Tema 5:
Autómatas a pila

Teoría de autómatas y lenguajes formales I

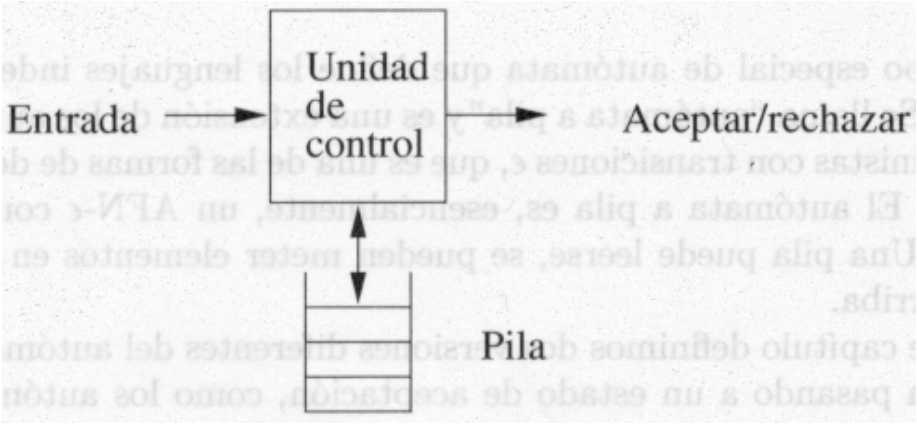
Bibliografía

- Hopcroft, J. E., Motwani, R., y Ullman, J. D. "Introducción a la Teoría de Autómatas, Lenguajes y Computación". Addison Wesley. 2002.
  - capítulos 6 y 7
- Sudkamp, Thomas A. "Languages and machines : an introduction to the theory of computer science". Addison-Wesley Publishing Company, 1998.
  - capítulos 8 y 5

Introducción a los autómatas a pila

- Un autómata a pila (AP) es un AFN con transiciones ε y con una pila en la que se puede almacenar una cadena de "símbolos de pila"
- El AP puede recordar una cantidad infinita de información
  - LIFO
- Los AP reconocen todos los LIC y sólo estos
  - existen lenguajes que no son LIC
    - $\{0^n1^n2^n \mid n \geq 1\}$

Introducción a los AP (II)



- Se consume de la entrada un símbolo, o bien ε
- Se pasa a un nuevo estado
- Se reemplaza el símbolo en lo alto de la pila por una cadena (podría ser ε)

Introducción a los AP (III)

- Ejemplo: diseñar el AP para el lenguaje $L_{wwr} = \{ww^R \mid w \text{ está en } (0+1)^*\}$
  - se comienza en el estado q_0
    - suponemos que la cadena w aún no ha finalizado
    - se almacena una copia de los símbolos de entrada leídos en la pila
  - en cualquier momento se puede suponer que w ha finalizado y se ha comenzado a leer w^R
    - el final de w estará en la cima de la pila
    - se transita al estado q_1
    - AP no determinista:
      - podemos suponer que hemos visto el final de w
      - también podemos continuar en q_0 almacenando las entradas en la pila
  - en el estado q_1 se compara el símbolo de entrada con el símbolo en la cima de la pila
    - son iguales: se elimina el símbolo de la pila
    - son distintos: no habíamos llegado al final de w. Esa rama muere
  - si se vacía la pila, hemos leído ww^R y se acepta la entrada

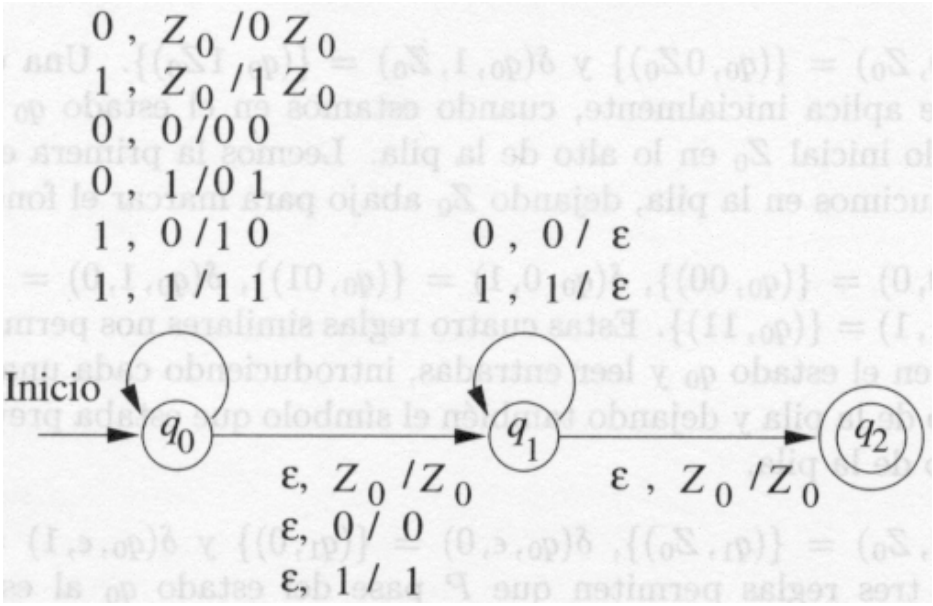
Definición formal de los AP

- $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$
  - Q: conjunto finito de estados
  - Σ: conjunto finito de símbolos de entrada
  - Γ: alfabeto de pila finito
  - δ: función de transición, $\delta(q, a, X) = (p, \gamma)$
  - q_0: estado inicial
  - Z_0: símbolo inicial de la pila
  - F: conjunto de estados de aceptación

Definición formal de los AP (II)

- Ejemplo: diseñar un AP que acepte el lenguaje L_{wwr}
  - $P = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_2\})$, donde δ se define:

Q	Σ	Γ	Movimiento
q_0	0	Z_0	$(q_0, 0Z_0)$
q_0	1	Z_0	$(q_0, 1Z_0)$
q_0	0	0	$(q_0, 00)$
q_0	0	1	$(q_0, 01)$
q_0	1	0	$(q_0, 10)$
q_0	1	1	$(q_0, 11)$
q_0	ε	Z_0	(q_1, Z_0)
q_0	ε	0	$(q_1, 0)$
q_0	ε	1	$(q_1, 1)$
q_1	0	0	(q_1, ε)
q_1	1	1	(q_1, ε)
q_1	ε	Z_0	(q_2, Z_0)

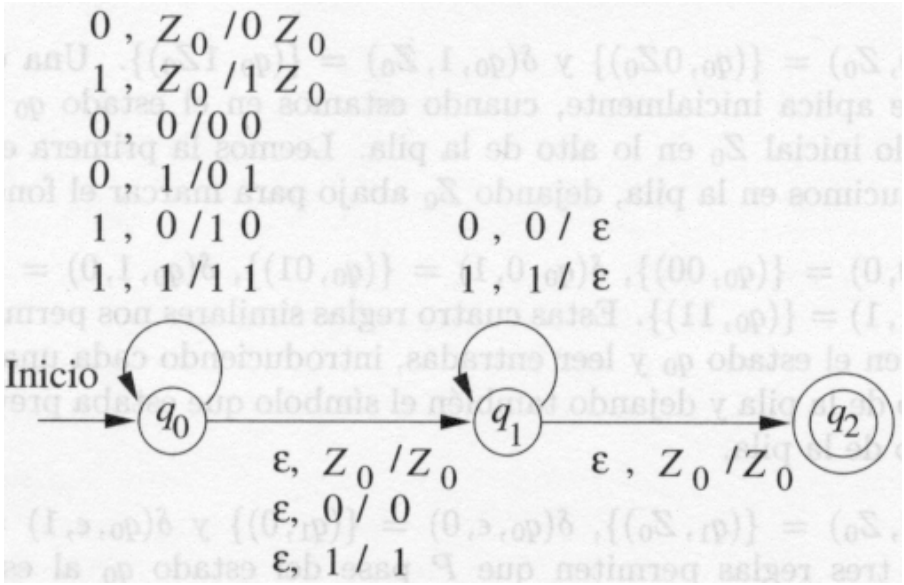


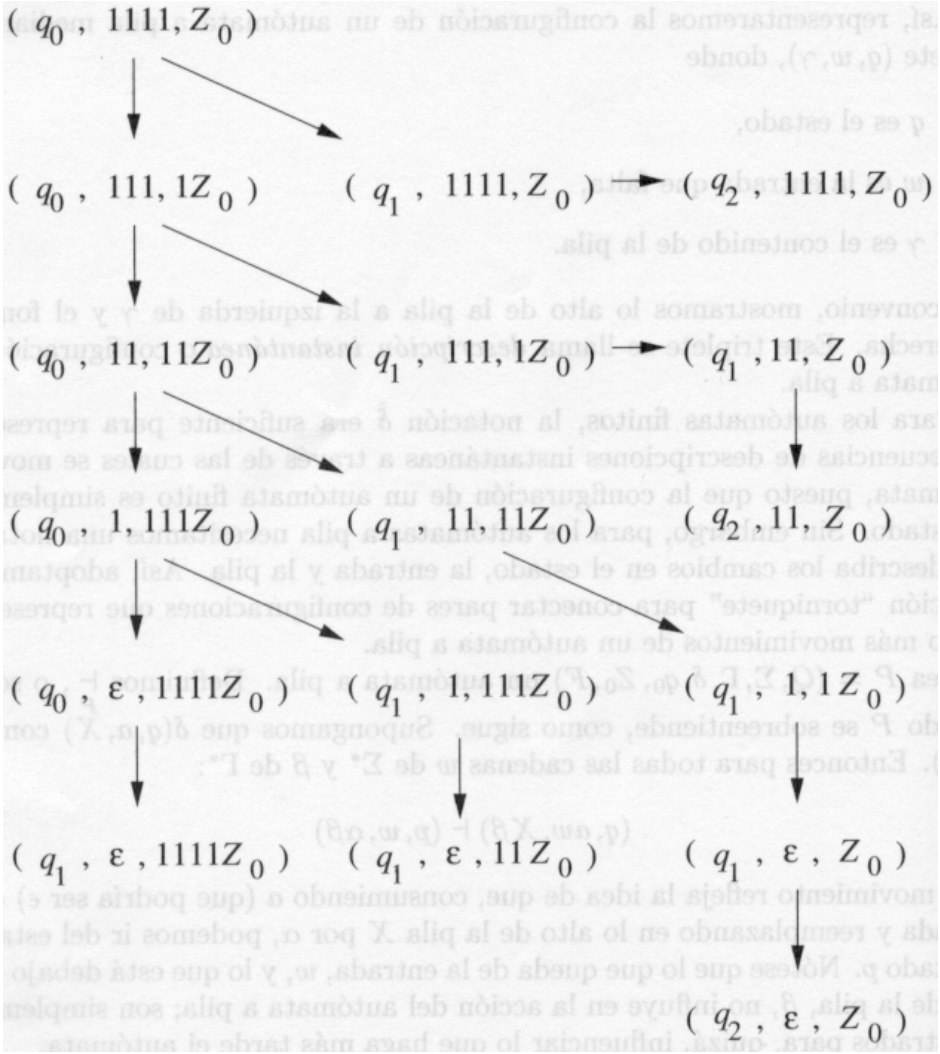
Descripción instantánea de un AP

- Descripción instantánea: (q, w, γ)
  - q es el estado
  - w es la entrada que falta por leer
  - γ es el contenido de la pila (la cima de la pila se muestra a la izquierda de γ, y el fondo a la derecha)
- Sea $\delta(q, a, X) = (p, \alpha)$. Para todas las cadenas w de Σ^* y β de Γ^*:
  - $(q, aw, X\beta) \vdash (p, w, \alpha\beta)$
  - cero o más movimientos del AP: $\vdash^*$
- Notación
  - alfabeto de entrada: *a, b, ...*
  - estados: *p, q, ...*
  - cadenas de símbolos de entrada: *..., w, x, y, z*
  - símbolos de pila: *..., X, Y, Z*
  - cadenas de símbolos de pila: $\alpha, \beta, \gamma, ...$

Descripción instantánea de un AP (II)

- Ejemplo: acción del AP para la entrada 1111





Descripción instantánea de un AP (III)

- Teorema:
  - si $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ es un AP y
  - $(q, x, \alpha) \vdash^* (p, y, \beta)$,
  - entonces para cualquier cadena w en Σ^* y γ en Γ^* es también cierto que $(q, xw, \alpha\gamma) \vdash^* (p, yw, \beta\gamma)$
- Teorema:
  - si $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ es un AP y
  - $(q, xw, \alpha) \vdash^* (p, yw, \beta)$,
  - entonces es también cierto que $(q, x, \alpha) \vdash^* (p, y, \beta)$

Problemas

1. Dado el AP $P = (\{q, p\}, \{0, 1\}, \{Z_0, X\}, \delta, q, Z_0, \{p\})$, donde δ se define en la siguiente tabla, mostrar las configuraciones alcanzables a partir de la inicial (q, w, Z_0), para w igual a 01, 0011 y 010

Q	Σ	Γ	Movimiento
q	0	Z_0	(q, XZ_0)
q	0	X	(q, XX)
q	1	X	(q, X)
q	ε	X	(p, ε)
p	ε	X	(p, ε)
p	1	X	(p, XX)
p	1	Z_0	(p, ε)

Soluciones:

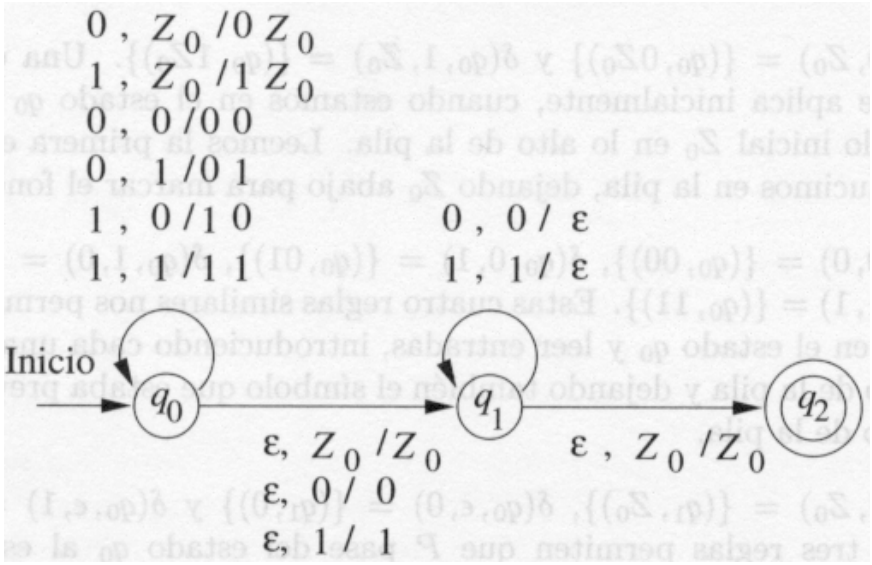
- $(q, 01, Z_0) \vdash^* (p, \varepsilon, Z_0) \vdash^* (p, \varepsilon, \varepsilon)$
- $(q, 0011, Z_0) \vdash^* (p, \varepsilon, XZ_0) \vdash^* (p, \varepsilon, XXZ_0)$
- $(q, 010, Z_0) \vdash^* (p, \varepsilon, XZ_0)$

Lenguajes aceptados por un AP

- Dos tipos de aceptación
  - por estado final
  - por vaciado de pila
- Ambos métodos son equivalentes
- Aceptación por estado final
  - sea $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ un AP
  - $L(P)$, el lenguaje aceptado por P por estado final es
  - $\{w \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, \alpha)\}$
  - para algún estado q de F y cualquier cadena de pila α

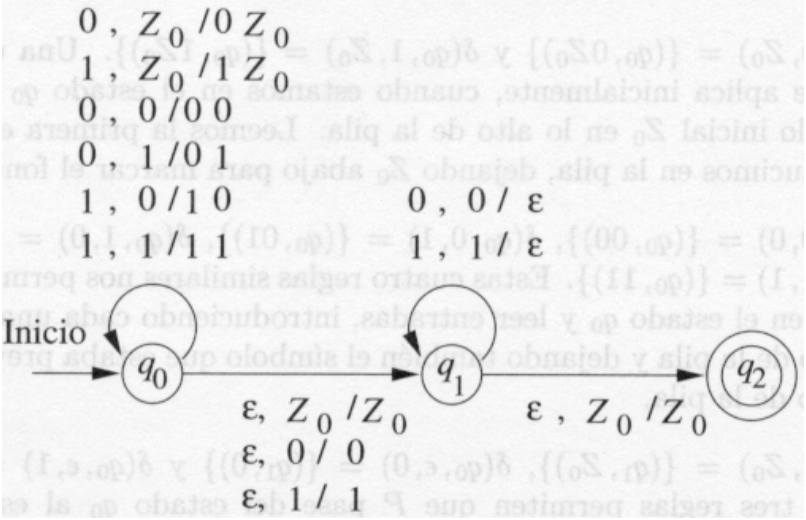
Lenguajes aceptados por un AP (II)

- Ejemplo: el AP de la figura acepta cadenas *x* por estado final si y sólo si *x* tiene la forma ww^R



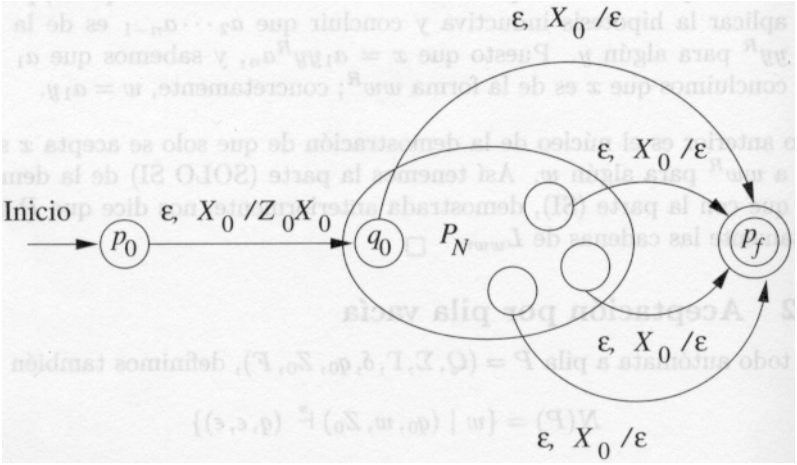
Aceptación por pila vacía

- Para todo AP $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ se define el lenguaje que acepta como:
  - $N(P) = \{w \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)\}$ para cualquier estado q
- Ejemplo: modificar el AP de la figura para que reconozca también por vaciado de pila
  - se cambia $\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}$ por $\delta(q_1, \varepsilon, Z_0) = \{(q_2, \varepsilon)\}$



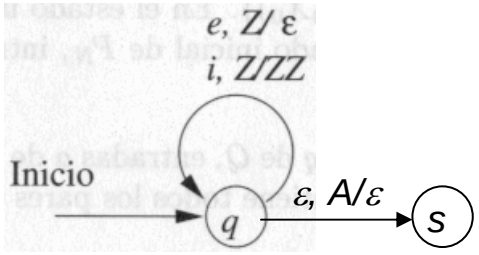
De pila vacía a estado final

- Teorema: si $L = N(P_N)$ para algún AP $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$, existe un AP P_F tal que $L = L(P_F)$
- Prueba: $P_F = (Q \cup \{p_0, p_F\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_F\})$, donde δ_F se define:
  1. $\delta_F(p_0, \varepsilon, X_0) = \{(q_0, Z_0X_0)\}$
  2. para todo estado q de Q, entrada a de Σ o $a = \varepsilon$, y símbolos de pila Y de Γ, $\delta_F(q, a, Y)$ contiene todos los pares de $\delta_N(q, a, Y)$
  3. además, $\delta_F(q, \varepsilon, X_0)$ contiene (p_F, ε) para todo estado q de Q

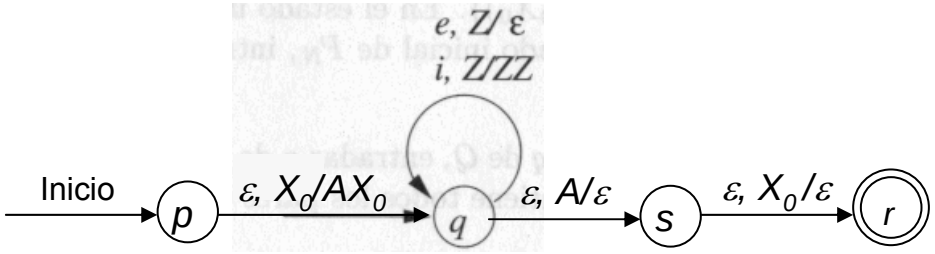


De pila vacía a estado final (II)

- Ejemplo: diseñar un AP que procese secuencias "*if else*", detectando cuando se introducen igual número de *else* que *if*
  - $P_N = (\{q, s\}, \{i, e\}, \{Z, A\}, \delta_N, q, A)$

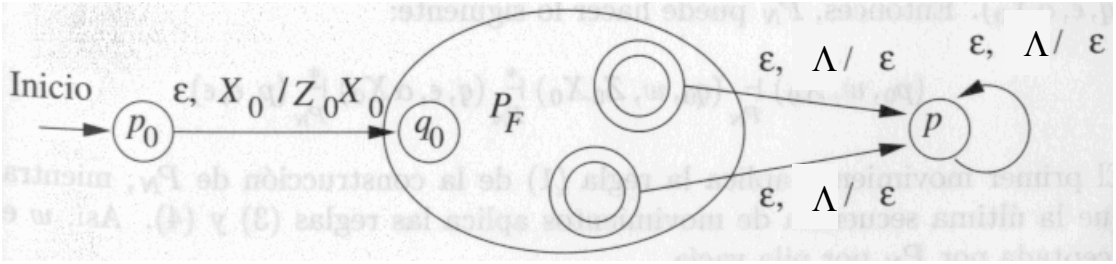


  - $P_F = (\{p, q, r, s\}, \{i, e\}, \{Z, A, X_0\}, \delta_F, p, X_0, \{r\})$



De estado final a pila vacía

- Teorema: sea L el lenguaje $L(P_F)$ de algún autómata a pila $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$. Entonces existe un AP P_N tal que $L = N(P_N)$
- Prueba: $P_N = (Q \cup \{p_0, p\}, \Sigma, \Lambda = \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$, donde δ_N se define:
  1. $\delta_N(p_0, \varepsilon, X_0) = \{(q_0, Z_0X_0)\}$
  2. para todo estado q de Q, entrada a de Σ o $a = \varepsilon$, y símbolos de pila Y en Γ, $\delta_N(q, a, Y)$ contiene todos los pares de $\delta_F(q, a, Y)$
  3. para todo estado de aceptación q en F, y símbolos de pila Y en Λ, $\delta_N(q, \varepsilon, Y)$ contiene (p, ε)
  4. para todos los símbolos de la pila Y en Λ, $\delta_N(p, \varepsilon, Y)=\{(p, \varepsilon)\}$



Problemas

1. Diseñar un AP que acepte el lenguaje $\{0^n1^n \mid n \geq 1\}$

2. Dado el AP $P = (\{q_0, q_1, q_2, q_3, f\}, \{a, b\}, \{Z_0, A, B\}, \delta, q_0, Z_0, \{f\})$, donde δ se define en la siguiente tabla
   1. mostrar la traza de ejecución que muestre que la cadena *bab* está en *L*
   2. mostrar la traza de ejecución que muestre que la cadena *abb* está en *L*

$\delta(q_0, a, Z_0) = (q_1, AAZ_0)$	$\delta(q_0, b, Z_0) = (q_2, BZ_0)$	$\delta(q_0, \varepsilon, Z_0) = (f, \varepsilon)$
$\delta(q_1, a, A) = (q_1, AAA)$	$\delta(q_1, b, A) = (q_1, \varepsilon)$	$\delta(q_1, \varepsilon, Z_0) = (q_0, Z_0)$
$\delta(q_2, a, B) = (q_3, \varepsilon)$	$\delta(q_2, b, B) = (q_2, BB)$	$\delta(q_2, \varepsilon, Z_0) = (q_0, Z_0)$
$\delta(q_3, \varepsilon, B) = (q_2, \varepsilon)$	$\delta(q_3, \varepsilon, Z_0) = (q_1, AZ_0)$	

Equivalencia entre AP y GIC

- El objetivo es demostrar que los tres siguientes lenguajes son todos de la misma clase
  1. lenguajes independientes del contexto
  2. lenguajes aceptados por estado final por algún AP
  3. lenguajes aceptados por vaciado de pila por algún AP

– (2) y (3) ya se ha demostrado
– demostraremos el paso de (1) a (3), aunque no de (3) a (1)

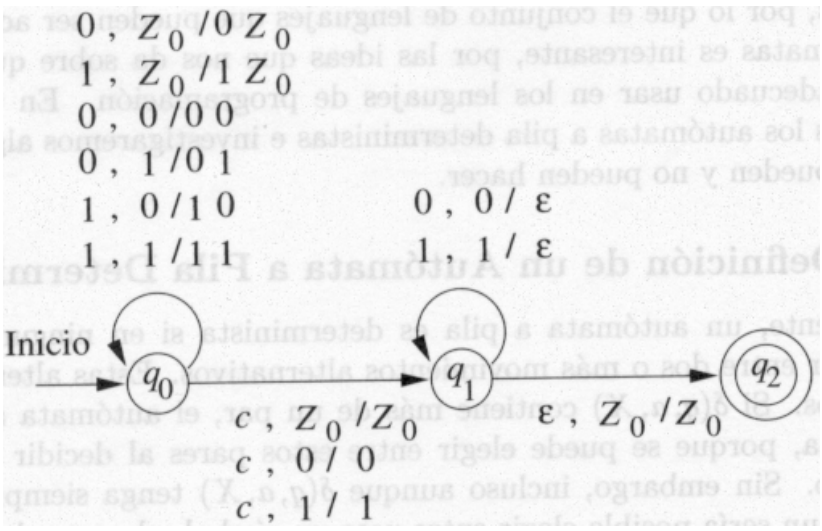
De gramáticas a AP

- Sea la GIC $G = (V, T, Q, S)$. El AP que acepta $L(G)$ por pila vacía será:
  - $P = (\{q\}, T, V \cup T, \delta, q, S)$
  - δ se define por:
    1. para cada variable A, $\delta(q, \varepsilon, A) = \{(q, \beta) \mid A \longrightarrow \beta$ es una producción de $P\}$
    2. para cada símbolo terminal a, $\delta(q, a, a) = (q, \varepsilon)$

- Ejemplo: obtener el AP que reconozca por vaciado la siguiente gramática: $I \longrightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$, $E \longrightarrow I \mid E*E \mid E+E \mid (E)$

Autómatas a pila deterministas

- Los APD aceptan un conjunto de lenguajes a medio camino entre los lenguajes regulares y las GIC
  - los analizadores sintácticos se comportan generalmente como APD
- Un AP $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ es determinista si:
  - $\delta(q, a, X)$ tiene como máximo un elemento para cualquier q en Q, a en Σ o $a = \varepsilon$, y X en Γ
  - si $\delta(q, a, X)$ no está vacío para algún a en Σ, $\delta(q, \varepsilon, X)$ debe estar vacío
- Ejemplo: sea $L_{wcwr} = \{wcw^R \mid w \text{ está en } (0 + 1)^*\}$. El APD que acepta dicho lenguaje es:



Problemas

1. Determinar si el siguiente AP es o no determinista: $P = (\{q, p\}, \{0, 1\}, \{Z_0, X\}, \delta, q_0, Z_0, \{p\})$, donde δ se define en la siguiente tabla

Q	Σ	Γ	Movimiento
q	0	Z_0	(q, XZ_0)
q	0	X	(q, XX)
q	1	X	(q, X)
q	ε	X	(p, ε)
p	ε	X	(p, ε)
p	1	X	(p, XX)
p	1	Z_0	(p, ε)

Formas normales para GIC

- Las gramáticas en formas normales pueden generar todos los LIC
  - Forma Normal de Chomsky
  - Forma Normal de Greibach
- Las gramáticas en formas normales reducen la complejidad para la obtención de las derivaciones
- Para obtener una gramática en forma normal es necesario realizar una serie de transformaciones que no modifican el lenguaje generado:
  - eliminación de producciones ε
  - eliminación de producciones unitarias (reglas de encadenamiento)
  - eliminación de símbolos inútiles

Eliminación de producciones ε

- Una variable es anulable si $A \Rightarrow^* \varepsilon$

- Algoritmo. Sea $G = (V, T, P, S)$ una GIC. Encontraremos todos los símbolos anulables de G mediante el siguiente algoritmo:
  - Base: si $A \rightarrow \varepsilon$ es una producción de G, A es anulable
  - Paso inductivo: si existe una producción $B \rightarrow C_1 C_2 \ldots C_k$ en la que las C_i son anulables, B es anulable

Eliminación de producciones ε (II)

- Construcción de una gramática sin producciones ε:
  - sea $G = (V, T, P, S)$ una GIC
  - se determinan todos los símbolos anulables de G
  - se construye la gramática $G_1 = (V, T, P_1, S)$ donde P_1 se determina como sigue:
    - para cada producción $A \longrightarrow X_1 X_2 \ldots X_k$ donde k >= 1, supongamos que m de los k símbolos son anulables
    - la nueva gramática tendrá 2^m versiones de esta producción (las X_i anulables estarán presentes o ausentes en todas las combinaciones posibles)
    - si, $m = k$ no se incluirá el caso de todas las X_i ausentes
    - además, las producciones de P de la forma $A \longrightarrow \varepsilon$ no estarán en P_1
    - si ε forma parte del lenguaje, se añade la producción $S \longrightarrow \varepsilon$
- Ejemplo: dada la gramática $S \rightarrow ABC$, $A \rightarrow aAA \mid \varepsilon$, $B \rightarrow bBC \mid \varepsilon$, $C \rightarrow bc \mid \varepsilon$, construir la gramática sin producciones ε
- Ejemplo: dada la gramática $S \rightarrow aS \mid AB \mid AC$, $A \rightarrow aA \mid \varepsilon$, $B \rightarrow bB \mid bS$, $C \rightarrow cC \mid \varepsilon$, construir la gramática sin producciones ε

Eliminación de producciones unitarias

- Producción unitaria: $A \rightarrow B$, donde A y B son variables
  - añaden pasos adicionales en las derivaciones
- Pares unitarios: pares de variables A y B tales que $A \Rightarrow^* B$, usando una secuencia que sólo hace uso de producciones unitarias
- Obtención de los pares unitarios (A, B)
  - Base: (A, A) es un par unitario para toda variable A
  - Paso inductivo: sea (A, B) un par unitario, y $B \rightarrow C$ una producción donde C es una variable. Entonces (A, C) es un par unitario

- Ejemplo: determinar los pares unitarios de la gramática

$$
\begin{array}{lcl}
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F & \rightarrow & I \mid (E) \\
T & \rightarrow & F \mid T * F \\
E & \rightarrow & T \mid E + T
\end{array}
$$

Eliminación de producciones unitarias (II)

- Eliminación de las producciones unitarias: dada la GIC $G = (V, T, P, S)$, se construye la GIC $G_1 = (V, T, P_1, S)$ como sigue:
  1. encontramos los pares unitarios de G
  2. para cada par unitario (A, B), añadimos a P_1 todas las producciones $A \rightarrow \alpha$ donde $B \rightarrow \alpha$ es una producción no unitaria de P

– Ejemplo: eliminar las producciones unitarias de la siguiente gramática

$$
\begin{array}{lcl}
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F & \rightarrow & I \mid (E) \\
T & \rightarrow & F \mid T * F \\
E & \rightarrow & T \mid E + T
\end{array}
$$

Eliminación de símbolos inútiles

- Un símbolo X es útil para una gramática $G = (V, T, P, S)$ si existe alguna derivación de la forma $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$, donde w está en T^*
- X es generador si $X \Rightarrow^* w$ (w está en T^*)
  - todo símbolo terminal es generador
- X es alcanzable si existe una derivación $S \Rightarrow^* \alpha X \beta$ para algún α y β
- Todo símbolo útil es
  - generador
  - alcanzable
- Eliminación de símbolos inútiles
  1. eliminamos los símbolos no generadores
  2. eliminamos los símbolos no alcanzables

Cálculo de los símbolos generadores

- Algoritmo: sea $G = (V, T, P, S)$ una gramática. Para calcular los símbolos generadores de G se lleva a cabo la siguiente inducción
  - Base: todo símbolo de T es generador
  - Paso inductivo: dada una producción $A \rightarrow \alpha$, donde todo símbolo de α es generador, entonces A es generador (se incluye el caso $\alpha = \varepsilon$)
- Ejemplo: dada la gramática $S \rightarrow AC \mid BS \mid B$, $A \rightarrow aA \mid aF$, $B \rightarrow CF \mid b$, $C \rightarrow cC \mid D$, $D \rightarrow aD \mid BD \mid C$, $E \rightarrow aA \mid BSA$, $F \rightarrow bB \mid b$, determinar los símbolos generadores

Cálculo de los símbolos alcanzables

- Algoritmo: sea $G = (V, T, P, S)$ una gramática. Para calcular los símbolos alcanzables de G se lleva a cabo la siguiente inducción
  - Base: S es alcanzable
  - Paso inductivo: dada una variable A alcanzable, para todas las producciones cuya cabeza es A, todos los símbolos de los cuerpos de dichas producciones serán alcanzables

- Ejemplo: dada la gramática $S \rightarrow AC \mid BS \mid B$, $A \rightarrow aA \mid aF$, $B \rightarrow CF \mid b$, $C \rightarrow cC \mid D$, $D \rightarrow aD \mid BD \mid C$, $E \rightarrow aA \mid BSA$, $F \rightarrow bB \mid b$, determinar los símbolos alcanzables

Resumen de las transformaciones

- Para convertir una GIC *G* en una GIC equivalente que no tiene símbolos inútiles, producciones ε, o producciones unitarias, es necesario realizar los siguientes pasos en el orden establecido:
  1. eliminar las producciones ε
  2. eliminar las producciones unitarias
  3. eliminar los símbolos inútiles

Forma Normal de Chomsky (FNC)

- Todo LIC no vacío tiene una gramática *G* en la que todas las producciones tienen una de las formas siguientes:
  - $A \rightarrow BC$, donde *A*, *B*, *C* son variables
  - $A \rightarrow a$, donde *A* es una variable y *a* un símbolo terminal
  - $S \rightarrow \varepsilon$
- Se dice que *G* está en FNC
  - Una cadena de longitud *n* se obtiene en $2n$-1 pasos
  - El árbol de derivación es binario, y su profundidad máxima de *n*
  - Se usa como algoritmo en el método CYK
- Ejemplos:
  - Gramática: $S \rightarrow AX_1$, $X_1 \rightarrow BX_2$, $X_2 \rightarrow CX_3$, $X_3 \rightarrow DE$, $A \rightarrow a$, $B \rightarrow b$, $C \rightarrow c$, $D \rightarrow d$, $E \rightarrow e$. Cadena: abcde
  - Gramática: $S \rightarrow SS \mid AB \mid CD$, $A \rightarrow a$, $B \rightarrow b$, $C \rightarrow c$, $D \rightarrow d$. Cadena: abcdab

FNC (II)

- Para transformar una gramática a FNC:
  - no puede tener producciones ε, producciones unitarias, ni símbolos inútiles
  - las producciones de esa gramática tienen la forma $S \rightarrow \varepsilon$, $A \rightarrow a$ (ya están en FNC), o bien un cuerpo de longitud dos o más. Habrá que:

    a) conseguir que en los cuerpos de longitud dos o más sólo aparezcan variables

    b) descomponer los cuerpos de longitud tres o más en una cascada de producciones en cuyos cuerpos sólo aparezcan dos variables

FNC (III)

- Construcción para (a):
  - para cada símbolo terminal *a* que aparece en un cuerpo de longitud dos o más, se crea una nueva variable *A*
  - esta variable sólo tendrá la producción $A \longrightarrow a$
  - se sustituyen las apariciones de *a* por *A* siempre que aparezca en un cuerpo de longitud mayor o igual que dos
- Construcción para (b):
  - se descomponen las producciones de la forma $A \longrightarrow B_1 B_2 \ldots B_k$ para $k >= 3$, en un grupo de producciones con dos variables en cada cuerpo
  - se introducen $k–2$ variables nuevas, $C_1, C_2, \ldots, C_{k-2}$
  - se reemplaza la producción original por las $k–1$ producciones $A \longrightarrow B_1 C_1$, $C_1 \longrightarrow B_2 C_2$, ..., $C_{k-3} \longrightarrow B_{k-2} C_{k-2}$, $C_{k-2} \longrightarrow B_{k-1} B_k$

FNC (IV)

- Ejemplo: convertir la gramática en FNC

$$
\begin{aligned}
E &\to E + T \mid T * F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
T &\to T * F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F &\to (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
I &\to a \mid b \mid Ia \mid Ib \mid I0 \mid I1
\end{aligned}
$$

$$
\begin{aligned}
E &\to EC_1 \mid TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
T &\to TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
F &\to LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
I &\to a \mid b \mid IA \mid IB \mid IZ \mid IO \\
A &\to a \\
B &\to b \\
Z &\to 0 \\
O &\to 1 \\
P &\to + \\
M &\to * \\
L &\to (\\
R &\to) \\
C_1 &\to PT \\
C_2 &\to MF \\
C_3 &\to ER
\end{aligned}
$$

Forma Normal de Greibach (FNG)

- Todo LIC no vacío es $L(G)$ para alguna gramática G cuyas producciones tienen la forma: $A \rightarrow a\alpha$, donde a es un símbolo terminal, y α una cadena de cero o más variables
- El uso de una producción introduce un símbolo terminal en una forma sentencial
  - una cadena de longitud n tiene una derivación de n pasos
  - Un analizador sintáctico descendente parará a profundidad n
  - Nunca habrá recursividad por la izquierda

Problemas

1. Encontrar una gramática equivalente a: $S \rightarrow AB \mid CA$, $A \rightarrow a$, $B \rightarrow BC \mid AB$, $C \rightarrow aB \mid b$, sin símbolos inútiles
2. Dada la gramática $S \rightarrow ASB \mid \varepsilon$, $A \rightarrow aAS \mid a$, $B \rightarrow SbS \mid A \mid bb$, obtener la gramática equivalente en FNC

Lema de bombeo para LIC

- Condición necesaria pero no suficiente
- Teorema: sea *L* un LIC. Entonces existe una constante *n* tal que si *z* es cualquier cadena de *L* de longitud |*z*| >= *n*, podemos escribir *z=uvwxy*, con las siguientes condiciones:

1. |*vwx*| <= *n*
2.
3. para todo *i* >= 0, uv^iwx^iy está en *L*

Aplicación del lema del bombeo

1. Elegimos un lenguaje *L* del que queremos demostrar que no es LIC
2. El valor de *n* es desconocido, por lo que debemos considerar cualquier posible valor
3. Elegimos *z* (podemos usar *n* como parámetro)
   - Si para un *n* dado no podemos escoger z, *L* será LIC
4. Se descompone *z* en *uvwxy*, sujeto a las restricciones:
   1. $|vx| \neq \varepsilon$
   2. $|vwx| \leq n$
5. El lenguaje no será LIC si siempre podemos elegir *k* y demostrar que uv^kwx^ky no está en *L*
   - Basta encontrar una descomposición de *z* en la que uv^kwx^ky esté en *L* para que el lenguaje sea LIC

Aplicación del lema del bombeo (II)

- Demostrar que los siguientes lenguajes no son LIC
  - $L=\{0^n1^n2^n \mid n >= 1\}$
    - un LIC no puede emparejar tres grupos de símbolos de acuerdo con su igualdad o desigualdad
  - $L=\{0^i1^j2^i3^j \mid i >= 1 \text{ y } j >= 1\}$
    - un LIC no puede emparejar dos pares de números iguales de símbolos que se entrelacen
  - $L=\{ww \mid w \text{ está en } \{0, 1\}^*\}$
    - un LIC no puede emparejar dos cadenas de longitud arbitraria si las cadenas se eligen de un alfabeto de más de un símbolo

Problemas

- Demostrar usando el lema de bombeo si los siguientes lenguajes son LIC
  1. $L=\{a^i b^j c^k \mid i < j < k\}$
  2. $L=\{0^n 1^n \mid n >= 1\}$
  3. El conjunto de palíndromos sobre el alfabeto {0, 1}

Problemas finales

1. ¿Es el lenguaje $L=\{a^i b^j c^k \mid i <= k \text{ o } j <= k\}$ independiente del contexto?
2. Obtener el autómata a pila que reconozca por **estado final** el lenguaje formado por los palíndromos sobre el alfabeto {a, b}. Restricción: el alfabeto de pila es {a, b, X}.
3. Dado el lenguaje $L = \{a^i b^j c^k \mid i=j \text{ o } j=k, i, j, k > 0\}$
   1. Determinar, aplicando el lema de bombeo, si el lenguaje es regular.
   2. Determinar, aplicando el lema de bombeo, si el lenguaje es independiente del contexto.
   3. Obtener el autómata a pila que reconozca por **estado final** dicho lenguaje.
4. Resolver las mismas cuestiones que en el problema anterior, pero para el lenguaje $L = \{a^i b^i a^j b^j \mid i, j > 0\}$