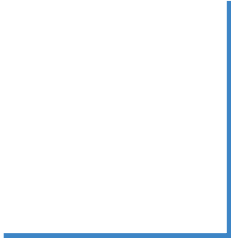




# Teoría de Lenguajes

AFND-epsilon  
ER  $\rightarrow$  AF



# Autómata Finito No Determinista - Epsilon

# Autómata Finito No Determinista - Epsilon

Un AFND- $\epsilon$  es una máquina de estados que se puede representar por la siguiente quintupla

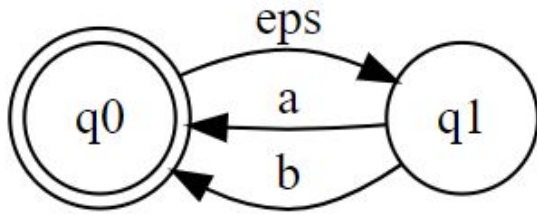
$M : (Q, \Sigma, \delta, q_0, F)$  donde:

- $Q$ : conjunto de estados
- $\Sigma$ : alfabeto
- $\delta$ : función de transición /  $\delta: Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$
- $q_0$ : estado inicial /  $q_0 \in Q$
- $F$ : conjunto de estados finales (aceptores) /  $F \subseteq Q$

# Autómata Finito No Determinista - $\epsilon$

Ejemplos dado  $\{a,b\}$

1)

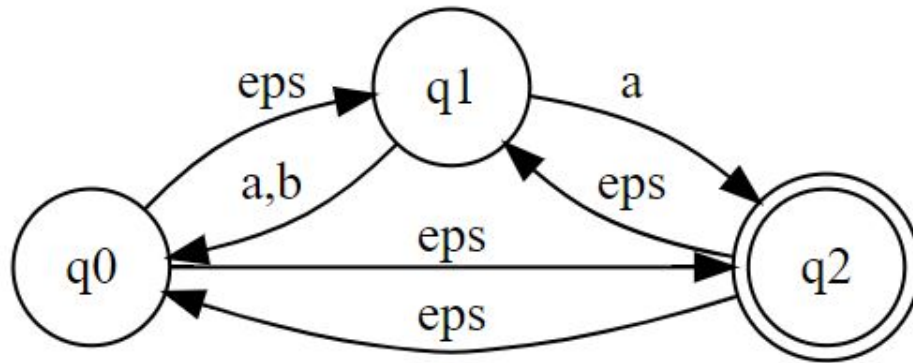


$\delta$	a	b	$\epsilon$
q0	$\emptyset$	$\emptyset$	{q1}
q1	{q0}	{q0}	$\emptyset$

# Autómata Finito No Determinista - $\epsilon$

Ejemplos dado  $\{a,b\}$

2)



$\delta$	a	b	$\epsilon$
q0	$\emptyset$	$\emptyset$	{q1,q2}
q1	{q0,q2}	{q0}	$\emptyset$
q2	$\emptyset$	$\emptyset$	{q0,q1}

# Epsilon-clausura

Definición:

$\epsilon$ -clausura( $q$ ) =  $\{ p / p \in Q \text{ y existe un camino de } q \text{ a } p \text{ a través de arcos}$

etiquetados exclusivamente con  $\epsilon$  }  $q \in Q$

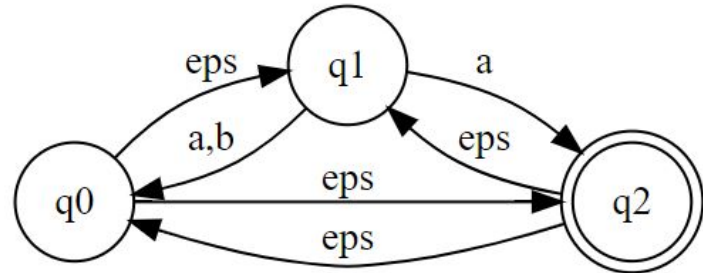
$$q \in \epsilon\text{-clausura}(q)$$

En el ejemplo ...

$$\epsilon\text{-clausura}(q_0) = \{q_0, q_1, q_2\}$$

$$\epsilon\text{-clausura}(q_1) = \{q_1\}$$

$$\epsilon\text{-clausura}(q_2) = \{q_0, q_1, q_2\}$$



# Epsilon-clausura

Definición:

$$\varepsilon\text{-clausura}(P) = \bigcup_{p \in P} \varepsilon\text{-clausura}(p) \quad P \subseteq Q$$

Propiedad:

$$\varepsilon\text{-clausura}(\varepsilon\text{-clausura}(q)) = \varepsilon\text{-clausura}(q) \quad q \in Q$$

# Autómata Finito No Determinista - $\epsilon$

Extensión para manejar strings

$$\delta^\wedge: Q \times \Sigma^* \rightarrow 2^Q$$

$$\delta^\wedge(q, \epsilon) = \epsilon\text{-clausura}(q) \quad \forall q \in Q$$

$$\delta^\wedge(q, wa) = \epsilon\text{-clausura}(\delta^\sim(\delta^\wedge(q, w), a)) \quad \forall q \in Q \quad a \in \Sigma \quad w \in \Sigma^*$$



# Autómata Finito No Determinista - $\epsilon$

Definición:

Lenguaje  $L$  aceptado por un AFND-  $\epsilon$   $M : (Q, \Sigma, \delta, q_0, F)$

$$L = L(M) = \{ x \in \Sigma^* / \delta^*(q_0, x) \cap F \neq \emptyset \}$$

# Lenguaje Regular

Definición (4):

Un Lenguaje  $L$  es Regular si es aceptado por un AFND-  $\epsilon$   $M:(Q,\Sigma,\delta,q_0,F)$

otra forma

Un Lenguaje  $L$  es Regular si existe un AFND-  $\epsilon$   $M:(Q,\Sigma,\delta,q_0,F)$  que lo reconoce

$$L = L(M)$$

# Equivalencia de AFND y AFND- $\epsilon$

1) AFND  $\Rightarrow$  AFND- $\epsilon$

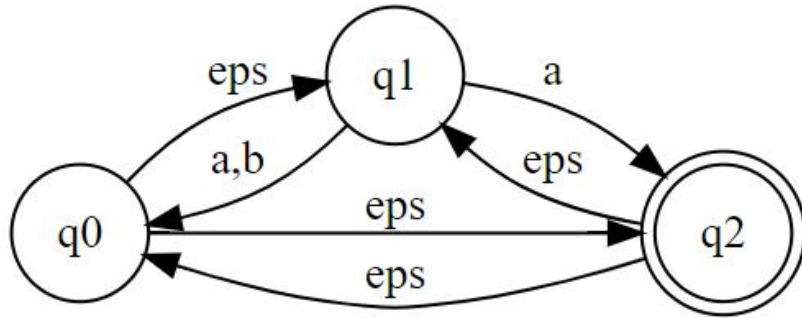
2) AFND- $\epsilon \Rightarrow$  AFND ::  $M:(Q,\Sigma,\delta,q_0,F) \Rightarrow M':(Q',\Sigma,\delta',q'_0,F')$

Algoritmo:

- $Q' = Q$
- $q'_0 = q_0$
- $F' = F$  o  $F \cup \{q_0\}$  (si  $q_0 \notin F$  y  $\epsilon$ -clausura( $q_0$ )  $\cap F \neq \emptyset$ )
- $\delta'(q,a) = \epsilon$ -clausura( $\delta^{\sim}(\epsilon$ -clausura( $q$ ), $a$ ))  $\forall q \in Q \quad a \in \Sigma$

Entonces  $\delta'^{\wedge}(q'_0,x) = \delta^{\wedge}(q_0,x)$

# Aplicación



Lo primero que se recomienda hacer es calcular todas las  $\epsilon$ -clausura( $q$ )  $\forall q \in Q$

$$\epsilon\text{-clausura}(q_0) = \{q_0, q_1, q_2\}$$

$$\epsilon\text{-clausura}(q_1) = \{q_1\}$$

$$\epsilon\text{-clausura}(q_2) = \{q_0, q_1, q_2\}$$

Luego aplicar  $\delta'(q,a) = \epsilon\text{-clausura}(\delta^{\sim}(\epsilon\text{-clausura}(q), a)) \quad \forall q \in Q \quad a \in \Sigma$

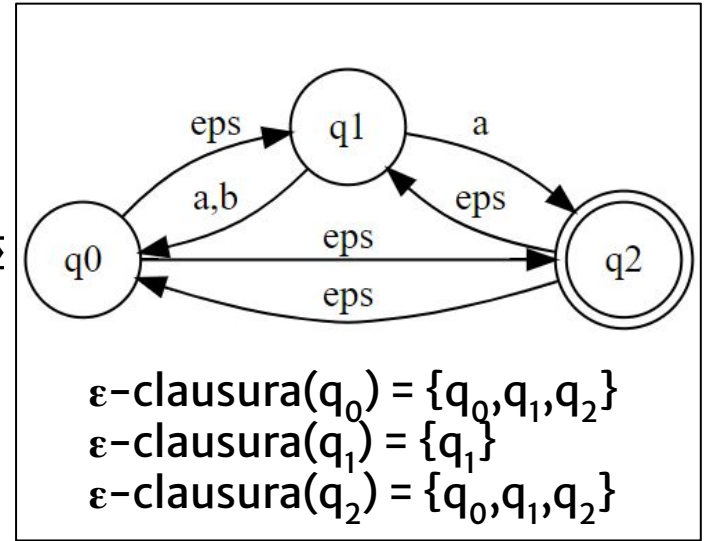
# Aplicación (cont.)

$$\delta'(q,a) = \varepsilon\text{-clausura}(\delta^{\sim}(\varepsilon\text{-clausura}(q),a)) \quad \forall q \in Q \quad a \in \Sigma$$

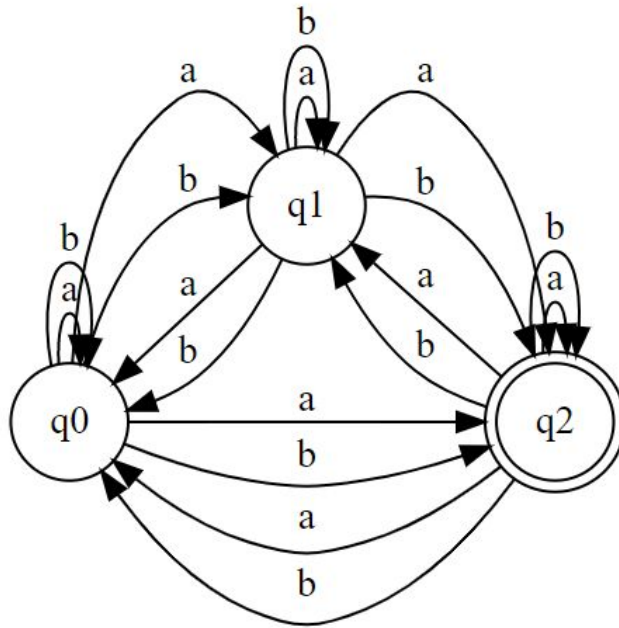
$$\begin{aligned}\delta'(q_0,a) &= \varepsilon\text{-clausura}(\delta^{\sim}(\{q_0,q_1,q_2\},a)) = \\ &= \varepsilon\text{-clausura}(\{\delta(q_0,a) \cup \{\delta(q_1,a)\} \cup \{\delta(q_2,a)\}\}) = \\ &= \varepsilon\text{-clausura}(\emptyset \cup \{\delta(q_1,a)\} \cup \emptyset) = \\ &= \varepsilon\text{-clausura}(\{q_0,q_2\}) = \varepsilon\text{-clausura}(q_0) \cup \varepsilon\text{-clausura}(q_2) = \{q_0,q_1,q_2\}\end{aligned}$$

$$\begin{aligned}\delta'(q_0,b) &= \varepsilon\text{-clausura}(\delta^{\sim}(\{q_0,q_1,q_2\},b)) = \\ &= \varepsilon\text{-clausura}(\{q_0\}) = \{q_0,q_1,q_2\}\end{aligned}$$

...



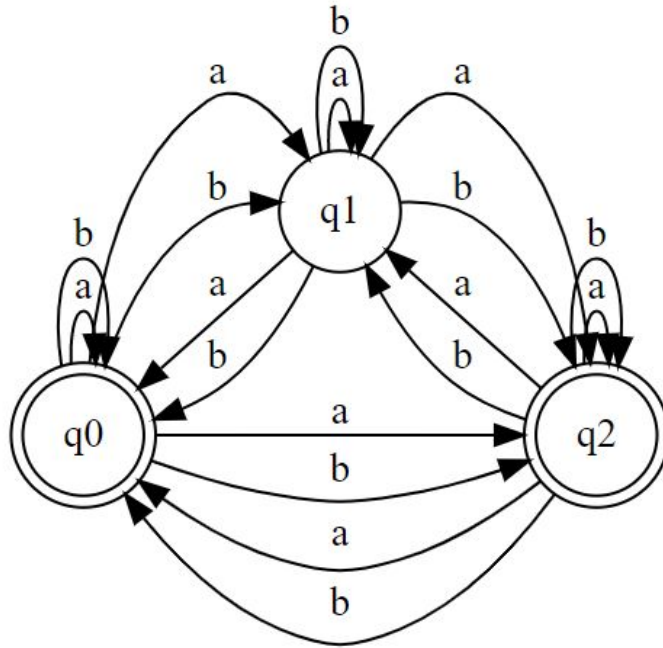
# Aplicación (cont.)



$\delta$	a	b
q0	{q0,q1,q2}	{q0,q1,q2}
q1	{q0,q1,q2}	{q0,q1,q2}
q2	{q0,q1,q2}	{q0,q1,q2}

¿Está bien?

# Aplicación (cont.)

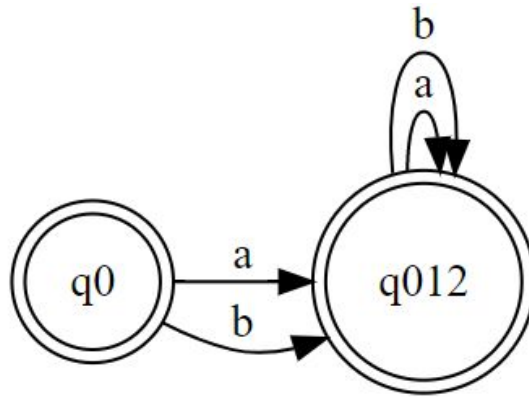


$\delta$	a	b
q0	{q0,q1,q2}	{q0,q1,q2}
q1	{q0,q1,q2}	{q0,q1,q2}
q2	{q0,q1,q2}	{q0,q1,q2}

Hay que agregar  $q_0$  como final

# Aplicación (cont.)

AFND  $\rightarrow$  AFD



$\delta$	a	b
q0	{q0,q1,q2}	{q0,q1,q2}
q1	{q0,q1,q2}	{q0,q1,q2}
q2	{q0,q1,q2}	{q0,q1,q2}



$\delta'$	a	b
[q0]	[q012]	[q012]
[q012]	[q012]	[q012]



# Equivalencia de AF y ER

AFD



AFND



AFND- $\epsilon$

ER

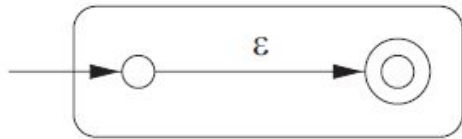


# Equivalencia de AF y ER

## 1) ER $\Rightarrow$ AFND- $\epsilon$

Se demuestra encontrando un AFND-  $\epsilon$  a partir de la definición de ER, con un único estado final y del cual NO salen transiciones

Paso Base:



(a)

$$r = \epsilon$$

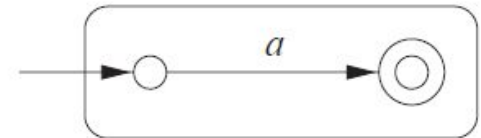
$$L(r) = \{\epsilon\}$$



(b)

$$r = \emptyset$$

$$L(r) = \emptyset$$



(c)

$$r = a \quad \forall a \in \Sigma$$

$$L(r) = \{a\}$$

# Equivalencia de AF y ER (cont.)

Paso Inductivo:

Existen  $r$  y  $s$  ER / para cada una existe un AFND- $\epsilon$  de forma que

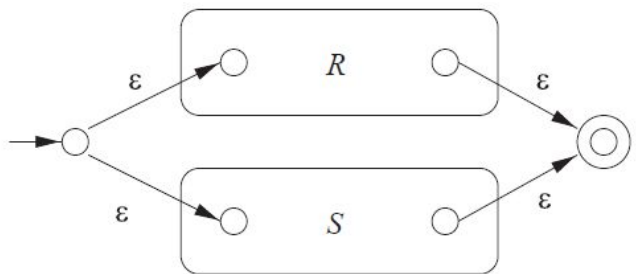
$$M_1 : (Q_1, \Sigma_1, \delta_1, q_1, \{q_{f1}\}) / L(r) = L(M_1)$$

$$M_2 : (Q_2, \Sigma_2, \delta_2, q_2, \{q_{f2}\}) / L(s) = L(M_2)$$

Asumimos  $Q_1 \cap Q_2 = \emptyset$

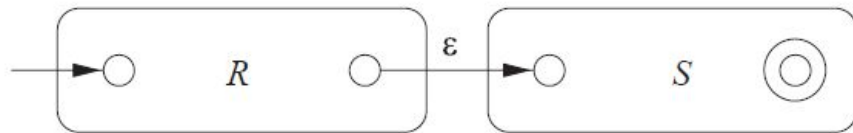
Se construyen los siguientes autómatas...

# Equivalencia de AF y ER (cont.)



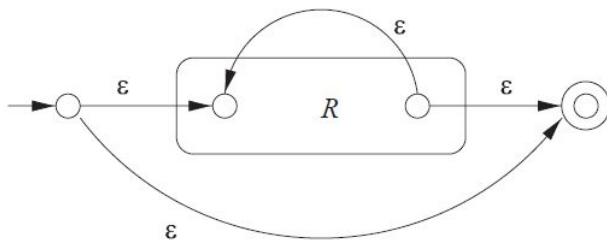
(a)

$$e = r \mid s$$
$$L(e) = R \cup S$$



(b)

$$e = r \cdot s$$
$$L(e) = R \cdot S$$

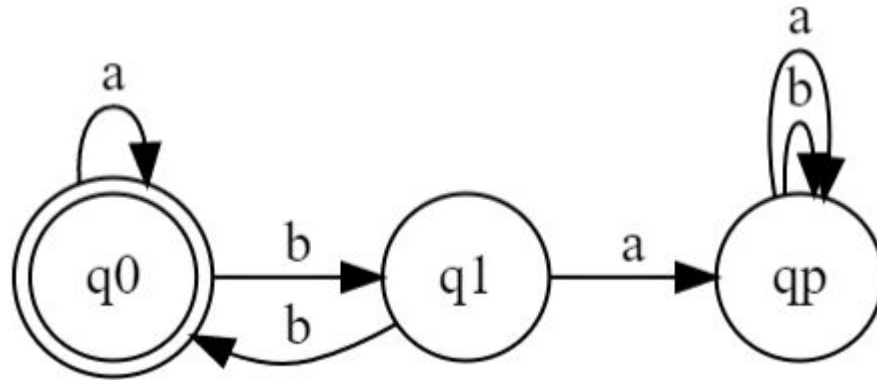


(c)

$$e = r^*$$
$$L(e) = R^*$$

# Aplicación

Construir un AFD para el lenguaje dado por  $(a|bb)^*$



# Ejemplo

**Lex:** generador de analizadores lexicográficos (escrito en C)

```
%%  
  /* Reglas */  
  
letra      [a-zA-Z]  
digito     [0-9]  
ident      letra(letra | digito)*  
%%  
  
if      {return (IF)};  
then   {return (THEN)};  
...  
{ident} {yyival = yytext[]; return (IDENTIF)}  
...  
%%  
  
main ()  
{  
    yyex();  
}
```