

# Autómatas Finitos

Juan Mendivelso

Universidad Nacional de Colombia  
Facultad de Ciencias  
Departamento de Matemáticas

Presentación basada en las Notas de Clase del Profesor Rodrigo De Castro.

# Outline

## 1 Autómatas Finitos Deterministas

- Definiciones Básicas
- Complemento
- Producto Cartesiano
- Minimización

## 2 Autómatas Finitos No Deterministas

- Definición y Representación
- Equivalencia Computacional entre los AFD y los AFN

## 3 Autómatas con Transiciones $\lambda$

- Definición y Representación
- Equivalencia Computacional entre los AFN y los AFN- $\lambda$

## 4 Lenguajes Regulares & Autómatas Finitos

- Teorema de Kleene
- Propiedades de Clausura de los Lenguajes Regulares
- Teorema de Myhill-Nerode

# Outline

## 1 Autómatas Finitos Deterministas

- Definiciones Básicas
- Complemento
- Producto Cartesiano
- Minimización

## 2 Autómatas Finitos No Deterministas

- Definición y Representación
- Equivalencia Computacional entre los AFD y los AFN

## 3 Autómatas con Transiciones $\lambda$

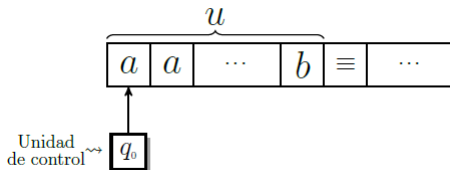
- Definición y Representación
- Equivalencia Computacional entre los AFN y los AFN- $\lambda$

## 4 Lenguajes Regulares & Autómatas Finitos

- Teorema de Kleene
- Propiedades de Clausura de los Lenguajes Regulares
- Teorema de Myhill-Nerode

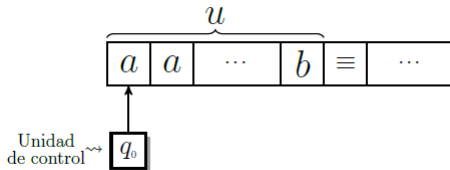
# Autómatas Finitos Deterministas (AFD)

- En 1936, Alan Turing introdujo las Máquinas de Turing.
- En los 1940's y 1950's se introdujeron máquinas de Turing con capacidad restringida.
- Los autómatas finitos son máquinas abstractas que tienen la capacidad de reconocer lenguajes.



# Autómatas Finitos Deterministas (AFD)

- Aceptan o rechazan cadenas de texto.
- Cinta semi-infinita dividida en celdas.
- Unidad de control, cabeza lectora, control finito o unidad de memoria.
- Estados del autómatas.
- Estado inicial.
- Estado final.



# Autómatas Finitos Deterministas (AFD)

Un Autómata Finito Determinista (AFD)  $M = (\Sigma, Q, q_0, F, \delta)$  tiene cinco componentes:

- 1 Alfabeto  $\Sigma$ .
- 2 Estados  $Q = \{q_0, q_1, \dots, q_n\}$ .
- 3 Estado inicial  $q_0 \in Q$ .
- 4 Estados finales o de aceptación  $F \subseteq Q$ . Se tiene que  $F \neq \emptyset$ .
- 5 Función de transición (o dinámica del autómata):  $\delta$ .

Un AFD acepta la cadena  $u$  si está en un estado de aceptación al leer la primera casilla vacía.

# Función de Transición (o Dinámica del Autómata)

- Lista de instrucciones de  $M = (\Sigma, Q, q_0, F, \delta)$  para procesar toda  $u \in \Sigma^*$ :

$$\begin{aligned}\delta : Q \times \Sigma &\rightarrow Q \\ (q, s) &\mapsto \delta(q, s)\end{aligned}$$

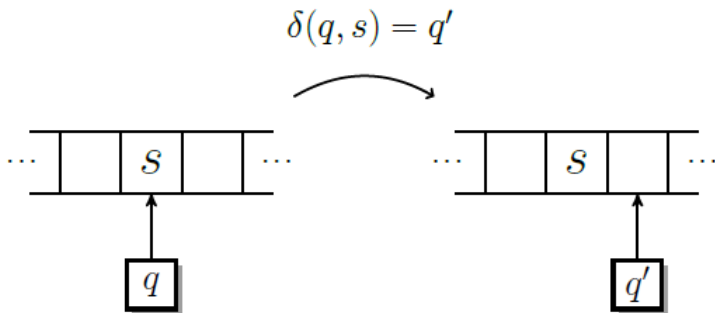
- La función  $\delta$  está definida para toda pareja  $(q, s)$ ,  $\forall q \in Q$  y  $\forall s \in \Sigma$ .

# Paso Computacional

- Sea  $M = (\Sigma, Q, q_0, F, \delta)$  un AFD y  $u \in \Sigma^*$ .
- La unidad de control apunta al primer caracter  $u$  inicialmente.
- Cada procesamiento  $\delta(q, s) = q'$ , para  $q, q' \in Q$  y  $s \in \Sigma$  es un **paso computacional**.
- Al final de cada paso computacional, la unidad de control se mueve una celda a la derecha.
- $M$  acepta la cadena  $u$  si está en un estado de aceptación al leer la primera casilla vacía.
- La cadena  $u$  se procesa de manera única.



# Paso Computacional



# Ejemplo de AFD

$$\Sigma = \{a, b\}.$$

$$Q = \{q_0, q_1, q_2\}.$$

$q_0$  : estado inicial.

$F = \{q_0, q_2\}$ , estados de aceptación.

Función de transición  $\delta$ :

$\delta$	$a$	$b$
$q_0$	$q_0$	$q_1$
$q_1$	$q_1$	$q_2$
$q_2$	$q_1$	$q_1$

$$\delta(q_0, a) = q_0$$

$$\delta(q_0, b) = q_1$$

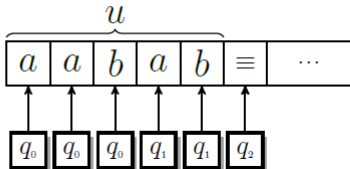
$$\delta(q_1, a) = q_1$$

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

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

$$\delta(q_2, b) = q_1.$$

1.  $u = aabab.$



# Ejemplo de AFD

$$\Sigma = \{a, b\}.$$

$$Q = \{q_0, q_1, q_2\}.$$

$q_0$  : estado inicial.

$F = \{q_0, q_2\}$ , estados de aceptación.

Función de transición  $\delta$ :

$\delta$	$a$	$b$
$q_0$	$q_0$	$q_1$
$q_1$	$q_1$	$q_2$
$q_2$	$q_1$	$q_1$

$$\delta(q_0, a) = q_0$$

$$\delta(q_0, b) = q_1$$

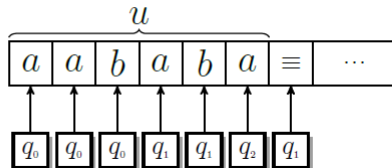
$$\delta(q_1, a) = q_1$$

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

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

$$\delta(q_2, b) = q_1.$$

2.  $v = aababa$ .



# Ejemplo de AFD

$$\Sigma = \{a, b\}.$$

$$Q = \{q_0, q_1, q_2\}.$$

$q_0$  : estado inicial.

$F = \{q_0, q_2\}$ , estados de aceptación.

Función de transición  $\delta$ :

$\delta$	$a$	$b$
$q_0$	$q_0$	$q_1$
$q_1$	$q_1$	$q_2$
$q_2$	$q_1$	$q_1$

$$\delta(q_0, a) = q_0$$

$$\delta(q_0, b) = q_1$$

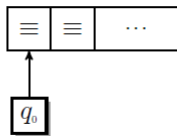
$$\delta(q_1, a) = q_1$$

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

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

$$\delta(q_2, b) = q_1.$$

3. Caso especial: la cadena  $\lambda$  es la cadena de entrada.



# Función de Transición Extendida

- Sea  $M = (\Sigma, Q, q_0, F, \delta)$  un AFD.
- La función de transición  $\delta, \delta : Q \times \Sigma \rightarrow Q$  se extiende a la **función de transición extendida**  $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$  definida así:

$$\begin{cases} \hat{\delta}(q, \lambda) = q & q \in Q \\ \hat{\delta}(q, a) = \delta(q, a) & q \in Q, a \in \Sigma \\ \hat{\delta}(q, ua) = \delta(\hat{\delta}(q, u), a) & q \in Q, a \in \Sigma, q \in \Sigma^* \end{cases}$$

- $\hat{\delta}(q, u)$ : Estado en el que queda la unidad de control después de procesar la cadena  $u \in \Sigma^*$  partiendo del estado  $q \in Q$ .
- $\hat{\delta}(q, u)$  se puede denotar como  $\delta(q, u)$  sin lugar a ambigüedad.

# Lenguaje Aceptado (Reconocido) por un AFD

- Sea  $M = (\Sigma, Q, q_0, F, \delta)$  un AFD.
- La unidad de control apunta al primer caracter  $u$  inicialmente.
- $M$  acepta o rechaza las cadenas en  $\Sigma^*$ .
- Se puede decir que  $M$  clasifica las cadenas de  $\Sigma^*$  en dos clases disyuntas: las aceptadas y las rechazadas.
- El conjunto de las cadenas aceptadas por  $M$  se denomina **lenguaje aceptado (reconocido)** por  $M$ :

$$\begin{aligned} L(M) &= \{u \in \Sigma^* : u \text{ es aceptada por } M\} \\ &= \{u \in \Sigma^* : M \text{ termina el procesamiento de } u \text{ en un } q \in F\} \\ &= \{u \in \Sigma^* : \hat{\delta}(q_0, u) \in F\}. \end{aligned}$$

# Estados Limbo

- Sea  $M = (\Sigma, Q, q_0, F, \delta)$  un AFD.
- Un estado  $q \in Q$  es **limbo** si no existe ninguna cadena  $u \in \Sigma^*$  tal que  $\hat{\delta}(q, u) \in F$ .
- En otras palabras, un estado  $q \in Q$  es **limbo** si una vez se llega a  $q$  ya no se puede llegar a un estado de aceptación.
- Se requieren para rechazar cadenas.
- Hacen parte del autómata y deben considerarse en la función de transición.




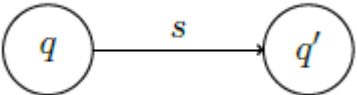
# Estados Accesibles

- Sea  $M = (\Sigma, Q, q_0, F, \delta)$  un AFD.
- Un estado  $q \in Q$  es **accesible** si existe una cadena de entrada  $u \in \Sigma^*$  tal que  $\hat{\delta}(q_0, u) = q$ .
- Los estados inaccesibles son inútiles ya que no serán accedidos por la unidad de control.
- Los estados inaccesibles se pueden eliminar sin alterar  $L(M)$ .
- Los estados inaccesibles son diferentes a los estados limbo, pues los primeros son inútiles mientras los últimos son necesarios para rechazar cadenas.



# Grafo de un AFD

Un AFD se puede representar por un grafo dirigido y etiquetado:

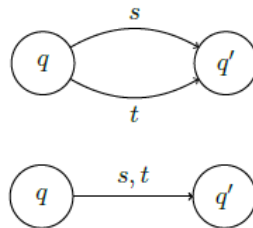
Estado $q_0 \in Q$	
Estado $q \in Q - F$	
Estado $q \in F$	
Transición $\delta(q, s) = q'$	

# Grafo de un AFD

Bucle  $\delta(q, s) = q$



Transiciones donde  $\delta(q, s) = \delta(q, t) = q'$



# Grafo de un AFD

- Sea  $M = (\Sigma, Q, q_0, F, \delta)$  un AFD.
- El grafo de  $M$  es útil para hacer seguimiento del procesamiento de una cadena  $u \in \Sigma^*$ .
- Si  $u$  es aceptada, existe una trayectoria desde  $q_0$  hasta un estado  $q \in F$  etiquetada por los caracteres de  $u$ .
- Como  $\delta$  está definida para toda pareja  $(q, s)$ ,  $\forall q \in Q$  y  $\forall s \in \Sigma$ , desde cada nodo del grafo salen  $|\Sigma|$  arcos.
- Un estado  $q \in Q$  es **limbo** si desde  $q$  no parten trayectorias que conduzcan a estados de aceptación.
- Los estados limbo hacen parte integrante del autómata. Sin embargo, en el grafo a veces se suprimen.

# Ejemplo de Grafo de un AFD

## Ejercicio

Dibuje el grafo para el siguiente autómata:

$$\Sigma = \{a, b\}.$$

$$Q = \{q_0, q_1, q_2\}.$$

$q_0$  : estado inicial.

$F = \{q_0, q_2\}$ , estados de aceptación.

Función de transición  $\delta$ :

$\delta$	$a$	$b$
$q_0$	$q_0$	$q_1$
$q_1$	$q_1$	$q_2$
$q_2$	$q_1$	$q_1$

$$\delta(q_0, a) = q_0$$

$$\delta(q_0, b) = q_1$$

$$\delta(q_1, a) = q_1$$

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

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

$$\delta(q_2, b) = q_1.$$

# Diseño de un AFD

- Se aborda el siguiente problema: Dado un lenguaje  $L$ , diseñar un AFD  $M$  que acepte  $L$ , i.e.  $L(M) = L$ .
- Se debe cumplir lo siguiente:
  - 1  $u \in L(M) \implies u \in L$ .
  - 2  $u \in L \implies u \in L(M)$ .

# Diseño

## Ejercicio

Dado el alfabeto  $\Sigma = \{a, b\}$  diseñe AFDs que acepten los siguientes lenguajes. Para cada uno muestre el grafo con y sin estados limbo.

- 1  $L = a^*$
- 2  $L = a^+$
- 3  $L = \{u \in \Sigma^* : u \text{ contiene exactamente dos aes}\}$
- 4  $L = \{u \in \Sigma^* : u \text{ tiene un número par de símbolos} \geq 0\}$
- 5  $L = \{u \in \Sigma^* : u \text{ tiene un número par de aes} \geq 0\}$
- 6  $L = (b \cup ab)^*$
- 7  $L = (a \cup b)^*$
- 8  $L = \emptyset$

# Outline

## 1 Autómatas Finitos Deterministas

- Definiciones Básicas
- **Complemento**
- Producto Cartesiano
- Minimización

## 2 Autómatas Finitos No Deterministas

- Definición y Representación
- Equivalencia Computacional entre los AFD y los AFN

## 3 Autómatas con Transiciones $\lambda$

- Definición y Representación
- Equivalencia Computacional entre los AFN y los AFN- $\lambda$

## 4 Lenguajes Regulares & Autómatas Finitos

- Teorema de Kleene
- Propiedades de Clausura de los Lenguajes Regulares
- Teorema de Myhill-Nerode

# Complemento

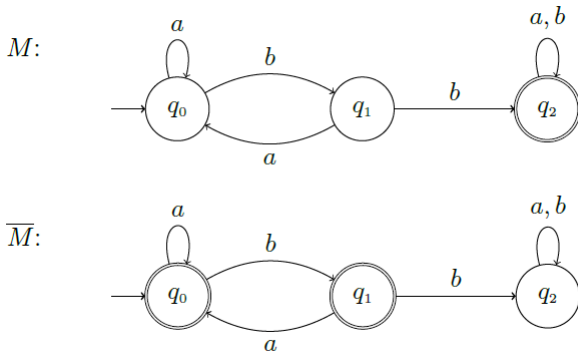
- Sea  $M = (\Sigma, Q, q_0, F, \delta)$  un AFD.
- El **complemento** de  $M$  es  $\bar{M} = (\Sigma, Q, q_0, \bar{F}, \delta)$  donde  $\bar{F} = Q - F$ .
- Se intercambian los estados de aceptación y no aceptación.
- Si  $L(M) = L$ ,  $L(\bar{M}) = \bar{L} = \Sigma^* - L$ .
- Son útiles cuando el lenguaje está definido por medio de una condición negativa.



# Complemento

## Ejercicio

- Sea  $\Sigma = \{a, b\}$ . Encontrar un AFD que acepte el lenguaje de todas las cadenas que no tienen dos  $b$ es consecutivas.



# Complemento

## Ejercicio

Sea  $\Sigma = \{a, b, c\}$ . Encontrar un AFD que acepte el lenguaje de todas las cadenas que ...

- No contienen la subcadena  $bc$ .
- No terminan en  $bb$ .

# Outline

## 1 Autómatas Finitos Deterministas

- Definiciones Básicas
- Complemento
- **Producto Cartesiano**
- Minimización

## 2 Autómatas Finitos No Deterministas

- Definición y Representación
- Equivalencia Computacional entre los AFD y los AFN

## 3 Autómatas con Transiciones $\lambda$

- Definición y Representación
- Equivalencia Computacional entre los AFN y los AFN- $\lambda$

## 4 Lenguajes Regulares & Autómatas Finitos

- Teorema de Kleene
- Propiedades de Clausura de los Lenguajes Regulares
- Teorema de Myhill-Nerode

# Producto Cartesiano

- Sean  $M_1 = (\Sigma, Q_1, q_1, F_1, \delta_1)$  y  $M_2 = (\Sigma, Q_2, q_2, F_2, \delta_2)$  dos AFDs definidos sobre el alfabeto  $\Sigma$ , tales que  $L(M_1) = L_1$  y  $L(M_2) = L_2$ .
- El **producto cartesiano** de  $M_1$  y  $M_2$  es

$$M_1 \times M_2 = (\Sigma, Q_1 \times Q_2, (q_1, q_2), F, \delta)$$

donde la función de transición  $\delta$  está dada por

$\delta : (Q_1 \times Q_2) \times \Sigma \rightarrow Q_1 \times Q_2$ , definida por

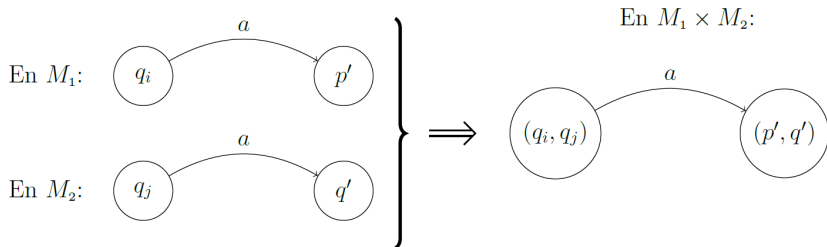
$\delta((q_i, q_j), a) = (\delta_1(q_i, a), \delta_2(q_j, a))$ , y el conjunto de estados de aceptación se escoge a conveniencia según el lenguaje a aceptar:

- 1 Para  $L(M_1 \times M_2) = L_1 \cup L_2$ , escoger  
 $F = \{(q_i, q_j) : q_i \in F_1 \vee q_j \in F_2\} = (F_1 \times Q_2) \cup (Q_1 \times F_2)$ .
- 2 Para  $L(M_1 \times M_2) = L_1 \cap L_2$ , escoger  
 $F = \{(q_i, q_j) : q_i \in F_1 \wedge q_j \in F_2\} = F_1 \times F_2$ .
- 3 Para  $L(M_1 \times M_2) = L_1 - L_2$ , escoger  
 $F = \{(q_i, q_j) : q_i \in F_1 \wedge q_j \notin F_2\} = F_1 \times (Q_2 - F_2)$ .

# Función de Transición en el Producto Cartesiano

- Sean  $M_1 = (\Sigma, Q_1, q_1, F_1, \delta_1)$  y  $M_2 = (\Sigma, Q_2, q_2, F_2, \delta_2)$  dos AFDs definidos sobre el alfabeto  $\Sigma$ , tales que  $L(M_1) = L_1$  y  $L(M_2) = L_2$ .
- Sea  $M_1 \times M_2 = (\Sigma, Q_1 \times Q_2, (q_1, q_2), F, \delta)$  su producto cartesiano.
- La función de transición  $\delta$  está dada por

$$\begin{aligned}\delta : (Q_1 \times Q_2) \times \Sigma &\rightarrow Q_1 \times Q_2 \\ \delta((q_i, q_j), a) &= (\delta_1(q_i, a), \delta_2(q_j, a))\end{aligned}$$



# Función de Transición Extendida en el Producto Cartesiano

## Lema

- Sean  $M_1 = (\Sigma, Q_1, q_1, F_1, \delta_1)$  y  $M_2 = (\Sigma, Q_2, q_2, F_2, \delta_2)$  dos AFDs definidos sobre el alfabeto  $\Sigma$ , tales que  $L(M_1) = L_1$  y  $L(M_2) = L_2$ .
- Sea  $M_1 \times M_2 = (\Sigma, Q_1 \times Q_2, (q_1, q_2), F, \delta)$  su producto cartesiano donde  $\delta((q_i, q_j), a) = (\delta_1(q_i, a), \delta_2(q_j, a))$ .
- La función de transición  $\delta$  se puede extender a cadenas arbitrarias:

$$\hat{\delta}((q_i, q_j), u) = (\hat{\delta}_1(q_i, u), \hat{\delta}_2(q_j, u)), \forall u \in \Sigma^*, q_i \in Q_1, q_j \in Q_2.$$

- Para la demostración (inductiva), recordar que para un AFD  $M = (\Sigma, Q, q_0, F, \delta)$ , la función de transición extendida se define:
  - 1  $\hat{\delta}(q, \lambda) = q$  para  $q \in Q$ .
  - 2  $\hat{\delta}(q, a) = \delta(q, a)$  para  $q \in Q$  y  $a \in \Sigma$ .
  - 3  $\hat{\delta}(q, ua) = \delta(\hat{\delta}(q, u), a)$  para  $q \in Q$ ,  $u \in \Sigma^*$  y  $a \in \Sigma$ .

# Estados de Aceptación del Producto Cartesiano

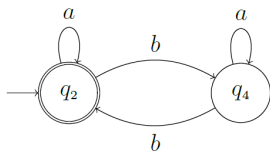
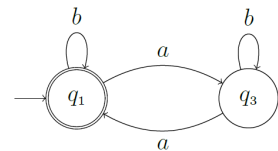
## Teorema

- Sean  $M_1 = (\Sigma, Q_1, q_1, F_1, \delta_1)$  y  $M_2 = (\Sigma, Q_2, q_2, F_2, \delta_2)$  dos AFDs definidos sobre el alfabeto  $\Sigma$ , tales que  $L(M_1) = L_1$  y  $L(M_2) = L_2$ .
- Sea  $M_1 \times M_2 = (\Sigma, Q_1 \times Q_2, (q_1, q_2), F, \delta)$  el producto cartesiano de estos.
- Entonces,
  - 1  $F = \{(q_i, q_j) : q_i \in F_1 \vee q_j \in F_2\} = (F_1 \times Q_2) \cup (Q_1 \times F_2)$ , si y solo si  $L(M_1 \times M_2) = L_1 \cup L_2$ .
  - 2  $F = \{(q_i, q_j) : q_i \in F_1 \wedge q_j \in F_2\} = F_1 \times F_2$ , si y solo si  $L(M_1 \times M_2) = L_1 \cap L_2$ .
  - 3  $F = \{(q_i, q_j) : q_i \in F_1 \wedge q_j \notin F_2\} = F_1 \times (Q_2 - F_2)$ , si y solo si  $L(M_1 \times M_2) = L_1 - L_2$ .

# Ejemplos de Producto Cartesiano de AFDs

## Ejercicio

Construir un AFD que acepte el lenguaje de todas las cadenas sobre  $\Sigma = \{a, b\}$  que tienen un número par de  $a$ s y un número par de  $b$ s.



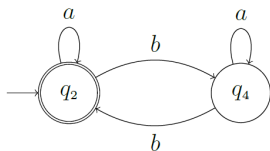
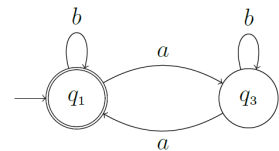
$$\left\{ \begin{array}{l} \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)) = (q_3, q_2), \\ \delta((q_1, q_2), b) = (\delta_1(q_1, b), \delta_2(q_2, b)) = (q_1, q_4), \\ \delta((q_1, q_4), a) = (\delta_1(q_1, a), \delta_2(q_4, a)) = (q_3, q_4), \\ \delta((q_1, q_4), b) = (\delta_1(q_1, b), \delta_2(q_4, b)) = (q_1, q_2), \\ \delta((q_3, q_2), a) = (\delta_1(q_3, a), \delta_2(q_2, a)) = (q_1, q_2), \\ \delta((q_3, q_2), b) = (\delta_1(q_3, b), \delta_2(q_2, b)) = (q_3, q_4), \\ \delta((q_3, q_4), a) = (\delta_1(q_3, a), \delta_2(q_4, a)) = (q_1, q_4), \\ \delta((q_3, q_4), b) = (\delta_1(q_3, b), \delta_2(q_4, b)) = (q_3, q_2). \end{array} \right.$$



# Ejemplos de Producto Cartesiano de AFDs

## Ejercicio

Construir un AFD que acepte el lenguaje de todas las cadenas sobre  $\Sigma = \{a, b\}$  que tienen un número par de  $a$ s y un número par de  $b$ s.

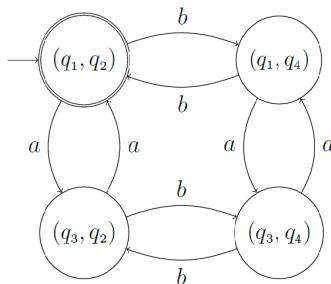
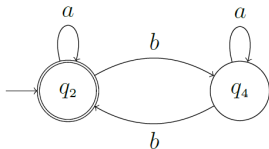
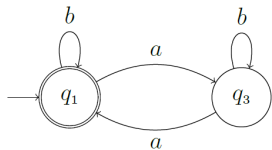


$$\left\{ \begin{array}{l} \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)) = (q_3, q_2), \\ \delta((q_1, q_2), b) = (\delta_1(q_1, b), \delta_2(q_2, b)) = (q_1, q_4), \\ \delta((q_1, q_4), a) = (\delta_1(q_1, a), \delta_2(q_4, a)) = (q_3, q_4), \\ \delta((q_1, q_4), b) = (\delta_1(q_1, b), \delta_2(q_4, b)) = (q_1, q_2), \\ \delta((q_3, q_2), a) = (\delta_1(q_3, a), \delta_2(q_2, a)) = (q_1, q_2), \\ \delta((q_3, q_2), b) = (\delta_1(q_3, b), \delta_2(q_2, b)) = (q_3, q_4), \\ \delta((q_3, q_4), a) = (\delta_1(q_3, a), \delta_2(q_4, a)) = (q_1, q_4), \\ \delta((q_3, q_4), b) = (\delta_1(q_3, b), \delta_2(q_4, b)) = (q_3, q_2). \end{array} \right.$$

# Ejemplos de Producto Cartesiano de AFDs

## Ejercicio

Construir un AFD que acepte el lenguaje de todas las cadenas sobre  $\Sigma = \{a, b\}$  que tienen un número par de  $a$ s y un número par de  $b$ s.



# Ejemplos de Producto Cartesiano de AFDs

## Ejercicio

Construir un AFD que acepte el lenguaje de todas las cadenas sobre  $\Sigma = \{a, b\}$  que ...

- 1 tienen longitud impar y que no contienen dos *bes* consecutivas.
- 2 tienen longitud impar o que no contienen dos *bes* consecutivas.

¿Cuándo se pueden omitir los estados limbo?

# Outline

## 1 Autómatas Finitos Deterministas

- Definiciones Básicas
- Complemento
- Producto Cartesiano
- Minimización

## 2 Autómatas Finitos No Deterministas

- Definición y Representación
- Equivalencia Computacional entre los AFD y los AFN

## 3 Autómatas con Transiciones $\lambda$

- Definición y Representación
- Equivalencia Computacional entre los AFN y los AFN- $\lambda$

## 4 Lenguajes Regulares & Autómatas Finitos

- Teorema de Kleene
- Propiedades de Clausura de los Lenguajes Regulares
- Teorema de Myhill-Nerode

# Minimización de AFDs

## Problema

Dado un AFD  $M$ , encontrar un AFD  $M'$  con el mínimo número de estados posible tal que  $L(M') = L(M)$ .

# Estados Equivalentes

- Sea  $M = (\Sigma, Q, q_0, F, \delta)$  un AFD y  $p, q \in Q$ . Se dice que  $p$  y  $q$  son **equivalentes**, notado  $p \approx q$ , si y solo si

$$(\forall u \in \Sigma^*)[\hat{\delta}(p, u) \in F \iff \hat{\delta}(q, u) \in F].$$

- La relación  $\approx$  cumple con las siguientes propiedades:
  - Reflexividad:  $p \approx p$ .
  - Simetría: Si  $p \approx q$ , entonces  $q \approx p$ .
  - Transitividad: Si  $p \approx q$  y  $q \approx r$ , entonces  $p \approx r$ .
- Entonces,  $\approx$  es una relación de equivalencia sobre  $Q$ .
- La **clase de equivalencia** de  $p \in Q$  es:

$$[p] := \{q \in Q : p \approx q\}.$$

# Autómata Cociente

- Sea  $M = (\Sigma, Q, q_0, F, \delta)$  un AFD.
- El autómata cociente de  $M$  es  $M' = (\Sigma, Q', q'_0, F', \delta')$  donde
  - $Q' = \{[p] : p \in Q\}$
  - $q'_0 = [q_0]$
  - $F' = \{[p] : p \in F\}$
  - $\delta'([p], a) = [\delta(p, a)]$  para todo  $a \in \Sigma$  y para todo  $p \in Q$ .
- Se requiere verificar que tanto  $F'$  como  $\delta'$  están bien definidos: no dependen del representante escogido de cada clase de equivalencia.

# Validez de $F'$ y $\delta'$

## Proposición

- ①  $\delta'$  está bien definida: si  $[p] = [q]$ , entonces  $\delta(p, a) \approx \delta(q, a)$ .
- ②  $F'$  está bien definido: si  $q \in F$  y  $p \approx q$ , entonces  $p \in F$ .
- ③  $p \in F \iff [p] \in F'$ .
- ④  $\hat{\delta}'([p], u) = [\hat{\delta}(p, u)]$  para toda cadena  $u \in \Sigma^*$ .



# Algoritmo de Minimización

## Algoritmo por llenado de tabla para determinar la equivalencia de estados en un AFD

### ENTRADA:

AFD  $M = (\Sigma, Q, q_0, F, \delta)$  completo (incluyendo estados limbo) cuyos estados son todos accesibles y tabla triangular que muestra todos los pares  $\{p, q\}$  de estados  $p, q \in Q$ .

### INICIALIZAR:

$i := 1$ . Se marca con  $\times$  la casilla  $\{p, q\}$  si  $p \in F$  y  $q \notin F$  (o viceversa).

### REPETIR:

$i := i + 1$ . Para cada casilla no marcada  $\{p, q\}$  y cada  $a \in \Sigma$ , hallar  $\{\delta(p, a), \delta(q, a)\}$ . Si para algún  $a \in \Sigma$ , la casilla  $\{\delta(p, a), \delta(q, a)\}$  ha sido marcada previamente, entonces se marca la casilla  $\{p, q\}$  con  $\times$ .

### HASTA:

No se puedan marcar más casillas en la tabla.

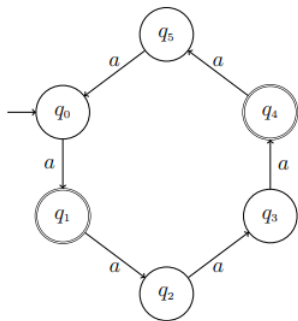
### SALIDA:

Si la casilla  $\{p, q\}$  está marcada con  $\times$ , entonces  $p \not\approx q$ . Si la casilla  $\{p, q\}$  no está marcada, entonces  $p \approx q$ .

# Algoritmo de Minimización

## Ejercicio

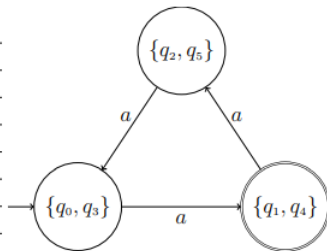
Minimizar el siguiente AFD, donde  $\Sigma = \{a\}$ .



$q_0$	$q_1$	$q_2$	$q_3$	$q_4$	$q_5$
×					
	×				
	×				
×		×	×		
	×			×	

$\{p, q\}$	$\{\delta(p, a), \delta(q, a)\}$
$\{q_0, q_2\}$	$\{q_1, q_3\}$ ×
$\{q_0, q_3\}$	$\{q_1, q_4\}$
$\{q_0, q_5\}$	$\{q_1, q_0\}$ ×
$\{q_1, q_4\}$	$\{q_2, q_5\}$
$\{q_2, q_3\}$	$\{q_3, q_4\}$ ×
$\{q_2, q_5\}$	$\{q_3, q_0\}$
$\{q_3, q_5\}$	$\{q_4, q_0\}$ ×

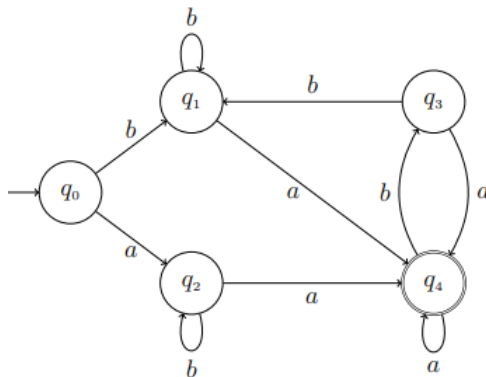
$q_0$	$q_1$	$q_2$	$q_3$	$q_4$	$q_5$
×					
×	×				
	×	×			
×		×	×		
×	×		×	×	



# Algoritmo de Minimización

## Ejercicio

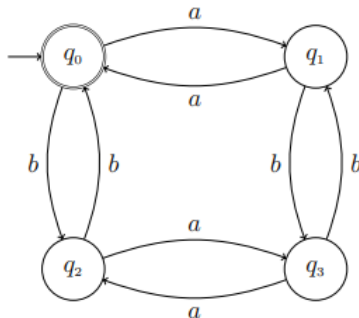
Minimizar el siguiente AFD, donde  $\Sigma = \{a, b\}$ .



# Algoritmo de Minimización

## Ejercicio

Mostrar que el siguiente AFD, donde  $\Sigma = \{a, b\}$ , no se puede simplificar.



# Outline

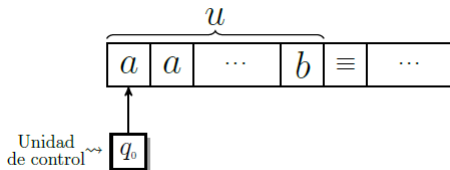
- 1 **Autómatas Finitos Deterministas**
  - Definiciones Básicas
  - Complemento
  - Producto Cartesiano
  - Minimización
- 2 **Autómatas Finitos No Deterministas**
  - Definición y Representación
  - Equivalencia Computacional entre los AFD y los AFN
- 3 **Autómatas con Transiciones  $\lambda$** 
  - Definición y Representación
  - Equivalencia Computacional entre los AFN y los AFN- $\lambda$
- 4 **Lenguajes Regulares & Autómatas Finitos**
  - Teorema de Kleene
  - Propiedades de Clausura de los Lenguajes Regulares
  - Teorema de Myhill-Nerode

# Outline

- 1 **Autómatas Finitos Deterministas**
  - Definiciones Básicas
  - Complemento
  - Producto Cartesiano
  - Minimización
- 2 **Autómatas Finitos No Deterministas**
  - Definición y Representación
  - Equivalencia Computacional entre los AFD y los AFN
- 3 **Autómatas con Transiciones  $\lambda$** 
  - Definición y Representación
  - Equivalencia Computacional entre los AFN y los AFN- $\lambda$
- 4 **Lenguajes Regulares & Autómatas Finitos**
  - Teorema de Kleene
  - Propiedades de Clausura de los Lenguajes Regulares
  - Teorema de Myhill-Nerode

# Autómatas Finitos No Deterministas (AFN)

- Aceptan o rechazan cadenas de texto.
- Cinta semi-infinita dividida en celdas.
- Unidad de control, cabeza lectora, control finito o unidad de memoria.
- Estados del autómatas, incluyendo inicial y final(es).
- Es no determinista porque una cadena se puede procesar de varias formas y puede haber procesamientos abortados.
- La función de transición para determinado estado y símbolo puede conducir a varios estados o a ningún estado.



# Autómatas Finitos No Deterministas (AFN)

Un Autómata Finito No Determinista (AFN)  $M = (\Sigma, Q, q_0, F, \Delta)$  tiene cinco componentes:

- 1 Alfabeto  $\Sigma$ .
- 2 Estados  $Q = \{q_0, q_1, \dots, q_n\}$ .
- 3 Estado inicial  $q_0 \in Q$ .
- 4 Estados finales o de aceptación  $F \subseteq Q$ . Se tiene que  $F \neq \emptyset$ .
- 5 Función de transición (o dinámica del autómata):  $\Delta$ .

Un AFN acepta la cadena  $u$  si existe algún procesamiento completo de  $u$  que termine en un estado de aceptación.



# Función de Transición (o Dinámica del Autómata)

- Lista de instrucciones de  $M = (\Sigma, Q, q_0, F, \delta)$  para procesar toda  $u \in \Sigma^*$ :

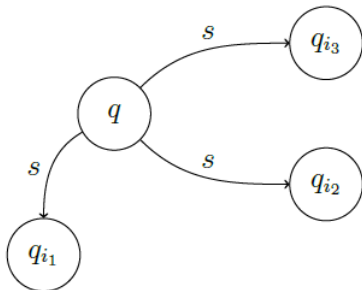
$$\Delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$$

$$(q, s) \mapsto \Delta(q, s) = \{q_{i_1}, q_{i_2}, \dots, q_{i_k}\}$$

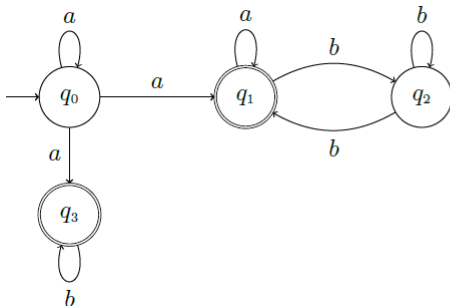
- Esto quiere decir que estando en el estado  $q$ , en presencia del símbolo  $s$ , la unidad de control puede pasar aleatoriamente a uno cualquiera de los estados  $q_{i_1}, q_{i_2}, \dots, q_{i_k}$ , después de lo cual se desplaza a la derecha.
- Puede suceder que  $\Delta(q, s) = \emptyset$ : si al procesar la cadena  $u$ , la cabeza lectora de  $M$  ingresa al estado  $q$ , leyendo sobre la cinta el símbolo  $s$ , el procesamiento se aborta.

# Función de Transición (o Dinámica del Autómata)

- Sea  $M = (\Sigma, Q, q_0, F, \Delta)$  un AFN con función de transición  $\Delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ , donde  $(q, s) \mapsto \Delta(q, s) = \{q_{i_1}, q_{i_2}, \dots, q_{i_k}\}$ .
- La representación del AFN como digrafo es similar a la de AFD.
- Si  $\Delta(q, s) = \{q_{i_1}, q_{i_2}, q_{i_3}\}$ , esto se puede representar en el autómata de la siguiente manera.

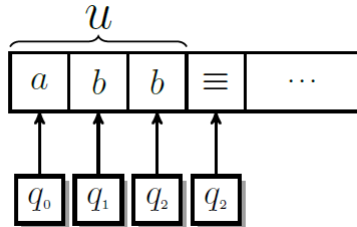


# Ejemplo de AFN

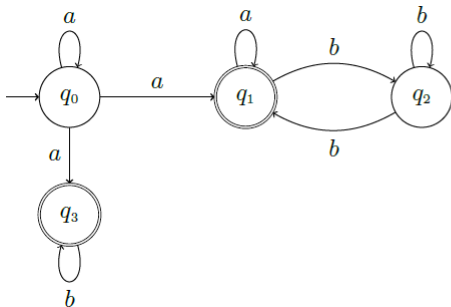


$\Delta$	$a$	$b$
$q_0$	$\{q_0, q_1, q_3\}$	$\emptyset$
$q_1$	$\{q_1\}$	$\{q_2\}$
$q_2$	$\emptyset$	$\{q_1, q_2\}$
$q_3$	$\emptyset$	$\{q_3\}$

Procesamiento de rechazo:

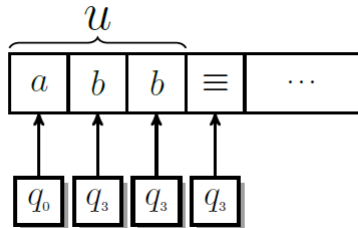


# Ejemplo de AFN

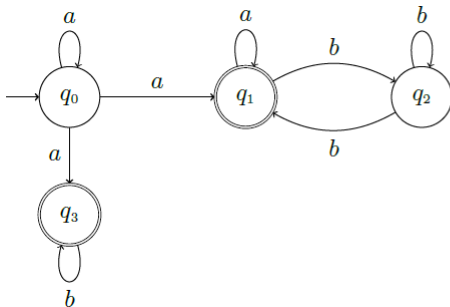


$\Delta$	$a$	$b$
$q_0$	$\{q_0, q_1, q_3\}$	$\emptyset$
$q_1$	$\{q_1\}$	$\{q_2\}$
$q_2$	$\emptyset$	$\{q_1, q_2\}$
$q_3$	$\emptyset$	$\{q_3\}$

Procesamiento de aceptación:

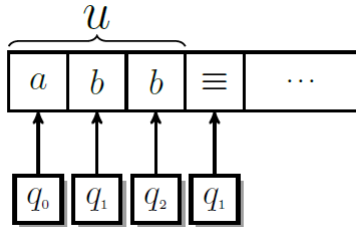


# Ejemplo de AFN

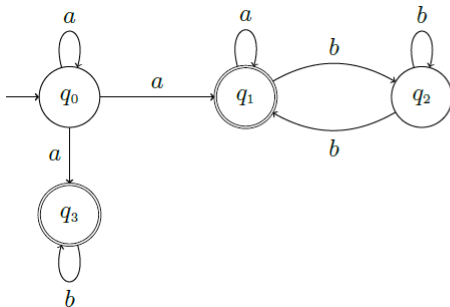


$\Delta$	$a$	$b$
$q_0$	$\{q_0, q_1, q_3\}$	$\emptyset$
$q_1$	$\{q_1\}$	$\{q_2\}$
$q_2$	$\emptyset$	$\{q_1, q_2\}$
$q_3$	$\emptyset$	$\{q_3\}$

Otro procesamiento de aceptación:

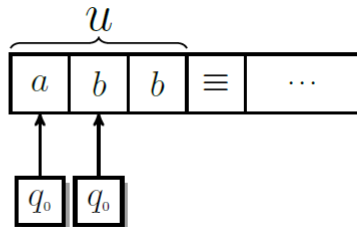


# Ejemplo de AFN

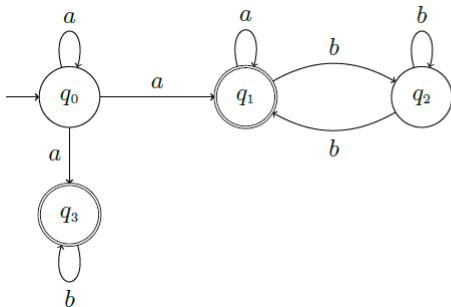


$\Delta$	$a$	$b$
$q_0$	$\{q_0, q_1, q_3\}$	$\emptyset$
$q_1$	$\{q_1\}$	$\{q_2\}$
$q_2$	$\emptyset$	$\{q_1, q_2\}$
$q_3$	$\emptyset$	$\{q_3\}$

Procesamiento abortado:

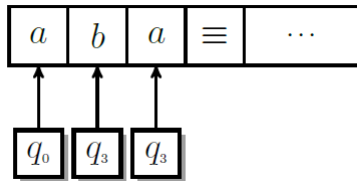


# Ejemplo de AFN



$\Delta$	$a$	$b$
$q_0$	$\{q_0, q_1, q_3\}$	$\emptyset$
$q_1$	$\{q_1\}$	$\{q_2\}$
$q_2$	$\emptyset$	$\{q_1, q_2\}$
$q_3$	$\emptyset$	$\{q_3\}$

Procesamiento abortado:



Las cadenas que empiezan por  $b$  no pueden ser aceptadas. Tampoco  $aba$ .

# Función de Transición para Conjuntos de Estados

- Sea  $M = (\Sigma, Q, q_0, F, \Delta)$  un AFN con función de transición  $\Delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ , donde  $(q, s) \mapsto \Delta(q, s) = \{q_{i_1}, q_{i_2}, \dots, q_{i_k}\}$ .
- La función de transición  $\Delta$  se extiende a la **función de transición para conjuntos de estados**  $\Delta : \mathcal{P}(Q) \times \Sigma \rightarrow \mathcal{P}(Q)$  definida así:  
Para  $a \in \Sigma$  y  $S \subseteq Q$ ,

$$\Delta(S, a) := \bigcup_{q \in S} \Delta(q, a).$$

- $\Delta(S, a)$  es el conjunto de estados a los que se puede llegar al procesar  $a$  partiendo de alguno de los estados en  $S$ .
- $\Delta(S, a) = \emptyset$  si  $\Delta(q, a) = \emptyset$  para todo  $q \in S$ .



# Función de Transición Extendida

- Sea  $M = (\Sigma, Q, q_0, F, \Delta)$  un AFN con función de transición  $\Delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ , donde  $(q, s) \mapsto \Delta(q, s) = \{q_{i_1}, q_{i_2}, \dots, q_{i_k}\}$ .
- La función de transición  $\Delta$  se extiende a conjuntos de estados definida así: Para  $a \in \Sigma$  y  $S \subseteq Q$ ,  $\Delta(S, a) := \bigcup_{q \in S} \Delta(q, a)$ .
- La función de transición  $\Delta$  se extiende a la **función de transición extendida**  $\hat{\Delta} : Q \times \Sigma^* \rightarrow Q$  definida así:

$$\begin{cases} \hat{\Delta}(q, \lambda) = q & q \in Q \\ \hat{\Delta}(q, a) = \Delta(q, a) & q \in Q, a \in \Sigma \\ \hat{\Delta}(q, ua) = \Delta(\hat{\Delta}(q, u), a) & q \in Q, a \in \Sigma, q \in \Sigma^*. \end{cases}$$

- $\hat{\delta}(q, u)$ : Estados a los que puede llegar la unidad de control con *procesamientos completos* de la cadena  $u \in \Sigma^*$  partiendo de  $q \in Q$ .
- $\hat{\Delta}(q, u)$  se puede denotar como  $\Delta(q, u)$  sin lugar a ambigüedad.

# Lenguaje Aceptado (Reconocido) por un AFN

- Sea  $M = (\Sigma, Q, q_0, F, \Delta)$  un AFN.
- La unidad de control apunta al primer caracter  $u$  inicialmente.
- $M$  acepta o rechaza las cadenas en  $\Sigma^*$ , i.e.  $M$  clasifica las cadenas de  $\Sigma^*$  en dos clases disjuntas: las aceptadas y las rechazadas.
- El conjunto de las cadenas aceptadas por  $M$  se denomina **lenguaje aceptado (reconocido)** por  $M$ :

$$\begin{aligned} L(M) &= \{u \in \Sigma^* : u \text{ es aceptada por } M\} \\ &= \{u \in \Sigma^* : M \text{ tiene al menos un procesamiento completo de} \\ &\quad u \text{ que parte de } q_0 \text{ y termina en } q \in F\} \\ &= \{u \in \Sigma^* : \hat{\Delta}(q_0, u) \text{ contiene alg\'un estado } q \in F\} \\ &= \{u \in \Sigma^* : \hat{\Delta}(q_0, u) \cap F \neq \emptyset\}. \end{aligned}$$

# Diseño de un AFN

- Se aborda el siguiente problema: Dado un lenguaje  $L$ , diseñar un AFN  $M$  que acepte  $L$ , i.e.  $L(M) = L$ .
- Se debe cumplir lo siguiente:
  - 1  $u \in L(M) \implies u \in L$ .
  - 2  $u \in L \implies u \in L(M)$ .

# Diseño

## Ejercicio

Dado el alfabeto  $\Sigma = \{a, b\}$  diseñe AFNs que acepten los siguientes lenguajes. Dibuje AFDs completos que acepten los mismos lenguajes.

- 1  $L = ab^* \cup a^+$
- 2  $L = (ab \cup aba)^*$
- 3  $L = a(a \cup ab)^*$
- 4  $L = a^+ b^* a$

# Outline

- 1 Autómatas Finitos Deterministas
  - Definiciones Básicas
  - Complemento
  - Producto Cartesiano
  - Minimización
- 2 Autómatas Finitos No Deterministas
  - Definición y Representación
  - Equivalencia Computacional entre los AFD y los AFN
- 3 Autómatas con Transiciones  $\lambda$ 
  - Definición y Representación
  - Equivalencia Computacional entre los AFN y los AFN- $\lambda$
- 4 Lenguajes Regulares & Autómatas Finitos
  - Teorema de Kleene
  - Propiedades de Clausura de los Lenguajes Regulares
  - Teorema de Myhill-Nerode

# Equivalencia Computacional entre los AFD y los AFN

- Los modelos AFD Y AFN son computacionalmente equivalentes: aceptan los mismos lenguajes.
- Un AFD  $M = (\Sigma, Q, q_0, F, \delta)$  puede ser considerado como un AFN  $M' = (\Sigma, Q, q_0, F, \Delta)$  definiendo  $\Delta(q, a) = \{\delta(q, a)\}$  para cada  $q \in Q$  y cada  $a \in \Sigma$ .
- La afirmación recíproca también es correcta.

# Equivalencia Computacional entre los AFD y los AFN

## Teorema

Dado un AFN  $M = (\Sigma, Q, q_0, F, \Delta)$ , se puede construir un AFD  $M' = (\Sigma, \mathcal{P}(Q), \{q_0\}, F', \delta)$  equivalente a  $M$ , i.e. tal que  $L(M) = L(M')$ , donde

$$\delta : \mathcal{P}(Q) \times \Sigma \rightarrow \mathcal{P}(Q)$$

$$(S, a) \mapsto \delta(S, a) := \Delta(S, a)$$

$$F' = \{S \subseteq Q : S \cap F \neq \emptyset\}.$$

# Algoritmo para encontrar un AFD equivalente a un AFN

Dado un AFN  $M = (\Sigma, Q, q_0, F, \Delta)$ , encontrar un AFD  $M' = (\Sigma, Q', q_0, F', \delta)$  tal que  $L(M) = L(M')$ :

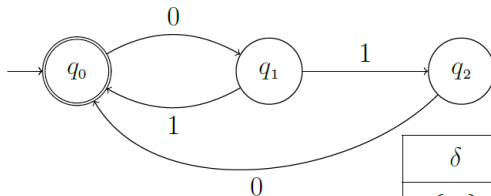
- 1 Para cada estado  $q \in Q$ , incluir el estado  $\{q\}$  en  $Q'$ .
- 2 Establecer como estado inicial a  $\{q_0\}$ .
- 3 Para cada  $q \in Q$  y cada  $a \in \Sigma$ , incluir el conjunto de estados  $\Delta(q, a) = \{q_{i_1}, q_{i_2}, \dots, q_{i_k}\} = S$  como un único estado en  $Q'$ . Incluir  $\delta(\{q\}, a) = \Delta(q, a)$  en la función  $\delta$ .
- 4 Para cada estado nuevo  $S \in Q'$  y para cada  $a \in \Sigma$ , hallar  $\delta(S, a) = \Delta(S, a) = S'$  y, si  $S' \notin Q'$ , incluir  $S'$  en  $Q'$  como un único estado.
- 5 Repetir el Paso 4 hasta que no se puedan añadir más estados a  $Q'$ .
- 6 Eliminar de  $Q'$  los estados inalcanzables.
- 7 Incluir en  $F'$  aquellos elementos de  $S \in Q'$  tales que  $S \cap F \neq \emptyset$ .



# Ejemplo

## Ejercicio

Hallar un AFD que acepte el lenguaje  $L = (01 \cup 010)^*$  sobre  $\Sigma = \{0, 1\}$ .



$\Delta$	0	1
$q_0$	$\{q_1\}$	$\emptyset$
$q_1$	$\emptyset$	$\{q_0, q_2\}$
$q_2$	$\{q_0\}$	$\emptyset$

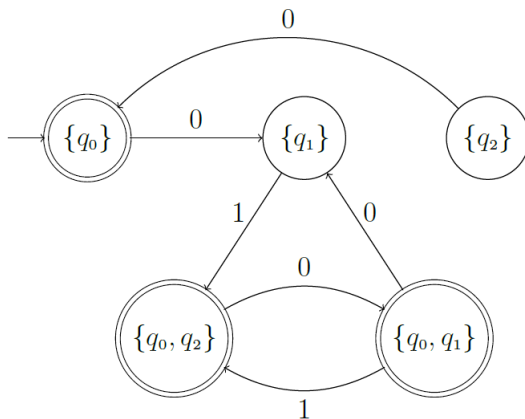
$\delta$	0	1
$\{q_0\}$	$\{q_1\}$	$\emptyset$
$\{q_1\}$	$\emptyset$	$\{q_0, q_2\}$
$\{q_2\}$	$\{q_0\}$	$\emptyset$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\emptyset$
$\{q_0, q_1\}$	$\{q_1\}$	$\{q_0, q_2\}$

# Ejemplo

## Ejercicio

Hallar un AFD que acepte el lenguaje  $L = (01 \cup 010)^*$  sobre  $\Sigma = \{0, 1\}$ .

$\delta$	0	1
$\{q_0\}$	$\{q_1\}$	$\emptyset$
$\{q_1\}$	$\emptyset$	$\{q_0, q_2\}$
$\{q_2\}$	$\{q_0\}$	$\emptyset$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\emptyset$
$\{q_0, q_1\}$	$\{q_1\}$	$\{q_0, q_2\}$

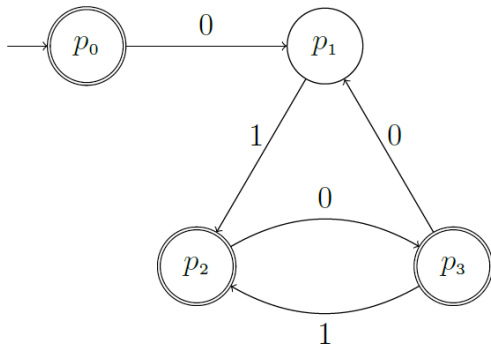


# Ejemplo

## Ejercicio

Hallar un AFD que acepte el lenguaje  $L = (01 \cup 010)^*$  sobre  $\Sigma = \{0, 1\}$ .

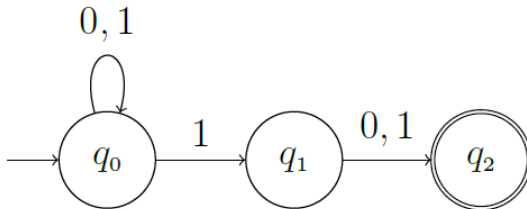
$\delta$	0	1
$\{q_0\}$	$\{q_1\}$	$\emptyset$
$\{q_1\}$	$\emptyset$	$\{q_0, q_2\}$
$\{q_2\}$	$\{q_0\}$	$\emptyset$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\emptyset$
$\{q_0, q_1\}$	$\{q_1\}$	$\{q_0, q_2\}$



# Ejercicio

## Ejercicio

Hallar un AFD que acepte el lenguaje de todas las cadenas de longitud  $\geq 2$  en las que el penúltimo símbolo es uno, sobre  $\Sigma = \{0, 1\}$ .



# Outline

- 1 Autómatas Finitos Deterministas
  - Definiciones Básicas
  - Complemento
  - Producto Cartesiano
  - Minimización
- 2 Autómatas Finitos No Deterministas
  - Definición y Representación
  - Equivalencia Computacional entre los AFD y los AFN
- 3 Autómatas con Transiciones  $\lambda$ 
  - Definición y Representación
  - Equivalencia Computacional entre los AFN y los AFN- $\lambda$
- 4 Lenguajes Regulares & Autómatas Finitos
  - Teorema de Kleene
  - Propiedades de Clausura de los Lenguajes Regulares
  - Teorema de Myhill-Nerode

# Outline

- 1 Autómatas Finitos Deterministas
  - Definiciones Básicas
  - Complemento
  - Producto Cartesiano
  - Minimización
- 2 Autómatas Finitos No Deterministas
  - Definición y Representación
  - Equivalencia Computacional entre los AFD y los AFN
- 3 Autómatas con Transiciones  $\lambda$ 
  - Definición y Representación
  - Equivalencia Computacional entre los AFN y los AFN- $\lambda$
- 4 Lenguajes Regulares & Autómatas Finitos
  - Teorema de Kleene
  - Propiedades de Clausura de los Lenguajes Regulares
  - Teorema de Myhill-Nerode

# Autómatas con Transiciones $\lambda$ (AFN- $\lambda$ )

- Un **autómata finito con transiciones  $\lambda$  (AFN- $\lambda$ )** es un autómata finito no determinista  $M = (\Sigma, Q, q_0, F, \Delta)$  en el que la función de transición está definida como:

$$\Delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow \mathcal{P}(Q)$$

- Además de las transiciones no deterministas usuales, permite hacer **transiciones nulas** o **transiciones espontáneas**:

$$\Delta(q, \lambda) = \{q_{i_1}, q_{i_2}, \dots, q_{i_k}\}.$$

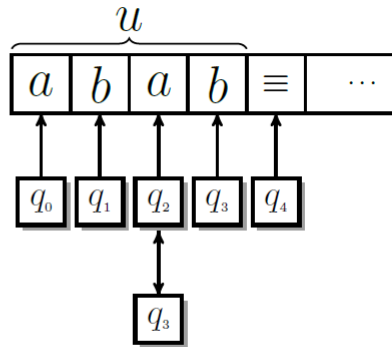
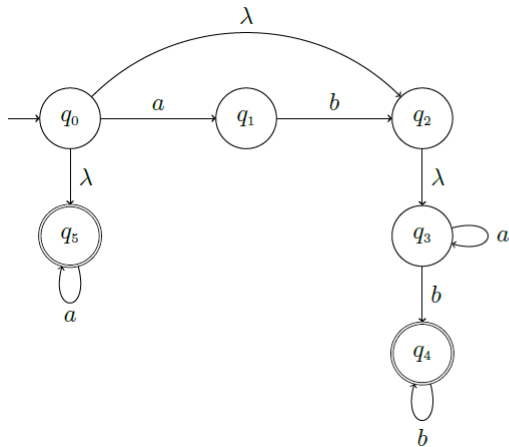
- Esta transición significa: estando en el estado  $q$ , el autómata puede cambiar aleatoriamente a cualquiera de los estados  $q_{i_1}, q_{i_2}, \dots, q_{i_k}$ , independientemente del símbolo leído y sin mover la unidad de control a la derecha.

# Autómatas con Transiciones $\lambda$ (AFN- $\lambda$ )

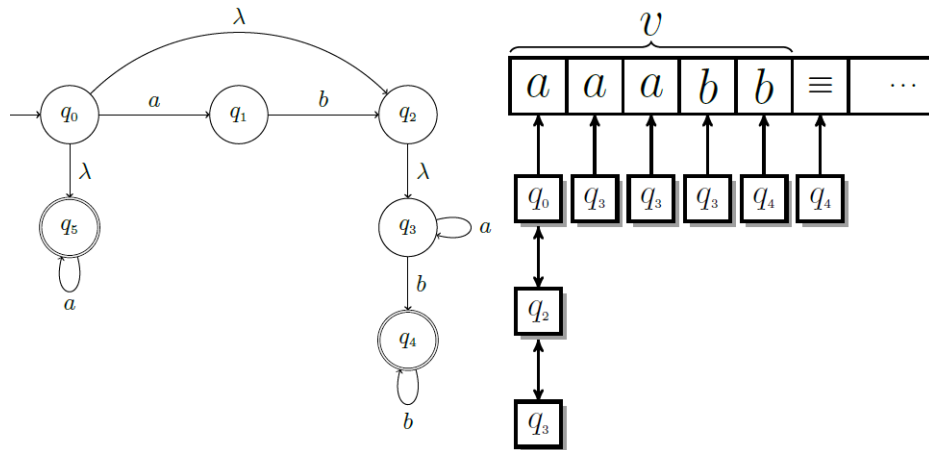
- Un AFN- $\lambda$   $M = (\Sigma, Q, q_0, F, \Delta)$  puede cambiar de estado sin consumir ningún símbolo de la cinta.
- En el grafo las transiciones pueden tener la etiqueta  $\lambda$ .
- Puede tener múltiples procesamientos de una cadena incluyendo procesamientos abortados, de aceptación y de rechazo.
- Una cadena  $u \in \Sigma^*$  es aceptada si existe por lo menos un procesamiento completo de  $u$ , desde  $q_0$ , que termina en un estado de aceptación, i.e.  $\hat{\Delta}(q_0, u) \cap F \neq \emptyset$ .
- En el grafo, se dice que  $u \in \Sigma^*$  es aceptada si existe por lo menos una trayectoria desde  $q_0$  hasta un estado de aceptación, cuyas etiquetas son los símbolos de  $u$  intercalados con cero, uno o más  $\lambda$ s.



# Ejemplo



# Ejemplo



# Ejercicio

## Ejercicio

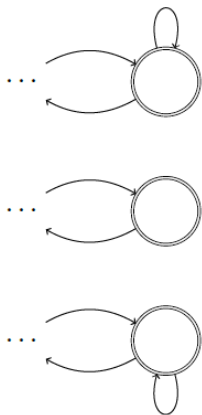
Dado  $\Sigma = \{a, b\}$ , diseñar AFN- $\lambda$ , diseñar autómatas que acepten los siguientes lenguajes:

- ①  $a^* \cup (ab \cup ba)^* \cup b^+$
- ②  $a^*(ab \cup ba)^*b^+$

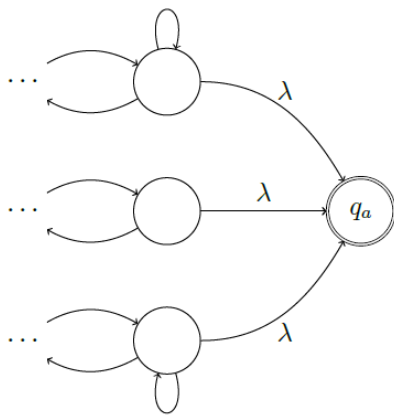
# Estado de Aceptación Único

- Un autómata cualquiera  $M$  se puede modificar, sin alterar el lenguaje aceptado, de manera que tenga un único estado de aceptación.
- Se agrega un estado  $q_a$  que será el único estado de aceptación.
- Se trazan transiciones  $\lambda$  desde los estados de aceptación originales hasta  $q_a$ .
- El resto de transiciones se dejan iguales.
- Los estados de aceptación originales se convierten en estados de no aceptación.

# Estado de Aceptación Único



Tres estados de aceptación



Único estado de aceptación  $q_a$

# Outline

## 1 Autómatas Finitos Deterministas

- Definiciones Básicas
- Complemento
- Producto Cartesiano
- Minimización

## 2 Autómatas Finitos No Deterministas

- Definición y Representación
- Equivalencia Computacional entre los AFD y los AFN

## 3 Autómatas con Transiciones $\lambda$

- Definición y Representación
- Equivalencia Computacional entre los AFN y los AFN- $\lambda$

## 4 Lenguajes Regulares & Autómatas Finitos

- Teorema de Kleene
- Propiedades de Clausura de los Lenguajes Regulares
- Teorema de Myhill-Nerode

# Equivalencia Computacional entre los AFN y los AFN- $\lambda$

- Los modelos AFN Y AFN- $\lambda$  son computacionalmente equivalentes: aceptan los mismos lenguajes.
- Un AFN puede ser considerado como un AFN- $\lambda$  en el que simplemente hay cero transiciones  $\lambda$ .
- Dado un AFN- $\lambda$ , también se puede construir un AFN que acepte el mismo lenguaje.
- Este proceso se basa en añadir transiciones que simulen las transiciones  $\lambda$ .
- Dichas simulaciones están basadas en el concepto de  $\lambda$ -clausura.

# $\lambda$ -Clausura

- Sea  $M = (\Sigma, Q, q_0, F, \Delta)$  un AFN- $\lambda$ .
- La  **$\lambda$ -clausura de un estado**  $q \in Q$ , denotada como  $\lambda[q]$  es el conjunto de estados de  $Q$  a los que se puede llegar desde  $q$  mediante 0, 1 o más transiciones  $\lambda$ .
- La  **$\lambda$ -clausura de un conjunto de estados**  $\{q_1, \dots, q_k\}$ , donde  $q_i \in Q$  para  $i \in \{1, 2, \dots, k\}$ , es:

$$\lambda[\{q_1, \dots, q_k\}] = \lambda[q_1] \cup \lambda[q_2] \cup \dots \cup \lambda[q_k].$$

- $\lambda[\emptyset] := \emptyset$ .



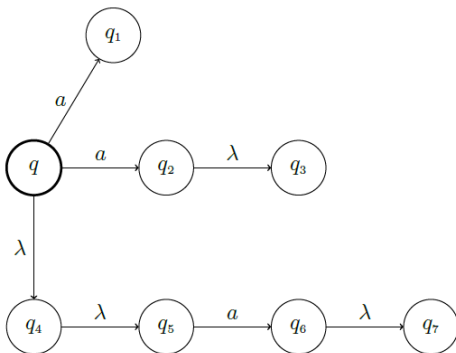
# Equivalencia Computacional entre los AFN y los AFN- $\lambda$

## Teorema

Dado un AFN- $\lambda$   $M = (\Sigma, Q, q_0, F, \Delta)$ , se puede construir un AFN  $M' = (\Sigma, Q, q_0, F', \Delta')$  sin transiciones  $\lambda$  equivalente a  $M$ , i.e. tal que  $L(M) = L(M')$ , donde

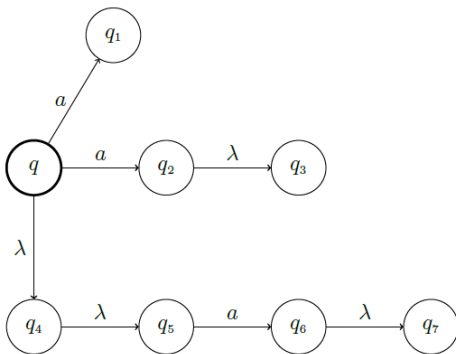
$$\begin{aligned}\Delta' : Q \times \Sigma &\rightarrow \mathcal{P}(Q) \\ (q, a) &\mapsto \Delta'(q, a) := \lambda[\Delta(\lambda[q], a)] \\ F' &= \{q \in Q : \lambda[q] \cap F \neq \emptyset\}.\end{aligned}$$

# Función de Transición $\Delta'$



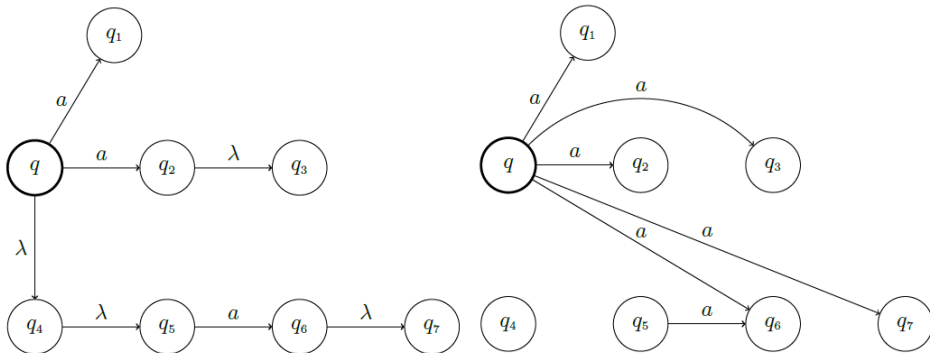
- Una vez procesada  $a$ , el autómata puede pasar desde el estado  $q$  a cualquier de  $\{q_1, q_2, q_3, q_6, q_7\}$ .
- Se tienen en cuenta todas las transiciones  $\lambda$  que preceden o prosiguen el procesamiento del símbolo  $a$  desde el estado  $q$ .

# Función de Transición $\Delta'$



$$\begin{aligned}\Delta'(q, a) &= \lambda[\Delta(\lambda[q], a)] \\ &= \lambda[\Delta(\{q, q_4, q_5\}, a)] \\ &= \lambda[\{q_1, q_2, q_6\}] \\ &= \lambda[q_1] \cup \lambda[q_2] \cup \lambda[q_6] \\ &= \{q_1\} \cup \{q_2, q_3\} \cup \{q_6, q_7\} \\ &= \{q_1, q_2, q_3, q_6, q_7\}\end{aligned}$$

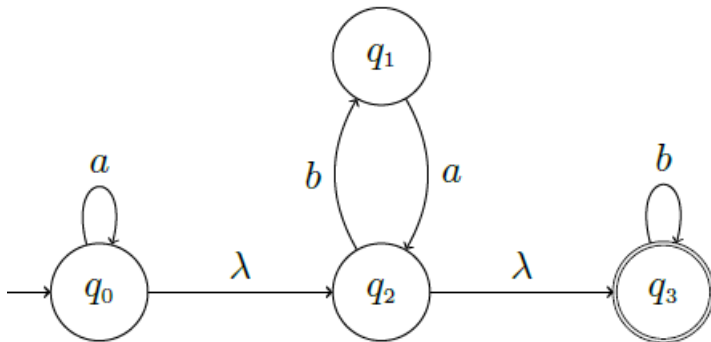
# Función de Transición $\Delta'$



# Ejercicio

## Ejercicio

Encontrar un AFN equivalente al siguiente AFN- $\lambda$ .



# Outline

## 1 Autómatas Finitos Deterministas

- Definiciones Básicas
- Complemento
- Producto Cartesiano
- Minimización

## 2 Autómatas Finitos No Deterministas

- Definición y Representación
- Equivalencia Computacional entre los AFD y los AFN

## 3 Autómatas con Transiciones $\lambda$

- Definición y Representación
- Equivalencia Computacional entre los AFN y los AFN- $\lambda$

## 4 Lenguajes Regulares & Autómatas Finitos

- Teorema de Kleene
- Propiedades de Clausura de los Lenguajes Regulares
- Teorema de Myhill-Nerode

# Outline

- 1 **Autómatas Finitos Deterministas**
  - Definiciones Básicas
  - Complemento
  - Producto Cartesiano
  - Minimización
- 2 **Autómatas Finitos No Deterministas**
  - Definición y Representación
  - Equivalencia Computacional entre los AFD y los AFN
- 3 **Autómatas con Transiciones  $\lambda$** 
  - Definición y Representación
  - Equivalencia Computacional entre los AFN y los AFN- $\lambda$
- 4 **Lenguajes Regulares & Autómatas Finitos**
  - Teorema de Kleene
  - Propiedades de Clausura de los Lenguajes Regulares
  - Teorema de Myhill-Nerode

# Teorema de Kleene

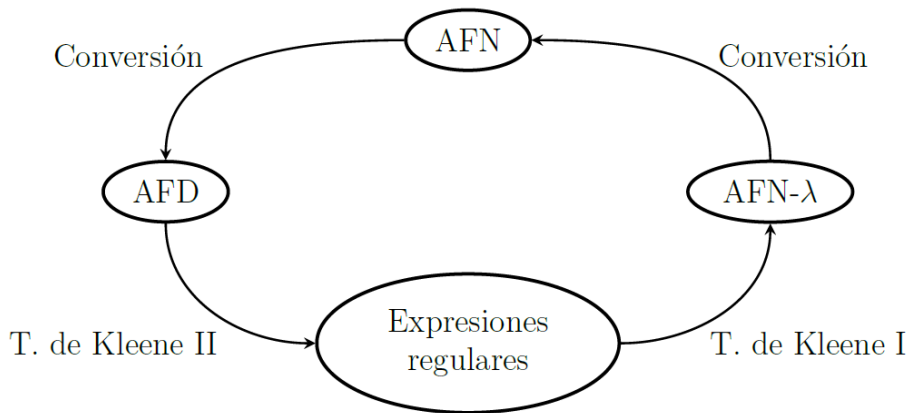
- Se ha mostrado la equivalencia computacional entre los modelos AFD, AFN y AFN- $\lambda$
- Es decir, aceptan la misma colección de lenguajes.
- El Teorema de Kleene establece que dicha colección corresponde a los lenguajes regulares.

## Teorema de Kleene

Sea  $\Sigma$  un alfabeto dado. Un lenguaje sobre  $\Sigma$  es regular si y sólo si es aceptado por un autómata finito (AFD, AFN o AFN- $\lambda$ ) con alfabeto de entrada  $\Sigma$ .



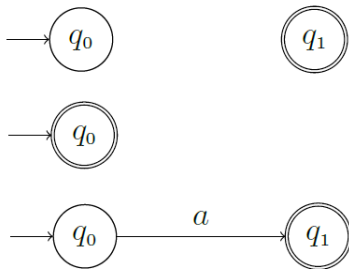
# Teorema de Kleene



# Teorema de Kleene - Parte I

Para un lenguaje regular, representado por una expresión regular  $R$  dada, se puede construir un AFN- $\lambda$   $M$  tal que  $L(M) = R$ .

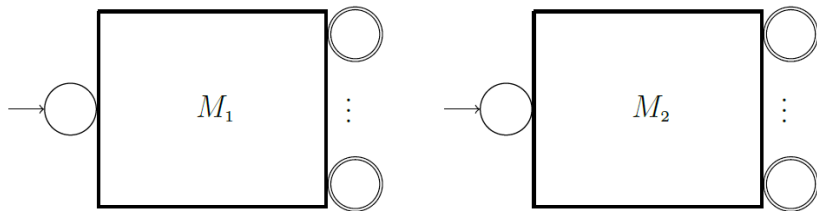
Se prueba para los lenguajes regulares básicos:  $\emptyset$ ,  $\{\lambda\}$ ,  $a \in \Sigma$ .



# Teorema de Kleene - Parte I

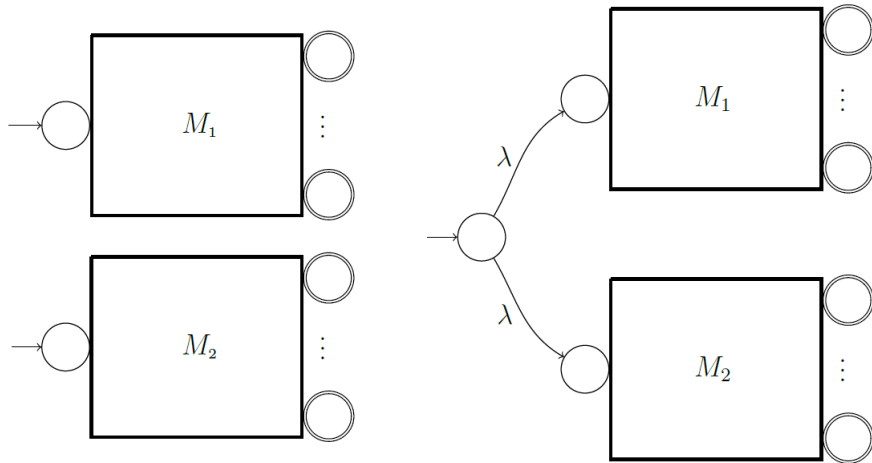
Para un lenguaje regular, representado por una expresión regular  $R$  dada, se puede construir un AFN- $\lambda$   $M$  tal que  $L(M) = R$ .

- Razonando recursivamente, supóngase que para las expresiones regulares  $R_1$  y  $R_2$  se tienen autómatas AFN- $\lambda$   $M_1$  y  $M_2$  tales que  $L(M_1) = R_1$  y  $L(M_2) = R_2$ .
- Probar que también se pueden construir autómatas AFN- $\lambda$  para aceptar  $R_1 \cup R_2$  y  $R_1 R_2$ .



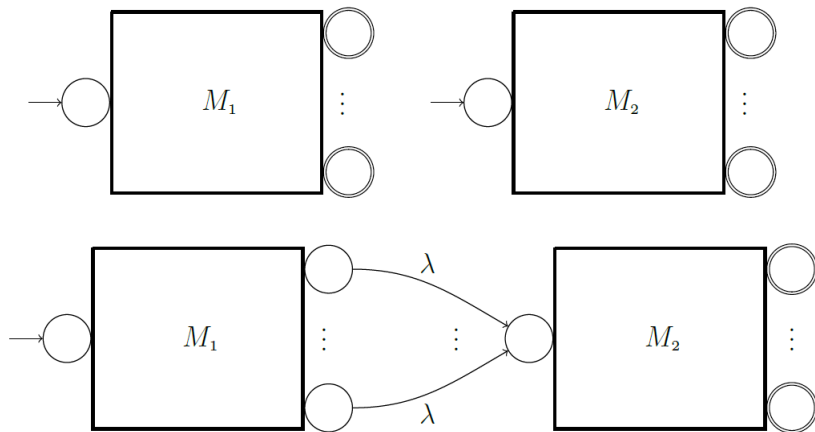
# Teorema de Kleene - Parte I

Para aceptar  $R_1 \cup R_2$ , se realiza una **conexión en paralelo** a  $M_1$  y  $M_2$ . Se agrega un estado inicial y se mantienen estados de aceptación.



# Teorema de Kleene - Parte I

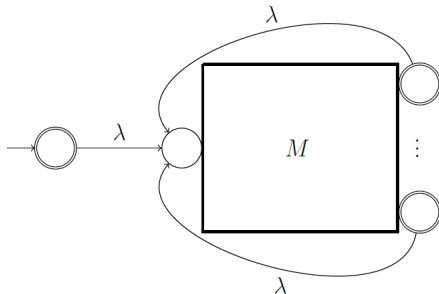
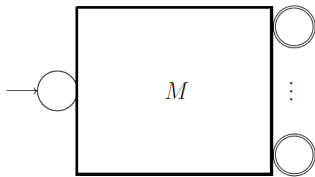
Para aceptar  $R_1R_2$ , se realiza una **conexión en serie** a  $M_1$  y  $M_2$ . Se mantiene el estado de inicial de  $M_1$  y los estados de aceptación de  $M_2$ .



# Teorema de Kleene - Parte I

Para un lenguaje regular, representado por una expresión regular  $R$  dada, se puede construir un AFN- $\lambda$   $M$  tal que  $L(M) = R$ .

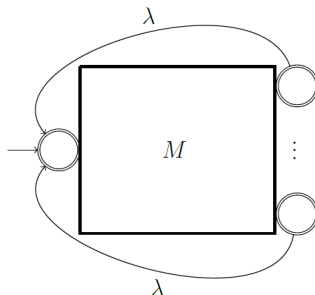
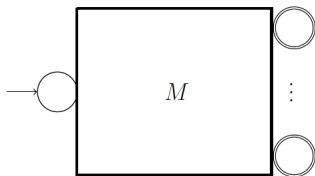
- Razonando recursivamente, supóngase que para la expresión regular  $R$  se tiene un autómata AFN- $\lambda$   $M$  tal que  $L(M) = R$ .
- Probar que también se pueden construir un autómata AFN- $\lambda$  para aceptar  $R^*$ .



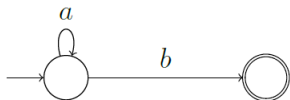
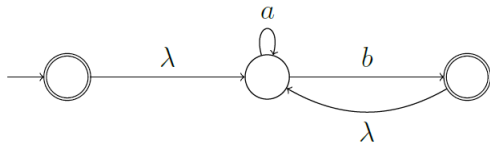
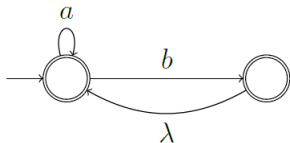
# Teorema de Kleene - Parte I

Para un lenguaje regular, representado por una expresión regular  $R$  dada, se puede construir un AFN- $\lambda$   $M$  tal que  $L(M) = R$ .

- Se podría pensar que el siguiente autómata acepta  $R^*$ , pero no es así.
- En la siguiente diapositiva, se muestra un contraejemplo.



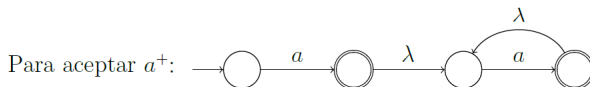
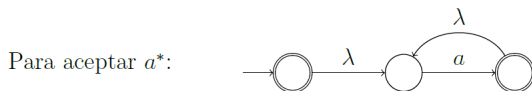
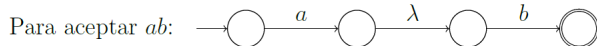
# Teorema de Kleene - Parte I

 $M$ : $M''$ : $M'$ :

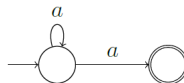
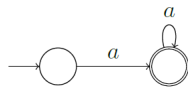


# Simplificaciones

Según el procedimiento:



Simplificación:



# Ejercicio

## Ejercicio

Construir un AFN- $\lambda$  que acepte el lenguaje  $(bc \cup cb)^* a^* b \cup (b^* ca)^* c^+$ .

# Teorema de Kleene - Parte II

## Teorema de Kleene - Parte II

Dado un autómata  $M$ , ya sea un AFD, AFN o AFN- $\lambda$ , se puede encontrar una expresión regular  $R$  tal que  $L(M) = R$ .

- La demostración también es constructiva.
- Se basa en la noción de Grafo Etiquetado Generalizado.

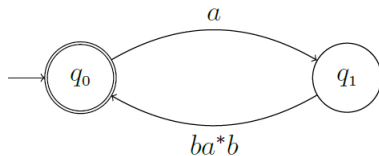
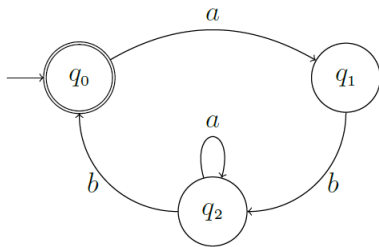
## Grafo Etiquetado Generalizado (GEG)

Un **Grafo Etiquetado Generalizado (GEG)** es un grafo como el del autómata excepto que las etiquetas de los arcos entre estados pueden ser expresiones regulares y no simplemente símbolos de un alfabeto.

## Teorema de Kleene - Parte II

La parte II del Teorema de Kleene se puede demostrar constructivamente mediante el siguiente procedimiento:

- 1 Eliminar uno a uno los estados del autómata original  $M$ , obteniendo en cada paso un GEG cuyo lenguaje aceptado es  $L(M)$ .
- 2 Cuando el grafo se reduce a dos estados (uno de ellos debe ser el estado inicial), el lenguaje aceptado se puede obtener por simple inspección.

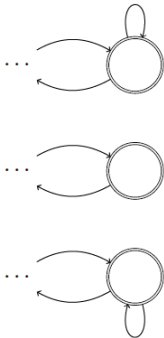


# Procedimiento para encontrar una $R$ tal que $L(M) = R$

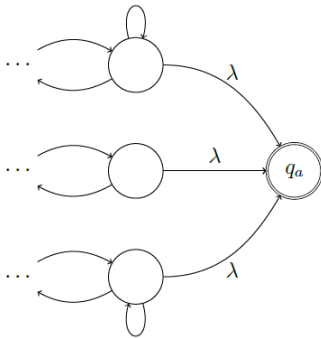
- 1 Reemplazar múltiples arcos entre dos estados por uno:



- 2 Modificar el GEG  $G$  para que tenga un único estado de aceptación.



Tres estados de aceptación

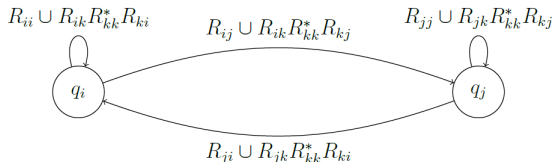
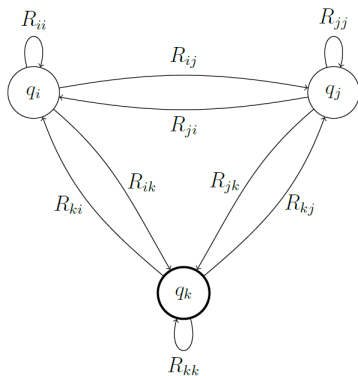


Único estado de aceptación  $q_a$

# Procedimiento para encontrar una $R$ tal que $L(M) = R$

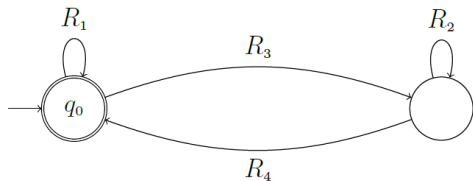
- ③ Ciclo iterativo por medio del cual se van eliminando uno a uno los estados de  $G$  hasta que permanezcan únicamente dos estados (uno de ellos debe ser  $q_0$  y uno de ellos debe ser el estado de aceptación).
- $R_{ij}$ : etiqueta entre  $q_i$  y  $q_j$ .
  - $R_{ij} = \emptyset$  si no existe arco entre  $q_i$  y  $q_j$ .
  - Escoger un estado cualquiera  $q_k$  diferente de  $q_0$  y que no sea de aceptación.
  - Eliminar  $q_k$  añadiendo transiciones entre los estados restantes de manera que el lenguaje aceptado no se altere.
  - Para  $q_i$  y  $q_j$  diferentes de  $q_k$ ,  $q_k$  sirve de puente entre  $q_i$  y  $q_j$  y entre  $q_j$  y  $q_i$ .
  - También  $q_k$  sirve de puente tanto para  $q_i$  como para  $q_j$  para conectarse consigo mismo.
  - Al eliminar  $q_k$  se deben tener en cuenta todas estas trayectorias en los arcos que queden.
  - Realizar esto para todas las parejas de  $q_i$  y  $q_j$  diferentes a  $q_k$ .

# Procedimiento para encontrar una $R$ tal que $L(M) = R$

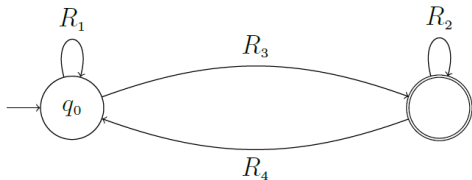


# Procedimiento para encontrar una $R$ tal que $L(M) = R$

- 4 Cuando haya solo dos estados, obtener la expresión regular de acuerdo a los siguientes casos:



$$L(M) = (R_1 \cup R_3 R_2^* R_4)^*.$$



$$L(M) = R_1^* R_3 (R_2 \cup R_4 R_1^* R_3)^*.$$

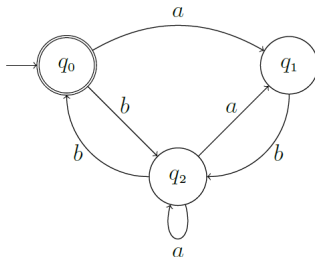


# Procedimiento para encontrar una $R$ tal que $L(M) = R$

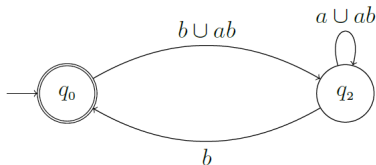
## Observaciones

- El procedimiento es flexible. Por ejemplo se puede dejar un estado de aceptación único después de haber eliminado algunos estados.
- No se pueden eliminar estados de aceptación ni el estado inicial.
- No es necesario memorizar las expresiones de los Pasos 3 y 4. Se pueden deducir por simple inspección.

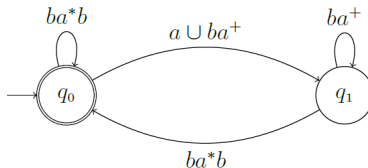
# Ejemplo



Eliminación del estado  $q_1$ :



Eliminación del estado  $q_2$ :



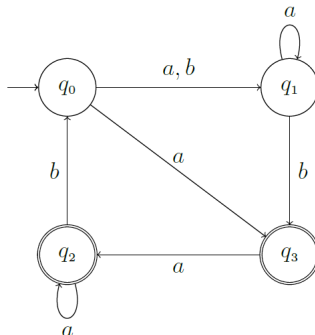
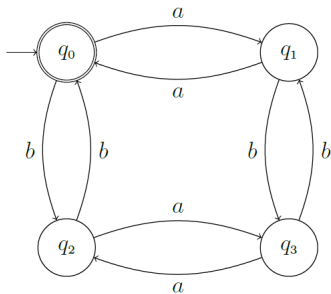
$$L(M) = [(b \cup ab)(a \cup ab)^*b]^*.$$

$$L(M) = [ba^*b \cup (a \cup ba^+)(ba^+)^*ba^*b]^*.$$

# Ejercicio

## Ejercicio

Encontrar una expresión regular para representar los lenguajes aceptados por los siguientes autómatas.



# Outline

## 1 Autómatas Finitos Deterministas

- Definiciones Básicas
- Complemento
- Producto Cartesiano
- Minimización

## 2 Autómatas Finitos No Deterministas

- Definición y Representación
- Equivalencia Computacional entre los AFD y los AFN

## 3 Autómatas con Transiciones $\lambda$

- Definición y Representación
- Equivalencia Computacional entre los AFN y los AFN- $\lambda$

## 4 Lenguajes Regulares & Autómatas Finitos

- Teorema de Kleene
- Propiedades de Clausura de los Lenguajes Regulares
- Teorema de Myhill-Nerode

# Propiedades de Clausura de los Lenguajes Regulares

- La regularidad es preservada por ciertas operaciones.
- Los lenguajes regulares son cerrados bajo dichas operaciones.

## Teorema

Sean  $M$ ,  $M_1$  y  $M_2$  autómatas finitos definidos sobre el alfabeto  $\Sigma$  tales que  $L(M) = L$ ,  $L(M_1) = L_1$  y  $L(M_2) = L_2$ . Se pueden construir autómatas finitos que acepten:

①  $L_1 \cup L_2$

②  $L_1 L_2$

③  $L^*$

④  $L^+$

⑤  $\bar{L} = \Sigma^* - L$

⑥  $L_1 \cap L_2$

⑦  $L_1 - L_2$

⑧  $L_1 \triangle L_2$

# Propiedades de Clausura de los Lenguajes Regulares

- Por el Teorema de Kleene, los lenguajes aceptados por los autómatas finitos son los regulares.

## Teorema

Si  $L$ ,  $L_1$  y  $L_2$  son lenguajes regulares definidos sobre el alfabeto  $\Sigma$ , también son regulares los siguientes lenguajes:

①  $L_1 \cup L_2$

②  $L_1 L_2$

③  $L^*$

④  $L^+$

⑤  $\bar{L} = \Sigma^* - L$

⑥  $L_1 \cap L_2$

⑦  $L_1 - L_2$

⑧  $L_1 \triangle L_2$

# Outline

- 1 Autómatas Finitos Deterministas
  - Definiciones Básicas
  - Complemento
  - Producto Cartesiano
  - Minimización
- 2 Autómatas Finitos No Deterministas
  - Definición y Representación
  - Equivalencia Computacional entre los AFD y los AFN
- 3 Autómatas con Transiciones  $\lambda$ 
  - Definición y Representación
  - Equivalencia Computacional entre los AFN y los AFN- $\lambda$
- 4 Lenguajes Regulares & Autómatas Finitos
  - Teorema de Kleene
  - Propiedades de Clausura de los Lenguajes Regulares
  - Teorema de Myhill-Nerode

# Bibliografía

- ① Rodrigo De Castro Korgi. **Notas de Clase de Introducción a la Teoría de la Computación**. 2023. Contenidos e imágenes de estas notas fueron incluidas en esta presentación.
- ② Harry R. Lewis, Christos H. Papadimitriou. **Elements of the Theory of Computation**. Prentice Hall. 1998.
- ③ John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman. **Introduction to Automata Theory, Languages and Computation, Third Edition**. Pearson. 2006.