Tema 1: Lenguajes Regulares

Lección 1.3

Autómatas Finitos No Deterministas (AFnD)

Jordi Bernad

Índice

1. No determinismo
2. Ejemplos Autómatas Finitos No Deterministas
3. Definición de Autómata Finito no Determinista
4. Equivalencia entre AFnD y AFD
5. Propiedades de clausura de los lenguajes regulares

Algoritmos no deterministas

Problema

Dado un conjunto $S = \{n_1, \ldots, n_k\}$ de números enteros, ¿existe un subconjunto $A = \{n_{i_1}, \ldots, n_{i_s}\} \subseteq S$ cuya suma sea cero?

Solución no determinista

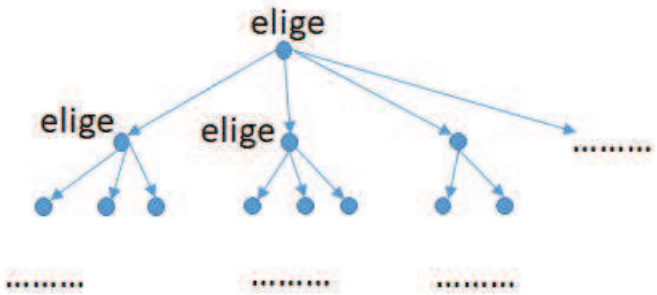
$C = \emptyset$
para i=1 **hasta** $|S|$ **hacer**
    elige $n_i \in$ S\C
    $C = C \cup \{n_i\}$
    si $\sum_{n_j \in C} n_j == 0$ **entonces devolver** cierto
fpara
devolver falso

Algoritmos no deterministas

- En un algoritmo no determinista podemos elegir entre un **número finito** de posibilidades

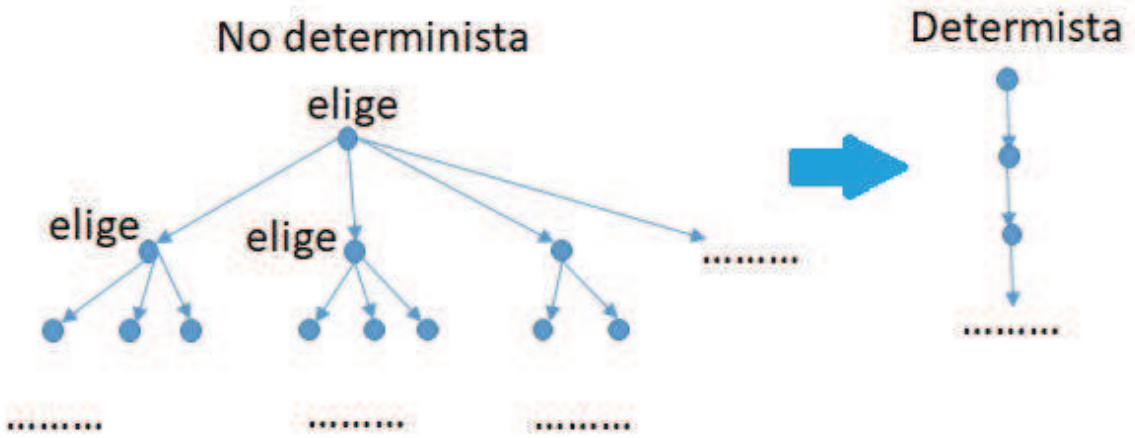
  elige i $\in$ S\C

- Por cada elección se ejecutará la búsqueda de una posible solución.



Problema

- Un computador es determinista.
- Tenemos que indicar explícitamente cómo se elige.
- Tenemos que transformar el algoritmo no determinista en un algoritmo determinista (sin usar `elige`).



¡En muchas ocasiones es más fácil pensar una solución desde un punto de vista no determinístico!

Sería maravilloso...

- Escribir un algoritmo no determinista.
- Pulsar un botón.
- Y obtener un "*buen*" algoritmo determinista.
- Buen algoritmo:
  - Se ejecute en el menor tiempo posible: complejidad en tiempo lineal, cuadrática, cúbica,...
  - Utilice los menores recursos posibles: cuanta menos memoria, mejor.
- ¿Existe ese botón?
- Depende del tipo de problema que intentemos resolver.

Otro problema

Problema

Dada una cadena w, ¿es cierto que w sea igual a una concatenación de subcadenas, donde cada subcadena es de la forma: una a seguida de una cadena formada por a, b, c, y terminada por a?

- Esto es, ¿$w = aw_1aaw_2a \cdots aw_ka$, $k \geq 0$, $w_i \in \{a, b, c\}^*$?
- Esto es, ¿$w \in L\Big(\big(a(a+b+c)^*a\big)^*\Big)$?

Expresiones regulares y algoritmos no deterministas

- Una expresión regular es una forma de describir un algoritmo no determinista.

$$\left(a(a+b+c)^*a\right)^*$$

- Cada vez que aparece el operador $+$:

$$(a+b+c) \quad \rightarrow \quad \text{elige } x \in \{a,b,c\}$$

- Cada vez que aparece el operador estrella de Kleene $(\dots)^*$:

$(a+b+c)^* \quad \rightarrow \quad$ elige cero o más grupos de $\{a,b,c\}$ hasta un máximo de grupos igual a la longitud de w

Preguntas

¿Existe el botón: dado un algoritmo no determinista genera un algoritmo determinista?

Cierto. Con un algoritmo de fuerza bruta.

- Caso subconjunto suma 0: generar todos los subconjuntos y comprobar si alguno suma 0.
- Caso expresión regular: dividir de todas las formas posibles la cadena w en $k = 0, 1, \ldots, |w|$ subcadenas. Comprobar si en alguna de esas divisiones, las k subcadenas son de la forma: una a seguida de un grupo de a, b, c y acabada en a

- Botón poco interesante: un algoritmo que recorre todas las posibilidades es en general fácil de implementar. No necesitamos el botón.

¡Nos gustaría tener un botón inteligente!

Preguntas

¿Existe el botón: dado un algoritmo no determinista genera un algoritmo determinista **"bueno"**?

Depende.

- Existen problemas para los que no hay un buen algoritmo determinista en tiempo polinomial (Tema 4).
- Caso subconjunto suma 0: nadie ha encontrado un buen algoritmo determinista, pero tampoco se ha demostrado que no exista (Tema 4, problemas NP).
- Caso expresión regular: existe un buen algoritmo determinista en tiempo lineal y memoria constante (independiente de la longitud de la cadena que queremos investigar).

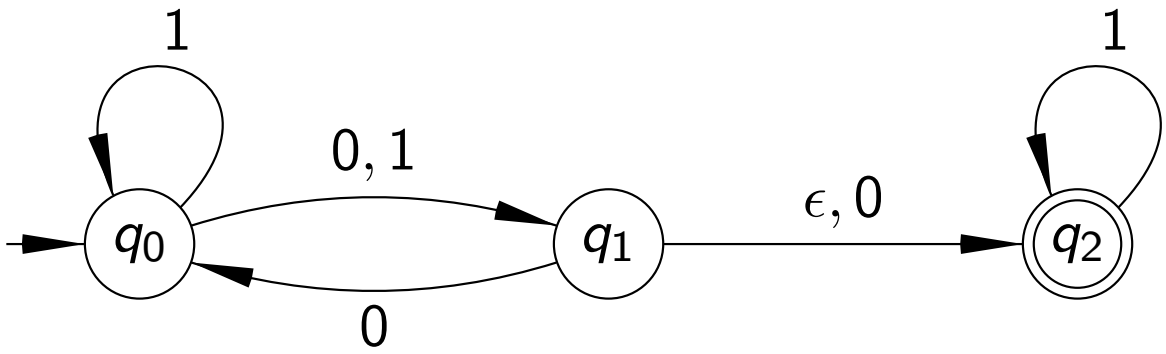
Moraleja

- Si identifico bien el tipo problema que quiero resolver, puedo pensar una solución sencilla no determinista y estar seguro de que podré encontrar un buen algoritmo determinista.
- O saber que no puedo encontrar el algoritmo y explorar otras posibilidades: no encontrar la mejor solución, pero sí una óptima.

Moraleja

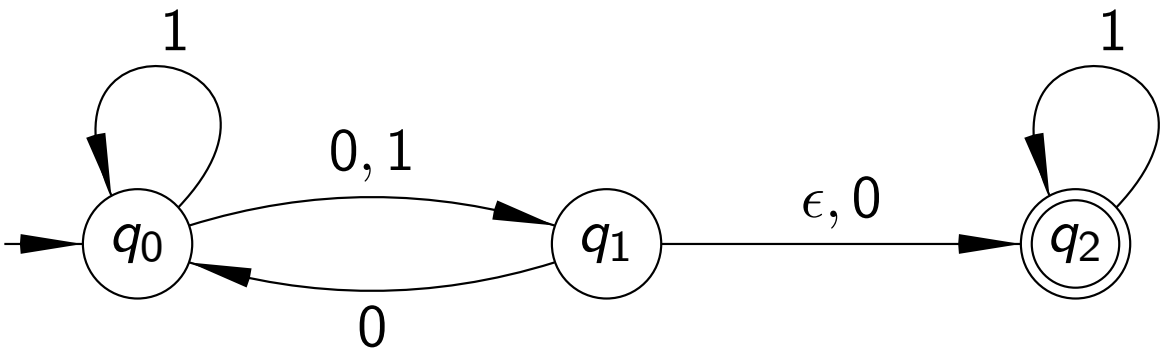
- En otras asignaturas se explorarán diferentes técnicas para pasar de un algoritmo no determinista a otro determinista:
  - Utilizar estructuras de datos para acelarar la búsqueda de la solución
  - Usar el azar a nuestro favor: métodos Montecarlo, algoritmos genéticos
  - Programación dinámica, algoritmos voraces, programación lineal, etc
- En definitiva, identificar un esquema de programación adecuado para resolver el problema.
- Para poder crear el primer botón inteligente necesitamos un poco más de teoría: Autómatas Finitos No Deterministas

Un Autómata Finito No Determinista



- **No determinista (transiciones)**: en un Autómata Finito No Determinista (AFnD), desde un estado puede haber varias transiciones posibles para el mismo símbolo.
- Desde q_0, si leemos el símbolo 1, podemos ir al estado q_1 o quedarnos en el estado q_0.
- **No determinista (estados)**:También pueden aparecer ϵ-transiciones. Sin leer ningún símbolo de la entrada, podemos pasar a otro estado.
- Desde el estado q_1, podemos ir al estado q_2 sin leer ningún símbolo de la entrada.

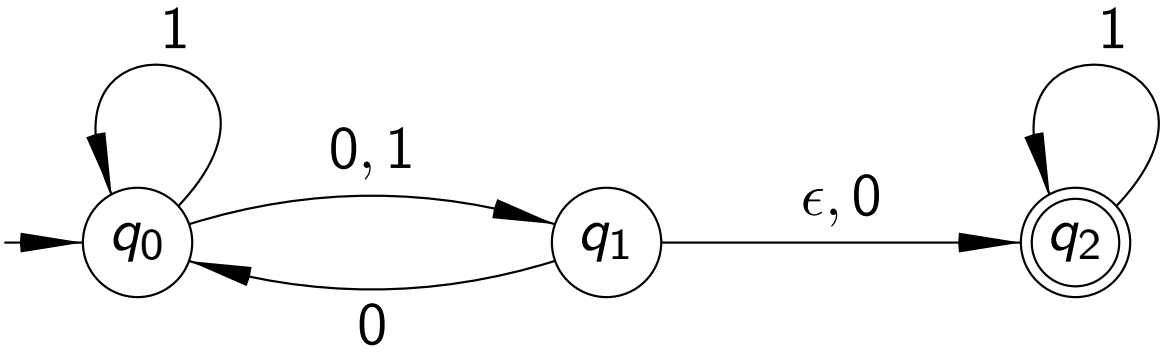
Computación en un AFnD



- Supongamos que tenemos la entrada 11 ¿Es aceptada?
- Tenemos que considerar todas las posibles computaciones.

Computación en un AFnD

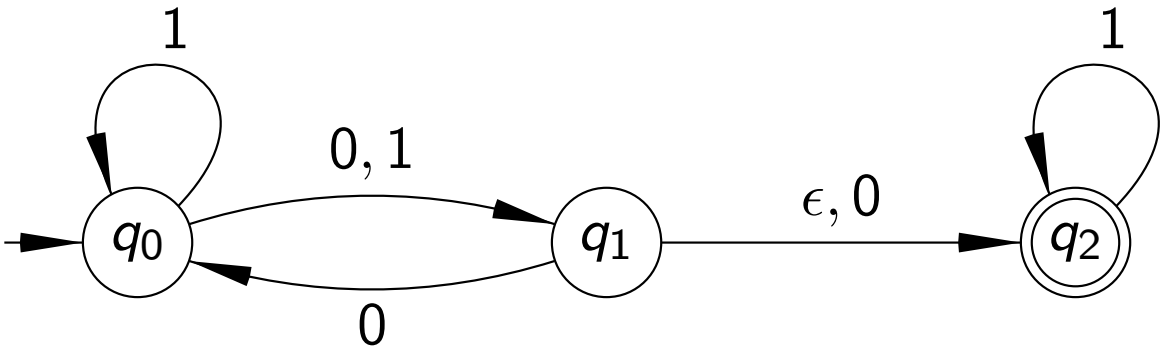
Entrada: 11

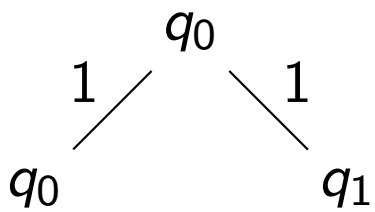


q_0

Computación en un AFnD

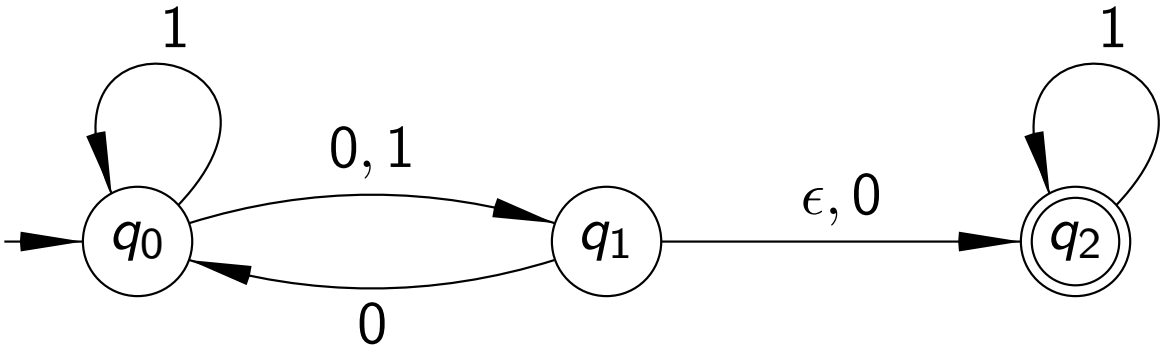
Entrada: 11

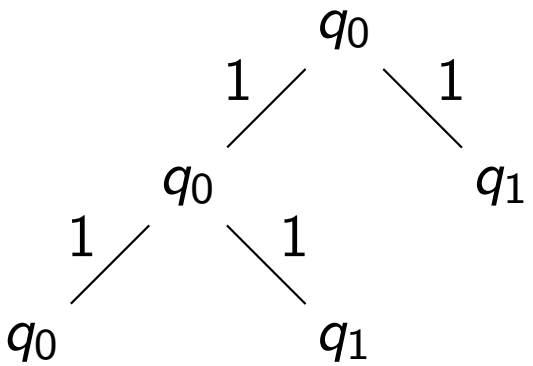




Computación en un AFnD

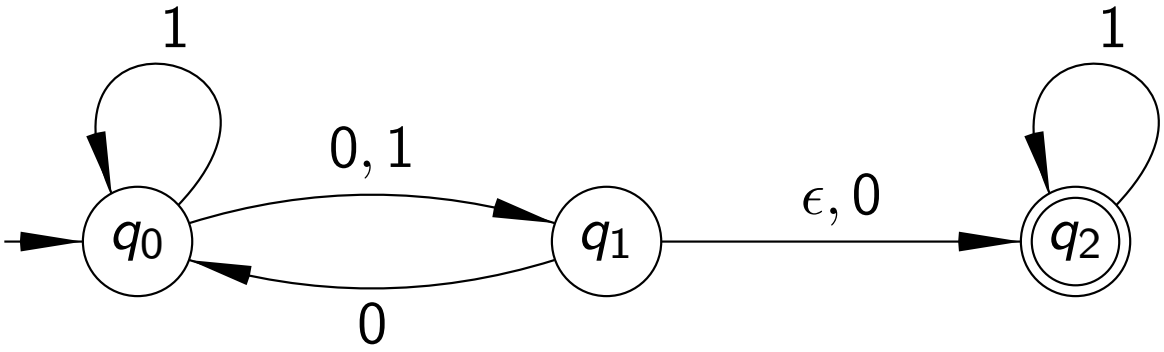
Entrada: 11

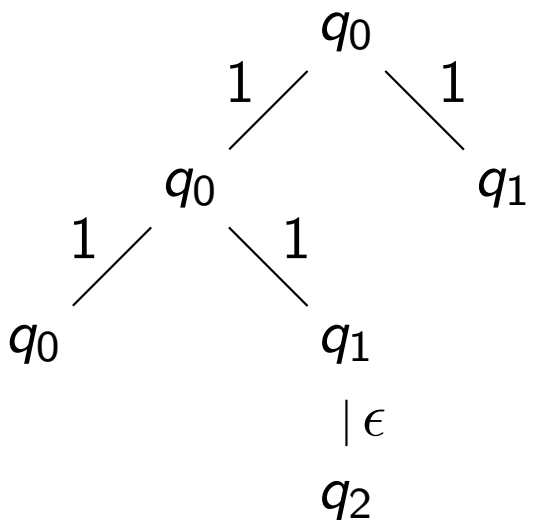




Computación en un AFnD

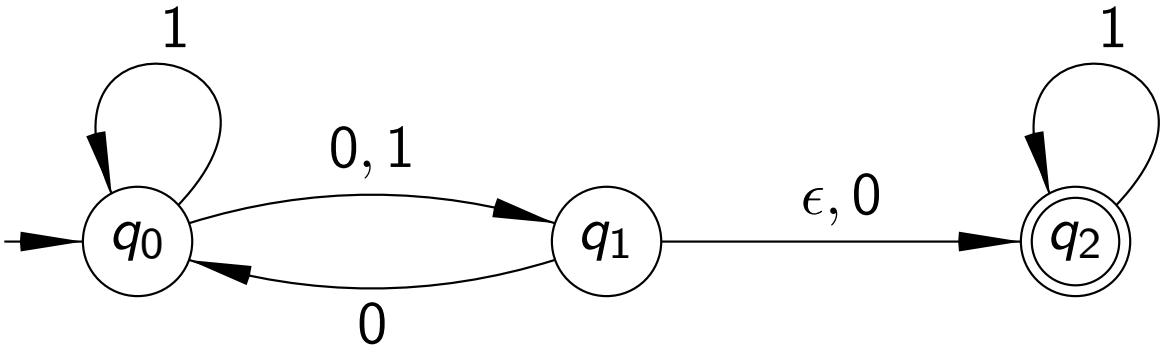
Entrada: 11

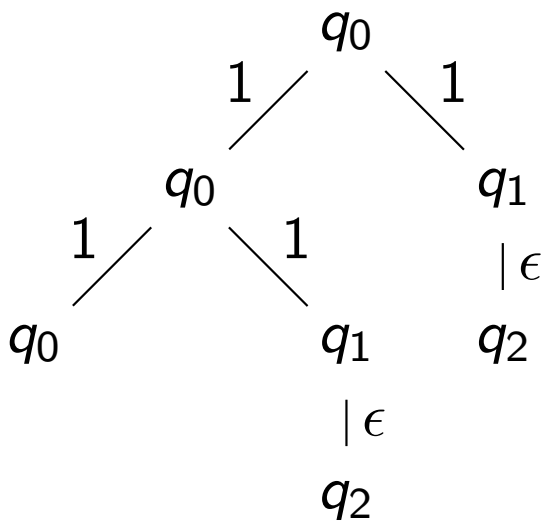




Computación en un AFnD

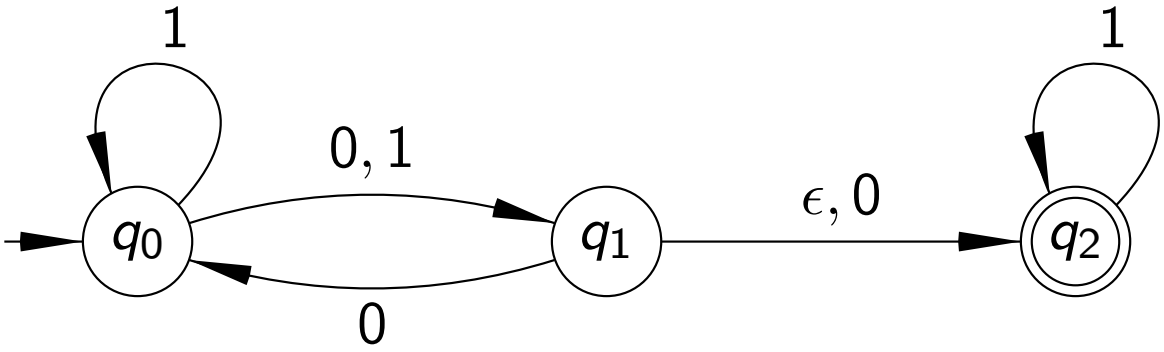
Entrada: 11

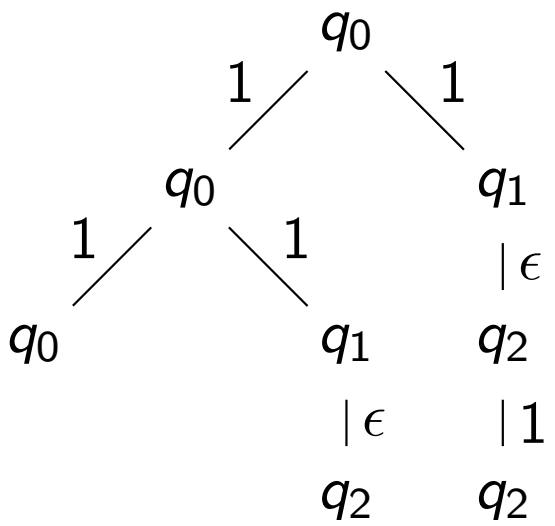




Computación en un AFnD

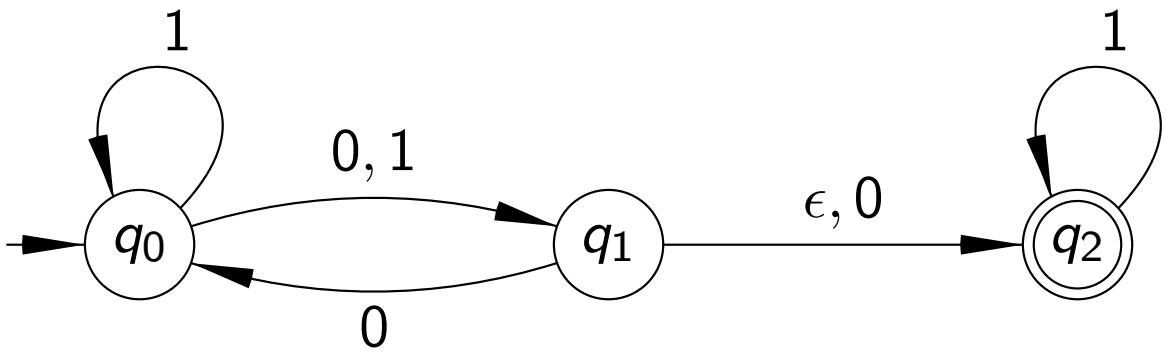
Entrada: 11





Computación en un AFnD

Entrada: 11

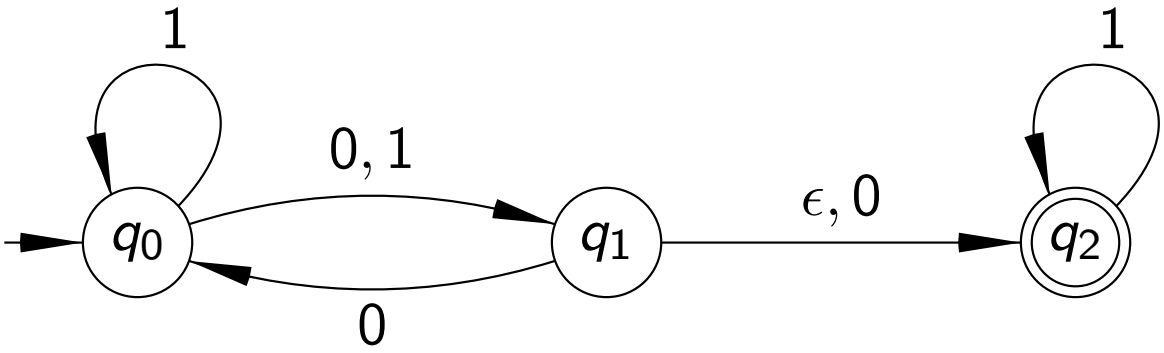


Alguna computación finaliza en un estado final

- (q_0, q_0, q_1, q_2)
- (q_0, q_1, q_2, q_2)
- Por tanto, 11 se acepta.

Computación en un AFnD

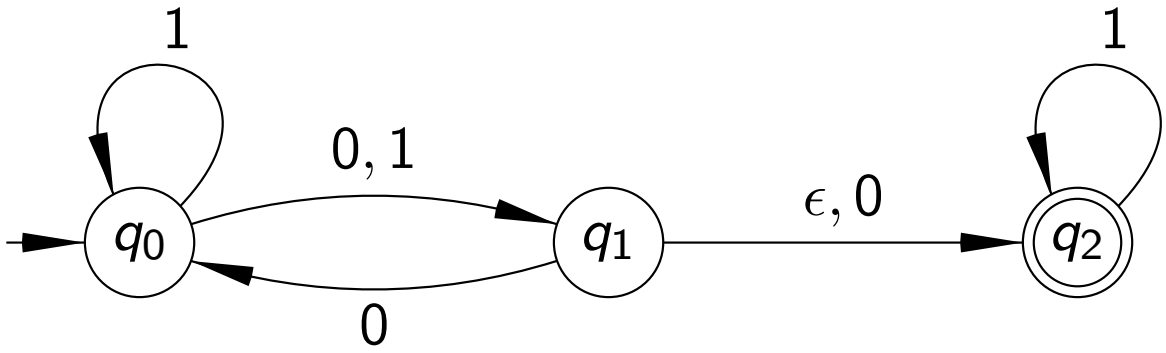
Entrada: 00010



q_0

Computación en un AFnD

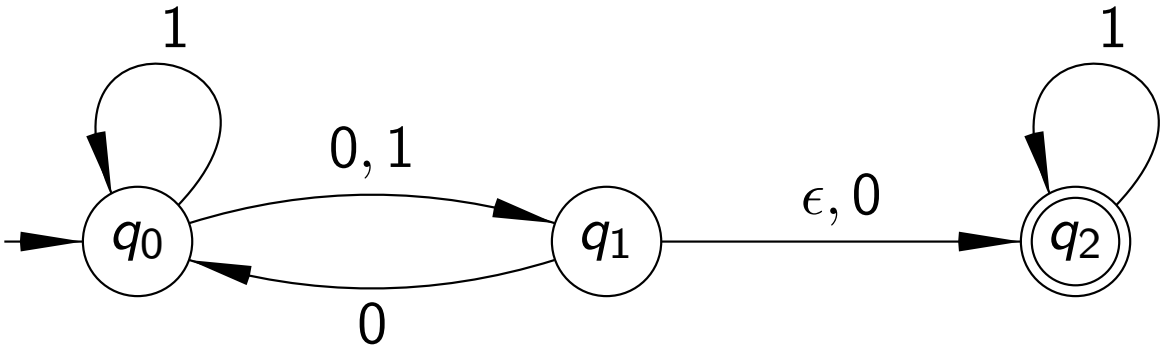
Entrada: 00010

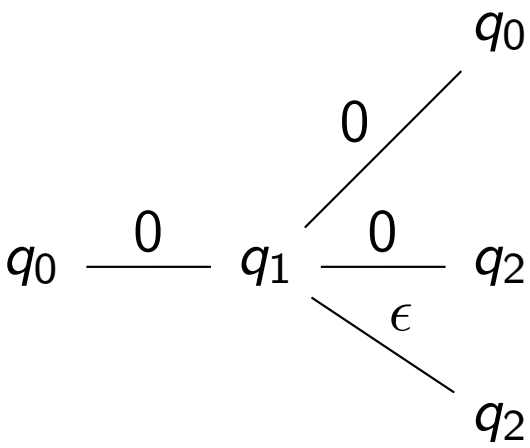


$q_0 \xrightarrow{0} q_1$

Computación en un AFnD

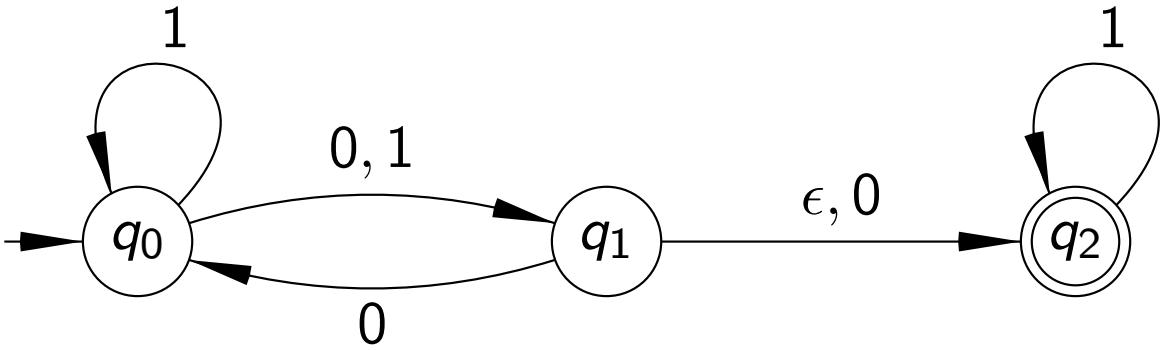
Entrada: 00010

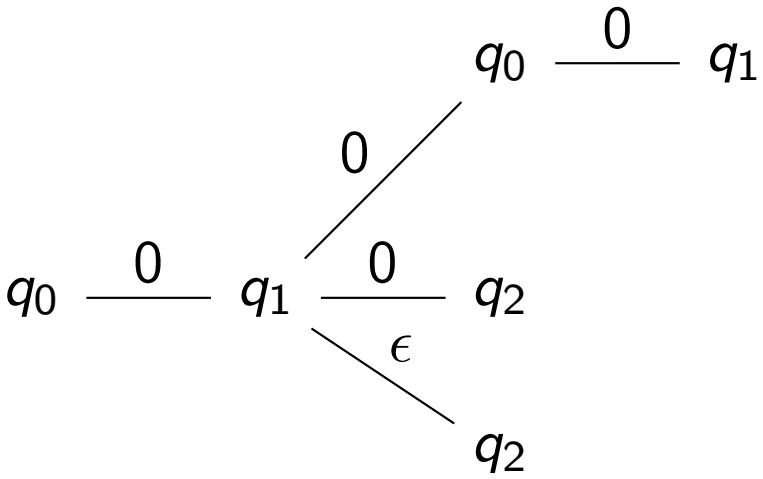




Computación en un AFnD

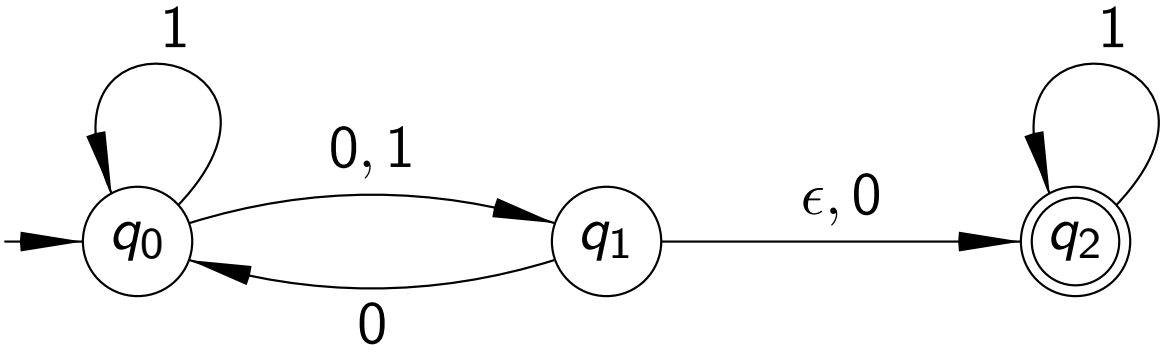
Entrada: 00010

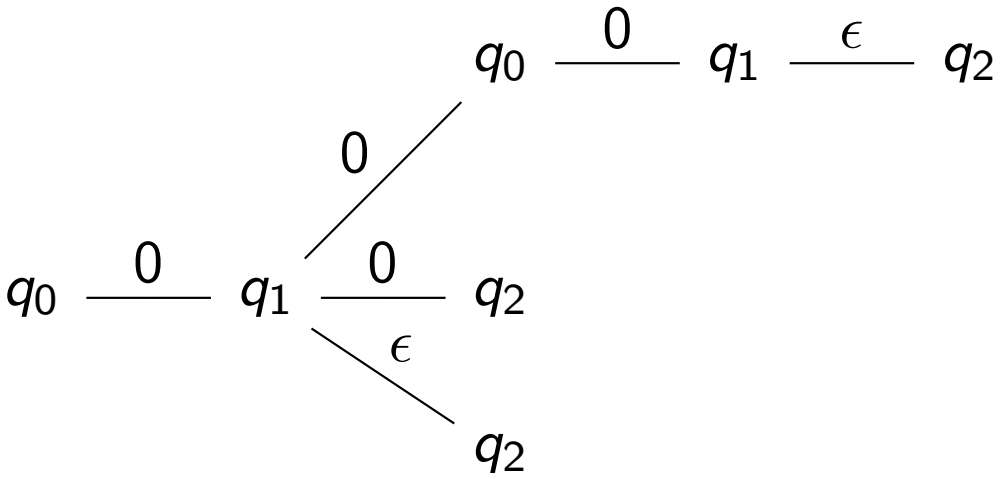




Computación en un AFnD

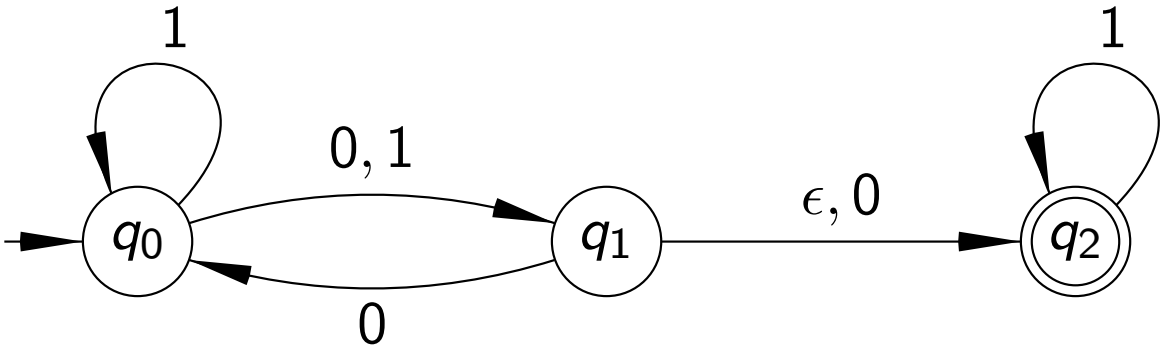
Entrada: 00010

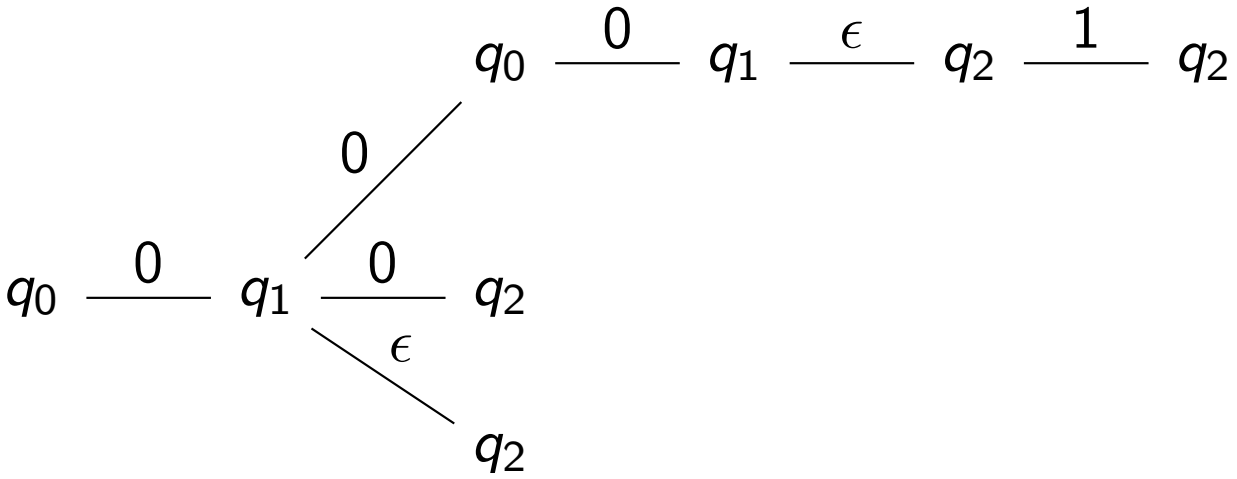




Computación en un AFnD

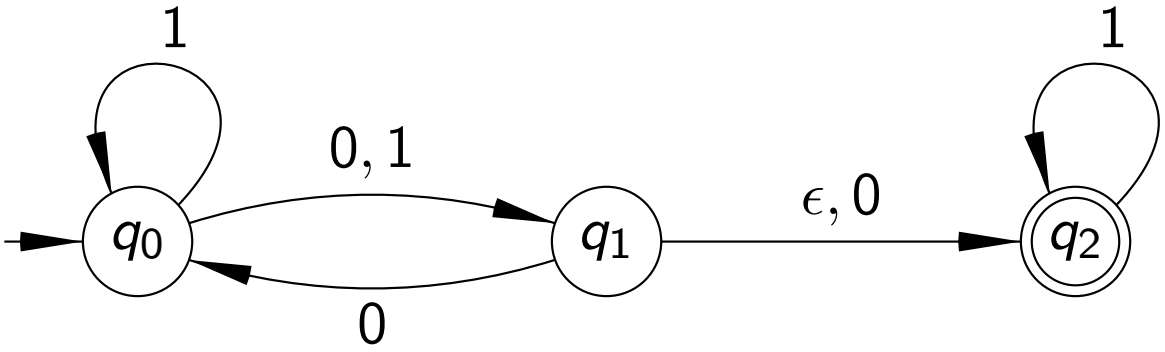
Entrada: 00010

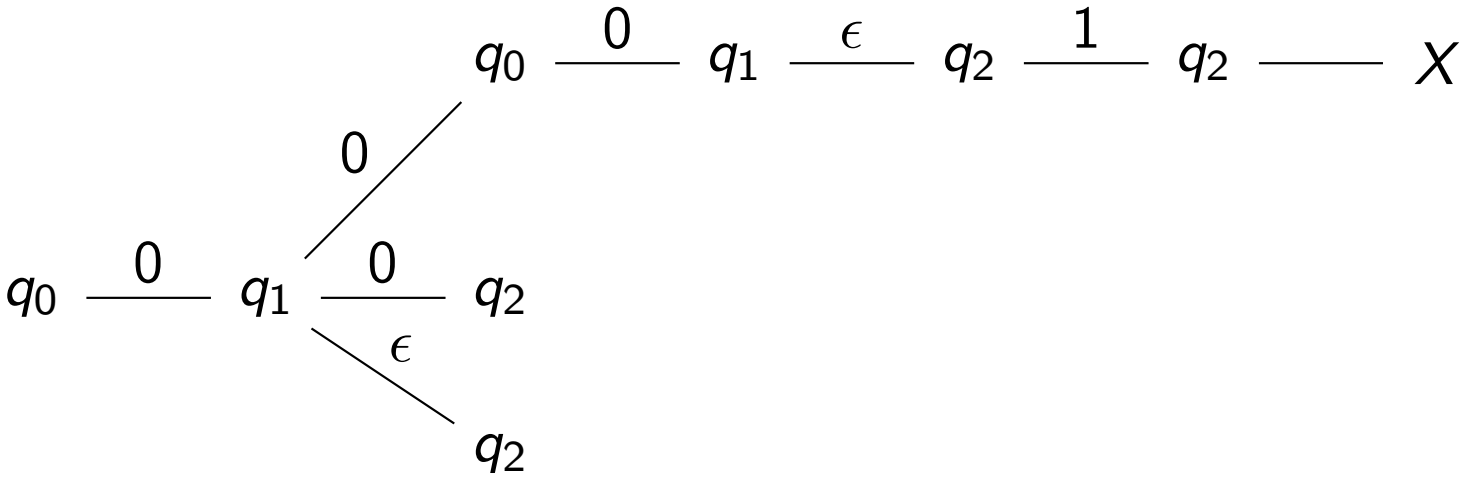




Computación en un AFnD

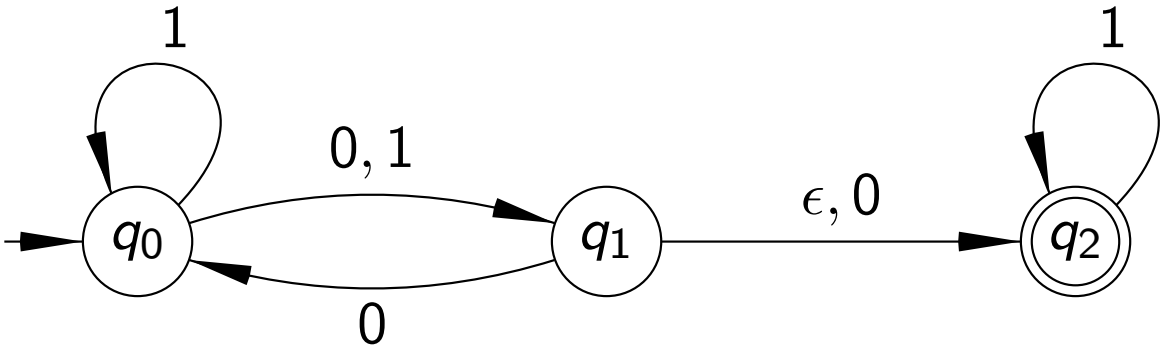
Entrada: 00010

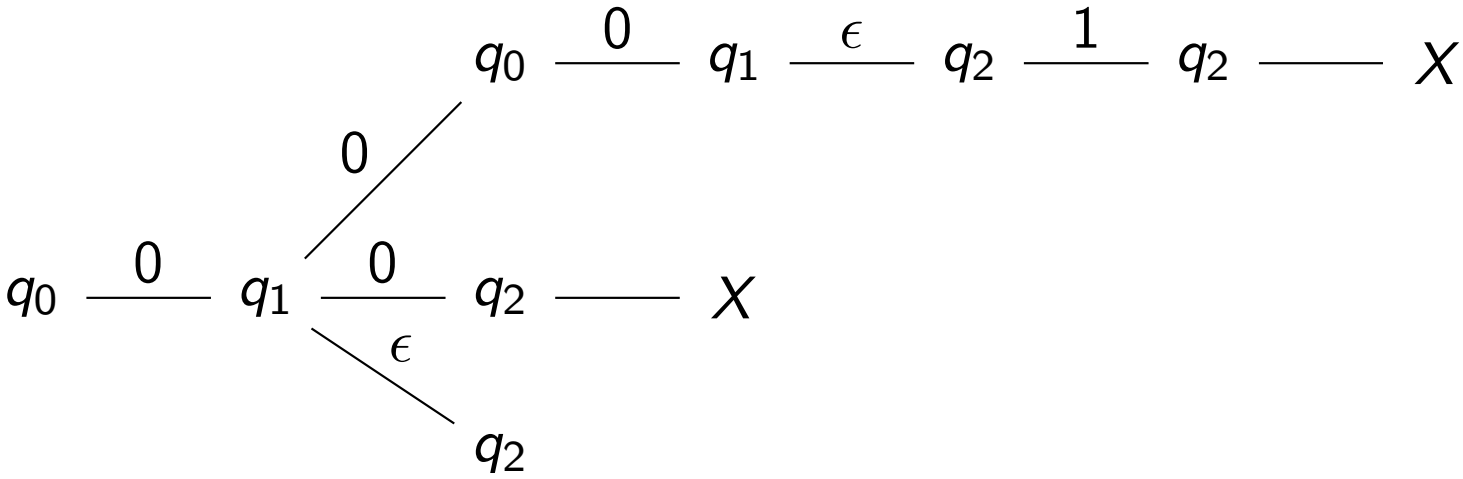




Computación en un AFnD

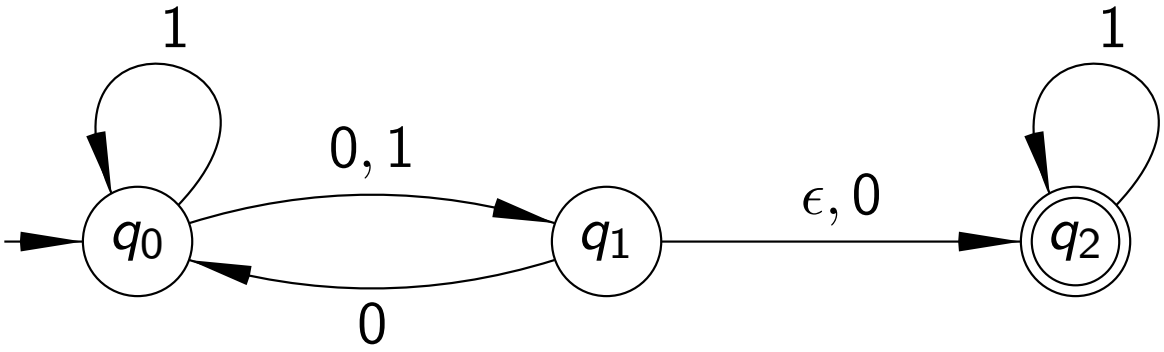
Entrada: 00010

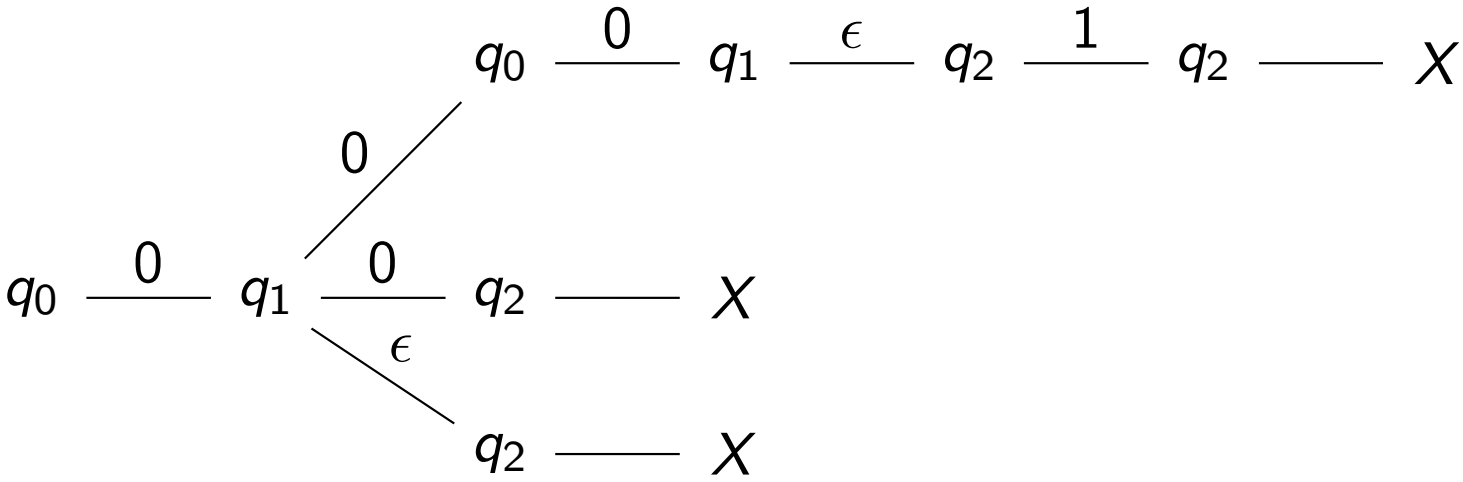




Computación en un AFnD

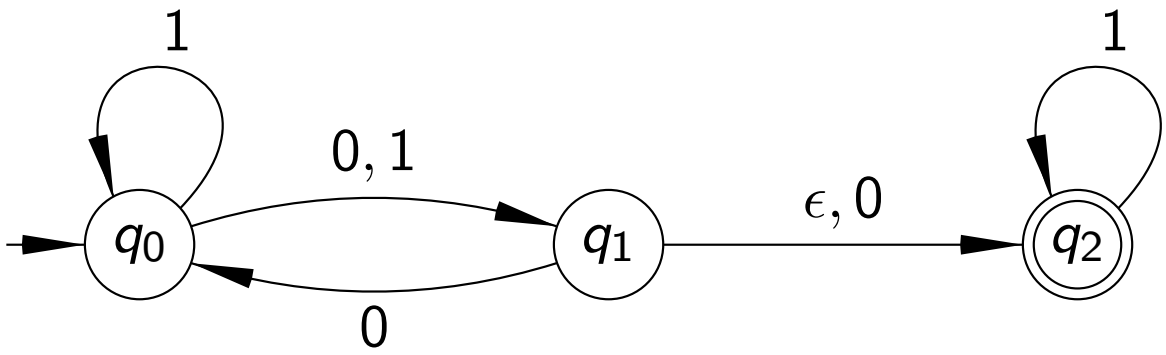
Entrada: 00010





Computación en un AFnD

Entrada: 00010



Ninguna computación finaliza en un estado final

- Se llega a un estado no final
- O no se puede seguir la transición
- Por tanto, 00010 no se acepta.

Definición formal AFnD

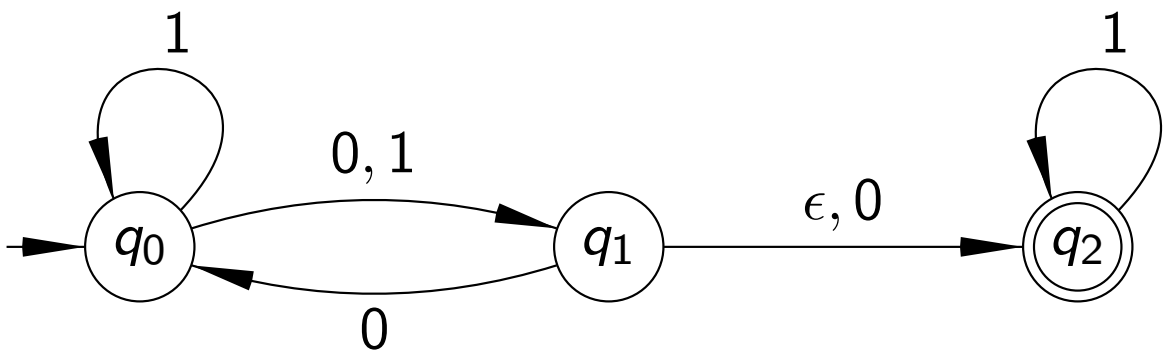
Definición

Un **Autómata Finito no Determinista (AFnD)** es una 5 -tupla $M = (Q, \Sigma, \delta, q_0, F)$ tal que

- Q es el conjunto finito de estados
- Σ es el alfabeto de entrada
- $\delta : Q \times (\Sigma \cup \epsilon) \rightarrow \mathcal{P}(Q)$ es la función de transición $\delta(q, a) = Q'$ quiere decir que si estoy en el estado q y leo el símbolo a puedo ir a cualquiera de los estados $q' \in Q'$
- $q_0 \in Q$ es el estado inicial
- $F \subseteq Q$ es el conjunto de los estados finales o de aceptación.

Representación de un AFnD

- La representación más usual sera un diagrama



Representación de un AFnD

También podemos indicar quiénes son los estados, el alfabeto, la tabla de transiciones, el estado inicial y los estados finales.

1. $Q = \{q_0, q_1, q_2\}$
2. $\Sigma = \{a, b\}$
3. La tabla de transiciones

δ	0	1	ϵ
q_0	$\{q_1\}$	$\{q_0, q_1\}$	$\emptyset$
q_1	$\{q_0, q_2\}$	$\emptyset$	$\{q_2\}$
q_2	$\emptyset$	q_2	$\emptyset$

4. Estado inicial q_0 (si se llama q_0 no hace falta decirlo)
5. $F = \{q_2\}$

Definición formal computación en un AFnD

Definición

Un AFnD $M = (Q, \Sigma, \delta, q_0, F)$ **acepta** una cadena $w = a_1 a_2 \cdots a_n$, $a_i \in \Sigma \cup \{\epsilon\}$, si existe una secuencia de estados $r_0, r_i, \ldots, r_n$ cumpliendo

1. $r_0 = q_0$
2. $r_{i+1} \in \delta(r_i, a_{i+1})$, $0 \leq i < n$
3. $r_n \in F$

Lenguaje reconocido por un AFnD

Definición

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un AFnD, denotaremos por $L = L(M)$ al lenguaje

$$L(M) = \{w \mid M \text{ acepta } w\}$$

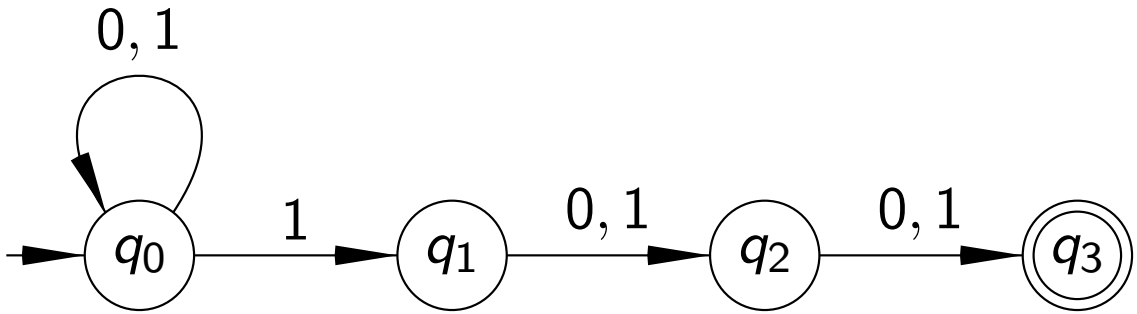
Diremos que el lenguaje L es **reconocido** por M.

Uso de AFnD

- Con AFnD podemos diseñar más fácilmente autómatas que reconozcan lenguajes.
- ¿Autómata que reconozca todas las cadenas cuyo antepenúltimo símbolo es un 1?

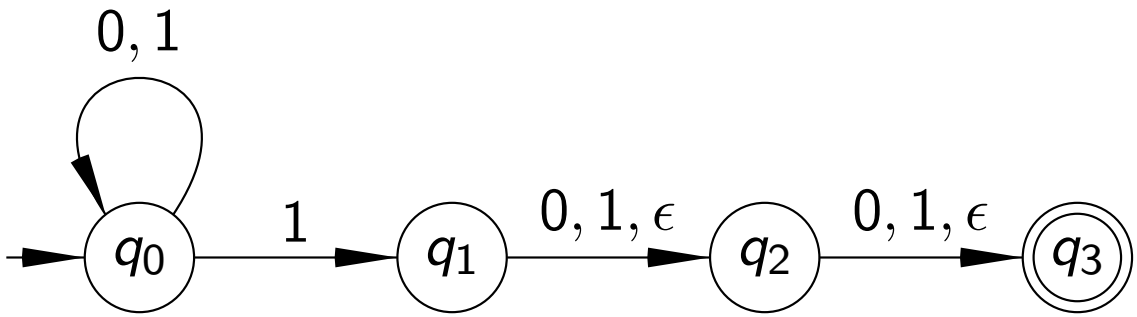
Uso de AFnD

- Con AFnD podemos diseñar más fácilmente autómatas que reconozcan lenguajes.
- ¿Autómata que reconozca todas las cadenas cuyo antepenúltimo símbolo es un 1?



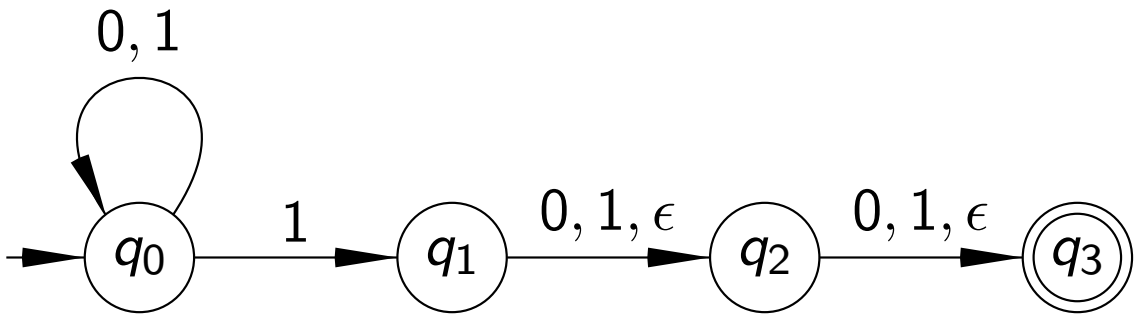
Uso de AFnD

- ¿Qué lenguaje reconoce el siguiente AFnD?



Uso de AFnD

- ¿Qué lenguaje reconoce el siguiente AFnD?



Todas las cadenas cuyo último, penúltimo o antepenúltimo símbolo es un 1

Pregunta

- ¿Qué relación hay entre los lenguajes reconocidos por AFDs y AFnDs?

Pregunta

- ¿Qué relación hay entre los lenguajes reconocidos por AFDs y AFnDs?
- Es obvio que si un lenguaje es reconocido por un AFD, también es reconocido por un AFnD.
- Un AFD es un caso particular de AFnD.

Pregunta

- ¿Qué relación hay entre los lenguajes reconocidos por AFDs y AFnDs?
- Es obvio que si un lenguaje es reconocido por un AFD, también es reconocido por un AFnD.
- Un AFD es un caso particular de AFnD.
- ¿Y lo contrario es cierto? ¿Si un lenguaje es reconocido por un AFnD, también es reconocido por un AFD?

Pregunta

- ¿Qué relación hay entre los lenguajes reconocidos por AFDs y AFnDs?
- Es obvio que si un lenguaje es reconocido por un AFD, también es reconocido por un AFnD.
- Un AFD es un caso particular de AFnD.
- ¿Y lo contrario es cierto? ¿Si un lenguaje es reconocido por un AFnD, también es reconocido por un AFD?
- Demostraremos que es cierto.
- En otras palabras, los lenguajes reconocidos por AFnDs son regulares.

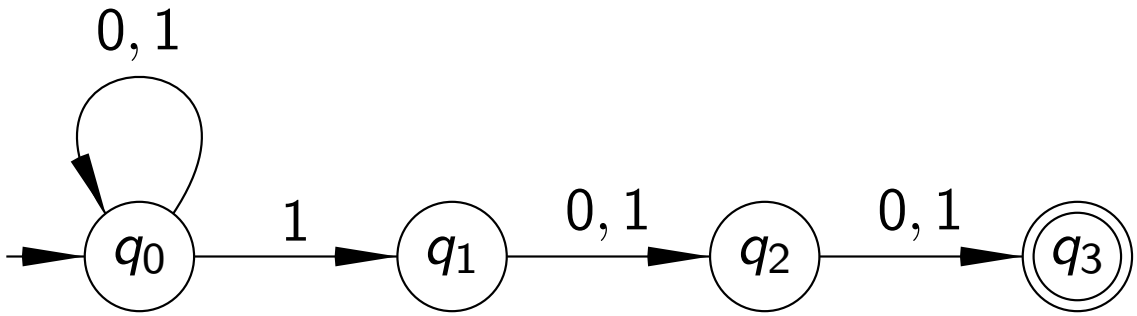
Equivalencia entre AFnDs y AFDs

Teorema

Sea $L(M)$ el lenguaje reconocido por un AFnD M. Entonces existe un AFD, M', tal que $L(M) = L(M')$.

Construcción del AFD equivalente sin ϵ-transiciones

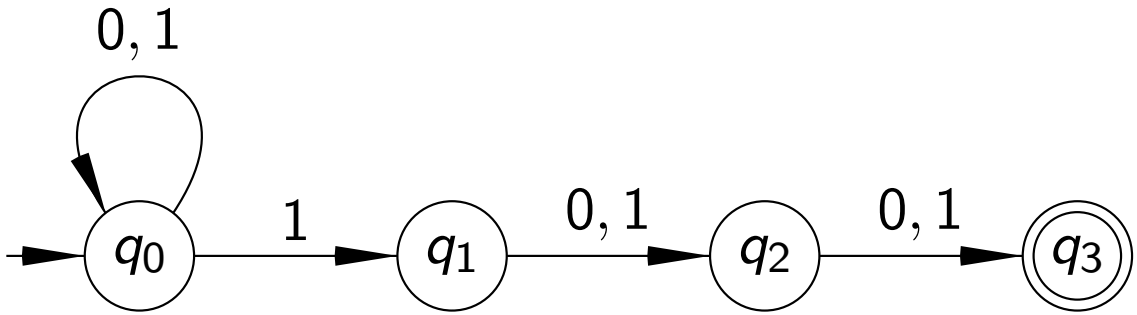
Partimos de un AFnD, $M = (Q, \Sigma, \delta, q_0, F)$
Vamos a construir un AFD equivalente $M' = (Q', \Sigma, \delta', q_0', F')$:



1. Empezar con una tabla con columnas para cada símbolo a del alfabeto.

δ'	0	1

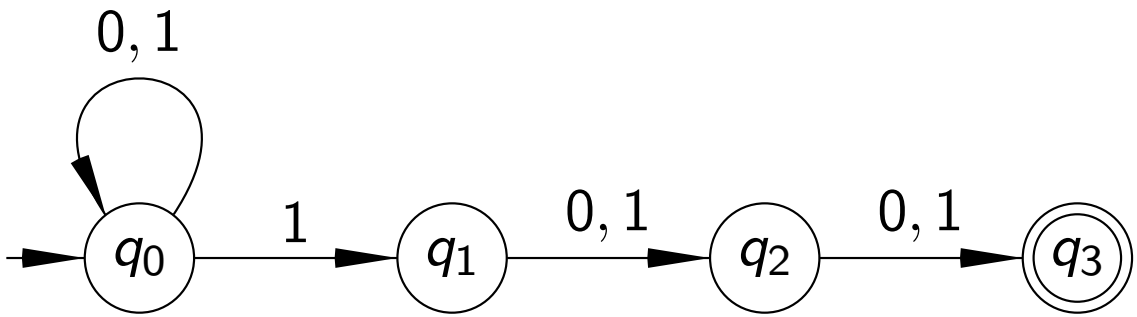
Construcción del AFD equivalente sin ϵ-transiciones



2. La primera fila de la tabla contendrá un nuevo estado, denotado por $\{q_0\}$. Para cada columna $a \in \Sigma$, pondremos el conjunto de estados a los que se llega desde q_0 mediante el símbolo a: $\delta'(\{q_0\}, a) = \delta(q_0, a)$

δ'	0	1
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$

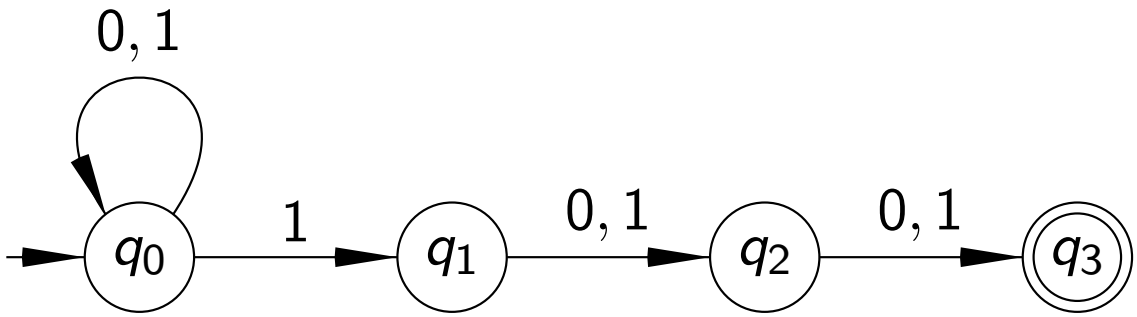
Construcción del AFD equivalente sin ϵ-transiciones



3. Copiar las casillas que han aparecido nuevas como nuevas filas (nuevos estados)

δ'	0	1
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$		

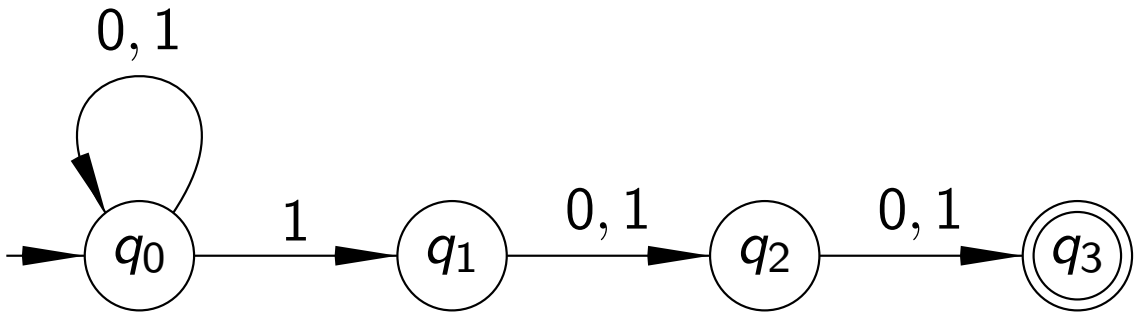
Construcción del AFD equivalente sin ϵ-transiciones



4. Para cada fila nueva R, poner en la columna a todos los estados a los que puedo llegar desde algún estado R con entrada a

δ'	0	1
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

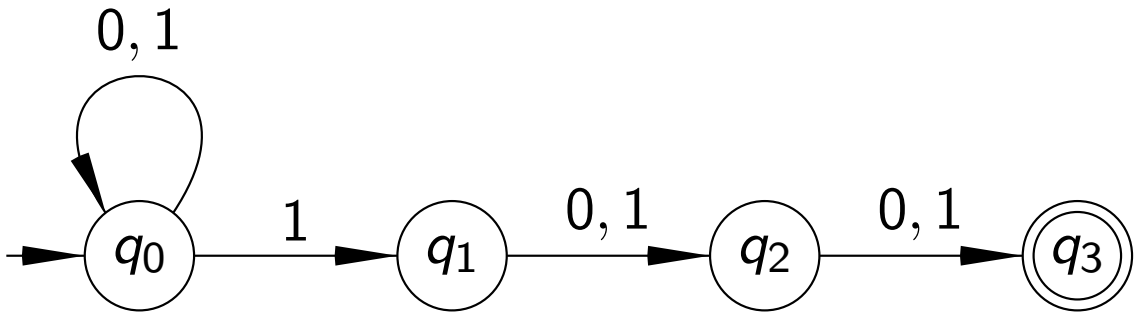
Construcción del AFD equivalente sin ϵ-transiciones



5. Repetir los pasos 3 y 4 hasta que no aparezcan nuevas filas.

δ'	0	1
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

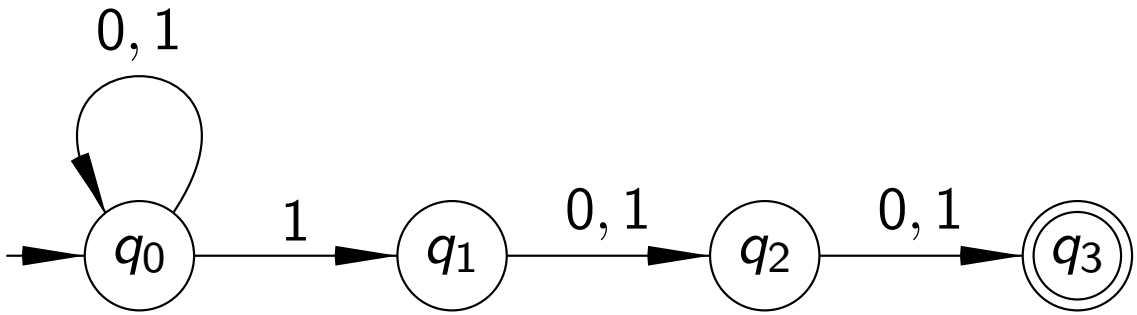
Construcción del AFD equivalente sin ϵ-transiciones



5. Repetir los pasos 3 y 4 hasta que no aparezcan nuevas filas.

δ'	0	1
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$

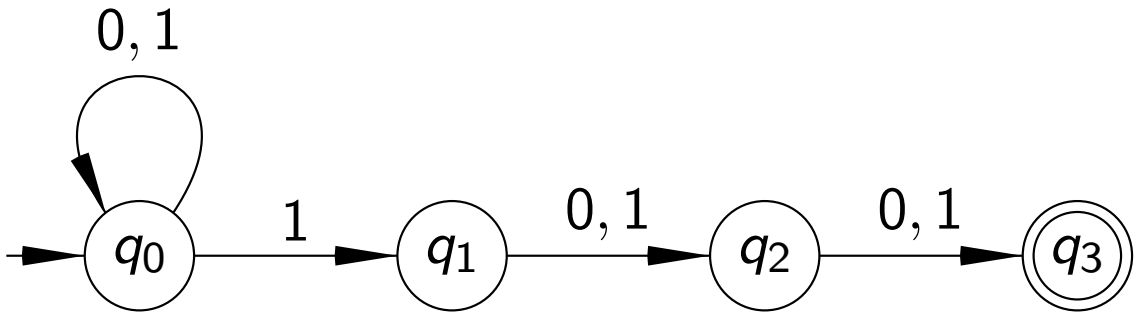
Construcción del AFD equivalente sin ϵ-transiciones



5. Repetir los pasos 3 y 4 hasta que no aparezcan nuevas filas.

δ'	0	1
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$

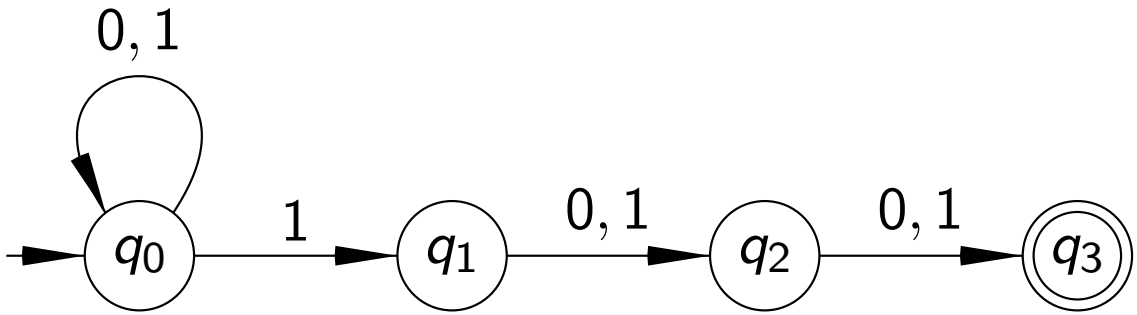
Construcción del AFD equivalente sin ϵ-transiciones



5. Repetir los pasos 3 y 4 hasta que no aparezcan nuevas filas.

δ'	0	1
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$
$\{q_0, q_3\}$	$\{q_0\}$	$\{q_0, q_1\}$

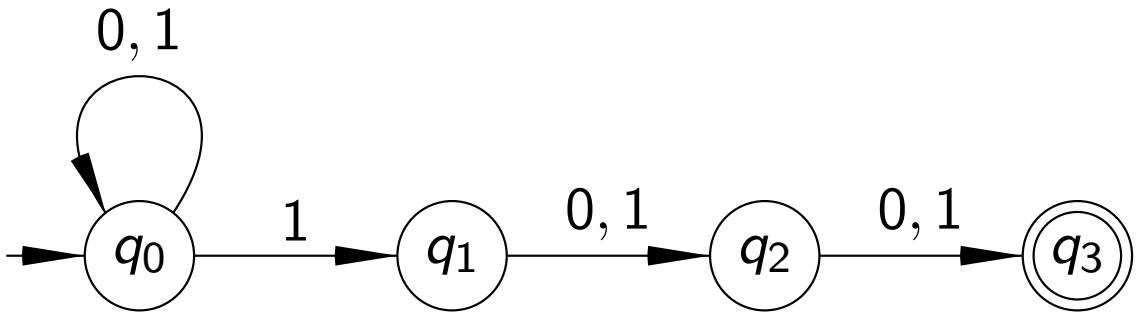
Construcción del AFD equivalente sin ϵ-transiciones



5. Repetir los pasos 3 y 4 hasta que no aparezcan nuevas filas.

δ'	0	1
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$
$\{q_0, q_3\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

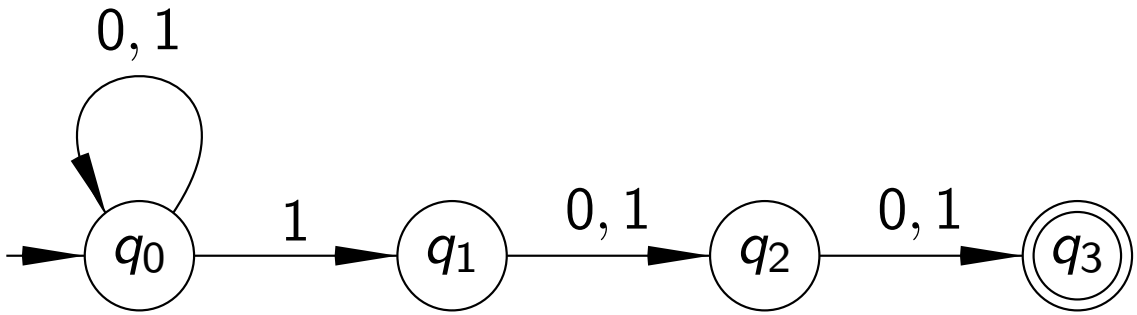
Construcción del AFD equivalente sin ϵ-transiciones



5. Repetir los pasos 3 y 4 hasta que no aparezcan nuevas filas.

δ'	0	1
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$
$\{q_0, q_3\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2, q_3\}$	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$

Construcción del AFD equivalente sin ϵ-transiciones

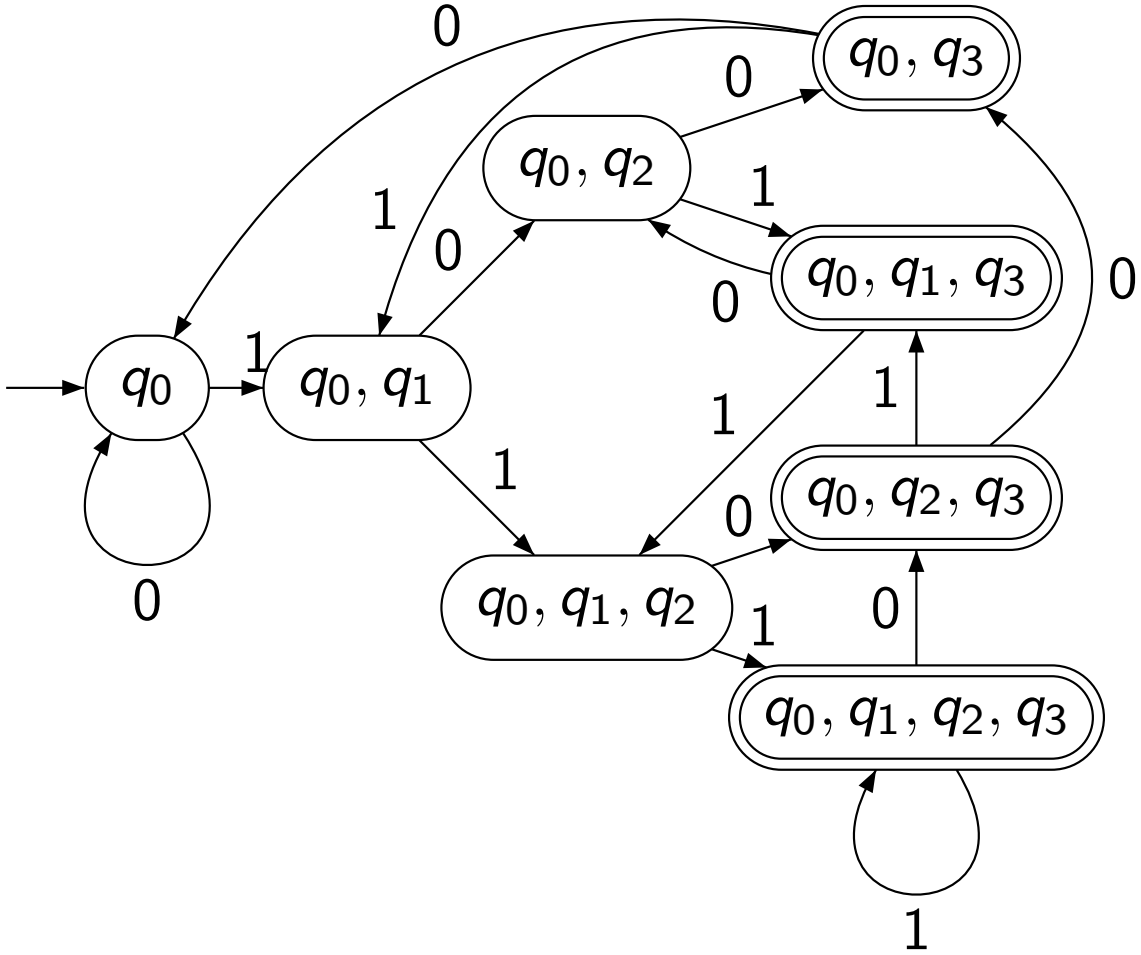


5. Repetir los pasos 3 y 4 hasta que no aparezcan nuevas filas.

δ'	0	1
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$
$\{q_0, q_3\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2, q_3\}$	$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$
$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$

Construcción del AFD equivalente sin ϵ-transiciones

6. Los estados finales del AFD equivalente son todas aquellas filas que contengan algún estado final.



Determinizar AFnD con ϵ-transiciones

ϵ-clausura de un estado q

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un AFnD. La ϵ-clausura de un estado $q \in Q$ se define:
$\epsilon\text{-}c(q) = \{q' \mid \text{estado alcanzable desde } q \text{ sin consumir símbolos}\}$

ϵ-clausura de un conjunto de estados $\{q_1, q_2, \ldots, q_k\}$

$$\epsilon\text{-}c(\{q_1, q_2, \ldots, q_k\}) = \cup_{i=1}^{k} \epsilon\text{-}c(q_i)$$

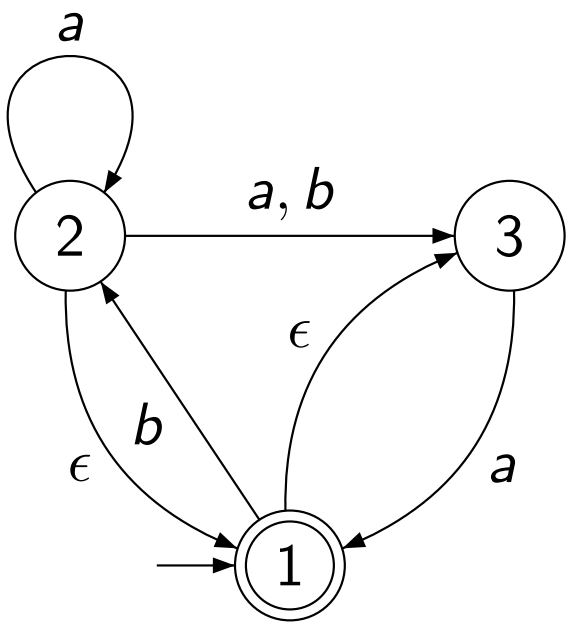
Determinizar AFnD con ϵ-transiciones

Estados alcanzables por un símbolo

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un AFnD. Los estados alcanzables desde un conjunto de estados $\{q_1, q_2, \ldots, q_k\} \subseteq Q$ por el símbolo $a \in \Sigma$ se define:

$$\delta(\{q_1, q_2, \ldots, q_k\}, a) = \cup_{i=1}^{k} \delta(q_i, a)$$

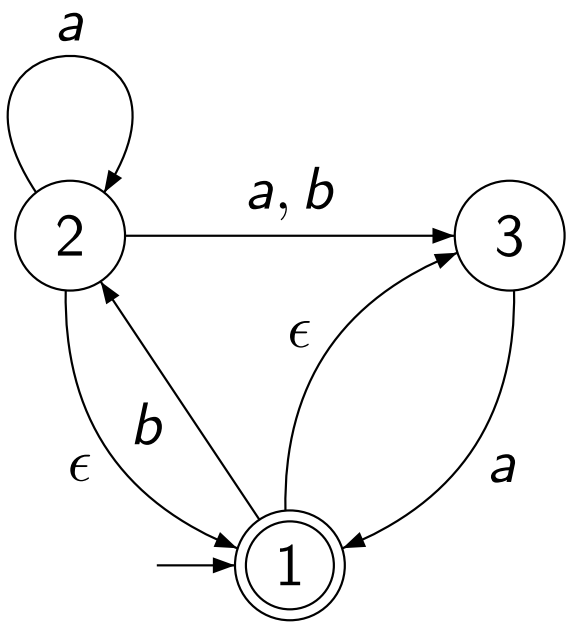
Determinizar AFnD con ϵ-transiciones



1. Empezar con una tabla con columnas para cada símbolo del alfabeto.

δ'	a	b

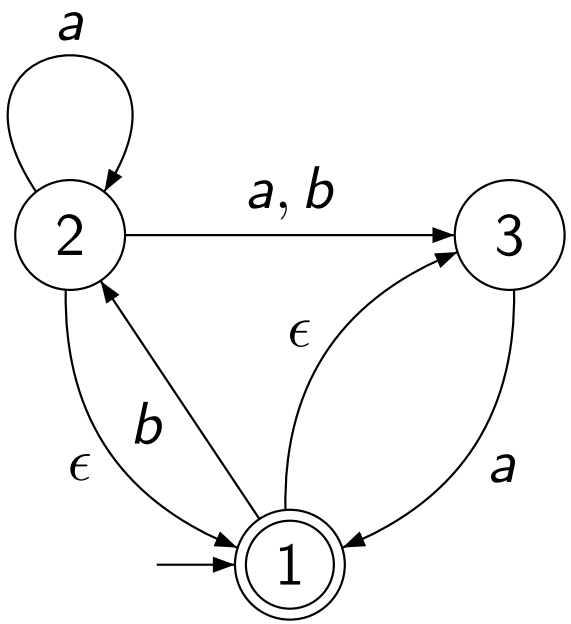
Determinizar AFnD con ϵ-transiciones



2. La primera fila de la tabla se marcará con los estados de la ϵ-$c(q_0)$

δ'	a	b
$\{1,3\}$		

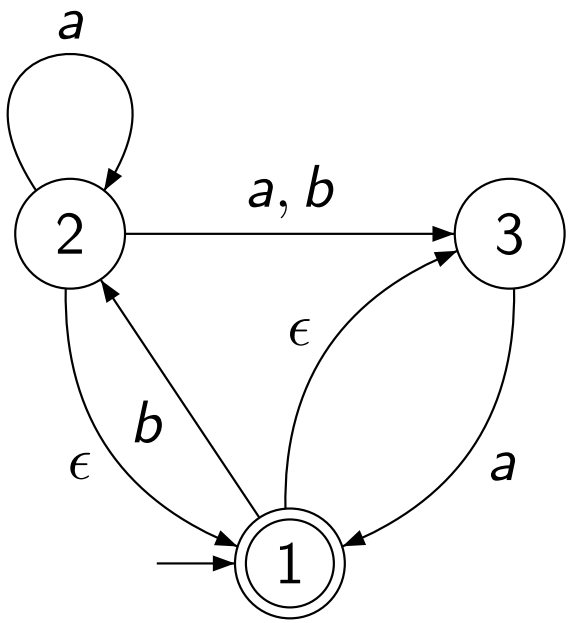
Determinizar AFnD con ϵ-transiciones



3. En la fila R,columna x de la tabla: se ponen todos los estados de ϵ-$c\big(\delta(R,x)\big)$

δ'	a	b
$\{1,3\}$	$\{1,3\}$	$\{1,2,3\}$

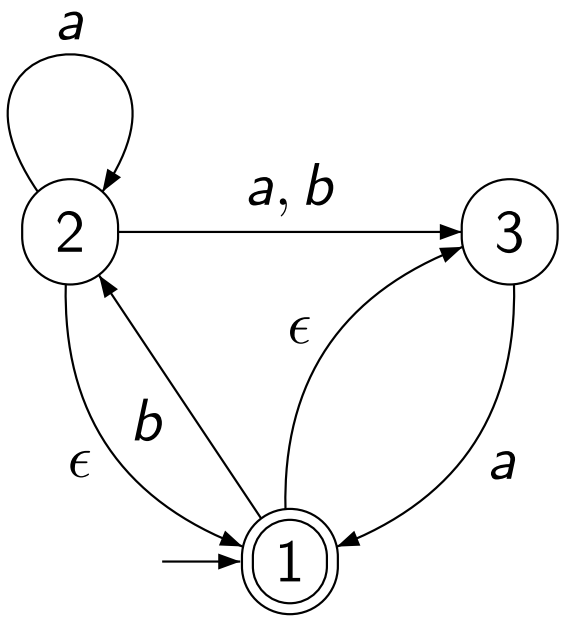
Determinizar AFnD con ϵ-transiciones



4. Copiar las casillas que han aparecido nuevas como nuevas filas (nuevos estados)

δ'	a	b
$\{1,3\}$	$\{1,3\}$	$\{1,2,3\}$
$\{1,2,3\}$		

Determinizar AFnD con ϵ-transiciones

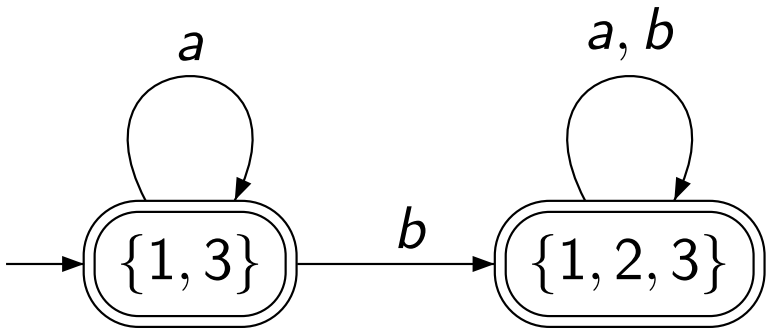


6. Repetir los pasos 3 y 4 hasta que no aparezcan nuevas filas.

δ'	a	b
$\{1,3\}$	$\{1,3\}$	$\{1,2,3\}$
$\{1,2,3\}$	$\{1,2,3\}$	$\{1,2,3\}$

Determinizar AFnD con ϵ-transiciones

6. Los estados finales del AFD equivalente son todas aquellas filas que contengan algún estado final.



Conclusiones

Teorema

Todo lenguaje reconocido por un AFnD es regular

Conclusiones

- ¿Qué hacer si queremos hallar un AFD que reconozca un lenguaje?
- Podemos encontrar un AFnD que reconozca el lenguaje y determinizar.
- ¿Cómo saber si dos autómatas reconocen el mismo lenguaje?
- Si los autómatas son AFnD, primero determinizar. Luego minimizar y ver si son el mismo autómata AFD.

Propiedades de clausura de los lenguajes regulares

- ¿La unión de lenguajes regulares es regular?
- ¿Y la concatenación, estrella de Kleene, complemento e intersección?
- Con la introducción de los AFnD probaremos fácilmente que son ciertas estas propiedades.

Propiedades de clausura de los lenguajes regulares

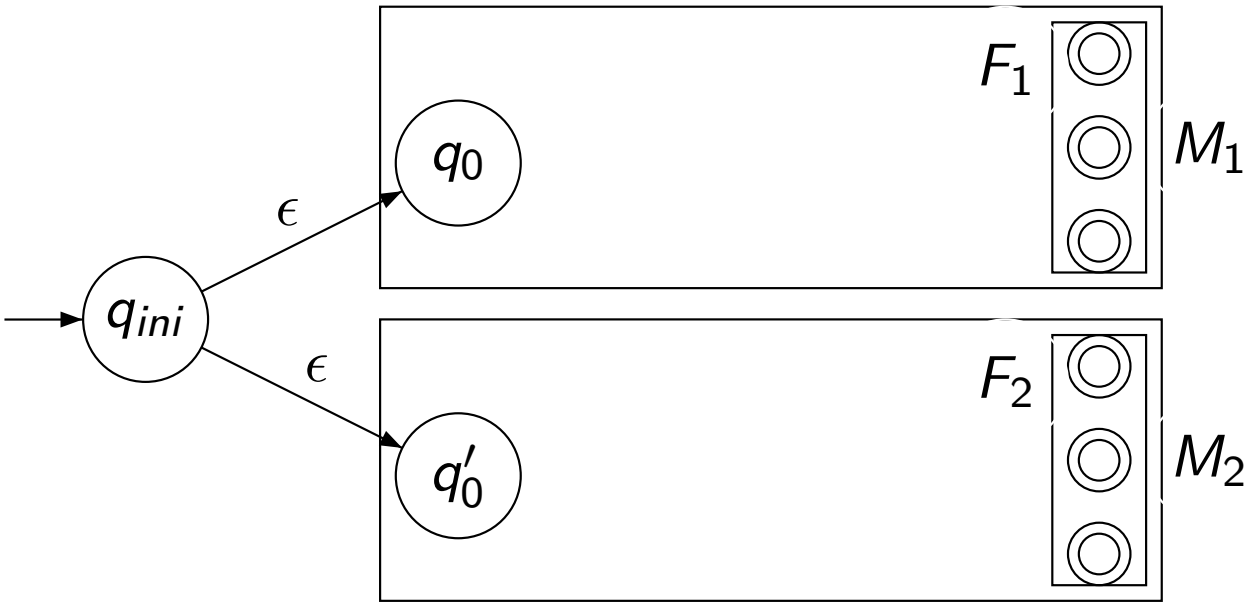
Teorema

Sean L_1 y L_2 lenguajes regulares. Entonces los siguientes lenguajes, también son regulares.

- $L_1 \cup L_2$
- $L_1 \cdot L_2$
- L_1^*
- $\overline{L}_1$
- $L_1 \cap L_2$

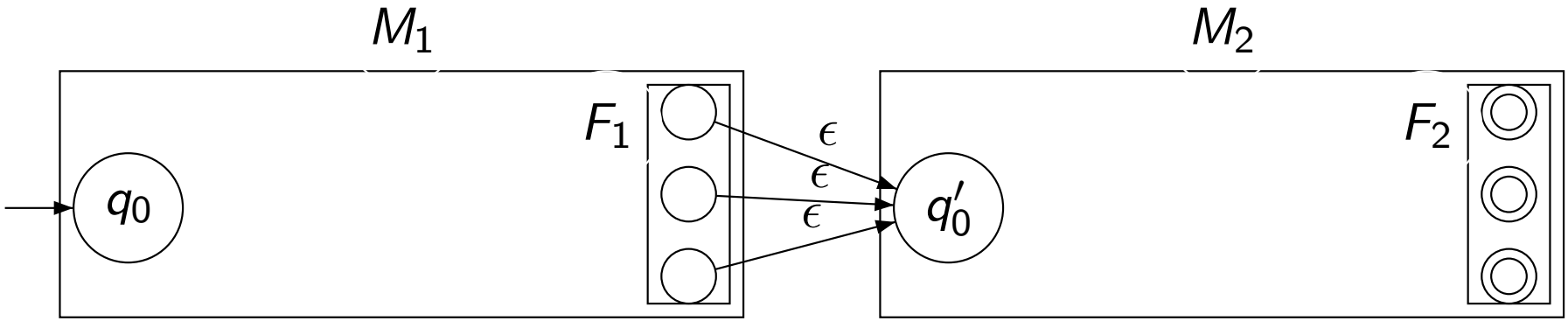
Unión de regulares

- L_1 reconocido por $M_1 = \{Q_1, \Sigma, q_0, \delta_1, F_1\}$
- L_2 reconocido por $M_2 = \{Q_2, \Sigma, q'_0, \delta_2, F_2\}$
- $L_1 \cup L_2$ reconocido por AFnD, $M = \{Q, \Sigma, q_{ini}, \delta, F\}$
  - $Q = Q_1 \cup Q_2 \cup \{q_{ini}\}$,
  - $F = F_1 \cup F_2$,
  - δ definida por



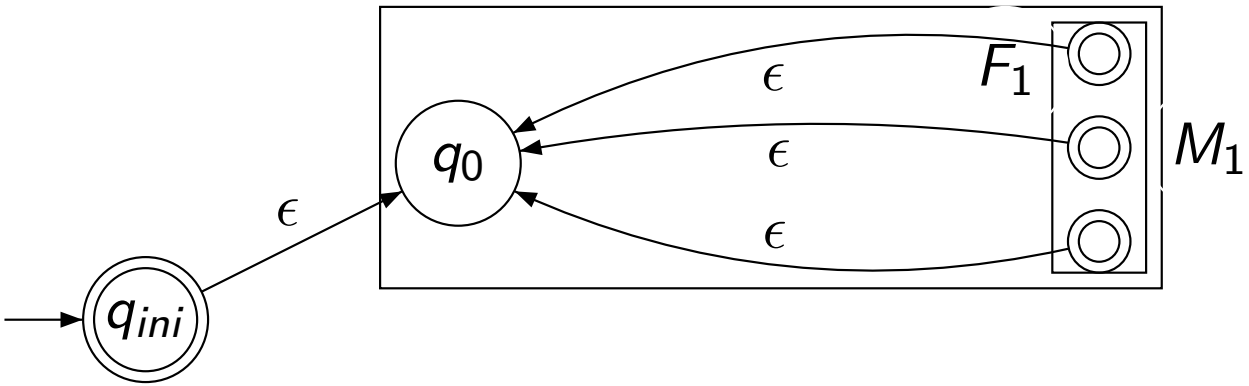
Concatenación de regulares

- L_1 reconocido por $M_1 = \{Q_1, \Sigma, q_0, \delta_1, F_1\}$
- L_2 reconocido por $M_2 = \{Q_2, \Sigma, q_0', \delta_2, F_2\}$
- $L_1 \cdot L_2$ reconocido por AFnD, $M = \{Q, \Sigma, q_0, \delta, F\}$
  - $Q = Q_1 \cup Q_2$,
  - $F = F_2$,
  - δ definida por



Estrella de Kleene

- L_1 reconocido por $M_1 = \{Q_1, \Sigma, q_0, \delta_1, F_1\}$
- L_1^* reconocido por AFnD, $M = \{Q, \Sigma, q_{ini}, \delta, F\}$
  - $Q = Q_1 \cup \{q_{ini}\}$,
  - $F = F_1 \cup \{q_{ini}\}$,
  - δ definida por



Complemento

- Cambiar estados finales por estados no finales, y viceversa.
- L_1 reconocido por $M_1 = \{Q_1, \Sigma, q_0, \delta_1, F_1\}$
- $\overline{L}_1$ reconocido por AFD, $M = \{Q, \Sigma, q_0, \delta, F\}$
  - $Q = Q_1$,
  - $F = Q - F_1$,
  - $\delta = \delta_1$

Intersección

- Notar que $L_1 \cap L_2 = \overline{\overline{L}_1 \cup \overline{L}_2}$
- Hemos probado que el complemento y uniones de lenguajes regulares es regular.
- Por tanto, la intersección de lenguajes regulares es regular.

Referencias

- Kelly: capítulo 2, secciones 2.5-2.7
- Sipser: capítulo 1, sección 1.2