

# 3. Autómatas Finitos

Grado Ingeniería Informática  
Teoría de Autómatas y Lenguajes Formales

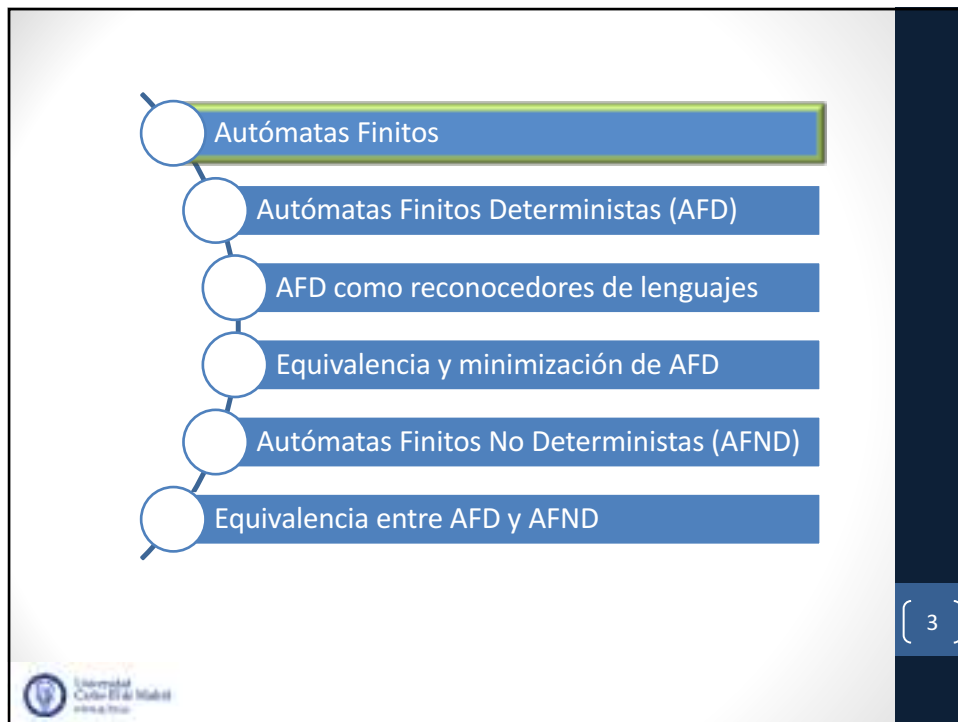


## Objetivos

- Definir el concepto de Autómata Finito Determinista (AFD).
- Definir el concepto de Autómata Finito No Determinista (AFND).
- Establecer las equivalencias entre AFD.
- Convertir un AFND en un AFD.
- Minimizar AFD.
- Identificar el tipo de lenguaje aceptado por un AFND.

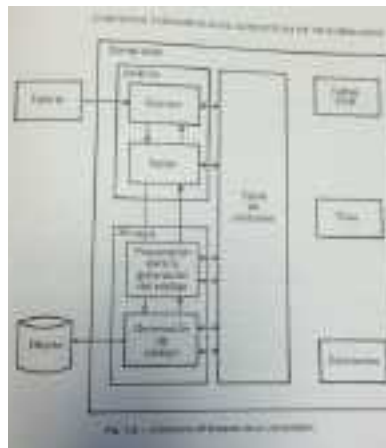
[ 2 ]





## Aplicaciones de AFs

- **Compilador:** Proceso de traducción que convierte un programa fuente escrito en un lenguaje de alto nivel a un programa objeto en código máquina y listo para ejecutarse en un ordenador –con poca o ninguna preparación adicional.
- **Fases:** análisis y síntesis



# Autómatas Finitos

- Los Autómatas Finitos son de dos tipos:

- Deterministas:**

- cada combinación (estado, símbolo de entrada) produce un solo (estado).

- No Deterministas:**

- cada combinación (estado, símbolo de entrada) produce varios (estado1, estado 2, ..., estado i).
    - son posibles transiciones con  $\lambda$

[ 5 ]



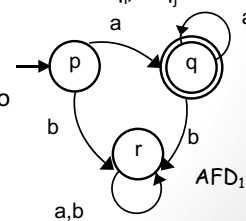
# Autómatas Finitos. Representación

- Se pueden representar mediante:

- Diagramas de transición o
- Tablas de transición

- Diagramas de transición:**

- Nodos etiquetados por los **estados** ( $q_i \in$  Conjunto de estados)
- Arcos** entre nodos  $q_i$  a  $q_j$  etiquetados con  $e_i$  ( $e_i$  es un símbolo de entrada) si existe la transición de  $q_i$  a  $q_j$  con  $e_i$
- El **estado inicial** se señala con  $\rightarrow$
- El estado final se señala con  $*$  o doble círculo



[ 6 ]



# Autómatas Finitos. Representación

## 2. Tablas de transición:

- Filas encabezadas por los estados ( $q_i \in \text{Conjunto de estados}$ )
- Columnas encabezadas por los símbolos de entrada ( $e_i \in \text{alfabeto de entrada}$ )

	$e_1$	$e_2$	...	$e_n$
$q_1$		$f(q_1, e_2)$		
...				
$*q_m$				

Estados

Símbolos de Entrada

[ 7 ]



[ 8 ]

## Autómatas Finitos Deterministas

- AF Deterministas, **AFD's**: se definen mediante una quintupla  $(\Sigma, Q, f, q_0, F)$ , donde:
  - $\Sigma$ : alfabeto de entrada
  - $Q$ : conjunto de estados, es conjunto finito no vacío, realmente un alfabeto para distinguir a los estados
  - $f: Q \times \Sigma \rightarrow Q$ , función de transición
  - $q_0 \in Q$ , estado inicial
  - $F \subseteq Q$ : conjunto de estados finales o de aceptación

[ 9 ]



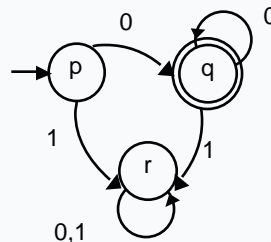
## Autómatas Finitos Deterministas

- Ejemplo: El  $AFD_1 = (\{0,1\}, \{p,q,r\}, f, p, \{q\})$ , donde  $f$  está definida por:

$$\begin{array}{ll} f(p,0) = q & f(p,1) = r \\ f(q,0) = q & f(q,1) = r \\ f(r,0) = r & f(r,1) = r \end{array}$$

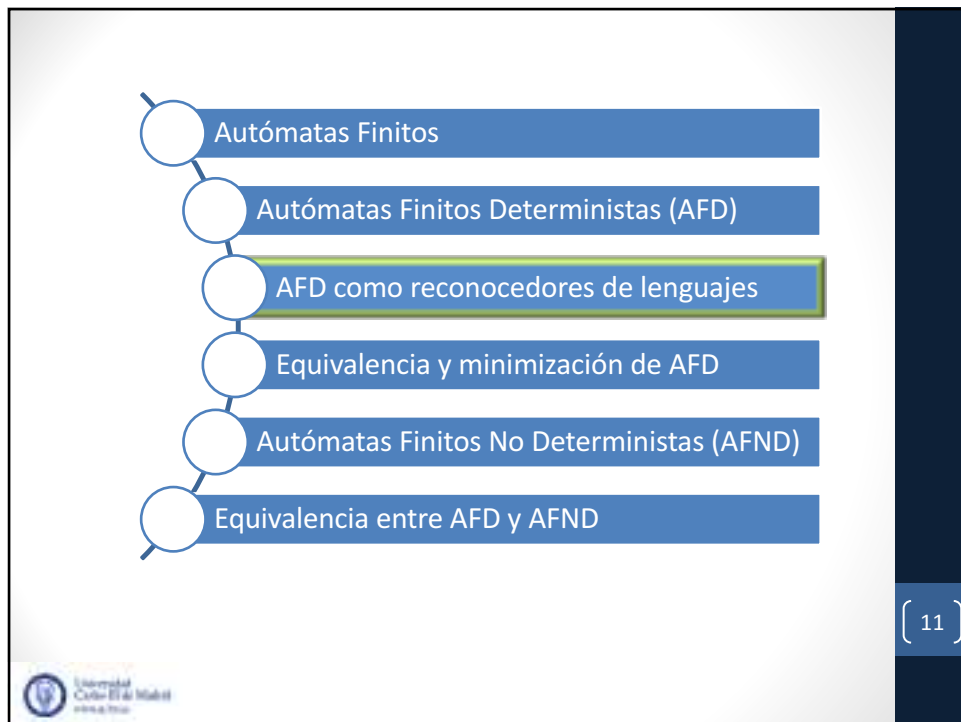
Tiene la tabla de transición y el diagrama de estados siguientes:

	0	1
p	q	r
*q	q	r
r	r	r



[ 10 ]





## AFD como reconocedores de Lenguajes

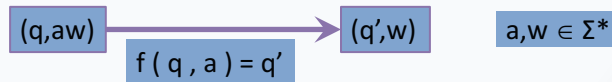
- Cuando un AF transita desde  $q_0$  a un estado final en varios movimientos, se ha producido el RECONOCIMIENTO o ACEPTACIÓN de la cadena de entrada
- Cuando un AF no es capaz de alcanzar un estado final, se dice que el AF NO RECONOCE la cadena de entrada y que ésta NO PERTENECE al lenguaje reconocido por el AF

[ 12 ]

## AFD. Conceptos Básicos

- **Configuración:** es un par ordenado de la forma  $(q,w)$  donde:
  - $q$ : estado actual del AF
  - $w$ : cadena que le queda por leer en ese instante,  $w \in \Sigma^*$  Universo del Discurso
- **Configuración inicial:**  $(q_0, t)$ 
  - $q_0$ : estado inicial
  - $t$ : cadena de entrada a reconocer por el AFD,  $t \in \Sigma^*$
- **Configuración final:**  $(q_f, \lambda)$ 
  - $q_f$ : estado final
  - $\lambda$  la cadena de entrada ha sido leída completamente

- **Movimiento:** es el tránsito entre dos configuraciones.



[ 13 ]



## AFD. Conceptos Básicos

### Extensión a palabra de la función de transición $f, f'$ :

Es la ampliación de la definición de  $f$  a palabras de  $\Sigma^*$ , i.e.  $w \in \Sigma^*$

- $f': Q \times \Sigma^* \rightarrow Q$   
a partir de  $f$ , que sólo considera palabras de longitud 1,  
hay que añadir:
  - $f'(q, \lambda) = q \quad \forall q \in Q$
  - $f'(q, a \cdot x) = f'(f(q, a), x) \quad \forall q \in Q, a \in \Sigma, x \in \Sigma^*$

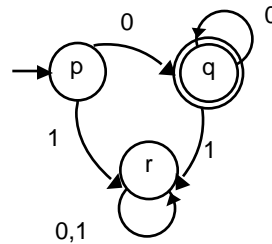
[ 14 ]



## AFD. Conceptos Básicos

- En el AFD<sub>1</sub> (de la figura), indicar el resultado de las siguientes expresiones:

- $f'(p, \lambda)$
- $f'(p, 0^n)$
- $f'(p, 11)$
- $f'(p, 0011010)$
- $f'(p, 100)$



[ 15 ]



## AFD. Conceptos Básicos

### Lenguaje asociado a un AFD:

- Sea un AFD  $= (\Sigma, Q, f, q_0, F)$ , se dice que una palabra  $x$  es aceptada o **reconocida** por el AFD si  $f'(q_0, x) \in F$
- Se llama **lenguaje asociado a un AFD** al conjunto de todas las palabras aceptadas por éste:

$$L = \{ x / x \in \Sigma^* \text{ and } f'(q_0, x) \in F \}$$

- Si  $F = \{ \} = \emptyset \Rightarrow L = \emptyset$
- Si  $F = Q \Rightarrow L = \Sigma^*$
- Otra definición:

$$L = \{ x / x \in \Sigma^* \text{ and } (q_0, x) \rightarrow (q, \lambda) \text{ and } q \in F \}$$

[ 16 ]

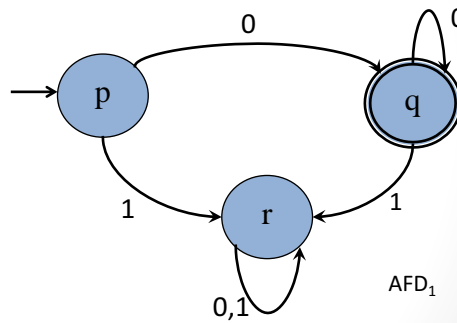




## AFD. Conceptos Básicos

En el AFD<sub>1</sub>

- Cuál es  $L(\text{AFD}_1) = \text{¿?}$
- Y si se hace  $F = \{r\}$ ,  
cuál es  $L(\text{AFD}_1) = \text{¿?}$



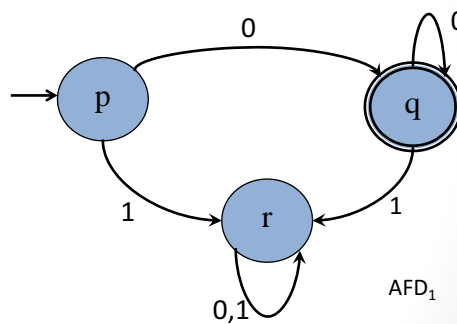
[ 17 ]



## AFD. Conceptos Básicos

En el AFD<sub>1</sub>

- Cuál es  $L(\text{AFD}_1) = \{0^n / n > 0\}$ .



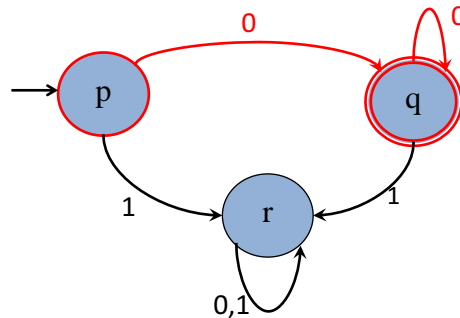
[ 18 ]



## AFD. Conceptos Básicos

En el AFD<sub>1</sub>

- Cuál es  $L(\text{AFD}_1) = \{0^n / n > 0\}$ . **Comprobación**



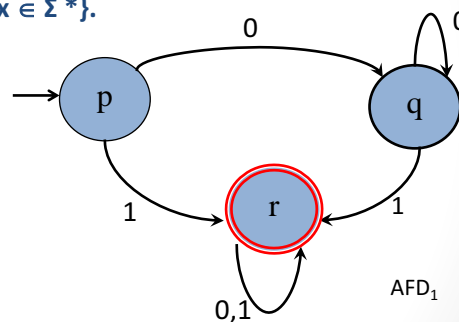
Desde p, con el número de 0's que sea, pero siempre al menos uno, se llega al estado final

[ 19 ]

## AFD. Conceptos Básicos

En el AFD<sub>1</sub>

- Y si se hace  $F = \{r\}$ ,  
 $L(\text{AFD}_1) = \{0^n 1x / n \geq 0, x \in \Sigma^*\}$ .



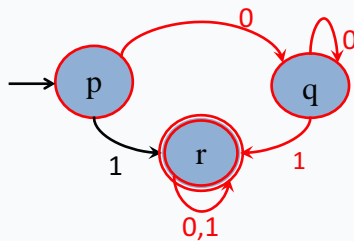
AFD<sub>1</sub>

[ 20 ]

## AFD. Conceptos Básicos

En el AFD<sub>1</sub>,

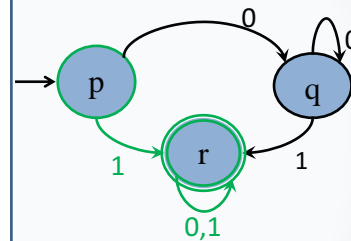
- comprobar que si se hace  $F = \{r\}$ ,  $L(\text{AFD}_1) = \{0^n 1x \mid n \geq 0, x \in \Sigma^*\}$ .



Desde p, con un "0" llego al estado q y desde allí se pueden aceptar tantos 0s como sean.

Luego con un 1 salto al estado final y allí puedo terminar o reconocer cualquier cadena de 0s y 1s.

**Expresión regular:  $L_A = 0^*1(0+1)^*$**



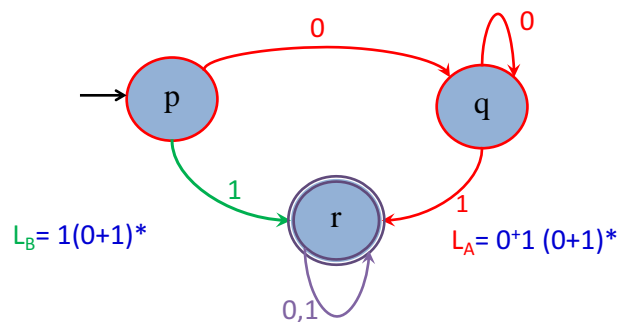
**Expresión regular:  $L_B = 1(0+1)^*$**

[ 21 ]

## AFD. Conceptos Básicos

En el AFD<sub>1</sub>,

- comprobar que si se hace  $F = \{r\}$ ,  $L(\text{AFD}_1) = \{0^n 1x \mid n \geq 0, x \in \Sigma^*\}$ .



$L_B = 1(0+1)^*$

$L_A = 0^*1(0+1)^*$

**Expresión regular  $L_A \cup L_B = 0^*1(0+1)^*$**

[ 22 ]

## AFD. Conceptos Básicos

### Estados accesibles y Autómatas conexos:

- Sea un AFD  $= (\Sigma, Q, f, q_0, F)$ , el estado  $p \in Q$  es ACCESIBLE desde  $q \in Q$  si  $\exists x \in \Sigma^* f'(q, x) = p$ . En otro caso se dice que INACCESIBLE.  
Todo estado es accesible desde sí mismo pues  $f'(p, \lambda) = p$

#### Teoremas:

- teorema 3.2.2, libro 1 de la bibliografía.  
Sea un AFD,  $|Q| = n, \forall p, q \in Q$   $p$  es accesible desde  $q$   
**sii**  $\exists x \in \Sigma^*, |x| < n / f'(q, x) = p$
- teorema 3.2.3, libro 1 de la bibliografía  
Sea un AFD,  $|Q| = n$ , entonces  $L_{AFD} \neq \emptyset$  **sii** el AFD acepta al menos una palabra  $x \in \Sigma^*, |x| < n$   
Nota: sii= "si y solo si"



[ 23 ]

## AFD. Conceptos Básicos

### Estados accesibles y Autómatas conexos:

Sea un AFD  $= (\Sigma, Q, f, q_0, F)$ . Diremos que el autómata es conexo si todos los estados de  $Q$  son accesibles desde  $q_0$

Dado un autómata no conexo, podemos obtener a partir de él otro autómata equivalente conexo eliminando los estados inaccesibles desde el estado inicial. Los autómatas reconocen el mismo lenguaje.

#### Eliminación de estados inaccesibles.

- ¿Qué algoritmo, para ser implementado en un programa, se podría implementar para marcar los accesibles?



[ 24 ]

## AFD. Ejercicios

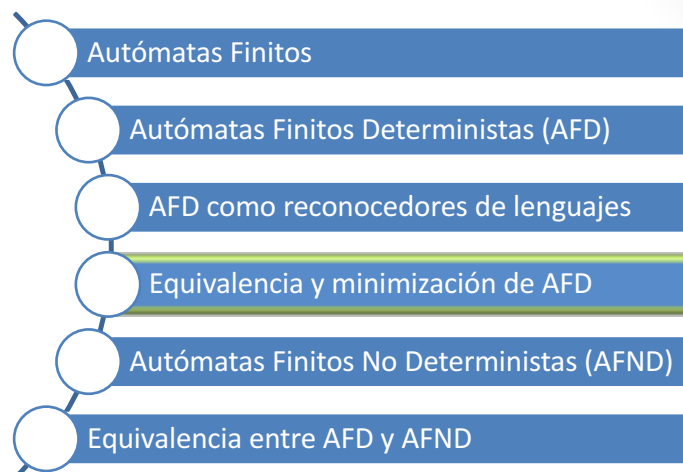
- Hallar el AFD conexo equivalente al dado:  $AF = (\{0,1\}, \{p,q,r,s\}, p, f, \{q,r,s\})$ , donde  $f$  viene dada por la tabla.

- Se eliminan todos los estados innacesibles y todos las transiciones (i.e. arcos) que salen desde dichos estados innacesibles.

	0	1
p	r	p
*q	r	p
*r	r	p
*s	s	s

- Indicar, además el lenguaje reconocido por ambos AFD's (original y conexo).

[ 25 ]



[ 26 ]



## AFD. Equivalencia y Minimización

- Es posible tener varios autómatas que reconozcan el mismo lenguaje.
- Para todo autómata se puede obtener un autómata equivalente (i.e. reconoce el mismo lenguaje) donde el número de estados del autómata sea el mínimo.
- **¿Por qué interesa obtener el mínimo?** (Apartado 4.4 Libro 2 bibliografía)

[ 27 ]



## AFD. Equivalencia y Minimización

### ¿Por qué interesa obtener el AFD mínimo? (Ap. 4.3 y 4.4 Libro 2 bibliografía)

- Se dispone de un descriptor del lenguaje (lenguaje regular): gramática tipo 3, AFD, AFND, expresión regular.
- Se plantean problemas de decisión:
  - ¿El lenguaje descrito es vacío?
  - ¿Existe una determinada cadena  $w$  en el lenguaje descrito?
  - ¿Dos descripciones de un lenguaje describen realmente el mismo lenguaje?
    - Nota: usualmente los lenguajes son infinitos, con lo que no es posible plantear la pregunta y recorrer el conjunto INFINITO de cadenas.

- Los algoritmos para responder a las dos primeras preguntas son sencillos. ¿Pero y para la última pregunta?

**¿Dos descripciones de un lenguaje describen realmente el mismo lenguaje?**

**Consecuencia de esta comprobación: es necesario obtener el AFD mínimo equivalente**

[ 28 ]



## AFD. Equivalencia y Minimización

### Teoremas:

- **Equivalencia de estados:**  
 $p E q$ , donde  $p, q \in Q$ , si  $\forall x \in \Sigma^*$  se verifica que  

$$f'(p, x) \in F \Leftrightarrow f'(q, x) \in F$$
- **Equivalencia de orden (o de longitud) "n"**  
 $p E_n q$ , donde  $p, q \in Q$ , si  $\forall x \in \Sigma^* / |x| \leq n$  se verifica que  

$$f'(p, x) \in F \Leftrightarrow f'(q, x) \in F$$

$E$  y  $E_n$  son relaciones de equivalencia.

[ 29 ]



## AFD. Equivalencia y Minimización

### Equivalencia de estados – Casos particulares:

- $E_0$ , x palabra  $|x| \leq 0 \Rightarrow x = \lambda$  se verifica que  
 $p E_0 q$ ,  $\forall p, q \in Q$ , si  $\forall x \in \Sigma^* / |x| \leq 0$  se verifica que  

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

$$f'(p, x) = f'(p, \lambda) = p \text{ (por definición de } f')$$
  

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

Todos los estados finales de son  $E_0$  equivalentes.

- ✓  $p, q \in F$  se cumple que  $p E_0 q$
- ✓  $p, q \in Q - F$  se cumple que  $p E_0 q$

[ 30 ]



# AFD. Equivalencia y Minimización

## Equivalencia de estados – Casos particulares:

- $E_1$ , x palabra  $|x| \leq 1$ , ( $x \in \Sigma$ ) se verifica que

$p E_1 q, \forall p, q \in Q$ , si  $\forall x \in \Sigma^* / |x| \leq 1$  se verifica que

$$f'(p, x) \in F \Leftrightarrow f'(q, x) \in F$$

x es lambda o símbolo del alfabeto.

$$f'(p, x) = f'(p, a) = f(p, a) \quad \text{ó} \quad f'(p, x) = f'(p, \lambda) = p \quad (\text{por definición de } f')$$

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

Partiendo de p y q con una sola transición se debe llegar a un estado final para ambos casos o uno no final para ambos casos.

[ 31 ]



# AFD. Equivalencia y Minimización

## Propiedades

Nota: en estas expresiones matemáticas, "n" no significa  $|Q|$

- Lema:  $p E q \Rightarrow p E_n q, \forall n, p, q \in Q$
- Lema:  $p E_n q \Rightarrow p E_k q, \forall n > k$
- Lema:  $p E_{n+1} q \Leftrightarrow p E_n q \text{ and } f(p, a) E_n f(q, a) \forall a \in \Sigma$

- **Teorema:**  $p E q \Leftrightarrow p E_{n-2} q$ , donde  $n = |Q| > 1$

Aquí "n" sí significa  $|Q|$

(Teorema 5.1 (pag 117 libro 4 bibliografía))

$p E q$  sii  $\forall x \in \Sigma^*, |x| = m \leq n-2$  se verifica que  $f(p, x) \in F \Leftrightarrow f(q, x) \in F$

$m = n-2$  es el valor más pequeño que cumple este teorema

(n-1 sí lo cumple, pero n-3 no se garantiza que se cumpla)

[ 32 ]





## AFD. Equivalencia y Minimización

"E" es una relación de equivalencia. ¿Qué significa Q/E?

- Q/E es una partición de Q,
- $Q/E = \{C_1, C_2, \dots, C_m\}$ , donde  $C_i \cap C_j = \emptyset$ 
  - $p E q \Leftrightarrow (p, q \in C_i)$ , por lo tanto

$$\forall x \in \Sigma^* \text{ se verifica que } f'(p, x) \in C_i \Leftrightarrow f'(q, x) \in C_i$$

Nota: en libro 1 biblio,  $p, q \in C_i$  se representa por  $p = q = C_i$

- Para la relación de orden n
  - $E_n: Q/E_n = \{C_1, C_2, \dots, C_m\}$ ,  $C_i \text{ intersección } C_j = \emptyset$
  - $p E_n q \Leftrightarrow p, q \in C_i$
  - por lo tanto  $\forall x \in \Sigma^*, |x| \leq n$  se verifica que

$$f'(p, x) \in C_i \Leftrightarrow f'(q, x) \in C_i$$



[ 33 ]

## AFD. Equivalencia y Minimización

Propiedades. (Lemas)

- Lema: Si  $Q/E_n = Q/E_{n+1} \Rightarrow Q/E_n = Q/E_{n+i} \forall i = 0, 1, \dots$
- Lema: Si  $Q/E_n = Q/E_{n+1} \Rightarrow Q/E_n = Q/E$  conjunto cociente
- Lema: Si  $|Q/E_0| = 1 \Rightarrow Q/E_0 = Q/E_1$
- Lema:  $n = |Q| > 1 \Rightarrow Q/E_{n-2} = Q/E_{n-1}$
- $p E_{n+1} q \Leftrightarrow (p E_n q \text{ and } f(p, a) E_n f(q, a) \forall a \in \Sigma)$



[ 34 ]

## AFD. Equivalencia y Minimización

### Interpretación lemas anteriores:

El objetivo es obtener la partición  $Q/E$ , puesto que será el autómata mínimo, sin estados equivalentes .

- En cuanto se obtienen dos particiones consecutivas  $Q/E_k = Q/E_{k+1}$ , se para.
- Para obtener  $Q/E$ , hay que empezar por  $Q/E_0$ ,  $Q/E_1$ , etc.
- Para obtener  $Q/E$ , hay que obtener  $Q/E_{n-2}$  en el peor caso, ya que si se obtiene  $Q/E_{n-k} = Q/E_{n-k+1}$ , con  $k \geq 3$ , se habría obtenido ya  $Q/E$ .
- El lema  $p E_{n+1} q \Leftrightarrow p E_n q$  and  $f(p,a) E_n f(q,a) \forall a \in \Sigma$ , permite es extender la equivalencia de orden  $n$  desde  $E_0$  y  $E_1$

[ 35 ]



## AFD. Equivalencia y Minimización

### Teorema:

$$pEq \Leftrightarrow pE_{n-2}q \text{ donde } |Q| = n > 1 (**)$$

Es decir,  $p E q \text{ sii } \forall x \in \Sigma^*, |x| \leq n-2, f'(p,x) \in F \Leftrightarrow f'(q,x) \in F$

$n-2$  es el valor más pequeño que cumple este teorema

[ 36 ]



## AFD. Equivalencia y Minimización

Algoritmo formal para obtener  $Q/E$ :

**1.  $Q/E_0 = \{ F, \text{no } F \}$**

1ª división en función de si son o no estados finales.

**2.  $Q/E_{i+1}$**

partiendo de  $Q/E_i = \{C_1, C_2, \dots, C_n\}$ , se construye  $Q/E_{i+1}$ :

$p$  y  $q$  están en la misma clase si:

$p, q \in C_k \in Q/E_i \forall a \in \Sigma \Rightarrow f(p,a) \text{ y } f(q,a) \in C_m \in Q/E_i$

**3. Si  $Q/E_i = Q/E_{i+1}$  entonces  $Q/E_i = Q/E$**

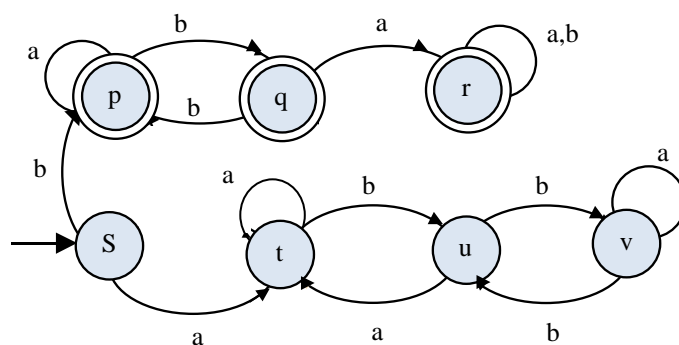
Si no, repetir el paso 2 partiendo de  $Q/E_{i+1}$

[ 37 ]



## AFD. Equivalencia y Minimización

- Ejercicio: Hallar el AFD mínimo equivalente



[ 38 ]



## AFD. Equivalencia

### Autómatas Equivalentes:

- **Estados equivalentes en AFD' s distintos:**
  - Sean 2 AFD' s:  $(\Sigma, Q, f, q_0, F)$  y  $(\Sigma', Q', f', q_0', F')$
  - Los estados  $p, q$  /  $p \in Q$  y  $q \in Q'$  son equivalentes ( $pEq$ ) si se verifica que  $f''(p, x) \in F \Leftrightarrow f''(q, x) \in F' \quad \forall x \in \Sigma^*$
- **Estados equivalentes en AFD' s distintos:**
  - Dos AFD' s son equivalentes si reconocen el mismo lenguaje, es decir: Si  $f(q_0, x) \in F \Leftrightarrow f(q_0', x) \in F' \quad \forall x \in \Sigma^*$ . Es decir:
  - **Dos AFD' s son equivalentes si lo son sus estados iniciales:  $q_0 E q_0'$**

[ 39 ]



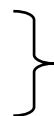
## AFD. Equivalencia

### ¿Qué es la suma directa de 2 AFD's?

Sean 2 AFD' s:

$$A1 = (\Sigma, Q_1, f_1, q_{01}, F_1)$$

$$A2 = (\Sigma', Q_2, f_2, q_{02}, F_2)$$



Donde  $Q_1 \cap Q_2 = \phi$

y  $\Sigma = \Sigma'$

Se llama **suma directa de A1 y A2** al AF A:

$$A = A1 + A2 = (\Sigma, Q_1 \cup Q_2, f, q_0, F_1 \cup F_2), \text{ donde:}$$

$q_0$  es el estado inicial de uno de los AF' s

$$f: f(p, a) = f_1(p, a) \text{ si } p \in Q_1$$

$$f(p, a) = f_2(p, a) \text{ si } p \in Q_2$$

[ 40 ]



## AFD. Equivalencia

□ **Teorema:** (el teorema (\*\*)) aplicado a la suma directa de dos autómatas):

sean  $A_1, A_2$  /  $Q_1 \cap Q_2 = \phi$ ,  $|Q_1| = n_1$ ,  $|Q_2| = n_2$

$A_1 \equiv A_2$  si  $q_{01} \equiv q_{02}$  en  $A = A_1 + A_2$

Es decir, si  $A_1$  y  $A_2$  aceptan las mismas palabras  $x$  /  $|x| \leq n_1 + n_2 - 2$

además,  $n_1 + n_2 - 2$  es el valor mínimo que cumple el teorema

[ 41 ]



## AFD. Equivalencia

Autómatas equivalentes, comprobación:

**Algoritmo para comprobar la equivalencia de AFDs**

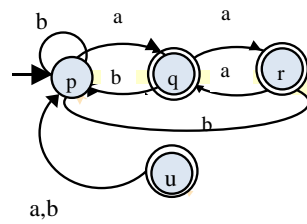
1. Se hace la suma directa de los dos AFD's
2. Se hace Q/E del AFD suma
3. Si los dos estados iniciales están en la misma clase de equivalencia de Q/E  $\Rightarrow$  los 2 AFD's son equivalentes

[ 42 ]

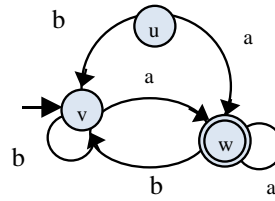


## AFD. Equivalencia

- **Ejercicio:** Comprobar que los autómatas A1 y A2 son equivalentes.



A1

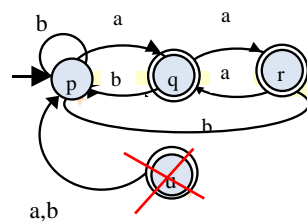


A2

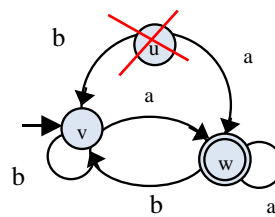
[ 43 ]

## AFD. Equivalencia

- **Ejercicio:** Comprobar que los autómatas A1 y A2 son equivalentes.



A1



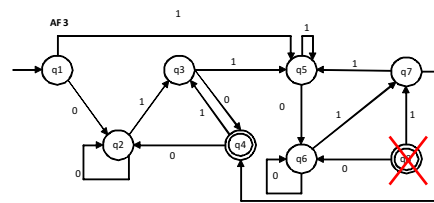
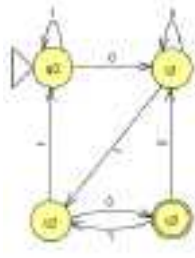
A2

Los estados u de ambos autómatas son inaccesibles y habría que eliminarlos antes de ver si son equivalentes.

[ 44 ]

## AFD. Equivalencia

- Ejercicio 7 de la hoja 2 de ejercicios: Comprobar si los dos AFD son equivalentes (obteniendo el mínimo para cada uno).



[ 45 ]

## AFD. Equivalencia

- Ejercicio 7 de la hoja 2 de ejercicios: AF1 es mínimo. AF2:
- $Q/E0 = \{\{q4, q8\}, \{q1, q2, q3, q5, q6, q7\}\} = C1, C2$  **OJO, Q8 es inaccesible y habría que quitarlo.** Aparece tachado en la solución.
- $Q/E1 = \{\{q4, q8\}, \{q1, q2, q5, q6\}, \{q3, q7\}\} = C1, C2, C3$
- $Q/E2 = \{\{q4, q8\}, \{q1, q5\}, \{q2, q6\}, \{q3, q7\}\} = C1, C2, C3, C4$
- $Q/E3 = Q/E2 = \{\{q4, q8\}, \{q1, q5\}, \{q2, q6\}, \{q3, q7\}\} = C1, C2, C3, C4$

	0	1	0	1	0	1	0	1
->q1	q2	q5	C2	C2	C2	C2	C3	C2
q2	q2	q3	C2	C2	C2	C3	C3	C4
q3	q4	q5	C1	C2	c1	C2	c1	C2
*q4	q2	q3	C2	C2	C2	C3	C3	C4
q5	q6	q5	C2	C2	C2	C2	C3	C2
q6	q6	q7	C2	C2	C2	C3	C3	C4
q7	q4	q5	c1	C2	c1	C2	c1	C2
*q8	q6	q7	C2	C2	C2	C3	C3	C4

- El AF2 y el AF1 son ISOMORFOS y por tanto son equivalentes.

[ 46 ]

## AFD. Equivalencia

- Sean dos autómatas:
  - $A_1 = (\Sigma, Q_1, f_1, q_{01}, F_1)$  y  $A_2 = (\Sigma', Q_2, f_2, q_{02}, F_2)$ , tales que  $|Q_1| = |Q_2|$
- Se dice que  $A_1$  y  $A_2$  **son isomorfos**, si existe una aplicación biyectiva  $i : Q_1 \rightarrow Q_2$  que cumple:
  1.  $i(q_{01}) = q_{02}$ , es decir, los estados iniciales son correspondientes
  2.  $q \in F_1 \Leftrightarrow i(q) \in F_2$  es decir, los estados finales son correspondientes
  3.  $i(f_1(q, a)) = f_2(i(q), a) \quad \forall a \in \Sigma, q \in Q_1$
- En definitiva, a cada estado le corresponde otro equivalente que solo se diferencia en el nombre de sus estados.
- Dos AFDs isomorfos, también son equivalentes y reconocen el mismo **lenguaje**.

[ 47 ]



## AFD. Minimización

Sea el AFD,  $A = (\Sigma, Q, f, q_0, F)$ :

1. Partir del AFD conexo, i.e. eliminar estados inaccesibles desde el estado inicial
2. Construir Q/E del autómata conexo
3. El AFD mínimo, salvo isomorfismos, es:

$$A' = (\Sigma, Q', f', q_0', F')$$

donde:

$$Q' = Q/E$$

$$f' \text{ se construye: } f'(C_i, a) = C_j \text{ si } \exists q \in C_i, p \in C_j / f(q, a) = p$$

$$q_0' = C_0 \text{ si } q_0 \in C_0, C_0 \in Q/E$$

$$F' = \{C / C \text{ contiene al menos un estado de } F (\exists \text{ un } q \in F \text{ tal que } q \in C)\}$$

### COROLARIO:

2 AFD's son equivalentes si sus AF mínimos respectivos son isomorfos.

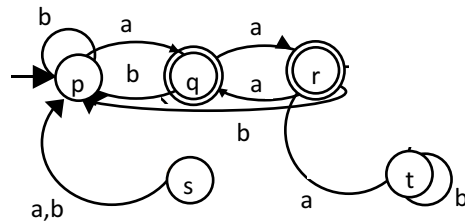
[ 48 ]





## AFD. Ejercicio

Hallar el AFD mínimo equivalente al dado:



[ 49 ]



[ 50 ]

# Autómatas Finitos No Deterministas

## Definiciones de AFND :

1. AFND =  $(\Sigma, Q, f, q_0, F)$ , donde

- $f: Q \times (\Sigma \cup \lambda) \rightarrow Q$  es No determinista,  
es decir, por ejemplo:  $f(p, a) = \{q, r\}$  y  $f(p, \lambda) = \{q, r\}$

2. AFND =  $(\Sigma, Q, f, q_0, F, T)$ , donde

- $f: Q \times \Sigma \rightarrow P(Q)$ : conjunto de las partes de  $Q$
- $T$ : Relación definida sobre pares de elementos de  $Q$ .

$pTq = (p, q) \in T$  si está definida la transición  $f(p, \lambda) = q$

Nota: "T" es la definición formal de la transición  $\lambda$ .

[ 51 ]



# Autómatas Finitos No Deterministas

Ejemplo: Sea el AFND siguiente:

$A = (\{a, b\}, \{p, q, r, s\}, f, p, \{p, s\}, T = \{(q, s), (r, r), (r, s), (s, r)\})$  donde  $f$ :

$f(p, a) = \{q\}$

$f(p, b) = \{\}$

$f(q, a) = \{p, r, s\}$

$f(q, b) = \{p, r\}$

$f(r, a) = \{\}$

$f(r, b) = \{p, s\}$

$f(s, a) = \{\}$

$f(s, b) = \{\}$

La tabla de transiciones es

	a	b	$\lambda$
$\rightarrow *p$	q		
q	{p, r, s}	p, r	s
r		p, s	r, s
*s			r

[ 52 ]



## AFNDs. Función de Transición extendida a palabras

- Se define a partir de  $f$ , una función de transición  $f''$ , que actúa sobre palabras de  $\Sigma^*$ ;

$f''$  es la función de transición sobre palabras.

- Es una aplicación:  $f'': Q \times \Sigma^* \rightarrow P(Q)$ . Donde :

$$1. f''(q, \lambda) = \{p / q T^* p \ \forall q \in Q\} \quad (T^* \text{ se define más adelante})$$

donde se cumple que  $q \in f'(q, \lambda)$

$$2. \text{ sea } x = a_1 a_2 a_3 \dots a_n, \ n > 0$$

$$f''(q, x) = \{p / p \text{ es accesible desde } q \text{ por medio de la palabra } \lambda^* a_1 \lambda^* a_2 \lambda^* a_3 \lambda^* \dots \lambda^* a_n \lambda^* \ \forall q \in Q\}$$

es idéntica a  $x$

Lectura recomendada: Apartado 3.3.4 del primer libro de la bibliografía básica

[ 53 ]



## AFNDs. Función de Transición extendida a palabras

### Calculo de $T^*$

Sea AFND =  $(\Sigma, Q, f, q_0, F, T)$ .

- Para calcular  $f''$  es necesario extender las transiciones con una  $\lambda$  a  $\lambda^*$ , es decir calcular  $T^*$  del AFND =  $(\Sigma, Q, f, q_0, F, \underline{T})$
- Para ello existe el método formal de las matrices booleanas, o el método de la matriz de pares (estado, estado).

[ 54 ]



## AFNDs. Función de Transición extendida a palabras

### Calculo de $T^*$ . Método de la matriz de pares de estados

- Se construye una matriz con tantas filas como estados.
- En la 1ª columna se coloca el par correspondiente al estado en cuestión, es decir, por ej.  $(p,p)$  puesto que cada estado es accesible desde sí mismo.
- En las columnas siguientes se añaden las transiciones  $\lambda$  definidas en el AFND, considerando si el hecho de añadirlas permite extender alguna transición más.
  - Pej. Si existe la transición  $\lambda (q,r)$  y se añade la transición  $\lambda (r,s)$ , habrá que añadir asimismo, la transición  $(q,s)$ .
- Cuando no sea posible añadir ningún par más, se habrá terminado  $T^*$

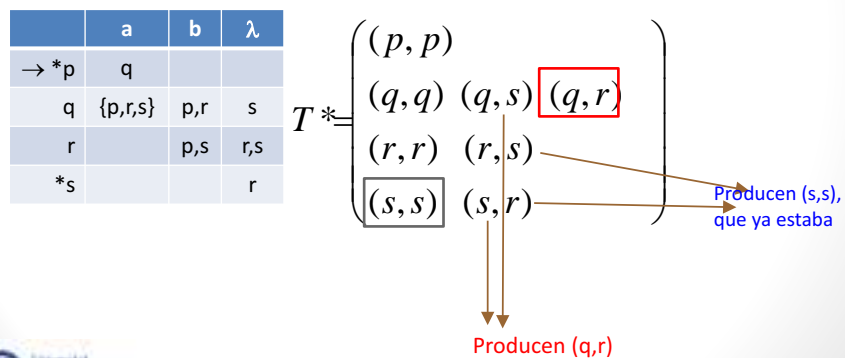
[ 55 ]



## AFNDs. Función de Transición extendida a palabras

### Calculo de $T^*$ . Ejemplo 1:

- Sea el AFND: A, definido anteriormente donde  $T = \{(q,s), (r,r), (r,s), (s,r)\}$ . Se trata de calcular  $T^*$



[ 56 ]



## AFNDs. Función de Transición extendida a palabras

### Calculo de $T^*$ . Ejemplo 2:

- Se extiende la tabla de transición anterior para contener  $T^*$ , insertando una nueva columna correspondiente a  $\lambda^*$

	a	b	<del><math>\lambda</math></del>	$\lambda^*$
$\rightarrow^*p$	q			p
q	p,r,s	p,r	<del>s</del>	q,s,r
r		p,s	<del>r,s</del>	r,s
$*s$			<del>r</del>	r,s

[ 57 ]



## AFNDs. Función de Transición extendida a palabras

### Calculo de $T^*$ . Ejemplo 3:

- Y ahora se calcula la tabla de transición correspondiente a  $f''$ , cambiando las transiciones con a por  $\lambda^*a\lambda^*$  y las de b por  $\lambda^*b\lambda^*$ .

	a	b	<del><math>\lambda</math></del>	$\lambda^*$		$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	q			p	$\rightarrow^*p$	q,r,s	$\Phi$
q	p,r,s	p,r	<del>s</del>	q,s,r	q	p,r,s	p,r,s
r		p,s	<del>r,s</del>	r,s	r	$\Phi$	p,r,s
$*s$			<del>r</del>	r,s	$*s$	$\Phi$	p,r,s

[ 58 ]



## AFND. Lenguaje aceptado por un AFND

- Una palabra  $x \in \Sigma^*$  es aceptada por un AFND si:
  - $f'(q_0, x)$  y  $F$  tienen al menos un elemento común, es decir, que  $f'(q_0, x)$  contiene al menos un estado final.
- El conjunto de todas las palabras aceptadas por un AFND es el lenguaje aceptado por ese AFND.

Formalmente:

$$L_{AFND} = \{x / x \in \Sigma^* \vee \exists q_0 \rightarrow F\} = \{x / x \in \Sigma^* \vee f'(q_0, x) \cap F \neq \emptyset\}$$

[ 59 ]



## AFND. Lenguaje aceptado por un AFND

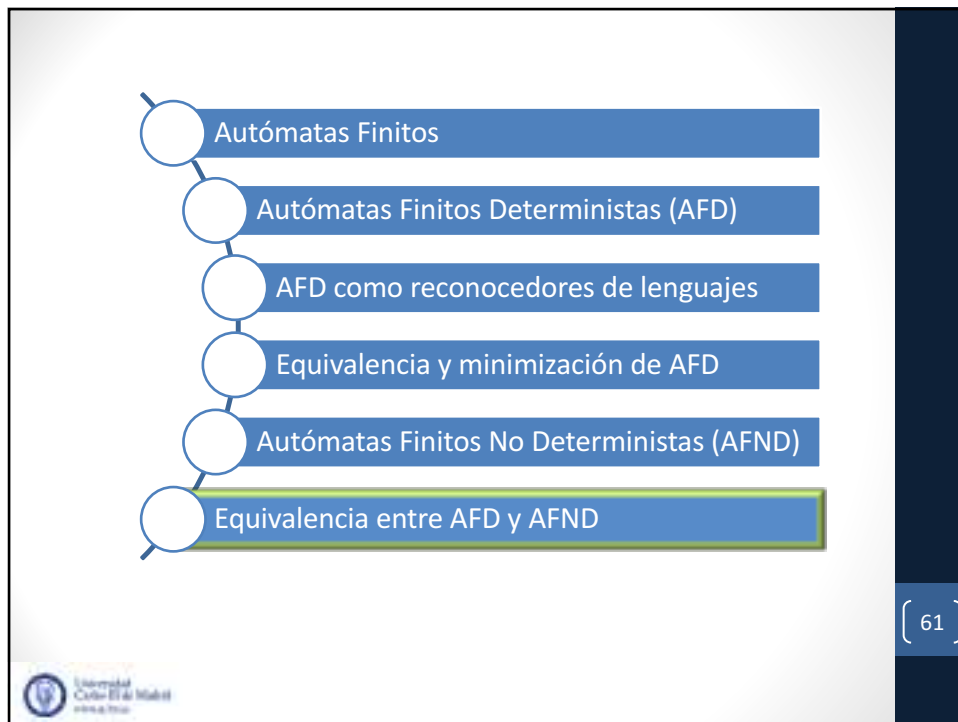
- Al ser un AFND, desde  $q_0$  puede haber más de un camino para la palabra " $x$ ", y " **$x$** " es aceptada sólo con que uno de los caminos lleve a un estado final.
- Además:
 

$\lambda \in L_{AFND}$  si :

  - $q_0 \in F$  ó
  - $\exists$  un estado final,  $q \in F$ , tal que está en relación  $T^*$  con  $q_0$  ( $q_0 T^* q$ )

[ 60 ]





## AFD equivalente a un AFND

- Dado un AFND siempre es posible encontrar un AFD que reconozca el mismo lenguaje:
  - El conjunto de los  $L_{AFND}$  = al conjunto de los  $L_{AFD}$ .
  - Un AFND no es más potente que un AFD, sino que un AFD es un caso particular de AFND.

**Paso de AFND a AFD:**

- Sea el AFND  $A = (\Sigma, Q, f, q_0, F, T)$ .
- Se define a partir de A el AFD B, donde:
  - $B = (\Sigma, Q', f', q_0', F')$ , tal que:
  - $Q' = P(Q)$  conjunto de las partes de Q que incluye a Q y a  $\emptyset$ .
  - $q_0' = f'(q_0, \lambda)$  ( $f'$  extensión a palabra de f, i.e. todos los estados que tengan relación  $T^*$  con  $q_0$ ).
  - $F' = \{C / C \in Q' \text{ y } \exists q \in C / q \in F\}$
  - $f'(C, a) = \{C' / C' = \bigcup_{q \in C} f(q, a)\}$

62

## AFD equivalente a un AFND

- EXPLICACIÓN Paso de AFND a AFD:

1. Calculamos  $\lambda^*$  (usando  $T^*$ )
2. Calculamos  $\lambda^*a \lambda^* \forall a \in \Sigma$
3.  $\lambda^*a \lambda^* = a$
4. Transformamos los caminos múltiples en estados combinados:  
p, q, r pasa a ser {pqr}
5. Construimos en AFD conexo considerando que las transiciones de {pqr} son las resultantes de hacer la unión de las transiciones de cada uno de los estados que lo forman.
6. Será estado inicial  $q_0$  o  $q_0' = f'(q_0, \lambda)$  ( $f'$  extensión a palabra de  $f$ , i.e. todos los estados que tengan relación  $T^*$  con  $q_0$ ).
7. Será estado final TODO estado combinado que contenga a un estado final o aquellos finales que estén conectados.

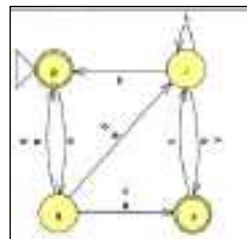
[ 63 ]



## AFD equivalente a un AFND. Ejemplo

- Obtener el AFD correspondiente al siguiente AFND

	a	b	$\lambda$
$\rightarrow^* p$	q		
q	p,r,s	p,r	s
r		p,s	r,s
$*s$			r



- Pasos:

1. Eliminar transiciones  $\lambda$ 
  - a) Determinar  $\lambda^*$  (el cierre de las transiciones  $\lambda$ ,  $T^*$ )
  - b) Obtener la tabla sin transiciones  $\lambda$
2. Aplicar algoritmo de creación de nuevos estados que pertenecen a  $P(Q)$ , añadiendo su transiciones.

[ 64 ]





## AFD equivalente a un AFND. Ejemplo

### 1. Eliminar transiciones $\lambda$

- a) Determinar  $\lambda^*$  (el cierre de las transiciones  $\lambda$ ) a partir de la tabla de transiciones.

	a	b	$\lambda$
$\rightarrow^*p$	q		
q	p,r,s	p,r	s
r		p,s	r,s
$*s$			r



	a	b	<del><math>\lambda</math></del>	$\lambda^*$
$\rightarrow^*p$	q			<b>p</b>
q	p,r,s	p,r	<del>s</del>	<b>q,s,r</b>
r		p,s	<del>r,s</del>	<b>r,s</b>
$*s$			<del>r</del>	<b>s,r</b>

[ 65 ]



## AFD equivalente a un AFND. Ejemplo

### 1. Eliminar transiciones $\lambda$

- a) Determinar  $\lambda^*$  (el cierre de las transiciones  $\lambda$ )

	a	b	<del><math>\lambda</math></del>	$\lambda^*$
$\rightarrow^*p$	q			<b>p</b>
q	p,r,s	p,r	<del>s</del>	<b>q,r,s</b>
r		p,s	<del>r,s</del>	<b>r,s</b>
$*s$			<del>r</del>	<b>r,s</b>

- b) Obtener la tabla sin transiciones  $\lambda$   
(transiciones con entrada  $\lambda^*a\lambda^*$ , para cada elemento, a, del alfabeto  $\Sigma$ )

	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	q,r,s	$\emptyset$
q	p,r,s	p,r,s
r	$\emptyset$	p,r,s
$*s$	$\emptyset$	p,r,s

[ 66 ]



## AFD equivalente a un AFND. Ejemplo

2. Aplicar algoritmo de creación de nuevos estados que pertenecen a  $P(Q)$ , añadiendo su transiciones.

	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	$q,r,s$	$\emptyset$
$q$	$p,r,s$	$p,r,s$
$r$	$\emptyset$	$p,r,s$
$*s$	$\emptyset$	$p,r,s$

	a	b
$\rightarrow^*p$	$\{q,r,s\}$	$\emptyset$
$q$	$\{p,r,s\}$	$\{p,r,s\}$
$r$	$\emptyset$	$\{p,r,s\}$
$*s$	$\emptyset$	$\{p,r,s\}$
	$\{q,r,s\}$	$\{p,r,s\} \cup \emptyset \cup \emptyset$
		$\{p,r,s\} \cup \{p,r,s\} \cup \{p,r,s\}$

	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	$\{q,r,s\}$	$\emptyset$
$q$	$\{p,r,s\}$	$\{p,r,s\}$
$r$	$\emptyset$	$\{p,r,s\}$
$*s$	$\emptyset$	$\{p,r,s\}$
	$\{q,r,s\}$	$\{p,r,s\}$



[ 67 ]

## AFD equivalente a un AFND. Ejemplo

2. Aplicar algoritmo de creación de nuevos estados que pertenecen a  $P(Q)$ , añadiendo sus transiciones.

	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	$q,r,s$	$\emptyset$
$q$	$p,r,s$	$p,r,s$
$r$	$\emptyset$	$p,r,s$
$*s$	$\emptyset$	$p,r,s$

	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	$\{q,r,s\}$	$\emptyset$
$q$	$\{p,r,s\}$	$\{p,r,s\}$
$r$	$\emptyset$	$\{p,r,s\}$
$*s$	$\emptyset$	$\{p,r,s\}$
	$\{q,r,s\}$	$\{p,r,s\}$

	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	$\{q,r,s\}$	$\emptyset$
$q$	$\{p,r,s\}$	$\{p,r,s\}$
$r$	$\emptyset$	$\{p,r,s\}$
$*s$	$\emptyset$	$\{p,r,s\}$
	$\{q,r,s\}$	$\{p,r,s\}$

	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	$\{q,r,s\}$	$\emptyset$
$q$	$\{p,r,s\}$	$\{p,r,s\}$
$r$	$\emptyset$	$\{p,r,s\}$
$*s$	$\emptyset$	$\{p,r,s\}$
$\{q,r,s\}$	$\{p,r,s\}$	$\{p,r,s\}$
$\{p,r,s\}$	$\{q,r,s\} \cup \emptyset \cup \emptyset$	$\emptyset \cup \{p,r,s\} \cup \{p,r,s\}$

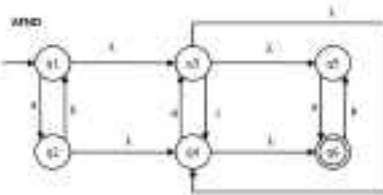
	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$
$\rightarrow^*p$	$\{q,r,s\}$	$\emptyset$
$\{q,r,s\}$	$\{p,r,s\}$	$\{p,r,s\}$
$\{p,r,s\}$	$\{q,r,s\}$	$\{p,r,s\}$



[ 68 ]

## Ejercicio 12 – Parte 3

Indica el grafo del Autómata Finito Determinista que se corresponde con el siguiente AFND:



	a	b	c	d	λ
->q1	q2				q3
q2		q1			q4
q3			q4		{q4, q5}
q4				q3	q6
q5	q6				
q6*		q5			

## Ejercicio 12 – Parte 3

	a	b	c	d	λ	λ*	λ*a λ*	λ*b λ*	λ*c λ*	λ*d λ*
->q1	q2				q3	{q1,q3,q4,q5,q6}	{q2,q4,q6}	{q5}	{q4,q6}	{q3,q4,q5,q6}
q2		q1			q4	{q2,q4,q6}	--	{q1,q3,q4,q5,q6}	--	{q3,q4,q5,q6}
q3			q4		{q4,q5}	{q3,q4,q5,q6}	{q6}	{q5}	{q4,q6}	{q3,q4,q5,q6}
q4				q3	q6	{q4,q6}	--	{q5}	--	{q3,q4,q5,q6}
q5	q6					{q5}	{q6}	--	--	--
q6*		q5				{q6}	--	{q5}	--	--

## Ejercicio 12 – Parte 3

	$\lambda^*$	$\lambda^*a\lambda^*$	$\lambda^*b\lambda^*$	$\lambda^*c\lambda^*$	$\lambda^*d\lambda^*$
$\rightarrow q1$	$\{q1, q3, q4, q5, q6\}$	$\{q2, q4, q6\}$	$\{q5\}$	$\{q4, q6\}$	$\{q3, q4, q5, q6\}$
$q2$	$\{q2, q4, q6\}$	--	$\{q1, q3, q4, q5, q6\}$	--	$\{q3, q4, q5, q6\}$
$q3$	$\{q3, q4, q5, q6\}$	$\{q6\}$	$\{q5\}$	$\{q4, q6\}$	$\{q3, q4, q5, q6\}$
$q4$	$\{q4, q6\}$	--	$\{q5\}$	--	$\{q3, q4, q5, q6\}$
$q5$	$\{q5\}$	$\{q6\}$	--	--	--
$q6^*$	$\{q6\}$	--	$\{q5\}$	--	--

[ 71 ]



## Ejercicio 12 – Parte 3

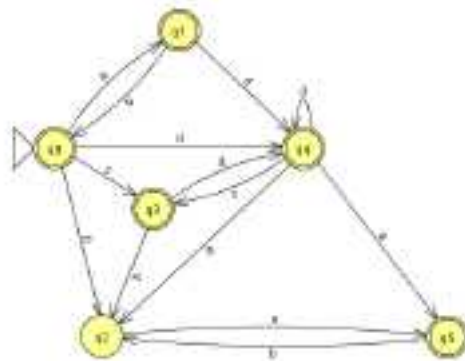
	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>
$\rightarrow \{q1, q3, q4, q5, q6\}^*$	$\{q2, q4, q6\}$	$\{q5\}$	$\{q4, q6\}$	$\{q3, q4, q5, q6\}$
$\{q2, q4, q6\}^*$	$\phi$	$\{q1, q3, q4, q5, q6\}$	$\phi$	$\{q3, q4, q5, q6\}$
$\{q5\}$	$\{q6\}$	$\phi$	$\phi$	$\phi$
$\{q4, q6\}^*$	$\phi$	$\{q5\}$	$\phi$	$\{q3, q4, q5, q6\}$
$\{q3, q4, q5, q6\}^*$	$\{q6\}$	$\{q5\}$	$\{q4, q6\}$	$\{q3, q4, q5, q6\}$
$\{q6\}^*$	$\phi$	$\{q5\}$	$\phi$	$\phi$
$\phi$	$\phi$	$\phi$	$\phi$	$\phi$

[ 72 ]



## Ejercicio 12 – Parte 3

Solución:



-> {q1,q3,q4,q5,q6}\*=Q0

{q2,q4,q6}\*=Q1

{q5}=Q2

{q4,q6}\*=Q3

{q3,q4,q5,q6}\*=Q4

{q6}\*=Q5

[ 73 ]



Por simplicidad en el dibujo, el estado sumidero no se ha representado

## 3. Autómatas Finitos

Grado Ingeniería Informática

Teoría de Autómatas y Lenguajes Formales



Universidad  
Carlos III de Madrid  
www.uc3m.es