Autómatas finitos

Curso 2019-2020

Senén Barro Ameneiro, CiTIUS

@SenenBarro

Material elaborado fundamentalmente por
el profesor Manuel Mucientes Molina

Bibliografía

- J.E. Hopcroft, R. Motwani y J.D. Ullman, "Teoría de Autómatas, Lenguajes y Computación", Addison Wesley, 2008.
  - Capítulo 2 y capítulo 4 (sección 4)

- P. Linz, ''An Introduction to Formal Languages and Automata'', Jones and Bartlett Publishers, Inc, 2001.
  - Capítulo 2

Introducción

- Máquinas secuenciales
  - Mealy
  - Moore
- Autómatas de [número de] estados finito: son máquinas secuenciales
- Reconocen los lenguajes regulares
- Clasificación:
  - Determinista (AFD): el autómata no puede estar en más de un estado simultáneamente
  - No determinista (AFN): puede estar en varios estados al mismo tiempo
- No determinismo
  - no añade ningún lenguaje a los ya definidos por los AFD
  - aumento de la eficiencia en la descripción de una aplicación

Autómatas finitos deterministas (AFD)

- Determinista: para cada entrada, existe un único estado al que el autómata puede llegar partiendo del estado actual
- Consta de:
  - un conjunto finito de ***estados***, Q
  - un conjunto finito de ***símbolos de entrada***, S
  - una ***función de transición*** (δ) que, dados un estado y una entrada, devuelve un estado. $\delta(q, a) = p$
  - un ***estado inicial*** (uno de los estados de Q), q_0
  - un conjunto de ***estados finales*** *o de aceptación* (subconjunto de Q), F
- $A = (Q, S, \delta, q_0, F)$

Funcionamiento de un AFD

- **Lenguaje del AFD**: conjunto de las cadenas que acepta

- **Ejemplo**: AFD que acepta todas las cadenas de ceros y unos que contienen la secuencia 01 en algún lugar de la cadena
  - {*w* | *w* tiene la forma *x*01*y*, donde *x* e *y* son cualesquiera cadenas de símbolos 0 y 1}
  - Si se ha leído la secuencia 01, independientemente de las futuras entradas, el estado debe ser de aceptación
  - Si no se ha leído la secuencia 01, pero la entrada más reciente es 0 y se lee un 1, se pasa a estado de aceptación
  - Si no se ha leído la secuencia 01 y la entrada más reciente es un 1 o no existe, se aceptará la cadena cuando se lea 01
  - $A = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$

Diagrama de transiciones

- Es un grafo
  - un nodo por cada estado de Q
  - un arco de q a p etiquetado con a para cada $\delta(q, a) = p$
  - una flecha dirigida al estado inicial
  - los estados finales están marcados por un doble círculo

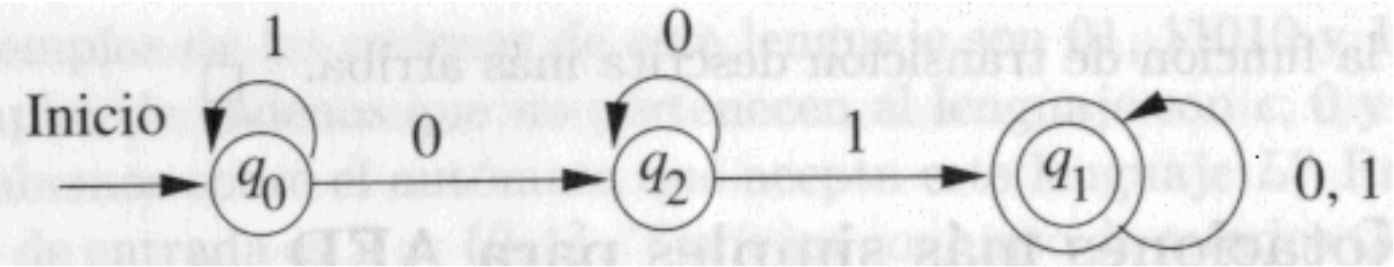


Tabla de transiciones

- Representación tabular de la función δ
- filas: estados
- columnas: entradas
- estado inicial: indicado por una flecha
- estados finales: indicado por: *

	0	1
$->q_0$	q_2	q_0
$*q_1$	q_1	q_1
q_2	q_2	q_1

Extensión a cadenas

- Función de transición: δ
- Función de transición extendida: $\hat{\delta}$
  - dado un estado *q* y una cadena *w*, devuelve un estado *p*
- Definición por inducción de la función de transición extendida
  - Base: $\hat{\delta}(q,\lambda) = q$

$$\hat{\delta}(q,w) = \delta(\hat{\delta}(q,x),a)$$

  - Paso inductivo: *w* = *xa*

- Lenguaje de un AFD: lenguaje regular

$$L(A) = \{w \mid \hat{\delta}(q_0,w) \, pertenece \, a \, F\}$$

Problemas

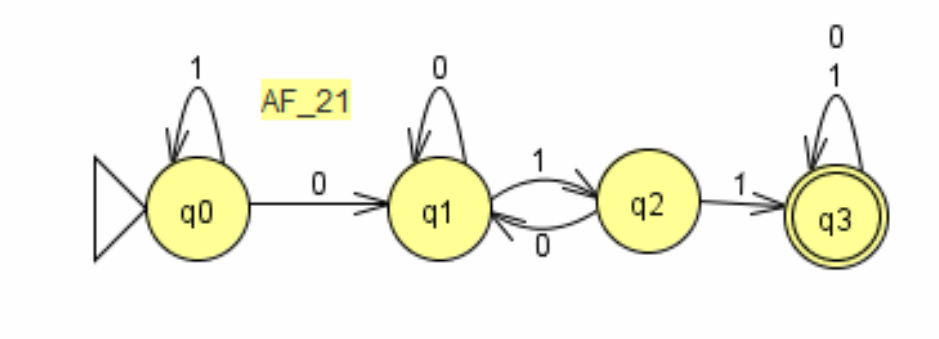
Construir los AFD que acepten los siguientes lenguajes sobre el alfabeto {0, 1}:

1. El conjunto de cadenas con 011 como subcadena
2. El conjunto de cadenas terminadas en 00
3. El conjunto de cadenas cuyo tercer símbolo desde el extremo derecho sea un 1
4. El conjunto de cadenas que empiezan o terminan por 01
5. El conjunto de palabras que **no** contienen las subcadenas "100"
6. El conjunto de cadenas que contengan un número par (se incluye 0) de subcadenas con un número par de ceros consecutivos. Por ejemplo, el autómata deberá reconocer la cadena **00**11000**0000**, pero no la cadena 0001000**00**

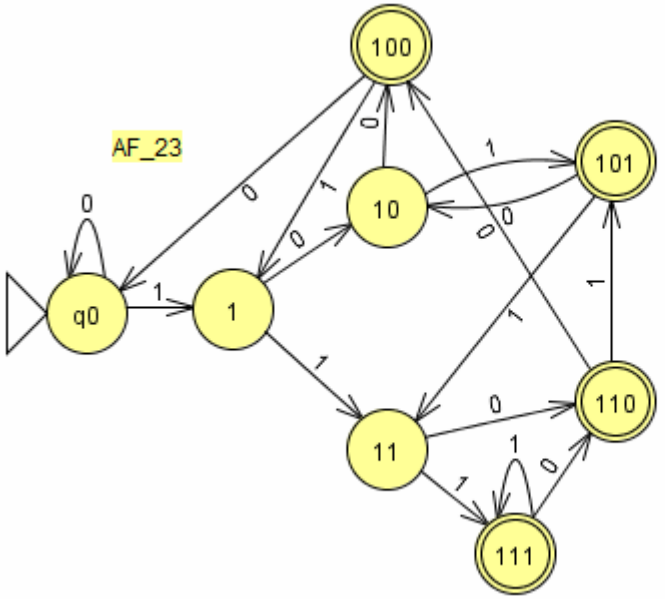
Problemas

Construir los AFD que acepten los siguientes lenguajes sobre el alfabeto {0, 1}:

1. El conjunto de cadenas con 011 como subcadena

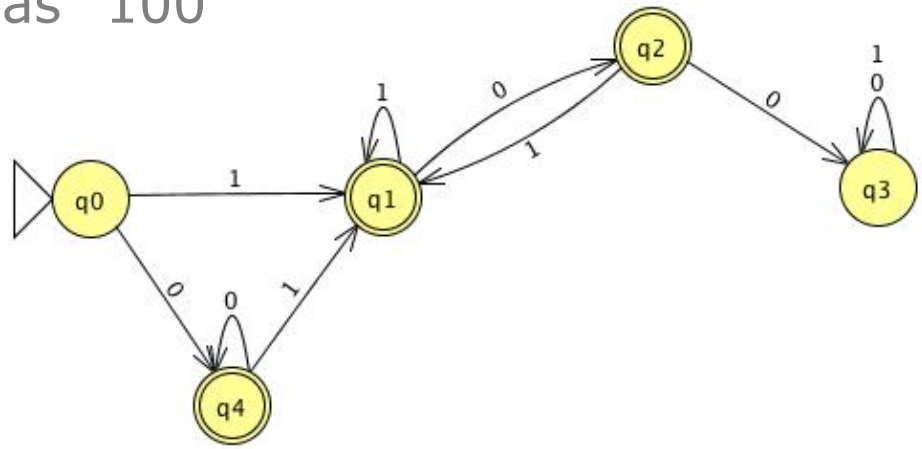


2. El conjunto de cadenas cuyo tercer símbolo desde el extremo derecho sea un 1

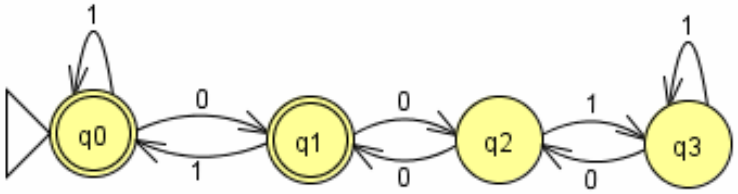


Problemas

5. Lenguaje formado por el conjunto de palabras que no contienen las subcadenas “100”

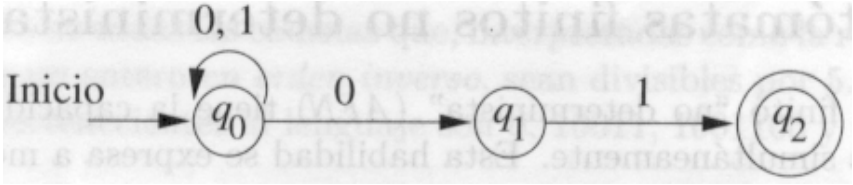


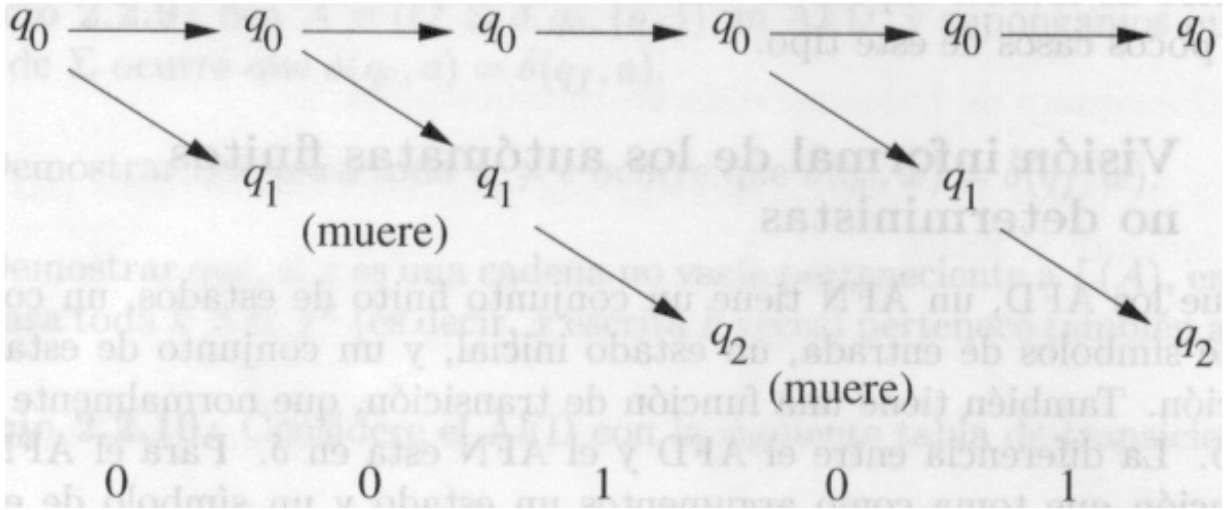
6. Conjunto de cadenas que contengan un número par (se incluye 0) de subcadenas con un número par de ceros consecutivos. Por ejemplo, el autómata deberá reconocer la cadena **00**11000**0000**, pero no la cadena 0001000**00**

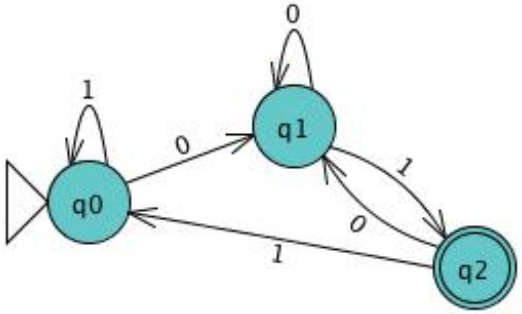


Autómatas finitos no deterministas

- AFN: capacidad de estar en varios estados simultáneamente
  - Aceptan los lenguajes regulares, igual que los AFD
  - Más compactos y fáciles de diseñar que los AFD
  - Siempre es posible convertir un AFN a un AFD
- Ejemplo: AFN que acepta las cadenas terminadas en 01



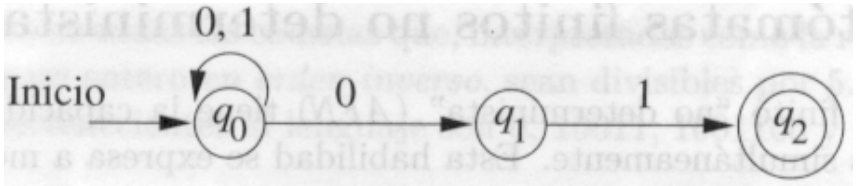




Definición de los AFN

- $A = (Q, \Sigma, \delta, q_0, F)$
- δ devuelve en este caso un conjunto de estados y no un sólo estado
- Ejemplo: AFN que acepta las cadenas terminadas en 01: $A=(\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\emptyset$



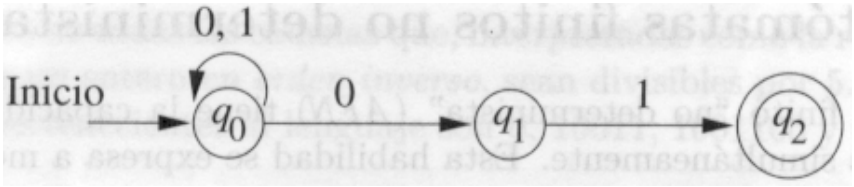
Función de transición extendida

- Base: $\hat{\delta}(q,\lambda)=\{q\}$
- Paso inductivo:
  - $w = xa$,

$$\hat{\delta}(q,x)=\{p_1,p_2,\ldots,p_k\}$$
$$\bigcup_{i=1}^{k}\delta(p_i,a)=\{r_1,r_2,\ldots,r_m\}$$

$$\Rightarrow \hat{\delta}(q,w)=\{r_1,r_2,\ldots,r_m\}$$

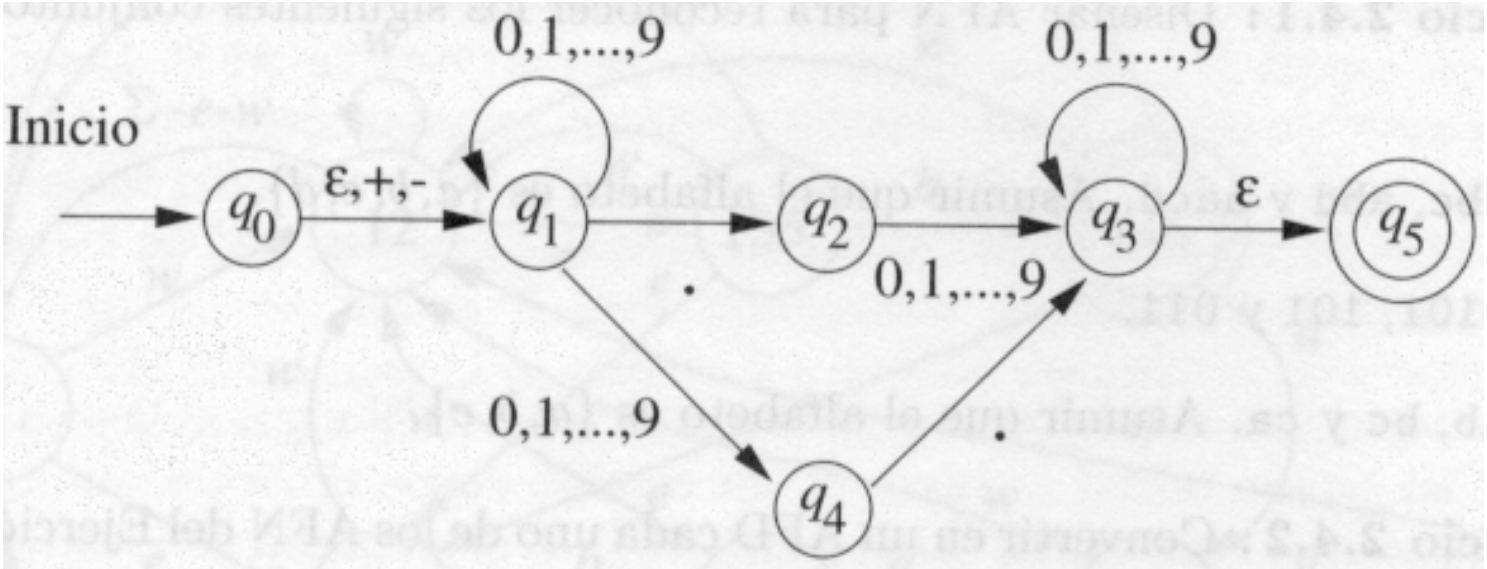
- Ejemplo: describir el proceso de la entrada 1001 para el autómata:



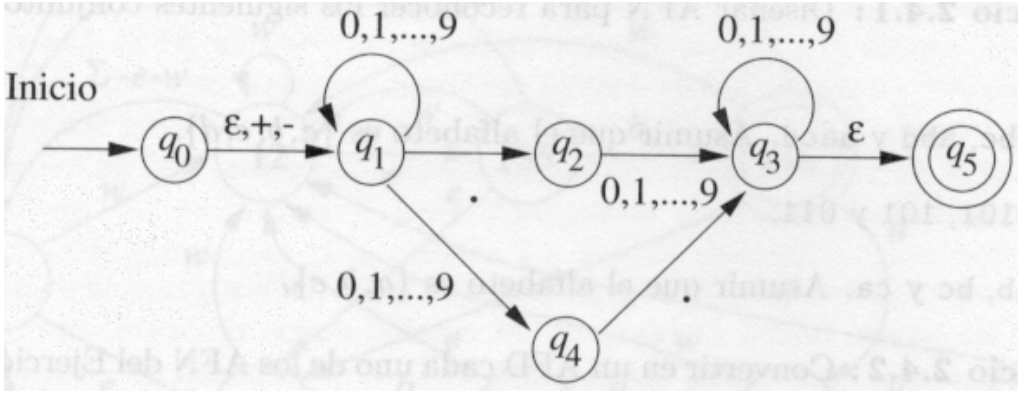
- Lenguaje de un AFN: $L(A)=\{w \mid \hat{\delta}(q_0,w)\cap F \neq \varnothing\}$

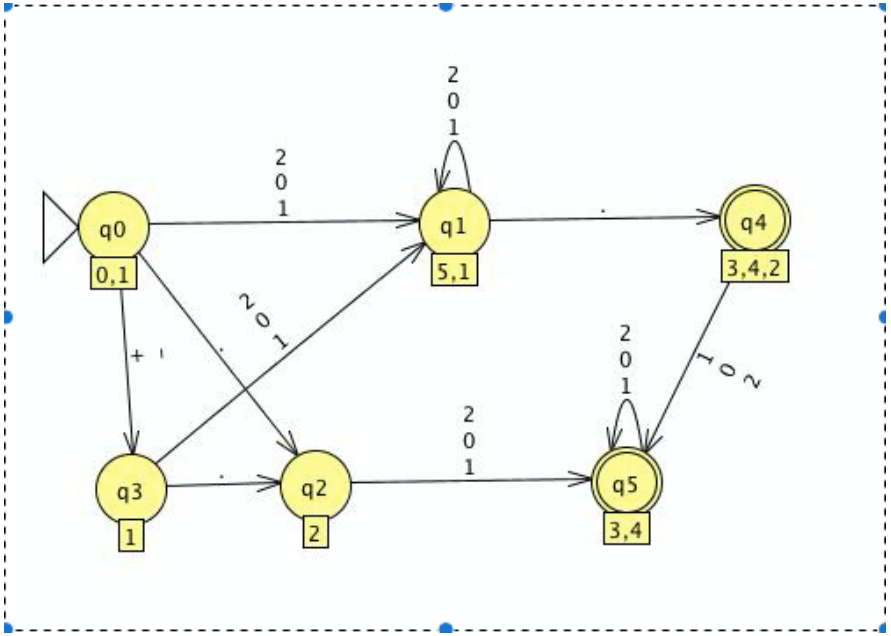
AFN con transiciones ε (AFN-ε)

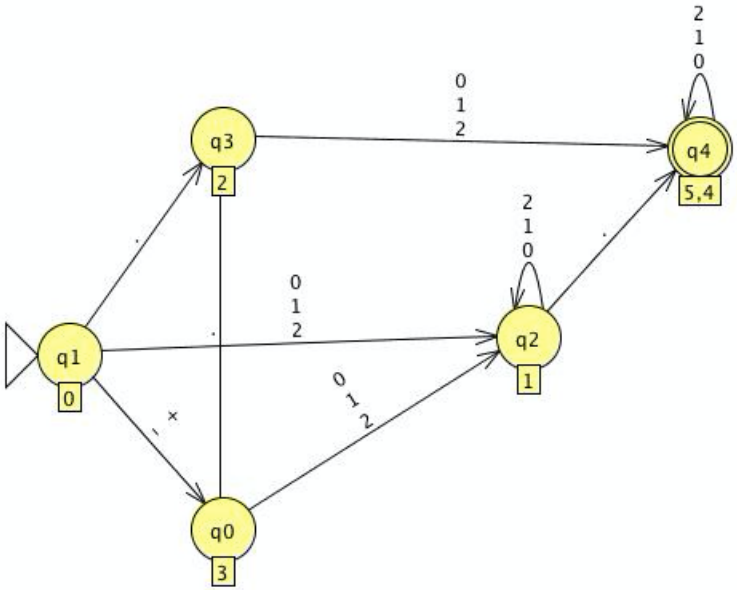
- Transiciones para la cadena vacía, ε
- No expande la clase de lenguajes que aceptan los AF
- Proporciona “facilidades de programación”
- Ejemplo: AFN-ε que acepta números decimales



AFN y su AFD equivalente

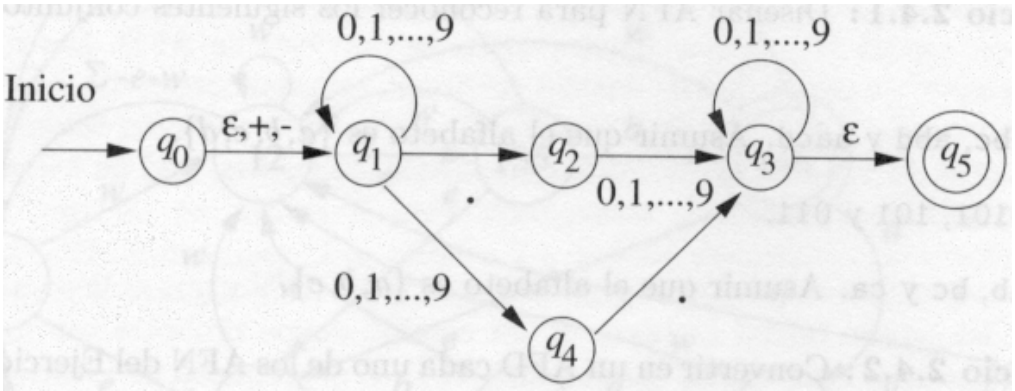






Notación

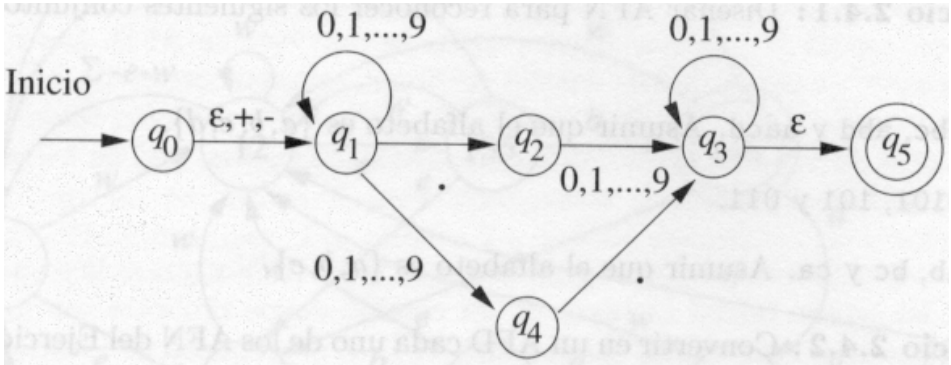
- $A = (Q, \Sigma, \delta, q_0, F)$, pero ahora δ es una función de:
  - un estado de Q
  - un elemento de Σ U $\{\varepsilon\}$
- El símbolo ε no puede formar parte del alfabeto
- Ejemplo:

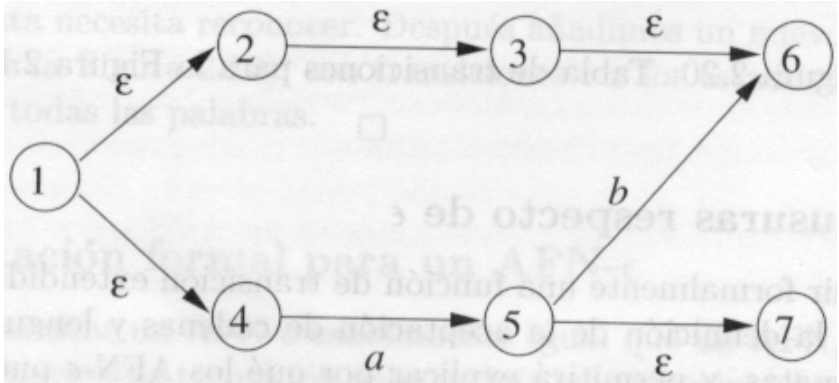


	ε	+, -	.	0, 1, ..., 9
->q_0	$\{q_1\}$	$\{q_1\}$	∅	∅
q_1	∅	∅	$\{q_2\}$	$\{q_1, q_4\}$
q_2	∅	∅	∅	$\{q_3\}$
q_3	$\{q_5\}$	∅	∅	$\{q_3\}$
q_4	∅	∅	$\{q_3\}$	∅
*q_5	∅	∅	∅	∅

Clausuras respecto de ε

- Clausura del estado *q* respecto de ε, *CLAUS*ε(*q*)
  - Base: el estado q está en *CLAUS*ε(*q*)
  - Paso inductivo: si δ es la función de transición del AFN-ε, y el estado *p* está en *CLAUS*ε(*q*), entonces *CLAUS*ε(*q*) contiene todos los estados de δ(*p*, ε)
- Ejemplos: determinar las clausuras de los estados de los autómatas de las siguientes figuras





Transiciones y lenguajes extendidos

- Definición recursiva de $\hat{\delta}$
  - Base: $\hat{\delta}(q,\varepsilon) = CLAUS_{\varepsilon}(q)$
  - Paso inductivo: sea *w* = *xa*, donde *a* es un elemento de Σ
    - Sea $\hat{\delta}(q,x) = \{p_1, p_2, \ldots, p_k\}$
    - Sea $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, \ldots, r_m\}$

$$\hat{\delta}(q,w) = \bigcup_{j=1}^{m} CLAUS_{\varepsilon}(r_j)$$

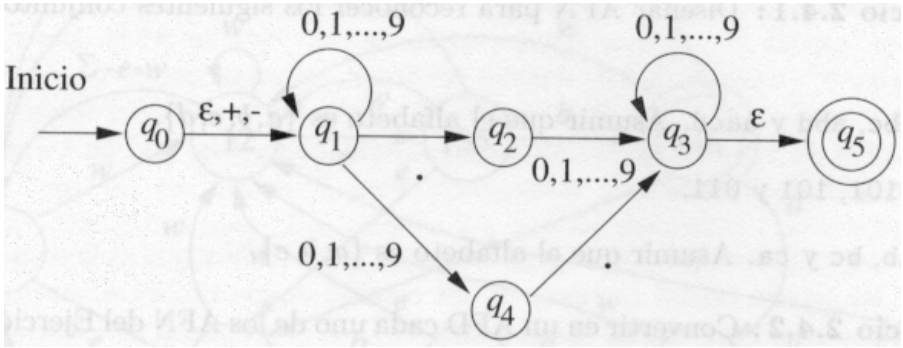
- Lenguaje de un AFN-ε, E=(*Q*, Σ, δ, q_0, *F*):

$$L(E) = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \varnothing\}$$

- Ejemplo: calcular

$$\hat{\delta}(q_0, 5.6)$$

para este autómata:



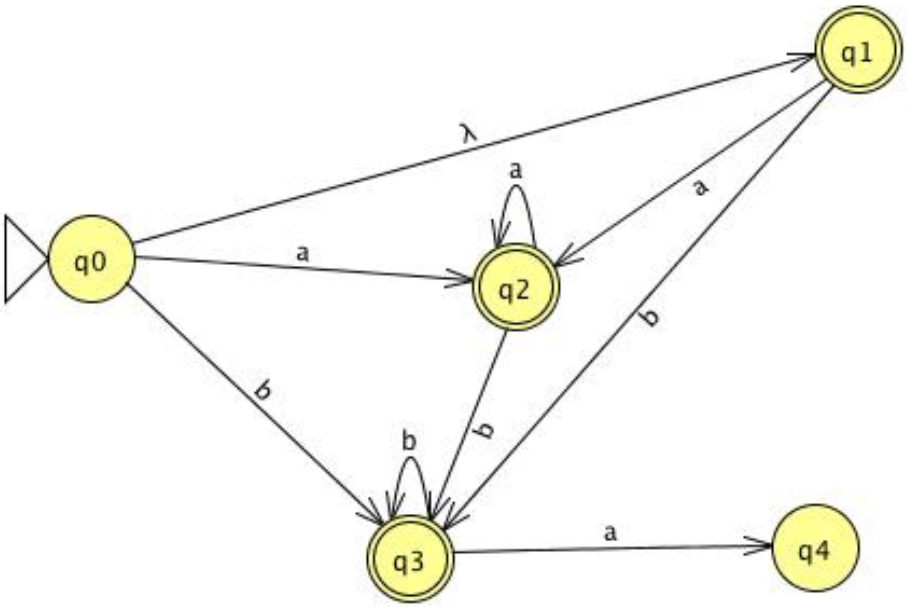
Problemas

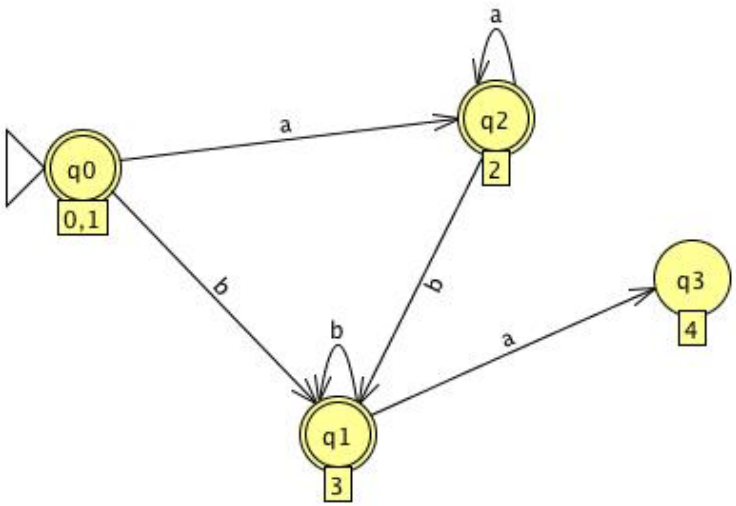
Diseñar el AFN-ε para los siguientes lenguajes:

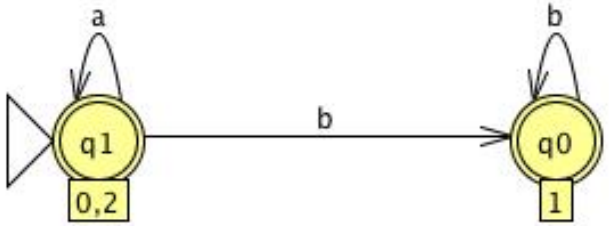
1. Conjunto de cadenas con cero o más letras *a* seguidas de cero o más letras *b*
2. El conjunto de cadenas formadas por *01* repetido una o más veces o por *010* repetido una o más veces
3. AF que sobre el alfabeto {X, Y, Z, 0, 1, 2} reconoce matrículas de vehículos válidas, para las provincias “X”, “YY” y “ZX”. El formato de las matrículas podrá ser:
   - Provincia, 4 números y 0, 1 o 2 letras
   - 4 números y tres letras

Problemas

Conjunto de cadenas con cero o más letras *a* seguidas de cero o más letras *b*

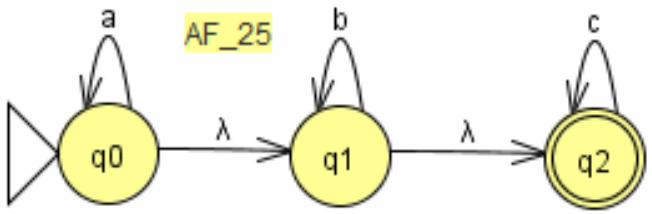




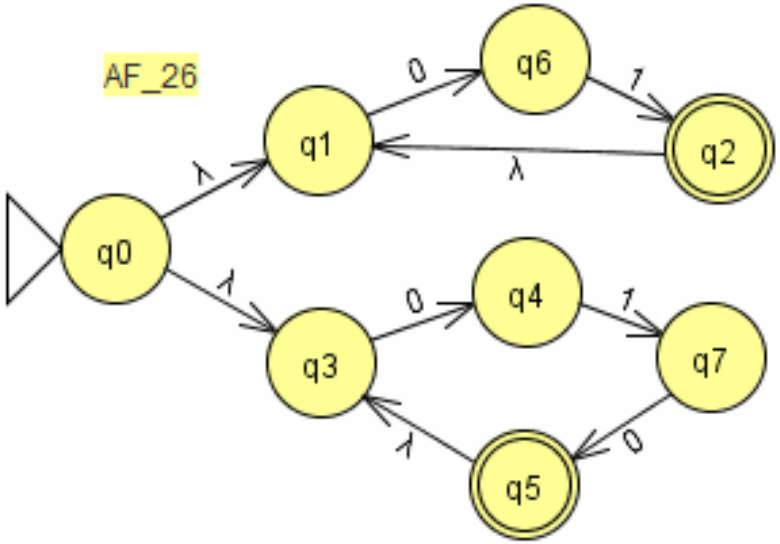


Problemas

1. Conjunto de cadenas con cero o más letras *a* seguidas de cero o más letras *b* seguidas de cero o más letras *c*



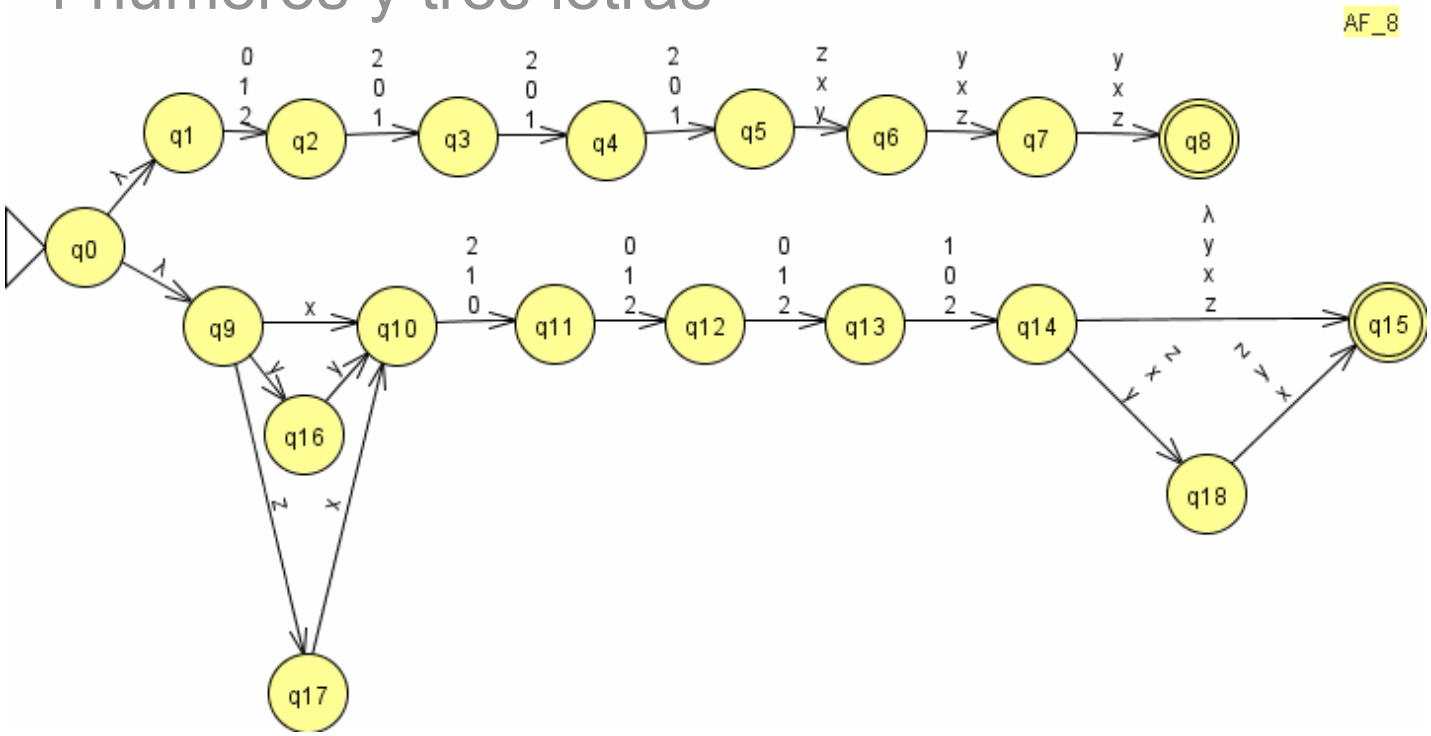
2. El conjunto de cadenas formadas por *01* repetido una o más veces o por *010* repetido una o más veces



Problemas

AF que sobre el alfabeto {X, Y, Z, 0, 1, 2}, que reconoce matrículas de vehículos válidas, para las provincias “X”, “YY” y “ZX”. El formato de las matrículas podrá ser:

- Provincia, 4 números y 0, 1 o 2 letras
- 4 números y tres letras

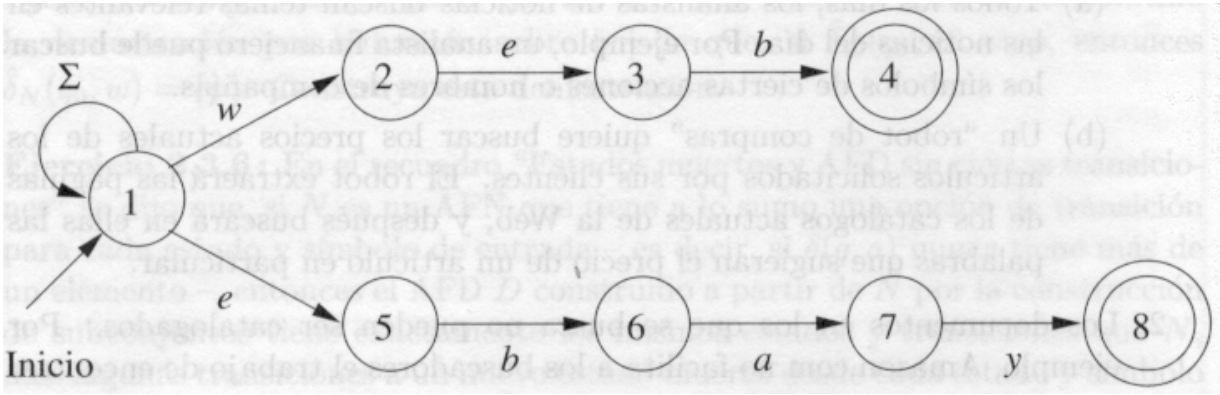


Ejemplo en la búsqueda de texto

- Dado un conjunto de palabras, encontrar todos los documentos que contienen una o más de esas palabras.
- Características que hacen apropiado el uso de autómatas en una aplicación:
  - El repositorio a analizar cambia rápidamente
    - noticias del día
    - robot de compras
  - Los documentos no pueden ser catalogados: Amazon (páginas generadas a partir de consultas)

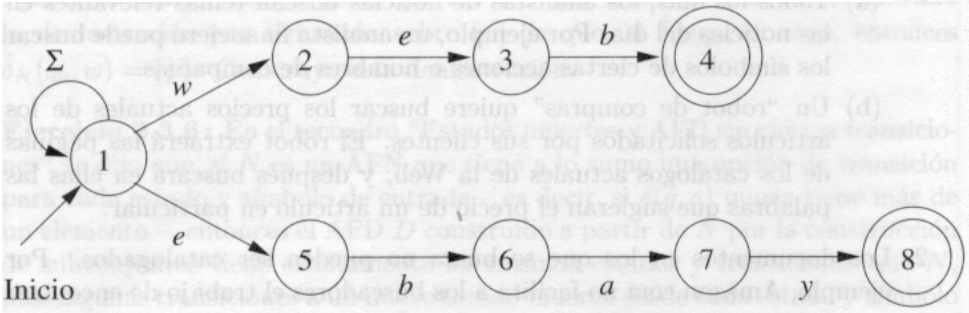
AFN para búsqueda de texto

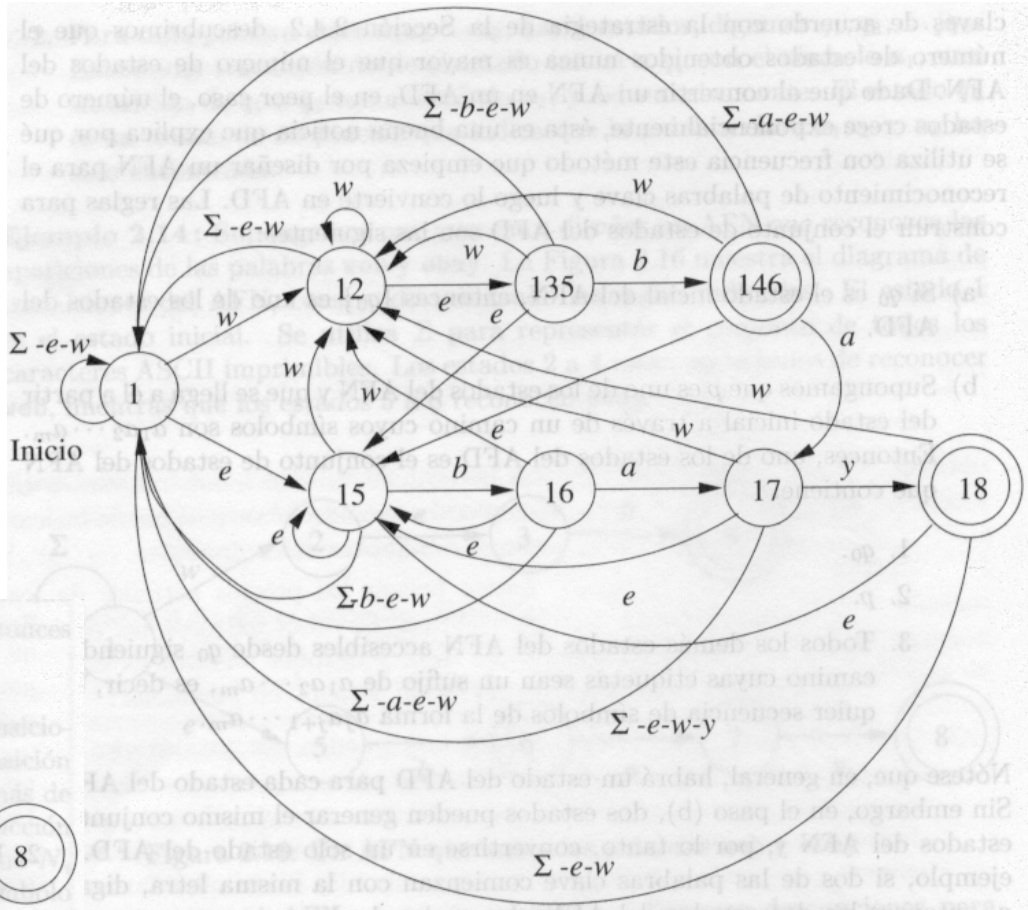
- Estado inicial con transición a sí mismo para cada símbolo de entrada (conjetura)
- Para cada palabra clave $a_1a_2...a_k$, hay k estados, q_1, q_2, ..., q_k
- Transición del estado inicial a q_1 con a_1, de q_1 a q_2 con a_2, etc. El estado q_k indicará que la palabra $a_1a_2...a_k$ ha sido aceptada
- Ejemplo: AFN que reconoce las palabras *web* y *ebay*



Ejemplo

- Conversión del siguiente AFN en AFD





Equivalencia entre AFD y AFN

- Todo lenguaje descrito por un AFN también puede ser descrito por un AFD
  - Para un AFN de *n* estados, el AFD equivalente tendrá como máximo 2^n estados
- Demostración de que un AFD puede hacer lo mismo que un AFN: construcción de subconjuntos
  - construcción de todos los subconjuntos del conjunto de estados del AFN

Construcción de subconjuntos

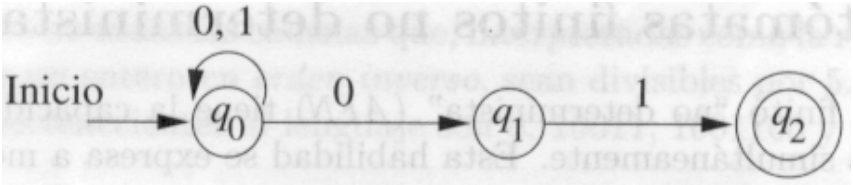
Dado el AFN N= (Q_N, Σ, δ_N, q_0, F_N), obtener el AFD D= (Q_D, Σ, δ_D, $\{q_0\}$, F_D) tal que L(D)=L(N)

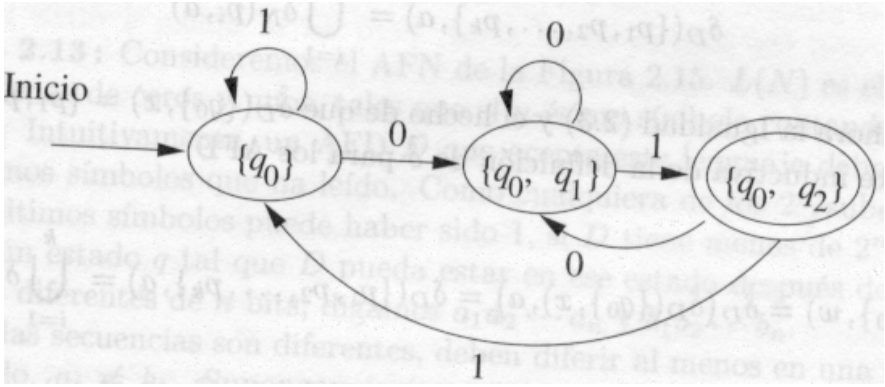
- Alfabetos iguales
- Estado inicial: conjunto con el estado inicial de N
- Q_D: conjunto de subconjuntos de Q_N. Si Q_N tiene n estados, Q_D tendrá 2^n. Eliminación de estados no accesibles
- F_D: conjunto de subconjuntos S de Q_N tales que $S \cap F_N \neq \varnothing$
- Para cada $S \subseteq Q_N$ y cada símbolo de entrada a:

$$\delta_D(S,a) = \bigcup_{p \text{ en } S} \delta_N(p,a)$$

Evaluación "perezosa"

- Base: el conjunto de un elemento que contiene el estado inicial de N es accesible
- Paso inductivo: hemos determinado que el conjunto *S* de estados es accesible. Entonces, para cada símbolo de entrada *a*, se calcula el conjunto de estados $\delta_D(S, a)$, que también serán accesibles
- Ejemplo: aplicar el proceso al siguiente autómata:





Equivalencia entre AFD y AFN

- Teorema: si $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ es el AFD construido a partir del AFN $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ mediante la construcción de subconjuntos, entonces $L(D) = L(N)$

- Teorema: un lenguaje L es aceptado por algún AFN si y sólo si L es aceptado por algún AFD

Problemas

- Verificar si este AFN y AFD son equivalentes

	0	1
->p	{p, q}	{p}
q	{r, s}	{t}
r	{p, r}	{t}
*s	∅	∅
*t	∅	∅

➔

A = {p}
B = {p, q}
C = {p, q, r, s}
D = {p, t}

	0	1
->A	B	A
B	C	D
*C	C	D
*D	B	A

Problemas

Completar el AFD equivalente al AFN dado

	0	1
->p	{p, q}	{p}
q	{r}	{r}
r	{s}	∅
*s	{s}	{s}

➔

A = {p}
B = {p, q}
C = {p, r}
D = {p, q, r}
E = {p, q, s}
F = {p, q, r, s}
G = {p, r, s}
H = {p, s}

	0	1
->A	B	A
B		C
C	E	
D	F	C
*E	F	
*F	F	G
*G	E	
*H	E	H

Eliminación de transiciones ε

- Dado un AFN-ε E = (Q_E, Σ, δ_E, q_0, F_E), encontrar un AFD D=(Q_D, Σ, δ_D, q_D, F_D) que acepte el mismo lenguaje

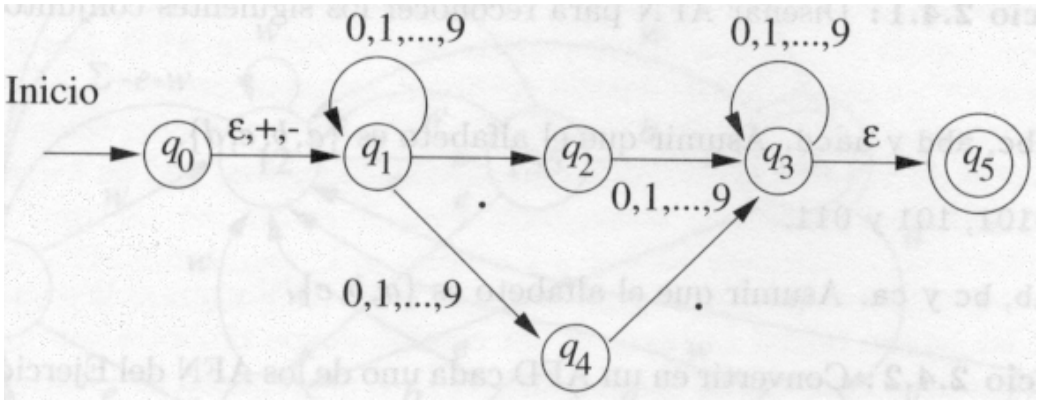
  - Q_D es el conjunto de subconjuntos de Q_E
  - q_D= CLAUSε (q_0)
  - F_D ={S | S pertenece a Q_D y $S \cap F_E \neq \emptyset$}
  - $\delta_D(S, a)$: $S = \{p_1, p_2, \ldots, p_k\}$

$$\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, \ldots, r_m\}$$

$$\delta_D(S, a) = \bigcup_{j=1}^{m} CLAUS \in (r_j)$$

Eliminación de transiciones ε

- Ejemplo: construir el AFD equivalente al de la figura:



	Punto	Signo	Núm.
->A	B	C	D
B			E
C	B		D
D	F		D
*E			E
*F			E

A = {q_0, q_1}, B = {q_2}, C = {q_1},
D = {q_1, q_4}, E = {q_3, q_5}, }, F = {q_2, q_3, q_5}

- Teorema: un lenguaje *L* es aceptado por algún AFN-ε E=(Q_E, Σ, δ_E, q_0, F_E) si y sólo si es aceptado por algún AFD D=(Q_D, Σ, δ_D, q_D, F_D)

Problemas

Dados los siguientes AFN-ε, calcular la clausura respecto de ε para cada estado y convertir los autómatas en AFD

	ε	a	b	c
->p	∅	{p}	{q}	{r}
q	{p}	{q}	{r}	∅
*r	{q}	{r}	∅	{p}

	ε	a	b	c
->p	{q, r}	∅	{q}	{r}
q	∅	{p}	{r}	{p, q}
*r	∅	∅	∅	∅

Problemas

Dados los siguientes AFN-ε, calcular la clausura respecto de ε para cada estado y convertir los autómatas en AFD

	ε	a	b	c
->p	∅	{p}	{q}	{r}
q	{p}	{q}	{r}	∅
*r	{q}	{r}	∅	{p}

➔

	a	b	c
->A	A	B	C
B	B	C	C
*C	C	C	C

A = {p}
B = {p, q}
C = {p, q, r}

	ε	a	b	c
->p	{q, r}	∅	{q}	{r}
q	∅	{p}	{r}	{p, q}
*r	∅	∅	∅	∅

➔

	a	b	c
->*A	A	B	A
*B	A	C	A
*C	D	D	D
D	D	D	D

A = {p, q, r}
B = {q, r}
C = {r}
D = {∅}

Equivalencia de estados

- Dos estados *p* y *q* de un AFD son equivalentes si para toda cadena de entrada *w*, $\hat{\delta}(p, w)$ es un estado de aceptación si y sólo si $\hat{\delta}(q, w)$ es un estado de aceptación
  - En caso contrario, los estados serán distinguibles
- Teorema: la equivalencia de estados es transitiva. Es decir, si para un AFD dos estados *p* y *q* son equivalentes y *q* y *r* son equivalentes, entonces *p* y *r* son equivalentes
- Teorema: si para cada estado *q* de un AFD se crea un bloque que contiene a *q* y a todos los estados equivalentes a *q*, los bloques de estados formarán una partición del conjunto de estados.
  - cada estado estará en un único bloque
  - todos los miembros de un bloque son equivalentes
  - dos estados de bloques diferentes no pueden ser equivalentes

Equivalencia de estados

Conjunto cociente, *Q/E*: partición del conjunto de estados en clases. Cada clase contiene estados equivalentes entre sí

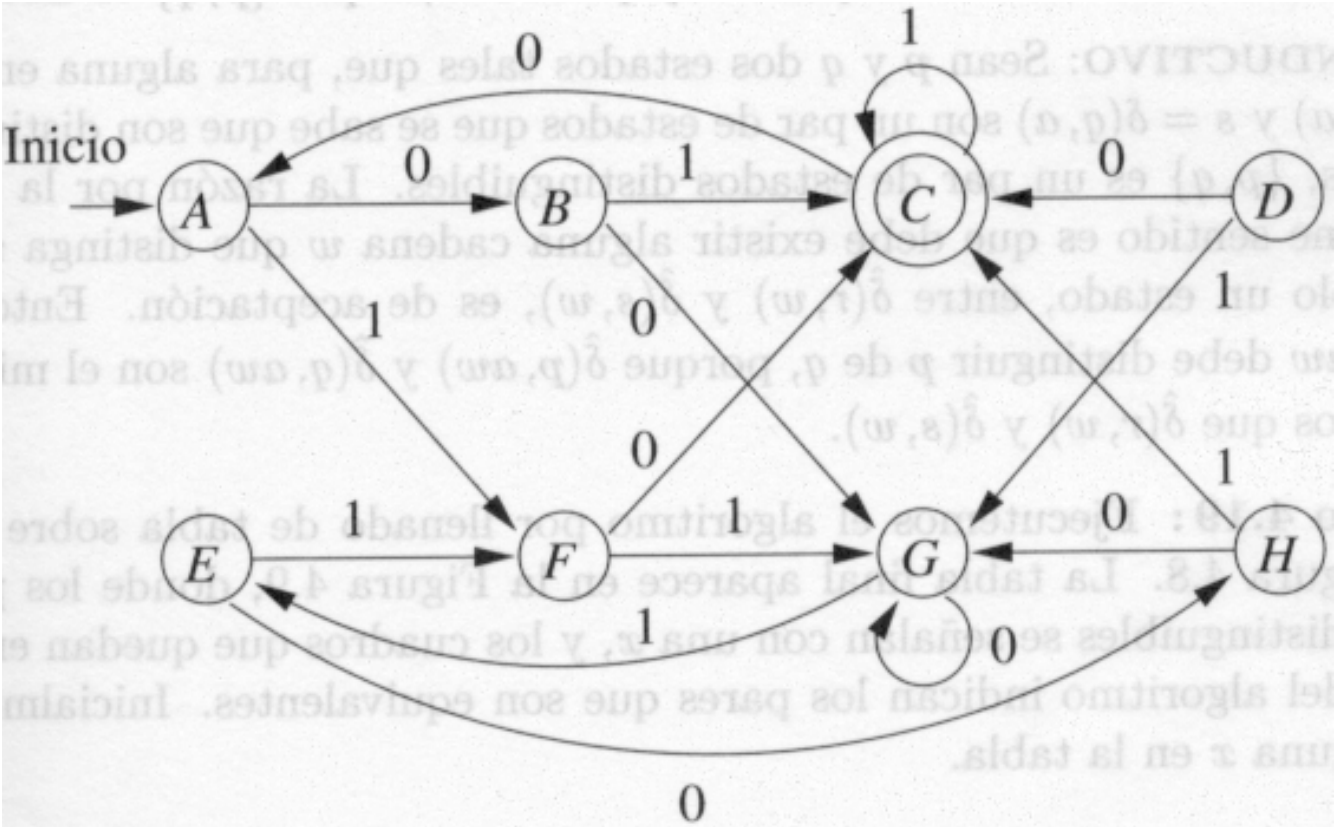
Función Conjunto-Cociente (A): Q_E

A: entrada a la función, AF definido por (Σ, Q, f, q_0, F)
Q_E: salida de la función, conjunto cociente del AF

$Q/E_0 := \{F, \bar{F}\}$;
i:=-1;
Repetir
    i:=i+1;
    Si pE_iq Y
        $\forall a \in \Sigma, f(p,a) \in c_k, f(q,a) \in c_k, (c_k \in Q/E_i)$
    Entonces $pE_{i+1}q$
Hasta que $Q/E_{i+1} = Q/E_i$
Devolver $Q/E = Q/E_i$

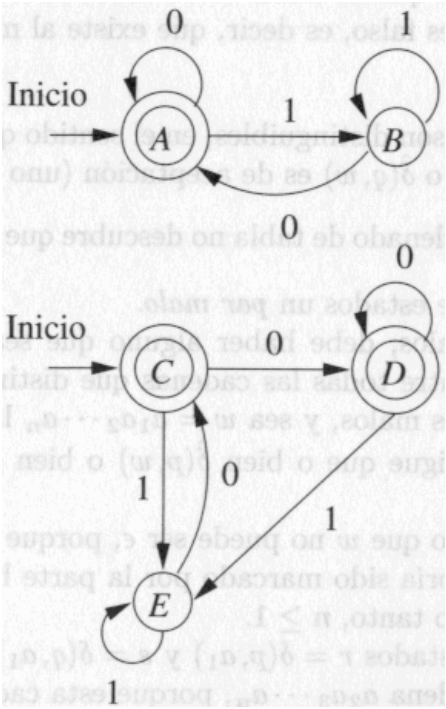
Equivalencia de estados

En el ejemplo de la figura, E es equivalente a A, B es equivalente a H y F es equivalente a D



Equivalencia de lenguajes regulares

- Sean *L* y *M* dos lenguajes representados de alguna forma
  - se convierten las representaciones en AFD
  - se comprueba si los estados iniciales de ambos AFD son equivalentes
  - si son equivalentes, entonces *L* = *M* y si no, serán diferentes
- Ejemplo:



Minimización de un AFD

Para todo AFD es posible encontrar un AFD equivalente con igual o menor número de estados que cualquier AFD que acepte el mismo lenguaje. Dicho AFD es único.

1. Se eliminan los estados no accesibles desde el inicial
2. Se divide el conjunto de estados *Q* en bloques de estados mutuamente equivalentes
3. Se construye el AFD mínimo *B* equivalente a *A*, utilizando los bloques de estados resultantes como estados del nuevo AFD
    1. La función de transición de *B* es $\gamma(S, a) = T$, donde *S* y *T* son bloques de estados. Cualquier estado del bloque *S* debe llevar con entrada *a* a un estado del bloque *T*. Si no fuese así, esos estados serían distinguibles, lo que no puede ser cierto por construcción de *B*
    2. el estado inicial de *B* es el bloque [único] que contiene el estado inicial de *A*
    3. el conjunto de estados de aceptación de *B* es el conjunto de bloques que contienen los estados de aceptación de *A*

Minimización de un AFD

Función Autómata-Mínimo (A): AFD_m

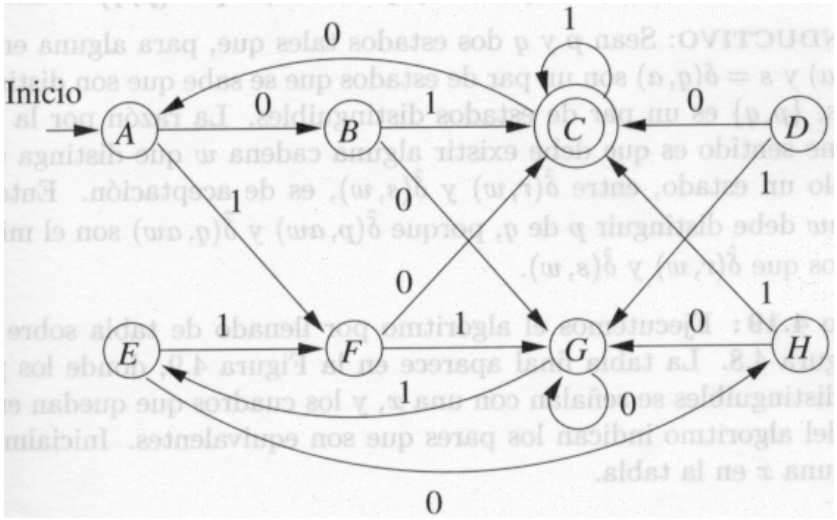
A: entrada a la función, AF definido por (Σ, Q, f, q_0, F)
AFD_m: salida de la función, autómata mínimo equivalente a A.

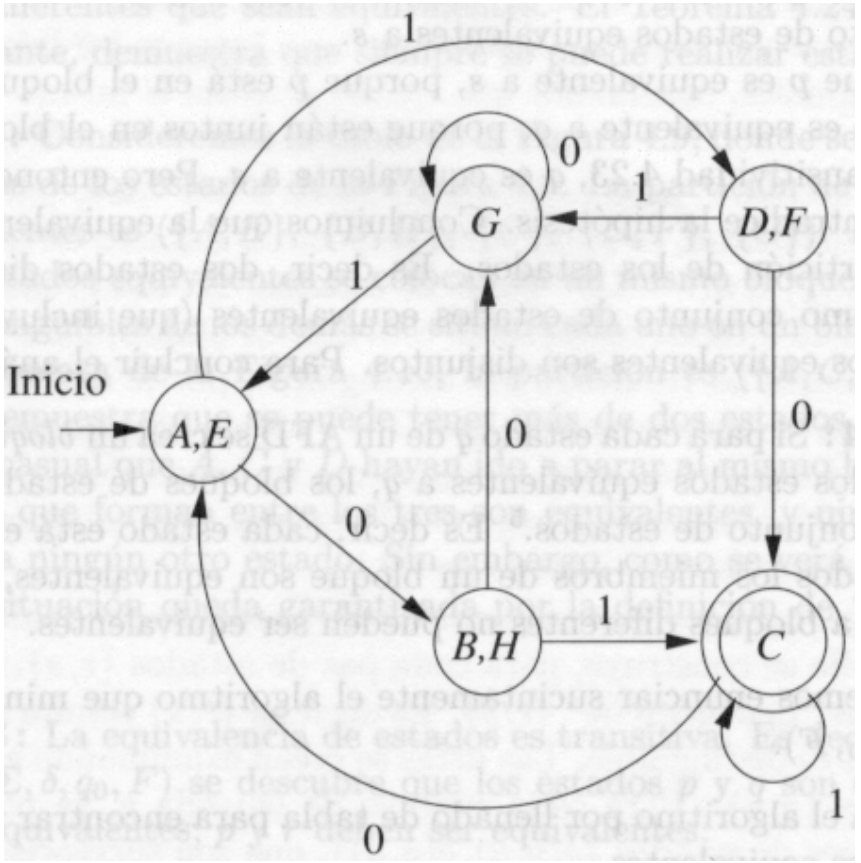
Eliminar de A todos los estados inaccesibles desde q_0;
Construir el AFD$_m$ $= (\Sigma, Q', f', q_0', F')$ donde

Q' =Conjunto-Cociente(A);
$q_0' = c_0$ si $q_0 \in c_0, c_0 \in Q'$;
$F' = \{c | \exists q \in c, q \in F\}$; y
$\forall a \in \Sigma, f'(c_i, a) = c_j$ si $\exists p \in c_j, q \in c_i | f(q, a) = p$

Problema

- Minimizar el siguiente AFD





Problemas

Dados los siguientes AFD, completar los AFD equivalentes mínimos

	0	1
->A	B	A
B	A	C
C	D	F
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

➔

H es no accesible
A={A, G}
B={B, F}
C={C, E}
D= {D}

	0	1
->A	B	
B	A	C
C	D	
*D		A

	0	1
->A	B	E
B	C	F
*C	D	H
D	E	H
E	F	I
*F	G	B
G	H	B
H	I	C
*I	A	E

➔

A={A, D, G}
B={B, E, H}
C={C, F, I}

	0	1
->A		B
B	C	
*C		B