n.º 213
0,39 €

MINIMIZACIÓN DE AUTÓMATAS FINITOS

Teorema: Sea A un autómata finito determinista. Existe Â, afd equivalente a A, con un número mínimo de estados, único salvo isomorfismo.

Proceso en varias etapas:

A – Estados equivalentes.
B – Autómatas equivalentes.
C – Isomorfismo de A.F.
D – Autómata mínimo.

A – Equivalencia de estados

Definiciones: Sea $A = (\Sigma, Q, q_0, f, F)$ un autómata finito determinista.

1. Sean $p,q \in Q$. Se dice que **p y q son equivalentes**

$$p \, E \, q \equiv \forall x \in \Sigma^* \quad f(p,x) \in F \Leftrightarrow f(q,x) \in F$$

2. Sean $p,q \in Q$, $k \in N$. Se dice que **p y q son equivalentes en longitud k o k-equivalentes**

$$p \, E_k \, q \equiv \forall x \in \Sigma^* \ |x| \leq k \quad f(p,x) \in F \Leftrightarrow f(q,x) \in F$$

Notas:

1. p E q significa que los dos estados evolucionan de manera paralela, funcionan igual, hacen lo mismo en el siguiente sentido:

$f(p,x) \in F \Leftrightarrow f(q,x) \in F$ se puede escribir

$f(p,x) \in F \Rightarrow f(q,x) \in F$ y
$f(q,x) \in F \Rightarrow f(p,x) \in F$

y puesto que la proposición $p \Rightarrow q$ es lógicamente equivalente a $\neg q \Rightarrow \neg p$, se puede escribir

$f(p,x) \in F \Rightarrow f(q,x) \in F$ y
$f(p,x) \notin F \Rightarrow f(q,x) \notin F$

es decir, para cualquier palabra, si desde p llega a un estado final, también desde q se llega a estado final y si desde p llega a estado no final, también desde q llega a estado no final.

2. La definición 1 **no** proporciona un procedimiento para saber si dos estados $p,q \in Q$ son equivalentes, puesto que hay infinitas palabras x en Σ^*, aunque el alfabeto Σ sólo tenga un símbolo.

En cambio, para un determinado número natural k, si es posible, con la definición 2, saber si dos estados $p,q \in Q$ son k-equivalentes, puesto que hay un número finito de palabras $x \in \Sigma^*$ cuya longitud es $|x| \leq k$.

No obstante, utilizar la definición 2 para ello, parece, cuando menos, muy pesado o laborioso.

3. Obviamente las relaciones binarias E y E_k son relaciones de equivalencia sobre el conjunto Q y determinarán una partición de Q, el conjunto cociente, que denotaremos de la siguiente forma:

Q/E se denotará como P_E

Q/E_k se denotará como P_k

4. En particular para $k = 0$, se tiene la partición $Q/E_0 = P_0$ de la siguiente forma

$$p\,E_0\,q \equiv \forall x \in \Sigma^* \quad |x| \leq 0 \quad f(p,x) \in F \Leftrightarrow f(q,x) \in F$$

$$\Leftrightarrow f(p,\lambda) \in F \Leftrightarrow f(q,\lambda) \in F$$

λ es la única palabra $|x| \leq 0$

$$\Leftrightarrow (p \in F \Leftrightarrow q \in F)$$

definición de f, extensión de la función de transición a palabras

es decir, dos estados $p,q \in Q$ son 0-equivalentes, si los dos son finales, o los dos son no finales. Por tanto el conjunto cociente $Q/E_0 = P_0$ es

$$\boxed{P_0 = \{F, Q - F\}}$$

es decir, hay dos clases de 0-equivalencia.

5. De las definiciones 1 y 2 se deducen las siguientes propiedades inmediatas:

5.1 : $p\,E\,q \Rightarrow p\,E_k\,q \quad \forall k$

En particular $p\,E\,q \Rightarrow p\,E_0\,q \Rightarrow p,q \in F$ ó $p,q \notin F$

5.2 : $p\,E_k\,q \Rightarrow p\,E_r\,q \quad \forall r \leq k$

En particular $p\,E_k\,q \Rightarrow p\,E_0\,q \Rightarrow p,q \in F$ ó $p,q \notin F$

Propiedades:

1. $p \, E \, q \Rightarrow f(p,x) \, E \, f(q,x) \quad \forall x \in \Sigma^*$

i.e, si dos estados son equivalentes, entonces los correspondientes estados siguientes desde p y q con cualquier palabra x, también son equivalentes.

2. $p \, E_k \, q \not\Rightarrow f(p,x) \, E_k \, f(q,x) \quad \forall x \in \Sigma^*$

es decir, la propiedad análoga para la relación de k-equivalencia no se cumple.

3. $p \, E_{k+1} \, q \Leftrightarrow p \, E_k \, q \quad y \quad f(p,e) \, E_k \, f(q,e) \quad \forall e \in \Sigma$

4. $P_k = P_{k+1} \Rightarrow P_{k+i} = P_k \quad \forall i \geq 0$

5. $P_k = P_{k+1} \Rightarrow P_E = P_k$

6. $Sea \; |Q| = n \quad \exists j \leq n-2 \mid P_j = P_{j+1}$

Antes de pasar a demostrar cada una de estas propiedades veamos su significado (¿para qué sirven?).

- La propiedad 3 puesto que es una condición necesaria y suficiente, permite obtener la relación de k+1 equivalencia a partir de la k-equivalencia o, dicho de otra forma, la partición P_{k+1} a partir de la partición P_k.
- La propiedad 4 dice que si, al ir obteniendo cada partición de k-equivalencia a partir de la anterior, se repiten dos, entonces todas las siguientes son ya iguales y
- Propiedad 5: esa partición que se repite es la partición de equivalencia P_E
- La propiedad 6 afirma que esa repetición existe siempre y se produce en un número finito de pasos o iteraciones.

Este conjunto de propiedades es, por consiguiente, un procedimiento para obtener la relación de equivalencia E que, como se indicó, no es posible hacerlo basándose solamente en la definición.

Pero es que, además, este resultado es muy importante porque el conjunto cociente $Q/E = P_E$ va a ser el conjunto de estados del autómata mínimo $\hat{A}$ equivalente al autómata A, es decir, cada clase de equivalencia de la relación E será un estado del autómata $\hat{A}$.

En consecuencia, un **algoritmo para la obtención de la partición de equivalencia P_E** es el siguiente:

1 - $P_0 = \{F, Q - F\}$

2 - Se obtiene P_{k+1} a partir de P_k, utilizando la propiedad 3.

3 - $P_{k+1} = P_k \Rightarrow P_E = P_k$ fin

4 - $P_{k+1} \neq P_k \Rightarrow$ volver a 2

Demostración Propiedad 1:

Hay que probar $f(f(p,x); y) \in F \Leftrightarrow f(f(q,x); y) \in F \quad \forall x \ \forall y \in \Sigma^*$

Pero $f(f(p, x), y) = f(p,xy) \quad \forall x \ \forall y \in \Sigma^*$, según se probó en máquinas secuenciales para la función f, extensión a palabras de la función de transición, y un autómata finito es un caso particular de máquina secuencial de Moore.

Por tanto, hay que probar: $f(p,xy) \in F \Leftrightarrow f(q,xy) \in F \quad \forall x \ \forall y \in \Sigma^*$

La hipótesis de la que partimos es $p \, E \, q$, es decir $f(p,x) \in F \Leftrightarrow f(q,x) \in F \quad \forall x \in \Sigma^*$

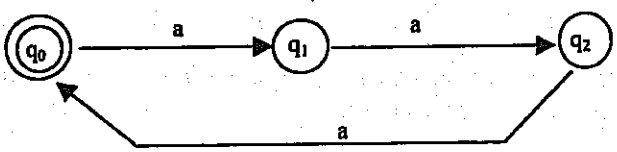
Luego, obviamente, se cumple lo que hay que probar.

Demostración Propiedad 2:

Bastará con encontrar un ejemplo, es decir, un autómata A y dos estados de él p y q, un número natural k y una palabra x tales que

$$pE_kq \quad y \quad f(p,x) \, E_k \, f(q,x)$$

Sea el autómata



$q_1 \, E_0 \, q_2$ pues $q_1 \notin F$ y $q_2 \notin F$

$f(q_1,a) = q_2 \notin F$

$f(q_2,a) = q_0 \in F$

$\Rightarrow \quad f(q_1,a) \not E_0 \, f(q_2,a)$

Demostración Propiedad 3:

$p\ E_{k+1}\ q \Leftrightarrow p\ E_k\ q$ y $f(p,e)\ E_k\ f(q,e)\ \forall e \in \Sigma$

$\Rightarrow$ ·) $p\ E_{k+1}\ q \Rightarrow p\ E_k\ q$ por definición de equivalencia en longitudes k y k+1

·) $f(p,e)\ E_k\ f(q,e) \quad \forall e \in \Sigma$

habrá que probar: $f(f(p,e),x) \in F \Leftrightarrow f(f(q,e),x) \in F \quad \forall e \in \Sigma,\ \forall x \in \Sigma^* \ |x| \leq k$

o lo que es lo mismo: $f(p,ex) \in F \Leftrightarrow f(q,ex) \in F \quad \forall e \in \Sigma,\ \forall x \in \Sigma^* \ |x| \leq k$

o expresado de otro modo $f(p,y) \in F \Leftrightarrow f(q,y) \in F \quad \forall y \in \Sigma^* \ 0 < |y| \leq k+1$

pero por hipótesis sabemos que $p\ E_{k+1}\ q$, es decir

$$p\ E_{k+1}\ q \Rightarrow f(p,x) \in F \Leftrightarrow f(q,x) \in F \quad \forall x \in \Sigma^* \ |x| \leq k+1$$

$\Leftarrow$ Hay que probar $f(p,x) \in F \Leftrightarrow f(q,x) \in F,\ \forall x \in \Sigma^* \ |x| \leq k+1$

Por hipótesis $p\ E_k\ q \Rightarrow f(p,x) \in F \Leftrightarrow f(q,x) \in F \quad \forall x \in \Sigma^* \ |x| \leq k$

falta por probar: $f(p,x) \in F \Leftrightarrow f(q,x) \in F \quad \forall x \in \Sigma^* \ |x| = k+1$

La segunda parte de la hipótesis es

$f(p,e)\ E_k\ f(q,e) \quad \forall e \in \Sigma$

$f(f(p,e),x) \in F \Leftrightarrow f(f(q,e),x) \in F \quad \forall e \in \Sigma,\ \forall x \in \Sigma^* \ |x| \leq k$

$f(p,ex) \in F \Leftrightarrow f(q,ex) \in F \quad \forall e \in \Sigma,\ \forall x \in \Sigma^* \ |x| \leq k$

que podemos expresar $f(p,y) \in F \Leftrightarrow f(q,y) \in F \quad \forall y \in \Sigma^* \ 0 < |y| \leq k+1$

en particular $f(p,y) \in F \Leftrightarrow f(q,y) \in F \quad \forall y \in \Sigma^* \ |y| = k+1$

que es lo que faltaba por probar.

Demostración Propiedad 4:

$P_k = P_{k+1} \Rightarrow P_{k+i} = P_k \quad \forall i \geq 0$

Se hará por inducción sobre i

•) Paso básico

$i = 0 \quad P_k = P_k$

(Para los valores $i = 1$ y $i = 2$ no es necesario probarlo, pero da "pistas" sobre cómo actuar en el paso de inducción)

$i = 1 \quad P_k = P_{k+1}$ por hipótesis

$i = 2$ hay que probar $P_k = P_{k+2}$, o lo que es lo mismo, $p\ E_k\ q \Leftrightarrow p\ E_{k+2}\ q \quad \forall p,q$

$\Leftarrow$ definición de equivalencia en longitud k y k+2

$\Rightarrow$ $p\ E_k\ q \Rightarrow p\ E_{k+1}\ q \Rightarrow f(p,e)\ E_k\ f(q,e)\ \forall e \Rightarrow f(p,e)\ E_{k+1}\ f(q,e)\ \forall e$, $p\ E_{k+1}\ q$ $\Rightarrow p\ E_{k+2}\ q$

($P_k = P_{k+1}$; prop 3; $P_k = P_{k+1}$; prop 3)

•) Paso de inducción:

Hipótesis de inducción: $P_k = P_{k+r}$

Hay que probar: $P_k = P_{k+r+1}$, o lo que es lo mismo, $p\ E_k\ q \Leftrightarrow p\ E_{k+r+1}\ q \quad \forall p,q$

$\Leftarrow$ definición de equivalencia en longitud k y k+r+1

$\Rightarrow$ $p\ E_k\ q \Rightarrow p\ E_{k+1}\ q \Rightarrow f(p,e)\ E_k\ f(q,e)\ \forall e \Rightarrow f(p,e)\ E_{k+r}\ f(q,e)\ \forall e$, $p\ E_{k+r}\ q$ $\Rightarrow p\ E_{k+r+1}\ q$

(hipótesis $P_k = P_{k+i}$; prop 3; hip inducción; prop 3)

Demostración Propiedad 5:

$\mathbf{P_k = P_{k+1} \Rightarrow P_E = P_k}$

Hay que probar $p\,E\,q \Leftrightarrow p\,E_k\,q$

$\Rightarrow$ definiciones de E y E_k

$\Leftarrow$ $p\,E_k\,q \Rightarrow p\,E_r\,q \quad \forall r < k$

$P_k = P_{k+1} \Rightarrow p\,E_{k+i}\,q \quad \forall i \geq 0$

$\Rightarrow p\,E_k\,q \ \forall k \Rightarrow p\,E\,q$

prop 4

def. de E y E_k

Demostración Propiedad 6:

Sea $|Q| = n \quad \exists j \leq n-2 \mid P_j = P_{j+1}$

•) $|Q| = 1$ Q sólo tiene un estado $Q = \{q\}$

$P_0 = P_1 = \ldots = P_E = \{Q\} = \{\{q\}\}$

•) $n \geq 2$

1er. caso.

$|P_0| = 1$ sólo hay una clase en $E_0 \Rightarrow P_0 = P_1 \Rightarrow j = 0$

$p\,E_1\,q \Rightarrow p\,E_0\,q$ (def. de k-equivalencia)

$|P_0| = 1 \Rightarrow p\,E_0\,q \ \forall p,q \Rightarrow f(p,e)\,E_0\,f(q,e) \ \forall e$

$p\,E_0\,q \Rightarrow p\,E_1\,q$

en particular

prop 3

2° caso. $|P_0| > 1$. Se hará por reducción al absurdo:

Supongamos $\forall j \quad 0 \leq j \leq n-2 \quad P_j \neq P_{j+1}$

$1 < |P_0| < |P_1| < |P_2| < \ldots |P_{n-1}| \Rightarrow |P_{n-1}| > n$!!!!!

$\Rightarrow \exists j \quad 0 \leq j \leq n-2 \quad P_j = P_{j+1}$

B – Autómatas finitos equivalentes

Definiciones: Sean $A_1 = (\Sigma, Q_1, q_{01}, f_1, F_1)$ y $A_2 = (\Sigma, Q_2, q_{02}, f_2, F_2)$

1. **$q_1 \in Q_1$ y $q_2 \in Q_2$ son equivalentes**

   $q_1 \, E \, q_2 \equiv \forall x \in \Sigma^* \quad f_1(q_1,x) \in F_1 \Leftrightarrow f_2(q_2,x) \in F_2$

2. **A_1 equivalente a A_2** $\quad A_1 \, E \, A_2 \equiv L(A_1) = L(A_2)$

   $\Leftrightarrow \quad \forall x \in \Sigma^* \quad f_1(q_{01},x) \in F_1 \Leftrightarrow f_2(q_{02},x) \in F_2$

   $\Leftrightarrow \quad q_{01} \, E \, q_{02}$ estados iniciales.

3. **Autómata suma** Sean A_1, A_2 tales que $Q_1 \cap Q_2 = \varnothing$

   $A_1 \oplus A_2 = (\Sigma, Q_1 \cup Q_2, q_0, f, F_1 \cup F_2)$ donde

   *) $f : (Q_1 \cup Q_2) \times \Sigma \rightarrow Q_1 \cup Q_2$ $\quad f(q,a) = \begin{cases} f_1(q,a) & \text{si } q \in Q_1 \\ f_2(q,a) & \text{si } q \in Q_2 \end{cases}$

   *) q_0 es uno cualquiera de $\{q_{01}, q_{02}\}$

   La tabla de transición del autómata suma, o el diagrama de transición, es simplemente el resultado de colocar juntas las dos tablas o de dibujar uno junto al otro los diagramas de transición.

Teorema: Sean A_1, A_2 tales que $Q_1 \cap Q_2 = \varnothing$

$$A_1 \, E \, A_2 \Leftrightarrow q_{01} \, E \, q_{02} \text{ en } A_1 \oplus A_2$$

es decir, para saber si dos autómatas A_1 y A_2 son equivalentes, se construye la partición P_E del autómata suma y se mira si los estados iniciales q_{01} y q_{02} están en la misma clase de equivalencia.

C – Autómatas finitos isomorfos

Definición: Sean A_1, A_2 con el mismo alfabeto.

A_1 y A_2 son **isomorfos**, $A_1 \simeq A_2 \equiv \exists\ i : Q_1 \to Q_2$ biyectiva tal que:

a) $i(q_{01}) = q_{02}$ Los estados iniciales son correspondientes.

b) $i[f_1(q,e)] = f_2[i(q),e] \qquad \forall q \in Q_1, \quad \forall e \in \Sigma$

c) $q \in F_1 \Leftrightarrow i(q) \in F_2 \qquad \forall q \in Q_1$ Los estados finales son correspondientes uno

Notas:

•) $A_1 \simeq A_2$ si uno puede convertirse en el otro renombrando estados.

•) $A_1 \simeq A_2 \Rightarrow A_1 \, E \, A_2$

$A_1 \simeq A_2 \not\Leftarrow A_1 \, E \, A_2$

Demostración de $\Rightarrow$:

$x \in L(A_1) \Leftrightarrow f_1(q_{01},x) \in F_1 \Leftrightarrow i[f_1(q_{01},x)] \in F_2 \Leftrightarrow f_2[i(q_{01}),x] \in F_2 \Leftrightarrow f_2(q_{02},x) \in F_2 \Leftrightarrow x \in L(A_2)$

c) $i[f_1(q,x)] = f_2[i(q),x] \quad \forall x \in \Sigma^*, \ \forall q$ a) a continuación

$i[f_1(q,x)] = f_2[i(q),x] \quad \forall q \ \forall x$: Inducción sobre $|x|$:

.) $x = \lambda \qquad i[f_1(q,\lambda)] = i(q)$

$f_2[i(q),\lambda] = i(q)$

.) Hipótesis de inducción: cierto para palabras de longitud n: $i[f_1(q,x)] = f_2[i(q),x] \quad \forall q, \ \forall x, \ |x| = n$

Sea y, $|y| = n+1 \quad y = xe, \ |x| = n$

Hay que probar $i[f_1(q,y)] = f_2[i(q),y]$

$i[f_1,(q,y)] = i[f_1(q,xe)] = i[f_1(f_1(q,x),e)] = f_2[i(f_1(q,x)),e] = f_2[f_2(i(q),x),e] = f_2[i(q),xe] = f_2[i(q),y]$

def de f_1 b) Hip. inducción, $|x| = n$ def de f_2

El paso de inducción también se puede hacer con $y = ex$:

$i[f_1(q,y)] = i[f_1(q,ex)] = i[f_1(f_1(q,e),x)] = f_2[i(f_1(q,e)),x] = f_2[f_2(i(q),e),x] = f_2[i(q),ex] = f_2[i(q),y]$

def de f_1 hip inducción, $|x| = n$ $\forall q$, en particular, $f_1(q,e)$ b) def de f_2

D – Autómata mínimo

Definición: **Autómata cociente:**

Sea $A = (\Sigma, Q, q_0, f, F)$ un afd. A partir de éste se construye otro:

$$\hat{A} = (\Sigma, Q/E, [q_0], \hat{f}, \hat{F}) \text{ donde}$$

$$\hat{f} : Q/E \times \Sigma \to Q/E \qquad \hat{f}([q],e) = [f(q,e)]$$

$$\hat{F} = \{ [q] \mid q \in F \}$$

Este autómata se llama **autómata cociente**.

Notas:

1) $\hat{f}$ está bien definida:

Puesto que $\hat{f}$ está definida a partir de representantes de una clase de equivalencia, habrá que comprobar que la imagen mediante $\hat{f}$ no depende del representante utilizado para obtener esa imagen. Es decir, habrá que probar:

$$[q] = [q'] \Rightarrow \hat{f}([q],e) = \hat{f}([q'],e) \quad \forall e \in \Sigma$$

En efecto:

$$[q] = [q'] \Rightarrow q \, E \, q' \Rightarrow f(q,x) \, E \, f(q',x) \quad \forall x \in \Sigma^* \Rightarrow f(q,e) \, E \, f(q',e) \quad \forall e \in \Sigma \Rightarrow$$

(propiedad 1; en particular)

$$\Rightarrow [f(q,e)] = [f(q',e)] \quad \forall e \in \Sigma \Rightarrow \hat{f}([q],e) = \hat{f}([q'],e) \quad \forall e \in \Sigma$$

2) $\hat{F}$ está bien definido:

En efecto, en cada clase de equivalencia del conjunto cociente Q/E, todos los estados son finales o todos los estados son no finales:

$$\left.\begin{array}{l} p \, E \, q \Rightarrow p \, E_k \, q \quad \forall k \Rightarrow p \, E_0 \, q \\ P_0 = \{F, Q-F\} \end{array}\right\} \Rightarrow \begin{array}{c} p \wedge q \in F \\ \text{ó} \\ p \wedge q \notin F \end{array}$$

Por fin !!! :

Teorema: **Sea A un autómata finito determinista. Existe $\hat{A}$, afd equivalente a A, con un número mínimo de estados, único salvo isomorfismo.**

Demostración: El autómata que estábamos buscando es $\hat{A}$, el autómata cociente. Hay que probar:

1. $\hat{A}$ es equivalente a A.
2. $\hat{A}$ es mínimo, es decir, si hay otro autómata A' equivalente a A, entonces A' tiene un número de estados mayor o igual que el autómata cociente.
3. $\hat{A}$ es único salvo isomorfismo, es decir, si hay otro autómata A' equivalente a A con el mismo número de estados que $\hat{A}$, entonces A' y $\hat{A}$ son isomorfos.

1. $L(\hat{A}) = L(A)$. En efecto

$$x \in L(\hat{A}) \Leftrightarrow \hat{f}([q_o],x) \in \hat{F} \Leftrightarrow [f(q_o,x)] \in \hat{F} \Leftrightarrow f(q_o,x) \in F \Leftrightarrow x \in L(A)$$

(def de $\hat{f}$) (def de $\hat{F}$)

(En rigor, además habría que probar $\hat{f}([q_o],x) = [f(q_o,x)]$) $\forall x \in \Sigma^*$, siendo $\hat{f}$ y f las correspondientes extensiones a palabras de las funciones de transición de los autómatas $\hat{A}$ y A.)

2. $\hat{A}$ es mínimo.

Bastará probar $A' \, E \, A \Rightarrow |Q'| \geq |Q/E|$

lo cual se consigue encontrando una aplicación del tipo

$$\varphi : Q' \rightarrow Q/E$$

que sea sobreyectiva.

Se define φ de la siguiente forma:

$$q' \in Q' \xrightarrow[A' \text{conexo}]{} \exists \, x \in \Sigma^* \mid f'(q'_o,x) = q'$$

$$f(q_o,x) = q \rightarrow \varphi(q') = [f(q_o,x)]$$

En particular, para obtener $\varphi(q'_o)$: $\varphi(q'_o) = [f(q_o,\lambda)] = [q_o]$

($f'(q'_o,\lambda) = q'_o$)

.) φ es sobreyectiva:

Sea $c \in Q/E$, ¿de qué elemento de Q' es imagen?

$c = [q] \quad \exists x \mid f(q_o,x) = q$

$f'(q'_o,x) = q' \Rightarrow \varphi(q') = c$

.) Pero además hay que probar que φ está bien definida, es decir, hay que probar:

$$\left.\begin{array}{l} f'(q'_o,x) = q' \\ f'(q'_o,y) = q' \end{array}\right\} \Rightarrow \quad [f(q_o,x)] = [f(q_o,y)]$$

es decir: $f(q_o,x)\ E\ f(q_o,y)$

Por tanto, hay que demostrar $f[f(q_o,x),z] \in F \Leftrightarrow f[f(q_o,y),z)] \in F \quad \forall z \in \Sigma^*$

$$f[f(q_o,x),z] \in F \Leftrightarrow f(q_o,xz) \in F \Leftrightarrow xz \in L(A) \Leftrightarrow xz \in L(A') \Leftrightarrow f'(q'_o,xz) \in F' \Leftrightarrow$$
$$A\ E\ A'$$

$$\Leftrightarrow f'[f'(q'_o,x),z] \in F' \Leftrightarrow f'[f'(q'_o,y),z] \in F' \Leftrightarrow f'(q'_o,yz) \in F' \Leftrightarrow yz \in L(A') \Leftrightarrow$$
$$f'(q'_o,x) = f'(q'_o,y)$$

$$\Leftrightarrow yz \in L(A) \Leftrightarrow f(q_o,yz) \in F \Leftrightarrow f[f(q_o,y),z] \in F$$
$$A'\ E\ A$$

3. Â es único salvo isomorfismo.

$$\left.\begin{array}{l} \text{Hay que probar:} \quad A'\ E\ A \\ \qquad\qquad\quad |Q'| = |Q/E| \end{array}\right\} \quad \Rightarrow \quad A' \text{ y } \hat{A} \text{ son isomorfos}$$

.) Sea $\varphi : Q' \rightarrow Q/E$ la del punto anterior

$$\left.\begin{array}{l} \text{ya se demostró que } \varphi \text{ es sobreyectiva} \\ |Q'| = |Q/E| \end{array}\right\} \quad \Rightarrow \quad \varphi \text{ es biyección}$$

.) φ es la función que define el isomorfismo:

a) $\varphi(q'_o) = [q_o]$

En efecto, pues $f'(q'_o,\lambda) = q'_o \Rightarrow \varphi(q'_o) = [f(q_o,\lambda)] = [q_o]$

b) $\varphi[f'(q',e)] = \hat{f}[\varphi(q'),e] \quad \forall q' \in Q' \quad \forall e \in \Sigma$

En efecto:

Sea $x \in \Sigma^* \mid f'(q'_o,x) = q' \Rightarrow \varphi(q') = [f(q_o,x)] \qquad (1)$

$$\varphi [f'(q',e)] = \varphi [f'(f'(q'_o,x),e)] = \varphi [f'(q'_o,xe)) = [f(q_o,xe)] = \hat{f}([q_o],xe) =$$

(def de φ; def de $\hat{f}$)

$$= \hat{f}(\hat{f}([q_o],x),e) = \hat{f}([f(q_o,x)]e) = \hat{f}(\varphi(q'),e)$$

(1)

c) $q' \in F' \Leftrightarrow \varphi(q') \in \hat{F}$

En efecto:

Sea $x \in \Sigma^* \mid f'(q'_o,x) = q' \quad \Rightarrow \quad \varphi(q') = [f(q_o,x)]$ (1)

$$q' \in F' \Leftrightarrow f'(q'_o,x) \in F' \Leftrightarrow x \in L(A') \Leftrightarrow x \in L(\hat{A}) \Leftrightarrow \hat{f}([q_o],x) \in \hat{F} \Leftrightarrow$$

($A' \equiv A$; $\hat{A} \equiv A$)

$$\Leftrightarrow [f(q_o,x)] \in \hat{F} \overset{(1)}{\Leftrightarrow} \varphi(q') \in \hat{F}$$

DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL

FACULTAD DE INFORMÁTICA
UNIVERSIDAD POLITÉCNICA DE MADRID

INFORMATICA TEÓRICA

Curso 2004-2005

Prácticas Tema 4

AUTÓMATAS FINITOS

Práctica 4.1: Construcción de autómatas finitos.

1.- Construir autómatas finitos deterministas que reconozcan los siguientes lenguajes:

$L_1 = \{ a^m b^n \mid m > 0, n > 0 \}$

$L_2 = \{ x \in \{0,1\}^* \mid$ en x aparece el 1 dos o tres veces, la primera y la segunda de las cuales no son consecutivas $\}$

$L_3 = \{ x \in \{a,b\}^* \mid Na(x)$ es par $\}$

$L_4 = \{ x \in \{a,b\}^* \mid x$ acaba en a $\}$

2.- Construir un AFD mínimo que reconozca las palabras sobre el alfabeto $\Sigma = \{a,b,c,d\}$ que contienen un número par (eventualmente cero) de apariciones de la subcadena bcd.
(Indicación: Se puede identificar las cadenas que son aceptadas y cómo se rompe la secuencia de estas cadenas con los posibles símbolos que se puedan procesar en cada momento.)

3.- Construir un AFD mínimo que reconozca el lenguaje sobre el alfabeto $\Sigma = \{0,1\}$ cuyas palabras verifican:

- si tiene menos de 5 unos, tiene que haber un número par de unos,
- si tiene 5 unos o más, tiene que haber un número impar de unos,
- cualquier palabra contiene al menos un uno.

4.- Sea el alfabeto $\Sigma = \{0,1\}$. Encontrar el AFD que reconoce el lenguaje:

$L = \{ \Sigma^* - \{\lambda\} \mid$ la subcadena 101 no aparezca $\}$

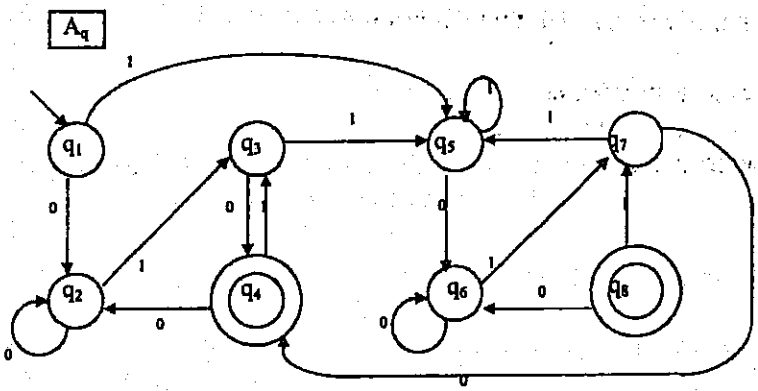
5.- Construir un AFD mínimo que reconozca el conjunto de los números positivos múltiplos de 3. (003 y 000 son válidos).

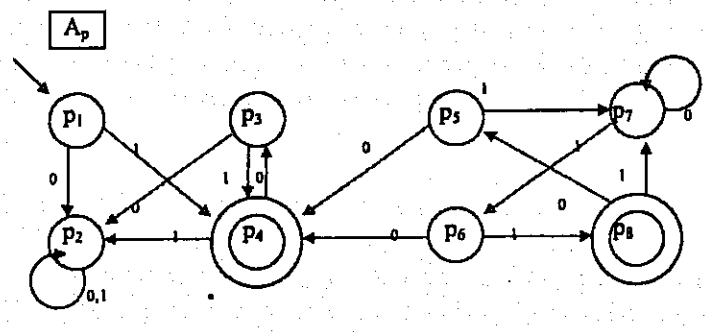
6.- Construir una autómata finito que reconozca el siguiente lenguaje:

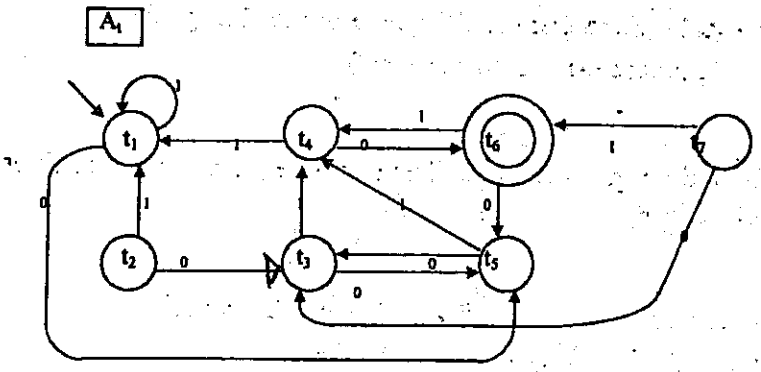
$$L = \{ x \in \{a,b\}^* \mid N_a(x) = 3 \text{ y } N_b(x) = 4 + 2 \}$$

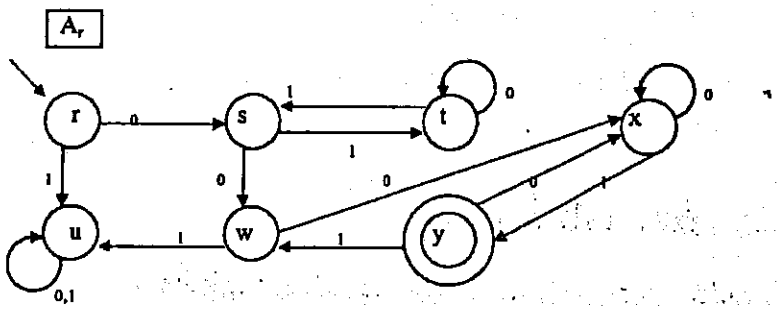
Práctica 4.2: Minimización de AF.

Dados los AF definidos por los siguientes diagramas de transición:









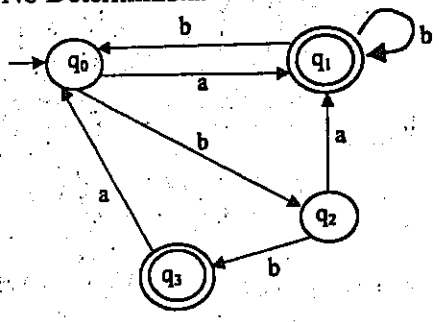
Obtener para cada uno de ellos el autómata mínimo.

Establecer si son o no equivalentes:

- por suma directa de autómatas.
- cuáles son isomorfos.

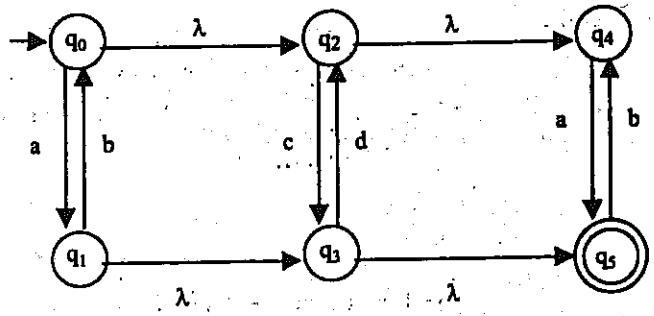
Práctica 4.3 : AFND ⇒ AFD

1. Dado el Autómata Finito No Determinista:

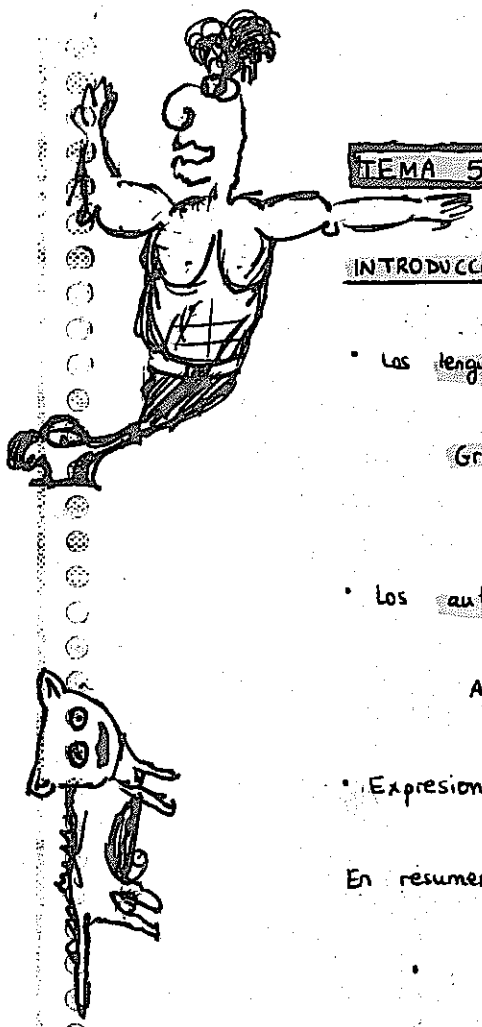


Calcular el autómata finito determinista equivalente.

2. Dado el Autómata Finito No Determinista:



Calcular el autómata finito determinista equivalente.



(12/12/05)

TEMA 5: LENGUAJES REGULARES

INTRODUCCIÓN

- Los lenguajes regulares son los lenguajes generados por las G_3.

Gramáticas Tipo 3 {
- GLI → Producciones de la forma $A ::= Va$
- GLD → " " " $A ::= aV$

A, V ∈ N
a ∈ T

- Los autómatas finitos aceptan los lenguajes regulares.

$$AF = \{\Sigma, Q, q_0, \delta, F\}$$

- Expresiones regulares → Por ejemplo, $(ab)^*$

En resumen:

- G_3 => Genera el lenguaje
- AF => Acepta " "
- ER => Expresa " "

} El mismo lenguaje

$$\boxed{L(AF) = L(G_3) = L(ER)}$$

Vamos a ir viendo esta igualdad por partes.

- Primero comprobaremos: $L(AF) = L(G_3)$

1.- $L(AF) \subset L(G_3)$ → Hay 2 algoritmos para esto.

$AF = \{\Sigma, Q, q_0, \delta, F\} \Rightarrow \exists\, G_3 / L(G_3) = L(AF)$
(Obtener una G_3 a partir de un AF)

2.- $L(G_3) \subset L(AF)$ → Hay 2 algoritmos.

$G_3 = \{\Sigma_T, \Sigma_N, P, S\} \Rightarrow \exists\, AF / L(AF) = L(G_3)$

De 1 y 2 se obtiene la igualdad $L(AF) = L(G_3)$.

- Ahora comprobaremos: $L(AF) = L(ER)$

3.- $L(AF) \subset L(ER)$ ⟶ Teorema de Análisis de Kleene (Ecuaciones características).

Obtener una ER a partir de un AF.

4.- $L(ER) \subset L(AF)$ ⟶ Teorema de Síntesis de Kleene (Algoritmo recursivo).

Obtener un AF a partir de una ER.

De 3 y 4 se obtiene la igualdad $L(AF) = L(ER)$.

En el próximo cuatrimestre

- Por último comprobaremos: $L(G_3) = L(ER)$

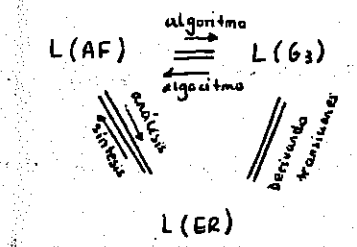
5.- $L(ER) \subset L(G_3)$ ⟶ Derivadas de una ER.

Obtener una G_3 a partir de una ER.

6.- $L(G_3) \subset L(ER)$

Obtener una ER a partir de una G_3

De 5 y 6 se obtiene la igualdad $L(G_3) = L(ER)$



(13/12/05)

1. CONSTRUIR UN AF A PARTIR DE UNA G_3 DADA. (GLI)

$L(G_3) \subseteq L(AF)$ $\quad G_3 = \{\Sigma_T, \Sigma_N, P, S\} \Rightarrow \exists\ AF \ / \ L(AF) = L(G_3)$

A partir de una GLI.

Las producciones de una GLI son de la forma. $P = \begin{cases} A ::= a \\ A ::= Va \\ S ::= \lambda \end{cases}$

$$S \longrightarrow A_1 a_n \longrightarrow A_2 a_{n-1} a_n \longrightarrow A_3 a_{n-2} a_{n-1} a_n \longrightarrow^* \dots \longrightarrow A_{n-2} a_3 \dots a_{n-1} a_n \longrightarrow$$

$A_{n-1} a_2 a_3 \dots a_{n-1} a_n \longrightarrow a_1 a_2 a_3 \dots a_{n-1} a_n$ $\Rightarrow$ Cada elemento no terminal se sustituye por una producción. Vamos haciendo sucesivas sustituciones hasta que obtenemos una palabra sin símbolos no terminales.

¡¡ LA GLI PROCESA LAS PALABRAS AL REVÉS QUE EL AF !!

GLI $\longrightarrow$ Se va generando hacia la izquierda. El primer elemento que se genera es a_n (el último).

AF $\longrightarrow$ Va leyendo los símbolos empezando por a_1 (el primero).

Para construir el AF a partir de la G_3 se hace lo siguiente:

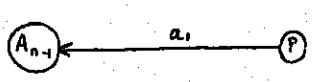
- **PARA EMPEZAR.** (Empezamos por el final para luego llegar al axioma y acab

Las producciones son del tipo: $A ::= a$ $\qquad$ $A ::= a \rightsquigarrow \delta(P, a) = A$

Añadimos un nuevo estado P para pasar a A_{n-1}. $Q = \Sigma_N \cup \{P\}$ $\quad P \in \Sigma_N$

$$A_{n-1} a_2 a_3 \dots a_{n-1} a_n \longrightarrow a_1 a_2 a_3 \dots a_{n-1} a_n$$

$$\downarrow A_{n-1} ::= a_1$$



· PARA INTERMEDIOS:

$$S \xrightarrow[①]{} A_1 a_n \xrightarrow[②]{} A_2 a_{n-1} a_n \xrightarrow[③]{} A_3 a_{n-2} a_{n-1} a_n \longrightarrow \cdots \longrightarrow A_{n-2} a_3 \ldots a_{n-1} a_n$$

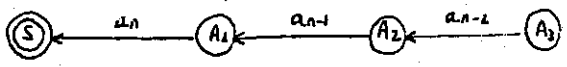
① $S ::= A_1 a_n$

② $A_1 ::= A_2 a_{n-1}$

③ $A_2 ::= A_3 a_{n-2}$

Las producciones son de la forma: $A ::= Ba$

$$\boxed{A ::= Ba \rightsquigarrow \delta(B, a) = A}$$



· PARA FINALIZAR:

Hemos alcanzado el axioma S. Debe ser estado final.

Si $\boxed{\lambda \in L}$ entonces el estado inicial $\boxed{P}$ debe ser final también.

$$F = \begin{cases} \{S\} & \text{si } S ::= \lambda \notin P \\ \{P, S\} & \text{si } S ::= \lambda \in P \end{cases}$$

Resumiendo

* Construir un AF a partir de una GLI dada.

$$G_3 = \{\Sigma_T, \Sigma_N, P, S\} \qquad AF = \{\Sigma, Q, q_0, \delta, F\}$$

$$\Sigma = \Sigma_T$$

$$Q = \Sigma_N \cup \{P\}$$

$$q_0 = P$$

δ = Movimientos del autómata: $\begin{cases} \text{Si } A ::= a \rightsquigarrow \delta(P, a) = A \\ \text{Si } A ::= Ba \rightsquigarrow \delta(B, a) = A \end{cases}$

$$F = \begin{cases} \{S\} & \text{si } S ::= \lambda \notin P \\ \{P, S\} & \text{si } S ::= \lambda \in P \end{cases}$$

X

EJ: Obtener el AF a partir de la gramática dada.

$G = \{\{a,b\}, \{S, A, B\}, S, P\}$ donde $P =$

$S ::= Ab \mid \lambda$

$A ::= Ba \mid a$ ← Es una GLI

$B ::= Ab$

$AF = \{\{a,b\}, \{P, S, A, B\}, P, f, \{P, S\}\}$

↑ P es final porque $\lambda \in L(G_3)$

Ahora voy cogiendo las producciones y transformándolas en movimientos del f

X

$S ::= Ab \rightarrow f(A, b) = S$

$A ::= Ba \rightarrow f(B, a) = A$

$A ::= a \rightarrow f(P, a) = A$

$B ::= Ab \rightarrow f(A, b) = B$

Expresamos los movimientos en forma de tabla:

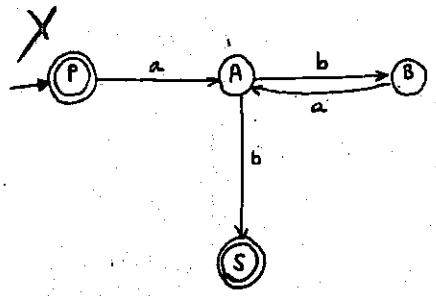
X

f	a	b
→ (P)	A	
(S)		
A		S, B
B	A	

OK!

Diagrama de estados:

X



○ = estado final
→ = estado inicial

$L(G) = L(AF) = (ab)^*$ $\lambda \in L$

abab $\in L$

SI LO PREGUNTAN EN UN EXAMEN, LO MEJOR ES HACER LAS TRES REPRESENTACIONES: DIAGRAMA DE ESTADOS, TABLA, MOVIMIENTOS (FUNCIÓN f).

EJ: Obtener el AF a partir de la gramática dada.

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

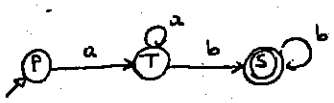
$AF = \{\{a,b\}, \{P, S, T\}, P, f, \{S\}\}$

$P =$

$S ::= Sb \mid Tb$

$T ::= Ta \mid a$

f	a	b
P	T	
S		S
T	T	S



① $S ::= Sb \rightarrow f(S, b) = S$

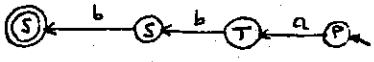
② $S ::= Tb \rightarrow f(T, b) = S$

③ $T ::= Ta \rightarrow f(T, a) = T$

④ $T ::= a \rightarrow f(P, a) = T$

$L = aa^* bb^+$

$x = abb \in L$: $S \xrightarrow{①} Sb \xrightarrow{②} Tbb \xrightarrow{④} abb$



(15/12/05)

2. CONSTRUIR UNA GLI PARTIENDO DE UN AF.

a) **Para los AFD.** AFD ⇝ GLI

$AF = \{\Sigma, Q, q_0, f, F\}$ $\quad G_3 = \{\Sigma_T, \Sigma_N, S, P\}$

- $\Sigma_T = \Sigma$
- $\Sigma_N = Q \cup \{S\}$ $\quad S \notin Q$ → Añadimos un nuevo elemento no terminal.
- S = axioma de G_3 (GLI)
- P
  - ① Si $q_0 \in F \rightarrow S ::= \lambda$
  - ② $f(q, a) = q' \rightarrow q ::= qa$ → Movimientos genéricos.
  - ③ $f(q_0, a) = q' \rightarrow q' ::= a$ → Caso particular del anterior.
  - ④ Si $q_k \in F \rightarrow S ::= q_k$ → Para cualquier estado que sea final añadimos dicha producción.

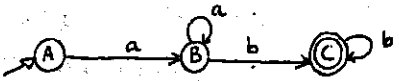
HAY QUE APLICAR A CADA PRODUCCIÓN TODAS LAS REGLAS QUE CUMPLA, NO SÓLO UNA.

b) **Para los AFND.** AFND ⇝ GLI

Además de lo de arriba:

- $f(q, a) = \{q_1, q_2 \dots q_j\} \rightarrow q_1 ::= qa$, $q_2 ::= qa$, …, $q_j ::= qa$ — Añado una nueva transición para cada estado al cual transita.
- $f(q, a) = \emptyset$ — Movimientos sin definir → No hago nada en la gramática.
- $f(q, \lambda) = q' \rightarrow q' ::= q$

EJ. Construir la GLI partiendo del AF dado.



$G_3 = \{\{a,b\}, \{A,B,C,S\}, S, P\}$

f	a	b
A	B	$\emptyset$
B	B	C
(C)	$\emptyset$	C

$f(A, a) = B$
$f(B, a) = B$
$f(B, b) = C$
$f(C, b) = C$

Es AFND

$S ::= \lambda$ → NO

- $B ::= Aa$
- $B ::= Ba$
- $C ::= Bb$
- $C ::= Cb$

③ $B ::= a$

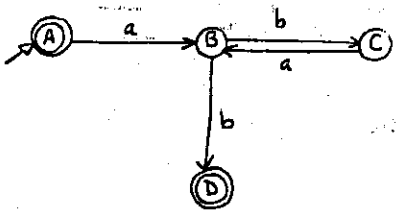
④ $S ::= C$

Paso ②

P:

$S ::= C$	$S ::= Bb \mid a$
$B ::= Aa \mid Ba \mid a$ →	$B ::= Aa \mid Ba \mid a$
$C ::= Bb \mid Cb$	

EJ:



Es un AFND

$\delta(A,a) = B$
$\delta(B,b) = C$
$\delta(B,b) = D$
$\delta(C,a) = B$

δ	a	b
→ A	B	∅
B	∅	C, D
C	B	∅
D	∅	∅

$G_3 = \{ \{a,b\}, \{A,B,C,D,S\}, S, P \}$

① $S ::= \lambda$

② $B ::= Aa$
$C ::= Bb$
$D ::= Bb$
$B ::= Ca$

③ $B ::= a$

④ $S ::= A$
$S ::= D$

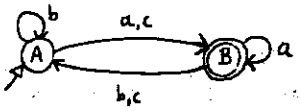
$S ::= A \mid D \mid \lambda$
$B ::= Aa \mid Ca \mid a$
$C ::= Bb$
$D ::= Bb$

⇓

$S ::= Bb \mid \lambda$
$B ::= Ca \mid a$
$C ::= Bb$

④ Quitamos las que contengan A porque A es un símbolo no generativo.

EJ:



$\delta(A,b) = A$
$\delta(A,a) = B$
$\delta(A,c) = B$
$\delta(B,a) = B$
$\delta(B,b) = A$
$\delta(B,c) = A$

δ	a	b	c
A	B	A	B
→ B	B	A	A

$ER = b^*(a+c)a^*[(b+c)b^*(a+c)a^*]^*$

$G_3 = \{ \{a,b,c\}, \{A,B,S\}, S, P \}$

① $S ::= \lambda$ → NO pertenece a P

② $A ::= Ab$
$B ::= Aa$
$B ::= Ac$
$B ::= Ba$
$A ::= Bb$
$A ::= Bc$

③ $A ::= b$
$B ::= a$
$B ::= c$

④ $S ::= B$

→

$S ::= B$
$A ::= Ab \mid Bb \mid Bc \mid b$
$B ::= Aa \mid Ac \mid Ba \mid a \mid c$

↳ cambio todas las B por S.

↓

$S ::= Aa \mid Ac \mid Sa \mid a \mid c$
$A ::= Ab \mid Sb \mid Sc \mid b$

3. CONSTRUIR UN AF PARTIENDO DE UNA GLD. GLD ⇝ AF

$\exists$ AF / $L(AF) = L(G_3)$ GLD

$G = \{\Sigma_T, \Sigma_N, P, S\}$

$\downarrow$

$AF = \{\Sigma, Q, q_0, f, F\}$ donde:

$\Sigma = \Sigma_T$

$Q = \Sigma_N \cup \{p\}$ $p \notin \Sigma_N$

$q_0 = S$ Axioma de G_3

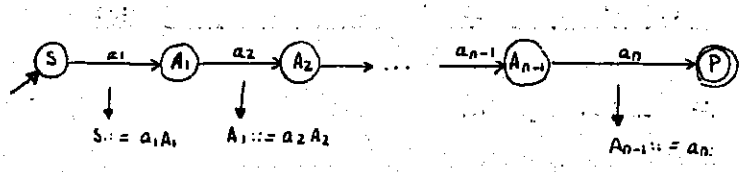
$$f = \begin{cases} A ::= aB \rightarrow f(A,a) = B \\ A ::= a \rightarrow f(A,a) = p \\ S ::= \lambda \rightarrow f(S,\lambda) = p \end{cases}$$

p

Si a un estado final!!

$F = \{p\}$

$S \rightarrow a_1A_1 \rightarrow a_1a_2A_2 \rightarrow \dots \rightarrow a_1a_2 \dots a_{n-1}A_{n-1} \rightarrow a_1a_2 \dots a_{n-1}a_n$



EJ: $G = \{\{0,1\}, \{S,A,B\}, S, P\}$ donde $P = \begin{cases} S ::= 1A \\ A ::= 1A \mid 0B \\ B ::= 0A \mid 1B \mid 1 \end{cases}$

A partir de G, construir el AF.

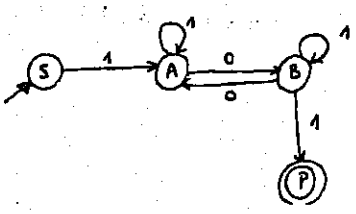
$AF = \{\{0,1\}, \{S,A,B,P\}, s, f, \{P\}\}$

f	0	1
→ S	∅	A
A	B	A
B	A	B,P
(P)	∅	∅

$f(S,1) = A$ $f(B,0) = A$

$f(A,1) = A$ $f(B,1) = B$

$f(A,0) = B$ $f(B,1) = P$



Autómata no determinista.

EJ: $G = \{a,b,c\}, \{S,A\}, S, P\}$ donde $P = \begin{cases} S ::= aA \mid bS \mid cA \mid a \mid c \\ A ::= aA \mid bS \mid cS \mid a \end{cases}$

A partir de esta GLD: obtener el AF.

$AF = \{\{a,b,c\}, \{S,A,P\}, S, f, \{P\}\}$

$f(S,a) = A$ $\quad f(A,a) = A$

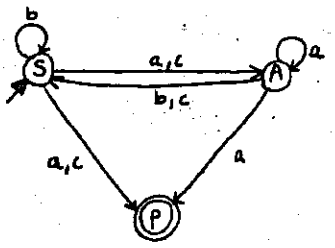
$f(S,b) = S$ $\quad f(A,b) = S$

$f(S,c) = A$ $\quad f(A,c) = S$

$f(S,a) = P$ $\quad f(A,a) = P$

$f(S,c) = P$

f	a	b	c
→ S	A,P	S	A,P
A	A,P	S	S
(P)	ϕ	ϕ	ϕ



Autómata finito no determinista.

4. CONSTRUIR UNA GLD A PARTIR DE UN AF DADO

$AF \longrightarrow G_3$ GLD

$\exists G_3 \;/\; L(G_3) = L(AF)$

$AF = \{\Sigma, Q, q_0, f, F\}$

GLD $G_3 = \{\Sigma_T, \Sigma_N, S, P\}$

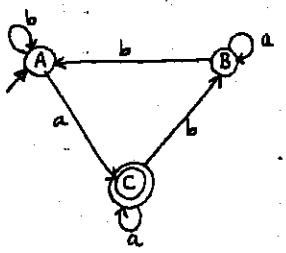
$\Sigma_T = \Sigma$

$\Sigma_N = Q$

$S = q_0$

$$P = \begin{cases} f(q,a) = p \longrightarrow q ::= ap \\ f(q,a) = p \xrightarrow{p \in F} q ::= a \\ q_0 \in F \longrightarrow q_0 ::= \lambda \end{cases}$$

EJ: AF de partida:



AFD

δ	a	b
→ A	C	A
B	B	A
©	C	B

$\delta(A,a) = C$
$\delta(A,b) = A$
$\delta(B,a) = B$
$\delta(B,b) = A$
$\delta(C,a) = C$
$\delta(C,b) = B$

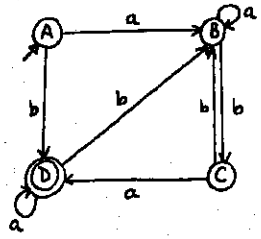
$AF = \{\{a,b\}, \{A,B,C\}, A, \delta, \{C\}\}$

(GLD) $G_3 = \{\{a,b\}, \{A,B,C\}, A, P\}$

$P =$ $A ::= aC$ | $A ::= bA$ | $B ::= aB$ | $B ::= bA$ | $C ::= aC$ | $C ::= bB$ | $A ::= a$ | $C ::= a$

$\longrightarrow$ $P =$ $A ::= aC \mid bA \mid a$ | $B ::= aB \mid bA$ | $C ::= aC \mid bB \mid a$

EJ:



δ	a	b
A	B	D
B	B	C
C	D	B
D	D	B

$\delta(A,a) = B$ $\quad \delta(C,a) = D$
$\delta(A,b) = D$ $\quad \delta(C,b) = B$
$\delta(B,a) = B$ $\quad \delta(D,a) = D$
$\delta(B,b) = C$ $\quad \delta(D,b) = B$

(GLD) $G_3 = \{\{a,b\}, \{A,B,C,D\}, A, P\}$

$A ::= aB$ | $A ::= bD$ | $B ::= aB$ | $B ::= bC$ | $C ::= aD$ | $C ::= bB$ | $D ::= aD$ | $D ::= bB$ | $A ::= b$ | $C ::= a$ | $D ::= a$

$\longrightarrow$ $A ::= aB \mid bD \mid b$ | $B ::= aB \mid bC$ | $C ::= aD \mid bB \mid a$ | $D ::= aD \mid bB \mid a$

↓ C = D : Quito uno

$\longrightarrow$ $A ::= aB \mid bD \mid b$ | $B ::= aB \mid bD$ | $D ::= aD \mid bB \mid a$ $= P$

EJ:

δ	0	1
→ A	B	C
B	B	D
©	A	E
D	D	D
Ⓔ	A	E

$\delta(A,0) = B$ $\quad \delta(D,0) = D$ $\quad$ AF ⇝ GLD

$\delta(A,1) = C$ $\quad \delta(D,1) = D$

$\delta(B,0) = B$ $\quad \delta(E,0) = A$

$\delta(B,1) = D$ $\quad \delta(E,1) = E$

$\delta(C,0) = A$

$\delta(C,1) = E$

(GLD) $G_3 = \{ \langle 0,1 \rangle, \{A,B,C,D,E\}, A, \mathcal{P} \}$

$A ::= 0B$
$A ::= 1C$
$B ::= 0B$
$B ::= 1D$
$C ::= 0A$
$C ::= 1E$
$D ::= 0D$
$D ::= 1D$
$E ::= 0A$
$E ::= 1E$

$A ::= 1$
$C ::= 1$
$E ::= 1$

→

$A ::= 0B \mid 1C \mid 1$
$B ::= 0B \mid 1D$
$C ::= 0A \mid 1E \mid 1$
$D ::= 0D \mid 1D$
$E ::= 0A \mid 1E \mid 1$

↓

C = E

Quito uno

→

$A ::= 0B \mid 1E \mid 1$
$B ::= 0B \mid 1D$
$D ::= 0D \mid 1D$
$E ::= 0A \mid 1E \mid 1$

5. EQUIVALENCIA ENTRE AF Y ER

TEOREMAS DE KLEENE:

a) SÍNTESIS: También conocido como algoritmo recursivo.
Permite obtener un AF a partir de una ER.

b) ANÁLISIS: También conocido como ecuaciones características.
Permite obtener una ER a partir de un AF.

a) TEOREMA DE SÍNTESIS DE KLEENE.

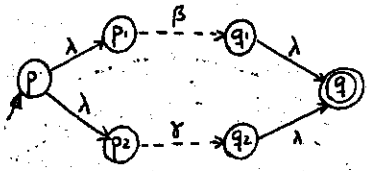
Este algoritmo permite construir, de forma recursiva, un autómata no determinista con un estado inicial y un solo estado final distinto del inicial, que acepta el lenguaje descrito por una expresión regular determinada α.

El algoritmo consta de 6 pasos:

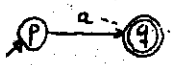
ER ↓ — AUTÓMATA correspondiente ↓

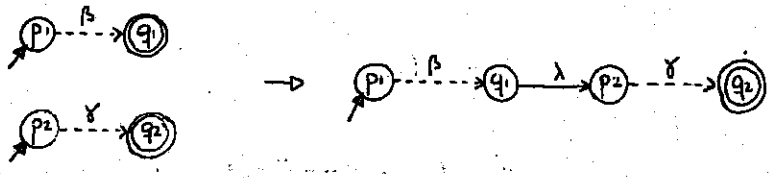
1) $\alpha = \emptyset$ 

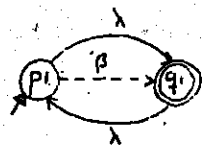
2) $\alpha = \lambda$ 

3) $\alpha = \beta + \gamma$ 

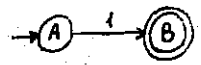
siendo
- $p_1 \dashrightarrow q_1$ el AF correspondiente a β.
- $p_2 \dashrightarrow q_2$ el AF de γ.

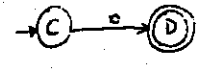
4) $\alpha = a \quad a \in \Sigma$ 

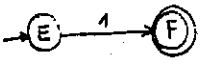
5) $\alpha = \beta \cdot \gamma$ 

6) $\alpha = \beta^*$ 

EJ: $ER = 1(0+1)^*$

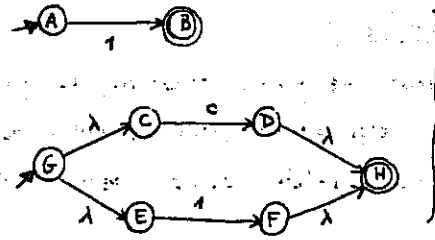
$ER = 1$ 

$ER = 0$ 

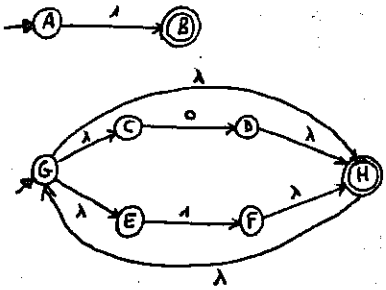
$ER = 1$ 

} 1, 0, 1

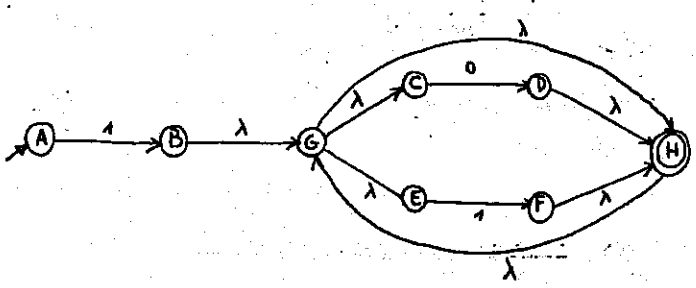
→ SIGUE



1, 0+1

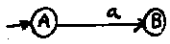


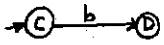
1,(0+1)*

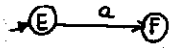


1(0+1)*

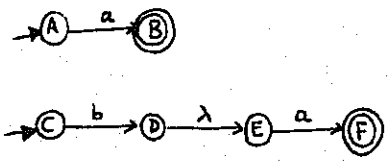
EJ: ER = a(ba)*

ER = a 

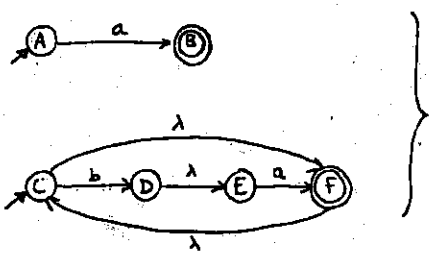
ER = b 

ER = a 

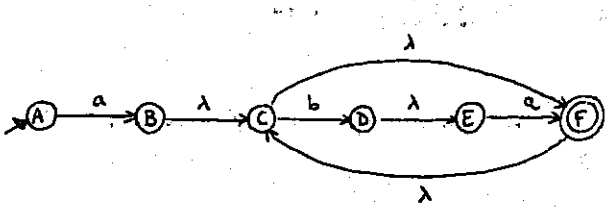
a, b, a



a, ba



$a,(ba)^+$



$a(ba)^+$

b) TEOREMA DE ANÁLISIS DE KLEENE.

Nos dice que todo lenguaje que es representado por un AF puede representarse también por una ER.

Dado un autómata A, hay que encontrar la ER que representa a L(A). Para ello utilizaremos las <u>ECUACIONES CARACTERÍSTICAS</u>.

$$AF = \{\Sigma, Q, q_0, f, F\} \Rightarrow \exists \alpha \text{ en ER } / L(\alpha) = L(AF)$$

* <u>DEF.</u>- Si AF tiene $Q = \{q_1, q_2, \dots, q_n\}$: $q_i \longrightarrow x_i = \bigcup_{j=0}^{n} A_{ij}\, x_j + C_i$

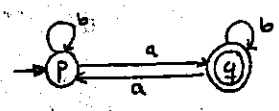
x_i = Variable correspondiente al estado q_i

$A_{ij} = \{a \in \Sigma / q_j \in f(q_i, a)\}$ Arco etiqueta entre q_i y q_j

x_j = Variable correspondiente al estado q_j

$$C_i = \begin{cases} \lambda & \text{si } q_i \in F \\ \emptyset & \text{si } q_i \notin F \end{cases} \qquad \forall q \in Q,\ q_i \in f(q, \lambda)$$

<u>EJ:</u>



Sus ecuaciones características serán:

$$x_0 = b.x_0 + a.x_1$$

$$x_1 = a.x_0 + b.x_1 + \lambda$$

- SIMPLIFICACIONES:

  - Estados trampa
  - Estados inaccesibles

  } $x_i = \emptyset$

- LEMA DE ARDEN:

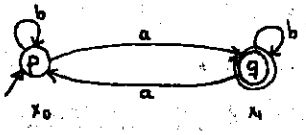
$X = AX + B \xrightarrow{\text{solución:}} X = A^* B$

$\lambda \notin A$
$\lambda \notin B$ } La solución es única si se dan estas dos condiciones.

Tendremos que partir de un AFD o AFND sin λ-transiciones para poder aplicar las ecuaciones características.

$L(AF) = ER_1$
$L(AF) = ER_2$ $\Longrightarrow$ ER_1 y ER_2 equivalentes.

- EJ:



δ	a	b
→ p	q	p
(q)	p	q

$\delta(p, a) = q$
$\delta(p, b) = p$
$\delta(q, a) = p$
$\delta(q, b) = q$

$$X_i = \bigcup_{j=0}^{n} A_{ij} x_j + C_i$$

Las ecuaciones características son:

$$\begin{cases} x_0 = bx_0 + ax_1 \\ x_1 = bx_1 + ax_0 + \lambda \end{cases}$$

X AX B

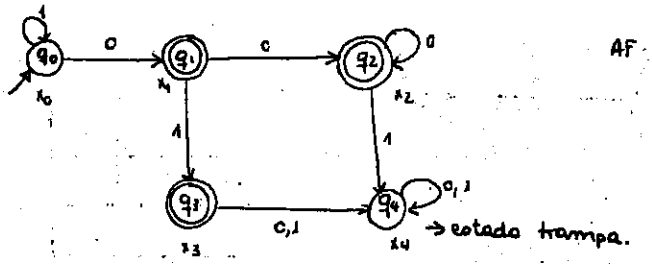
$x_0 = b^*(ax_1) = b^*a x_1$
$x_1 = b^*(ax_0 + \lambda) = b^*a x_0 + b^*$
$x_0 = b^*a(b^*a x_0 + b^*)$
$x_0 = (b^*ab^*a)x_0 + b^*ab^*$
$\boxed{x_0 = (b^*ab^*a)^* b^*ab^*}$

ER_2

$x_1 = b^*(ax_0 + \lambda) = b^*a x_0 + b^*$
$x_0 = bx_0 + ax_1 \Longrightarrow$
$x_0 = bx_0 + ab^*a x_0 + ab^*$
$x_0 = (b + ab^*a)x_0 + ab^*$
$\boxed{x_0 = (b + ab^*a)^* ab^*}$

ER_1

EJ:



AF ⟶ ER

δ	0	1
→ q_0	q_1	q_0
(F) q_1	q_2	q_3
(F) q_2	q_2	q_4
(F) q_3	q_4	q_4
q_4	q_4	q_4

$\delta(q_0, 0) = q_1$ $\quad$ $\delta(q_3, 0) = q_4$

$\delta(q_0, 1) = q_0$ $\quad$ $\delta(q_3, 1) = q_4$

$\delta(q_1, 0) = q_2$ $\quad$ $\delta(q_4, 0) = q_4$

$\delta(q_1, 1) = q_3$ $\quad$ $\delta(q_4, 1) = q_4$

$\delta(q_2, 0) = q_2$

$\delta(q_2, 1) = q_4$

$$x_0 = 1x_0 + 0x_1$$
$$x_1 = 1x_3 + 0x_2 + \lambda$$
$$x_2 = 1x_4 + 0x_2 + \lambda$$
$$x_3 = 1x_4 + 0x_4 + \lambda$$
$$x_4 = \emptyset$$

$$x_0 = 0x_1 + 1x_0$$
$$x_1 = 0x_2 + 1x_3 + \lambda$$
$$x_2 = 0x_2 + \lambda$$
$$x_3 = \lambda$$

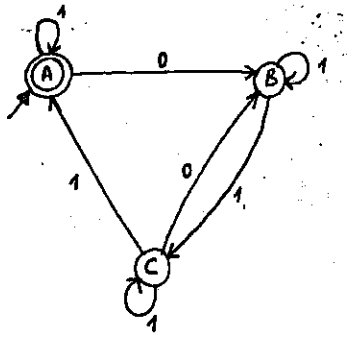
$x_2 = 0^* \cdot \lambda = 0^*$

$x_1 = \overbrace{0.0^*}^{0^*} + 1 + \lambda = 0^* + 1$

$x_0 = 0(0^* + 1) + 1x_0$; $x_0 = 00^* + 01 + 1x_0$;

$$\boxed{x_0 = 1^*(00^* + 01)}$$

Ej:



δ	0	1
→ (A)	B	A
B		B, C
C	B	A, C

ER

$$\begin{cases} x_0 = 1x_0 + 0x_1 + \lambda \\ x_1 = 1x_1 + 1x_2 \\ x_2 = 1x_0 + 1x_2 + 0x_1 \end{cases}$$

; Empiezo por x_2 y voy sustituyendo en x_1, x_0... De abajo a arriba

$x_2 = 1^*(1x_0 + 0x_1) = 1^*1x_0 + 1^*0x_1$;

$x_1 = 1x_1 + 1(1^*1x_0 + 1^*0x_1)$; $x_1 = 1x_1 + 11^*1x_0 + 1^*0x_1$;

$x_1 = (1 + 11^*0)x_1 + 11^*1x_0$

$x_1 = (1 + 11^*0)^* 11^*1x_0$;

$x_0 = 1x_0 + 0\left((1 + 11^*0)^* 11^*1x_0\right) + \lambda$;

$x_0 = (1 + 0(1 + 11^*0)^* 11^*1)x_0 + \lambda$;

$$\boxed{x_0 = 1 + 0(1 + 11^*0)^* 11^*1)^*}$$

DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL

FACULTAD DE INFORMÁTICA

UNIVERSIDAD POLITÉCNICA DE MADRID

INFORMÁTICA TEÓRICA

Curso 2004-2005

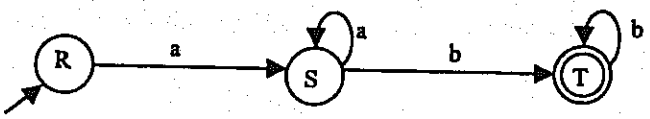
Prácticas Tema 5

LENGUAJES REGULARES

Práctica 5.1: Gramáticas lineales izquierdas y Autómatas Finitos

1. a) Sea A un autómata finito determinista, $A = (\Sigma, Q, q_0, f, F)$. Construir un algoritmo para obtener directamente una gramática lineal izquierda G tal que $L(G) = L(A)$.

b) Sea el autómata finito de la figura que acepta el lenguaje aa*bb*. Encontrar una gramática lineal izquierda G tal que $L(G) = L(A)$.



2. a) Dada una gramática lineal izquierda cualquiera $G = (\Sigma_T, \Sigma_N, S, P)$, obtener un autómata finito A tal que $L(A) = L(G)$.

b) Dada la gramática $G = (\{a, b\}, \{S, T\}, S, P)$ donde las producciones son:

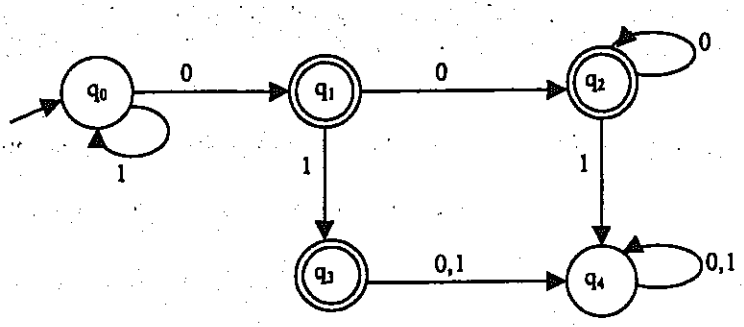
$$S := Sb \mid Tb$$

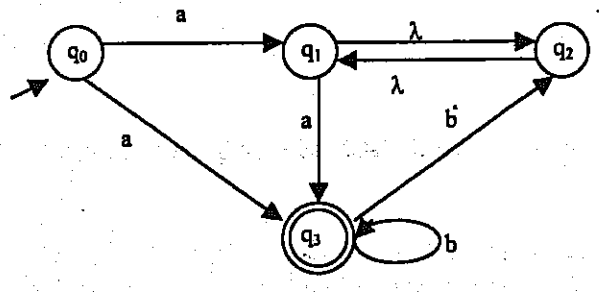
$$T := Ta \mid a$$

Obtener directamente un autómata finito A tal que $L(A) = L(G)$.

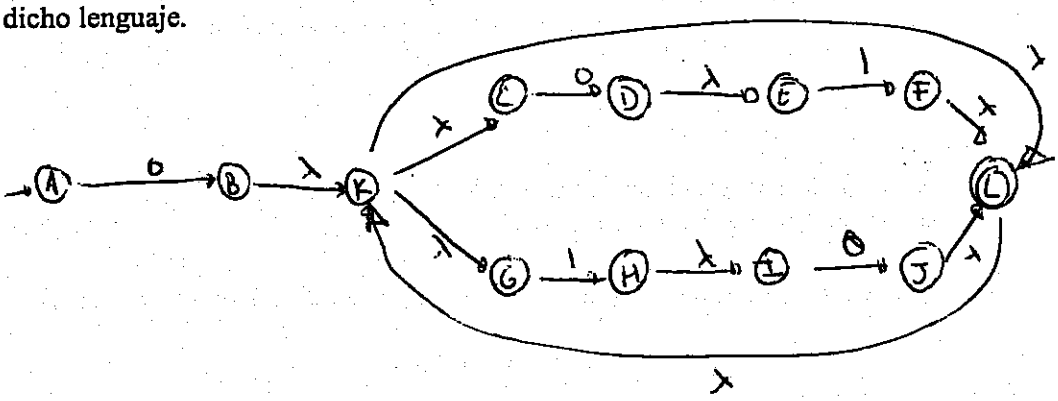
Práctica 5.2: Autómatas Finitos y Expresiones Regulares

1.- Dados los autómatas finitos siguientes, obtener el lenguaje que aceptan resolviendo el correspondiente sistema de ecuaciones características.





2.- Dado el lenguaje representado por la expresión regular $R = 0(01 + 10)^*$, obtener, utilizando el teorema de síntesis, un autómata finito mínimo determinista que reconozca dicho lenguaje.



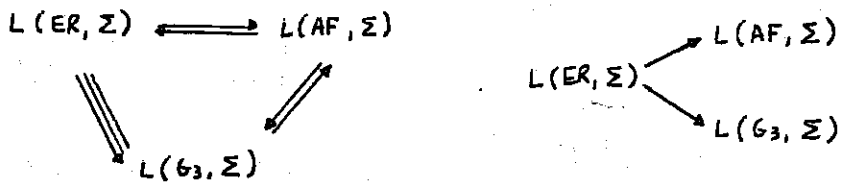
Práctica 5.3: Derivadas

1.- Dada la expresión regular R = 0(01 + 10)* obtener, empleando derivaciones sucesivas por la izquierda, una gramática lineal derecha G y un autómata finito A que verifiquen L(G) = L(R) = L(A).

2.- Demostrar la igualdad o desigualdad de las expresiones regulares siguientes:

$$E_1 = 0^*(10^*)^*0 \quad E_2 = (0^*1^*)^*(0 + 1)^*00^*$$

TEMA 6: PROPIEDADES DE LOS LENGUAJES REGULARES.



- **DERIVADA (POR LA IZQUIERDA) DE UNA EXPRESIÓN REGULAR:**

Supongamos una ER y una $a \in \Sigma$

Definimos la derivada respecto a "a" de la ER como el conjunto de palabras $x \in \Sigma^*$ tales que $ax \in ER$.

$$D_a(ER) = \{ x \in \Sigma^* \ / \ ax \in ER \}$$

→ Es una ER más simple: le hemos quitado la "a" que había al principio.

El número de derivadas sucesivas de una ER es finito.

EJ: $ER = aa^*$ $\qquad D_a(ER) = a^*$

El concepto de derivada de una ER nos va a permitir obtener una gramática de tipo 3 (G_3) que describa el lenguaje L(ER). A partir de esta gramática podremos construir el AF que acepta dicho lenguaje.

- **PROPIEDADES.-**

1.- $D_a(\phi) = \phi$
2.- $D_a(\lambda) = \phi$
} $\forall a$

3.- $D_a(a) = \lambda$

4.- $D_a(b) = \phi \qquad \forall b \in \Sigma, \ b \neq a$

5.- $Da(R+S) = Da(R) + Da(S)$ $\quad R, S$ son ER.

6.- $Da(R.S) = \begin{cases} Da(R).S & \lambda \notin R \\ Da(R).S + Da(S) & \lambda \in R \end{cases}$

⇓ Otra forma de expresarlo.

$Da(R.S) = Da(R).S + \delta(R).Da(S)$ donde $\delta(R) = \begin{cases} \phi & \text{si } \lambda \in \\ \lambda & \text{si } \lambda \in \end{cases}$

7.- $Da(R^*) = Da(R).R^*$

• DEM.-

$$Da(R^*) = Da(\lambda + R + R^2 + R^3 + \dots) = Da(\lambda) + Da(R) + Da(RR) +$$
$$+ Da(RR^2) + Da(RR^3) + \dots = \phi + Da(R) +$$
$$+ [Da(R).R + Da(R)] + [Da(R).R^2 + \underbrace{Da(R).R + Da(R}_{Da(R^2)}) +$$
$$+ [Da(R).R^3 + Da(R).R^2 + Da(R).R + Da(R)] + \dots =$$
$$= Da(R) + Da(R).R + Da(R).R^2 + \dots =$$
$$= Da(R)[\lambda + R + R^2 + \dots] = Da(R).R^*$$

8.- $D_\lambda(R) = \{x \in \Sigma^* : \lambda x \in R\} = (R)$

• DERIVADAS SUCESIVAS:

La operación de derivación de ER puede aplicarse reiteradamente respecto al mismo símbolo u otros diferentes.

$$D_{ab}(R) = D_b[Da(R)] = \{x \in \Sigma^* : abx \in R\}$$

El conjunto de las derivadas de una ER es finito, ya que en cada derivación la longitud de la cadena disminuye.

Ej: $R = \underbrace{aa^*}_{s}\underbrace{bb^*}_{t}$

$D_a(R) = a^*bb^* = D_a(S).T \quad (\lambda \notin S)$

$D_b = \emptyset$

$D_{aa}(R) = D_a[D_a(R)] = D_a(\underbrace{a^*}_{S}\underbrace{bb^*}_{T}) = D_a(a^*).T + \underbrace{D_a(T)}_{\emptyset} \quad (\lambda \in S) =$

$= D_a(a).a^*bb^* = \lambda a^*bb^* + \emptyset = \underline{a^*bb^*}$

↳ Se repite!!

$D_a(R) = D_{aa}(R)$

Aquí cortamos porque sabemos que $D_{aaa}(R) = a^*bb^*$

$D_{ab}(R) = D_b[D_a(R)] = D_b(a^*bb^*) = D_b(a^*)bb^* + D_b(bb^*) =$

$= \emptyset + D_b(b)b^* = \emptyset + b^* = b^*$

$D_{aab}(R) = D_b[D_{aa}(R)] = D_b(a^*bb^*) = D_b(a^*)bb^* + D_b(bb^*) = \emptyset + b^* = b^*$

$D_{aba}(R) = D_a[D_{ab}(R)] = D_a(b^*) = \emptyset$

$D_{abb}(R) = D_b[D_{ab}(R)] = D_b(b^*) = D_b(b).b^* = \lambda b^* = b^*$

- OBTENCIÓN DE LA GRAMÁTICA GENERADORA (GLD) DE LA ER:

Es a partir de las derivadas izquierdas.

A partir de una ER que define un Lenguaje Regular, se puede construir la gramática que genera dicho lenguaje → Si R_0 ER(Σ) ⇒ $\exists G_3$ (GLD) / $L(G_3) = L(R_0)$

Esta gramática tendrá la estructura:

$G = (\Sigma_T, \Sigma_N, S, \mathbb{P})$ donde $\begin{cases} \Sigma_T = \Sigma \\ \Sigma_N = \{ D_x(R_0) \cdot x \in \Sigma^* \} \setminus \{\lambda, \emptyset\} \end{cases}$

↳ ER de la que se parte y todas las ER que se obtienen en las distintas derivadas sucesivas respecto a todos los símbolos de entrada.

$\mathbb{P} = \begin{cases} \text{Si } D_a(R) = S \neq \lambda, \emptyset \longrightarrow R ::= aS \\ \text{Si } \lambda \in D_a(R) \longrightarrow R ::= a \\ \text{Si } \lambda = D_a(R) \longrightarrow R ::= a \\ \text{Si } \lambda \in R_0 \longrightarrow R_0 ::= \lambda \end{cases}$ $\quad S = R_0 = D_\lambda(R_0)$

↳ ésta sólo se aplica a la de partida.

EJ: $R_0 = \underbrace{1^*}_{S}\ \underbrace{(0+1)^*}_{T}$ Dada esta ER, se obtiene la gramática que genera el lenguaje asociado, calculando todas las derivadas distintas posibles.

$$D_0(R_0) = D_0(S)\,T + D_0(T) = D_0(1^*)(0+1)^* + D_0[(0+1)^*] = D_0(0+1)(0+1)^* = (0+1)^* =$$

($D_0(1^*) = \emptyset$; $D_0(0) + D_0(1) = \lambda + \emptyset$)

$$D_1(R_0) = D_1(1^*)(0+1)^* + D_1[(0+1)^*] = D_1(1)\,1^*(0+1)^* + D_1(0+1)(0+1)^* =$$

($D_1(1^*) = \lambda$; $D_1(0) + D_1(1) = \emptyset + \lambda$)

$$= 1^*(0+1)^* + (0+1)^* = (1^* + \lambda)(0+1)^* = 1^*(0+1)^* = R_0$$

$$D_{00}(R_0) = D_0[(0+1)^*] = D_0(0+1)(0+1)^* = \lambda(0+1)^* = (0+1)^* = D_0(R_0) = T$$

$$D_{01}(R_0) = D_1[(0+1)^*] = D_1(0+1)(0+1)^* = \lambda(0+1)^* = (0+1)^* = D_0(R_0) = T$$

Ya no se dan derivadas diferentes, se repiten todas.

Esto ha dado lugar a la siguiente gramática:

$G = (\Sigma_T, \Sigma_N, S, P)$ donde.

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

$\Sigma_N = \{R_0, T\}$

$S = R_0$

$$P = \begin{cases} R_0 ::= 0T \mid 1R_0 \mid 01 \mid 1 \mid \lambda \\ T ::= 0T \mid 1T \mid 01 \mid 1 \end{cases}$$

Ej: $R_0 = \underbrace{a}_{R}\underbrace{(b^* + a^*a)}_{S}b$

$D_a(R_0) = D_a(R).S \quad (\lambda \notin R) = \lambda(b^* + a^*a)b = (b^* + a^*a)b = S$

$D_b(R_0) = D_b(R).S = D_b(a)(b^* + a^*a)b = \emptyset$

$D_\lambda(R_0) = R_0 \qquad R_0 \equiv q_0 \equiv S$ $\quad (q_0) \xrightarrow{a} (q_1) \Rightarrow$ Derivada con respecto a "a" de R_0.

$\delta(q_0, a) = q_1$ siendo q_1 la ER S / $R_0 ::= aS$

$D_{aa}(R_0) = D_a(D_a(R_0)) = D_a(S) = D_a[\underbrace{(b^* + a^*a)}_{T}\underbrace{b}_{V}] = D_a(T).V =$

$= (\underbrace{D_a(b^*)}_{\emptyset} + D_a(a^*a))b = (\underbrace{D_a(a^*)}_{\lambda}.a + D_a(a))b = (\lambda.a^*a + \lambda)b = a^*b =$

$D_{ab}(R_0) = D_b(D_a(R_0)) = D_b(S) = D_b[(b^* + a^*a)b] = (D_b(b^*) + \underbrace{D_b(a^*a)}_{\emptyset})b =$

$= (\lambda b^*).b = b^*b = V.$ → Está mal. Es b^*

$D_{aaa}(R_0) = D_a(T) = D_a(\underbrace{a^*}\underbrace{b}) = D_a(a^*)b + \underbrace{D_a(b)}_{\emptyset} = D_a(\underbrace{a}_{\lambda})a^*b = a^*b = T$ Finito.

$D_{aab}(R_0) = D_b(T) = D_b(\underbrace{a^*}\underbrace{b}) = \underbrace{D_b(a^*)}_{\emptyset}b + D_b(b) = \lambda$ Finito

$D_{aba}(R_0) = D_a(V) = D_a(b^*b) = \emptyset$

$D_{abb}(R_0) = D_b(V) = D_b(\underbrace{b^*}\underbrace{b}) = D_b(b^*)b + \underbrace{D_b(b)}_{\lambda} = \underbrace{D_b(b)}_{\lambda}b^*b + \lambda = b^*b + \lambda =$

$= b^* = W$ (prop de I)

$D_a(W) = D_a(b^*) = \emptyset$

$D_b(W) = D_b(b^*) = D_b(b)b^* = b^* = W$

Gramática resultante: $G = (\Sigma_T, \Sigma_N, R_0, P)$ donde:

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

$\Sigma_N = \{R_0, S, T, V, W\}$

$$P \equiv \begin{cases} R_0 ::= aS \\ S ::= aT \mid bV \\ T ::= aT \mid b \end{cases} \qquad \begin{aligned} V &::= bW \mid b \\ W &::= bW \mid b \end{aligned}$$

EJ: $R_0 = \underbrace{(b + ab^*a)^*}_{S}\underbrace{ab^*}_{T}$

$$D_a(R_0) = D_a[(b+ab^*a)^*]\,T + D_a(ab^*) = D_a[(b+ab^*a)]\,(b+ab^*a)^*ab^* + b^* =$$

$$= [\underset{\emptyset}{D_a(b)} + \underset{b^*a}{D_a(ab^*a)}]\,(b+ab^*a)^*ab^* + b^* = b^*a\,(b+ab^*a)^*ab^* + b^* = S$$

$$D_b(R_0) = D_b[(b+ab^*a)^*]\,T + \underset{\emptyset}{D_b(ab^*)} = D_b[(b+ab^*a)]\,(b+ab^*a)^*ab^* =$$

$$= [\underset{\lambda}{D_b(b)} + \underset{\emptyset}{D_b(ab^*a)}]\,(b+ab^*a)^*ab^* = \lambda\,(b+ab^*a)^*ab^* =$$

$$= (b+ab^*a)^*ab^* = R_0$$

EJ: $R_0 = aa^*bb^*$ Lo primero es calcular las derivadas sucesivas.

$D_a(R_0) = a^*bb^* = R_1$

$D_b(R_0) = \emptyset$

$D_{aa}(R_0) = D_a(R_1) = D_a(a^*bb^*) = D_a(a^*)bb^* + D_a(bb^*) = a^*bb^* = R_1 \rightarrow$ Ya no proporciona nuevas expresiones

$D_{ab}(R_0) = D_b(R_1) = D_b(a^*bb^*) = D_b(a^*)bb^* + D_b(bb^*) = D_b(b)b^* = b^* = R_2$

$D_{aba}(R_0) = D_a(R_2) = D_a(b^*) = \emptyset$

$D_{abb}(R_0) = D_b(R_2) = D_b(b^*) = D_b(b)b^* = b^* = R_2$

* Hemos terminado el proceso, pues las nuevas derivadas serán iguales a $\emptyset$ o a alguna de las anteriores *

$D_a(R_0) = R_1 \qquad \lambda \notin D_a(R_0)$

$D_a(R_1) = R_1 \qquad \lambda \notin D_a(R_1)$

$D_b(R_1) = R_2 \qquad \lambda \in D_b(R_1)$

$D_b(R_2) = R_2 \qquad \lambda \in D_b(R_2)$

Por lo tanto la gramática será: $G = (\Sigma_T, \Sigma_N, \overset{\text{axioma}}{\uparrow R_0}, P)$

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

$\Sigma_N = \{R_0, R_1, R_2\}$

$$P \equiv \begin{cases} R_0 ::= aR_1 \\ R_1 ::= aR_1 \mid bR_2 \mid b \\ R_2 ::= bR_2 \mid b \end{cases}$$

- OBTENCIÓN DEL AUTÓMATA FINITO:

A partir de la gramática obtenida a través de las derivadas de ER se puede obtener el AF por el método visto anteriormente en otro tema. Sin embargo, también se puede obtener el AF directamente desde las derivadas.

Sea R una ER(Σ), entonces existe un AF tal que el lenguaje aceptado por el AF es igual que el lenguaje generado por la ER; es decir, vamos a construir un AF que acepte el mismo lenguaje que la ER.

Este AF se define como:

$$AF = (\Sigma, Q, q_0, \delta, F)$$

donde:

$Q = \{ D_y(R) : y \in \Sigma^* \}$ → Todas las distintas expresiones que se obtienen al derivar sucesivamente

$q_0 = D_\lambda(R) = R$ → ER de la que se parte.

δ: $D_a(S) = T \longrightarrow \delta(S, a) = T$ (S —a→ T)

$D_a(S) = \phi \longrightarrow \delta(S, a) = \phi$

$F = \{ D_x(R) : x \in \Sigma^* \} \mid \overline{\lambda \in D_x(R)}$ (S —a→ (T))

↳ Todas las derivadas a las que pertenezca λ.

EJ: Calcular la gramática equivalente a la ER R y luego el autómata equivalente

$R = \lambda + aa^*b\,(aa^*b)^* + aa^*$

↳ Primero la simplifico todo lo posible.

$R = (aa^*b)^* + aa^* \longrightarrow \lambda \in R$ { Entonces el estado inicial R es también estado final.

$D_a(R) = D_a(aa^*b)^* + D(aa^*) = [D_a(aa^*b)](aa^*b)^* + D_a(a)a^* =$

$= [D_a(a) \cdot (a^*b)]\,(aa^*b)^* + a^* = a^*b\,(aa^*b)^* + a^* = S \longrightarrow \lambda \in S$ Por lo tanto S es estado final.

$D_b(R) = D_b(aa^*b)^* + D_b(aa^*) = D_b(aa^*b).(aa^*b)^* = \phi$ → Por aquí no puedo seguir.
(aa* = ∅)

$D_{aa}(R) = D_a(S) = D_a[a^*b(aa^*b)^*] + D_a(a^*) = D_a(a^*).b(aa^*b)^* + D_a[b(aa^*b)^*] + D_a(a)a$

$= a^*b(aa^*b)^* + \phi + a^* = a^*b(aa^*b)^* + a^* = S \qquad \lambda \in S$

$D_{ab}(R) = D_b(S) = D_b[a^*b(aa^*b)^*] + D_b(a^*) = D_b(a^*).b(aa^*b)^* + D_b[b(aa^*b)^*] + \phi =$

$= \phi + (aa^*b)^* + \phi = (aa^*b)^* = T$

$\lambda \in T$ → T es estado final.

$D_a(T) = D_a(aa^*b)^* = D_a(aa^*b).(aa^*b)^* = a^*b(aa^*b)^* = U$

$\lambda \notin U$ U no es estado final

$D_b(T) = D_b(aa^*b)^* = \phi$

$D_a(U) = D_a[a^*b(aa^*b)^*] = a^*b(aa^*b)^* = U$

$D_b(U) = D_b[a^*b(aa^*b)^*] = (aa^*b)^* = T \qquad \lambda \in T$ T es un estado final.

$G = (\Sigma_T, \Sigma_N, R, P)$ (axioma) donde. $\Sigma_T = \{a, b\}$

$\Sigma_N = \{R, S, T, U\}$

$P = \begin{cases} R ::= \lambda \mid a \mid aS \\ S ::= a \mid b \mid aS \mid bT \\ T ::= aU \\ U ::= aU \mid bT \mid b \end{cases}$

Ahora calcular el autómata equivalente:

Pongo el estado ∅

$AF = (\{a,b\}, \{R, S, T, U, \phi\}, R, f, \{R, S, T\})$

$f(R,a) = S$ $\quad f(U,a) = U$

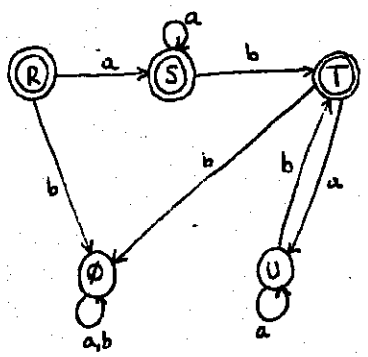
$f(R,b) = \phi$ $\quad f(U,b) = T$

$f(S,a) = S$ $\quad f(\phi,a) = \phi$

$f(S,b) = T$ $\quad f(\phi,b) = \phi$

$f(T,a) = U$

$f(T,b) = \phi$



¡¡PONGO TODAS LAS COMBINACIONES!!

→ algunas darán vacío (∅)

EJ: $R = 0^*111^*$

$D_0(R) = D_0(0^*111^*) = D_0(0^*)\,111^* + D_0(111^*) = D_0(0)\,0^*\,111^* = 0^*111^* = R$ (ya no sigo) $\quad \lambda \notin R \rightarrow$ R no es estado final.

$D_1(R) = D_1(0^*111^*) = D_1(0^*)\,111^* + D_1(111^*) = \phi + 11^* = 11^* = S$

$\lambda \notin S \rightarrow$ S no es estado final.

$D_{10}(R) = D_0(S) = D_0(11^*) = \phi$

$D_{11}(R) = D_1(S) = D_1(11^*) = 1^* = T \quad \lambda \in T \rightarrow$ T es estado final.

$D_0(T) = D_0(1^*) = \phi$

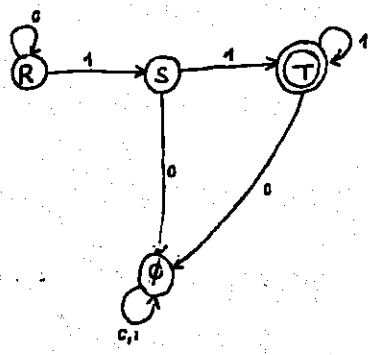
$D_1(T) = D_1(1^*) = 1^* = T$

Gramática resultante $\rightarrow$ $G = (\{0,1\}, \{R,S,T\}, R, P)$ (Σ_T, Σ_{NT}, axioma) donde:

$$P = \begin{cases} R ::= 0R \mid 1 \\ S ::= 1T \mid 1 \\ T ::= 1T \mid 1 \end{cases}$$

Autómata $\rightarrow$ $AF = (\{0,1\}, \{R,S,T,\phi\}, R, \delta, \{T\})$ (R: estado inicial)

$\delta(R,0) = R$
$\delta(R,1) = S$
$\delta(S,0) = \phi$
$\delta(S,1) = T$
$\delta(T,0) = \phi$
$\delta(T,1) = T$
$\delta(\phi,0) = \phi$
$\delta(\phi,1) = \phi$



EJ: $R = b(b^* + a^*a)a$

$D_a(R) = D_a[b(b^* + a^*a)a] = D_a(b).(b^* + a^*a)a = \emptyset$

$D_b(R) = D_b(b).(b^* + a^*a)a = (b^* + a^*a)a = S \qquad \lambda \notin S$
$\qquad\qquad\quad \lambda$

$D_a(S) = D_a(b^* + a^*a)a = D_a(b^* + a^*a)a = [D_a(b^*) + D_a(a^*a)]a =$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \emptyset$
$= [D_a(a^*).a + D_a(a)]a = (a^*a + \lambda)a =$
$\qquad\qquad\quad = a^*$
$= a^*a = T$
→ Puede que falte un λ

$D_b(S) = D_b[(b^* + a^*a)a] = (b^* + \emptyset)a = b^*a = U$

$D_a(T) = D_a(a^*a) = D_a(a^*).a + D_a(a) = a^*a + \lambda = a^* = V \qquad \lambda \in V$

PROPIEDADES DE LAS ER.

$D_b(T) = \emptyset$

$D_a(U) = D_a(b^*a) = D_a(b^*)a + D_a(a) = \emptyset + \lambda = \lambda = X \rightarrow \lambda \in D_a(U) = X$

$D_b(U) = D_b(b^*a) = D_b(b^*)a + D_b(a) = b^*a = U$

$D_a(V) = D_a(a^*) = a^* = V$

$D_b(V) = D_b(a^*) = \emptyset$

Gramática → $G = (\{a,b\}, \{R,S,T,U,V\}, R, P)$

$P =$
- $R ::= bS$
- $S ::= aT \mid bU$
- $T ::= aV \mid a$
- $U ::= bU \mid a$
- $V ::= aV \mid a$

$f =$
- $f(R,a) = \emptyset$
- $f(R,b) = S$
- $f(S,a) = T$
- $f(S,b) = U$
- $f(T,a) = V \qquad V \in F$
- $f(T,b) = \emptyset$
- $f(U,a) = X \qquad \lambda \in F$
- $f(U,b) = U$
- $f(V,a) = V \qquad V \in F$
- $f(V,b) = \emptyset$
- $f(\emptyset,a) = f(\emptyset,b) = \emptyset$

↑ estados finales

- DERIVADA (POR LA DERECHA) DE UNA EXPRESIÓN REGULAR:

Definimos la derivada por la derecha respecto a "a" de una ER como:

$$R \in ER(\Sigma) \qquad D_a(R) = \{ x \in \Sigma^* : xa \in R \}$$

· PROPIEDADES.-

1.- $D_a(\phi) = \phi$

2.- $D_a(\lambda) = \phi$

3.- $D_a(a) = \lambda$

4.- $D_a(b) = \phi \quad \forall b \in \Sigma,\ b \neq a$

5.- $D_a(R+S) = D_a(R) + D_a(S)$

ÚNICA DIFERENTE QUE POR LA IZQ.

6.- $$D_a(R.S) = \begin{cases} R.D_a(S) & \lambda \notin S \\ R.D_a(S) + D_a(R) & \lambda \in S \end{cases}$$

otra forma

$$D_a(R.S) = R.D_a(S) + D_a(R).\delta(S) \quad \text{donde:} \quad \delta(S) = \begin{cases} \phi & \text{si } \lambda \notin S \\ \lambda & \text{si } \lambda \in S \end{cases}$$

7.- $D_a(R^*) = R^*.D_a(R)$

· DEM.-

$$\begin{aligned} D_a(R^*) &= D_a(\lambda + R + R^2 + R^2 + \dots) = D_a(\lambda) + D_a(R) + D_a(RR) \\ &+ D_a(R^2R) + \dots = \phi + D_a(R) + [R.D_a(R) + D_a(R)] + \\ &+ [R^2 D_a(R) + D_a(R^2)] + \dots = \\ &= D_a(R) + R.D_a(R) + D_a(R) + [R^2 D_a(R) + R.D_a(R) + D_a(R \\ &= D_a(R) + RD_a(R) + R^2 D_a(R) + \dots = \\ &= [\lambda + R + R^2 + R^3 + \dots]\, D_a(R) = R^*.D_a(R) \end{aligned}$$

- OBTENCIÓN DE LA GRAMÁTICA GENERADORA (GLI) DE LA ER:

Es a partir de las derivadas derechas.

$\in ER(\Sigma)$ $\quad R \in ER(\Sigma) \Rightarrow \exists$ GLI. $G_3 : L(G_3) = L(R)$

$[G = (\Sigma_T, \Sigma_N, S, P)]$ donde:

$\Sigma_T = \Sigma$

$\Sigma_N = \{ D_x(R) : x \in \Sigma^* \} - \{\lambda, \phi\}$ → ER de la que se parte y todas las distintas derivadas sucesivas.

S = axioma = $R = D_\lambda(R)$

$$P = \begin{cases} \lambda \in R & \longrightarrow R ::= \lambda \\ D_a(R) = \lambda & \longrightarrow R ::= a \\ D_a(R) = S & \longrightarrow R ::= Sa \\ \lambda \in S & \longrightarrow R ::= a \end{cases}$$

EJ: $R = 0(01+10)^*$

porque $\lambda \in (01+10)^*$

$D_0(R) = 0 \cdot D_0((01+10)^*) + D_0(0) = 0 \cdot (01+10)^* \cdot D_0(01+10) + D_0(0) =$

$= 0 \cdot (01+10)^* \cdot [D_0(01) + D_0(10)] + \lambda = 0 \cdot (01+10)^* 1 + \lambda = S \quad \lambda \in S$

$D_1(R) = 0 \cdot D_1((01+10)^*) + D_1(0) = 0 \cdot (01+10)^* \cdot D_1(01+10) + D_1(0) =$

$= 0 \cdot (01+10)^* \cdot 0 + \phi = 0(01+10)^* 0 = T \quad \lambda \notin T$

$D_0(S) = 0 \cdot D_0((01+10)^* 1) + D_0(\lambda) = 0 \cdot \phi + \phi = \phi$

$D_1(S) = 0 \cdot D_1((01+10)^* 1) + D_1(\lambda) = 0(01+10)^* + \phi = 0 \cdot (01+10)^* = R$

$D_0(T) = D_0[0(01+10)^* 0] = 0(01+10)^* = R$

$D_1(T) = D_1[0(01+10)^* 0] = \phi$

Gramática → $G = \{\{0,1\}, \{R,S,T\}, R, P\}$ donde: (R = axioma)

$D_0(R) = S \quad \lambda \in S$

$D_1(R) = T$

~~$D_0(S) = \phi$~~

$D_1(S) = R$

$D_0(T) = R$

~~$D_1(T) = \phi$~~

$$P = \begin{cases} R ::= S0 \mid T1 \mid 0 \\ S ::= R1 \\ T ::= R0 \end{cases}$$

PROPIEDADES DE LOS LENGUAJES REGULARES:

1.- PROPIEDADES DE CIERRE

- "El conjunto de los lenguajes regulares es cerrado respecto a la unión, la concatenación y cierre de Kleene".

Sean dos lenguajes L y L' representados por las expresiones regulares R y S. Entonces:

- $L \cup L'$ está representado por $\rightarrow R+S$: $L(R+S) = L(R) \cup L(S) = L \cup L'$
- $L.L' \rightarrow R.S$
- $L^* \rightarrow R^*$

Estos tres lenguajes son, por lo tanto, representables por medio de ER, luego también son regulares.

- "El conjunto de los lenguajes regulares es cerrado respecto a la complementación".

Es decir, si L es un lenguaje regular sobre el alfabeto Σ, $\Sigma^* - L$ es también lenguaje regular.

Autómata que acepta el lenguaje L:

$A = (\Sigma, Q, q_0, \delta, F)$ $\quad L(A) = L$

Autómata que acepta el lenguaje complementario:

$B = (\Sigma, Q, q_0, \delta, Q-F)$

$x \in \Sigma^* \quad \delta(q_0, x) \in Q-F \rightarrow x \notin L$ porque $\{Q-F\} \cap F = \emptyset$

EJ: Dada la ER : $R = (0+1)^*01$ calcular el lenguaje complementario y su autómata.

Calculo el autómata complementario a partir del autómata de R construido por derivación izquierda.

$D_0(R) = D_0[(0+1)^*]01 + D_0(01) = (0+1)^*01 + 1 = S$

$D_1(R) = D_1[(0+1)^*]01 + \underset{\emptyset}{D_1(01)} = (0+1)^*01 = R$

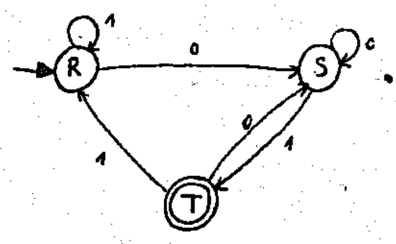
$D_0(S) = D_0[(0+1)^*01 + 1] = D_0((0+1)^*01) + D_0(1) = (0+1)^*01 + 1 = S$

$D_1(S) = D_1[(0+1)^*01 + 1] = D_1((0+1)^*01) + D_1(1) = (0+1)^*01 + \phi + \lambda = T \quad \lambda \in T$

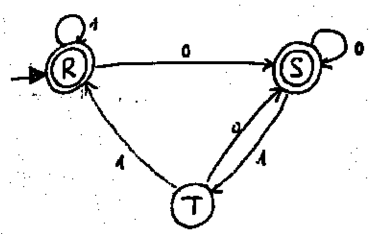
$D_0(T) = D_0((0+1)^*01) + \underset{\emptyset}{D_0(\lambda)} = S$

$D_1(T) = D_1((0+1)^*01) + \underset{\emptyset}{D_1(\lambda)} = R$

Autómata: $L = L(R)$



Autómata complementario: $\overline{L}$



Ahora hallo el lenguaje $\overline{L}$ por las ecuaciones características.

$R = 1R + 0S + \lambda$ ← porque es estado final.

$S = 0S + 1T + \lambda$

$T = 0S + 1R$

$S = 0S + 1(0S + 1R) + \lambda = (0+10)S + 11R + \lambda \; ; \quad S = (0+10)^*(11R + \lambda)$

$R = 1R + 0[(0+10)^*(11R+\lambda)] + \lambda = [1 + 0(0+10)^*11]R + 0(0+10)^* + \lambda$

$R = [1 + 0(0+10)^*11]^*[0(0+10)^* + \lambda]$

EJ: Sea $R = (a + b^*ab)^*$ la ER de un lenguaje L. Encontrar el autómata que acepta L.

$$D_a(R) = D_a(a+b^*ab)^* = D_a(a+b^*ab).(a+b^*ab)^* = [D_a(a) + D_a(b^*ab)].(a+b^*ab)^* =$$
($a \to \lambda$)
$$= [\lambda + D_a(b)^*.ab + D_a(ab)].(a+b^*ab)^* = (\lambda+b)(a+b^*ab)^* = S \quad \lambda \in S$$
($b \to \emptyset$, $ab \to b$)

$$D_b(R) = [D_b(a) + D_b(b^*ab)].(a+b^*ab)^* = [D_b(b^*).ab + D_b(ab)](a+b^*ab)^* =$$
($a \to \emptyset$, $ab \to \emptyset$)
$$= b^*ab(a+b^*ab)^* = T$$

$$D_a(S) = D_a[(\lambda+b)(a+b^*ab)^*] = D_a(\lambda+b).(a+b^*ab)^* + D_a(a+b^*ab)^* =$$
$$= [D_a(\lambda) + D_a(b)](a+b^*ab)^* + D_a(a+b^*ab).(a+b^*ab)^* =$$
($\lambda \to \emptyset$, $b \to \emptyset$)
$$= [D_a(a) + D_a(b^*ab)](a+b^*ab)^* = [\lambda + D_a(b^*)ab + D_a(ab)]$$
($a \to \lambda$, $ab \to \emptyset$)
$$.(a+b^*ab)^* = (\lambda+b)(a+b^*ab)^* = S$$

$$D_b(S) = D_b[(\lambda+b)(a+b^*ab)^*] = D_b(\lambda+b).(a+b^*ab)^* + D_b(a+b^*ab)^* =$$
$$= [D_b(\lambda) + D_b(b)].(a+b^*ab)^* + D_b(a+b^*ab)^* = (a+b^*ab)^* +$$
($\lambda \to \emptyset$, $b \to \lambda$)
$$+ D_b(a+b^*ab).(a+b^*ab)^* = \overset{(a+b^*ab)^+}{D_b}(b^*ab).(a+b^*ab)^* =$$
($a \to \emptyset$, $ab \to \emptyset$)
$$= [D(b^*)ab + D_b(ab)].(a+b^*ab)^* =$$
$$= (a+b^*ab)^* + b^*ab(a+b^*ab)^* = U$$

$$D_a(T) = D_a(b^*ab(a+b^*ab)^*) = D_a(b^*ab).(a+b^*ab)^* = [D_a(b^*)ab + D_a(ab)]$$
($b^* \to \emptyset$)
$$.(a+b^*ab)^* = b.(a+b^*ab)^* = V$$

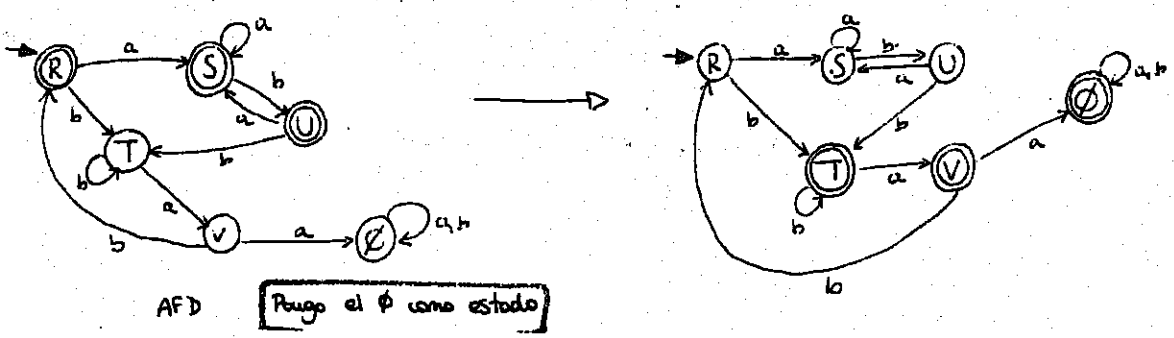
$$D_b(T) = b^*ab^*(a+b^*ab)^* = T$$

$$D_a(U) = (\lambda+b)(a+b^*ab)^* + b(a+b^*ab)^* = S$$

$$D_b(U) = b^*ab(a+b^*ab)^* + b^*ab(a+b^*ab)^* = T$$

$$D_a(V) = \emptyset$$

$$D_b(V) = (a+b^*ab)^* = R$$



AFD — Pongo el $\emptyset$ como estado

- "El conjunto de lenguajes regulares es cerrado respecto a la intersección".

Siendo L_1 y L_2 lenguajes regulares $\longrightarrow$ $\boxed{L_1 \cap L_2}$ es un lenguaje regular.

$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ $\rightarrow$ Por las Leyes de Morgan.

Sea $A_1 = (\Sigma, Q_1, q_{01}, f_1, F_1)$ un autómata cuyo $L(A_1)$ es lenguaje regular.

Sea $A_2 = (\Sigma, Q_2, q_{02}, f_2, F_2)$ un autómata cuyo $L(A_2)$ es lenguaje regular.

Obtenemos el autómata intersección:

$A = (\Sigma, Q_1 \times Q_2, [q_{01}, q_{02}], f, F)$ donde:

$$\begin{cases} f : (Q_1 \times Q_2) \times \Sigma \longrightarrow Q_1 \times Q_2 \\ f\,[(q_1, q_2), e] \longrightarrow [f(q_1, e), f(q_2, e)] \end{cases}$$

$$F = \{(q_1, q_2) : q_1 \in F_1 \wedge q_2 \in F_2\}$$

$L(A) = L(A_1) \cap L(A_2)$

$x \in L(A) \iff \exists\, q_1 \in F_1 : q_1 \in f(q_{01}, x)$ $\quad x \in L(A_1)$ $\iff$ (si hay una palabra que me lleve del estado inicial a uno final.)

$\wedge$

$\exists\, q_2 \in F_2 : q_2 \in f(q_{02}, x)$ $\quad x \in L(A_2)$

$f[(q_{01}, q_{02}), x] = [f_1(q_{01}, x), f_2(q_{02}, x)]$ $\quad f_1(q_{01}, x) \in F_1$, $f_2(q_{02}, x) \in F_2$

- UNIÓN DE LENGUAJES REGULARES:

$$L(A) = L(A_1) \cup L(A_2)$$

$$A = (\Sigma, Q_1 \times Q_2, [q_{01}, q_{02}], \delta, F)$$

donde $F = \{(q_1, q_2) \ / \ q_1 \in F_1 \ \boxed{\vee} \ q_2 \in F_2\}$

- DIFERENCIA DE AUTÓMATAS / LENGUAJES REGULARES:

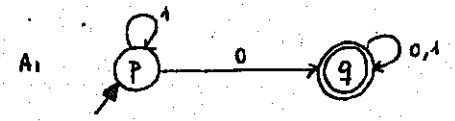
Dados $L(A_1)$ y $L(A_2)$, ambos lenguajes regulares, entonces la diferencia de estos lenguajes es también un lenguaje regular:

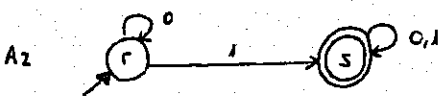
$$L(A_1),\ L(A_2) \text{ regulares} \longrightarrow L(A_1) - L(A_2) \text{ regular} = \underline{L(A_1) \cap \overline{L(A_2)}}$$

$$L, M \longrightarrow L - M = \underline{L \cap \bar{M}}$$

$$\left[F = \{(q_1, q_2) \ / \ q_1 \in F_1 \ \boxed{\wedge} \ q_2 \in \underline{Q_2 - F_2}\}\right.$$

EJ: Dados los siguientes autómatas A_1 y A_2, calcular la unión, intersección y diferencia.





$$A = \{\underbrace{(0,1)}_{\Sigma}, \underbrace{[(p,r),(p,s),(q,r),(q,s)]}_{Q_1 \times Q_2}, \underbrace{(p,r)}_{\text{estado inicial}}, \delta, F\}$$

- Cálculo de δ:

$$\delta((p,r),0) = [\delta_1(p,0), \delta_2(r,0)] = [q,r]$$
$$\delta((p,r),1) = [\delta_1(p,1), \delta_2(r,1)] = [p,s]$$

→

$\delta((p,s),0) = [\delta_1(p,0), \delta_2(s,0)] = [q,s]$

$\delta((p,s),1) = [\delta_1(p,1), \delta_2(s,1)] = [p,s]$

$\delta((q,r),0) = [\delta_1(q,0), \delta_2(r,0)] = [q,r]$

$\delta((q,r),1) = [\delta_1(q,1), \delta_2(r,1)] = [q,s]$

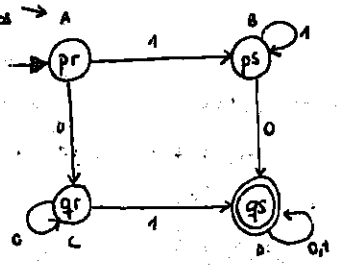
$\delta((q,s),0) = [\delta_1(q,0), \delta_2(s,0)] = [q,s]$

$\delta((q,s),1) = [\delta_1(q,1), \delta_2(s,1)] = [q,s]$

- Estados finales: INTERSECCIÓN → $F = \{(q_1,q_2) / q_1 \in F_1 \wedge q_2 \in F_2\}$

$F = [q,s]$

Pongo nombre más cómodos →



REDENOMINAMOS →

	0	1
→ A	C	B
B	D	B
C	C	D
(D)	D	D

Hallo el lenguaje con las ecuaciones características:

$A = 0C + 1B$

$B = 0D + 1B$

$C = 0C + 1D$

$D = 0D + 1D = (1+0)D + \lambda$

$D = (1+0)^*$

$C = 0C + 1(1+0)^*$; $C = 0^*1(1+0)^*$

$B = 0(1+0)^* + 1B$; $B = 1^*0(1+0)^*$

$A = 00^*1(1+0)^* + 11^*0(1+0)^*$

UNIÓN → Todo es igual, excepto los estados finales.

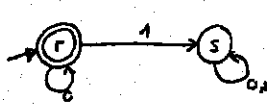
Los estados finales son:

$F = \{(q_1,q_2) / q_1 \in F_1 \vee q_2 \in F_2\}$

$F = \{ps, qr, qs\}$

DIFERENCIA → $L(A_1) - L(A_2) = L(A_1) \cap \overline{L(A_2)}$ ⇒ Lenguaje de las palabras de A_1 que no pertenecen a A_2.

$\overline{L(A_2)}$



$F_2 = \{r\}$

Ahora que tengo $\overline{L(A_2)}$ calculo el autómata intersección igual que antes.

$F = [q,r]$

- LENGUAJE INVERSO:

Si un lenguaje L es lenguaje regular, su inverso también lo es. Por lo tanto, es una operación cerrada.

Podemos calcularlo mediante un autómata o una expresión regular.

$$L = \{100, 01, 010\}$$
$$L^{-1} = \{001, 10, 010\}$$

• POR AUTÓMATA:

Dado un autómata $A = \{\Sigma, Q, q_0, \delta, F\}$ que acepta un lenguaje $L(A)$ vamos a obtener otro autómata A' cuyo lenguaje $L(A')$ va a ser el lenguaje inverso: $L(A)^{-1}$

→ Si en A sólo $\exists$ un estado final:

↓ cambiar las flechas de sentido.

$$A' = \{\Sigma, Q, q_F, \delta^{-1}, \{q_0\}\}$$

q_F: el estado final de A, ahora es el estado inicial.
q_0: el estado inicial de A, ahora es el estado final.

→ Si en A $\exists$ más de un estado final:

$$A' = \{\Sigma, \{Q + q_0'\}, q_0', \delta', \{q_0\}\}$$

q_0': se unirá mediante λ-transiciones a todos los estados finales de A. Esto es para obtener el autómata equivalente con un sólo estado final

• POR EXPRESIONES REGULARES:

→ Propiedades.-

1. $\lambda^{-1} = \lambda$
2. $\emptyset^{-1} = \emptyset$
3. $a^{-1} = a$
4. $\alpha, \beta \in ER(\Sigma)$.
$$\begin{cases} (\alpha + \beta)^{-1} = \alpha^{-1} + \beta^{-1} \\ (\alpha\beta)^{-1} = \beta^{-1}\alpha^{-1} \\ (\alpha^*)^{-1} = (\alpha^{-1})^* \end{cases}$$

EJ: $L(A_1) = \{100, 01, 010\}$

$L(A_2) = \{11, 001\}$

$L(A_1)^{-1} = \{001, 10, 010\}$

$L(A_2)^{-1} = \{11, 100\}$

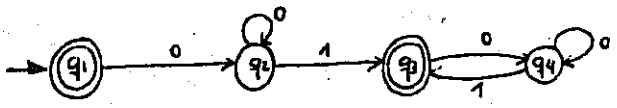
⚠ $L(A_1A_2) = \{10011, 100001, 0111, 01001, 01011, 010001\}$

⚠ $L(A_2A_1) = \{11100, 1101, 11010, 001100, 00101, 001010\}$

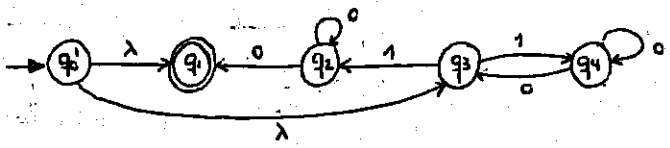
⚠ $(L(A_1)\,L(A_2))^{-1} = L(A_2)^{-1}\,L(A_1)^{-1} = \{11001, 1110, 11010, 100001, 10010, 100010\}$

EJ: Dado el siguiente autómata A, cuyo lenguaje es L(A), calcular el lenguaje inverso del lenguaje L(A) aceptado por el autómata.

A:



A':



$L(A') = L(A)^{-1}$

Pongo estos tres juntos como estado inicial porque están unidas al estado inicial mediante λ-transiciones.

	0	1
→ $\{q_0' q_1 q_3\}$	∅	$\{q_2 q_4\}$
$\{q_2 q_4\}$	$\{q_1 q_2 q_3 q_4\}$	∅
$\{q_1 q_2 q_3 q_4\}$	$\{q_1 q_2 q_3 q_4\}$	$\{q_2 q_4\}$

RENOMBRO

	0	1
→ (A)	D	B
B	C	D
(C)	C	B
D	D	D

$A = \{q_0' q_1 q_3\}$ $\quad C = \{q_1 q_2 q_3 q_4\}$

$B = \{q_2 q_4\}$ $\quad D = \emptyset$

Calculo el lenguaje mediante las ecuaciones características:

$A = 0D + 1B + \lambda$

$B = 0C + 1D$

$C = 0C + 1B + \lambda$

$D = (0+1)D \Rightarrow D = \emptyset$

$A = 1B + \lambda$

$B = 0C$

$C = 0C + 1B + \lambda \Rightarrow C = 0^*(1B + \lambda)$ ✓ $\quad L = 00^*(100^*)^*$

$B = 00^*(1B) + 00^*$; $B = (00^*1)^*00^*$

$(\alpha\beta)^*\alpha = \alpha(\beta\alpha)^*$ ↪ $A = 1(00^*1)^*00^* + \lambda = (100^*)^*100^* + \lambda = (100^*)^* = \underline{\underline{(10^*0)^*}}$

Ahora calculamos el lenguaje inverso por ER en lugar de por autómatas.

$x_1 = 0x_2 + \lambda$ (Esto se hace sobre A, el autómata inicial)

$x_2 = 0x_2 + 1x_3$

$x_3 = 0x_4 + \lambda$

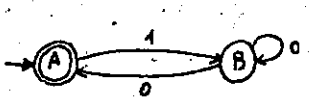
$x_4 = 0x_4 + 1x_3$; $x_4 = 0^*1x_3$

$x_3 = 00^*1x_3 + \lambda$; $x_3 = (00^*1)^*$

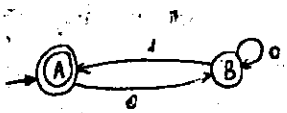
$x_2 = 0x_2 + 1(00^*1)^* = 0^*1(00^*1)^*$

$x_1 = L(A) = 00^*1(00^*1)^* + \lambda = (00^*1)^*$

$$L(A)^{-1} = [(00^*1)^*]^{-1} = [(00^*1)^{-1}]^* = (10^*0)^*$$

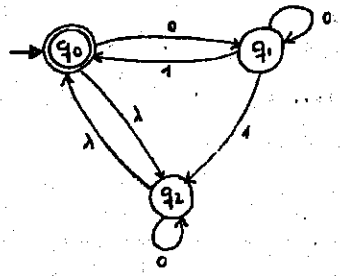


$L(A)^{-1}$



$L(A)$

EJ:



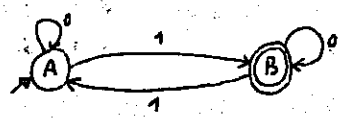
Obtener a partir del autómata expresiones regulares para los lenguajes:

- $\overline{L}$
- $L\overline{L}$
- L^{-1}
- $L \cdot L^{-1}$

(Hecho en la siguiente hoja)

EJ:

A_1 :

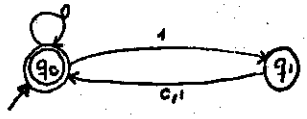


Calcular

a) $L(A_1) \cap L(A_2)$

b) $L(A_1) \cap L(A_1)^{-1}$

A_2 :



a)

A_1	0	1
→ A	A	B
(B)	B	A

A_2	0	1
→ (q0)	q0	q1
q1	q0	q0

∩	0	1
→ Aq0	Aq0	Bq1
Aq1	Aq0	Bq0
(Bq0)	Bq0	Aq1
Bq1	Bq0	Aq0

Renombro →

∩	0	1
→ A	A	D
B	A	C
(C)	C	B
D	C	A

Ecuaciones características:

$A = 0A + 1D$

$B = 0A + 1C$

$C = 0C + 1B + \lambda$

$D = 0C + 1A$

$C = 0^*(1B + \lambda) = 0^*1B + 0^*$

$B = 0A + 1C^*1B + 1C^* = 1C^*1B + (0A + 1C^*) = (1C^*1)^*(0A + 1C^*)$

$D = 00^*1B + 00^* + 1A = 00^*1(1C^*1)^*(0A + 1C^*) + 00^* + 1A =$

$= (00^*1(1C^*1)^*0 + 1)A + \underbrace{00^*1(1C^*1)^*1C^* + 00^*}_{\text{II}} = \textcircled{*}$

$00^*(1C^*)^*1C^* + 00^* = 00^*(1C^*)^*$

$\Rightarrow 00^*[\underbrace{(1C^*)^*1C^*}_{(1C^*)^*} + \lambda]$

$\textcircled{*} = \underbrace{(00^*1(1C^*1)^*0 + 1)}_{\alpha}A + 00^*(1C^*)^*$

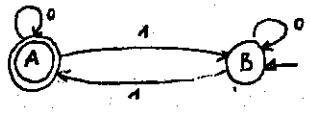
$A = 0A + 1\alpha A + 00^*(1C^*)^* \Rightarrow$ $\boxed{A = (0 + 1\alpha)^* 00^*(1C^*)^*}$

b) A_1: $A = 0A + 1B$ $\quad B = 0B + 1A + \lambda$

$B = 0^*(1A + \lambda)$

$A = 0A + 10^*1A + 10^*$ $\quad \boxed{L(A_1) = (0 + 10^*1)^*10^*}$

$A_1^{-1} = A^1$ $\quad L(A^1) = L(A)^{-1}$



$B = 0B + 1A = 0B + 10^*(1B + \lambda) = (0 + 10^*1)B + 10^*$

$A = 0A + 1B + \lambda$

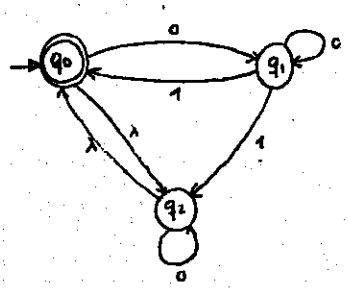
$A = 0^*(1B + \lambda)$ $\quad \boxed{L(A)^{-1} = (0 + 10^*1)^*10^*}$

$L(A)$ es igual que $L(A)^{-1}$. Por lo tanto, $L(A) \cap L(A)^{-1}$ y $L(A) \cup L(A)^{-1}$ seguirán siendo el mismo lenguaje $L(A)$.

CASO ESPECIAL DE LENGUAJE INVERSO.

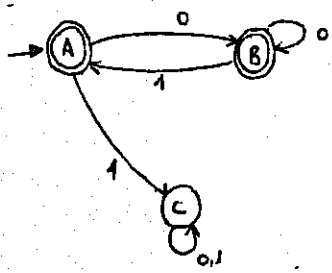
EJ:

A:



$\Omega(q_0) = \Omega(q_2) = \{q_0, q_2\}$

		0	1
(A)	→ $\{q_0\ q_2\}$	$\{q_1 q_2\}$	$\emptyset$
(B)	$\{q_1\ q_2\}$	$\{q_1 q_2\}$	$\{q_0 q_2\}$
C	$\emptyset$	$\emptyset$	$\emptyset$



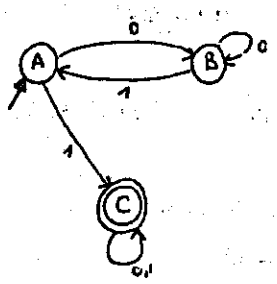
$A = 0B + \lambda = 00^*(1A + \lambda) + \lambda = 00^*1A + 00^* + \lambda = 00^*1A + C^*$

$B = 0B + 1A + \lambda$

$B = 0^*(1A + \lambda)$ $\quad L(A) \Rightarrow \boxed{A = (00^*1)^*0^*}$

$\overline{L(A)}$

A' :



$A = 0B + 1C$

$B = 0B + 1A$

$C = (0+1)C + \lambda$; $C = (0+1)^*$ $B = 0^*1A$

$A = 00^*1A + 1(0+1)^* = (00^*1)^*1\,(0+1)^* =$

$= \overline{L(A)} = (00^*1)^*1(0^*1)^*0^*$

- CORRESPONDENCIA ENTRE AUTÓMATAS Y GRAMÁTICAS DE TIPO 3:

$a_1 .. a_n = x \in L(G_3)$ ↓ GLD

$S ::= a_1 A_1$

$A_1 ::= a_2 A_2$

$A_2 ::= a_3 A_3$

$\vdots$

$A_{n-2} ::= a_{n-1} A_{n-1}$

$A_{n-1} ::= a_n$

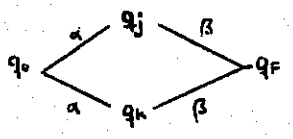
$S \to a_1 A_1 \to a_1 a_2 A_2 \to a_1 a_2 a_3 A_3 \to \dots \to a_1 .. a_{n-1} A_{n-1} \to a_1 .. a_n$

$S \to a_1 .. a_i A_i \to \begin{cases} a_1 .. a_{i+1} A_j \\ a_1 .. a_{i+1} A_k \end{cases} \to a_1 a_2 .. a_n$

$A_i ::= a_{i+1} A_j$
$A_i ::= a_{i+1} A_k$ → 2 producciones distintas.

El autómata que acepte la palabra x será:

$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} q_3 \to \dots \to q_{n-1} \xrightarrow{a_n} q_F$



$x = \alpha\beta$

G_3 GLD ambigüa $\Rightarrow$ AFND ($\not\Leftarrow$)

Es condición necesaria para que una gramática sea ambigüa que el autómata asociado a esa gramática lineal derecha sea no determinista.

* Si al hacer el autómata asociado a la gramática nos da que es un AFD ⇒ La gramática no es ambigüa

Un lenguaje regular nunca es inherentemente ambigüo:

$G \longrightarrow AFND \longrightarrow AFD \longrightarrow G'$

$L(G) = L(G')$

* PROCEDIMIENTO PARA SABER SI UNA GRAMÁTICA ES AMBIGUA.

- Conjunto de estados accesibles para una palabra dada:

$$X_{[x]} = \{ q \ / \ q_0 \to q \}$$

- Conjunto de estados abandonados para una palabra dada.

$$Y_{[x]} = \{ q' \ / \ q' \to q_F \}$$

① Se hace la equivalencia entre la gramática y el autómata.

GLD → A
- AFND : Puede ser ambigüa.
- AFD : No es ambigüa.

② Calculamos el conjunto de los estados $[X_{[x]}, Y_{[x]}]$

3) La gramática es ambigüa ⟺ existe en los árboles $X_{[x]}, Y_{[x]}$ al menos un conjunto A' que tenga como mínimo 2 estados comunes en $X_{[x]}$ e $Y_{[x]}$.

Una palabra ambigüa será la que en el conjunto X va desde un estado inicial a un conjunto A, y en el conjunto Y la que va desde el conjunto A al estado final.

El conjunto de todas las palabras que son ambigüas es el siguiente:

Para un conjunto A, cogemos la palabra mínima que nos lleva al conjunto A y la concatenamos con la intersección de todas las palabras que partiendo de ese estado A nos llevan al estado final.

EJ: $G_3 = \{(0,1), (S,A,B), S, P\}$ donde $P =$

$S ::= 1A$

$A ::= 0A \mid 0B$

$B ::= 0A \mid 1B \mid 1$

Decir si G_3 es ambigüa, y si lo es, decir el conjunto de palabras ambigüas.

$\delta(S,1) = A$

$\delta(A,0) = A$

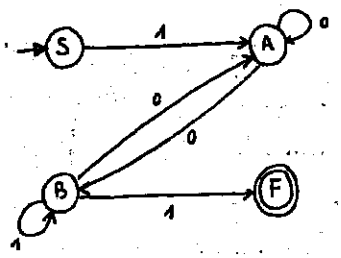
$\delta(A,0) = B$

$\delta(B,C) = A$

$\delta(B,1) = B$

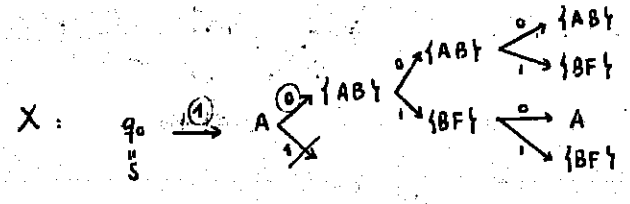
$\delta(B,1) = F$

$Q = \{\Sigma_N\} \cup F$

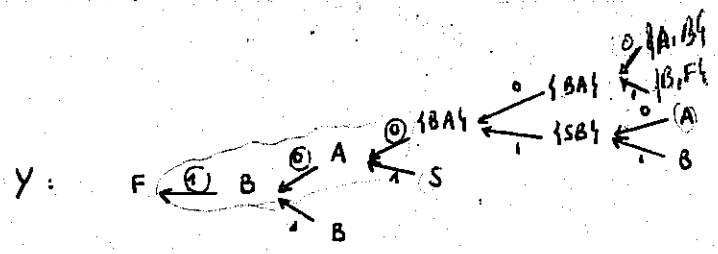


	0	1
S	–	A
A	AB	–
B	A	BF

X:



Y:



10001

$S \to 1A \to 1CA \to 100A \to 1000B \to 10001$

$S \to 1A \to 10B \to 100CA \to 1000B \to 10001$

2 derivaciones izquierdas distintas $\to G_3$ ambigüa

Mínima cadena que llega a $\{AB\} = 10$ $(x : \delta(A, x) = F \wedge \delta(B, x) = F)$

(A: 4 ecs. características; B: 6 ecs. características)

$A = 0A + 0B$

$B = 0A + 1B + 1F$

$F = \lambda$

EJ: Dada la siguiente GLD

- Probar que es ambigüa.
- Para la palabra más corta encontrar 2 árboles de derivación diferentes.
- Encontrar una gramática equivalente no ambigüa.

$S ::= aA \mid aB \mid bB$

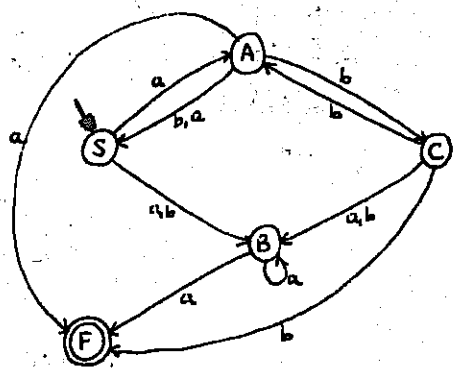
$A ::= bS \mid aS \mid bC \mid a$

$B ::= aB \mid a$

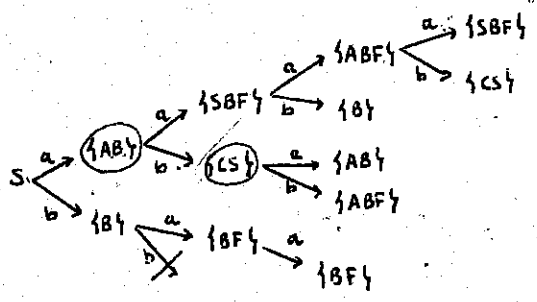
$C ::= bA \mid aB \mid bB \mid b$

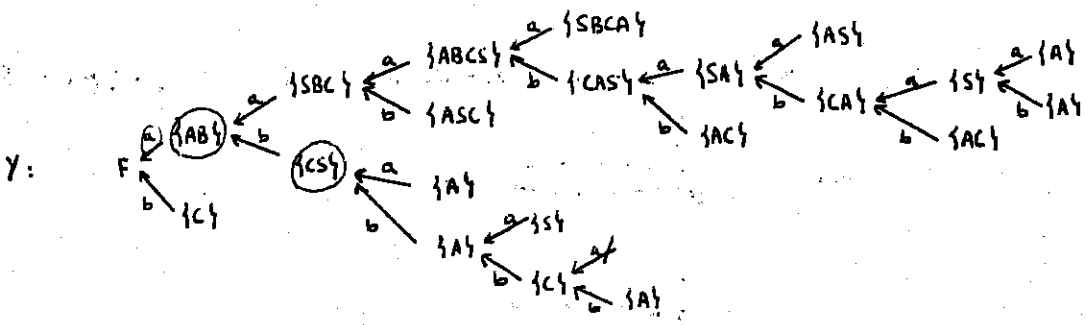
Rango F cuando la producción es $A ::= a$

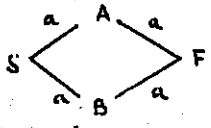
	a	b
→ S	AB	B
A	SF	SC
B	BF	—
C	B	ABF
(F)	—	—



X:



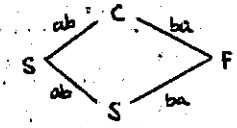




S S

a A a B ⇒ aa

a a

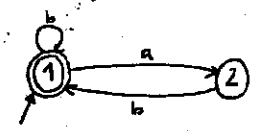


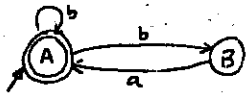
⇒ abba

	a	b
{S}	{AB}	{B}
{AB}	{SBF}	{SC}
{B}	{BF}	∅
{SBF}	{ABF}	{B}
{SC}	{AB}	{ABF}
{BF}	{BF}	∅
{ABF}	{SBF}	{SC}

EJ: Dado el siguiente autómata A, hallar: $L' = L \cap L^{-1}$ A' acepta esta intersec

A:





L^{-1}

A'	a	b
→1A	∅	{(1A),(1B)}
1B	2A	∅
2A	∅	{(1A),(1B)}
2B	∅	∅

Renombramos →

	a	b
→q_0	∅	q_0q_2
q_2	q_3	∅
q_3	∅	q_0q_2
q_1	∅	∅

$g[(1,A),a] = \{g(1,a)\} \times \{g(A,a)\} = \{2\} \times \{\emptyset\} = \emptyset$

$g[(1,A),b] = \{g(1,b)\} \times \{g(A,b)\} = \{1\} \times \{A,B\} = \{(1A),(1B)\}$

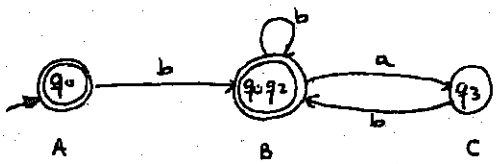
$g[(1,B),a] = \{2\} \times \{A\} = 2A$

$g[(1,B),b] = \{1\} \times \{\emptyset\} = \emptyset$

$g[(2,A),a] = \emptyset$

$g[(2,A),b] = \{1\} \times \{A,B\} = \{(1A),(1B)\}$

	a	b
→q_0	∅	$\{q_0q_2\}$
$\{q_0q_2\}$	$\{q_3\}$	$\{q_0q_2\}$
$\{q_3\}$	∅	$\{q_0q_2\}$



$A = bB + \lambda$

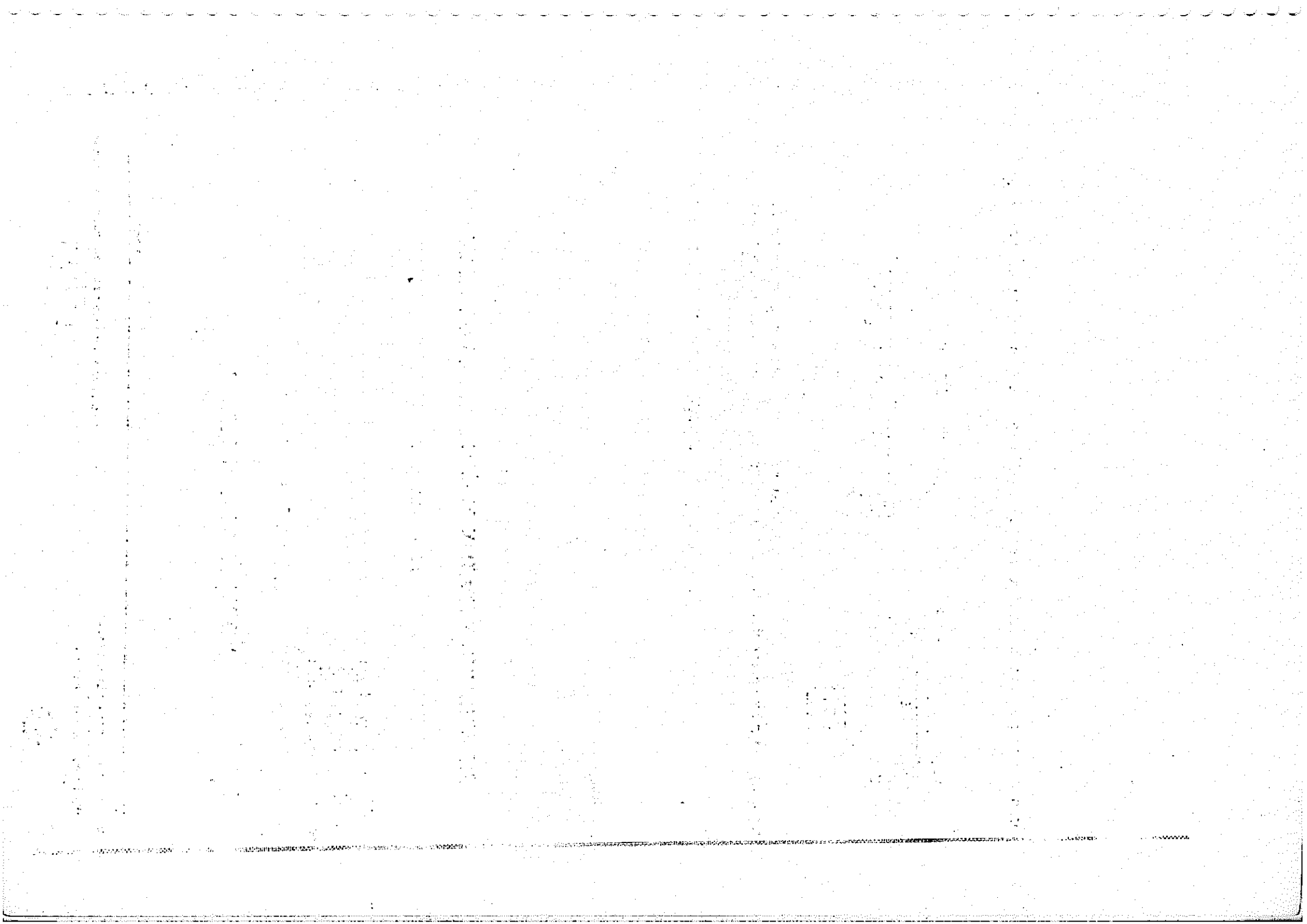
$B = bB + aC + \lambda$

$C = bB$

$B = bB + abB + \lambda$

$B = (b + ab)^*$

$A = b(b+ab)^* + \lambda = L \cap L^{-1}$



DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL

FACULTAD DE INFORMÁTICA
UNIVERSIDAD POLITÉCNICA DE MADRID

INFORMÁTICA TEÓRICA

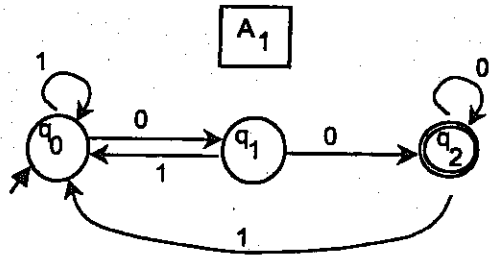
Curso 2004-2005

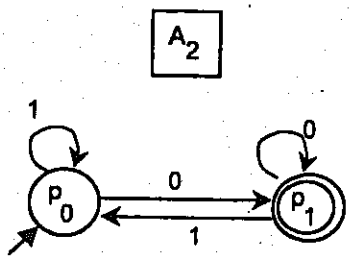
Prácticas Tema 6

PROPIEDADES DE LOS LENGUAJES REGULARES

Práctica 6.1: Propiedades de cierre

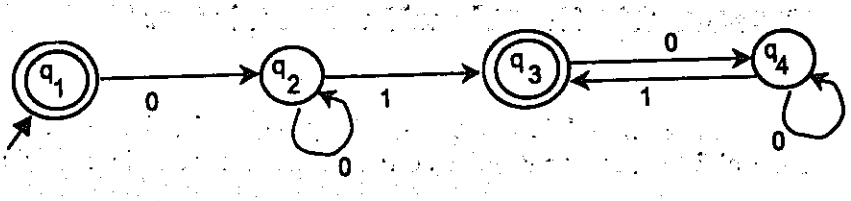
1.- a) Obtener el autómata finito determinista mínimo que reconoce el lenguaje unión de los lenguajes reconocidos por los autómatas siguientes:





b) El autómata que se obtiene es muy particular. Justificar por qué se obtiene dicho resultado.

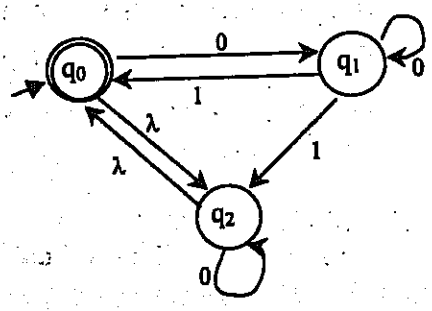
2.- Dado el autómata A



obtener una expresión regular del lenguaje $L(A)^{-1}$, inverso del reconocido por el autómata A. Este lenguaje se obtendrá utilizando tres procedimientos; en cada uno de ellos se obtendrá dicho lenguaje partiendo de:

a) la expresión regular del lenguaje L(A).
b) el autómata A.
c) una gramática lineal por la derecha que genere L(A).

3.- Dado el autómata siguiente, que reconoce el lenguaje L, obtener (a partir de este autómata) expresiones regulares para los lenguajes $\overline{L}$ y $L.\overline{L}$.



Práctica 6.2: Combinación de autómatas.

Sean $A_1 = (\Sigma, Q_1, q_{01}, f_1, F_1)$ y $A_2 = (\Sigma, Q_2, q_{02}, f_2, F_2)$ autómatas finitos deterministas. A partir de ellos definimos otro autómata de la siguiente manera:

$A = (\Sigma, Q_1 \times Q_2, (q_{01}, q_{02}), f, F)$, donde f es la función definida así:

$$f : (Q_1 \times Q_2) \times \Sigma \longrightarrow Q_1 \times Q_2$$

$$f[(q_1, q_2), e] = [f_1(q_1, e), f_2(q_2, e)]$$

Se pide:

a) No se acabó la definición anterior. ¿Quién es el conjunto de estados finales F? El lenguaje que reconoce el autómata que se ha definido dependerá de cómo se elija dicho conjunto F. ¿Qué lenguaje reconoce en los casos en que el conjunto de estados finales sea:

i) $F = \{ (q_1, q_2) \mid q_1 \in F_1 \vee q_2 \in F_2 \}$ lenguaje unión

ii) $F = \{ (q_1, q_2) \mid q_1 \in F_1 \wedge q_2 \in F_2 \}$ lenguaje intersección

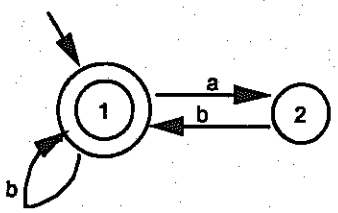
iii) $F = \{ (q_1, q_2) \mid q_1 \in F_1 \wedge q_2 \notin F_2 \}$ lenguaje diferencia

iv) $F = \{ (q_1, q_2) \mid q_1 \notin F_1 \}$

b) Aplicación: Diseñar una AFD mínimo que reconozca el lenguaje formado por los números enteros que son múltiplos simultáneamente de 3 y de 4.

c) ¿Es posible extender la definición anterior a autómatas no deterministas sin λ–transiciones?

d) Nuevo ejercicio de aplicación. Dado el autómata A, que reconoce el lenguaje L, obtener una expresión regular para el lenguaje $L \cap L^{-1}$, donde L^{-1} es el lenguaje inverso de L.



Práctica 6.3: Ambigüedad en lenguajes regulares.

1.- Dada la gramática lineal por la derecha

$S ::= aA \mid bC \mid b$

$A ::= aS \mid bB \mid a$

$B ::= bB \mid bS \mid b$

$C ::= aA \mid bC \mid b$

a) Probar que es una gramática ambigua. Para la palabra ambigua más corta encontrar dos árboles de derivación diferentes.

b) Obtener una gramática lineal por la derecha equivalente a la dada y que no sea ambigua.

2.- Sea la gramática G:

$S ::= AA$

$A ::= AAA \mid bA \mid Ab \mid a$

1) Describir el lenguaje que genera L(G).

(**Nota importante**: La respuesta correcta a este apartado es muy importante, puesto que condiciona fuertemente la solución de los siguientes apartados).

2) Probar que G es ambigua.

3) Probar que L(G) no es inherentemente ambiguo, obteniendo por medio de los algoritmos explicados en clase una gramática lineal por la derecha no ambigua, con el mínimo número de producciones posibles.

4) Obtener una expresión regular para L(G).

TEMA 7: AUTÓMATAS A PILA.

1.- Introducción.

a) Funcionamiento (movimientos).

b) Descripciones instantáneas (configuraciones).

c) Lenguajes aceptados por un AP.

d) A. Pila deterministas.

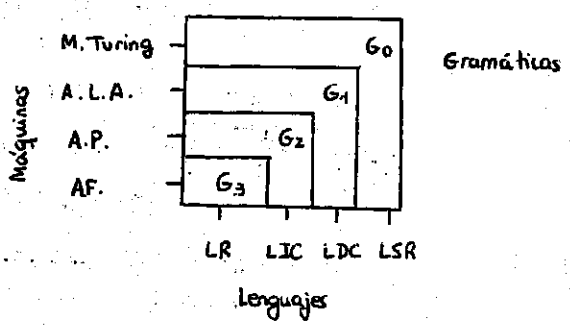
2.- A.Pila y lenguajes independientes de contexto.

a) Equivalencia aceptación de lenguajes $LV(AP) = LF(AP)$.

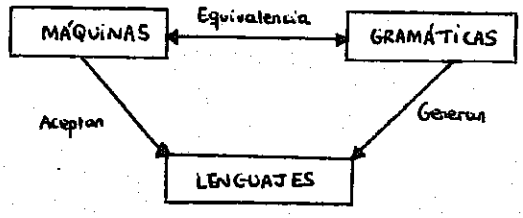
b) $LIC = L(AP) = L(G_2)$

1. INTRODUCCIÓN.

De igual manera que los lenguajes regulares pueden representarse mediante AFD, a los lenguajes independientes de contexto (G_2) les corresponde un tipo diferente de dispositivo: el autómata a pila.



Un autómata a pila es un autómata finito que, además, tiene acceso a una memoria intermedia que funciona como una pila, es decir, que el último dato que se introdujo tiene que ser necesariamente el primero en salir de ella.



$L(AF) = L(G_3) = LR$ AFD = AFND

$L(AP) = L(G_2) = LIC$ APD ⊂ APND

↓

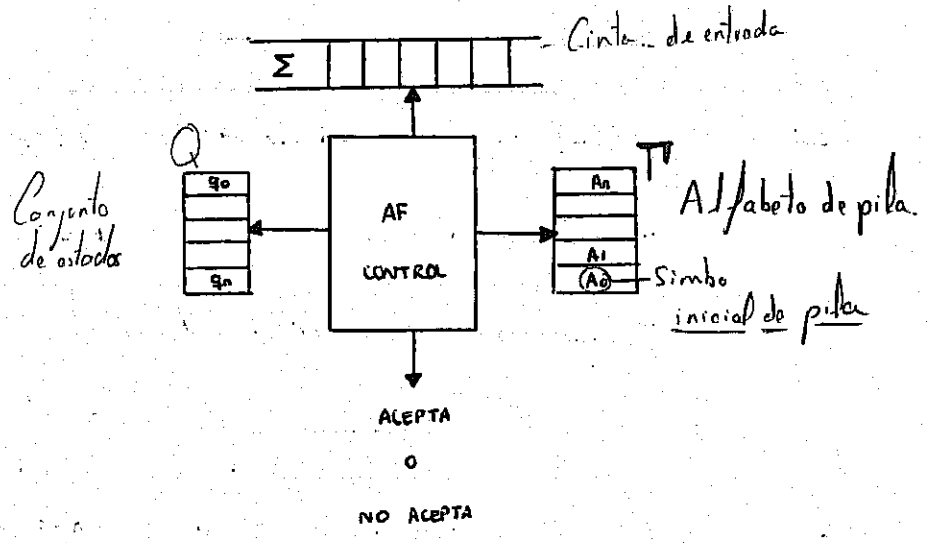
Lenguajes Independientes de Contexto

$$G_3 \subset G_2 \subset G_1 \subset G_0$$

Un autómata a pila es un dispositivo que tiene acceso a los símbolos de una cinta de entrada, al símbolo superior de la memoria en pila, y puede encontrarse en un estado determinado entre varios posibles. En cada momento, el estado siguiente depende del estado anterior, del símbolo de entrada y del símbolo superior de la pila. También le es posible cambiar de estado y modificar el contenido de la pila sin necesidad de leer un símbolo de entrada ni de avanzar la cabecera lectora de esta cinta. Además, el autómata será no determinista, en el sentido de que podrá elegir entre varias acciones posibles en cada paso.

Diremos que un autómata a pila "acepta un lenguaje por vaciado de pila" si cada una de las palabras de este lenguaje, grabada en la cinta de entrada del autómata, causa que éste, al final del proceso, se quede con la pila vacía. Por otra parte, diremos que el AP "acepta un lenguaje por estados finales" si, después de analizar la cinta de entrada donde está grabada una palabra cualquiera de dicho lenguaje, el autómata se detiene en uno de sus estados finales, mientras que queda en un estado no final con cualquier palabra que no pertenezca al lenguaje de que se trate.

DEF.- Un autómata a pila es un autómata finito que controla una memoria intermedia (pila), lee una cinta de entrada (Σ) y es susceptible de cambiar de estado.



APD

APND — Más general

$a^n b^n$; $n \geq 1$

$\exists$ AF / L(AF) = $a^n b^n$ $n \geq 1$ → FALSO

$\exists$ AP / L(AP) = $a^n b^n$ $n \geq 1$

Formas de producciones de G_2 -

$$A ::= X \text{ con } X \in \Sigma^* = (\Sigma_N \cup \Sigma_T)^*$$

Operaciones en la pila.

1.- Escribir → Introducir en la pila.
2.- Leer → Extraer de la pila.

El comportamiento de la pila puede ser:

- LIFO → Last in First Out
- FIFO → First in First Out

Definición formal

Un autómata a pila es una séptupla:

→ $AP = (\Sigma, \Gamma, Q, q_0, A_0, \delta, F)$ donde:

Σ → Alfabeto de entrada. $a, b, c \in \Sigma$ $x, y, z \in \Sigma^*$

Γ → Alfabeto de la pila. $A, B, C \in \Gamma$ $X, Y, Z \in \Gamma^*$ (Mayúsculas) Puede tener elementos en común con Σ.

Q → Conjunto finito de estados.

q_0 → Estado inicial del autómata. $q_0 \in Q$

A_0 → Símbolo inicial de la pila $A_0 \in \Gamma$

δ → Función de transición del AP:

$$\delta : (Q \times (\Sigma \cup \{\lambda\}) \times \Gamma) \rightarrow P(Q \times \Gamma^*)$$

↑ partes

Por cada estado, símbolo de entrada o palabra vacía, y símbolo en la cima de la pila, determina la transición a otro estado y desde qué se debe escribir en la pila.

F → Conjunto de estados finales. $F \subset Q$

$F = \emptyset \Rightarrow$ AP por vaciado de pila

$F \neq \emptyset \Rightarrow$ AP por estados finales

AP_F => El AP se para cuando alcanza un estado final.
Acepta un lenguaje por estados finales.

AP_V => El AP se para cuando borra A_0 (queda la pila vacía).
Acepta un lenguaje por vaciado de pila.

a) FUNCIONAMIENTO (MOVIMIENTOS):

La función de transición es: $\delta : (Q \times (\Sigma \cup \{\lambda\}) \times \Gamma) \to P(Q \times \Gamma^*)$

Un AP tendrá dos tipos de movimientos:

GENÉRICOS

1) TIPO 1.- $\delta(q, a, A) = \{\{q_1, Z_1\}, \{q_2, Z_2\} \dots \{q_n, Z_n\}\}$

$q, q_1, q_2, \dots, q_n \in Q$

$a \in \Sigma$

$A \in \Gamma \qquad Z_1, Z_2, \dots, Z_n \in \Gamma^*$

Si el AP se encontraba inicialmente en el estado q, leyó el símbolo "a" en la entrada y disponía del símbolo A en lo alto de la pila, pasará en el instante siguiente al estado q_i $(1 \le i \le n)$, borrará el símbolo A de lo alto de la pila, introducirá en ésta la palabra Z_i y avanzará una posición la cinta de entrada. La cabeza de Z_i pasa a ser el nuevo símbolo superior de la pila.
Una vez elegido el valor de i, el valor q_i y Z_i queda automáticamente asignado.

λ-MOVIMIENTOS

2) TIPO 2.- $\delta(q, \lambda, A) = \{\{q_1, Z_1\}, \{q_2, Z_2\}, \dots, \{q_n, Z_n\}\}$

Si el AP se encontraba inicialmente en el estado q y disponía del símbolo A en lo alto de la pila, pasará en el instante siguiente al estado q_i $(1 \le i \le n)$, borrará el símbolo A de lo alto de la pila, introducirá en ella la palabra Z_i y dejará la cinta de entrada en la misma posición que inicialmente (no avanza). La cabeza de Z_i pasa a ser el nuevo símbolo superior de la pila.

EJ: $L = \{ 1^n 0^n \ / \ n \geq 1 \}$

$AP_V = \{ \{1,0\}, \{A_0\}, Q, q_0, A_0, \delta, \phi \}$

Σ $\quad$ Γ $\quad$ F

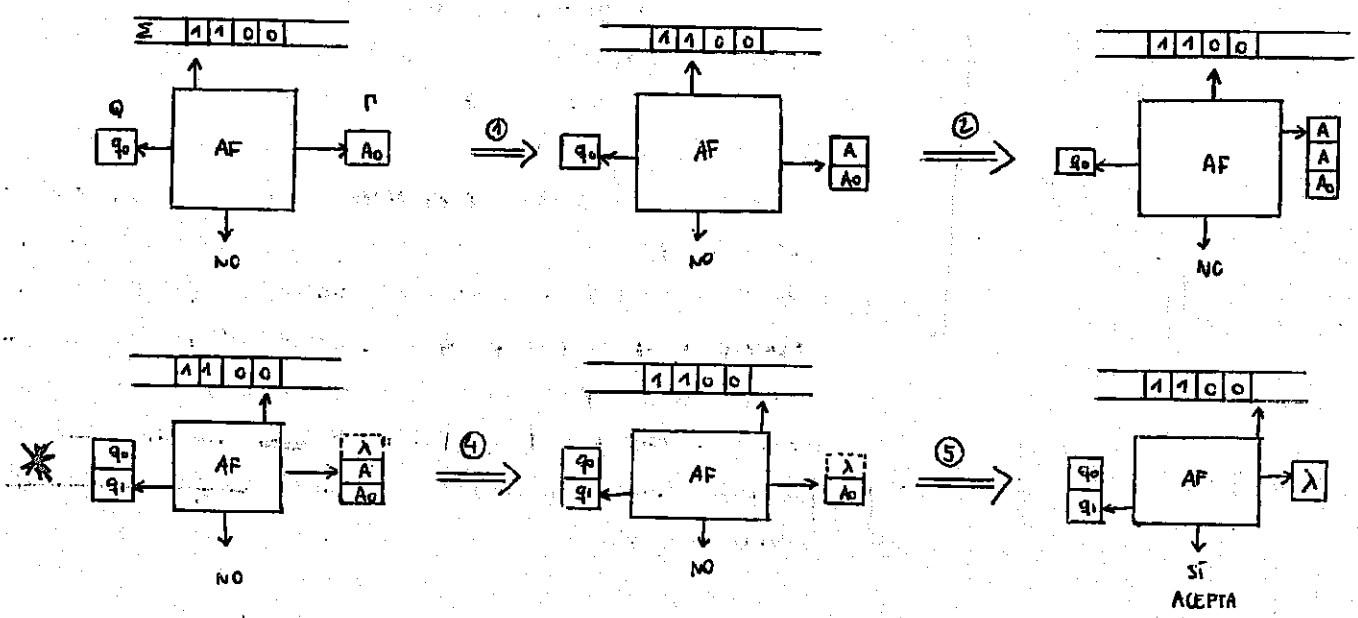
① $\delta(q_0, 1, A_0) = (q_0, \underbrace{AA_0}_{z})$ Lee "1" y mete "A" en la pila.

② $\delta(q_0, 1, A) = (q_0, AA)$ Lee "1" y mete "A" en la pila (Resto de 1).

⚠ → ③ $\delta(q_0, 0, A) = (q_1, \lambda)$ ✱ Lee "0" y borra "A" de la pila

④ $\delta(q_1, 0, A) = (q_1, \lambda)$ " " " " " (Resto de 0)

⑤ $\delta(q_1, \lambda, A_0) = (q_1, \lambda)$ Borra A_0. Se para. Acepta.



b) DESCRIPCIONES INSTANTÁNEAS (CONFIGURACIONES):

Podemos describir el proceso de aceptación o rechazo de una palabra de Σ^* mediante una sucesión de "descripciones instantáneas".

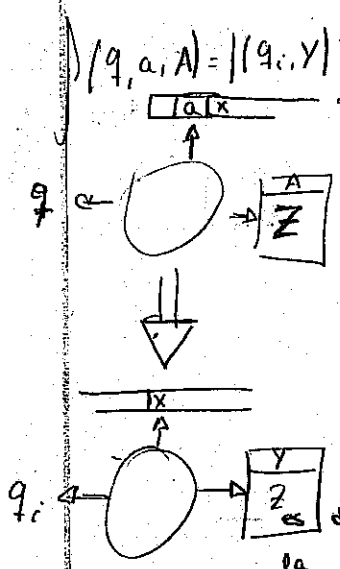
Una descripción instantánea es la terna:

$$(q, x, Z) \quad \text{donde:}$$

- q → Estado actual del AP.
- x → Palabra o subpalabra de la entrada que queda por leer.
- Z → Contenido de la pila en ese momento.

* Representación de movimientos.-

Se dice que una descripción instantánea (q, ax, AZ) precede a otra $(\bar{q}_i, x, YZ)$, y se representa de la siguiente forma, si cumple:



$$\begin{cases} (q, ax, AZ) \overset{\text{precede}}{\vdash} (q_i, x, YZ) & \text{si } (q_i, Y) \in \delta(q, a, A) \quad \text{(TIPO 1)} \\ (q, ax, AZ) \vdash (q_i, ax, YZ) & \text{si } (q_i, Y) \in \delta(q, \lambda, A) \quad \text{(TIPO 2)} \\ & \quad \text{"} \\ & \quad x \text{ pq } \lambda = a \\ q, q_i \in Q \qquad a \in \Sigma \qquad x \in \Sigma^* \qquad A \in \Gamma \qquad Y, Z \in \Gamma^* & \end{cases}$$

Utilizaremos el símbolo $\vdash^*$ para el cierre de la relación $\vdash$. Es decir, si I_i es una descripción instantánea de la aceptación o rechazo de la palabra (x), se verifica:

$$I_i \vdash^* I_j \rightarrow I_i \vdash I_k \vdash \dots \vdash I_j$$

EJ: ¿1100 ∈ L?

$$(q_0, 1100, A_0) \overset{①}{\vdash} (q_0, 100, AA_0) \overset{②}{\vdash} (q_0, 00, AAA_0) \overset{③}{\vdash} (q_1, 0, AA_0)$$
$$\overset{④}{\vdash} (q, \lambda, A_0) \overset{⑤}{\vdash} (q_f, \lambda, \lambda)$$

Acepta x. Por lo tanto, $x = 1100 \in L$

EJ: $L = \{ a^n b^m c^m d^n \ / \ (n \geq 2,\ m \geq 1) \}$ → Creo por necesitamos cambiar de estado

Palabra mínima: aabcdd

$AP_v = \{a,b,c,d\}, \{A_0, A, B\}, \{q_0, q_1, q_2\}, q_0, A_0, f, \phi$

aaa

A_0
AAA_0
$AAAA_0$

Definición de f:

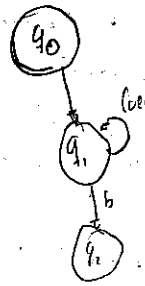
① $f(q_0, a, A_0) = (q_0, AA_0)$ Lee "a" en la cinta de entrada y mete "A" en la pila.

② $f(q_0, a, A) = (q_1, AA)$ Lee la 2ª "a" y mete "A" en la pila.

③ $f(q_1, a, A) = (q_1, AA)$ Lee "a" y mete "A" en la pila (Resto "a")

④ $f(q_1, b, A) = (q_1, BA)$ Lee "b" y mete "B" en la pila.

⑤ $f(q_1, b, B) = (q_1, BB)$ Lee "b" y mete "B" en la pila (Resto "b")



⑥ $f(q_1, c, B) = (q_2, \lambda)$ Lee "c" y borra "B" de la pila.

⑦ $f(q_2, c, B) = (q_2, \lambda)$ " " " " (Resto "c")

⑧ $f(q_2, d, A) = (q_2, \lambda)$ Lee "d" y borra "A" (Todos)

⑨ $f(q_2, \lambda, A_0) = (q_2, \lambda)$ Borra A_0 (Acepta)

Descripciones instantáneas de las palabras:

1) aabcdd

$(q_0, aabcdd, A_0) \overset{①}{\vdash} (q_0, abcdd, AA_0) \overset{②}{\vdash} (q_1, bcdd, AAA_0) \overset{④}{\vdash}$
$(q_1, cdd, BAAA_0) \overset{⑤}{\vdash} (q_2, dd, AAA_0) \overset{⑦}{\vdash} (q_2, d, AA_0) \overset{⑧}{\vdash}$
$(q_2, \lambda, AA_0) \overset{⑨}{\vdash} (q_2, \lambda, \lambda)$ ACEPTA.

↳ $(q_0, aabcdd, A_0) \overset{*}{\vdash} (q_2, \lambda, \lambda)$ ↓ sucesión de movimientos

2) abcd

$(q_0, abcd, A_0) \overset{①}{\vdash} (q_0, bcd, AA_0)$ NO ACEPTA.

EJ: El autómata $(\{0,1\}, \{A,0,1\}, \{p,q\}, A, p, f, \emptyset)$ donde f se define así:

$f(p,0,A) = \{(p,0A)\}$
$f(p,q,0) = \{(p,00),(q,\lambda)\}$
$f(p,0,1) = \{(p,01)\}$
$f(p,1,A) = \{(p,1A)\}$
$f(p,1,0) = \{(p,10)\}$
$f(p,1,1) = \{(p,11),(q,\lambda)\}$
$f(q,0,0) = \{(q,\lambda)\}$
$f(q,1,1) = \{(q,\lambda)\}$
$f(q,\lambda,A) = \{(q,\lambda)\}$
$f(q,\lambda,A) = \{(q,\lambda)\}$

Sucesión de descripciones instantáneas cuando se procesa la palabra de entrada 1001:

$$(p,1001,A) \vdash (p,001,1A) \vdash (p,01,01A) \vdash (q,1,1A) \vdash (q,\lambda,A) \vdash (q,\lambda,\lambda)$$

Al ser un autómata no determinista, hay otras series de descripciones instantáneas posibles. Sin embargo, sólo ésta termina con la pila vacía. Basta que exista una derivación así para que la palabra sea aceptada por el AP.

c) LENGUAJES ACEPTADOS POR UN AP:

i) ESTADOS FINALES: Llamaremos "lenguaje aceptado por estado final" al conjunto:

$$L = \{ x / (q_0, x, A_0) \vdash^* (p, \lambda, X) \} \quad p \in F \text{ (estado final)}; \ X \in \Gamma^*$$

$AP_F = \{\Sigma, \Gamma, Q, q_0, A_0, f, F\}$ $\quad q_0 \in Q \quad F \subset Q$

ii) VACIADO DE PILA: Llamaremos "lenguaje aceptado por vaciado de pila" al conjunto:

$$L = \{ x / (q_0, x, A_0) \vdash^* (p, \lambda, \lambda) \} \quad q_0, p \in Q$$

$AP_V = \{\Sigma, \Gamma, Q, q_0, A_0, f, \emptyset\}$

El conjunto de lenguajes aceptados por estado final por los AP_F es igual al conjunto de los lenguajes aceptados por vaciado de pila por los AP_V

d) AUTÓMATAS PILA DETERMINISTAS:

... Hasta ahora hemos estado viendo el no determinista.

Un Autómata Pila

$\boxed{AP} = (\Sigma, \Gamma, Q, q_0, A_0, \delta, F)$

es determinista si se cumplen dos condiciones:

① $\forall q \in Q \ , \ \forall A \in \Gamma$

$$|\delta(q, \lambda, A)| > 0 \Rightarrow \delta(q, a, A) = \emptyset$$

↑ cardinal

Previene la elección de movimientos $\begin{cases} \delta(q, \lambda, A) = (q_c, X) \\ \delta(q, a, A) = (q_i, Y) \end{cases}$

② $\forall q \in Q \ , \ \forall A \in \Gamma \ , \ \forall a \in \Sigma \cup \{\lambda\}$

$$|\delta(q, a, A)| < 2$$

Previene de la elección de más de un movimiento:

$$\delta(q, a, A) \neq \{(q_1, z_1), (q_2, z_2), \ldots, (q_n, z_n)\}$$

> Siempre que leemos un símbolo se hace un movimiento.

EJ: $L = \{ xcx^{-1} \ / \ x \in (0,1)^* \}$

↑ es una letra "c", no es "contenido"

$AP_V = (\underbrace{\{0,1,c\}}_{\Sigma}, \Gamma, Q, q_0, A_0, \delta, \emptyset)$

① $\delta(q_0, 0, A_0) = (q_0, AA_0)$ Lee 0 y mete A

② $\delta(q_0, 1, A_0) = (q_0, BA_0)$ Lee 1 y mete B

③ $\delta(q_0, 0, A) = (q_0, AA)$ Lee 0 y mete A

④ $\delta(q_0, 1, A) = (q_0, BA)$ Lee 1 y mete B

} Tras 0

⑤ $\delta(q_0, c, B) = (q_0, AB)$ Lee 0 y mete A

⑥ $\delta(q_0, 1, B) = (q_0, BB)$ Lee 1 y mete B

} Tras 1

⑦ $\delta(q_0, c, A) = (q_1, A)$

⑧ $\delta(q_0, c, B) = (q_1, B)$

⑨ $\delta(q_0, c, A_0) = (q_1, \lambda)$

⑩ $\delta(q_1, 0, A) = (q_1, \lambda)$

⑪ $\delta(q_1, 1, B) = (q_1, \lambda)$

⑫ $\delta(q_1, \lambda, A_0) = (q_1, \lambda)$

¿ 001c100 $\in L$?

$(q_0, 001c100, A_0) \overset{①}{\vdash} (q_0, 01c100, AA_0) \overset{③}{\vdash} (q_0, 1c100, AAA_0) \overset{④}{\vdash}$
$(q_0, c100, BAAA_0) \overset{⑥}{\vdash} (q_1, 100, BAAA_0) \overset{⑭}{\vdash} (q_1, 00, AAA_0) \overset{⑩}{\vdash}$
$(q_1, 0, AA_0) \overset{⑩}{\vdash} (q_1, \lambda, A_0) \overset{⑪}{\vdash} (q_1, \lambda, \lambda)$

ACEPTADA.

EJ: $L = \{ ww^{-1} / w \in (0,1)^* \}$ $\quad AP_V = (\{0,1\}, \Gamma, \delta, q_0, A_0, \delta, \emptyset)$

① $\delta(q_0, 0, A_0) = (q_0, AA_0)$

② $\delta(q_0, 1, A_0) = (q_0, BA_0)$

③ $\delta(q_0, 0, A) = (q_0, AA)$ ó (q_1, λ)

④ $\delta(q_0, 1, A) = (q_0, BA)$

⑤ $\delta(q_0, 0, B) = (q_0, AB)$

⑥ $\delta(q_0, 1, B) = (q_0, BB)$ ó (q_1, λ)

⑦ $\delta(q_1, 0, A) = (q_1, \lambda)$

⑧ $\delta(q_1, 1, B) = (q_1, \lambda)$

⑨ $\delta(q_1, \lambda, A_0) = (q_1, \lambda)$

⑩ $\delta(q_0, \lambda, A_0) = (q_1, \lambda)$

¿ $\underbrace{110}_{w}\underbrace{011}_{w^{-1}}$ $\in L$?

$(q_0, 110011, A_0) \overset{②}{\vdash} (q_0, 10011, BA_0) \overset{⑥}{\vdash} (q_0, 0011, BBA_0) \overset{⑤}{\vdash} (q_0, 011, ABBA_0) \overset{④}{\vdash}$
T⑥
$(q_1, 0011, A_0)$
T⑨
$(q_1, 0011, \lambda)$
T③

NO

$\dots (q_0, 011, ABBA_0) \overset{⑧}{\vdash} (q_1, 1, BA_0) \overset{⑥}{\vdash} (q_1, \lambda, A_0) \overset{⑨}{\vdash} (q, \lambda, \lambda)$ SÍ

$\top ③$

$(q_0, 11, AABBA_0)$

$\top ④$

$(q_0, 1, BAABBA_0) \overset{⑥}{\vdash} (q_1, \lambda, AABBA_0)$ NO

$\top ⑤$

$(q_0, \lambda, BBAABBA_0)$

NO

EJ: $L = \{ x \in (a,b)^* \ / \ N_a(x) = N_b(x) \}$

$AP_V = (\{a,b\}, \Gamma, \{q_0\}, q_0, A_0, f, \phi)$

① $f(q_0, a, A_0) = (q_0, AA_0)$

② $f(q_0, b, A_0) = (q_0, BA_0)$

③ $f(q_0, a, A) = (q_0, AA)$

④ $f(q_0, b, A) = (q_0, \lambda)$

⑤ $f(q_0, a, B) = (q_0, \lambda)$

⑥ $f(q_0, b, B) = (q_0, BB)$

⑦ $f(q_0, \lambda, A_0) = (q_0, \lambda)$

2. A. PILA Y LENGUAJES INDEPENDIENTES DE CONTEXTO.

a) EQUIVALENCIA ACEPTACIÓN DE LENGUAJES:

□ conjunto de lenguajes aceptados por estado final por los autómatas a pila es igual al conjunto de lenguajes aceptados por vaciado de pila:

$$L(AP_V) = L(AP_F)$$

$$LV(AP) = LF(AP)$$

$$\left.\begin{array}{l} a)\ L(AP_V) \subset L(AP_F) \\ b)\ L(AP_V) \supset L(AP_F) \end{array}\right\} \ L(AP_V) = L(AP_F)$$

$M_1 = (\Sigma, M, Q, q_0, A_0, \delta, \phi)$ es un AP por vaciado de pila.

$M_2 = (\Sigma, \Gamma \cup \{A_0'\}, Q \cup \{q_0', q_F\}, q_0', A_0', \delta', \{q_F\})$ es un AP por estados finales

M_1 es AP $L(M_1) = L \Rightarrow \exists\ M_2 \ / \ L(M_2) = L$

a) $L(AP_V) \subset L(AP_F)$. La función de transición de M_2, δ', se define así:

① $\delta'(q_0', \lambda, A_0') = (q_0, A_0 A_0')$

Permite a M_2 acceder a la descripción instantánea inicial de M_1. M_2 puede empezar a simular M_1.

② $\delta'(q, a, A) = \delta(q, a, A)$

Permite a M_2 realizar los mismos movimientos que M_1.

Con ① y ②, M_2 simula M_1. $\quad \forall q \in Q\ ;\ \forall a \in \Sigma \cup \{\lambda\}\ ;\ \forall A \in \Gamma$

③ $\delta'(q, \lambda, A_0) = (q_F, \lambda) \quad \forall q \in Q$

λ- movimiento que borra A_0^* y alcanza estado final q_F

DIFERENCIAS:

* M_2 tiene en fondo de pila A_0
* M_1, cuando acepta $x \in L$, borra A_0.
* M_2, cuando acepta $x \in L$, aflora A_0' a la cima de la pila.

EJ: $L = \{ a^n b^n \ / \ n \geq 3 \}$ $\lambda \notin L$ aaabbb

$AP_1 = (\{a,b\}, \{A_0\}, Q, q_0, A_0, \delta, \phi)$

1) $\delta(q_0\ a\ A_0) = (q_1\ AA_0)$
2) $\delta(q_1\ a\ A) = (q_2\ AA)$
3) $\delta(q_2\ a\ A) = (q_3\ AA)$

} $n=3$

4) $\delta(q_3\ a\ A) = (q_3\ AA)$ $n>3$
5) $\delta(q_3\ b\ A) = (q_4\ \lambda)$
6) $\delta(q_4\ b\ A) = (q_4\ \lambda)$ Borra todas las A con b
7) $\delta(q_4\ \lambda\ A_0) = (q_4\ \lambda)$

$(q_0\ aaabbb\ A_0) \vdash^{①} (q_1\ aabbb\ AA_0) \vdash^{②} (q_2\ abbb\ AAA_0) \vdash^{③} (q_3\ bbb\ AAAA_0) \vdash$

$\vdash^{⑤} (q_4\ bb\ AAA_0) \vdash^{⑥} (q_4\ b\ AA_0) \vdash^{⑥} (q_4\ \lambda\ A_0) \vdash^{⑦} (q_4\ \lambda\ \lambda)$ aaabbb $\in L$

aabb $\notin L$: $(q_0\ aabb\ A_0) \vdash^{①} (q_1\ abb\ AA_0) \vdash^{②} (q_2\ bb\ AAA_0)$ // NO

(y es que no puedo seguir).

→ Se puede construir con un estado menos:

1) $\delta(q_0\ a\ A_0) = (q_0\ AA_0)$
2) $\delta(q_0\ a\ A) = (q_1\ AA)$
3) $\delta(q_1\ a\ A) = (q_2\ AA)$
4) $\delta(q_2\ a\ A) = (q_2\ AA)$
5) $\delta(q_2\ b\ A) = (q_3\ \lambda)$
6) $\delta(q_3\ b\ A) = (q_3\ \lambda)$
7) $\delta(q_3\ \lambda\ A_0) = (q_3\ \lambda)$

(Sobre el autómata anterior simplificado). Construimos el AP de estados finales.

$AP_F = (\{a, b\}, \{A, A_0, A_0'\}, \{q_0, q_1, q_2, q_3, q_0', q_4\}, q_0', A_0', \delta', \{q_4\})$

① 1) $\delta'(q_0' \ \lambda \ A_0') = (q_0 \ A_0 A_0')$

② 2) $\delta'(q_0 \ a \ A_0) = (q_0 \ A A_0)$
3) $\delta'(q_0 \ a \ A) = (q_1 \ AA)$
4) $\delta'(q_1 \ a \ A) = (q_2 \ AA)$
5) $\delta'(q_2 \ a \ A) = (q_2 \ AA)$
6) $\delta'(q_2 \ b \ A) = (q_3 \ \lambda)$
7) $\delta'(q_3 \ b \ A) = (q_3 \ \lambda)$
8) $\delta'(q_3 \ \lambda \ A_0) = (q_3 \ \lambda)$

③ 9) $\delta'(q_3 \ \lambda \ A_0') = (q_4 \ \lambda)$ Acceder a estado final.

aaabbb

no avanza (leyó λ)

$(q_0' \ aaabbb \ A_0') \overset{①}{\vdash} (q_0 \ aaabbb \ A_0 A_0') \overset{②}{\vdash} (q_0 \ aabbb \ AA_0 A_0') \overset{③}{\vdash} (q_1 \ abbb \ AAA_0 A_0')$

$\overset{④}{\vdash} (q_2 \ bbb \ AAAA_0 A_0') \overset{⑥}{\vdash} (q_3 \ bb \ AAA_0 A_0') \overset{⑦}{\vdash} (q_3 \ b \ AA_0 A_0') \overset{⑧}{\vdash} (q_3 \ \lambda \ A_0 A_0')$

$\overset{⑨}{\vdash} (q_3 \ \lambda \ A_0') \overset{⑨}{\vdash} (q_4 \ \lambda \ \lambda)$

estado final.

b) $L_V(AP) \supset L_F(AP)$

$L = L_F(AP) \Rightarrow \exists AP \;/\; L_V(AP) = L$

- $A_1 = (\Sigma, \Gamma, Q, q_0, A_0, \delta, F) \longrightarrow AP_F$
- $A_2 = (\Sigma, \Gamma \cup \{A_0'\}, Q \cup \{q_0', q_V\}, q_0', A_0', \delta', \phi) \longrightarrow AP_V$

δ' se define así:

1. $\delta'(q_0', \lambda, A_0') = (q_0, A_0 A_0')$

   A_2 accede a la descripción instantánea inicial de A_1.
   A_2 comienza a simular A_1

2. $\delta'(q, a, A) = \delta(q, a, A) \quad \forall q \in Q \;;\; \forall a \in \Sigma \cup \{\lambda\} \;,\; \forall A \in \Gamma$

   A_2 realiza los mismos movimientos que A_1.

   A_2 simula A_1 <
   - A_0' símbolo inicial de la pila
   - q_V estado vaciado.

   A_2 ha vaciado la pila aún

3. a) $\delta(q_F, \lambda, B) = (q_V, \lambda)$ A_2 accede a q_F y pasa a q_V.

   b) $\delta(q_V, \lambda, B) = (q_V, \lambda)$ Vaciar la pila (símbolo a símbolo).

EJ: $AP_F = (\underbrace{\{a,b\}}_{\Sigma}, \underbrace{\{A_1, A_0\}}_{\Gamma}, \underbrace{\{q_0, q_1, q_2\}}_{Q}, q_0, A_0, \delta, \underbrace{\{q_2\}}_{F})$

1) $\delta(q_0, a, A_0) = (q_0, AA_0)$ Lee 'a' y mete 'A'.
2) $\delta(q_0, a, A) = (q_0, AA)$ Lee 'a' y mete 'A'.
3) $\delta(q_0, b, A) = (q_1, \lambda)$ Lee 'b' y borra 'A'.
4) $\delta(q_1, b, A) = (q_1, \lambda)$ Lee 'b' y borra 'A'.
5) $\delta(q_1, \lambda, A_0) = (q_2, AA_0)$ Alcanza q_2 y se para.

$L = \{ a^n b^n \ / \ n \geq 1 \}$ Palabra mínima → ab

$(q_0 \ ab \ A_0) \overset{①}{\vdash} (q_0 \ b \ AA_0) \overset{③}{\vdash} (q_1 \ \lambda \ A_0) \overset{⑤}{\vdash} (q_2 \ \lambda \ AA_0)$

$APv = (\{a,b\}, \{A, A_0, A_0'\}, \{q_0, q_1, q_2, q_3\}, q_0', A_0', \delta', \emptyset)$

1) $\delta'(q_0' \ \lambda \ A_0') = (q_0 \ A_0 A_0')$
2) $\delta'(q_0 \ a \ A_0) = (q_0 \ AA_0)$
3) $\delta'(q_0 \ a \ A) = (q_0 \ AA)$
4) $\delta'(q_0 \ b \ A) = (q_1 \ \lambda)$
5) $\delta'(q_1 \ b \ A) = (q_1 \ \lambda)$
6) $\delta'(q_1 \ \lambda \ A_0) = (q_2 \ AA_0)$
7) $\delta'(q_2 \ \lambda \ A) = (q_3 \ \lambda)$
8) $\delta'(q_3 \ \lambda \ A_0) = (q_3 \ \lambda)$
9) $\delta'(q_3 \ \lambda \ A_0') = (q_3 \ \lambda)$

$(q_0' \ ab \ A_0') \overset{①}{\vdash} (q_0 \ ab \ A_0 A_0') \overset{②}{\vdash} (q_0 \ b \ AA_0 A_0') \overset{④}{\vdash} (q_1 \ \lambda \ A_0 A_0') \overset{⑤}{\vdash} (q_2 \ \lambda \ AA_0 A_0')$

$\overset{⑦}{\vdash} (q_3 \ \lambda \ A_0 A_0') \overset{⑧}{\vdash} (q_3 \ \lambda \ A_0') \overset{⑨}{\vdash} (q_3 \ \lambda \ \lambda)$

$(q_0' \ ab \ A_0') \vdash_v (q_3 \ \lambda \ \lambda)$

EJ: $L = \{ 0^n 1^m \ / \ n \geq m \geq 1 \}$

$APv = (\{0,1\}, \{A, A_0\}, \{q_0, q_1\}, q_0, A_0, \delta, \emptyset)$ Palabra mínima → 01.

1) $\delta(q_0 \ 0 \ A_0) = (q_0 \ AA_0)$
2) $\delta(q_0 \ 0 \ A) = (q_0 \ AA)$
3) $\delta(q_0 \ 1 \ A) = (q_1 \ \lambda)$
4) $\delta(q_1 \ 1 \ A) = (q_1 \ \lambda)$ — $n = m$
5) $\delta(q_1 \ \lambda \ A) = (q_1 \ \lambda)$
6) $\delta(q_1 \ \lambda \ A_0) = (q_1 \ \lambda)$ — $n > m$

¿ $001 \in L$?

$(q_0 \ 001 \ A_0) \overset{①}{\vdash} (q_0 \ 01 \ AA_0) \overset{②}{\vdash} (q_0 \ 1 \ AAA_0) \overset{③}{\vdash} (q_1 \ \lambda \ AA_0) \overset{⑤}{\vdash} (q_1 \ \lambda \ A_0) \overset{⑥}{\vdash} (q_1 \ \lambda \ \lambda)$

ACEPTA

b) LIC = L(AP) = L(G2)

↳ Lenguajes Independientes de contexto.

L(AP) = L(G2)

1) L(AP) ⊂ L(G2)

L = L(AP) → ∃ L(G2) = L

AP ⟶ G2 A partir del autómata AP construiremos la G2.

$AP_V = (\Sigma, \Gamma, Q, q_0, A_0, \delta, \phi)$

$G_2 = (\Sigma_T, \Sigma_N, P, S)$ donde:

- $\Sigma_T = \Sigma$ (entrada AP)
- $\Sigma_N = \{S\} \cup \{(q, A, p) \mid q, p \in Q, A \in \Gamma\}$
- S = axioma
- P = Se describen a continuación.

P =

- $S ::= (q_0\ A_0\ p) \quad \forall p \in Q$ ✓

  Ej: $Q = \{q_0, q_1, q_2\}$ →
  $S ::= (q_0\ A_0\ q_0) \mid (q_0\ A_0\ q_1) \mid (q_0\ A_0\ q_2)$

- $(q\ A\ q_{m+1}) ::= a\,.\,(q_1\ B_1\ q_2).(q_2\ B_2\ q_3)\ldots(q_m\ B_m\ q_{m+1})$
  ($a \in \Sigma_T$)

  Cada terna refleja un símbolo que se introduce en la pila.

  Si $(q_1 B_1 B_2 \ldots B_m) \in \delta(q\ a\ A)$ $\forall q, q_1, q_2, \ldots, q_{m+1} \in Q$
  $\forall B_1, B_2, \ldots, B_n \in \Gamma$

- Si m=0 → no se introduce nada en la pila.
  Si $(q_i\ \lambda) \in \delta(q\ a\ A)$ $\forall q_i \in Q$ $(q\ A\ q_i) ::= a$
  ($q \in \Sigma_N$, $a \in \Sigma_T$)

→ **TERNAS**.- (q, A, p)

q => Estado actual del AP.

A => Símbolo de la pila.

p => Estado al que accede el AP.

AP	G_2
Borra símbolo A	$(\ A\ ) ::=$
Introduce $B_1 \dots B_m$	$::= (\ B_1\ )(\ B_2\ ) \dots$
Lee a	Genera $a \in \Sigma_T$

EJ: $L = \{ 0^n 1^m \ / \ n \geq m \geq 1 \}$

$AP_V = (\{0,1\}, \{A_1, A_0\}, \{q_0, q_1\}, q_0, A_0, \delta, \phi)$

$$\left.\begin{array}{l} \delta(q_0, 0, A_0) = (q_0\ A A_0) \\ \delta(q_0, 0, A) = (q_0\ AA) \\ \delta(q_0, 1, A) = (q_1\ \lambda) \\ \delta(q_1, 1, A) = (q_1\ \lambda) \end{array}\right\} m = n$$

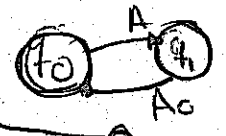
$$\left.\begin{array}{l} \delta(q_1, \lambda, A) = (q_1\ \lambda) \\ \delta(q_1, \lambda, A_0) = (q_1\ \lambda) \end{array}\right\} m > n$$

¿NO TIENE NINGÚN PUTO SENTIDO!

$G_2 = (\{0,1\}, \Sigma_N, P, S)$

P:

① $S ::= \underbrace{(q_0\ A_0\ q_0)}_{A} \mid \underbrace{(q_0\ A_0\ q_1)}_{B} \quad \forall p \in Q$



② 1.- $\underbrace{(q_0\ A_0\ q_0)}_{A} ::= 0\,\underbrace{(q_0\ A\ q_0)}_{C}\,\underbrace{(q_0\ A_0\ q_0)}_{A} \mid 0\,\underbrace{(q_0\ A\ q_1)}_{D}\,\underbrace{(q_1\ A_0\ q_0)}_{G}$

$\underbrace{(q_0\ A_0\ q_1)}_{B} ::= 0\,\underbrace{(q_0\ A\ q_0)}_{C}\,\underbrace{(q_0\ A_0\ q_1)}_{B} \mid 0\,\underbrace{(q_0\ A\ q_1)}_{D}\,\underbrace{(q_1\ A_0\ q_1)}_{F}$

2.- $\underbrace{(q_0\ A\ q_0)}_{C} ::= 0\,\underbrace{(q_0\ A\ q_0)}_{C}\,\underbrace{(q_0\ A\ q_0)}_{C} \mid 0\,\underbrace{(q_0\ A\ q_1)}_{D}\,\underbrace{(q_1\ A\ q_0)}_{H}$

$\underbrace{(q_0\ A\ q_1)}_{D} ::= 0\,\underbrace{(q_0\ A\ q_0)}_{C}\,\underbrace{(q_0\ A\ q_1)}_{D} \mid 0\,\underbrace{(q_0\ A\ q_1)}_{D}\,\underbrace{(q_1\ A\ q_1)}_{E}$

③ $(q_0\ A\ q_1) ::= 1$ — D

④ $(q_1\ A\ q_1) ::= 1$ — E

⑤ $(q_1\ A\ q_1) ::= \lambda$ — E

⑥ $(q_1\ A_0\ q_1) ::= \lambda$ — F

P:
S ::= A | B
~~A ::= 0CA | CDG~~
B ::= ~~0BB~~ | 0D1 = λ
~~C ::= 0CC | 0D11~~
D ::= ~~0DD~~ | 0DE | 1
E ::= 1 | λ | 0D1 | 0D
~~F ::= 0~~

⇒

S ::= 0D*
D ::= 0D1 | 0D | 1

Comprobamos 0011 : S → 0D → 00D1 → 000D1 → 0011

* Método simplificado.-

$[q_0\ A_0\ q_0]$ ✓ $[q_0\ A_0\ q_1]$ ✓ $[q_1\ A_0\ q_0]$ ✓ $[q_1\ A_0\ q_1]$ ✓
$[q_0\ A\ q_0]$ $[q_0\ A\ q_1]$ ✓ $[q_1\ A\ q_0]$ ✓ $[q_1\ A\ q_1]$ ✓ Tacho las no válidas.

$S ::= [q_0\ A\ q_0] \mid [q_0\ A_0\ q_1]$ (A)

① $[q_0\ A_0\ q_0] ::=$

$[q_0\ A_0\ q_1] ::= 0[q_0\ A\ q_0][q_0\ A_0\ q_1] \mid 0[q_0\ A\ q_1][q_1\ A_0\ q_1]$ (B, D)

② $[q_0\ A\ q_0] ::=$

$[q_0\ A\ q_1] ::= 0[q_0\ A\ q_0][q_0\ A\ q_1] \mid 0[q_0\ A\ q_1][q_1\ A\ q_1]$ (B; B, C)

③ $[q_0\ A\ q_1] ::= 1$ (B)

④ $[q_1\ A\ q_1] ::= 1$ (C)

⑤ $[q_1\ A\ q_1] ::= \lambda$ (C)

⑥ $[q_1\ A_0\ q_1] ::= \lambda$ (D)

⇒

S ::= A
A ::= 0BD
B ::= 0BC | 1
C ::= 1 | λ
D ::= λ

⇒ S ::= B ::= =

2) $L(AP) \supset L(G_2)$

$L = L(G_2) \rightarrow \exists AP \;/\; L(AP) = L$

$\{ G_2 = (\Sigma_T, \Sigma_N, P, S)$ FNG (Suponemos que está en la forma normal de Greibach).

$G_2 \longrightarrow AP$ A partir de la gramática construiremos el AP.

• Método 1.

$\{ AP_1 = (\Sigma, \Gamma, Q, q_0, A_0, f, \phi)$ donde:

$$\begin{cases} \Sigma = \Sigma_T & q_0 = q \\ \Gamma = \Sigma_N & A_0 = S \\ Q = \{q\} & \end{cases}$$

La función de transición f se obtiene de la siguiente forma:

① Si $A ::= az \longrightarrow (q, z) \in f(q, a, A)$

② Si $S ::= \lambda \longrightarrow (q, \lambda) \in f(q, \lambda, S)$

EJ: $P = \begin{cases} S ::= ACy \mid Ay \mid \lambda \\ A ::= xACy \mid xAy \mid x \\ C ::= y \end{cases}$ $\quad G_2 = (\{x, y\}, \{A, C, S\}, P, S)$

$$\begin{cases} S ::= \underbrace{xACy}_{A}Cy \mid \underbrace{xAy}_{A}Cy \mid \underbrace{xCy}_{A} \mid \underbrace{xACyy}_{A} \mid \underbrace{xAyy}_{A} \mid \underbrace{xy}_{A} \mid \lambda \\ A ::= xACy \mid xAy \mid x \\ C ::= y \end{cases} \Longrightarrow$$

$S ::= xACCCC \mid xACCC \mid xCC \mid xACC \mid xC \mid \lambda$

$A ::= xACC \mid xAC \mid x$

$C ::= y$

$AP_v = (\{x, y\}, \{S, A, C\}, \{q\}, q, A_0, f, \phi)$ ($A_0 = S$)

$f(q\ x\ S) = (q\ ACCCC)\ (q\ ACCC)\ (q\ CC)\ (q\ ACC)\ (q\ C)$

$f(q\ \lambda\ S) = (q\ \lambda)$

$f(q\ x\ A) = (q\ ACC)\ (q\ AC)\ (q\ \lambda)$

$f(q\ y\ C) = (q\ \lambda)$

- <u>Método 2.</u>

$G_2 = (\Sigma_T, \Sigma_N, P, S)$ No hace falta que esté en FNG.

$AP_v = (\Sigma, \Gamma, Q, q, A_0, f, \emptyset)$ donde:

$$\begin{cases} \Sigma = \Sigma_T \\ \Gamma = \Sigma_T \cup \Sigma_N \\ Q = \{q\} \end{cases}$$

La <u>función</u> de <u>transición</u> f se obtiene de la siguiente forma:

- $\forall A \in \Sigma_N \quad x \in \Sigma_T \cup \Sigma_N$

$A ::= x \longrightarrow (q\ x) \in f(q\ \lambda\ A)$

- $\forall a \in \Sigma_T$

$(q\ \lambda) \in f(q\ a\ a)$

EJ: $G_2 = (\{a,b\}, \{S,A,B\}, P, S)$

$$P = \begin{cases} S ::= aAb \mid aBbb \mid ab \mid abb \mid \lambda \\ A ::= aAb \mid ab \\ B ::= aBbb \mid abb \end{cases}$$

$AP = (\{a,b\}, \{S,A,B,a,b\}, \{q\}, q, s, \delta, \emptyset)$

① $\delta(q\ \lambda\ S) = \{(q\ aAb)(q\ aBbb)(q\ ab)(q\ abb)(q\ \lambda)\}$

② $\delta(q\ \lambda\ A) = \{(q\ aAb)(q\ ab)\}$

③ $\delta(q\ \lambda\ B) = \{(q\ aBbb)(q\ abb)\}$

④ $\delta(q\ a\ a) = (q\ \lambda)$

⑤ $\delta(q\ b\ b) = (q\ \lambda)$

- Lenguaje reconocido:

$L = \{a^n b^m / n, m \geq 0, m = n \text{ ó } m = 2n\}$ Palabra mínima: ab

$(q\ ab\ S) \overset{①}{\vdash} (q\ ab\ ab) \overset{④}{\vdash} (q\ b\ b) \overset{⑤}{\vdash} (q\ \lambda\ \lambda)$

$(q\ ab\ S) \vdash^* (q\ \lambda\ \lambda)$

- Ahora de AP a G_2.

$G_2 = (\Sigma_T, \Sigma_N, S, P)$

$S ::= \underbrace{[q\ s\ q]}_{s'}$

→

① $[q,S,q] ::= [q\ a\ q][q\ A\ q][q\ b\ q] \mid [q\ a\ q][q\ B\ q][q\ b\ q][q\ b\ q] \mid$
$\mid [q\ a\ q][q\ b\ q] \mid [q\ a\ q][q\ b\ q][q\ b\ q] \mid \lambda$

② $[q,A,q] ::= [q\ a\ q][q\ A\ q][q\ b\ q] \mid [q\ a\ q][q\ b\ q]$

③ $[q,B,q] ::= [q\ a\ q][q\ B\ q][q\ b\ q][q\ b\ q] \mid [q\ a\ q][q\ b\ q][q\ b\ q]$

④ $[q\ a\ q] ::= a$

⑤ $[q\ b\ q] ::= b$

$$\begin{cases} S ::= S' \\ S' ::= CAD \mid CBDD \mid CD \mid CDD \mid \lambda \\ A ::= CAD \mid CD \\ B ::= CBDD \mid CDD \\ C ::= a \\ D ::= b \end{cases} \Longrightarrow \begin{cases} S ::= aAb \mid aBbb \mid ab \mid abb \mid \lambda \\ A ::= aAb \mid ab \\ B ::= aBbb \mid abb \end{cases} \xRightarrow{FNG}$$

$$\begin{cases} S ::= aAC \mid aBCC \mid aC \mid aCC \mid \lambda \\ A ::= aAC \mid aC \\ B ::= aBCC \mid aCC \\ C ::= b \end{cases}$$

Por el Método 1:

① $\delta(q\ a\ S) = (q\ AC)(q\ BCC)(q\ C)(q\ CC)$

② $\delta(q\ \lambda\ S) = (q\ \lambda)$

③ $\delta(q\ a\ A) = (q\ AC)(q\ C)$

④ $\delta(q\ a\ B) = (q\ BCC)(q\ CC)$

⑤ $\delta(a\ b\ C) = (q\ \lambda)$

DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL

FACULTAD DE INFORMÁTICA

UNIVERSIDAD POLITÉCNICA DE MADRID

Nº 235.
0'06 €

INFORMATICA TEÓRICA

Curso 2004-2005

Prácticas Tema 7

AUTÓMATAS de PILA

Práctica 7.1: Construcción de Autómatas de Pila

1.- Construir autómatas de pila que reconozcan por vaciado de pila los lenguajes siguientes:

a) $L_1 = \{ a^n b^m / n \geq m > 0 \}$

b) $L_2 = \{ a^n b^m / n > m \geq 0 \}$

c) $L_3 = \{ a^n b^m / n \neq m \}$

d) $L_4 = \{ a^n b^{2n} / n > 1 \}$

e) $L_5 = \{x \in \{0,1\}^* \mid N_0(x) = N_1(x) \}$

f) $L_6 = \{ a^n b^m c^p \mid m = n + p , n > 0 , p \geq 0 \}$ (Examen Junio 2001)

2.- Construir un autómata de pila tal que $LV = \Sigma^*$ y $LF = \varnothing$

b) Construir un autómata de pila tal que $LV = \varnothing$ y $LF = \Sigma^*$

(Examen Junio 2004)

Práctica 7.2: Autómatas de Pila y Lenguajes Independientes del Contexto

1.- Sea el lenguaje L_{dp} (dobles paréntesis) sobre el alfabeto { (,) , [,] } definido recursivamente de la forma siguiente:

i) $(), [], \lambda \in L$
ii) $x \in L \Rightarrow (x) \in L$ y $[x] \in L$
iii) $x,y \in L \Rightarrow xy \in L$
iv) son palabras de L todas las que se obtienen aplicando las reglas i), ii) y iii) un número finito de veces.

a) Construir una gramática independiente del contexto que genere dicho lenguaje.

b) A partir de la gramática construida en a) obtener, mediante el algoritmo explicado en clase, un autómata de pila que reconozca dicho lenguaje por vaciado de pila.

c) Construir directamente un autómata de pila que reconozca el lenguaje L_{dp}.

2.- Obtener la gramática independiente del contexto que genera el lenguaje reconocido por el siguiente AP definido por sus transiciones:

1. $f(q_0, a, A_0) = (q_0, AA_0)$
2. $f(q_0, a, A) = (q_0, AA)$
3. $f(q_0, b, A) = (q_0, BA)$
4. $f(q_0, b, B) = (q_0, BB)$
5. $f(q_0, c, B) = (q_1, \lambda)$
6. $f(q_1, c, B) = (q_1, \lambda)$
7. $f(q_1, d, A) = (q_1, \lambda)$
8. $f(q_1, e, A) = (q_1, A)$
9. $f(q_1, \lambda, A_0) = (q_1, \lambda)$

¿Qué lenguaje reconoce por vaciado de pila dicho autómata?

3.- Sea el autómata de pila definido por sus transiciones:

$f(q_0, a, A_0) = (q_0, AAA_0)$
$f(q_0, a, A) = (q_0, AA)$
$f(q_0, \lambda, A) = (q_0, AA)$
$f(q_0, b, A) = (q_1, \lambda)$
$f(q_1, b, A) = (q_1, \lambda)$
$f(q_1, \lambda, A_0) = (q_0, \lambda)$

a) ¿Qué lenguaje reconoce?

b) Utilizando el algoritmo correspondiente obtener una gramática independiente del contexto que genere el lenguaje que reconoce el autómata. Dicha gramática debe estar bien formada.

c) Construir directamente de a) una gramática independiente del contexto para ese lenguaje.

(Examen Septiembre 2004)

TEMA 8: MÁQUINAS DE TURING

1.- Introducción.
2.- Definición y funcionamiento.
3.- Ejemplos de MT.
4.- Máquina de Turing Universal
5.- MT que aceptan $L(MT, \Sigma) = L_0$

1. INTRODUCCIÓN.

Las Máquinas de Turing son el autómata más general, capaz de reconocer los lenguajes generados por las gramáticas menos restrictivas, las de Tipo 0.

Es el modelo de cómputo más potente que existe.

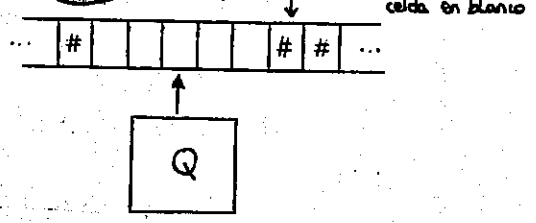
El primer modelo abstracto que permitió formalizar el concepto de algoritmo es el de la MT.

Los lenguajes para los cuales no existe una MT que los acepte son lenguajes indecidibles. Una función es computable cuando existe una MT que permite calcular el valor de esa función.

2. DEFINICIÓN Y FUNCIONAMIENTO

Una Máquina de Turing es un dispositivo capaz de adoptar un estado determinado (uno de los elementos de Q), conectado a una cabeza de lectura y escritura, que puede leer o escribir un símbolo en la cinta. En un momento dado, en función del símbolo que ha leído y del estado en que se encuentra, realizará las 3 acciones siguientes:

Mientras que la cinta de las MT es ∞, la de los A. Linealmente Acotad está acotada a ambos lados.



1.- Pasa a un nuevo estado
2.- Imprime un símbolo en la cinta, en la misma posición donde acaba de leer un símbolo de entrada.
3.- Mueve la cabeza de lectura y escritura una posición hacia la izquierda (I) o hacia la derecha (D) o la máquina se detiene (P).

Las MT vamos a definirlas como un dispositivo de cómputo no determinista.

$$M = (\Sigma, Q, q_0, \delta)$$

$$\delta : Q \times (\Sigma \cup \{\#\}) \longrightarrow P(Q \times (\Sigma \cup \{\#\}) \times \{I, D\})$$

↳ P( ) ≡ Conjunto de las partes.

La función de transición δ puede representarse mediante una tabla: las filas estarán encabezadas por los estados, las columnas por los símbolos de cinta y habrá 3 elementos en cada casilla:

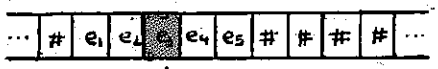
- El estado siguiente.
- El símbolo que se debe escribir en la cinta.
- Movimiento de la cabeza: I, D, P.

EJ:

	1	0	b
p	q0D	p0I	rbD
q	q1D	q0D	p0I
r		r1D	sbP
s			

Algunas casillas pueden estar en blanco si desde ese estado no es posible que lea ese símbolo.

- Configuración de una MT.



≡ $e_1 e_2 e_3 e_4 e_5$

Posible transición => $\delta(q, e_3) = (q', e', D)$

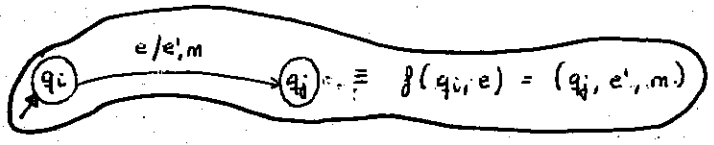
$o\, e_1 e_2 q e_3 e_4 e_5 \vdash e_1 e_2 e' q' e_4 e_5$

↳ Precede si $\delta(q, e_3) = (q', e', D)$

$o\, e_1 e_2 q e_3 e_4 e_5 \vdash e_1 q' e_2 e' e_4 e_5$

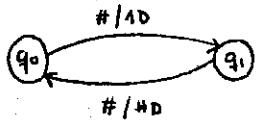
↳ Precede si $\delta(q, e_3) = (q', e', I)$

$\delta(q, e) = \{(q', e', m), (q'', e'', m_2), \ldots, (q^m, e^m, m_m)\}$



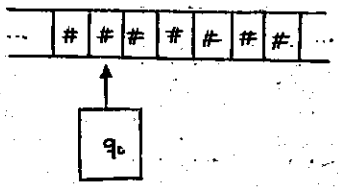
3. EJEMPLOS DE MT.

EJ:



Esta máquina no para nunca. Escribe un 1, deja una celda en blanco, y así sucesivamente.

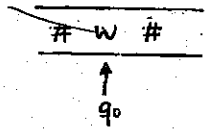
Configuración inicial:



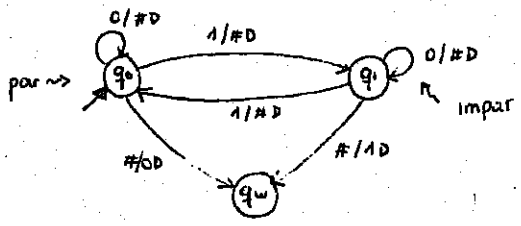
EJ: DETECTOR DE PARIDAD

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

$w \in \Sigma^*$



Cinta infinita



Si $N_1(w)$ par → # 0 # q_w

Si $N_1(w)$ impar → # 1 # q_w

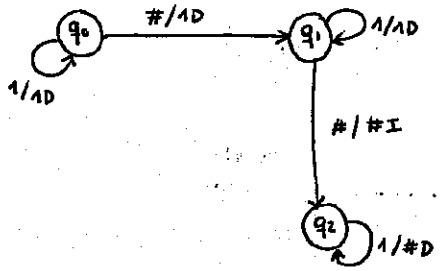
Ej: $v, w \in 1^*$

# v # w #	Configuración inicial
$\uparrow q_0$	
# v w #	Configuración final

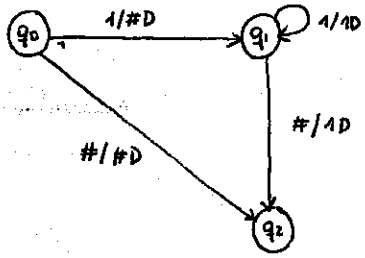
$\Rightarrow$ Por ejemplo: $v = 111$, $w = 1111$ $\}$ $vw = 1111111$

ES UN SUMADOR DE NÚMEROS CODIFICADOS EN UNARIO.

* Posible diseño:



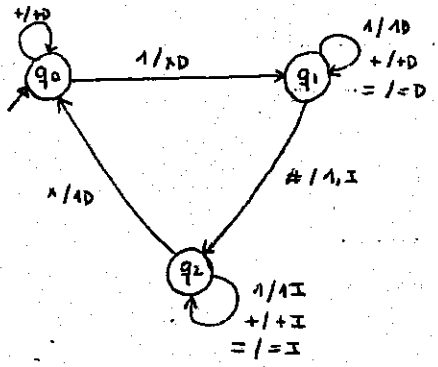
* Posible diseño:



Ej: Otro SUMADOR UNARIO.

# U + V = #	Conf. inicial
# U + V = UV #	Conf. final

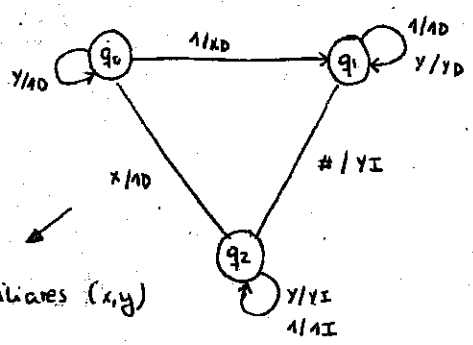
$u, v \in 1^+$ → clausura positiva: No puede ser λ.



EJ: MULTIPLICADOR x2 EN UNARIO

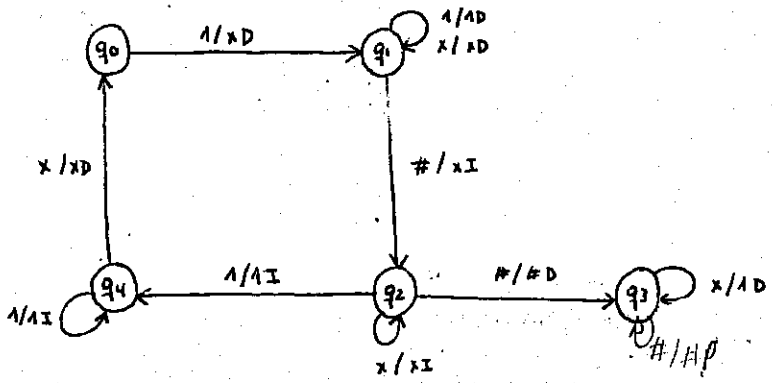
$w \in 1^* \quad \delta(q_0, \#) = \phi$

# w #	C. Inicial
$\uparrow q_0$	
# ww #	C. Final
$\uparrow q_0$	



Máquina con 2 símbolos auxiliares (x,y)

Máquina con 1 sólo símbolo auxiliar (x)

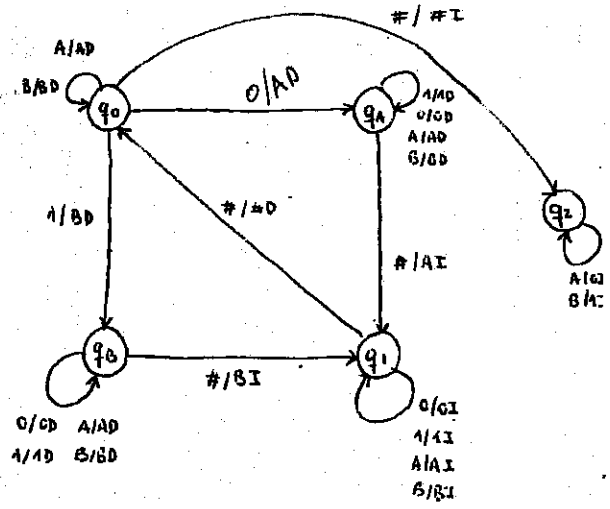


EJ:

# w #	C. Inicial
#ww #	C. Final

$w \in (0+1)^*$

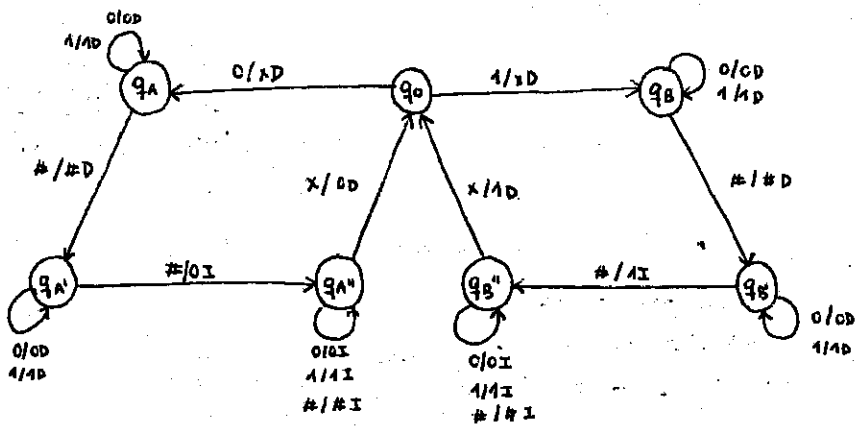
# C11 #	$0/A \rightarrow q_A$
	$1/B \rightarrow q_B$



EJ:

# w #	C. Inicial
$\uparrow q_0$	
# w # w #	C. Final

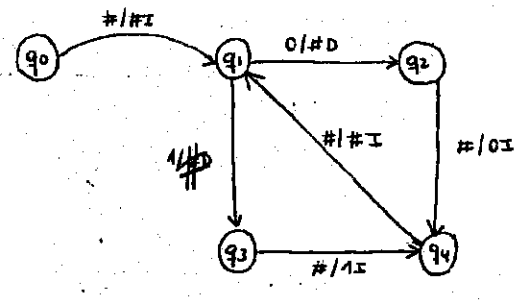
$w \in (0+1)^*$



EJ:

w # w
ww

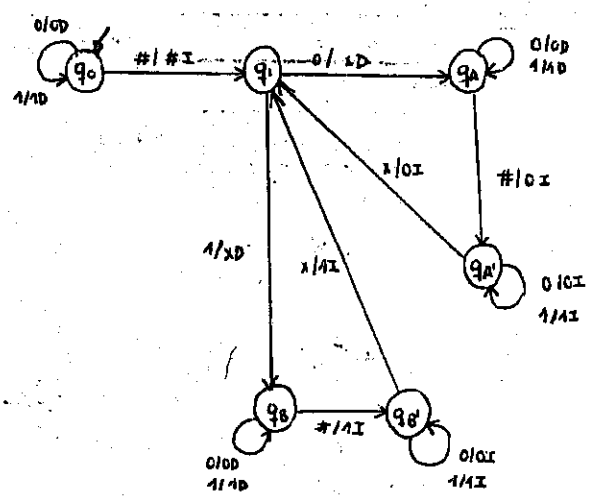
$w \in (0+1)^*$



Todo lo demás es igual

Ej:

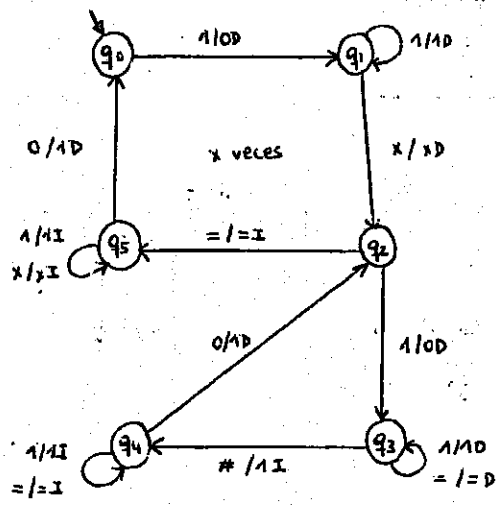
# w #	C.I.
# ww^{-1} #	C.F.



Ej: MULTIPLICADOR UNARIO

# $1^x \times 1^y = \#$	C.I.
# $1^x \times 1^y = 1^{x \cdot y}$	C.F.

x = 111
y = 11



EJ:

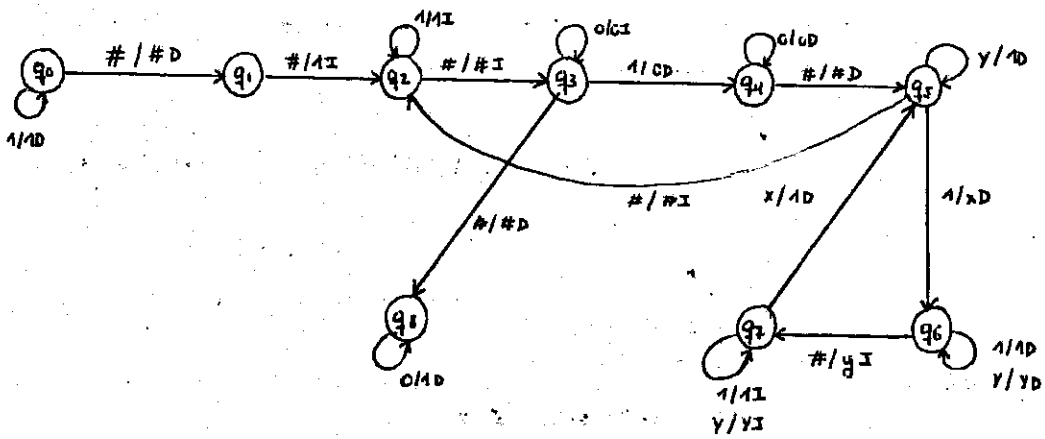
# n #	C.I.	$n \geq 0$
↑ q_0		
# n # 2^n #	C.F.	

$n=0 \rightarrow 2^0 = 1$ | 1

$n=1 \rightarrow 2^1 = 2$ | 11

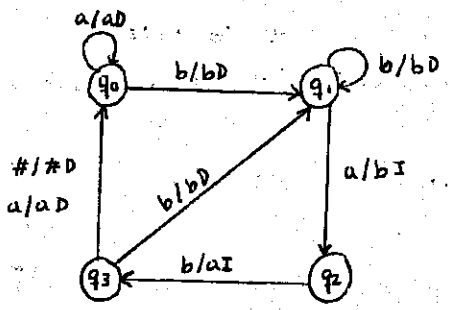
$n=2 \rightarrow 2^2 = 4$ | 1111

$n=3 \rightarrow 2^3 = 8$ | 11111111



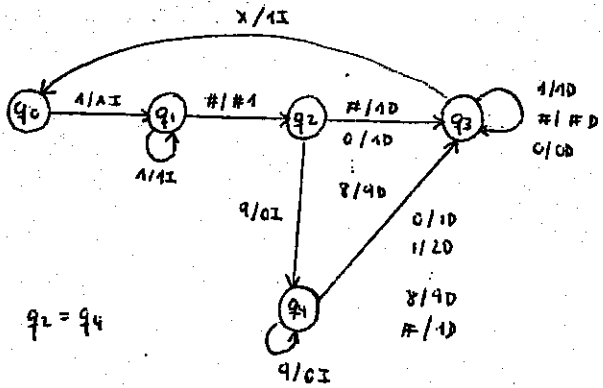
EJ:

# w #	C.I.	$w \in \{a,b\}^*$
# $a^m b^n$ #	C.F.	$N_a(w) = m$ $N_b(w) = n$ $m,n \geq 0$



EJ:

# n #	$n \in 1^+$
↑ q_0	
# n # n #	
↙ decimal ↑ q_0 ↘ unario	



MTU / Descripción MT
\ Datos

4. MÁQUINA DE TURING UNIVERSAL (MTU).

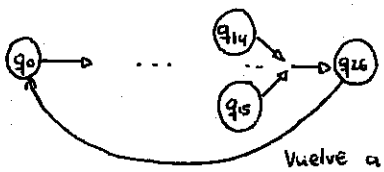
La **MTU** es una máquina que recibe en la cinta una descripción de otra MT y el contenido de la cinta de esta MT, y produce como resultado de su ejecución, el mismo resultado que produciría la MT sobre su cinta. Dicho de otra forma, la MTU es una MT programable (el programa de una MTU está almacenado en su cinta). La MTU se puede programar para que simule el comportamiento de cualquier MT.

$MTU = (M, w) = f(w)$ $\quad M(w) = f(w)$

- w → entrada sobre la que opera M.
- M → máquina a simular

Para simular un movimiento de M, la MTU transita 27 veces.

MTU :



Vuelve a q_0 preparada para simular otro movimiento. Así hasta que se pare

El programa de la MTU es una secuencia binaria y contiene:

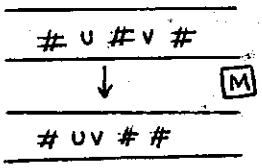
- → Descripción binaria de la máquina M.
- → Contenido inicial de la cinta.
- → Estado inicial en el que se encuentra M.

La MTU tiene 3 módulos:

- **MÓDULO LOCALIZADOR** → Permite a la MTU saber qué movimiento tiene que simular $(q_0 - q_8)$
- **MÓDULO TRANSCRIPTOR** → Copia información de una parte de la cinta en otras celdas (referencia inicial) $(q_9 - q_{13})$
- **MÓDULO SIMULADOR** → $(q_{14} - 26)$

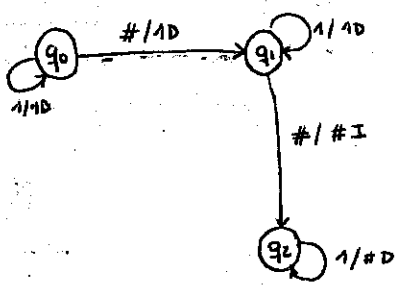
Nuestra MTU sólo podrá simular máquinas que operen a lo sumo sobre 2 símbolos distintos.

EJ: Sumador binario.



$u, v \in 1^*$

$u+v$



#11#111#

$u = 11$; $v = \lambda\lambda\lambda$

#11111##

Cuando la máquina se para, la cinta contiene la suma.

$\delta(q_0, 1) = (q_0, 1, D)$

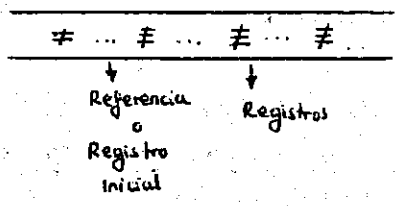
$\delta(q_2, \#) = \phi$

Nuestra máquina universal va a simular este sumador.

- Símbolos auxiliares de la MTU:

(≠) - Símbolo de inicio de información

≠ Separador de registros.



→ Cinta de la máquina universal: una serie de registros, cada uno formado por varias celdas, que contiene información sobre la máquina M.

El sumador que queremos simular tiene 3 estados → Necesito 2 bits para codificarlo.

q_0	→	00	# (blanco)	→ 0	D → 0
q_1	→	01	1	→ 1	I → 1
q_2	→	10			

Movimiento → $\delta(q_i, e) = (q_j, e', m)$

q_j: est. futuro; e': escribe; m: dcha/izq

≠ q_i e q_j e' m ≠ → Contenido de un registro:

- primeras celdas → Estado inicial.
- Elemento que está leyendo
- Estado al que accede el control.
- Elemento que escribe.
- últimas celdas → Desplazamiento de la cabeza (última celda)

1er movimiento → $\delta(q_0, 1) = (q_0, 1, D)$ → (codificación en binario) ≠ 0010010 ≠

≠ 1011100 ≠ 001 ≠ 001001 ≠ 000110 ≠ 110110 ≠ 010001 ≠ 1011000 ≠ # ...

$(q_0, 1)$ 1er mov 2º mov

∞ celdas en blanco

↳ Registro inicial → Estado inicial en el que se encuentra la máquina que vamos a simular y símbolo que inicialmente está leyendo.

Un registro para cada movimiento...

5 movimientos → 5 registros.

Blancos de la máquina M ≠ Blancos de la máquina universal.

(1) ● La máquina universal lee el registro inicial → q_0 e 001.

(2) ● El módulo localizador busca la secuencia 001 al comienzo de los registros, es decir, busca un movimiento en el que se parta de q_0 (00) y se lea 1. Ese registro que localizaremos codifica el movimiento que será simulado por la MTU.

(3) * El módulo transcriptor copia las últimas tres celdas (sin contar el desplazamiento), que contienen el estado al que accede y el símbolo que escribe, y lo escribe en el registro inicial. El símbolo de desplazamiento no cabe en el reg. inicial, por lo que el transcriptor la memoriza. El símbolo que escribe (e') lo pone donde señale *.

⊛ La forma de buscarlo es la siguiente: Examina uno a uno los distintos registros y los que no coinciden con el registro inicial los marca con A y B.

0 → A
1 → B
} Para recordar qué movimientos (registros) ha comparado ya con la referencia inicial.

Si no existe ningún registro que comience por la referencia inicial, es decir, ya los ha marcado todos con A y B y al coger el registro siguiente encuentra un vacío, la MTU (al igual que la M simulada) se para! ↳ el módulo localizador.

2 formas de acabar el localizador:
- Encuentra un registro
- Rechaza todos los registros.

* 10111 CO # 001 # 001001 # ... (1 q0 D)
↓ Mod. localizador
* 10111 CO # 001 # AABAABA # ... #*10111 00 ≠ aab ≠ aab 0010
↓ Mod. Transcriptor
1 * 0111 CO # 001 # AABAABA # ... #*10111 00 ≠ 001 ≠ aabaaba
↓ Mod. Simulador
1 * 0111 00 # 001 # 0010010 # ... #1*0111 00 ≠ 001 ≠ aab 0010
↓
111110 * # 100 # 0010010 # ... #

5. LENGUAJE QUE ACEPTA UNA MÁQUINA DE TURING.

Si después de leer la cadena se para, la acepta.

$\begin{cases} w \in L(M) \equiv & M \text{ se para al procesar } w. \\ w \notin L(M) \equiv & M \text{ no se para al procesar } w. \end{cases}$

$$L(M, \Sigma) = L_0$$

$w \in \Sigma^*$

$$L(M) = \{ w \in \Sigma^* \ / \ M \text{ se para al procesar } w \}$$