



UNIVERSIDAD
DE GRANADA



Modelos de Computación

Grado en Ingeniería Informática

Tema 2 – Autómatas finitos y expresiones regulares

Este documento está protegido por la
Ley de Propiedad Intelectual (Real
Decreto Ley 1/1996 de 12 de abril).
Queda expresamente prohibido su uso o
distribución sin autorización del autor.

Manuel Pegalajar Cuéllar

manupc@ugr.es

Departamento de Ciencias de la
Computación e Inteligencia Artificial
<http://decsai.ugr.es>

Objetivos del tema

- Conocer los fundamentos de los autómatas finitos determinísticos.
- Conocer los fundamentos de los autómatas finitos no determinísticos.
- Conocer las relaciones entre gramáticas y autómatas.
- Conocer las expresiones regulares
- Conocer las relaciones entre gramáticas de tipo 3, autómatas y expresiones regulares



Anotación sobre estas diapositivas:

El contenido de estas diapositivas es esquemático y representa un apoyo para las clases presenciales teóricas. No se considera un sustituto para apuntes de la asignatura.

Se recomienda al alumno completar estas diapositivas con notas/apuntes propios, tomados en clase y/o desde la bibliografía principal de la asignatura.





Autómatas finitos y expresiones regulares



1. Autómatas finitos deterministas
2. Autómatas finitos no deterministas
3. Autómatas finitos con transiciones nulas
4. Expresiones regulares
5. Gramáticas regulares
6. Máquinas de estados finitos

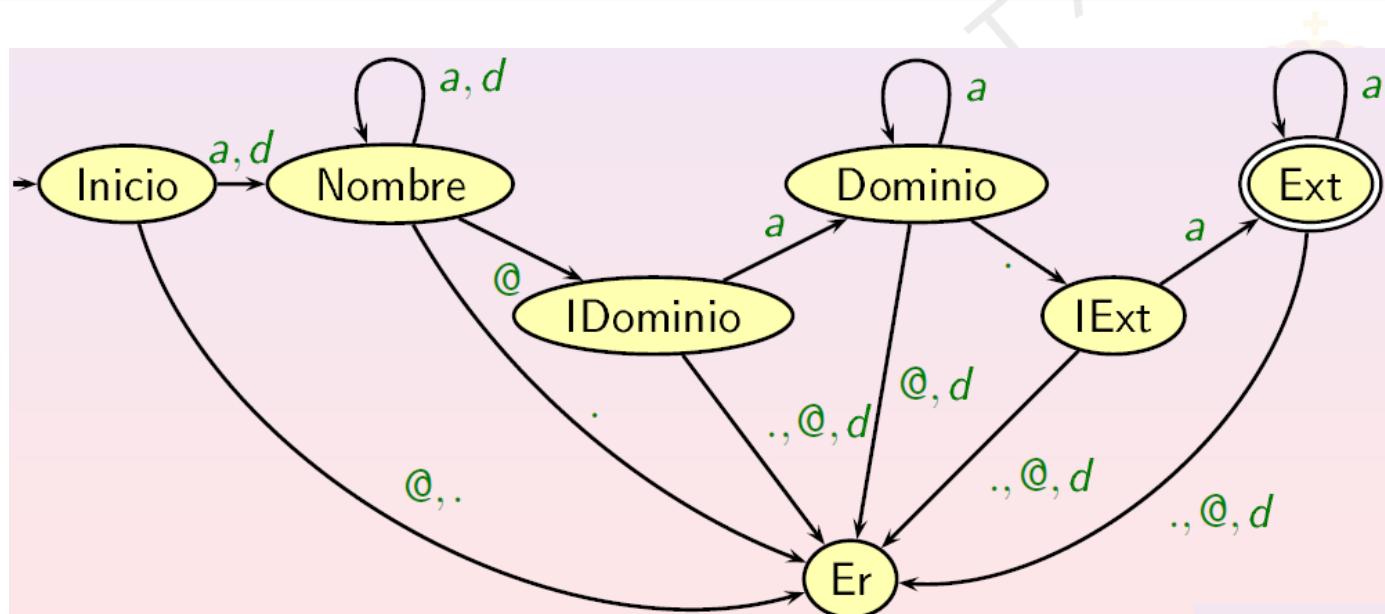
¿Por qué estudiar autómatas?

Los autómatas regulares son de extrema importancia en el diseño de soluciones para multiples aplicaciones:

- Software para el diseño y verificación de circuitos digitales.
- Construcción de analizadores léxicos de compiladores.
- Software para analizar grandes conjuntos de textos para buscar palabras, estructuras u otros patrones (p.e. páginas web).
- Software para comprobar la corrección de cualquier tipo de sistemas que tengan un número de estados diferentes (p.e. protocolos de comunicación)
- Y un muy largo etcétera...

Un ejemplo: Reconocimiento de e-mails bien escritos

Un email (simplificado) puede verse como una palabra de un lenguaje con el formato nombre@dominio.extensión, donde nombre, dominio y extensión son una secuencia de caracteres (a) y dígitos (d) que comienzan por un carácter.



Definición: Autómata finito determinista

Un autómata finito determinista (AFD) es una quintupla $M=(Q,A,\partial,q_0, F)$ donde:

- Q es un conjunto finito llamado conjunto de estados.
- A es un alfabeto llamado alfabeto de entrada
- ∂ es una aplicación llamada función de transición

$$\partial: Q \times A \rightarrow Q$$

- q_0 es un elemento de Q , llamado estado inicial
- F es un subconjunto de Q , llamado conjunto de estados finales

Un ejemplo:

Sea el autómata $M=(Q, A, \delta, q_0, F)$, donde

- $Q=\{q_0, q_1, q_2\}$
- $A=\{a, b\}$
- La función de transición viene dada por:

$$\begin{aligned}\delta(q_0, a) &= q_1, \quad \delta(q_0, b) = q_2, \\ \delta(q_1, a) &= q_1, \quad \delta(q_1, b) = q_2, \\ \delta(q_2, a) &= q_1, \quad \delta(q_2, b) = q_0\end{aligned}$$

- $F=\{q_1\}$

Definición: Diagrama de transición

Un diagrama de transición es una representación de un autómata utilizando grafos, donde:

- Hay un nodo por cada estado
- Por cada transición $\delta(q, a) = p$, donde q y p son estados y a un símbolo del alfabeto, hay un arco de q a p con la etiqueta a .
- El estado inicial está indicado con un ángulo entrante.
- Los estados finales están indicados con una doble circunferencia.

Un ejemplo:

Sea el autómata $M = (Q, A, \delta, q_0, F)$, donde

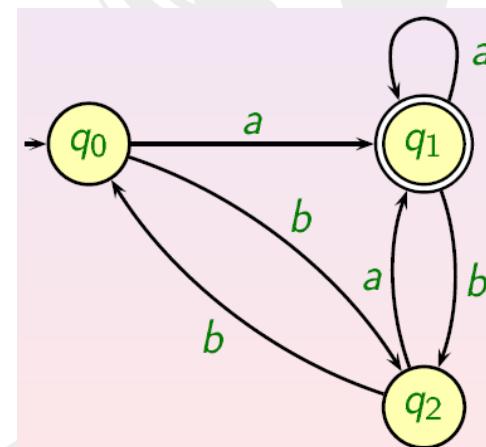
- $Q = \{q_0, q_1, q_2\}$ $A = \{a, b\}$
- La función de transición viene dada por:

$$\delta(q_0, a) = q_1, \quad \delta(q_0, b) = q_2,$$

$$\delta(q_1, a) = q_1, \quad \delta(q_1, b) = q_2,$$

$$\delta(q_2, a) = q_1, \quad \delta(q_2, b) = q_0$$

- $F = \{q_1\}$

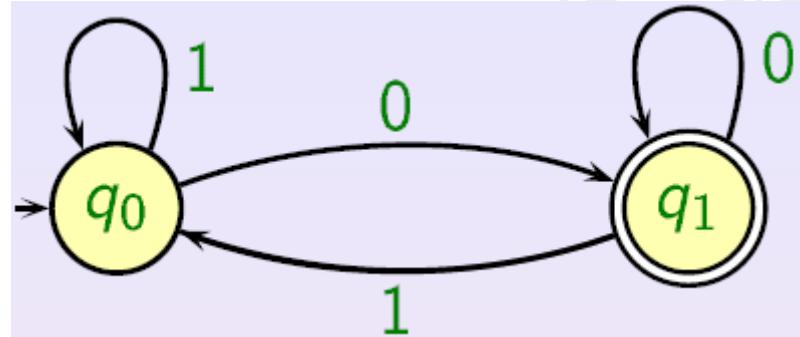


Trazas de palabras en un autómata

Para una palabra de entrada dada, comenzando desde el estado inicial, se siguen las aristas del grafo asociadas a cada símbolo de la palabra, de forma secuencial. Si al finalizar la traza el autómata se encuentra en un estado final, entonces la palabra es reconocida por el lenguaje

Un ejemplo:

Haremos la traza para la palabra 100 en el siguiente autómata:



Proceso de cálculo

- **Descripción Instantánea o Configuración:** Un elemento de $Q \times A^*$: (q, u) . Indica el estado actual en el que se encuentra el autómata, y la palabra que aún queda por procesar.
- **Configuración Inicial** para $u \in A^*$: (q_0, u)
- Relación o **paso de cálculo entre dos configuraciones**:

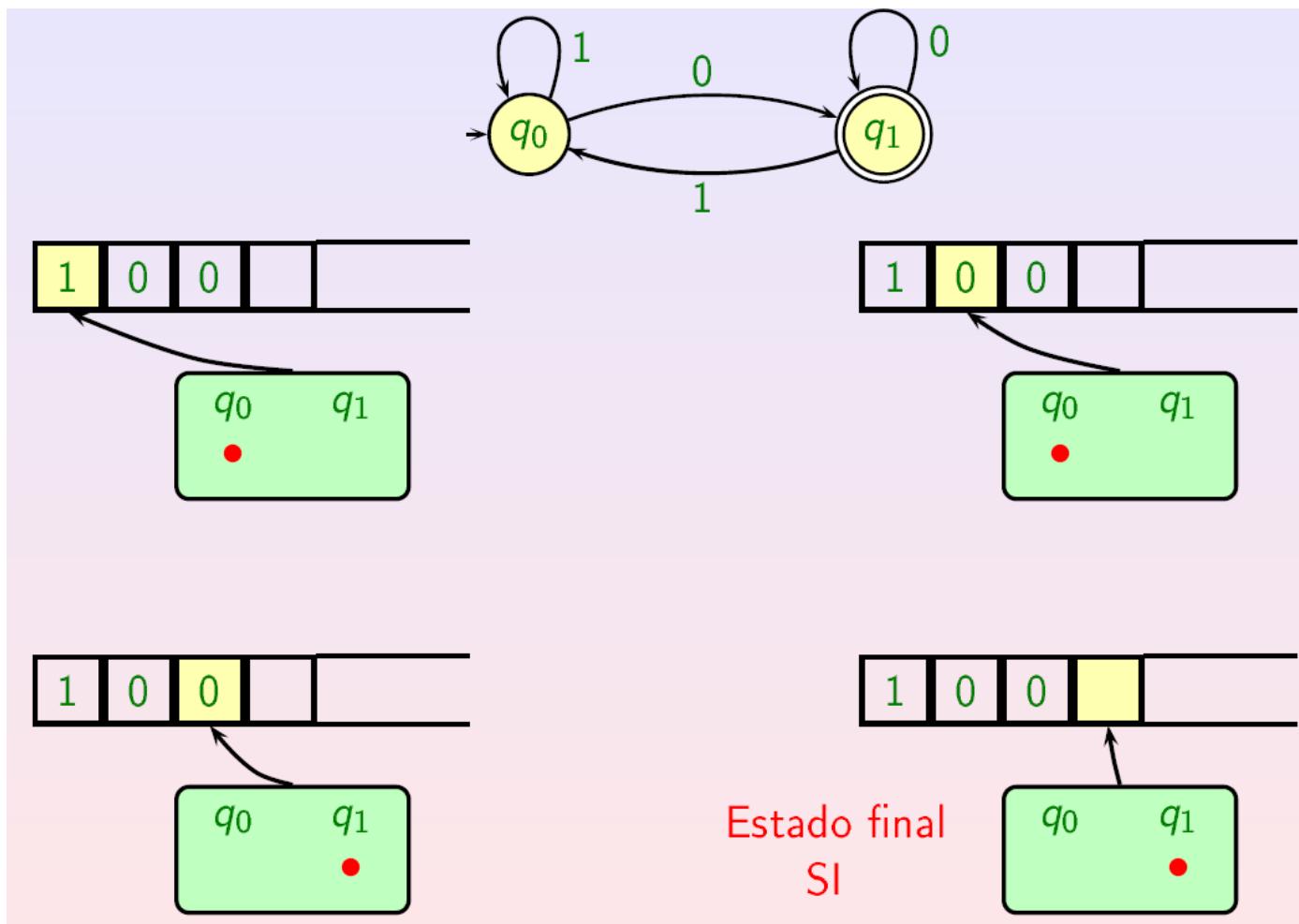
$$((q, au) \vdash (p, u)) \Leftrightarrow \delta(q, a) = p$$

(desde el estado q se puede pasar al estado p con el símbolo a si, y solo si, hay una transición de q a p con el símbolo a .)

De una configuración sólo se puede pasar a lo máximo a una configuración en un paso de cálculo en un automata finito determinista.

Un ejemplo:

Haremos la traza para la palabra 100 en el siguiente autómata:



Lenguaje aceptado por un AFD

- La relación de cálculo entre dos configuraciones (q,u) y (p,v) se denota como:

$$((q,u) \xrightarrow{*} (p,v))$$

Existe una relación de cálculo entre dos configuraciones si hay una secuencia de configuraciones C_1, C_2, \dots, C_n tal que $C_1 = (q,u)$ y $C_n = (p,v)$ y, además:

$$\forall i \leq n-1, C_i \vdash C_{i+1}$$

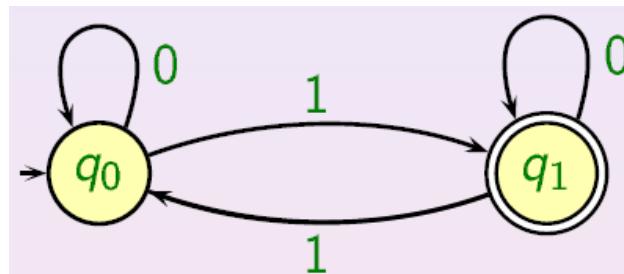
- Se define el **lenguaje L aceptado por un autómata M** como:

$$L(M) = \{u \in A^* : (q_0, u) \xrightarrow{*} (q, \varepsilon), q \in F\}$$

- Las palabras de $L(M)$ se dice que son **aceptadas por el autómata**.

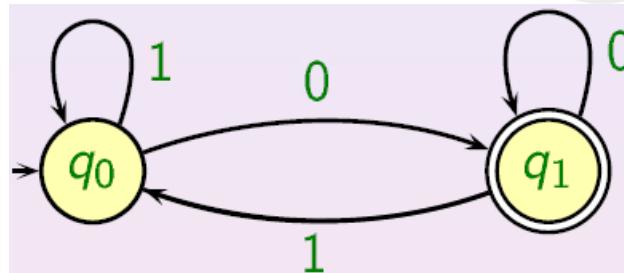
Un ejemplo:

Para el autómata siguiente, el lenguaje aceptado es aquel que contiene una secuencia de 0's y 1's (al menos un 1, no necesariamente 0's), con un número impar de 1's.



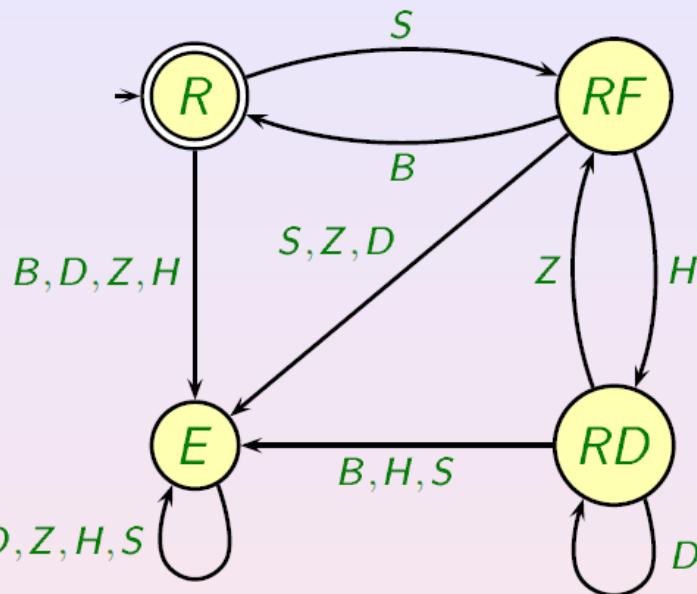
Otro ejemplo:

Para el autómata siguiente, el lenguaje aceptado es aquel que contiene una secuencia de 0's y 1's (no necesariamente debe tener 1's), pero sí necesariamente al menos un 0, y siempre acaba en 0.



Otro ejemplo:

Modelado de procesos para realizar transferencias de datos correctamente:



R: Estado de espera

RD: Recepción de datos

S: Comienza recepción

H: Cabecera de fichero

D: Datos

RF: Estado de recepción de ficheros

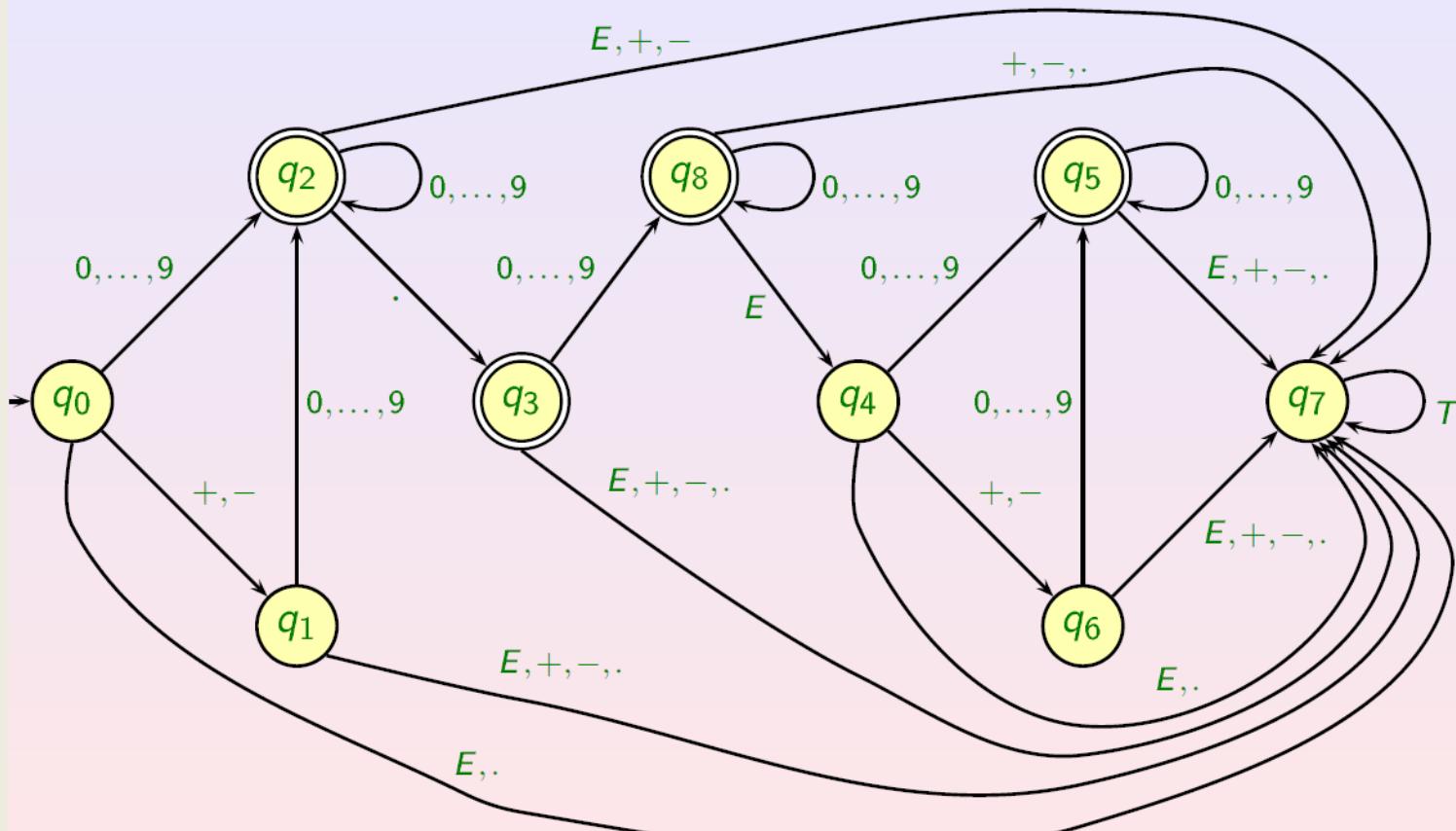
E: Error

B: Fin de recepción

Z: Fin de fichero

Otro ejemplo:

Autómata para reconocer números en notación científica (ej. 2.19E+12):



Otro ejemplo (continuación):

Gramática $G=(V,T,P,S)$ asociada al ejemplo anterior:

- $T = \{+, -, E, 0, 1, \dots, 9, .\}$
- $V = \{\langle \text{Signo} \rangle, \langle \text{Digito} \rangle, \langle \text{Natural} \rangle, \langle \text{Entero} \rangle, \langle \text{Real} \rangle\}$
- $S = \langle \text{Real} \rangle$
- P contiene las siguientes producciones:
 - $\langle \text{Signo} \rangle \rightarrow + | -$
 - $\langle \text{Digito} \rangle \rightarrow 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$
 - $\langle \text{Natural} \rangle \rightarrow \langle \text{Digito} \rangle | \langle \text{Digito} \rangle \langle \text{Natural} \rangle$
 - $\langle \text{Entero} \rangle \rightarrow \langle \text{Natural} \rangle | \langle \text{Signo} \rangle \langle \text{Natural} \rangle$
 - $\langle \text{Real} \rangle \rightarrow \langle \text{Entero} \rangle | \langle \text{Entero} \rangle .$
 - $\langle \text{Real} \rangle \rightarrow \langle \text{Entero} \rangle . \langle \text{Natural} \rangle$
 - $\langle \text{Real} \rangle \rightarrow \langle \text{Entero} \rangle . \langle \text{Natural} \rangle E \langle \text{Entero} \rangle$

Moraleja:

Todo autómata finito determinista (AFD) tiene su gramática asociada.
Además, todos los lenguajes regulares también tienen un AFD asociado.



Autómatas finitos y expresiones regulares



1. Autómatas finitos deterministas
2. Autómatas finitos no deterministas
3. Autómatas finitos con transiciones nulas
4. Expresiones regulares
5. Gramáticas regulares
6. Máquinas de estados finitos

Definición: Autómata finito no determinista

Un autómata finito no determinista (AFND) es una quintupla $M=(Q,A,\partial,q_0, F)$ donde:

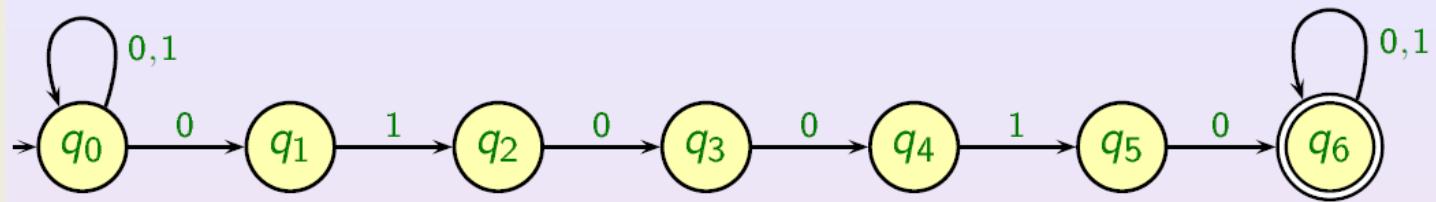
- Q es un conjunto finito llamado conjunto de estados.
- A es un alfabeto llamado alfabeto de entrada
- ∂ es una aplicación llamada función de transición

$$\partial: Q \times A \rightarrow P(Q)$$

Donde $P(Q)$ representa al conjunto de las partes de Q .

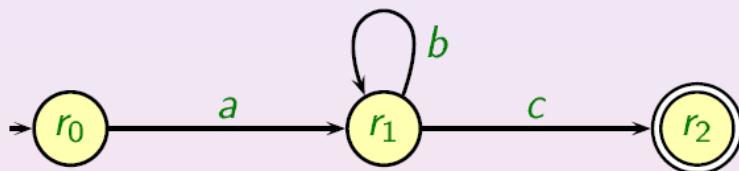
- q_0 es un elemento de Q , llamado estado inicial
- F es un subconjunto de Q , llamado conjunto de estados finales

Ejemplo:



- Se pueden usar también diagramas de transición.
- **Puede haber estados que para una entrada tenga dos transiciones (Ej. q_0)**
Transición $\delta(q_0, 0) = \{q_0, q_1\}$
- **Puede haber estados que para una entrada no tengan ninguna transición (Ej. q_1 no puede leer un 0)-**
- El autómata acepta el conjunto de palabras que tienen a 010010 como subcadena: palabras que se pueden leer pasando de q_0 a un estado final.

Ejemplo:



- Autómata que acepta cadenas formadas por una **a**, seguida de un número indefinido de **b**'s, y terminadas en una **c**.

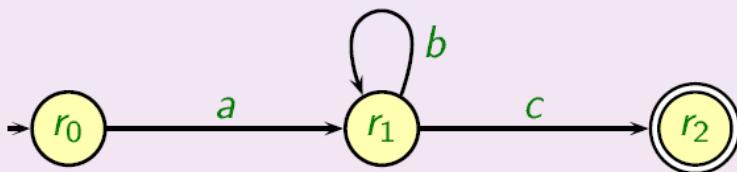


Transformación simple de AFND a AFD

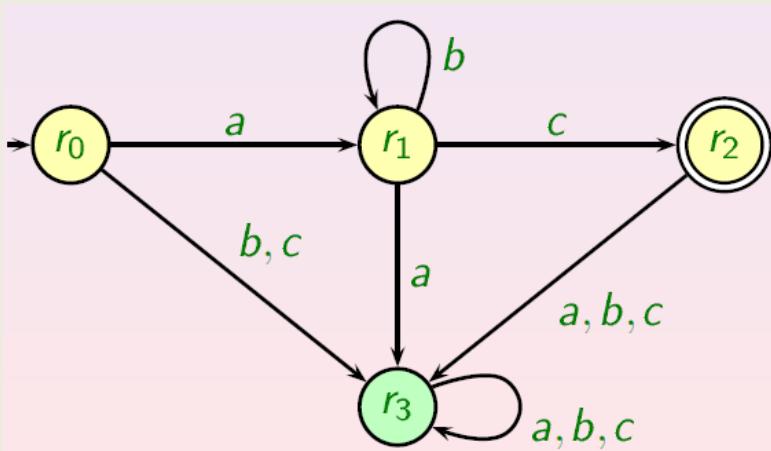
Sólo cuando no hay estados que para un mismo símbolo de entrada haya varias transiciones:

Un AFND se puede transformar fácilmente a un AFD, simplemente añadiendo un nuevo estado (denominado **estado de error**) y completando las transiciones que falten con transiciones hacia el estado de error.

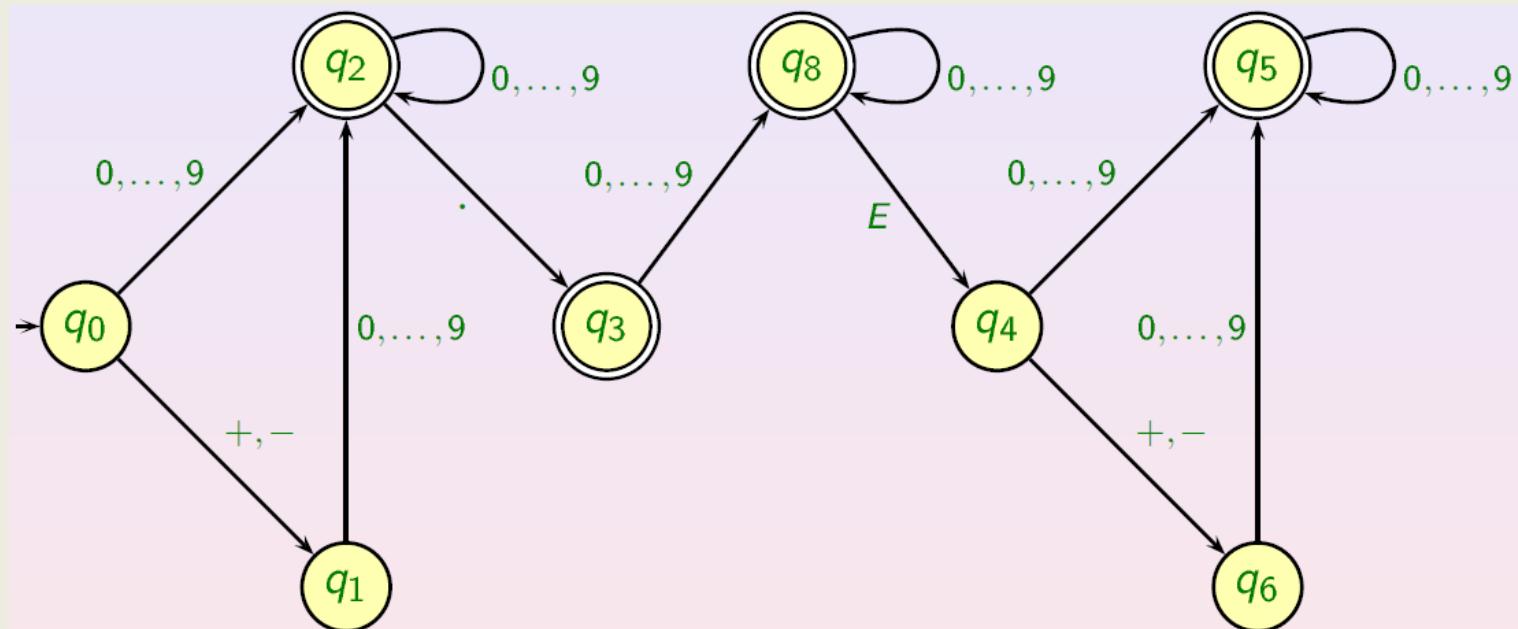
Ejemplo:



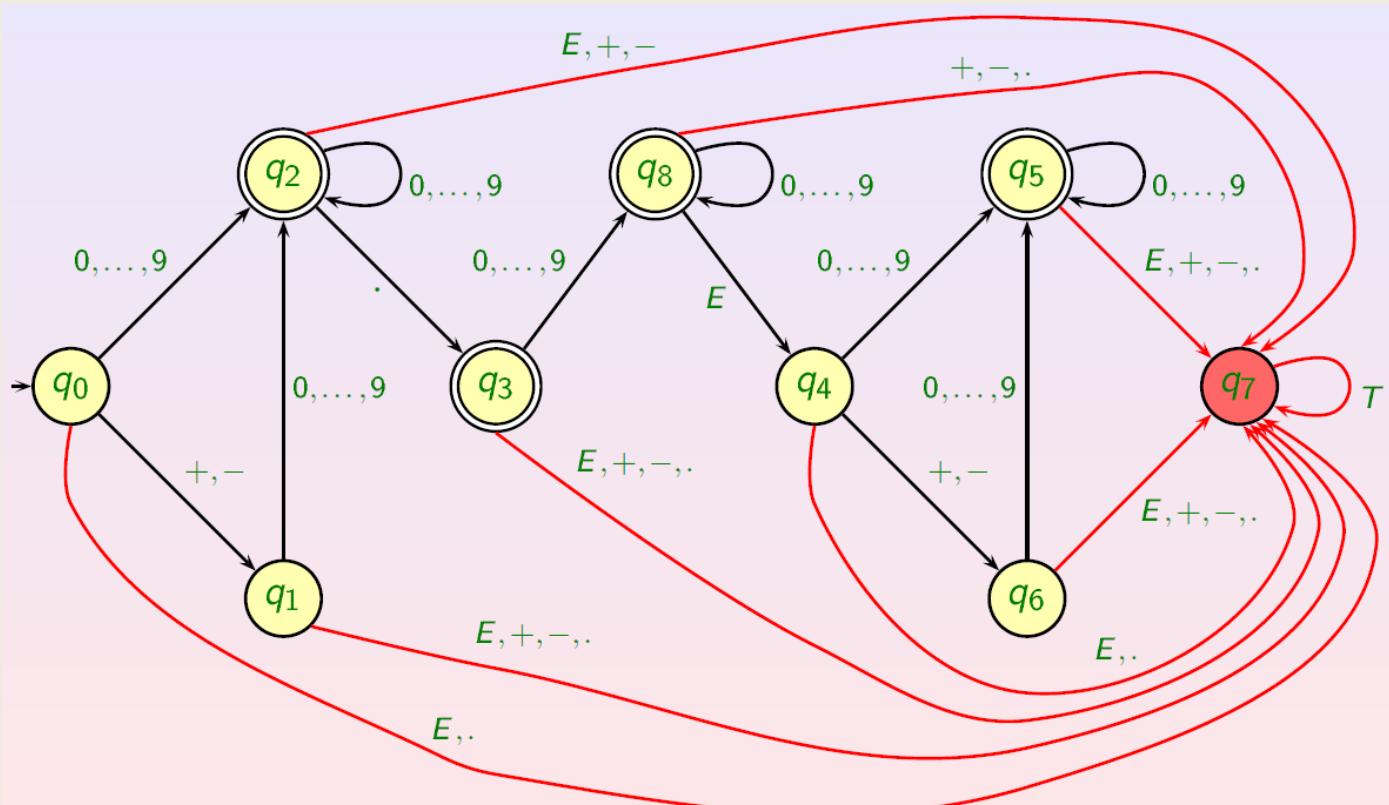
- Su paso a AFD es fácil: No hay estados que tengan varias transiciones para el mismo símbolo. Se crea el estado de error (r_3) y se completan las transiciones que faltan.



Ejemplo: AFND para reconocer números en not. científica



Ejemplo: Su paso a AFD



Proceso de cálculo

- **Descripción Instantánea o Configuración:** Un elemento de $Q \times A^*$: (q, u) . Indica el estado actual en el que se encuentra el autómata, y la palabra que aún queda por procesar.
- **Configuración Inicial** para $u \in A^*$: (q_0, u)
- Relación o **paso de cálculo entre dos configuraciones**:

$$((q, au) \vdash (p, u)) \Leftrightarrow p \in \delta(q, a)$$

(desde el estado q se puede pasar al estado p con el símbolo a si, y solo si, hay una transición de q con el símbolo a que contenga a p)

De una configuración se puede pasar a varias configuraciones distintas en un mismo paso de cálculo en un automata finito no determinista.

De hecho, podemos hasta no pasar a ninguna configuración.

Lenguaje aceptado por un AFND

- La relación de cálculo entre dos configuraciones (q,u) y (p,v) se denota como:

$$((q,u) \xrightarrow{*} (p,v))$$

Existe una relación de cálculo entre dos configuraciones si hay una secuencia de configuraciones C_1, C_2, \dots, C_n tal que $C_1 = (q,u)$ y $C_n = (p,v)$ y, además:

$$\forall i \leq n-1, C_i \vdash C_{i+1}$$

- Se define el **lenguaje L aceptado por un autómata M** como:
- $$L(M) = \{u \in A^* : \exists q \in F, (q_0, u) \xrightarrow{*} (q, \varepsilon)\}$$
- Las palabras de $L(M)$ se dice que son **aceptadas por el autómata**.

Definiciones

Dado un autómata $M=(Q, A, \delta, q_0, F)$. Definimos a $\delta^*(B, \cdot)$ como:

$$\delta^*(B, a) = \bigcup_{q \in B} \delta(q, a) \text{ donde } a \in A, B \in P(Q)$$

$$\delta^*(B, \varepsilon) = B$$

$$\delta^*(B, au) = \delta^*(\delta^*(B, a), u), \text{ donde } a \in A, B \in P(Q), u \in A^*$$

$$\delta^*(q, u) = \delta^*(\{q\}, u), \text{ donde } q \in Q$$

$\delta^*(B, u)$ representa el conjunto de todos los estados a los que se puede llegar desde cualquiera de los estados de B , después de leer la palabra u .

Por tanto, podemos también definir $L(M)$ como:

$$L(M) = \{u \in A^* : \delta^*(q_0, u) \cap F \neq \emptyset\}$$

Equivalencia entre AFD y AFND

- Todo AFD es también un AFND: Él mismo, donde $\delta(q,a)$ tiene un único estado.
- Por tanto, todo lenguaje aceptado por un AFD lo es también para algún AFND.

Al revés también tenemos equivalencia:

- Todo lenguaje aceptado por un AFND lo deberá ser también para un AFD.

Por tanto, con los AFND no podemos representar más lenguajes que con los AFD. Simplemente, tenemos una mayor variedad para representar un mismo lenguaje.

Paso de AFND a AFD

Dado un AFND $M = (Q, A, \delta, q_0, F)$. Denominaremos como $\bar{M} = (\bar{Q}, A, \bar{\delta}, \overline{q_0}, \bar{F})$ al **AFD asociado a M**, donde:

$$\bar{Q} = P(Q)$$

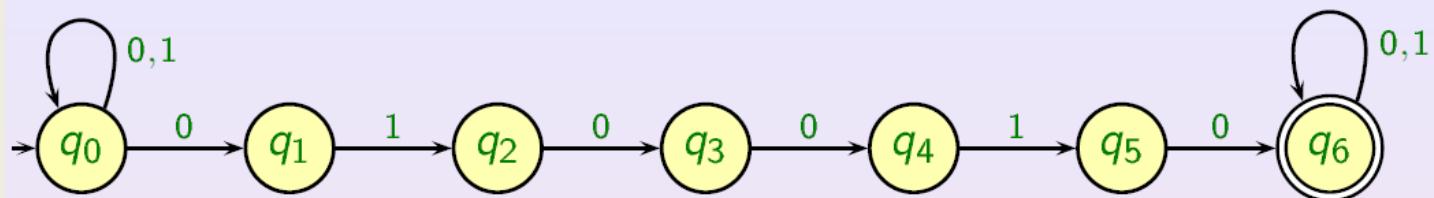
$$\overline{q_0} = \{q_0\}$$

$$\bar{\delta}(B, a) = \delta^*(B, A) = \bigcup_{q \in B} \delta(q, a)$$

$$\bar{F} = \{B \in P(Q) \mid B \cap F \neq \emptyset\}$$

Idea básica: Al AFND se le asocia un AFD cuyos estados representan el recorrido de todos los caminos posibles al mismo tiempo.

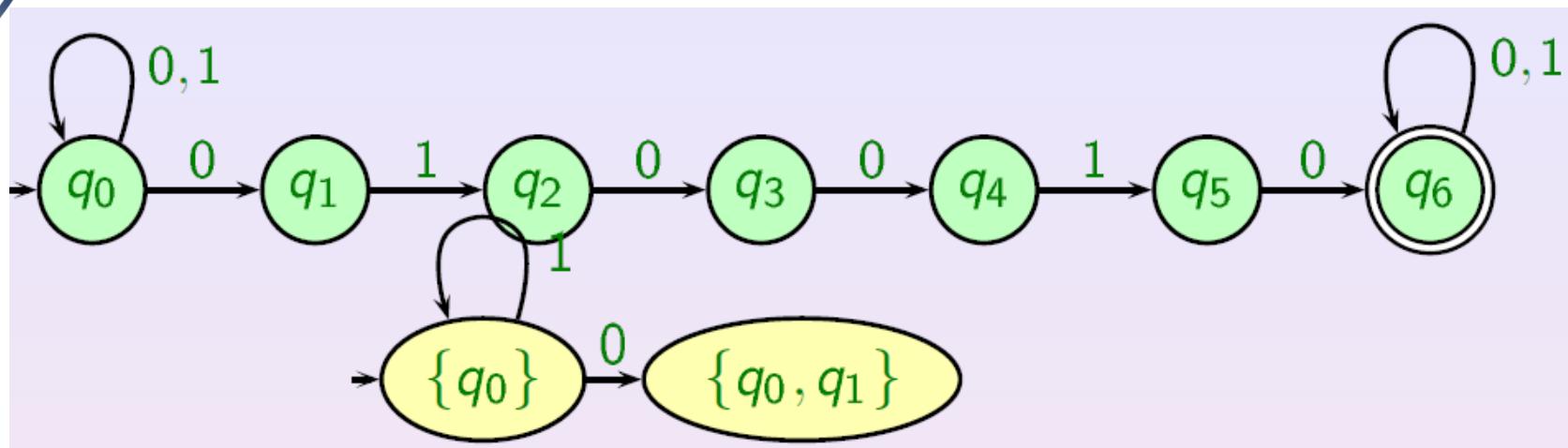
Ejemplo: Pasar el siguiente AFND a AFD

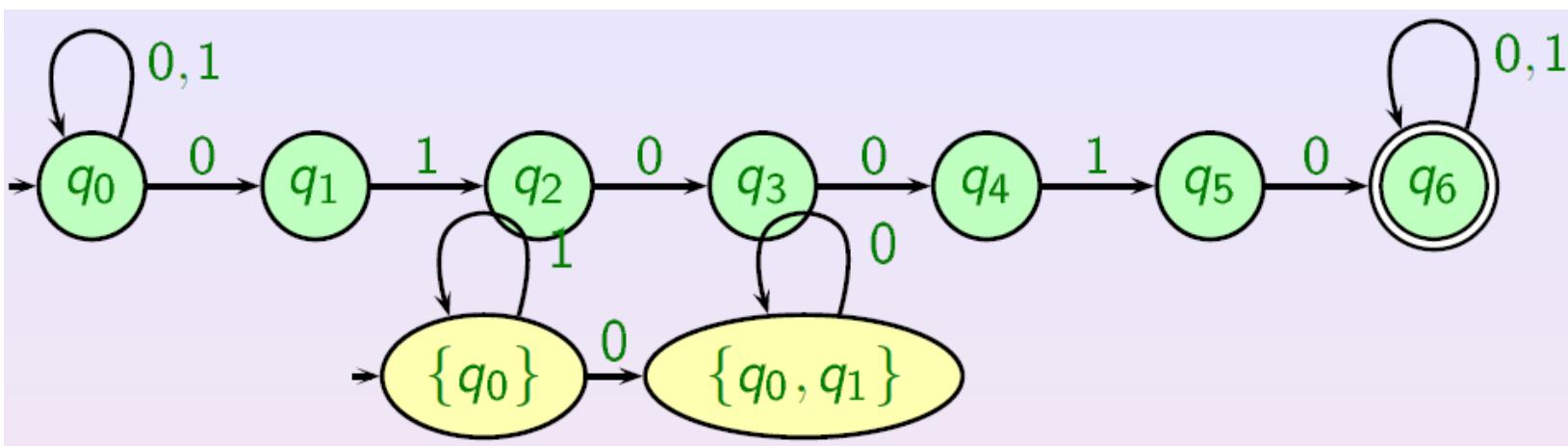


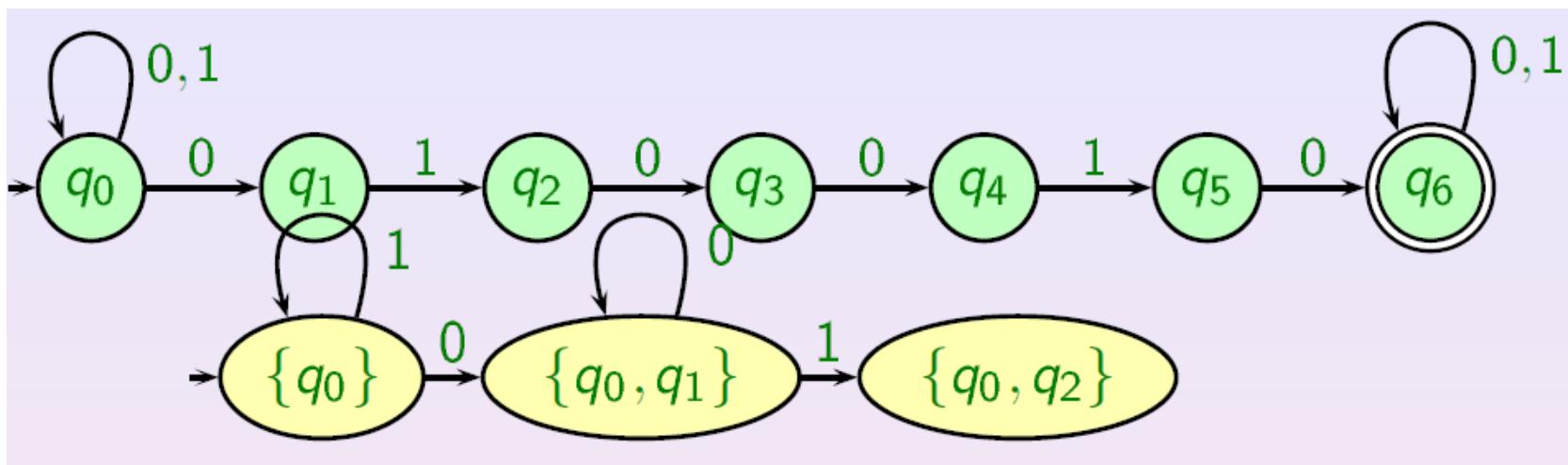
- Comenzamos con el conjunto de estados iniciales como el estado inicial del AFD:

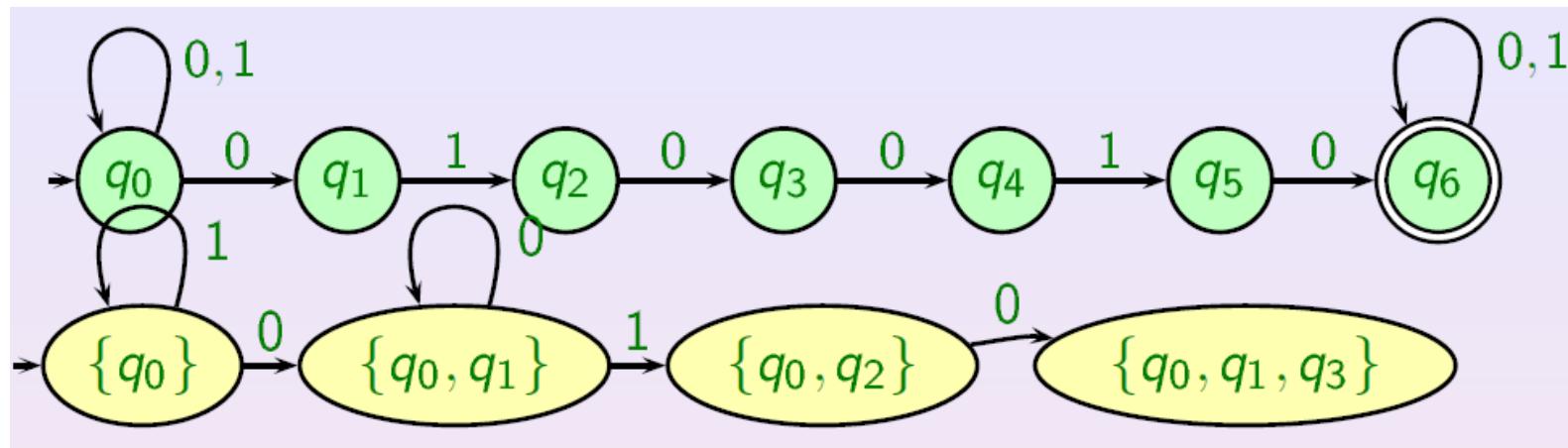


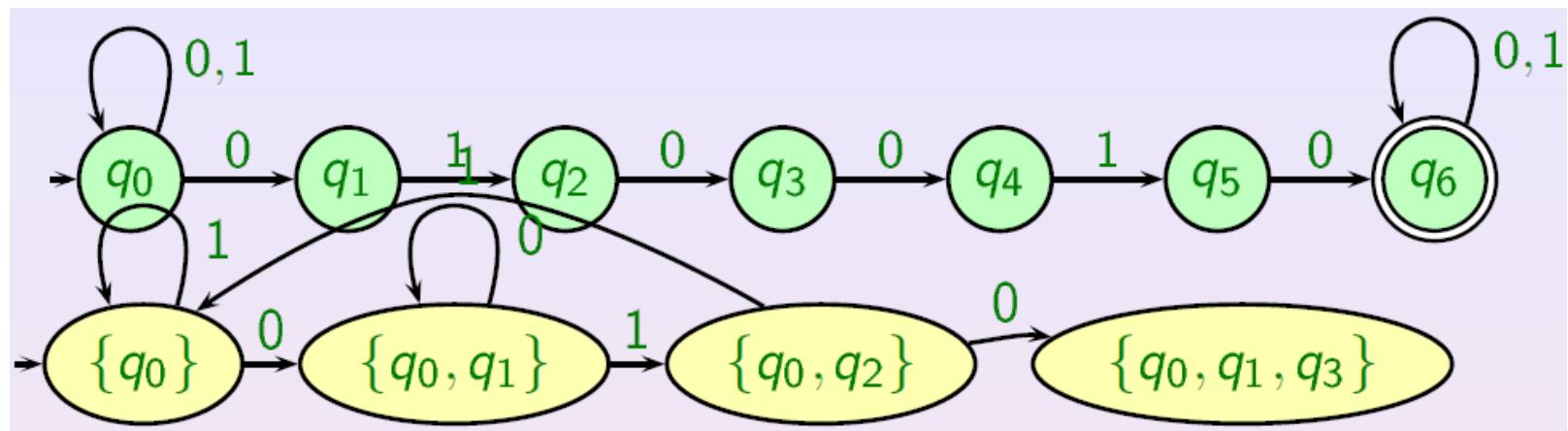
- Seguidamente, construiremos los estados del AFD agrupando aquellos estados del AFND a los que llegan las transiciones

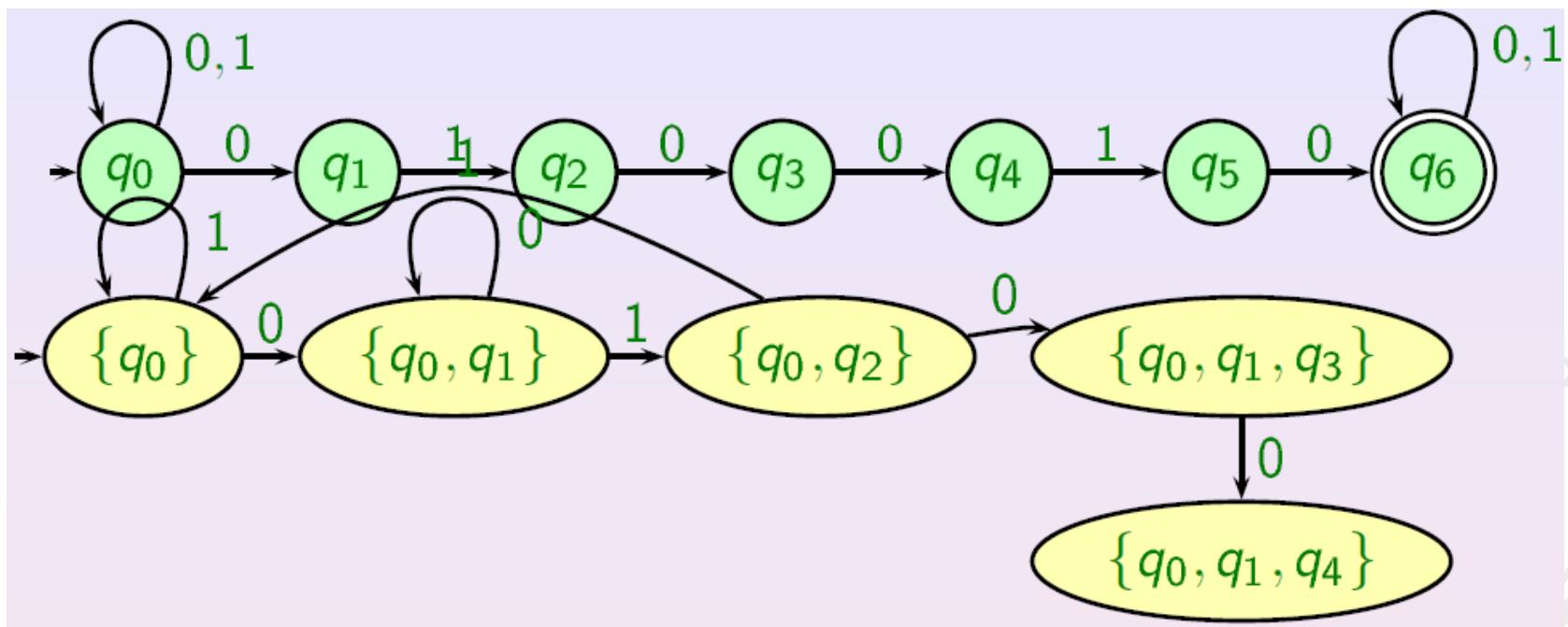


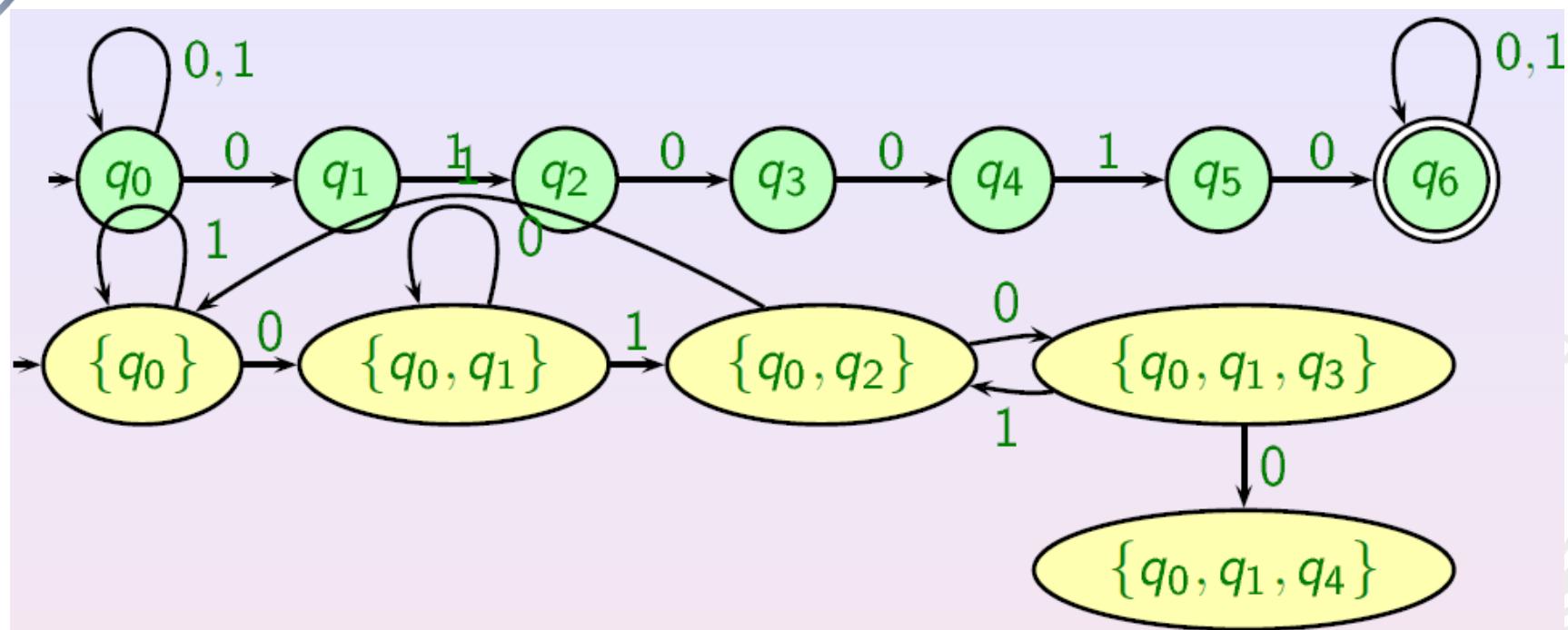


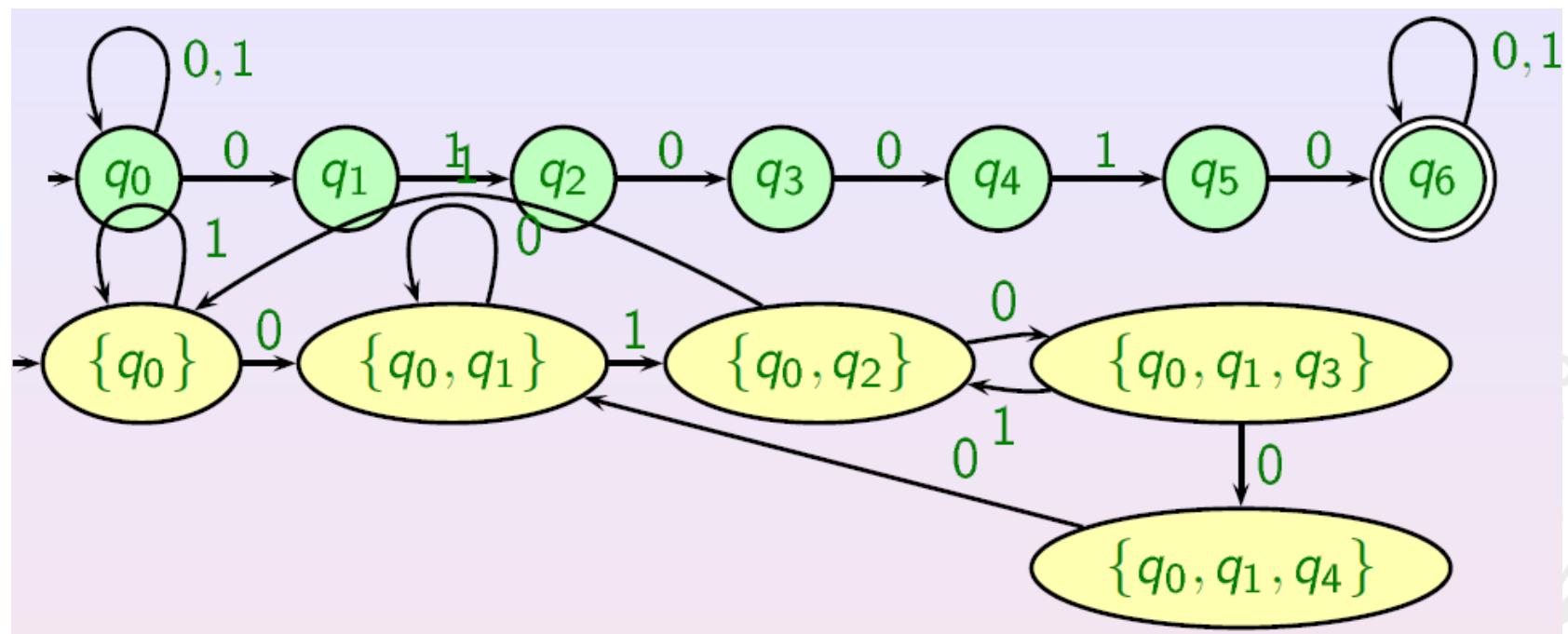


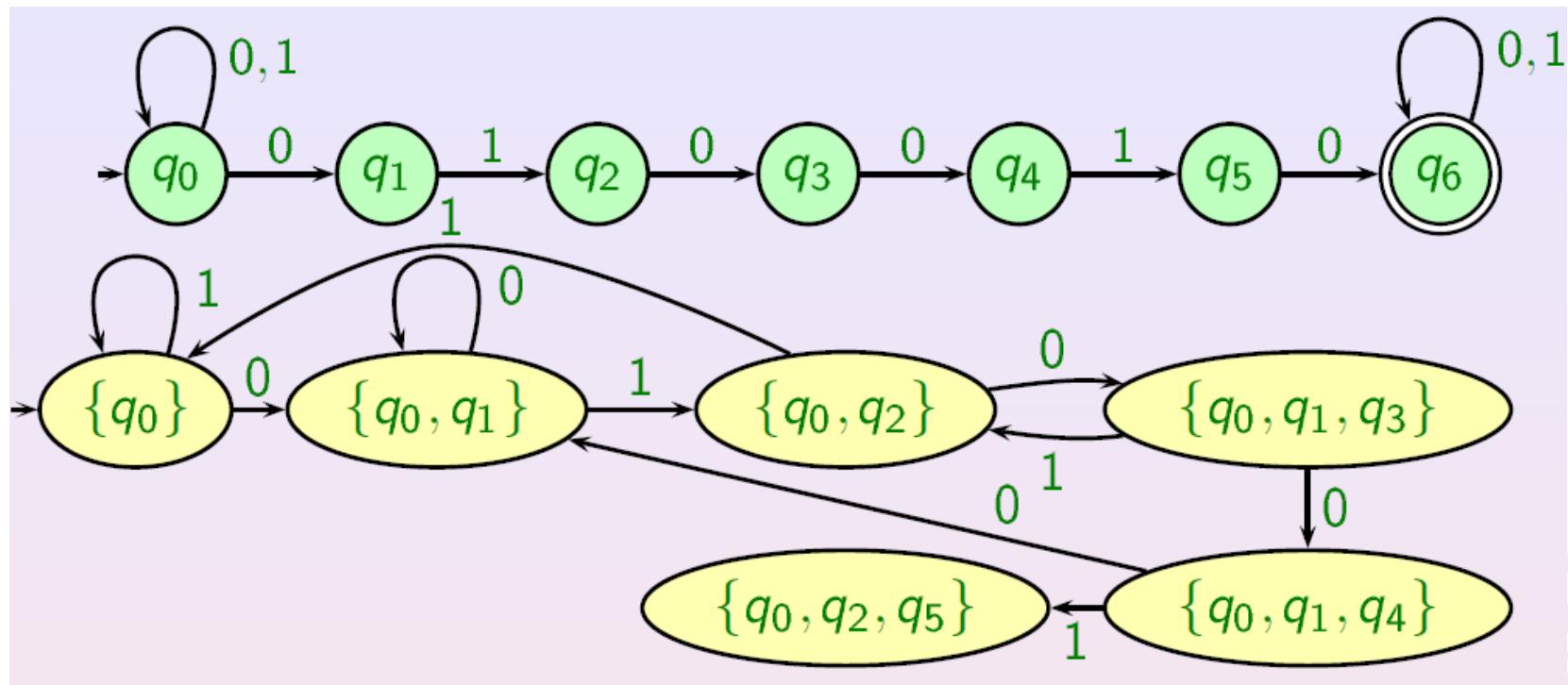


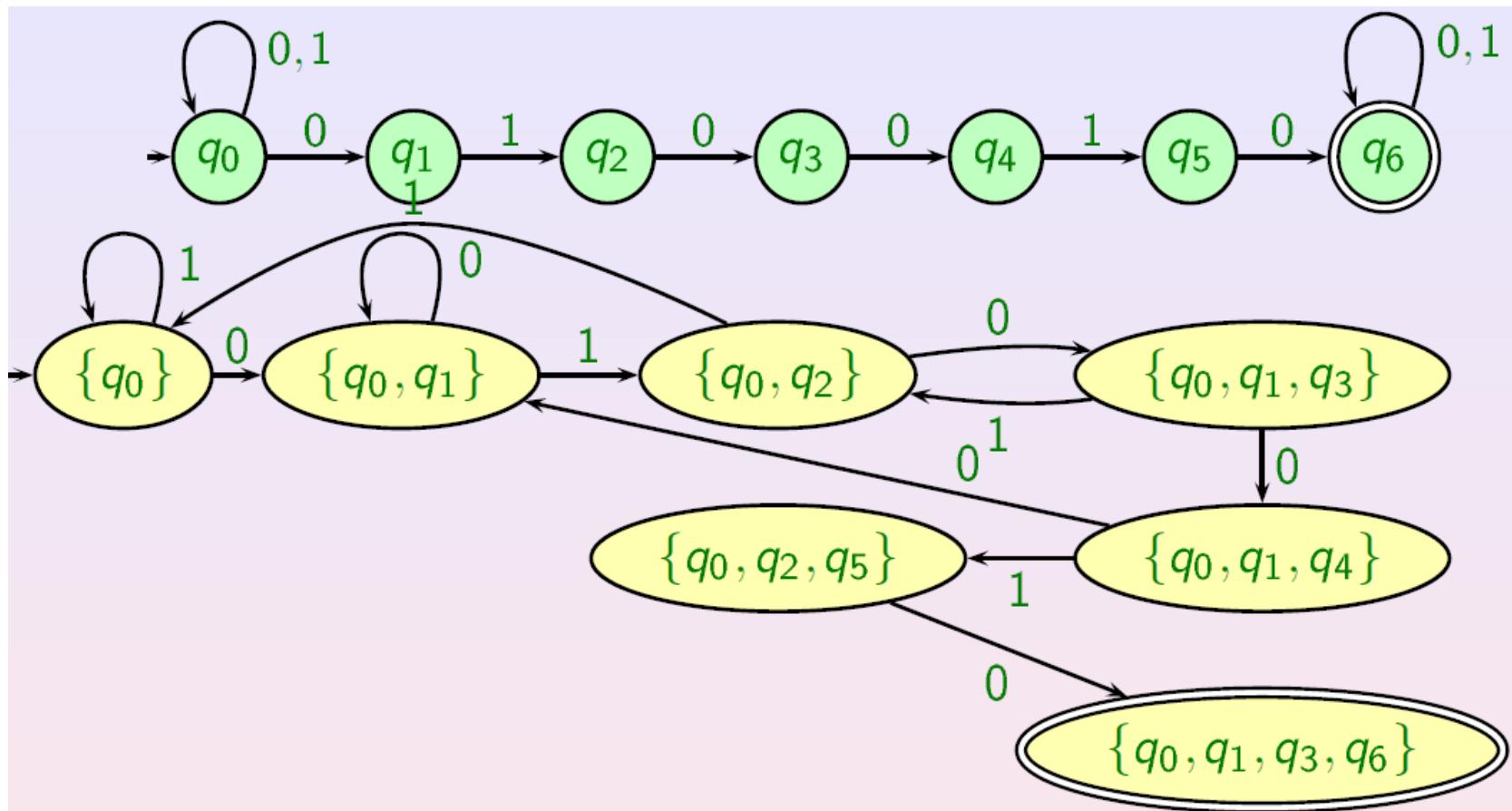


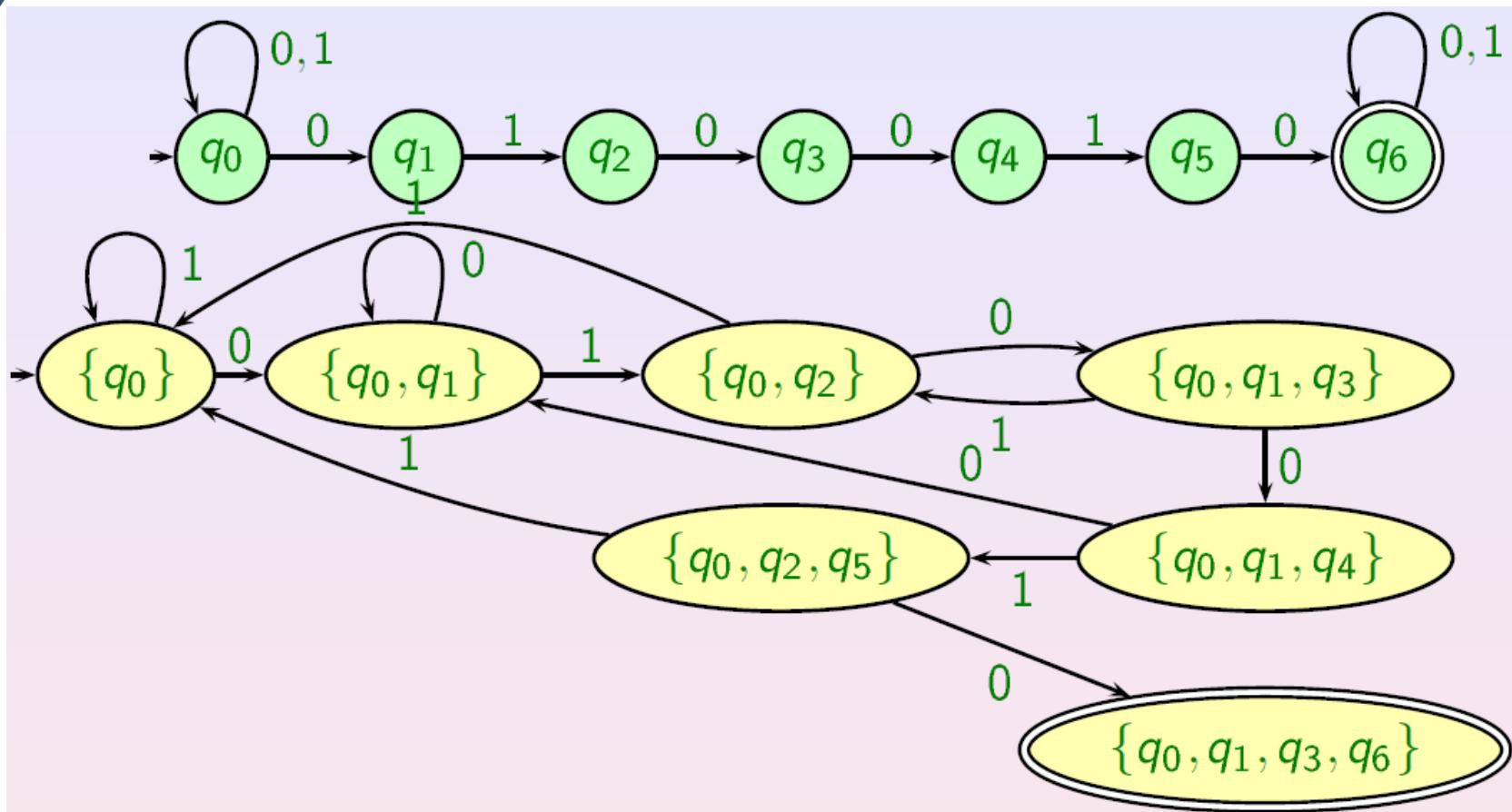


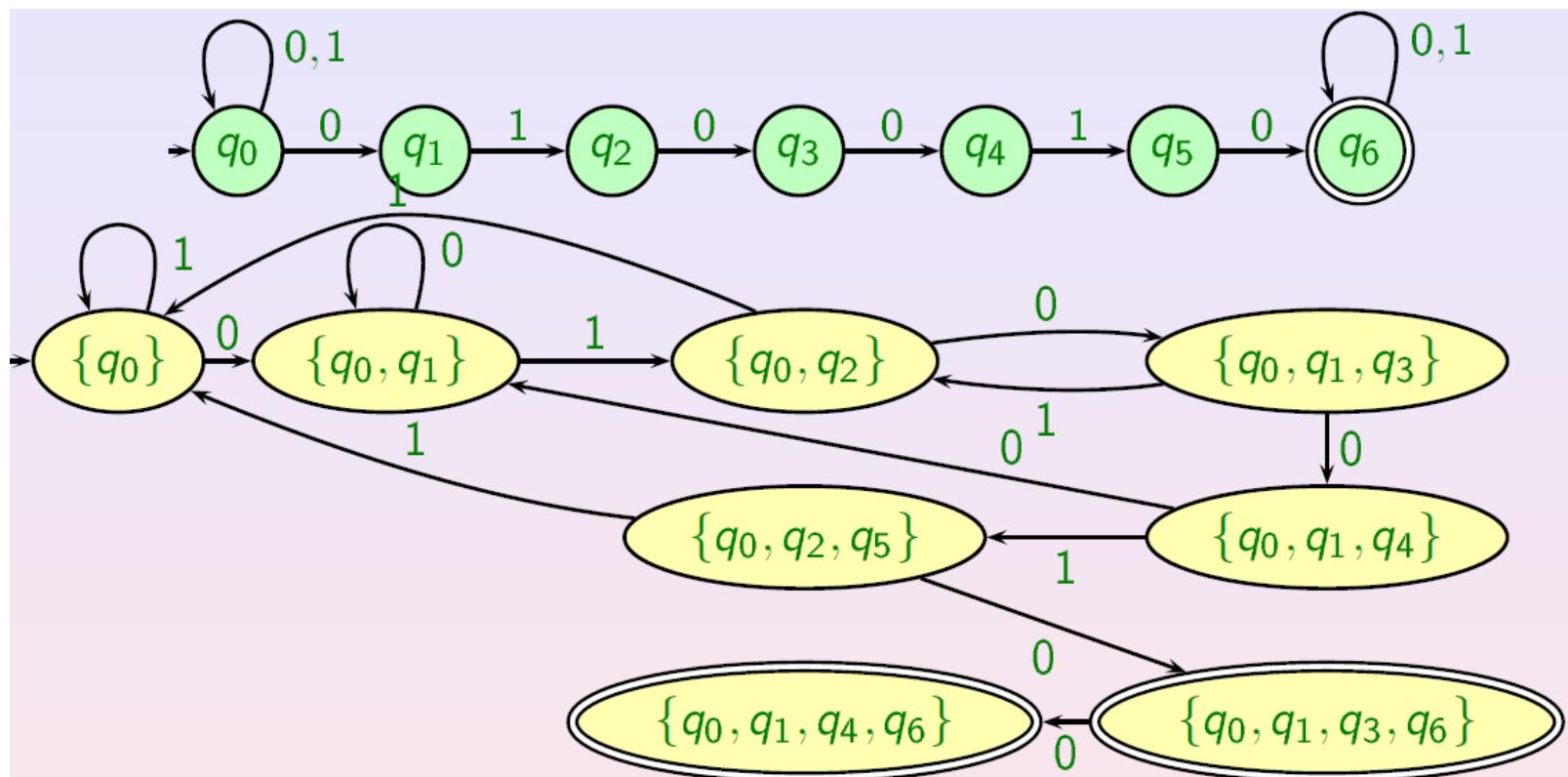


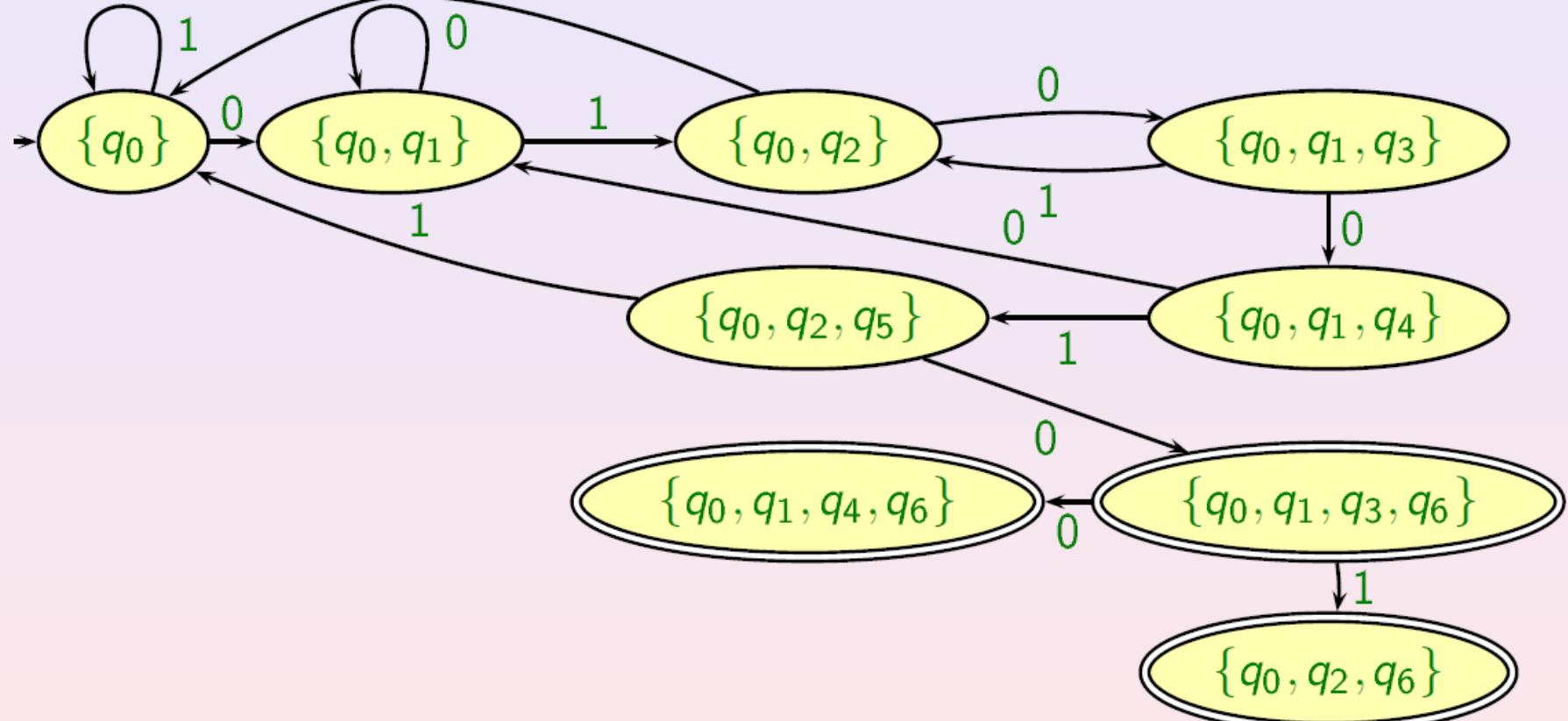


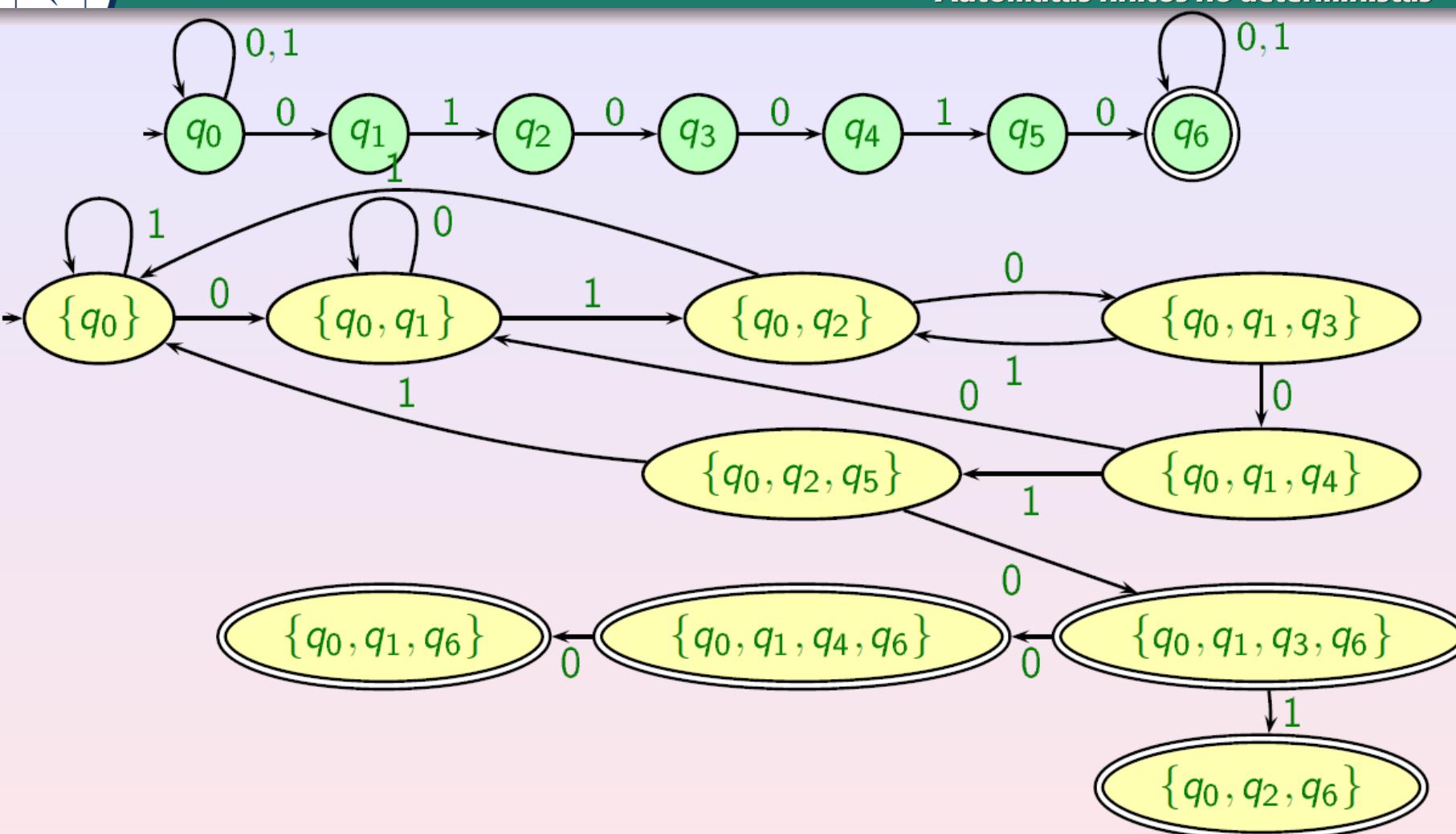


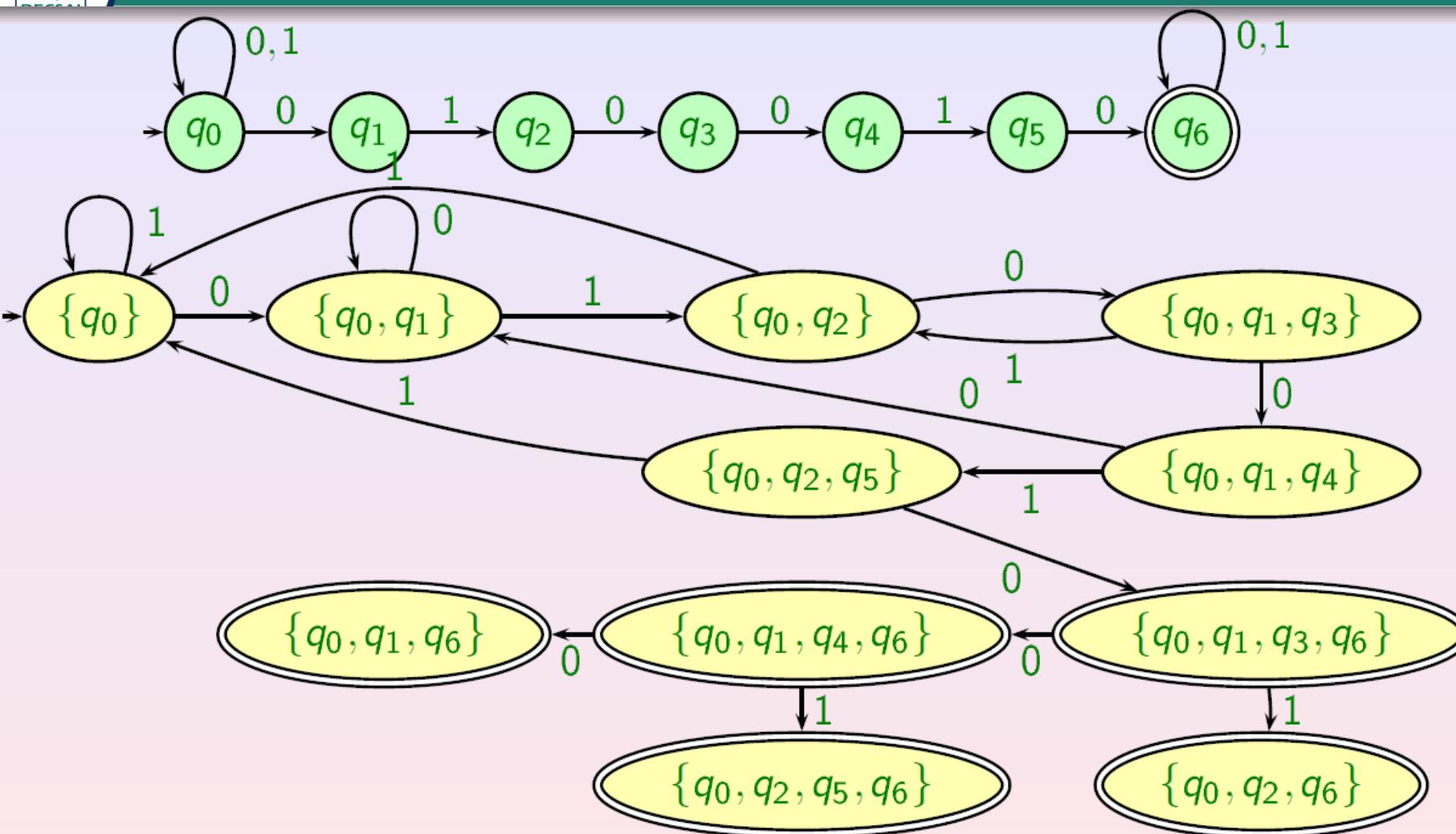


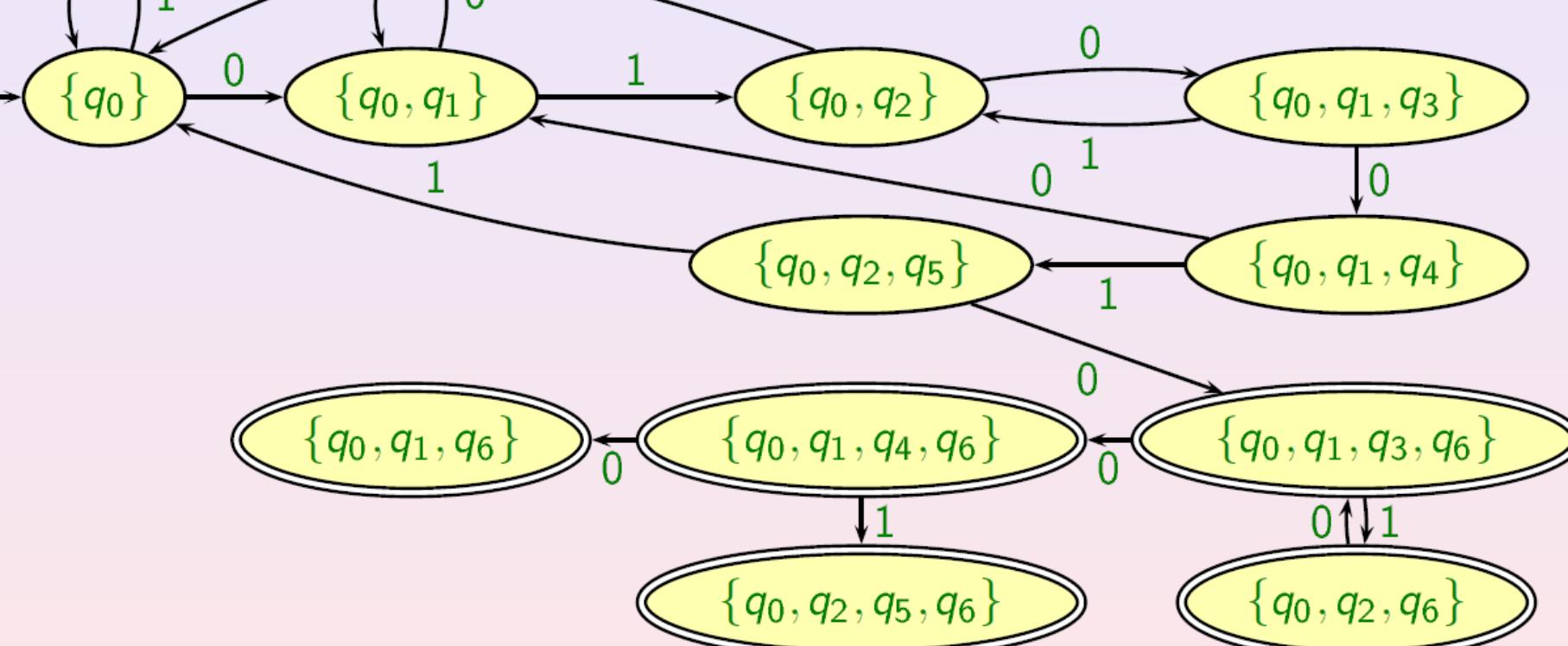


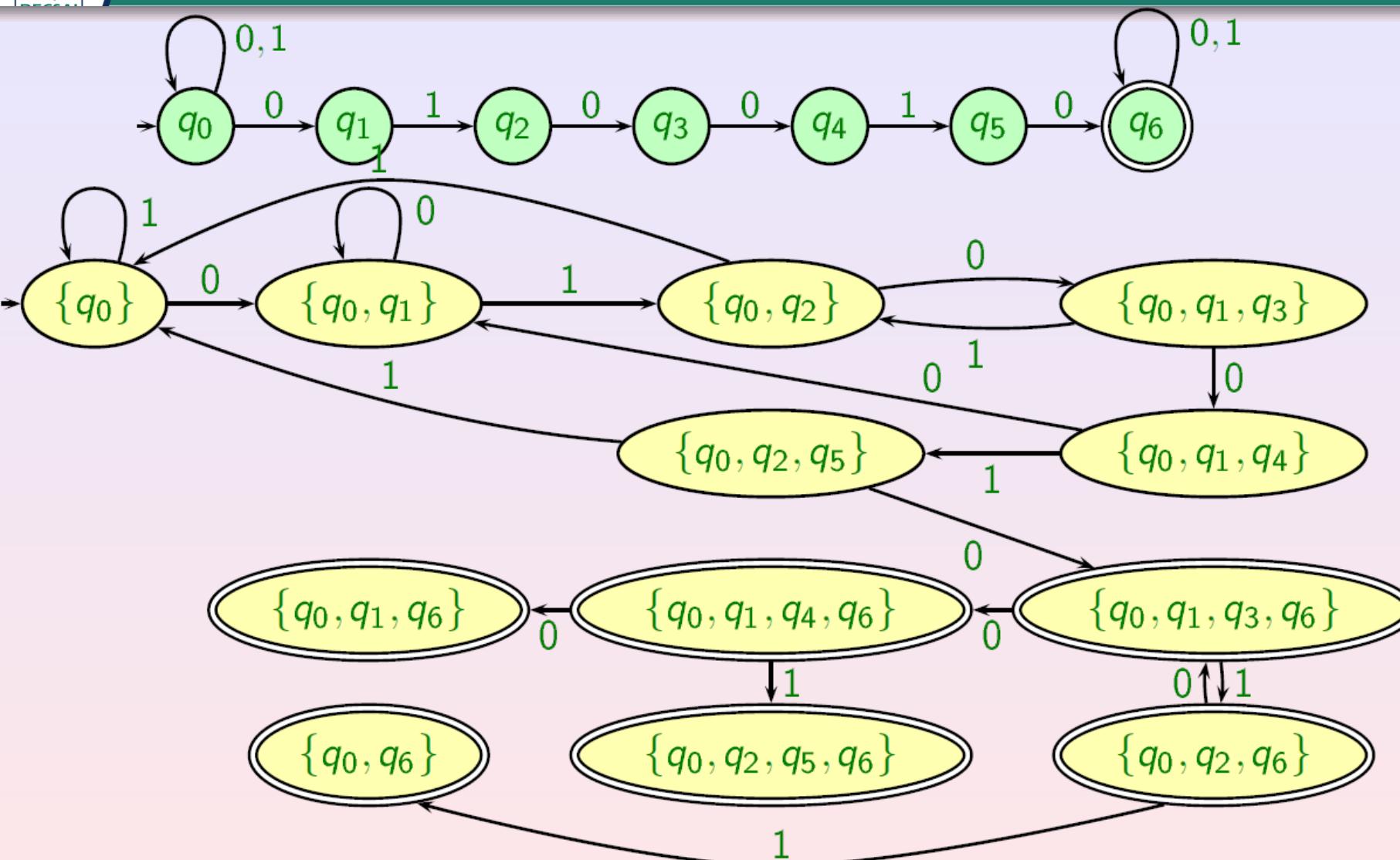


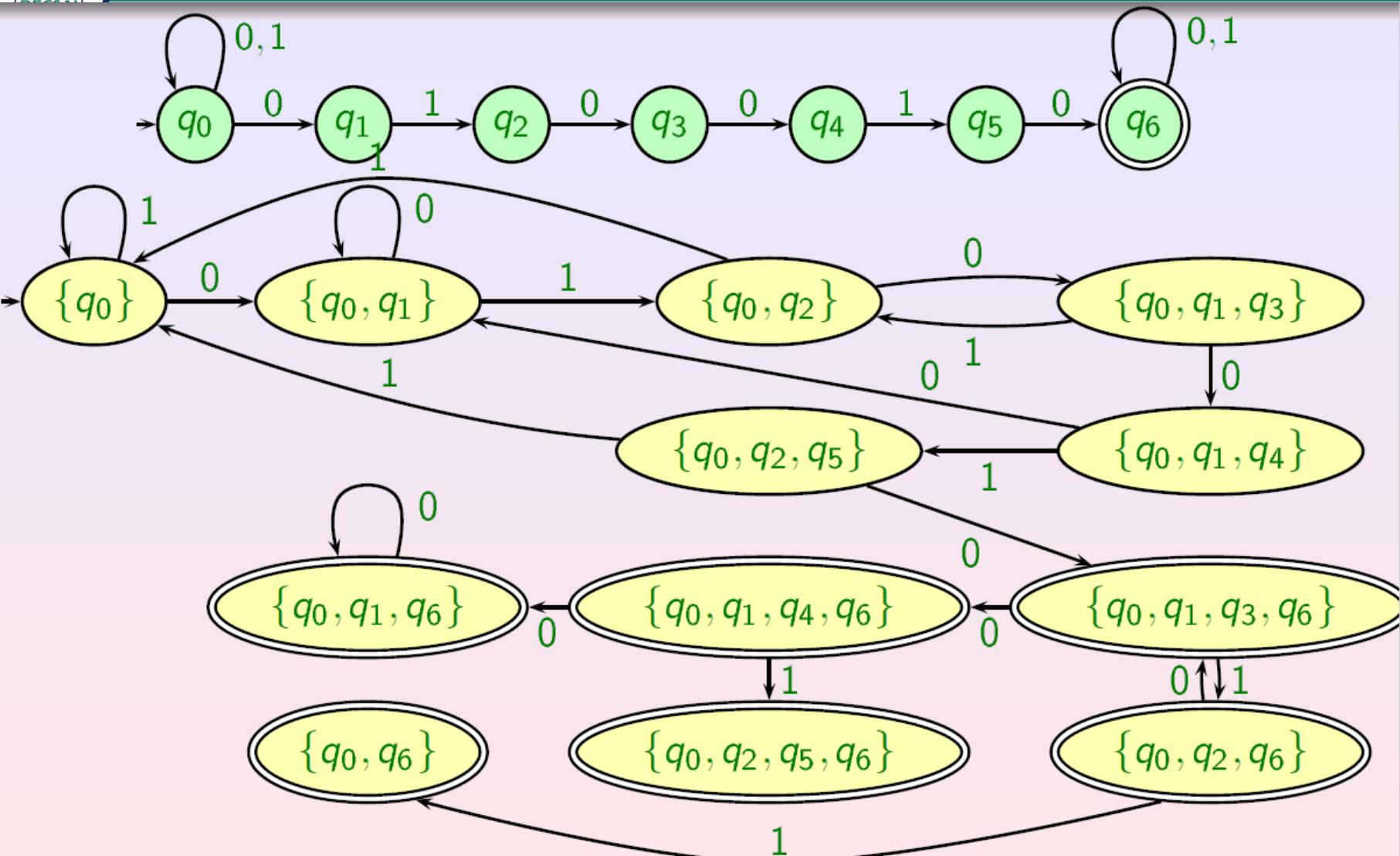


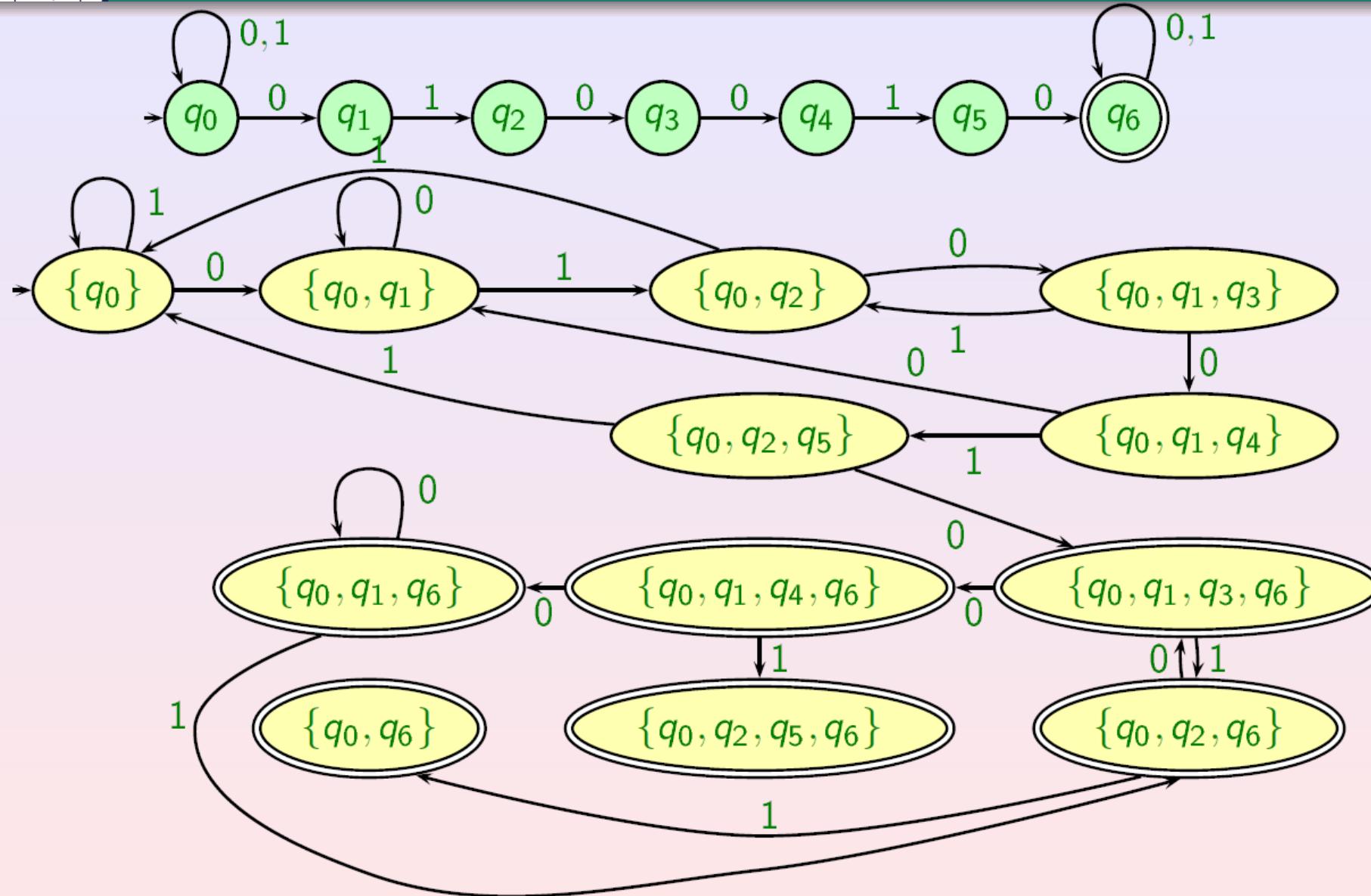


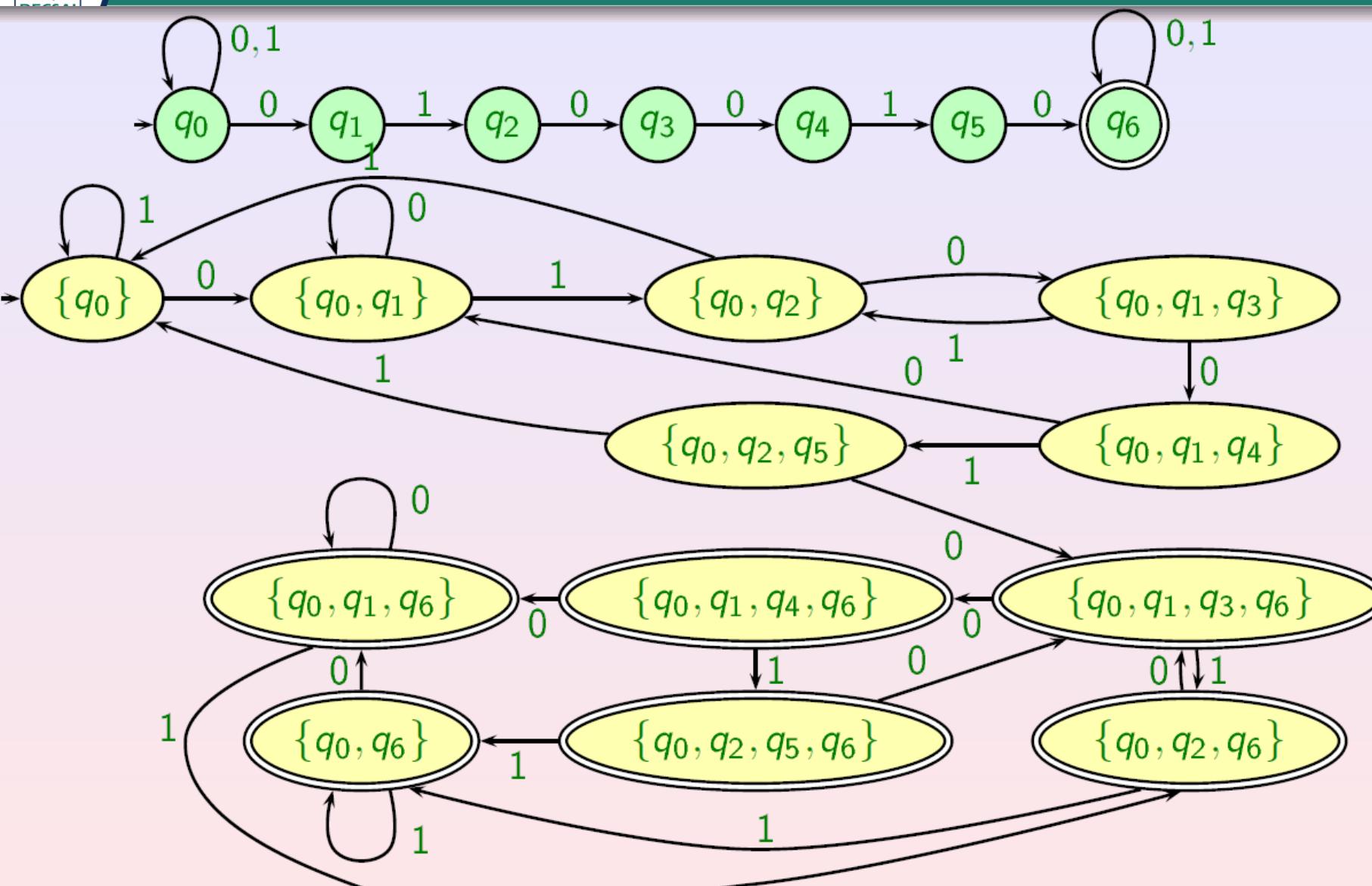














Autómatas finitos y expresiones regulares



1. Autómatas finitos deterministas
2. Autómatas finitos no deterministas
3. Autómatas finitos con transiciones nulas
4. Expresiones regulares
5. Gramáticas regulares
6. Máquinas de estados finitos

Definición: AFND con transiciones nulas (ε -AFND)

Un autómata finito no determinista con transiciones nulas (ε -AFND) es una quintupla $M=(Q,A, \delta, q_0, F)$ donde:

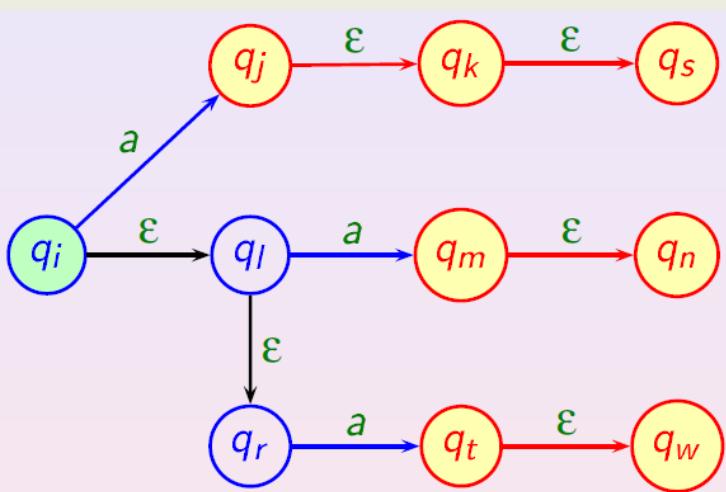
- Q es un conjunto finito llamado conjunto de estados.
- A es un alfabeto llamado alfabeto de entrada
- δ es una aplicación llamada función de transición

$$\delta: Q \times \{A \cup \{\varepsilon\}\} \rightarrow P(Q)$$

Donde $P(Q)$ representa al conjunto de las partes de Q .

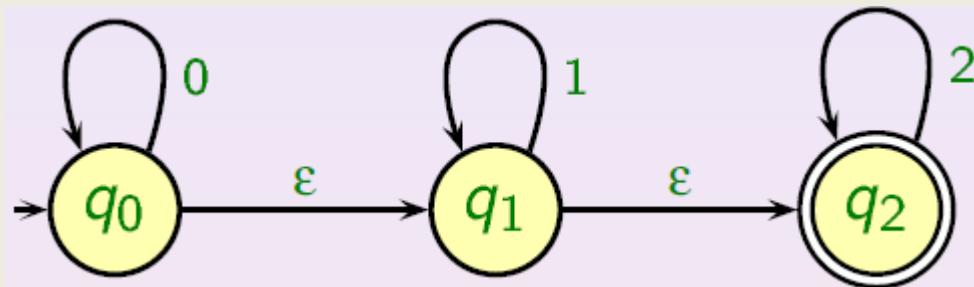
- q_0 es un elemento de Q , llamado estado inicial
- F es un subconjunto de Q , llamado conjunto de estados finales

Ejemplo:



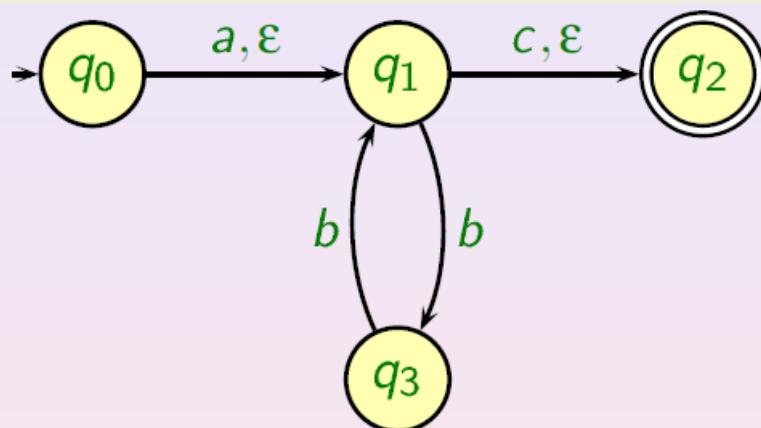
- Desde el estado q_i se puede llegar directamente a los estados $q_j, q_k, q_s, q_m, q_n, q_r, q_t, q_w$ tras leer una a .

Otro ejemplo:



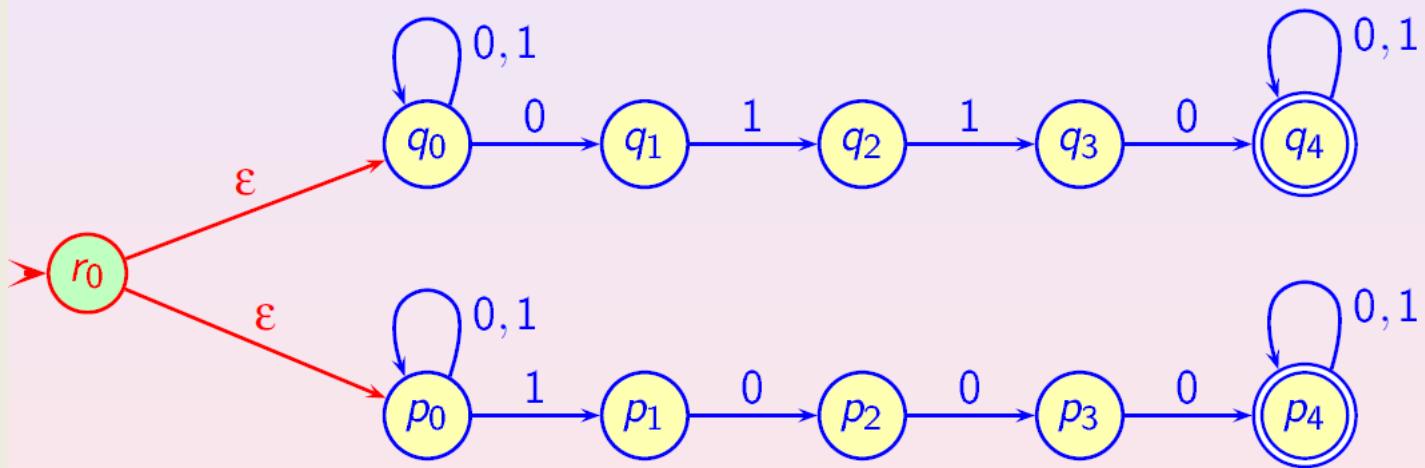
- El lenguaje aceptado es $L(M)=\{0^i1^j2^k, i,j,k \geq 0\}$

Otro ejemplo:



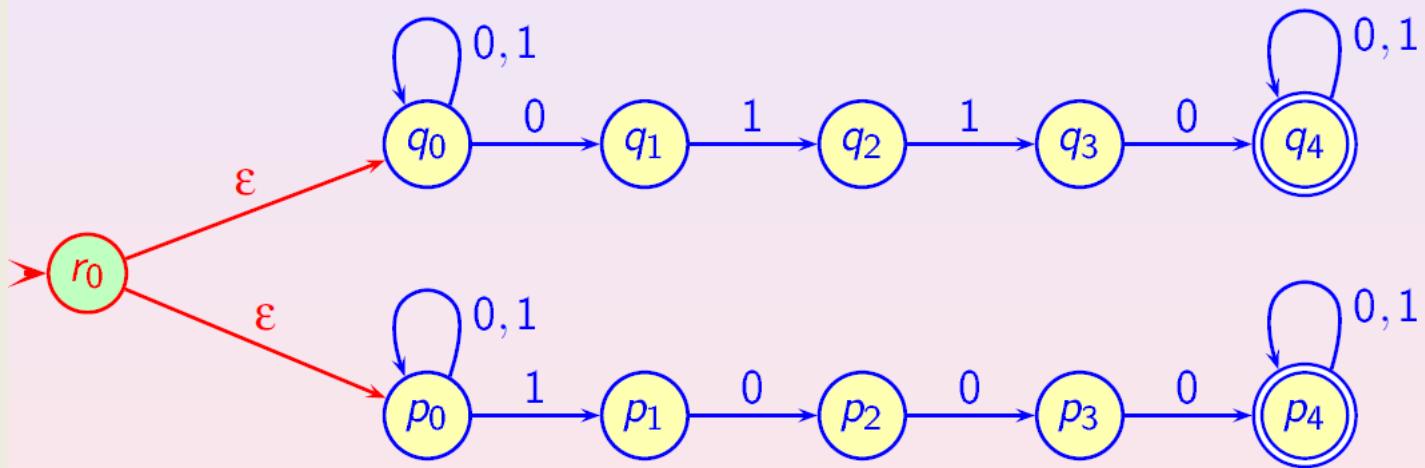
- El lenguaje aceptado es $L(M)=\{a^i b^{2j} c^k : i, k=0,1; j \geq 0\}$

Otro ejemplo:



- El lenguaje es: Secuencias de 0's y 1's que tienen como subcadena 0110 o 1000.

Otro ejemplo:



- El lenguaje es: Secuencias de 0's y 1's que tienen como subcadena 0110 o 1000.

Proceso de cálculo

- **Descripción Instantánea o Configuración:** Un elemento de $Q \times A^*$: (q, u) . Indica el estado actual en el que se encuentra el autómata, y la palabra que aún queda por procesar.
- **Configuración Inicial** para $u \in A^*$: (q_0, u)
- Relación o **paso de cálculo entre dos configuraciones**:
 $((q, u) \vdash (p, v)) \Leftrightarrow (u = av)^{\wedge} p \in \delta(q, a) \text{ ó } (u = v)^{\wedge} p \in \delta(q, \varepsilon)$

(desde el estado q se puede pasar al estado p con el símbolo a si, y solo si, hay una transición de q con el símbolo a que contenga a p)

De una configuración se puede pasar a varias configuraciones distintas en un mismo paso de cálculo en un automata finito no determinista con transiciones nulas.

De hecho, podemos hasta no pasar a ninguna configuración.

Lenguaje aceptado por un ε -AFND

- La relación de cálculo entre dos configuraciones (q,u) y (p,v) se denota como:

$$((q,u) \xrightarrow{*} (p,v))$$

Existe una relación de cálculo entre dos configuraciones si hay una secuencia de configuraciones C_1, C_2, \dots, C_n tal que $C_1 = (q,u)$ y $C_n = (p,v)$ y, además:

$$\forall i \leq n-1, C_i \vdash C_{i+1}$$

- Se define el **lenguaje L aceptado por un autómata M** como:
- $$L(M) = \{u \in A^* : \exists q \in F, (q_0, u) \xrightarrow{*} (q, \varepsilon)\}$$
- Las palabras de $L(M)$ se dice que son **aceptadas por el autómata**.

Equivalencia entre AFD y ϵ -AFND

- Todo AFD es también un ϵ -AFND: Él mismo, donde $\delta(q, \epsilon)=\emptyset$
- Por tanto, todo lenguaje aceptado por un AFD lo es también para algún ϵ -AFND.

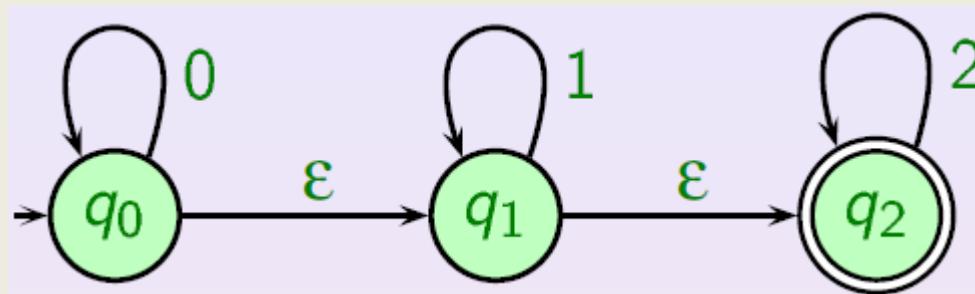
Al revés también tenemos equivalencia:

- Todo lenguaje aceptado por un ϵ -AFND lo deberá ser también para un AFD.

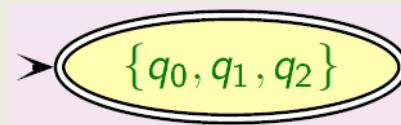
El paso de un ϵ -AFND a un AFD se realiza de forma equivalente a como se pasa de un AFND a AFD, considerando siempre el paso por las transiciones nulas.

Ejemplo:

- Pasar el siguiente ϵ -AFND a AFD:

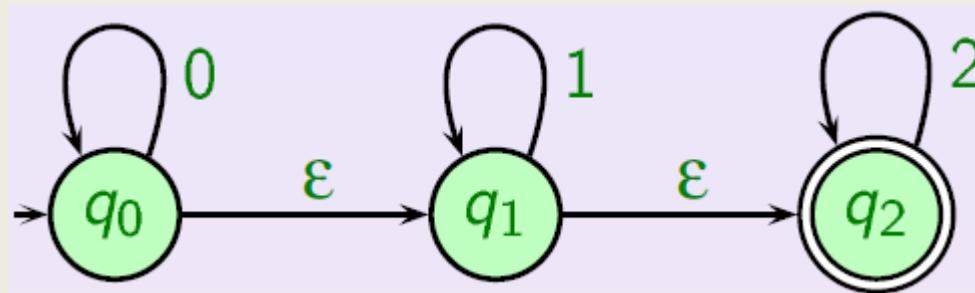


- Estado inicial:

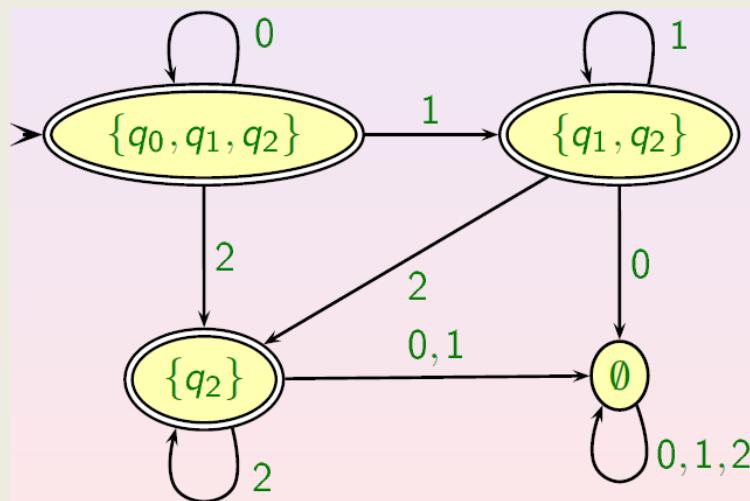


Ejemplo:

- Pasar el siguiente ϵ -AFND a AFD:



- Solución:



Definición: Clausura

Sea $M = (Q, A, \delta, q_0, F)$ un ε -AFND. Se define la **clausura de un estado q** , notado como $C/(q)$ a:

$$C/(q) = \{p : \exists p_1, \dots, p_n, p_1 = q, p_n = p, p_i \in \delta(p_{i-1}, \varepsilon), i = 2 \dots n\}$$

Por tanto, la **clausura de un conjunto de estados P** se define como:

$$C/(P) = \bigcup_{q \in P} C/(q)$$

(Nota: La clausura de un estado q es el conjunto de todos los estados a los que se puede llegar desde q con transiciones de la palabra vacía).

Definición: Función δ^*

Sea $B \subseteq Q, a \in A, u \in A^*, p \in Q$. Se define la función δ^* como:

$$\delta^*(B, a) = C / \left(\bigcup_{q \in B} \delta(q, a) \right)$$

$$\delta^*(B, \varepsilon) = C / (B)$$

$$\delta^*(B, au) = \delta^*(\delta^*(B, a), u)$$

$$\delta^*(p, u) = \delta^*(\{p\}, u)$$



Paso de ϵ -AFND a FND (construcción formal)

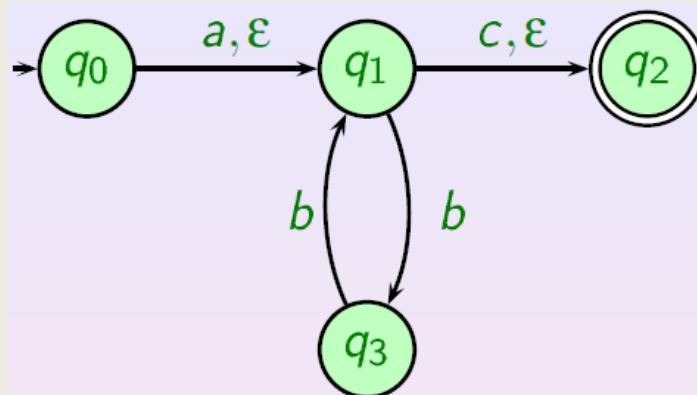
Sea $M = (Q, A, \delta, q_0, F)$ un ϵ -AFND, y $\bar{M} = (\bar{Q}, A, \bar{\delta}, \bar{q}_0, \bar{F})$ un AFD. Ambos autómatas aceptan el mismo lenguaje si:

$$\begin{aligned}\bar{Q} &= P(Q) \\ \bar{\delta}(B, a) &= \delta^*(B, a) = C / \left(\bigcup_{q \in B} \delta(q, a) \right) \\ \bar{q}_0 &= C / (q_0) \\ \bar{F} &= \{B \mid B \cap F \neq \emptyset\}\end{aligned}$$

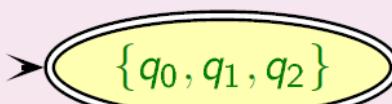
Donde $P(Q)$ denota a las partes de Q .

Ejemplo:

- Pasar el siguiente ϵ -AFND a AFD:

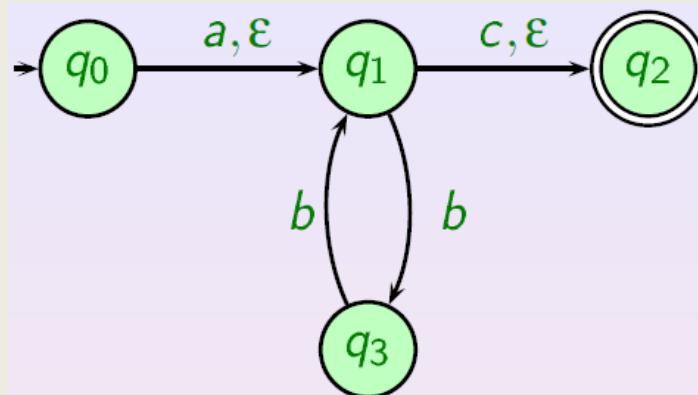


- Solución:

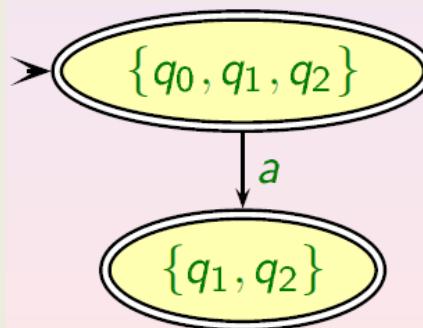


Ejemplo:

- Pasar el siguiente ϵ -AFND a AFD:

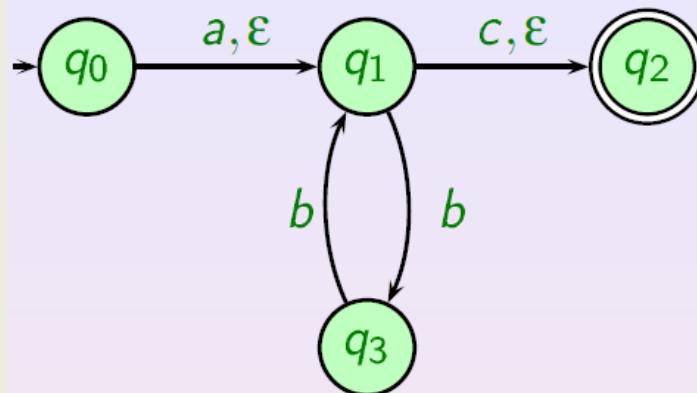


- Solución:

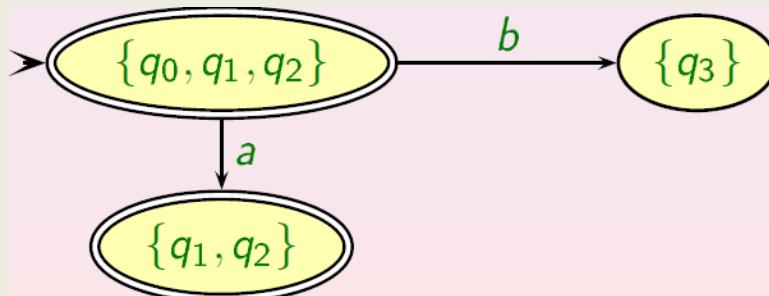


Ejemplo:

- Pasar el siguiente ϵ -AFND a AFD:

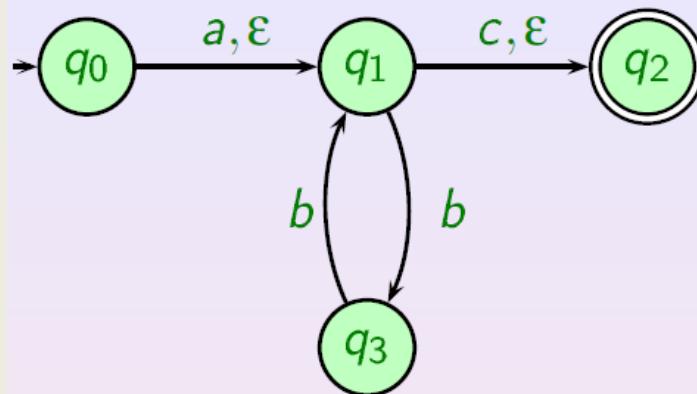


- Solución:

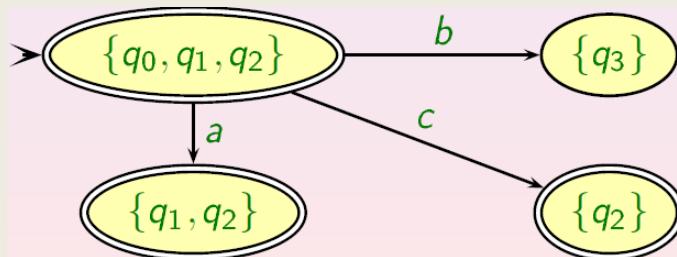


Ejemplo:

- Pasar el siguiente ϵ -AFND a AFD:

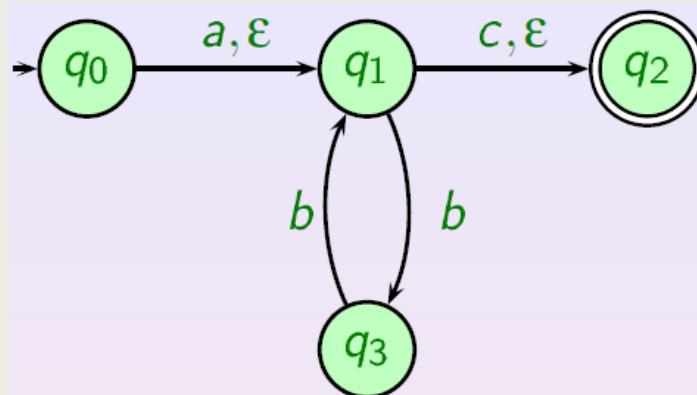


- Solución:

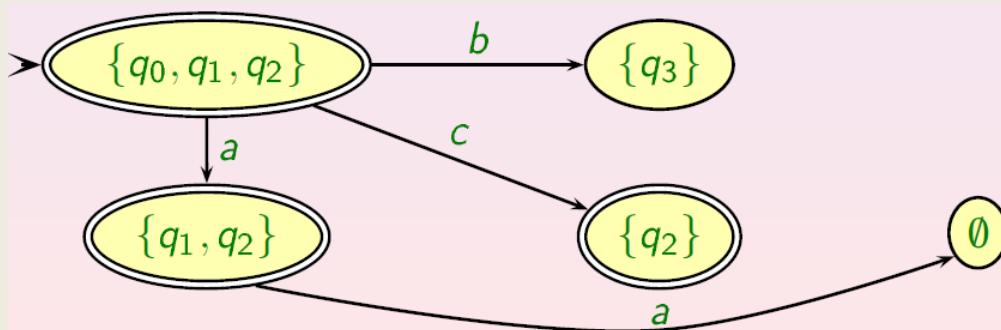


Ejemplo:

- Pasar el siguiente ϵ -AFND a AFD:

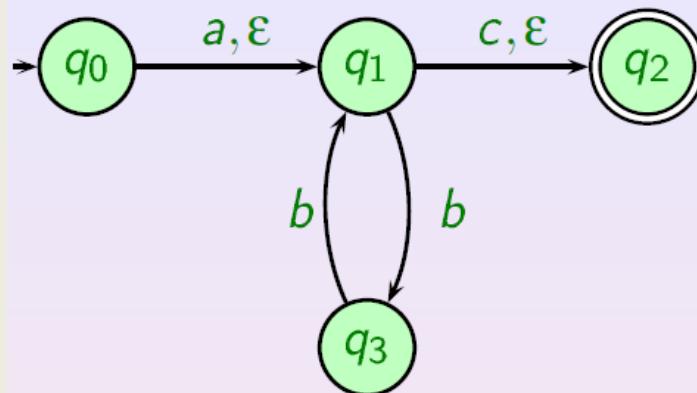


- Solución:

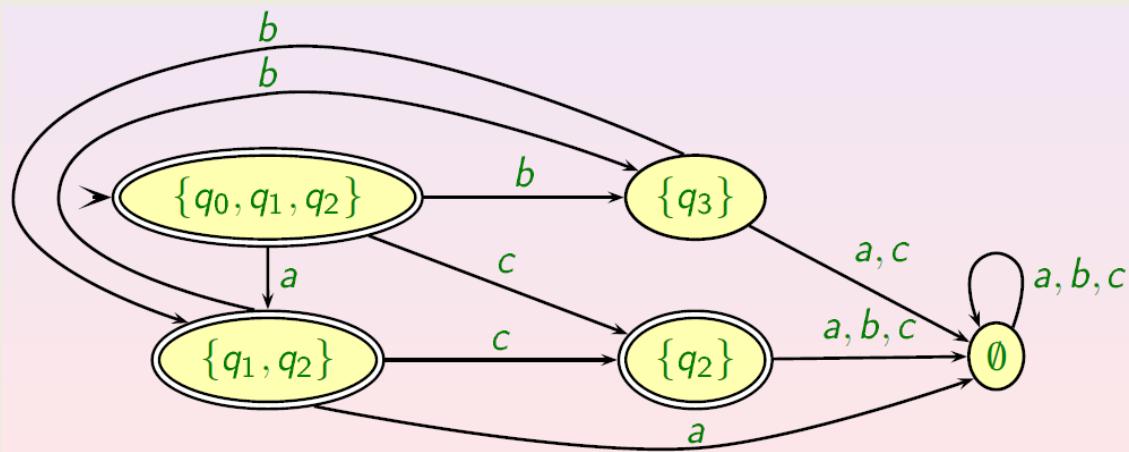


Ejemplo:

- Pasar el siguiente ϵ -AFND a AFD:



- Solución:





Autómatas finitos y expresiones regulares

- 1. Autómatas finitos deterministas
- 2. Autómatas finitos no deterministas
- 3. Autómatas finitos con transiciones nulas
- 4. Expresiones regulares
- 5. Gramáticas regulares
- 6. Máquinas de estados finitos



Definición: Expresión regular

Si A es un alfabeto, una expresión regular sobre este alfabeto se define de la siguiente forma:

- \emptyset es una expresión regular que denota el lenguaje vacío
- ϵ es una expresión regular que denota el lenguaje $\{\epsilon\}$
- Si $a \in A$, a es una expresión regular que denota el lenguaje $\{a\}$
- Si r y s son expresiones regulares denotando los lenguajes R y S entonces definimos las siguientes operaciones:
 - *Unión:* $(r+s)$ denota el lenguaje $R \cup S$.
 - *Concatenación:* (rs) denota el lenguaje RS .
 - *Clausura:* r^* denota el lenguaje R^*

Ejemplo:

Sea el alfabeto $A = \{0, 1\}$. Ejemplos de expresiones regulares:

- **00** El conjunto $\{00\}$
- **01* + 0** Conjunto de palabras que empiezan por 0 y después tienen una sucesión de unos.
- **(1+0)*** Conjunto de palabras en los que los ceros están precedidos siempre por unos.
- **(0+1)* 011** Conjunto de palabras que terminan en 011
- **0*1*** Conjunto de palabras formadas por una sucesión de ceros seguida de una sucesión de unos. Ambas sucesiones pueden ser vacías
- **00*11*** Conjunto de palabras formadas por una sucesión de ceros seguida de una sucesión de unos. Ninguna de las sucesiones pueden ser vacías. Es equivalente a 0^+1^+

A r^*r se le denota como r^+ . Por tanto, la última expresión quedaría como : **0⁺1⁺**.

Propiedades de las expresiones regulares

1. $r_1 + r_2 = r_2 + r_1$
2. $r_1 + (r_2 + r_3) = (r_1 + r_2) + r_3$
3. $r_1(r_2r_3) = (r_1r_2)r_3$
4. $r\varepsilon = r$
5. $r\emptyset = \emptyset$
6. $r + \emptyset = r$
7. $\varepsilon^* = \varepsilon$
8. $r_1(r_2 + r_3) = r_1r_2 + r_1r_3$
9. $(r_1 + r_2)r_3 = r_1r_3 + r_2r_3$
10. $r^* + \varepsilon = r^*$
11. $r^* + \varepsilon = r^*$
12. $(r + \varepsilon)^* = r^*$
13. $(r + \varepsilon)^+ = r^*$
14. $(r_1^* + r_2^*)^* = (r_1 + r_2)^*$
15. $(r_1^* r_2^*)^* = (r_1 + r_2)^*$

Ejercicios:

Suponiendo el alfabeto $A = \{0,1\}^*$...

1. Construir una expresión regular para las palabras con un número par de ceros.
2. Construir una expresión regular para las palabras que contengan a 011 como subcadena.
3. Construir una expresión regular para el conjunto de palabras que empiezan por 000 y después no aparece nunca más esta cadena.
4. Construir una expresión regular para el conjunto de palabras que tienen a 000 o a 101 como subcadena

Soluciones:

1. Construir una expresión regular para las palabras con un número par de ceros: $1^*(01^*01^*)^*$
2. Construir una expresión regular para las palabras que contengan a 011 como subcadena: $(0+1)^*0110(0+1)^*$
3. Construir una expresión regular para el conjunto de palabras que empiezan por 000 y después no aparece nunca más esta cadena: $(000)(1+10+100)^*$
4. Construir una expresión regular para el conjunto de palabras que tienen a 000 o a 101 como subcadena:
 $(0+1)^*(000+101)(0+1)^*$

Ejercicios: Verdadero/Falso

Sean r , s y t expresiones regulares. Entonces...

- ¿ $r^* = (rs)^*$?
- ¿ $(r+s)^* = r^* + s^*$?
- Si el lenguaje asociado a tanto r como s tiene la palabra vacía, entonces ¿ $(rs)^* = (sr)^*$?
- ¿ $(r+\varepsilon)^* = r^*$?
- ¿ $r(r+s)^* = (r+s)^*r$?
- ¿ El lenguaje de r^*s^* está incluido en el lenguaje de $(rs)^*$?
- ¿ $(r+s)^*t = r^*t + s^*t$?
- ¿ $(r^*s^*)^* = (r+s)^*$?



Soluciones: Verdadero/Falso

Sean r , s y t expresiones regulares. Entonces...

- ¿ $r^* = (rs)^*$? **FALSO**
- ¿ $(r+s)^* = r^* + s^*$? **FALSO**
- Si el lenguaje asociado a tanto r como s tiene la palabra vacía, entonces ¿ $(rs)^* = (sr)^*$? **VERDADERO**
- ¿ $(r+\epsilon)^* = r^*$? **VERDADERO**
- ¿ $r(r+s)^* = (r+s)^*r$? **FALSO**
- ¿ El lenguaje de r^*s^* está incluido en el lenguaje de $(rs)^*$? **FALSO**
- ¿ $(r+s)^*t = r^*t + s^*t$? **FALSO**
- ¿ $(r^*s^*)^* = (r+s)^*$? **VERDADERO**

Ejercicios: Verdadero/Falso

Sean r , s y t expresiones regulares. Entonces...

- ¿ $(rs)^* = (r+s)^*$?
- ¿ $r^*r^* = r^*$?
- ¿ $(r\emptyset)^* = r+\emptyset$?
- ¿ $r^* \varepsilon = r+ \varepsilon$?
- ¿ $r(s+r)^* = (rs)^*r$?
- ¿ El lenguaje generado por $(rr)^*$ está incluido en r^* ?
- ¿ $t(r+s)^* = tr^* + ts^*$?



Soluciones: Verdadero/Falso

Sean r , s y t expresiones regulares. Entonces...

- ¿ $(rs)^* = (r+s)^*$? **FALSO**
- ¿ $r^* r^* = r^*$? **VERDADERO**
- ¿ $(r\emptyset)^* = r+\emptyset$? **FALSO**
- ¿ $r^* \epsilon = r+ \epsilon$? **FALSO**
- ¿ $r(s+r)^* = (rs)^* r$? **FALSO**
- ¿ El lenguaje generado por $(rr)^*$ está incluido en r^* ?
VERDADERO
- ¿ $t(r+s)^* = tr^* + ts^*$? **FALSO**



Propiedad: Equivalencia entre expresiones regulares y AFD

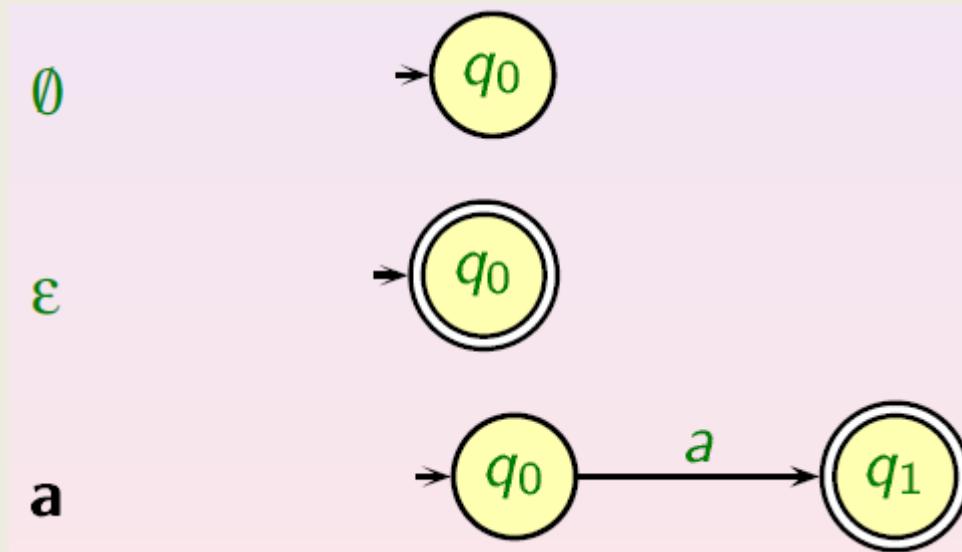
Un lenguaje es aceptado por un AFD si, y solo si, este lenguaje puede representarse mediante una expression regular.

La demostración de esta propiedad se debe realizar en dos pasos:

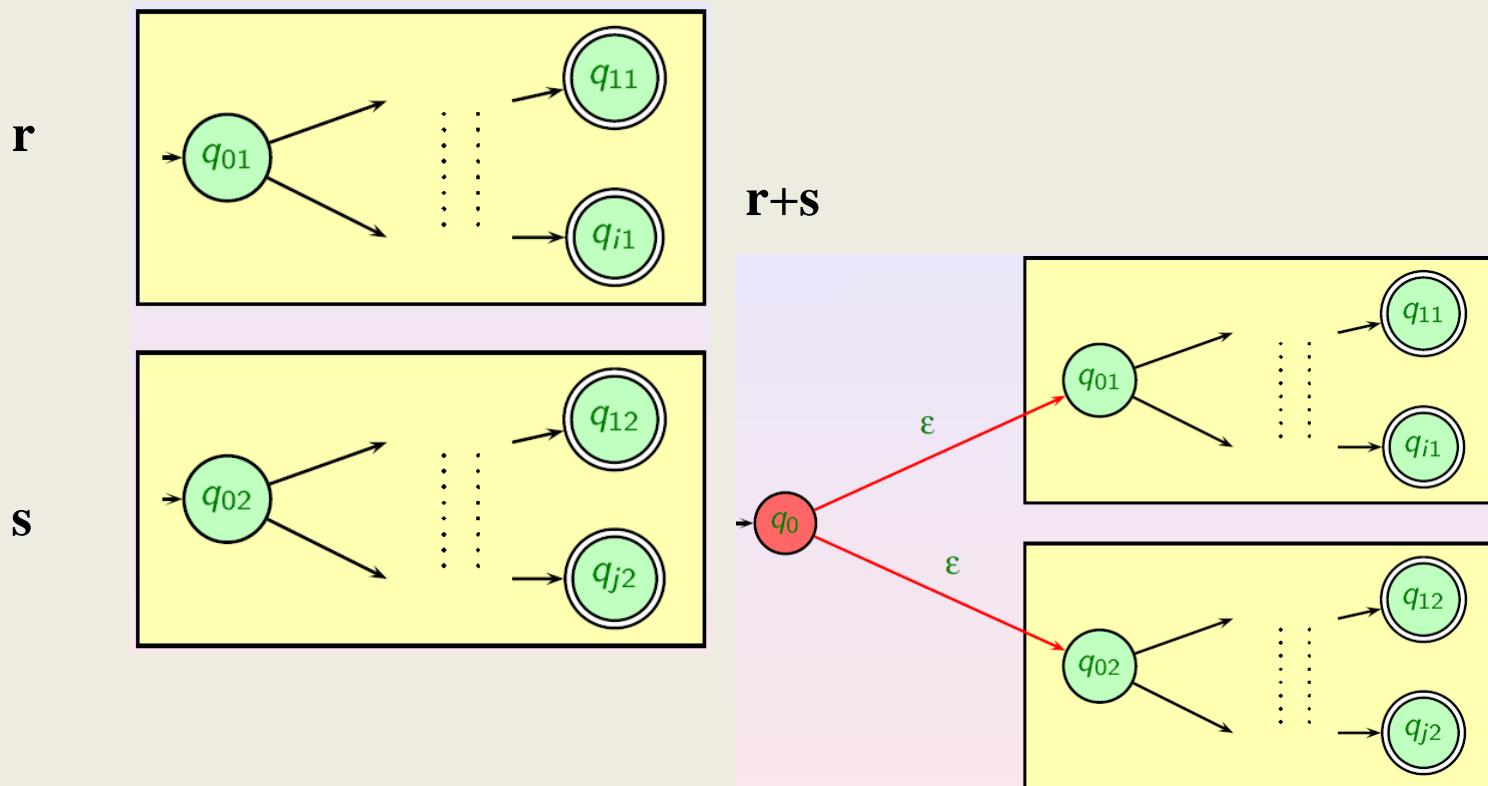
- Dada una expression regular, comprobar que existe un automata que genera el mismo lenguaje.
- Dado un automata AFD, comprobar que siempre existe una expresión regular asociada que genera el mismo lenguaje.

Paso de expresiones regulares a AFD (I)

Comencemos por los lenguajes más básicos:

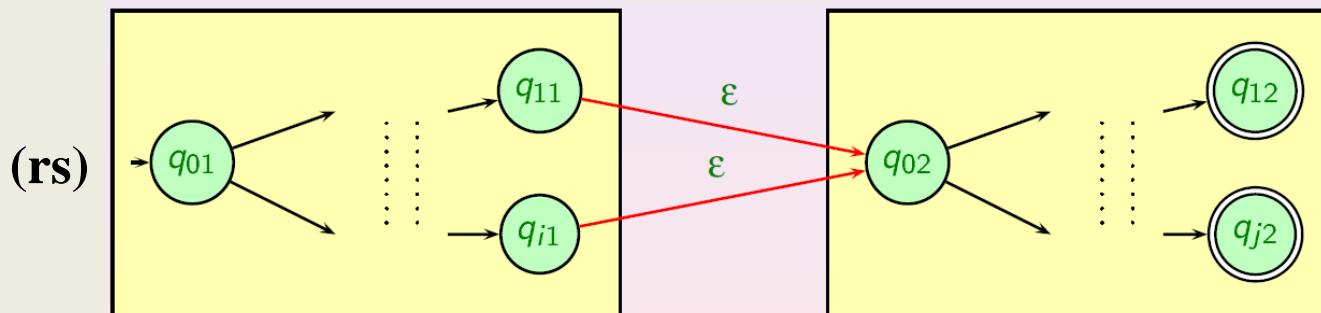
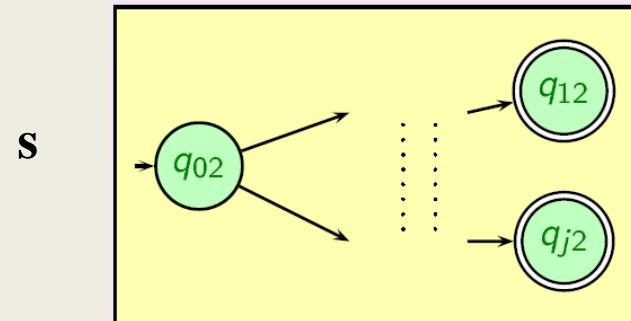
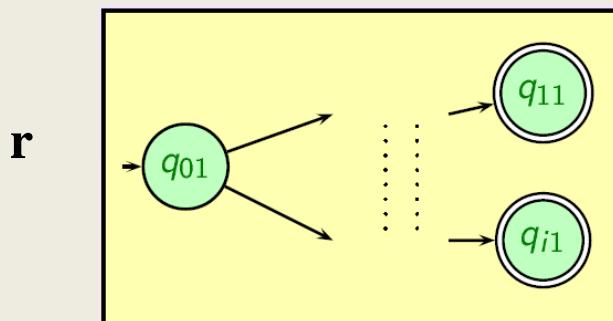


Paso de expresiones regulares a AFD (II)

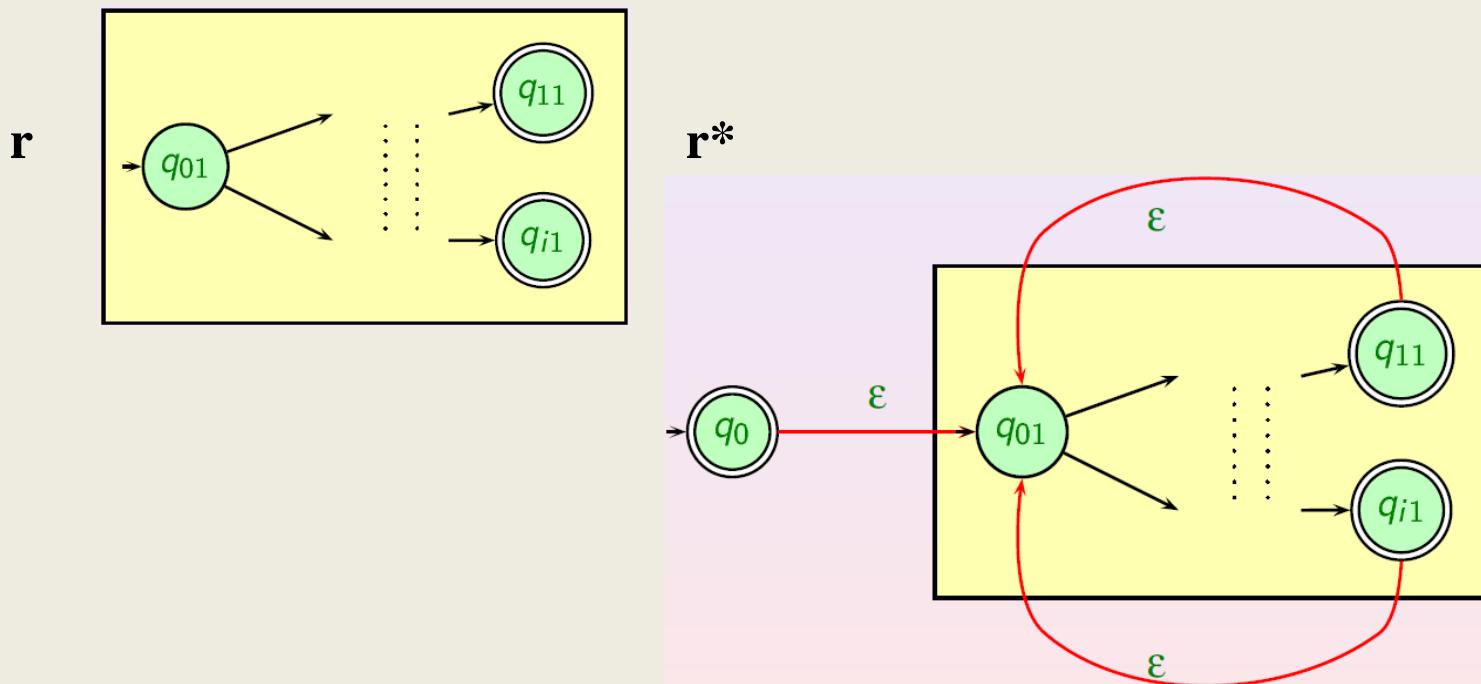
Paso de la unión ($r+s$):

Paso de expresiones regulares a AFD (III)

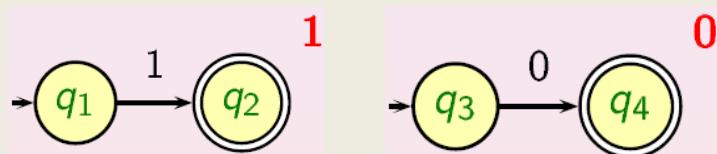
Paso de la concatemación (rs):



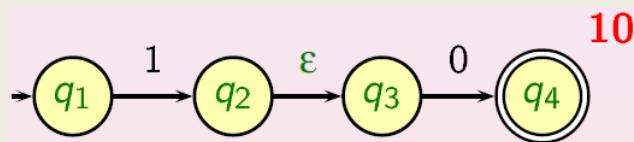
Paso de expresiones regulares a AFD (IV)

Paso de la clausura (r^*):

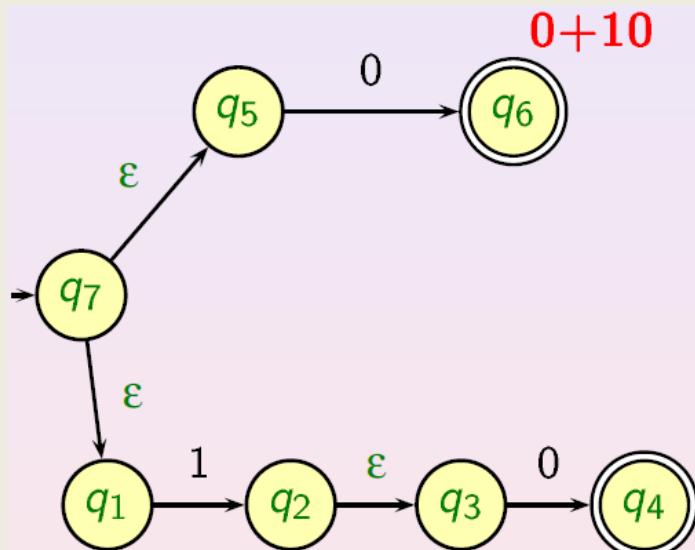
Ejemplo: Encontrar un autómata para $r=(0+10)^*011$



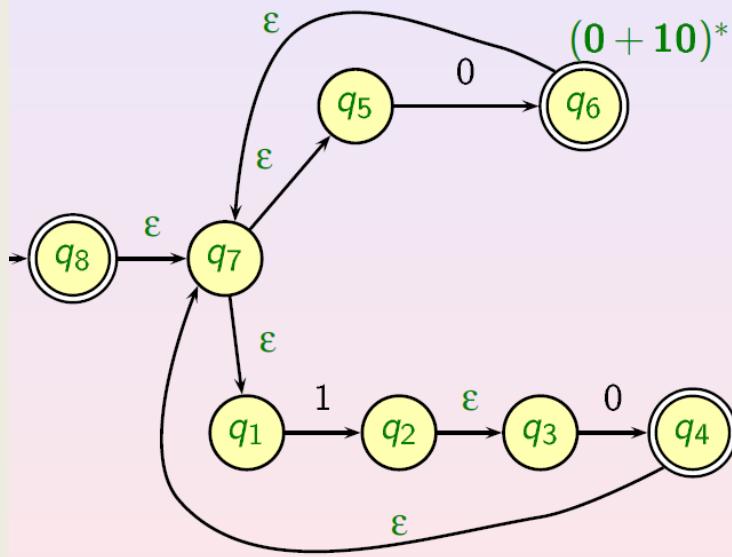
Ejemplo: Encontrar un autómata para $r=(0+10)^*011$



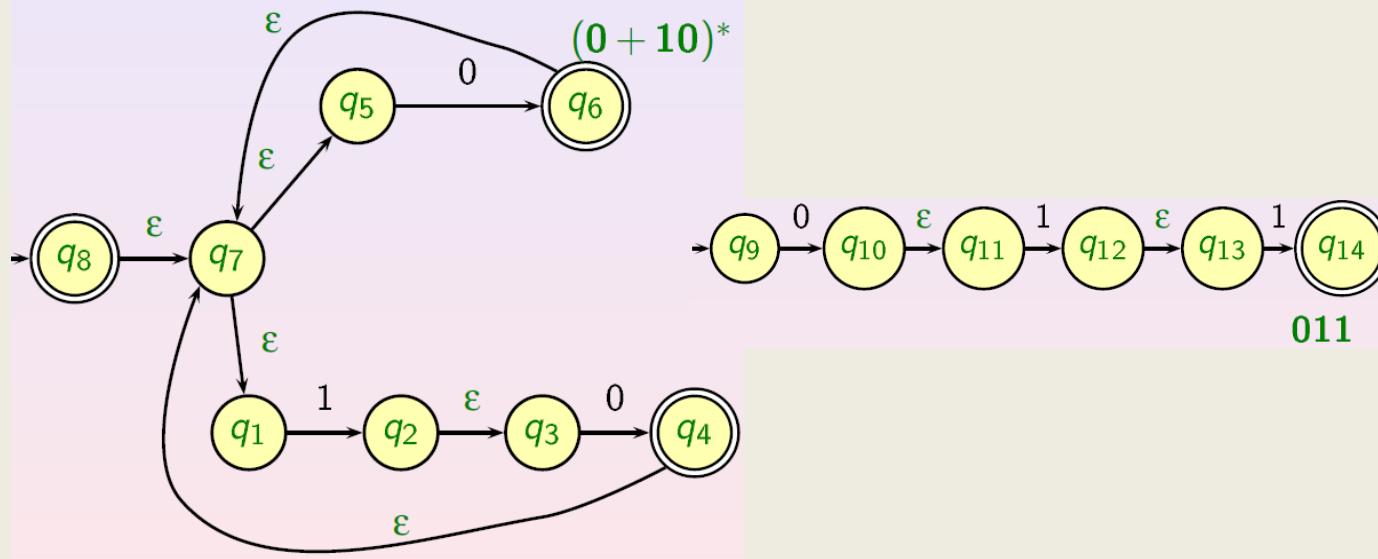
Ejemplo: Encontrar un autómata para $r=(0+10)^*011$



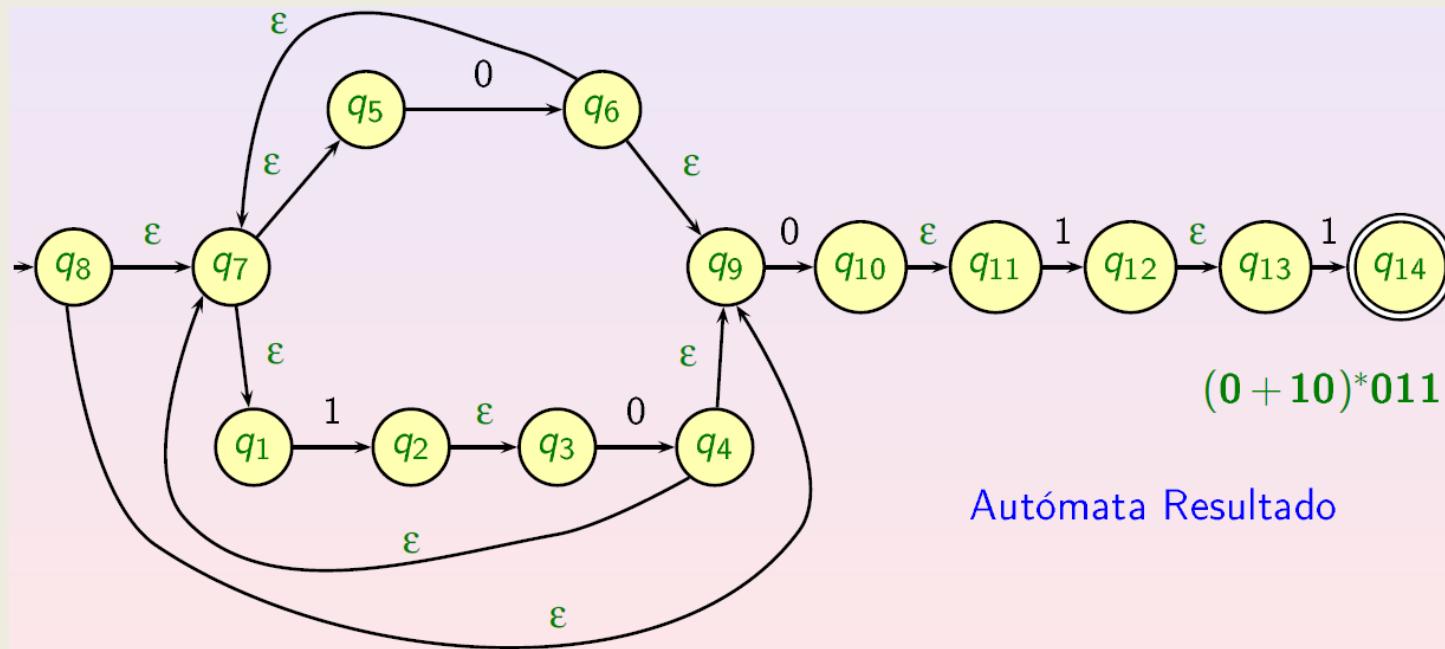
Ejemplo: Encontrar un autómata para $r=(0+10)^*011$



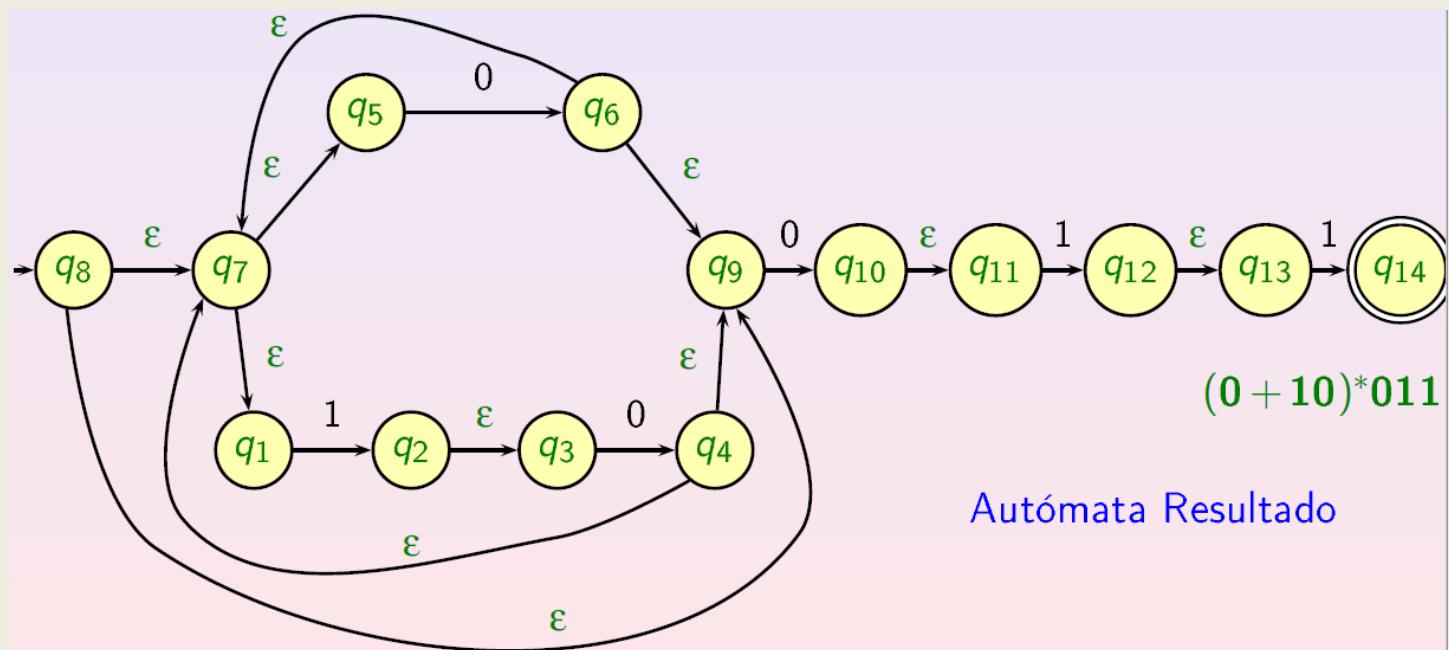
Ejemplo: Encontrar un autómata para $r=(0+10)^*011$



Ejemplo: Encontrar un autómata para $r=(0+10)^*011$



Ejemplo: Encontrar un autómata para $r=(0+10)^*011$



Paso de AFD a expresiones regulares (I)

Ecuación Característica o Fundamental:

1. x_i : Conjunto de palabras que permiten pasar desde el estado q_i a un estado final.
2. Si $q_i \in F$, entonces $\epsilon \in x_i$, ya que para todo $q \in Q$, $q \in \delta(q, \epsilon)$. Si $q_i \notin F$, entonces ϵ no tiene por qué estar en x_i .
3. Si $\delta(q_i, a) = q_j \in F$, entonces el símbolo “a” debe pertenecer a x_i , según la definición 1.
4. Si $\delta(q_i, a) = q_j$, entonces la concatenación de la entrada del símbolo a y x_j (conjunto de cadenas que permiten pasar de q_j a un estado final) debe estar en x_i (conjunto de cadenas que permiten pasar de q_i a un estado final). Así, $ax_j \subseteq x_i$.

Paso de AFD a expresiones regulares (II)

Ecuación Característica o Fundamental:

5. Para cada estado q_i se puede definir el denominado sistema de ecuaciones de conjunto lineales por la derecha, que se calcula como:

$$x_i = C_i \bigcup_{j=1}^{|Q|} D_{ij} x_j$$

Donde:

$$C_i = \begin{cases} \emptyset & \text{si } q_i \notin F \\ \varepsilon & \text{si } q_i \in F \end{cases}$$

$$D_{ij} = \{a \in A : \delta(q_i, a) = q_j\}$$

Paso de AFD a expresiones regulares (III)

Ecuación Característica o Fundamental:

- Tras definir las ecuaciones, nos quedan de la forma:

$$x_i = Ax_i + B$$

Donde $\varepsilon \notin A$ y $x_i \notin B$. Por tanto, la solución es:

$$x_i = A^*. B$$

Al efecto de que se cumpla que $\varepsilon \notin A$, lo más sencillo es eliminar las ε -transiciones del autómata cuando sea no determinista.

Paso de AFD a expresiones regulares (III)

Ecuación Característica o Fundamental:

- Tras definir las ecuaciones, nos quedan de la forma:

$$x_i = Ax_i + B$$

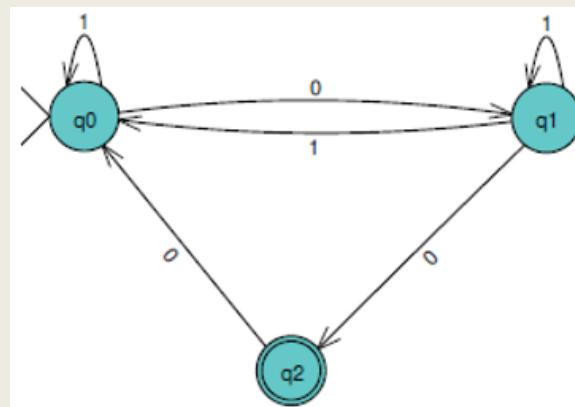
Donde $\varepsilon \notin A$ y $x_i \notin B$. Por tanto, la solución es:

$$x_i = A^*. B$$

Al efecto de que se cumpla que $\varepsilon \notin A$, lo más sencillo es eliminar las ε -transiciones del autómata cuando sea no determinista.

- Cuando se haya calculado la ecuación para x_0 (la asociada al estado inicial), esta será la expresión regular resultante que describe el lenguaje asociado.

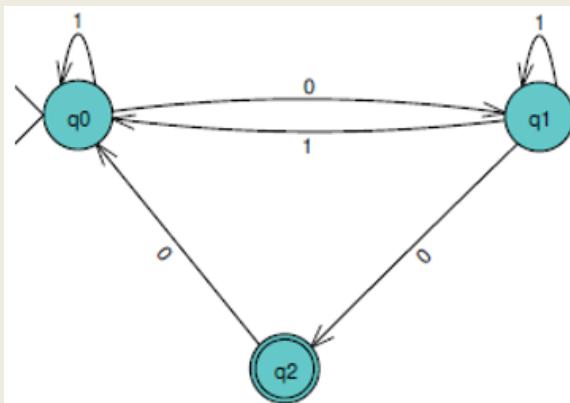
Ejemplo: Encontrar una expresión regular para el AFD



Las ecuaciones que salen son:

$$\begin{aligned}x_0 &= 1x_0 + 0x_1 \\x_1 &= 1x_0 + 1x_1 + 0x_2 \\x_2 &= 0x_0 + \epsilon\end{aligned}$$

Ejemplo: Encontrar una expresión regular para el AFD



$$x_0 = 1x_0 + 0x_1$$

$$x_1 = 1x_0 + 1x_1 + 0x_2$$

$$x_2 = 0x_0 + \epsilon$$

Simplificamos x_0 : $x_0 = 1x_0 + 0x_1 = 1^*0x_1$

Sustituimos en x_2 : $x_2 = 0x_0 + \epsilon = 01^*0x_1 + \epsilon$

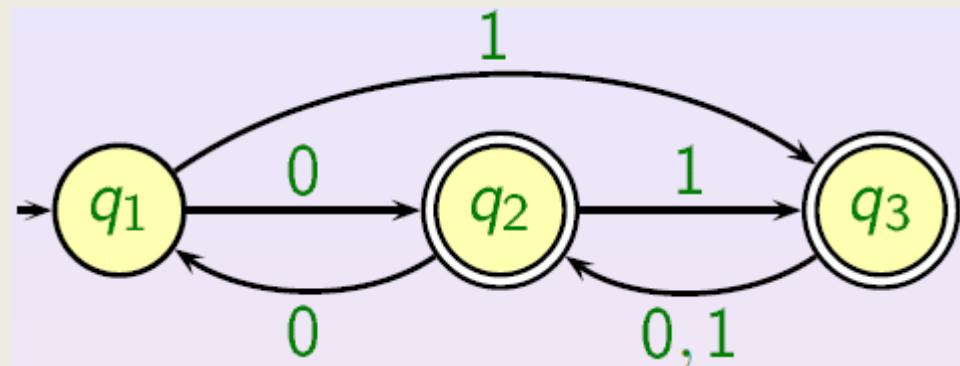
Sustituimos en x_1 :

$$\begin{aligned} x_1 &= 1x_0 + 1x_1 + 0x_2 = 1x_1 + 11^*0x_1 + 001^*0x_1 + 0 = \\ &(1 + 11^*0 + 001^*0)x_1 + 0 = (1 + 11^*0 + 001^*0)^*0 \end{aligned}$$

Sustituimos en x_0 : $x_0 = 1^*0x_1 = 1^*0(1 + 11^*0 + 001^*0)^*0$

Solución: $1^*0(1 + 11^*0 + 001^*0)^*0$

Ejercicio: Pasar el siguiente AFD a expresión regular



Solución: $(0+1(0+1))(1(0+1))^*0) * (0+1(0+1))(1(0+1)) * +$
 $(00+(1+01)((0+1)1)^*(0+1)0) * (1+01)((0+1)1) *$



Autómatas finitos y expresiones regulares

1. Autómatas finitos deterministas
2. Autómatas finitos no deterministas
3. Autómatas finitos con transiciones nulas
4. Expresiones regulares
5. Gramáticas regulares
6. Máquinas de estados finitos



Gramáticas regulares (o de Tipo 3)

- **Lineales por la derecha.** Cuando todas las producciones tienen la forma:

$$\begin{aligned} A &\rightarrow uB \\ A &\rightarrow u \end{aligned}$$

- **Lineales por la izquierda.** Cuando todas las producciones tienen la forma

$$\begin{aligned} A &\rightarrow Bu \\ A &\rightarrow u \end{aligned}$$

Ejemplos:

Gramática Lineal por la Derecha:

$$S \rightarrow 0A, \quad A \rightarrow 10A, \quad A \rightarrow \epsilon$$

(La misma) Gramática Lineal por la Izquierda

$$S \rightarrow S10, \quad S \rightarrow 0$$

La expresión regular asociada

$$0(01)^*$$



Paso de gramáticas regulares a autómatas

Si L es un lenguaje generado por una gramática regular, entonces existe un autómata finito determinista que lo reconoce.

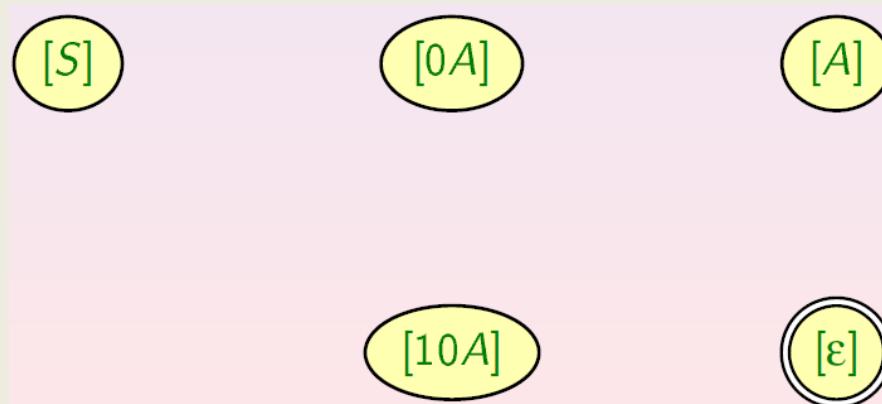
Dada la **gramática G lineal por la derecha** que genera el lenguaje L , entonces el Σ -AFND que acepta L es $M=(Q,A,\delta,q_0, F)$ donde:

- $Q = \{[\alpha] : (\alpha = S) \vee (\exists A \in V, u \in T, \text{ tales que } A \rightarrow u\alpha \in P)\}$
- $q_0 = [S]$
- $F = \{[\varepsilon]\}$
- δ viene definida por
 - Si A es una variable: $\delta([A], \varepsilon) = \{[\alpha] : (A \rightarrow \alpha) \in P\}$
 - Si $a \in T$ y $\alpha \in (T^*V \cup T^*)$, entonces
$$\delta([\alpha a], a) = [\alpha]$$

Ejemplo:

Sea la Gramática Lineal por la Derecha: $S \rightarrow 0A, A \rightarrow 10A, A \rightarrow \epsilon$

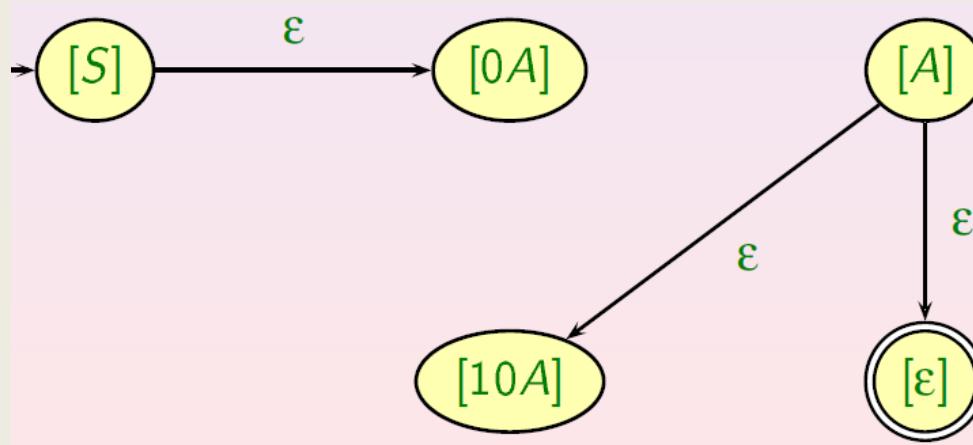
El automata asociado se construye:



Ejemplo:

Sea la Gramática Lineal por la Derecha: $S \rightarrow 0A$, $A \rightarrow 10A$, $A \rightarrow \epsilon$

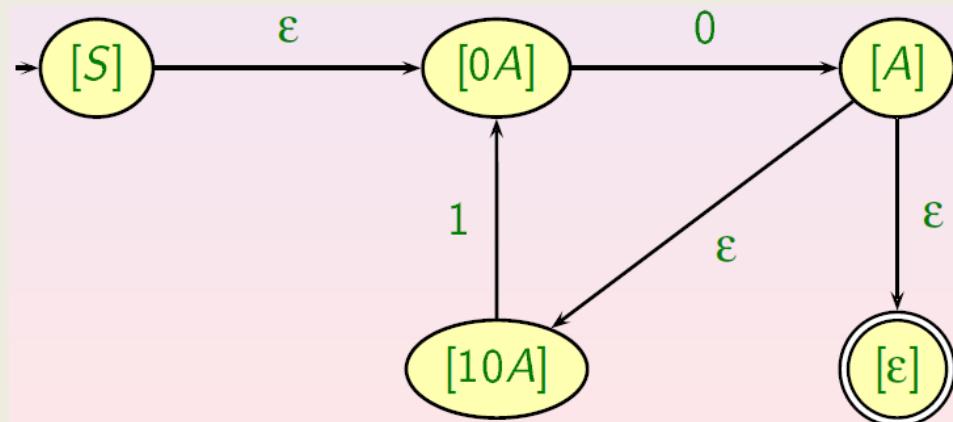
El automata asociado se construye:



Ejemplo:

Sea la Gramática Lineal por la Derecha: $S \rightarrow 0A, A \rightarrow 10A, A \rightarrow \epsilon$

El automata asociado se construye:



Paso de gramáticas regulares a autómatas

Si L es un lenguaje generado por una gramática regular, entonces existe un autómata finito determinista que lo reconoce.

Dada la **gramática G lineal por la izquierda**, $G=(V, T, P, S)$

1. Consideramos la gramática $G'=(V, T, P', S)$ donde

$$P' = \{A \rightarrow \alpha : A \rightarrow \alpha^{-1} \in P\}$$

Es inmediato que $L(G')=L(G)^{-1}$

2. Sea M' el AFND que acepta el lenguaje $L(G')$.
3. Calcular M a partir de M' invirtiendo el autómata:

Dejar sólo un estado final

Invertir las transiciones

Intercambiar el estado inicial y el final

El lenguaje aceptado por M es:

$$L(M')^{-1}=L(G')^{-1}=(L(G)^{-1})^{-1}=L(G)$$

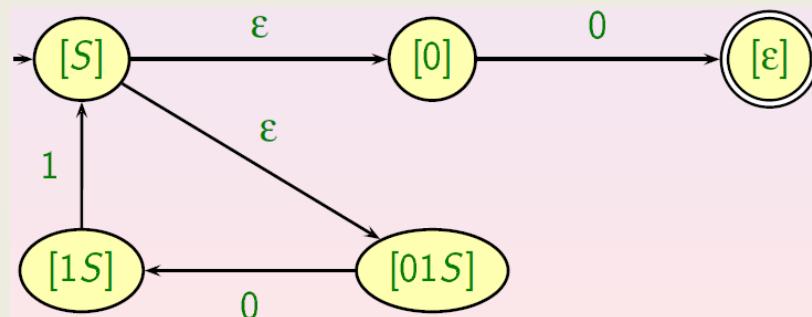
Ejemplo:

Sea la Gramática Lineal por la Izquierda: $S \rightarrow S10, S \rightarrow 0$

El automata asociado se construye primero invirtiendo la parte derecha de las producciones (y así tenemos una gramática lineal por la derecha):

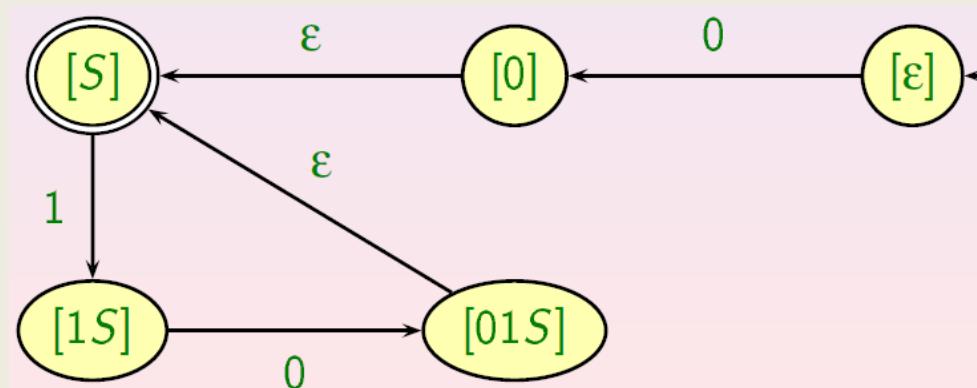
$$\begin{aligned}S &\rightarrow 01S \\S &\rightarrow 0\end{aligned}$$

Seguidamente, se construye su automata:



Ejemplo:

Por último, se invierte el autómata:



Paso de autómatas a gramáticas regulares

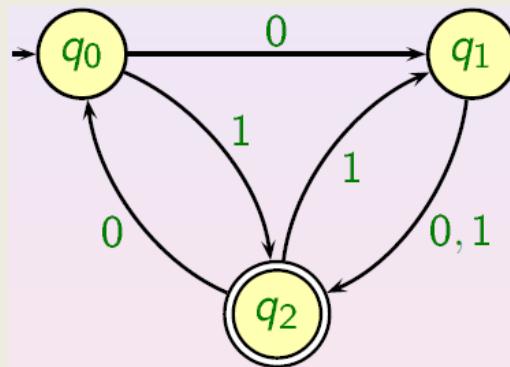
Si L es aceptado por un AFD entonces L puede generarse por una gramática lineal derecha/izquierda

- **Gramática Lineal por la derecha**, la variable inicial es q_0 y P contiene las producciones:

$$\begin{aligned} p \rightarrow aq, & \text{ si } \delta(p,a)=q \\ p \rightarrow \epsilon, & \text{ si } p \in F \end{aligned}$$

- **Gramática Lineal por la izquierda**, invertimos el autómata, construimos la gramática lineal por la derecha asociada e invertimos la parte derecha de las producciones

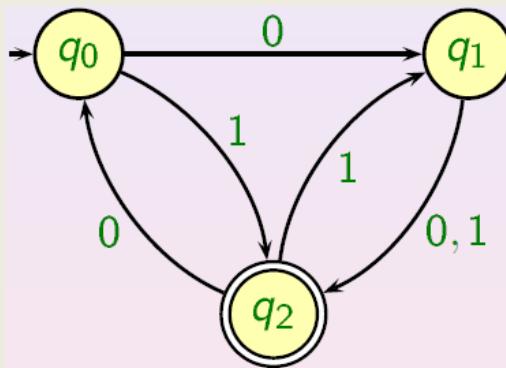
Ejemplo: Pasar autómata a gramática lineal por la derecha



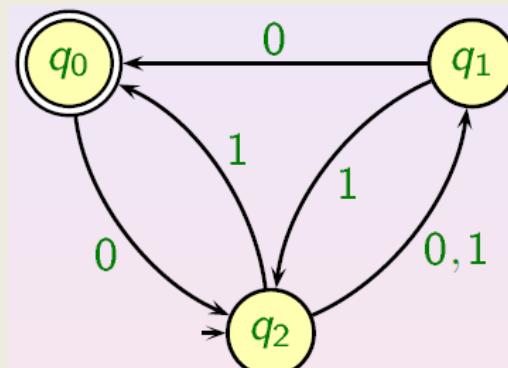
Gramática resultante (símbolo inicial q_0):

$$\begin{aligned} q_0 &\rightarrow 0q_1, q_0 \rightarrow 1q_2 \\ q_1 &\rightarrow 0q_2, q_1 \rightarrow 1q_2 \\ q_2 &\rightarrow 0q_0, q_2 \rightarrow 1q_1, q_2 \rightarrow \epsilon \end{aligned}$$

Ejemplo: Pasar autómata a gramática lineal por la izquierda



invertido:



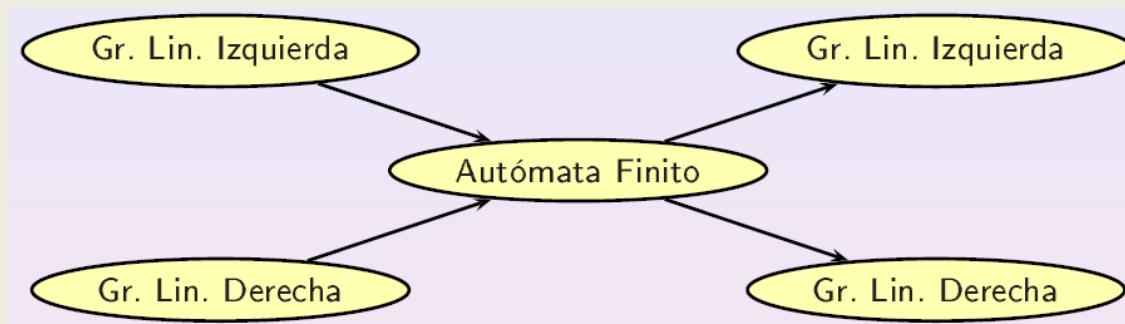
La gramática es (símbolo inicial q_2):

$q_2 \rightarrow 1 q_0$, $q_2 \rightarrow 1 q_1$, $q_2 \rightarrow 0 q_1$, $q_1 \rightarrow 0 q_0$
 $q_1 \rightarrow 1 q_2$, $q_0 \rightarrow 0 q_2$, $q_0 \rightarrow \epsilon$

Invertimos la parte derecha de las producciones:

$q_2 \rightarrow q_0 1$, $q_2 \rightarrow q_1 1$, $q_2 \rightarrow q_1 0$, $q_1 \rightarrow q_0 0$,
 $q_1 \rightarrow q_2 1$, $q_0 \rightarrow q_2 0$, $q_0 \rightarrow \epsilon$

Conclusiones



- Dada cualquier gramática lineal por la derecha, existe otra gramática lineal por la izquierda que genera el mismo lenguaje.
- Dada cualquier gramática lineal por la izquierda, existe otra gramática lineal por la derecha que genera el mismo lenguaje.
- Dada cualquier gramática regular, existe un automata AFD que genera el mismo lenguaje.
- Dado cualquier AFD existe una gramática regular (izquierda o derecha, ambas) que genera el mismo lenguaje.



Autómatas finitos y expresiones regulares

1. Autómatas finitos deterministas
2. Autómatas finitos no deterministas
3. Autómatas finitos con transiciones nulas
4. Expresiones regulares
5. Gramáticas regulares
6. Máquinas de estados finitos



Máquinas de estados finitos

Veremos dos tipos:

- **Máquinas de Mealy**
- **Máquinas de Moore**

¿Para qué sirven?

- Un automata no proporciona salida: Únicamente se conoce si acepta o no una palabra.
- Una máquina de estados finitos sí tiene salida. Se utilizan ampliamente en diseño de automatismos, comportamientos de robots y programas en IA, y un largo etcetera.

Máquinas de Moore

Una máquina de Moore es una sextupla $M=(Q,A,B, \delta, q_0, G)$ donde:

- Q es un conjunto finito llamado conjunto de estados.
- A es un alfabeto llamado alfabeto de entrada
- **B es un alfabeto de salida**
- δ es una aplicación llamada función de transición

$$\delta: Q \times A \rightarrow Q$$

- q_0 es un elemento de Q , llamado estado inicial
- **G es una aplicación $G:Q \rightarrow B$; es decir, una aplicación que asocia una salida a cada estado.**

(Nota: Como vemos una máquina de Moore= automata + alfabeto de salida, sin estados finales)

Máquinas de Moore

Una máquina de Moore es una sextupla $M=(Q,A,B, \delta, q_0, G)$ donde:

- Q es un conjunto finito llamado conjunto de estados.
- A es un alfabeto llamado alfabeto de entrada
- **B es un alfabeto de salida**
- δ es una aplicación llamada función de transición

$$\delta: Q \times A \rightarrow Q$$

- q_0 es un elemento de Q , llamado estado inicial
- **G es una aplicación $G:Q \rightarrow B$; es decir, una aplicación que asocia una salida a cada estado.**

(Nota: Como vemos una máquina de Moore= automata + alfabeto de salida, sin estados finales)

Máquinas de Moore

Si una máquina de Moore lee una cadena de entrada u y pasa por los estados q_0, q_1, \dots, q_n , entonces produce la salida:

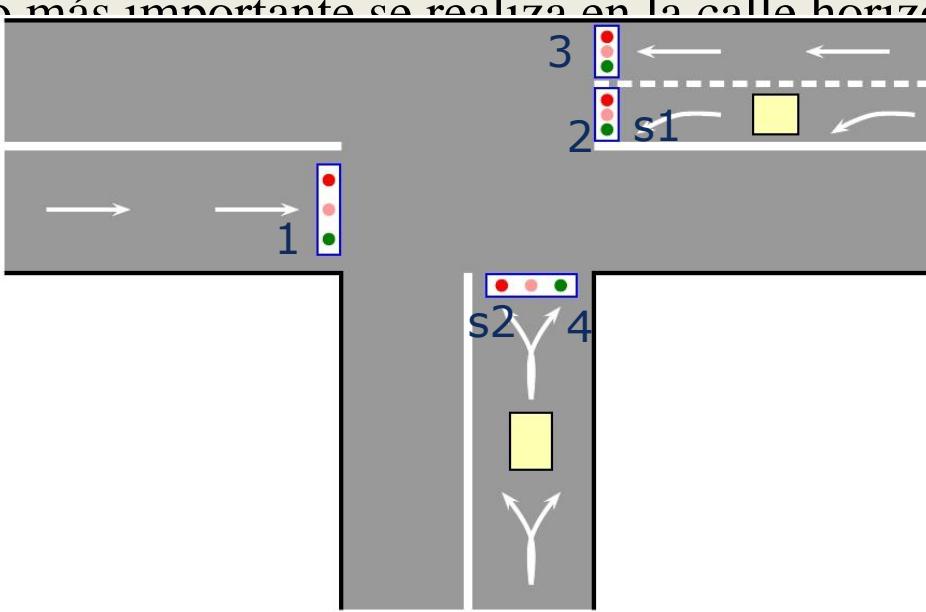
$$G(q_0) G(q_1), \dots, G(q_n)$$



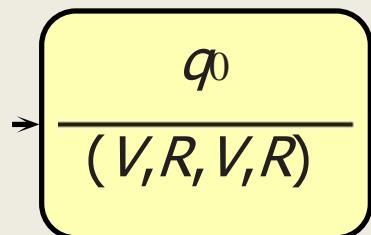
Ejemplo: Gestión de los semáforos en un cruce

- Hay cuatro semáforos: 1,2,3,4.
- El tráfico más importante se realiza en la calle horizontal y, por defecto, los semáforos 1 y 3 están abiertos.
- Hay dos sensores que detectan si hay coches esperando: s1 para el 2 y s2 para el 4.
- Cuando se detectan coches en cualquiera de los dos semáforos 2 ó 4, se cierran los semáforos necesarios y se abre el semáforo para que pasen estos coches.
- El alfabeto de entrada está formado por los pares $(i, j), i, j = 0, 1$, donde i indica si s1 detecta coches y j lo mismo para s2.
- El alfabeto de salida estará formada por los vectores (C_1, C_2, C_3, C_4) , donde C_i es el color del semáforo i . Los posibles colores son: R (Rojo), A (Ámbar), V (Verde).

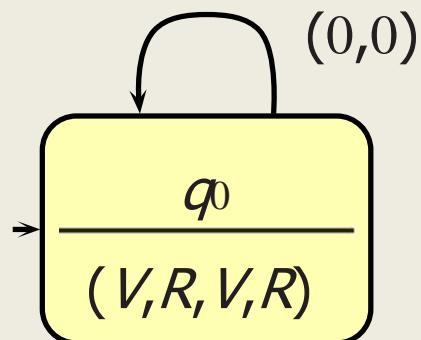
Ejemplo: Gestión de los semáforos en un cruce

- Hay cuatro semáforos: 1,2,3,4.
 - El tráfico más importante se realiza en la calle horizontal y, por defecto,
 - Hay dos para el 2
 - Cuando 4, se cierra que pase
 - El alfabeto donde i es
 - El alfabeto $(C1, C2, C3, C4)$, donde C_i es el color del semáforo i . Los posibles colores son: R (Rojo), A (Ámbar), V (Verde).
- 
- semáforos 2 ó
máforo para
 $i), i, j = 0, 1,$
a s2.

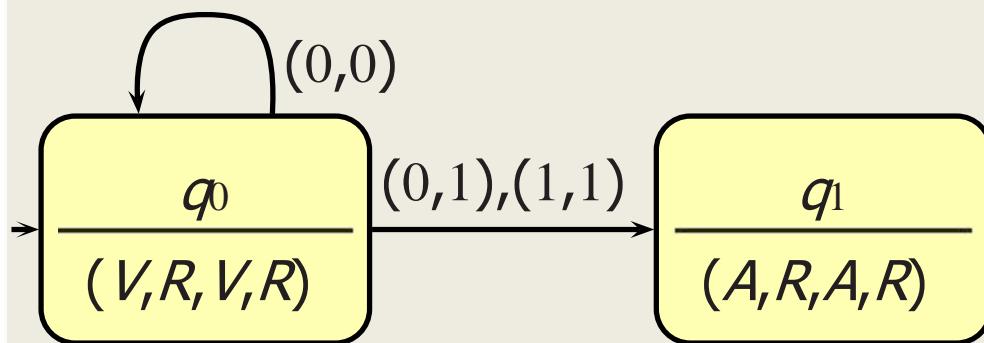
Ejemplo: Gestión de los semáforos en un cruce



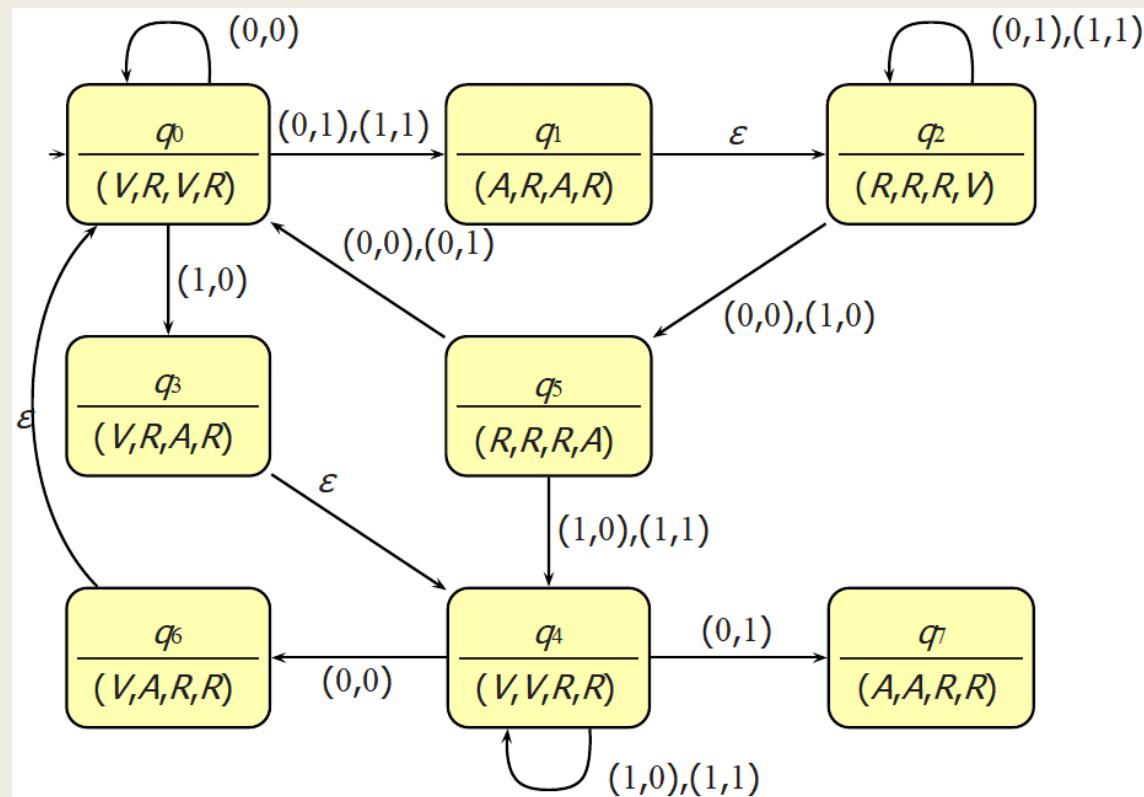
Ejemplo: Gestión de los semáforos en un cruce



Ejemplo: Gestión de los semáforos en un cruce



Ejemplo: Gestión de los semáforos en un cruce



Máquinas de Mealy

Una máquina de Mealy es una máquina de estados finitos con salidas asociadas a las transiciones. Formalmente, es una sextupla $M=(Q,A,B,\delta,q_0,G)$ donde:

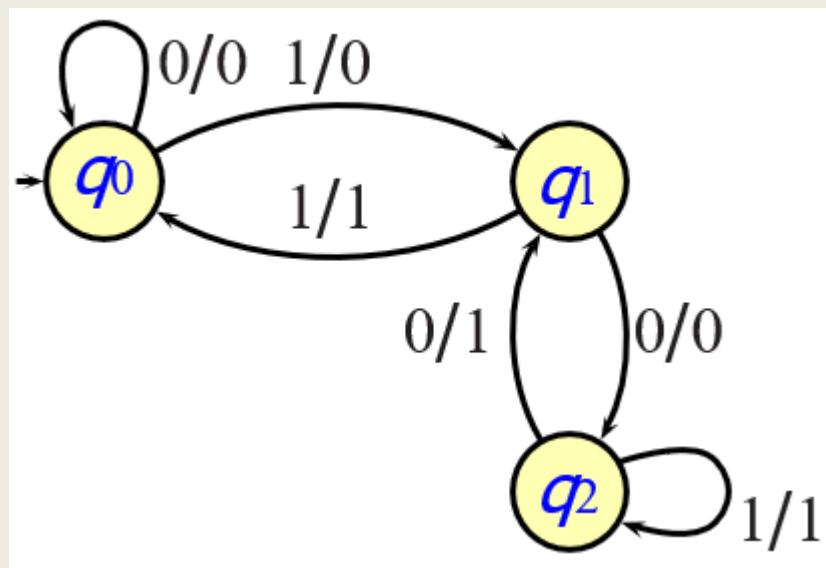
- Q es un conjunto finito llamado conjunto de estados.
- A es un alfabeto llamado alfabeto de entrada
- **B es un alfabeto de salida**
- δ es una aplicación llamada función de transición

$$\delta: Q \times A \rightarrow Q$$

- q_0 es un elemento de Q , llamado estado inicial
- **G es una aplicación $G:Q \times A \rightarrow B$; es decir, una aplicación que asocia una salida a una transición.**

Ejemplo: Máquina de Mealy para calcular la división entre 3

(asumimos división entera)



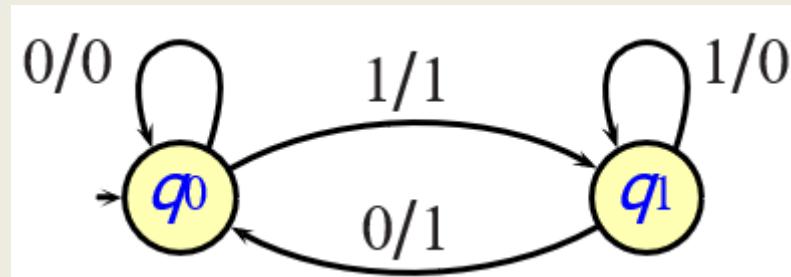
Ejercicio: Máquina de Mealy para codificar

Supongamos que queremos hacer una máquina que codifique una cadena. Los alfabetos de entrada y salida son los mismos, $A=B=\{0,1\}$. Las reglas para codificar son las siguientes:

- El primer símbolo es el mismo $0 \rightarrow 0; 1 \rightarrow 1$
- Los siguientes símbolos se calculan como:
 - Si el anterior es un 0: $0 \rightarrow 0; 1 \rightarrow 1$
 - Si el anterior es un 1: $0 \rightarrow 1; 1 \rightarrow 0$

Es decir, si lee **0101**, la salida correspondiente es **0111**.

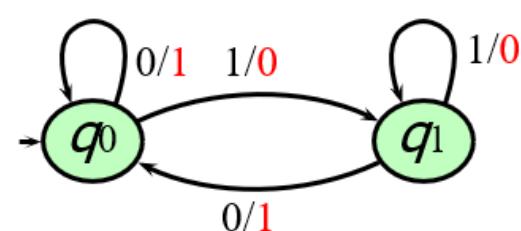
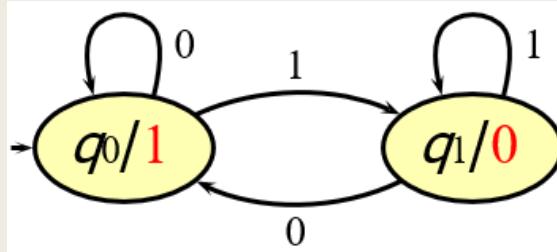
Solución: Máquina de Mealy para codificar



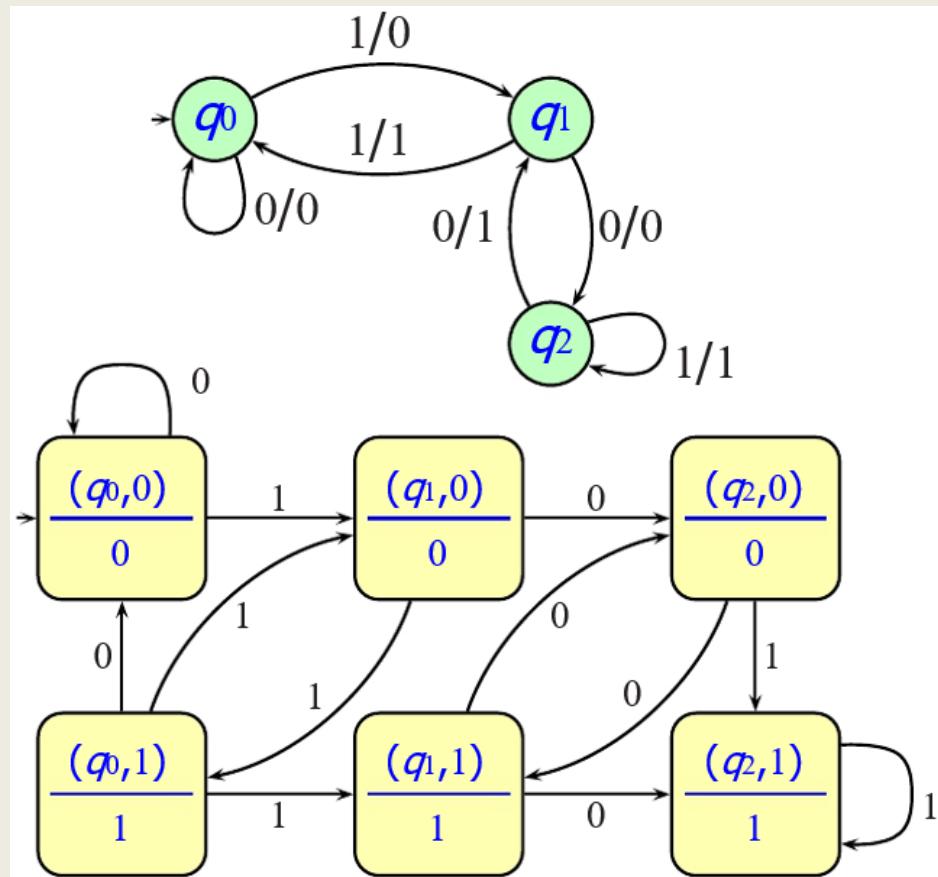
Equivalencias entre máquinas de estados finitos

- Dada una máquina de Moore, existe una máquina de Mealy equivalente.
- Dada una máquina de Mealy, existe una máquina de Moore equivalente.

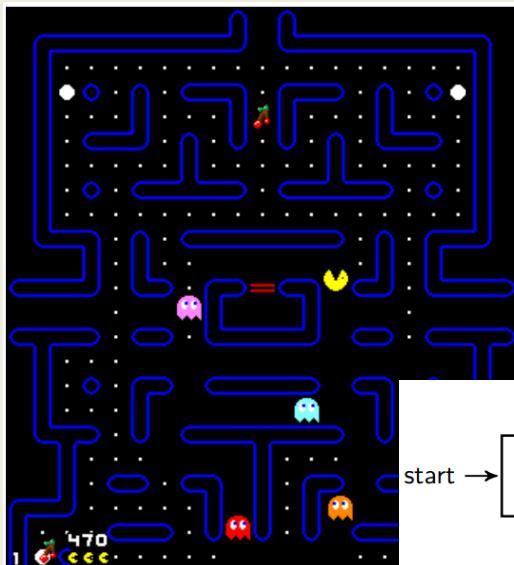
Ejemplo



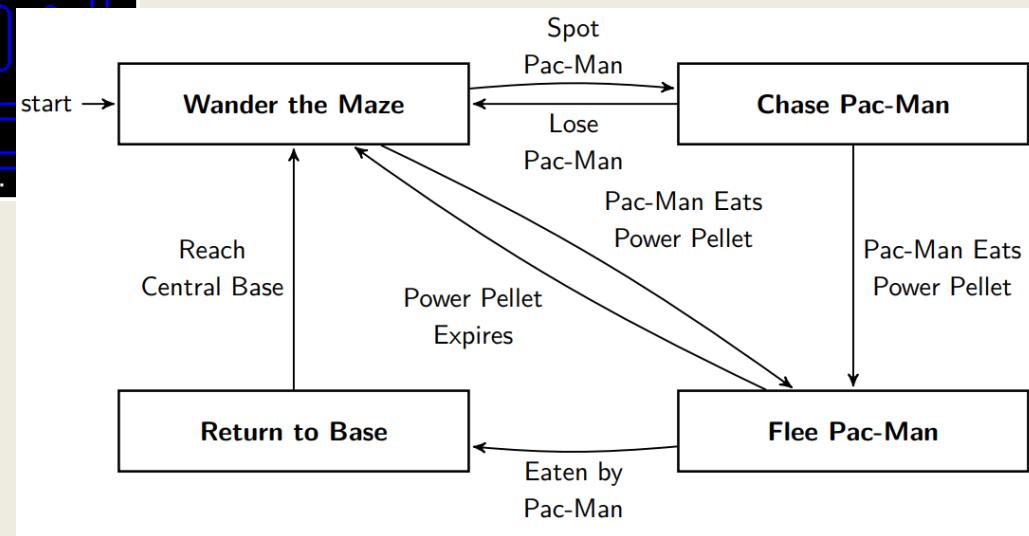
Otro ejemplo



Otro ejemplo



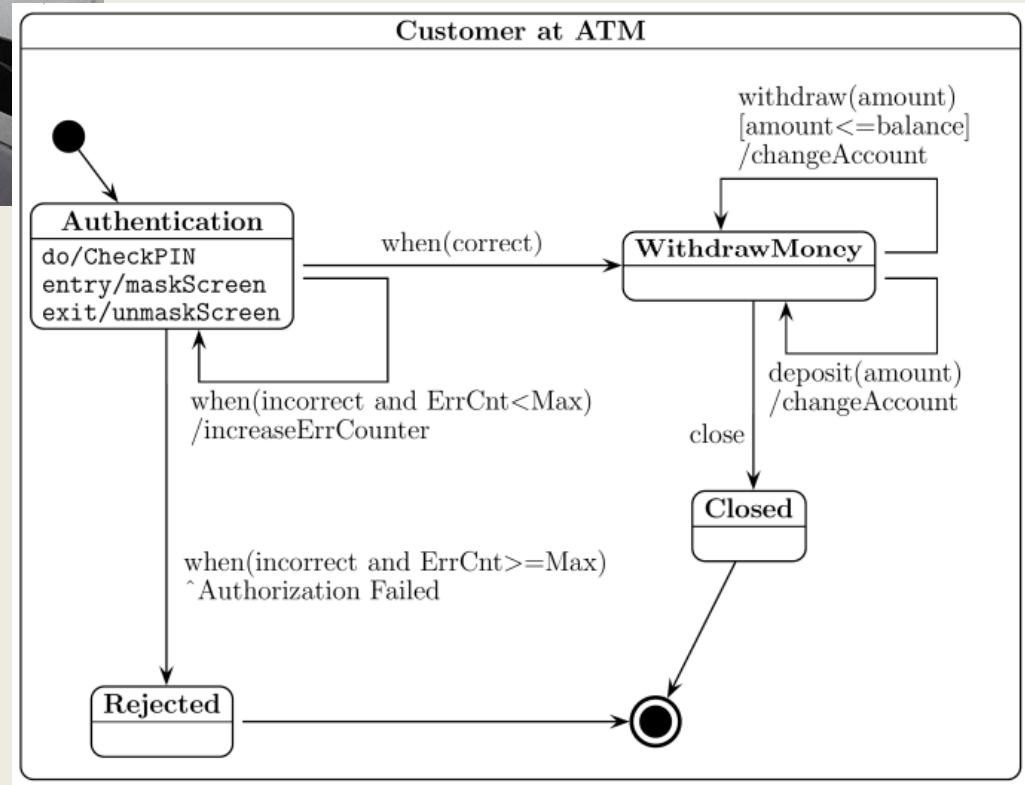
¿Máquina de Mealy o de Moore?



Otro ejemplo



¿Máquina de Mealy o de Moore?





UNIVERSIDAD
DE GRANADA



Modelos de Computación

Grado en Ingeniería Informática

Tema 2 – Autómatas finitos y expresiones regulares

Este documento está protegido por la
Ley de Propiedad Intelectual (Real
Decreto Ley 1/1996 de 12 de abril).
Queda expresamente prohibido su uso o
distribución sin autorización del autor.

Manuel Pegalajar Cuéllar

manupc@ugr.es

Departamento de Ciencias de la
Computación e Inteligencia Artificial
<http://decsai.ugr.es>