivación izquierda mínima permitida por G, y aquellas que son permitidas por G'.

Sea ω alguna cadena en $L(G)$, y supongamos que tenemos una derivación de ω permitida por G, si en la derivación no se emplea la produción $A \longrightarrow \varphi B\psi$, la derivación está permitida por G', por otra parte si la derivación si emplea la producción, entonces la última sustitución de $A \longrightarrow \varphi B\psi$, estará seguida de la aplicación de alguna regla de B en G

$$\Sigma \Longrightarrow \cdots \Longrightarrow \eta A\xi \Longrightarrow \eta\varphi B\psi\xi \Longrightarrow \cdots$$

$$\Longrightarrow \sigma B\tau \Longrightarrow \sigma\beta\tau \Longrightarrow \cdots \Longrightarrow \omega$$

en la derivación correspondiente de ω en G', el paso

$$\eta A\xi \Longrightarrow \eta\varphi B\psi\xi$$

se reemplaza por

$$\eta A\xi \Longrightarrow \eta\varphi\beta\psi\xi$$

y el paso

$$\sigma B\tau \Longrightarrow \sigma\beta\tau$$

se omite. En caso de que se vuelva a aplicar la producción $A \longrightarrow \varphi B\psi$, en la derivación original repetiremos los pasos anteriores.
La nueva derivación es mínima pues no se aplican reglas no generativas, esto es las derivaciones fueron hechas por la izquierda y no se modificó el orden en que se deben hacer.

□

De la misma forma es posible invertir el procedimiento anterior para obtener una derivación mínima única de ω en G a partir de cualquier derivación de G'. A este proceso contrario a la sustitución lo llamamos *expansión*. Suponga que reemplazamos una producción de la forma

$$A \longrightarrow \varphi\psi \qquad \varphi\psi \in (N \cup T)^* - \lambda$$

en una gramática independiente del contexto con alguna de las siguientes producciones

$$A \longrightarrow X\psi \qquad X \longrightarrow \varphi$$
$$A \longrightarrow \varphi X \qquad X \longrightarrow \psi$$

donde X es el nuevo símbolo no terminal. En cualquiera de los dos casos se dice que se *expande* la regla $A \longrightarrow \varphi\psi$. Si la gramática G' se obtiene de G por expansión de una regla $A \longrightarrow \varphi$, entonces G podría ser obtenida de G'. Por lo tanto la proposición anterior y la simetría de la equivalencia fuerte, muestran que G y G' son fuertemente equivalentes.

Proposión 12 *(**Expansión**) Si la gramática G' está formada a partir de una gramática independiente del contexto G por expansión, entonces G y G' son fuertemente equivalentes.*

4.1.3. Producciones inútiles y pruebas de vacío

Una gramática podría contener producciones y símbolos no terminales que pueden resultar en alguna derivación de cadenas no reconocidas.

Definición 37 *Sea* $G = (N, T, P, \Sigma)$ *cualquier gramática independiente del contexto; se dice que una producción* $A \longrightarrow \alpha$ *de* G *es* útil *si* G *permite una derivación*

$$\Sigma \stackrel{*}{\Longrightarrow} \varphi A \psi \Longrightarrow \varphi \alpha \psi \stackrel{*}{\Longrightarrow} \omega \qquad \omega \in T^*$$

de otra manera se dice que $A \longrightarrow \alpha$ *es* inútil.
Un símbolo terminal de G es útil si es la parte izquierda de una producción útil, en otro caso es un terminal inútil. Para poder identificar las producciones útiles en una gramática independiente del contexto se realiza un proceso llamado *marcado*; del cual existen dos tipos el $marcado - T$ y el $marcado - \Sigma$.

El procedimiento $marcado - T$ marca cada regla de G con el símbolo T sólo si hay una derivación en la cadena reconocida que emplea dicha regla. El conjunto de producciones con $marcado - T$ se denota por P_T y el conunto de no terminales que son llamados $T - conectados$ es el siguiente:

$$N_T = \{A \in N \cup \{\Sigma\} \mid es \ \ una \ \ regla \ \ A \longrightarrow \alpha \ \ en \ \ P_T\}$$

por lo tanto cada producción útil en G debe estar en P_T y cada símbolo no termial útil debe ser un terminal conectado. Sin embargo, que la producción se encuentre en P_T y el símbolo no terminal en N_T no es suficiente para asegurar que la producción o el símbolo son útiles; un símbolo no terminal en N_T o una producción $A \longrightarrow \alpha$ en P_T, sólo son útiles si la oración que contiene el símbolo A puede ser derivada a partir de Σ. Para poder analizar si la oración puede ser derivada, empleamos el segundo procedimiento de marcado.

El procedimiento de $marcado - \Sigma$, marca cada regla $A \longrightarrow \alpha$ en P_T, con el símbolo Σ sólo si G permite la derivación

$$\Sigma \stackrel{*}{\Longrightarrow} \varphi A \psi$$

usando las reglas en P_T, la gramática debe permitir la derivación

$$\Sigma \stackrel{*}{\Longrightarrow} \varphi A \psi \Longrightarrow \varphi \alpha \psi \stackrel{*}{\Longrightarrow} \omega \qquad \omega \in T^*$$

ya que cada terminal en φ o ψ es $T - conectado$, si denotamos por P_Σ el conjunto de producciones $\Sigma - marcadas$, entonces P_Σ contiene las producciones útiles de la gramática. Ambos procedimientos nos llevan a lo siguiente:

Procedimiento $marcado - T$. Sea G una gramática independiente del contexto con producciones P, es posible construir una sucesión de producciones $P_0, P_1, \ldots$ de subconjuntos de P, y una sucesión de $N_0, N_1, \ldots$ de subconjuntos de N de acuerdo con el siguiente algoritmo.

1. P_0 , $N_0 = \emptyset$.
2. Sea $P_{i+1} = \{A \longrightarrow \alpha \ \ en \ \ P \,|\alpha \ \ esta \ \ en \ \ (N_i \cup T)^*\}\,, \quad i = 0, \ldots$
3. Sea $N_{i+1} = \{A \in N \mid P_{i+1} \ \ tenga \ \ una \ \ regla \ \ A \longrightarrow \alpha\}\,, \quad i = 0, \ldots$
4. Si $P_{i+1} \neq P_i$ ir al paso 2.
5. Sea $P_T = P_{i+1}$ y $N_T = N_{i+1}$

Cada conjunto P_i contiene símbolos no terminales de G de los cuales se deriva una cadena reconocida en i pasos. Entonces $P_i \subseteq P_{i+1} \subseteq P$ para cada i el procedimiento terminará después de un número de pasos no mayor al número de reglas en G.

Procedimiento $marcado - \Sigma$. Sea G una gramática independiente del contexto y sea P_T el conjunto de producciones $T - marcadas$; es posible construir una sucesión $Q_1, Q_2, \ldots$ de subconjuntos de P_T según el siguiente algoritmo.

1. Sea $Q_1 = \{\Sigma \longrightarrow \alpha \in P_T\}$. Sea $i = 1, \ldots$
2. Sea $Q_{i+1} = Q_i \cup \{A \longrightarrow \alpha \in P_T \mid Q_i \ \ contiene \ \ una \ \ regla \ \ B \longrightarrow \varphi A \psi\}$.
3. Si $Q_{i+1} \neq Q_i$ y tenemos que $i = i + 1$, ir al paso 2.
4. $P_\Sigma = Q_{i+1}$.

Cada conjunto Q_i contiene la producción $A \longrightarrow \alpha$ si y sólo si G permite la derivación

$$\Sigma \stackrel{*}{\Longrightarrow} \varphi A \psi \Longrightarrow \varphi \alpha \psi$$

en no más de i pasos usando sólo producciones en P_T. Dado que $Q_i \subseteq Q_{i+1} \subseteq P_T$; este procedimiento se detiene después de un número de pasos no mayor al número de producciones en P_T.

Observación 13 *Los argumentos anteriores nos muestran que las producciones útiles son aquellas que están en P_Σ, despúes de que el proceso de marcado ha concluido. Entonces los procesos de marcado siempre terminan después de un número finito de pasos.*

Proposión 13 *Sea $G = (N, T, P, \Sigma)$ una gramática independiente del contexto. Para cada producción en P, es posible decidir si es o no es útil en G.*

Demostración
Solo es necesario aplicar los procedimientos de $marcado - T$ y $marcado - \Sigma$ junto con la observación anterior.

□

Ejemplo 17 *Sea la siguiente gramática: Hemos numerado las producciones*

$$\begin{array}{lcccc} G\text{:} & \Sigma & \longrightarrow & S & (1) \\ & S & \longrightarrow & AB & (2) \\ & S & \longrightarrow & aS & (3) \\ & S & \longrightarrow & a & (4) \\ & B & \longrightarrow & Sb & (5) \\ & C & \longrightarrow & aC & (6) \end{array}$$

para que sea más fácil trabajar con ellas. Aplicando el procedimiento de $marcado - T$ marcamos la producción (4), pues es una producción que deriva un símbolo terminal, después marcamos las reglas (1) y (3), pues a partir de ellas también llegamos a alguna cadena conformada solamente de símbolos terminales, y por último marcamos la producción (5); así tenemos que

$$P_T = \{(4), (1), (3), (5)\}$$

y

$$N_T = \{\Sigma, S, B\}$$

ahora aplicando el procedimiento de $marcado - \Sigma$, marcamos las reglas (1), (2) y (3), pues la regla (5) no pertenece a la derivación de ninguna cadena reconocida, aunque la parte izquierda de la producción tiene un símbolo no terminal $T -$ conectado. Así las reglas útiles son:

$$P_\Sigma = \{\Sigma \longrightarrow S,\ \ S \longrightarrow aS,\ \ S \longrightarrow a\}$$

$$\begin{array}{rccc} G\text{:} & \Sigma & \longrightarrow & S \\ & S & \longrightarrow & aS \\ & S & \longrightarrow & a \end{array}$$

y la gramática simplificada es:
El procedimiento de $marcado - T$ nos proporciona una forma de revisar si $L(G, A)$ es vacío para alguna $A \in N \cup \{\Sigma\}$, lo cual es cierto si y sólo si A está en N_T, de aqui tenemos el siguiente:

Teorema 8 *(**Prueba de vacuidad**) Para alguna gramática independiente del contexto G, es posible decidir si $L(G, A) = \emptyset$ para cualquier símbolo no terminal en G; en particular es posible decidir si la gramática genera alguna cadena o todas, esto es $L(G) = L(G, \Sigma) = \emptyset$.*

Al borrar las producciones inútiles de la gramática se producirá una nueva gramática fuertemente equivalente a la original, con lo cual se demuestra el teorema.

Proposión 14 *(**Remoción de producciones inútiles**) Si la gramática G' se obtiene a partir de una gramática independiente del contexto G removiendo las producciones inútiles, entonces, G y G' son fuertemente equivalentes.*

4.1.4. Reemplazo de producciones no generativas

Decimos que una gramática sin producciones no generativas y símbolos no terminales cíclicos está en su *forma canónica*, por lo cual es importante establecer procesos para eliminar tales estructuras. Se presentará éste procedimiento mediante un ejemplo.
Sea G la gramática con producciones:

$$\begin{array}{lll} \Sigma \longrightarrow A & A \longrightarrow B & A \longrightarrow aB \\ \Sigma \longrightarrow B & A \longrightarrow C & C \longrightarrow Aa \\ & B \longrightarrow C & A \longrightarrow b \\ & C \longrightarrow B & \end{array}$$

es posible representar en la siguiente figura la sucesión de derivaciones no generativas consecutivas por un autómata finito deterministico $M(G)$, que tiene

un estado para cada símbolo no terminal que aparece en las producciones no generativas de G.

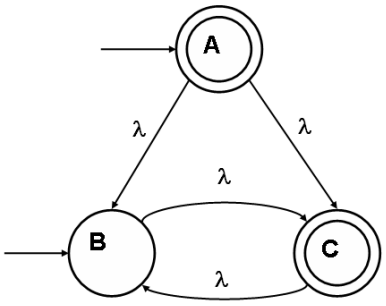

Figura 4.1:

Cada transición λ de la máquina corresponde a una producción no generativa de G, y los estados iniciales y finales corresponden a los símbolos no terminales, en los cuales una serie de pasos no generativos, pueden comenzar o terminar respectivamente.
Supongamos la siguiente trayectoria del estado inicial A al estado final C

$$A \longrightarrow B \longrightarrow C \longrightarrow B \longrightarrow C$$

es decir, la siguiente sucesión de derivaciones

$$\varphi A\psi \Longrightarrow \varphi B\psi \Longrightarrow \varphi C\psi \Longrightarrow \varphi B\psi \longrightarrow \varphi C\psi$$

es permitida en G, la cual esta precedida y seguida por derivaciones generativas. Este es el caso en la derivación de la cadena reconocida ba.

$$\Sigma \Longrightarrow A \Longrightarrow B \Longrightarrow C \Longrightarrow B \Longrightarrow C \Longrightarrow Aa \Longrightarrow ba$$

En general, si $G = (N, T, P, \Sigma)$ es una gramática independiente del contexto con reglas no generativas, el autómata finito asociado es $M(G) = (Q, \emptyset, P', I, F)$, donde

$$Q = \{A | A \longrightarrow B \; o \; B \longrightarrow A \; es \; una \; regla \; de \; G \; para \; algun \; A, B \in N\}$$

$$P' = \left\{ A \xrightarrow{\lambda} B |\; A \; es \; una \; regla \; de \; G \right\}$$

$$I = \left\{ A \in Q \;\middle|\; \begin{array}{l} \Sigma \longrightarrow A \; es \; una \; regla \; de \; G \; o \\ B \longrightarrow \varphi A \psi \; es \; una \; regla \; de \; G \\ con \; \varphi\psi \neq \lambda \end{array} \right\}$$

$$F = \{A \in Q |\; A \longrightarrow \beta \; es \; un \; regla \; de \; G \; con \; \beta \notin N\}$$

Observamos que los conjuntos I y F incluyen todos los símbolos no terminales en los cuales inician y terminan, las secuencias de pasos de derivación con producciones no generativas; incluyendo la secuencia vacía.
Es posible obtener una gramática equivalente a G pero con producciones no generativas, reescribiendo cada producción generativa como varias producciones que dan el efecto de producciones no generativas permitidas por G. La nueva gramática $G' = (T, N', P'', \Sigma)$ incluye un nuevo símbolo no terminal para cada secuencia de pasos no generativos que pueden ocurrir como parte de una derivación mínima en G, cada una de estas secuencias corresponde a una trayectoria libre de ciclos en $M(G)$ desde algún estado inicial X, hasta algún estado final Y.

$$N' = N \cup \left\{ \begin{array}{l} X_1, X_2, \cdots, X_n = Y \in F, X_i \neq X_j \; para \; i \neq j \\ y \; X_1 \xrightarrow{\lambda} X_2 \xrightarrow{\lambda} \cdots \xrightarrow{\lambda} X_n \; en \; M(G) \end{array} \right\}$$

Las reglas de la nueva gramática se diseñan así, siempre que la secuencia de pasos de derivación

$$A \Longrightarrow \varphi X \psi \Longrightarrow \cdots \Longrightarrow \varphi Y \psi \Longrightarrow \varphi B \psi$$

es permitida en una derivación mínima de G, la secuencia

$$A \Longrightarrow \varphi \, [X, \cdots, Y] \, \psi \Longrightarrow \varphi B \psi$$

es permitida por G'. Las producciones de G' son dadas por las siguientes reglas de construcción

1. Si una poducción de G no contiene símbolos correspondientes a un estado en Q, ésta es una producción de G'

2. Si G contiene una producción generativa

$$a \longrightarrow \alpha_0 B_1 \alpha_1 \cdots B_k \alpha_K \qquad k \geq 1$$

en la cual $\{B_i | 1 \leq i \leq k\}$ son los símbolos en la parte derecha de las producciónes correspondientes a los miembros de Q, entonces G' contiene cada producción

$$U \longrightarrow \alpha_0 V_1 \alpha_1, \cdots, V_k \alpha_k$$

tal que

$$U \in \left\{ \begin{array}{l} \{A\} \; si \; A \; no \; es \; un \; estado \; final \; de \; M(G) \\ \{[X \cdots Y] |\, Y = A\} \; si \; A \; es \; un \; estado \; final \; de \; M(G) \end{array} \right\}$$

$$V_i \in \left\{ \begin{array}{l} \{B_i\} \; si \; B_i \; no \; es \; un \; estado \; inicial \; de \; M(G) \\ \{[X \cdots Y] |\, X = B_i\} \; si \; B_i \; es \; un \; estado \; inicial \; de \; M(G) \end{array} \right\}$$

Continuando con la figura (4.1) la gramática G que aumenta a la máquina $M(G)$; necesita de los nuevos no terminales:

$$\{[A], [AC], [BC], [ABC]\}$$

Siguiendo las reglas de construcción, encontramos que ninguna reglas de G es retenida en la nueva gramática, porque cada regla tiene un aspecto de algún símbolo no terminal correspondiente a un estado en Q. Las producciones de G' resultantes de la regla (2) son las siguientes:

$\Sigma \longrightarrow [A]$	$\Sigma \longrightarrow [AC]$	$\Sigma \longrightarrow [BC]$	$\Sigma \longrightarrow [ABC]$
$[A] \longrightarrow b$	$[A] \longrightarrow a[BC]$	$[AC] \longrightarrow [A]a$	$[AC] \longrightarrow [AC]a$
$[AC] \longrightarrow [ABC]a$	$[BC] \longrightarrow [A]a$	$[BC] \longrightarrow [AC]a$	$[BC] \longrightarrow [ABC]a$
$[ABC] \longrightarrow [AC]a$	$[ABC] \longrightarrow [ABC]a$	$[ABC] \longrightarrow [A]a$	

Dada alguna derivación mínima permitida por G, podríamos construir una derivación correspondiente, con G' reemplazando cada secuencia de derivación simple, identificando la secuencia reemplazada, por ejemplo, G permite la derivación

$$\Sigma \Longrightarrow \underbrace{B \Longrightarrow C}_{[BC]} \Longrightarrow \underbrace{Aa \Longrightarrow Ba \Longrightarrow Ca}_{[ABC]} \Longrightarrow \underbrace{Aaa}_{[A]} \Longrightarrow baa$$

que contiene tres secuencias de pasos no generativos (el último vacío), separado en cada caso por un paso generativo. La derivación correspondiente de acuerdo con G' es

$$\Sigma \Longrightarrow [BC] \Longrightarrow [ABC]a \Longrightarrow [A]aa \Longrightarrow baa$$

la correspondencia de las derivaciones ilustradas en este ejemplo se aplica a las gramaticas G y G' siempre que G' sea obtenida de G de acuerdo a las reglas dadas anteriormente.

Proposión 15 *Para cualquier gramática independiente del contexto se puede construir una gramática G' fuertemente equivalente, que contenga producciones no generativas.*

4.1.5. Gramáticas balanceadas

El poder eliminar las producciones no generativas e inútiles motivan la creación de una subclase de una gramatica.

Definición 38 *Una gramática independiente del contexto $G = (N, T, P, \Sigma)$ se dice* balanceada *si cada producción tiene una de las formas:*

$$\Sigma \longrightarrow \lambda \qquad \Sigma \longrightarrow B \qquad A \longrightarrow \alpha$$

donde $A, B \in N$, $\alpha \in (N \cup T)^* - N - \lambda$ y cada producción es útil.

En una gramática balanceada las reglas de Σ, además de $\Sigma \longrightarrow \lambda$ son necesarias para obtener la producción $\Sigma \longrightarrow B$ y las reglas no generativas o inútiles no son permitidas.
La ausencia de producciones no generativas, asegura que cada derivación izquierda, permitida por la gramática sea mínima. Ahora, si G es una gramática independiente del contexto arbitraria, por la proposición (12), es posible expandir todas las reglas de Σ a la forma $\Sigma \longrightarrow B$ y obtener una gramática fuertemente equivalente a la original. La proposición (15), nos permite transformar ésta gramática en una gramática sin reglas no generativas fuertemente equivalente a la original. Empleando la proposición (14) las reglas inútiles podrían ser omitidas produciendo una gramática fuertemente equivalente a la original. Por lo tanto tenemos el siguiente

Teorema 9 *A partir de una gramática independiente del contexto G, es posible construir una gramática balanceada G' fuertemente equivalente a G*

En las gramáticas balanceadas cada producción (diferente de las producciones que parten desde Σ) es de la forma

$$A \longrightarrow \alpha \quad donde \qquad |\alpha| \geq 1 \quad o \qquad \alpha \in T^* - \lambda$$

así, cada paso en una derivación, (con excepción del primero) incrementa la longitud de la oración, o genera al menos un símbolo terminal en la cadena; de aquí podemos decir que la derivación de la cadena reconocida con longitud n necesita a lo más de $2n$ pasos; que consiste de un paso inicial, a lo más $n-1$ pasos que generan n símbolos no terminales y n pasos que reemplacen símbolos no terminales por terminales.

4.2. Formas canónicas de gramáticas

Con base a todo lo que se ha expuesto anteriormente tenemos que, si es posible obtener una gramática fuertemente equivalente a partir de una gramática independiente del contexto arbitraria, diremos que ésta es su *forma canónica*. Las formas canónicas son útiles por dos razones

1. Es más fácil establecer algunas propiedades para las gramáticas en forma canónica.

2. Representar lenguajes independientes del contexto por gramáticas en forma canónica es más fácil, sobre todo para procesos de análisis sintáctico.

Las gramáticas *balanceadas* constituyen una forma canónica de una gramática independiente del contexto, que simplifican su estudio, pues no poseen producciones no generativas e inútiles.
Existen otras dos formas canónicas para las gramáticas independientes del contexto, éstas son

1. La gramática de *forma normal*, en la cual las producciones son restringidas a unas pocas formas muy simples.

2. La gramática de *forma estándar*, la cual es útil para el análisis sintáctico debido a que la recursividad izquierda no está permitida.

4.2.1. Gramáticas en forma normal

Definición 39 *Una gramática independiente del contexto* $G = (N, T, P, \Sigma)$ *se dice que está en* forma normal*, si cada producción tiene una de las siguientes formas*

$$\Sigma \longrightarrow \lambda \quad \Sigma \longrightarrow A \quad A \longrightarrow BC \quad A \longrightarrow a$$

con $A, B, C \in N$ *y* $a \in T$

A esta forma canónica también la conocemos con el nombre de *forma normal de Chomsky*, debido a que no tiene producciones no generativas, una gramática en forma normal es balanceada si no contiene producciones inútiles, el principio para transformar una gramática independiente del contexto G en una gramática G' fuertemente equivalente se ilustra en la siguiente figura.

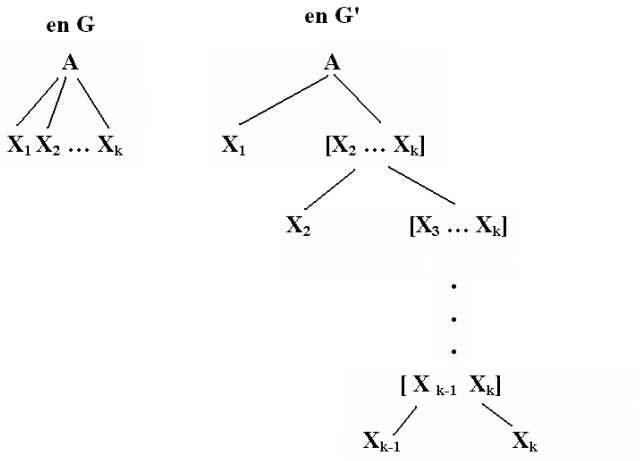

Figura 4.2:

Apoyandonos en el último teorema de la sección anterior podemos asumir que G está balanceada, de donde cada producción de G tiene una de las siguientes formas

Aplicando la proposición (12) a G, es posible construir una gramática G', expandiendo cada regla $A \longrightarrow \alpha$, con $|\alpha| > 2$, en un conjunto de reglas

$$\begin{array}{llll} \Sigma \longrightarrow \lambda & A \longrightarrow \alpha & A \in N & a \in T \\ \Sigma \longrightarrow A & A \longrightarrow a & \alpha \in (N \cup T)^*,\ |\alpha| \geq 2 & \end{array}$$

donde cada una de las partes derechas contengan 2 símbolos. Para cada regla

$$A \longrightarrow X_1 X_2, \cdots, X_k \qquad k \geq 2 \quad X_i \in N \cup T$$

en G, introducimos $k-2$ nuevos terminales

$$[X_2 \cdots X_k]\,[X_3 \cdots X_k], \cdots, [X_{k-1} X_k]$$

y reemplazaremos

$$\begin{array}{c} A \longrightarrow X_1\,[X_2 \cdots X_k] \\ [X_2 \cdots X_k] \longrightarrow X_2\,[x_3 \cdots X_k] \\ \vdots \\ [X_{k-1} X_k] \longrightarrow X_{k-1} X_k \end{array}$$

aplicando la producción $A \longrightarrow X_1 X_2 \cdots X_k$ en G corresponde a aplicar éstas nuevas producciones en secuencia. Cada regla de G se encuentra ahora en alguna de las siguientes formas con $A, B, C, \in N$ y $a, b, c \in T$.

$$\begin{array}{ll} \Sigma \longrightarrow \lambda & A \longrightarrow bC \\ \Sigma \longrightarrow A & A \longrightarrow Bc \\ A \longrightarrow BC & A \longrightarrow bc \\ A \longrightarrow a & \end{array}$$

Las producciones de la columna izquierda están en forma normal, pero las de la columna derecha no; las últimas producciones se colocan en forma normal, introduciendo un símbolo no terminal $[X]$, para cada símbolo terminal x en la gramática, y reemplazando cada ocurrencia de x en cualquier producción por $[X]$. Entonces la regla $[X] \longrightarrow x$ es agregada a G'.
Por ejemplo, una producción de la forma $A \longrightarrow bC$ se convierte en la producción $A \longrightarrow [b]C$, agregando la regla $[b] \longrightarrow b$ a la gramática, ahora, por la proposición (13), los cambios anteriores producen una gramática fuertemente equivalente a la original, de donde obtenemos el siguiente teorema.

Teorema 10 *(**Teorema de la forma normal**) A partir de cualquier gramática independiente del contexto es posible construir una gramática fuertemente equivalente en forma normal.*

4.2.2. Gramáticas de forma estándar

Para este capítulo emplearemos el siguiente resultado:
Regla de Arden Sea Σ un alfabeto y sean $A, B \subset \Sigma^*$. La ecuación $X = Ax + B$ tiene solución $x = A^*B$ y es única si $x \notin A$.

Las gramáticas independientes del contexto en forma estándar tiene símbolos terminales como controladores de todas las reglas, porque no son recursivas por la izquierda, y éstos son muy importantes en el análisis de sintaxis.

Definición 40 *Una gramática independiente del contexto, se encuentra en forma estándar, si cada producción tiene alguna de las formas*

$$\Sigma \longrightarrow \lambda$$

$$\Sigma \longrightarrow A$$

$$A \longrightarrow \alpha\beta$$

con $A \in G$, $a \in T$ *y* $\beta \in (N \cup T)^*$.
A esta forma canónica también se le conoce con el nombre de *forma normal de Greibach*. Dado que no se pueden tener reglas de la forma $A \longrightarrow B$, una gramática en su forma estándar es balanceada si no tiene producciones inútiles.
En una gramática de forma estándar cada paso después del primero en una derivación izquierda tiene la forma

$$\varphi A\psi \Longrightarrow \varphi a\beta\psi$$

y genera al menos un símbolo terminal, por lo tanto la derivación de una cadena reconocida de n símbolos, necesita a lo mas de $n + 1$ pasos, debido a que ninguna producción permitida por la gramática puede ser de la forma

$$A \overset{*}{\Longrightarrow} A\psi$$

la recursividad izquierda es imposible.

Para construir una gramática de forma estándar equivalente a alguna gramática independiente del contexto arbitraria G, sin pérdida de generalidad podemos suponer que G es balanceada, entonces, cada producción de G será de alguna de las siguientes formas

$$\Sigma \longrightarrow \lambda \qquad A \longrightarrow a\alpha$$

$$\Sigma \longrightarrow A \qquad A \longrightarrow B\beta$$
$$A, B \in N \qquad \alpha \in T \qquad \alpha, \beta \in (N \cup T)^* \beta \neq \alpha$$

las producciónes están en forma estándar, con excepción de las del tipo $A \longrightarrow B\beta$, por lo tanto será necesario reemplazar las producciones no terminales, por producciones controladoras en forma estándar.
Sea A algún símbolo no terminal en G y consideremos una derivación izquierda desde A, en la cual cada paso tiene un símbolo no terminal X_i como su controlador

$$A \Longrightarrow X_1\beta_1 \Longrightarrow X_2\beta_2\beta_1 \Longrightarrow \cdots \Longrightarrow X_k\beta_k \cdots \beta_2\beta_1 \Longrightarrow \cdots \tag{4.1}$$

La derivación anterior será finita en longitud, salvo que algún símbolo no terminal se repita en la secuencia

$$A, X_1, X_2, \cdots, X_k, \cdots$$

ahora, si tenemos que $X_i = X_j = X$ para algún $i < j$, la derivación

$$X \overset{*}{\Longrightarrow} X\beta_j, \cdots, \beta_{i+1}$$

se permite, y X es un símbolo no terminal izquierdo recursivo.
En caso de que la gramática G no tenga símbolos no terminales recursivos (recursión izquierda) cada derivación como en (4.1), producirá una oración, la cual comenzará con un símbolo terminal y en un número finito de pasos y sustituciones convertirá las reglas de G a la forma estándar.

Ejemplo 18 *Sea G_1 la gramática con las siguientes reglas*

$$G_1 : \Sigma \longrightarrow A \quad \begin{array}{llll} A \longrightarrow Ba & B \longrightarrow Cb & C \longrightarrow cC \\ A \longrightarrow Ca & B \longrightarrow CAb & C \longrightarrow c \end{array}$$

transformémosla a su forma estándar. Observamos que esta gramática no tiene símbolos no terminales recursivos izquierdos (es recursiva derecha), usando la proposición (11) es posible transformar a G sustituyendo las reglas controladoras de A por

$$A \longrightarrow Ba \quad \textit{se convierte} \begin{cases} A \longrightarrow Cba \\ A \longrightarrow CAba \end{cases}$$

$$A \longrightarrow Ca \quad se\ convierte \begin{cases} A \longrightarrow cCa \\ A \longrightarrow ca \end{cases}$$

el resultado es la gramática G_2

$$\begin{array}{lll} G_2 : \Sigma \longrightarrow A & A \longrightarrow cCa & B \longrightarrow Cb \\ & A \longrightarrow ca & B \longrightarrow CAb \\ & A \longrightarrow Cba & C \longrightarrow cC \\ & A \longrightarrow CAba & C \longrightarrow c \end{array}$$

ahora como las reglas de B *son inútiles es necesario borrarlas; además dos de las nuevas reglas de* A*, no se encuentran en su forma estándar, pero si sustituimos una vez más, se completará la conversión y el resultado será*

$$\begin{array}{lll} G_2 : \Sigma \longrightarrow A & A \longrightarrow cCa & A \longrightarrow aCAba \\ & A \longrightarrow ca & A \longrightarrow cAba \\ & A \longrightarrow cba & C \longrightarrow cC \\ & A \longrightarrow cCba & C \longrightarrow c \end{array}$$

la construcción de G_3 *a partir de* G_1*, satisface la condición dada en la proposición (11); además una producción no transformada duplica cualquier producción que ya estuvierá en la gramática. Así* G_3 *es fuertemente equivalente a* G_1*.*

Si la gramática que se convertirá a su forma estándar tiene símbolos no terminales recursivos izquierdos, el procedimiento que usamos anteriormente nunca termina. Para tal circunstancia usamos el procedimiento mostrado en el siguiente ejemplo.

Ejemplo 19 *Sea* G *la gramática con las siguientes reglas*

$$G_1 : \Sigma \longrightarrow A \qquad A \longrightarrow AbA \qquad A \longrightarrow a$$

observamos que la producción $A \longrightarrow AbA$ *es recursiva izquierda. Denotemos* A *por el conjunto* $L(G_1, A)$ *de acuerdo a la gramática,* A *debe satisfacer la ecuación*

$$A = A(bA) \cup a$$

empleando la regla de Arden obtenemos una expresión alternativa para A

$$A = a(bA)^* = a \cup abA(bA)^*$$

a partir de esta expresión es fácil encontrar producciones de forma estándar que generen las cadenas en A. El conjunto $Z = bA(bA)^*$ *es generado por las producciones* $Z \longrightarrow bAZ$ *y* $Z \longrightarrow bA$*, donde* Z *es un símbolo no terminal nuevo. Sin embargo,* Z *no es un símbolo no terminal controlador, pues la cerradura se desarrolla a través de recursión derecha, más aún estas reglas se encuentran en forma estándar. El conjunto* $A = a \cup aZ$ *es generado por las reglas de A*

$$A \longrightarrow aZ \qquad A \longrightarrow a$$

así la gramática describe el mismo lenguaje que G. Además, cada derivación

$$G_2 : \Sigma \longrightarrow A \quad \begin{array}{ll} A \longrightarrow aZ & Z \longrightarrow bAZ \\ A \longrightarrow a & Z \longrightarrow bA \end{array}$$

en G, que usa recursión izquierda sobre A

$$\Sigma \Longrightarrow A \Longrightarrow AbA \Longrightarrow abA \Longrightarrow abAbA \Longrightarrow \cdots \Longrightarrow (ab)^*A \Longrightarrow (ab)^k a$$

con $k \geq 1$*, correspondiendo únicamente a la derivación*

$$\Sigma \Longrightarrow A \Longrightarrow aZ \Longrightarrow abAZ \Longrightarrow \cdots \Longrightarrow a(ba)^{k-1}Z \Longrightarrow a(ab)^k$$

con $k \geq 1$ *usando la recursión derecha de* Z *en* G_2*.*

Para resolver la recursión izquierda en algún símbolo no terminal X de una gramática arbitraria, se necesitan manejar muchas reglas recursivas izquierdas de X, el siguiente resultado generaliza este proceso.

Proposión 16 *Sea G una gramática independiente del contexto y suponga que las reglas X de G son*

$$X \longrightarrow \alpha_1, \cdots, X \longrightarrow \alpha_n$$

$$X \longrightarrow X\beta_1, \cdots, X \longrightarrow X\beta_n$$

donde el segundo grupo de reglas incluye cada regla X como el controlador. Sea G' *la gramática obtenida, reemplazando las reglas X de G con las reglas*

$$X \longrightarrow \alpha_1, \cdots, X \longrightarrow \alpha_n$$

$$X \longrightarrow \alpha_1 Z, \cdots, X \longrightarrow \alpha_n Z$$

$$Z \longrightarrow \beta_1, \cdots, Z \longrightarrow \beta_m$$
$$Z \longrightarrow \beta_1 Z, \cdots, Z \longrightarrow \beta_m Z$$

donde Z es un símbolo no terminal nuevo. Entonces G y G' son fuertemente equivalentes.

Demostración:
Denotamos por X al conjunto de cadenas en G

$$X = (\alpha_1 \cup \cdots \cup \alpha_n) \cup X(\beta_1 \cup \cdots \cup \beta_m)$$

donde α_i y β_j denotan el conjunto de cadenas derivables de oraciones α_i y β_j. Aplicando la regla de Arden encontramos

$$X = (\alpha_1 \cup \cdots \cup \alpha_n)(\beta_1 \cup \cdots \cup \beta_m)^*$$

el conjunto de cadenas (también denotado por X) en G' es

$$X = (\alpha_1 \cup \cdots \cup \alpha_n) \cup (\alpha_1 \cup \cdots \cup \alpha_n)Z$$

donde

$$Z = (\beta_1 \cup \cdots \cup \beta_m) \cup (\beta_1 \cup \cdots \cup \beta_m)Z$$

si resolvemos la seguna ecuación para Z usando la regla de Arden y sustituyendo el resultado en la primera ecuación tenemos

$$X = (\alpha_1 \cup \cdots \cup \alpha_n) \cup (\alpha_1 \cup \cdots \cup \alpha_n)(\beta_1 \cup \cdots \cup \beta_m)^*$$

$$= (\alpha_1 \cup \cdots \cup \alpha_n)(\beta_1 \cup \cdots \cup \beta_m)^*$$

así X representa la misma cadena en ambas gramáticas, y como solo cambiamos las reglas de X en G, entonces $L(G) = L(G')$. Además G y G' son fuertemente equivalentes porque la derivación en G

$$X \Longrightarrow X\beta_j \Longrightarrow \cdots \Longrightarrow X\beta_{j_k} \cdots \beta_{j_i} \Longrightarrow \alpha_i\beta_{j_k} \cdots \beta_{j_i}$$

está en correspondencia uno a uno, con derivaciones de la forma

$$X \Longrightarrow \alpha_i Z \Longrightarrow \alpha_i\beta_{j_k} Z \Longrightarrow \cdots \Longrightarrow \alpha_i\beta_{j_k} \cdots \beta_{j_i}$$

en G'.

□

Esta proposición forma la base de un procedimiento para convertir cualquier gramática independiente del contexto balanceada a una gramática de forma estándar equivalente. Para este procedimiento es necesaria la siguiente

Definición 41 *Sea G una gramática independiente del contexto y M un subconjunto de símbolos no terminales de G. La* subgramática $G(M)$ *tiene el conjunto de producciones*

$$\{A \longrightarrow \alpha \mid A \in M \ \ y \ \ A \longrightarrow \alpha \ es \ una \ regla \ de \ G\}$$

así, $G(M)$ contiene cada regla de G cuya parte izquierda está en M.
Si G es cualquier gramática balanceada, y M un conjunto de símbolos no terminales tal que ningún elemento es recursivo izquierdo, en la subgramática $G(M)$. El conjunto M no puede contener un símbolo no terminal A si hay una regla $A \longrightarrow A\alpha$ en G; aún así, M podría tener símbolos no terminales que sean recursivos izquierdos en G. Ahora, sea G una gramática balanceada con símbolos no terminales recursivos izquierdos, si deseamos poner a G en su forma estándar, tomamos $G(M)$ una subgramática de G en la cual ningún símbolo no terminal es recursivo izquierdo, y sea X algún símbolo no terminal que es recursivo izquierdo en G; el procedimiento anteriormente mencionado transformará G en una gramática equivalente G' tal que $G'(M \cup \{X\})$ no tenga símbolos no terminales recursivos izquierdos, tal procedimiento consta de dos pasos y se repetirá para cada símbolo no terminal recursivo izquierdo de G, para el caso de las producciones que no se encuentran en forma estándar podemos recurrir a el procedimiento de sustitución. En el primer paso usamos sustitución para transformar las reglas de X en G para que sus símbolos no terminales controladores no sean miembros de M; en el segundo paso empleamos la proposición (16) para reemplazar cualquier regla X recursiva izquierda.

1. Considere cualquier regla X de G con su símbolo controlador en M

$$X \longrightarrow Y\beta \qquad Y \in M$$

ahora, sean las reglas Y de G

$$Y \longrightarrow \alpha_1, \cdots, Y \longrightarrow \alpha_n$$

usando sustitución reemplazamos la regla $X \longrightarrow Y\beta$ con las nuevas reglas

$$X \longrightarrow \alpha_1\beta, \cdots, X \longrightarrow \alpha_k\beta$$

y repetimos este procedimiento hasta que G no tenga las reglas X, como símbolos no terminales controladores en M. La proposición (11) asegura que para cada sustitución hecha duarante la ejecución del paso 1, la gramática transformada genera el mismo lenguaje que la gramática original. Dado que sólo las reglas de X son agregadas a G, el paso 1 no puede introducir una recursión izquierda dentro de la subgramática $G(M)$.

2. Después de aplicar el primer paso las reglas X de G son

$$X \longrightarrow \alpha_1, \cdots X \longrightarrow \alpha_n \, (\textit{en las cuales } X \textit{ no es recursiva izquierda})$$

$$X \longrightarrow X\beta_1, \cdots X \longrightarrow X\beta_m \, (\textit{en las cuales } X \textit{ es recursiva izquierda})$$

en donde ningún símbolo controlador está en M. Reemplazamos las reglas anteriores con las siguientes

$$X \longrightarrow \alpha_1, \cdots X \longrightarrow \alpha_n$$

$$X \longrightarrow \alpha_1 Z, \cdots X \longrightarrow \alpha_n Z$$

$$Z \longrightarrow \beta_1, \cdots Z \longrightarrow \beta_m$$

$$Z \longrightarrow \beta_1 Z, \cdots Z \longrightarrow \beta_m Z$$

donde Z es un nuevo símbolo no terminal.
La proposición (16) nos garantiza que el proceso hecho en el paso 2 no cambia el lenguaje generado pro G, sólo las reglas X y las reglas Z son agregadas, así que todos los símbolos no terminales en M siguen siendo no recursivos. Puesto que X y Z no aparecen como símbolos controladores en ninguna de las reglas nuevas, ninguna de las dos son recursivas izquierdas, así $G(M \cup \{X\})$ no contien símbolos no terminales recursivos izquierdos.

Después de eliminar todas las recursiones izquierdas de la gramática aplicando repetidamente los pasos 1 y 2, los símbolos no terminales de la gramática serán

$$N \cup R \qquad \textit{donde} \;\; R = \{Z_1, \cdots, Z_p\}$$

que contiene los símbolos no terminales introducidos por el paso 2. La gramática aún podría tener reglas con símbolos no terminales como controladores. Dado que ninguna recursión izquierda está presente, estas reglas podrían ser reemplazadas de manera finita con reglas que se encuentren en su forma estándar. Algunas de las reglas Z introducidas por el paso 2 serán no generativas porque algunas de las cadenas $\beta_1, \cdots, \beta_m$ podrían estar formadas solamente por un símbolo terminal. Sin embargo, al sustituir por sus símbolos controladores, se tendrán reglas generativas, debido a que ningún elemento de R puede ser el controlador de ninguna regla Z, y solamente las reglas Z pueden ser no generativas. Como consecuencia de lo anterior tenemos

Teorema 11 ***Teorema de la forma estándar*** *Si G es una gramática independiente del contexto es posible construir una gramática G' en forma estándar tal que G y G' sean fuertemente equivalentes.*

Ejemplo 20 *Sea*

$$G_1 : \Sigma \longrightarrow A \quad A \longrightarrow AaB \quad B \longrightarrow BaC \quad C \longrightarrow c$$
$$A \longrightarrow B \qquad B \longrightarrow C$$

observemos que los símbolos no terminales B son recursivos izquierdos en G, pero C no lo es. Así M inicialmente contiene a C. Primero removemos la recursión izquierda en B. Después de sustituir por el símbolo no terminal controlador de $B \longrightarrow C$ la gramática tiene las reglas

$$B \longrightarrow BaC$$

$$B \longrightarrow c$$

después del paso 2 las reglas se convierten en

$$B \longrightarrow c$$

$$B \longrightarrow cZ$$

$$Z \longrightarrow aC$$

$$Z \longrightarrow aCZ$$

donde Z es un símbolo no terminal nuevo; la gramática resultante aún es recursiva izquierda en A, pero ahora M contiene a B y a C. Después de sustituir por B en la regla $A \longrightarrow B$, las reglas de A en la gramática son

$$A \longrightarrow Aab$$

$$A \longrightarrow c$$

$$A \longrightarrow cZ$$

después de aplicar el paso 2 la primera de estas reglas es remplazada con las reglas

$$A \longrightarrow cZ$$

$$A \longrightarrow cZZ'$$

$$Z' \longrightarrow aB$$

$$Z' \longrightarrow aBZ'$$

donde Z' es un símbolo no terminal nuevo. La nueva gramática completa en forma estándar es

$$\begin{array}{lccc} G_2: & \Sigma \longrightarrow A & A \longrightarrow cZZ' & B \longrightarrow cZ \\ & A \longrightarrow c & Z' \longrightarrow aB & Z \longrightarrow aC \\ & A \longrightarrow cZ & Z' \longrightarrow aBZ' & Z \longrightarrow aCZ \\ & A \longrightarrow cZ' & B \longrightarrow c & C \longrightarrow c \end{array}$$

4.3. Estructura de las gramáticas independientes del contexto

Los lenguajes independientes del contexto frecuentemente contienen colecciones de oraciones en las cuales las frases están anidadas en pares. Para ilustrar este hecho tomemos la producción

$$A \longrightarrow \varphi A \psi$$

donde $\varphi, \psi \neq \emptyset$ son cadenas de símbolos terminales y no terminales; esta producción permite la derivación mostrada en la figura (4.3) y cada una de las oraciones

$$\varphi^k A \psi^k \qquad\qquad k \geq 0$$

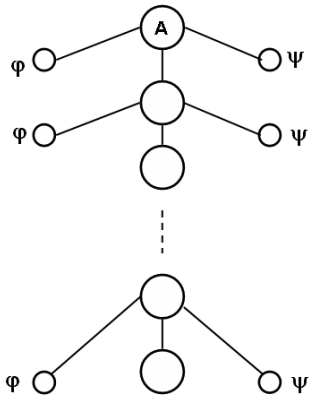

Figura 4.3: árbol de producciones auto generativas

es derivable desde A.

Esta propiedad de *auto producción*, es importante y distingue a los lenguajes independientes del contexto de los lenguajes de estado finito, además de proporcionarnos un criterio para demostrar que ciertos lenguajes no son independientes del contexto, lo anterior se caracteriza mediante tres teoremas. El *teorema de la estructura* qye especifica bajo que circunstancias una gramática independiente del contexto genera una infinidad de oraciones; un corolario de tal teorema, llamado el *Lema de bombeo*, nos dice bajo que condiciones un número infinito de cadenas de un lenguaje independiente del contexto contiene parejas correspondientes de frases. El teorema de *auto-producción* nos dice cuando el lenguaje generado por una gramática independiente del contexto no es un conjunto regular.

4.3.1. Teorema de la estructura y el lema de bombeo

Teorema 12 *(**Teorema de la estructura**) Sea $G = (N, T, P, \Sigma)$ una gramática balanceada. Por cada A en $N \cup \{\Sigma\}$, $L(G, A)$ es infinito si y sólo si G permite la siguiente derivación para algún símbolo no terminal B*

1. $A \overset{*}{\Longrightarrow} \alpha B \beta, \quad \alpha, \beta \in T^*$
2. $B \overset{*}{\Longrightarrow} \varphi B \psi, \quad \varphi, \psi \in T^* - \lambda$
3. $B \overset{*}{\Longrightarrow} \sigma, \quad \sigma \in T^* - \lambda$

donde podemos observar que σ y al menos una de φ, ψ, son no vacías.

Demostración
Necesidad: Supongamos que todas la derivaciones son válidas; entonces para cada $k \geq 0$, la derivación

$$A \overset{*}{\Longrightarrow} \alpha B \beta \overset{*}{\Longrightarrow} \alpha \psi B \psi \beta \overset{*}{\Longrightarrow} \alpha \varphi^k B \psi^k \beta \overset{*}{\Longrightarrow} \alpha \varphi^k \sigma \psi^k \beta$$

es válida. Dado que $\varphi\psi$ es no vacía, cada una de estas derivación genera una cadena bien definida en $L(G, A)$ y por lo tanto el lenguaje es infinito.
Suficiencia: Sea n el número de símbolos no terminales de G, y sea m la longitud de la parte derecha más larga de cualquier producción en G. Supongamos que $L(G, A)$ es infinito, decimos que para una cadena ω suficientemente larga en $L(G, A)$, existe una trayectoria desde un nodo raíz hasta algún nodo terminal, en el árbol de derivación para $A \overset{*}{\Longrightarrow} \omega$ en el cual algún símbolo no terminal B está repetido. Para analizar esto tomemos en cuenta la colección de todos los nodos que pueden ser alcanzados desde el nodo raíz del árbol por una trayectoria de longitud k (o menos).
Dado que cada parte derecha de una produccíon de G , no contiene más de m símbolos, esta colección contiene no más de m^k nodos. Adem á si suponemos que ninguna trayectoria en el árbol contiene la repetición de un símbolo no terminal, entonces cada símbolo terminal deberá encontrarse a una distancia no mayor que $n + 1$ del nodo raíz (esto porque G sólo contiene n símbolos no terminales) pero para cualquier cadena $\omega \in L(G, A)$, el número de nodos hojas para $A \overset{*}{\Longrightarrow} \omega$ es igual a la longitud de ω. Dado que $L(G, A)$ es infinito, podemos escoger ω tal que $|\omega| > m^{n+1}$ y el árbol para $A \overset{*}{\Longrightarrow} \omega$ contendrá alguna trayectoria con repeticiones de un símbolo no terminal.
Se ha demostrado que si $|\omega| > m^{n+1}$, G permite las derivaciones (1),(2) y (3). La cadena σ no puede ser vacía, porque $B \overset{*}{\Longrightarrow} \lambda$ no es una derivación permitida en una gramática independiente del contexto, además como por hipótesis G es balanceada, no puede tener $B \overset{*}{\Longrightarrow} B$, por lo tanto φ, ψ deben ser no vacías.

□

Con ayuda de este teorema es posible decidir si una gramática independiente del contexto genera un lenguaje infinito.

Teorema 13 *Para cualquier G gramática independiente del contexto, es posible decidir si $L(G)$ es finito o infinito.*

Demostración
La existencia de derivaciones de la forma $B \overset{*}{\Longrightarrow} \sigma$, puede ser determinada por el procedimiento de $marcado - T$, y la existencia de derivaciones de la forma $\Sigma \overset{*}{\Longrightarrow} \alpha B \beta$ y $B \overset{*}{\Longrightarrow} \varphi B \psi$ pueden ser determinadas por el procedimiento de $marcado - \Sigma$ (con B en lugar de Σ en el caso de las derivaciones posteriores.)

□

Otro resultado importante y que deriva a partir de la demostración del teorema anterior es el *Lema de bombeo*, el cual nos dice que si es posible encontrar una cadena $\alpha\varphi B\psi\beta$, en el lenguaje, también encontraremos la cadena $\alpha\varphi^k B\psi^k\beta$ $(k \geq 0)$ en el lenguaje.

Teorema 14 ***(Lema de bombeo)*** *Si L es un lenguaje independiente del contexto, existe p entero positivo con la siguiente propiedad:*
Siempre que $\omega \in L$ y $|\omega| > p$, existen cadenas $\alpha, \varphi, \psi, \sigma$ y B con $\varphi, \psi, \sigma \neq \emptyset$ y $|\varphi\sigma\psi| \leq p$, tal que $\omega\alpha\varphi\sigma\psi\beta$ y $\alpha\varphi^k\sigma\psi^k\beta$ está en L, $\forall\ k \geq 0$.

Demostración
Sea G una gramática balanceada que genera a L, supongamos que G contiene n símbolos no terminales, y m es la longitud de la parte derecha más larga de cualquier regla; tomemos $p = m^{n+1}$. La demostración del teorema (12) nos dice que si $\omega \in L$ y $|\omega| > p$ entonces cualquier derivación $\Sigma \overset{*}{\Longrightarrow} \omega$ tiene un árbol en el cual alguna trayectoria desde Σ hasta un nodo hoja debe tener al menos $n + 1$ nodos etiquetados con símbolos no terminales.
Si consideramos la trayectoria más larga trazada desde un nodo hoja hasta la raíz, debemos encontrar dos veces a algún símbolo no terminal B, dentro de $n+1$ movimientos; el cual además será a lo más el $n+1-esimo$ nodo a lo largo de cualquier trayectoria ascendente desde la hoja de un árbol, debido a que la trayectoria que analizamos es la más larga, en la cual hallaremos multiples ocurrencias de un símbolo no terminal.

Así la derivación $B \overset{*}{\Longrightarrow} \varphi\sigma\psi$ satisface $|\varphi\sigma\psi| \leq p = m^{n+1}$. Dado que $\Sigma \overset{*}{\Longrightarrow} \alpha B\beta$, $B \overset{*}{\Longrightarrow} \varphi B\psi$ y $B \overset{*}{\Longrightarrow} \sigma$ y $B \overset{*}{\Longrightarrow} \sigma$ deben ser permitidas por G y $L(G)$ contiene las cadenas $\alpha\varphi^k\sigma\psi^k\beta \ \forall \ k \geq 0$.

□

Observación 14 *El teorema anterior puede ser usado para demostrar que ciertos lenguajes no son independientes del contexto.*

Teorema 15 *La clase de los lenguajes independientes del contexto está contenida propiamente en la clase de los lenguajes sensibles al contexto pero no independientes del contexto. En particular*

$$L_{dm} = \{a^n b^n c^n \ \mid \ n \geq 1\}$$

es sensible al contexto pero no independiente del contexto.
El lema de bombeo es un resultado importante, para demostrar que un lenguaje es independiente del contexto; pero no siempre puede ser usado; entre este caso están las estructuras que necesitan que dos o más derivaciones ocurran en igual número pero en regiones separadas de la oración; otra de las estructuras es aquella en la cual dos o más parejas de derivaciones no están separadas ni anidadas en oraciones del lenguaje pero son requeridas para corresponder independientemente; en estos casos pueden emplearse las propiedades de cerradura de los lenguajes independientes del contexto, junto con el conocimiento de algunos lenguajes independientes del contexto.

4.3.2. Teorema de la auto-producción

Si nosotros tenemos una gramática independiente del contexto, sin derivaciones izquierda o derecha, también podemos generar un lenguaje regular. La propiedad de auto-producción de una gramática, nos da un criterio para probar si el lenguaje representado por la gramática es regular o no. En el estudio de las gramáticas independientes del contexto observamos que las derivaciones de la siguiente forma, son muy comunes.

$$A \Longrightarrow \varphi A\psi \Longrightarrow \ldots \Longrightarrow \varphi^k A\psi^k$$

si tuvieramos una gramática con derivaciones izquierdas o derechas, las producciones anteriores sólo serían posibles si φ o ψ fueran vacías.

Definición 42 *Una gramática independiente del contexto G es* auto productiva *si para algún símbolo útil no terminal A, hay una derivación*

$$A \overset{*}{\Longrightarrow} \varphi A \psi$$

en G con φ y ψ son diferentes del vacío, A es conocido como el símbolo no terminal auto productivo de G.
Si embargo el hecho de que una gramática sea auto-productiva no es por sí mismo un argumento suficiente para asegurar que genera un conjunto regular; así que sólo podemos esperar mostrar que la ausencia de auto-producciones asegura que la gramática genera un conjunto regular. El procedimiento a seguir, es construir una gramática lineal derecha G' equivalente a partir de una gramática G independiente del contexto no auto-productiva y arbitraria. Dado que G puede tener símbolos terminales recursivos tanto izquierdos como derechos, mientras que los símbolos no terminales de G' solo podrán ser recursivos derechos, es conveniente convertir a G a una gramática de forma estándar libre de recursiones izquierdas; sin embargo será necesario verificar que esta conversión no introduce símbolos no terminales auto productivos.

Proposión 17 *Si G es una gramática independiente del contexto no auto-productiva, existe una gramática de forma estándar G' no auto-productiva fuertemente equivalente a G.*

Demostración

Podría verificarse que la conversión de una gramática independiente del contexto arbitraria en una gramática balanceada no introduce símbolos no terminales auto productivos nuevos, así suponemos que G es balanceada. Una gramática de forma estándar fuertemente equivalente a G se obtiene aplicando las transformaciones sucesivas de dos tipos

1. Sustitución de las partes derechas derechas de las reglas por los controladores de las otras reglas.

2. Reemplazamiento de las producciones recursivas izquierdas.

Es necesario demostrar que ninguna de estas transformaciones pueden introducir símbolos no terminales auto productivos. El reemplazamiento de las reglas X recursivas izquierdas en G, se obtiene sustituyendo las reglas de las formas

$$X \longrightarrow \alpha \qquad Z \longrightarrow \beta Z$$
$$X \longrightarrow \alpha Z \qquad Z \longrightarrow \beta$$

por reglas de la siguiente forma

$$X \longrightarrow \alpha \qquad X \longrightarrow X\beta$$

donde Z es un símbolo no terminal nuevo. Debemos demostrar que tanto X como Z no son símbolos auto productivos en la gramática G' resultante. Primero mostramos que no se permite ninguna derivación de la forma

$$\beta \stackrel{*}{\Longrightarrow} \eta X \xi \tag{4.2}$$

por G o G'. En caso de que sea permitida por G, entonces la derivación

$$X \Longrightarrow X\beta \stackrel{*}{\Longrightarrow} X\eta X\xi \Longrightarrow X\eta X\beta\xi$$

también sería permitida, lo cual contradice el hecho de que X no es auto-productiva en G. Supongamos que la derivación (4.2) es permitida en G', dado que β no contiene Z, el uso de alguna regla X en la derivación debe preceder el uso de cualquier regla Z; así G' debe permitir la derivación

$$\beta \stackrel{*}{\Longrightarrow} \eta' X \xi'$$

en la cual no empleamos ninguna regla Z, y entonces esta derivación también es permitida por G, con lo cual se contradice el hecho de que X no es auto-productiva en G. Un argumento análogo puede usarse para mostrar que no se permite ninguna derivación de la forma

$$\alpha \stackrel{*}{\Longrightarrow} \eta X \xi, \qquad \eta \neq \xi$$

por G o G'.
Ahora, si Z es un símbolo terminal auto productivo en G', entonces G' debe permitir la derivación

$$Z \stackrel{*}{\Longrightarrow} \varphi Z \psi$$

el comportamiento auto productivo de Z no puede ser generado sin el uso de la regla $X \longrightarrow \alpha Z$, así la derivación anterior debe incluir la subderivación $\beta \stackrel{*}{\Longrightarrow} \eta X \xi$, lo cual no es posible; por lo tanto concluimos que Z no es auto-productiva en G'.
Supongamos que X es auto-productiva en G'. Entonces una de las derivaciones

1. $X \Longrightarrow \alpha \stackrel{*}{\Longrightarrow} \varphi X \psi$
2. $X \Longrightarrow \alpha Z \stackrel{*}{\Longrightarrow} \varphi X \psi Z$
3. $X \Longrightarrow \alpha \stackrel{*}{\Longrightarrow} \alpha \varphi X \psi$

debe ser permitida por G'. La tercera derivación implica que $Z \stackrel{*}{\Longrightarrow} \varphi X \psi$ es permitida en G' pero esto no necesariamente implica la subderivación en $\beta \stackrel{*}{\Longrightarrow} \eta X \xi$, la cual es imposible. De manera similar para las dos primeras derivaciones, cada una incluye $\alpha \stackrel{*}{\Longrightarrow} \varphi X \xi$ con $\varphi \neq \lambda$, lo cual también es imposible. Así X no puede ser auto-productiva en G'
En una gramática en forma estándar, cada paso en la derivación izquierda tiene la forma

$$\varphi A \psi \Longrightarrow \varphi \alpha \beta \psi$$

donde φ es una cadena de símbolos terminales y $A \longrightarrow \alpha\beta$ es una producción de la gramática. Puesto que β (excepto que se encuentre vacía), siempre contiene el símbolo no terminal izquierdo de la nueva oración, cada paso reescribe el primer símbolo no terminal introducido por el paso anterior. Si la gramática no es ajustada, podemos pensar que sólo un número finito de cadenas claramente definidas $\beta\psi$ podrían aparecer en una oración $\varphi\alpha\beta\psi$. Basándonos en esto es posible construir una gramática lineal derecha fuertemente equivalente y concluir que el lenguaje es regular.

□

Proposión 18 *Si una gramática independiente del contexto G no es auto-productiva y de forma estándar, es posible construir una gramática lineal derecha fuertemente equivalente.*

Demostración

Supongamos que $A \longrightarrow \alpha\beta$ es una producción de G en la cual símbolos terminales aparecen en β. Es posible reemplazar cada aparición de un símbolo terminal x en β con el símbolo no terminal nuevo $[x]$, y agregar la regla $[x] \longrightarrow x$ a G. La gramática resultante es fuertemente equivalente a la gramática dada, y es auto-productiva si y sólo si la gramática dada lo es. Así, sin pérdida de generalidad suponemos que G tiene reglas de la forma

$$\left.\begin{array}{l}\Sigma \longrightarrow A \\ A \longrightarrow \alpha\beta\end{array}\right\} \quad A \in N,\ \alpha \in T,\ \beta \in N^*$$

Sea n el número de símbolos no terminales en G, y sea m la longitud de la cadena más larga de símbolos no terminales en la parte derecha de cualquier producción (m es el valor más grande de $|\beta|$ sobre todas las reglas $A \longrightarrow \alpha\beta$ de G.
Ahora, consideremos cualquier derivación izquierda en G

$$\begin{aligned}\Sigma &\Longrightarrow \alpha_1 A_1 \psi_1 \\ &\Longrightarrow \alpha_1\alpha_2 A_2 \psi_2 \\ &\vdots \\ &\Longrightarrow \alpha_1\alpha_2 \ldots \alpha_r A_r \psi_r\end{aligned}$$

donde la producción aplicada en el i-ésimo paso es

$$A_{i-1} \longrightarrow \alpha_i\theta_i, \qquad A_0 = \Sigma, \qquad \theta_i \in N^* \qquad |\theta_i| \leq m \ \ 1 \leq i \leq r$$

Note que algunas de las θ_i podrían estar vacías. Tenemos

$$|A_r\psi_r| = 1 + (|\theta_1| - 1) + (|\theta_2| - 1) + \cdots + (|\theta_r| - 1) \leq 1 + r\,(m-1) \quad (4.3)$$

porque cada derivación de una producción puede reemplazar un símbolo no terminal, con al menos m símbolos no terminales. Decimos que si G no es auto-productiva $|A_r\psi_r| \leq mn$, de otra forma, si suponemos que $|A_r\psi_r| > mn$, de (4.3) observamos que $r > n$ y debemos tener al menos n pasos en la derivación $\Sigma \overset{*}{\Longrightarrow} \alpha_1 \ldots \alpha_r A_r \psi_r$ deben tener $|\theta_i| > 1$. Dado que G tiene solamente n símbolos no terminales; al menos dos de estos paso deben reemplazar el mismo símbolo no terminal. suponga que $A_i = A_j$ para $i < j$, donde $|\theta_{i+1}| > 1$. Entonces la derivación $\Sigma \overset{*}{\Longrightarrow} \alpha_1 \ldots \alpha_r A_r \psi_r$ incluye los pasos

$$A_i \Longrightarrow \alpha_{i+1} A_{i+1} \varphi_{i+1} \overset{*}{\Longrightarrow} \alpha_{i+1} \ldots a_j A_j \varphi_j$$

donde $A_{i+1}\varphi_{i+1} = \theta_{i+1}$, esto demuestra que los símbolos no terminales $A_i = A_j$ deben ser auto productivos en G, contradiciendo la hipótesis. Se ha mostrado que en cada derivación izquierda

$$\Sigma \overset{*}{\Longrightarrow} \alpha A\psi, \qquad \alpha \in T^*$$

permitida por una gramática G auto-productiva, $|A\psi| \leq mn$. Ahora, definimos una nueva gramática G' con símbolos no terminales $[A\psi]$ y reglas escogidas para que $\Sigma \overset{*}{\Longrightarrow} \alpha[A\psi]$ si y sólo si G tiene $\Sigma \overset{*}{\Longrightarrow} \alpha A\psi$. Los símbolos no terminales de G' son

$$N' = \left\{ [A\psi] \mid \Sigma \overset{*}{\Longrightarrow} \alpha A\psi \text{ en } G, \text{ donde } \alpha \in T^*,\ A \in N,\ \psi \in^* \right\}$$

las producciones de G' son

1. Si G tiene $\Sigma \longrightarrow A$, G' tiene $\Sigma \longrightarrow [A]$.
2. Si G tiene $A \longrightarrow \alpha$, G' tiene $[A\psi] \longrightarrow \alpha[\psi]$ para cada $[A\psi]$ en N'.
3. Si G tiene $A \longrightarrow \alpha\beta$, G' tiene $[A\psi] \longrightarrow \alpha[\beta\psi]$ para cada $[A\psi]$ en N'.

esta construcción está diseñada para poder tener derivaciones de la forma

$$\Sigma \Longrightarrow \alpha_1 A_1\psi_1 \Longrightarrow \alpha_1\alpha_2 A_2\psi_2 \Longrightarrow \ldots \Longrightarrow \alpha_1\alpha_2 \ldots \alpha_r A_r\psi_r$$

permitidas por G están en correspondencia uno a uno con derivaciones de la forma

$$\Sigma \Longrightarrow \alpha_1[A_1\psi_1] \Longrightarrow \alpha_1\alpha_2[A_2\psi_2] \Longrightarrow \ldots \Longrightarrow \alpha_1\alpha_2 \ldots \alpha_r[A_r\psi_r]$$

permitida por G', Así G' es equivalente a G, y dado que G' es una gramática lineal derecha $L(G) = L(G')$ es un conjunto regular.

□

Por la proposición (18) cada gramática no auto-productiva genera un conjunto regular; y un lenguaje que tiene asociada al menos una gramática no auto-productiva, es regular. De forma inversa, cada conjunto regular tiene una gramática lineal derecha no auto-productiva, de donde tenemos el siguiente resultado.

Teorema 16 *(**Teorema de auto-producción**) Un lenguaje independiente del contexto no es regular si y sólo si cada gramática generadora del lenguaje es auto-productiva.*

Conclusiones

Para poder construir un autómata de pila partiendo de una gramática independiente del contexto, fue necesario emplear un procedimiento, el cual analizando los símbolos terminales y no terminales, intercambia las producciones de la gramática por instrucciones en el programa del autómata que deseamos construir. Además se demostró que ambos (el autómata y la gramática) reconocen el mismo lenguaje.

Para el proceso de construir una gramática a partir de un autómata, se emplearon los conjuntos travesía y se buscó una relación entre éstos y los símbolos no terminales que tendrá la gramática construida; de la misma forma que en el caso anterior se demostró que el lenguaje que reconocen ambas estructuras es el mismo. Es posible desarrollar programas de computadora para realizar los procedimientos descritos por los algoritmos.

Por último se presentan los teoremas de estructura para las gramáticas independientes del contexto; los cuales no fuerón demostrados pues se encontraban fuera del alcance de la tesis.

Bibliografía

[1] Denning, Dennis, Qualitz, *Machines, languages and computation*. Editorial Prentice Hall 1978, pp. 260 - 375.

[2] John E. Hopcroft y Jeffrey D. Ullman, *Introducción a la teoría de autómatas, lenguajes y computación*.Editorial CECSA

[3] Howie, John M. *Automata and languages*. Editorial Oxford science publications, 1991, pp. 93-154.

[4] Dean, Kelley. *Teoría de autómatas y lenguajes formales*. Editorial Prentice Hall, 1998, pp. 114-160.