Teoría de Lenguajes

AFND-epsilon
ER → AF

Autómata Finito No Determinista - Epsilon

Autómata Finito No Determinista - Epsilon

Un AFND-ε es una máquina de estados que se puede representar por la siguiente quíntupla

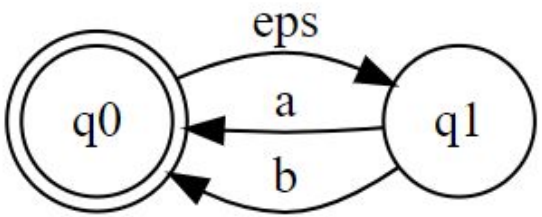
$$M : (Q, \Sigma, \delta, q_o, F) \text{ donde:}$$

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

Autómata Finito No Determinista - ε

Ejemplos dado {a,b}

1)

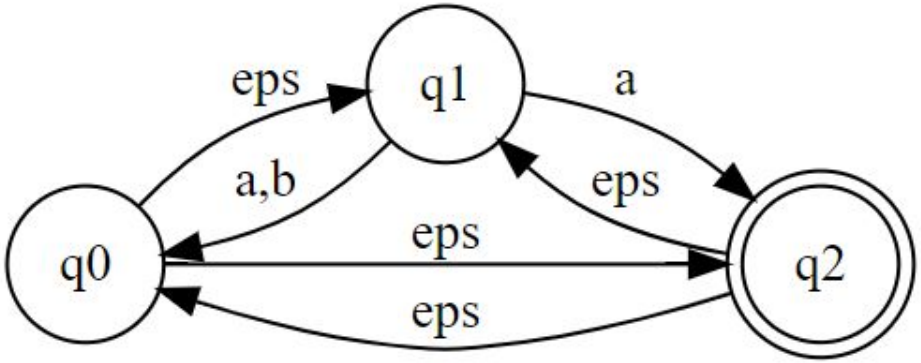


δ	a	b	ε
q0	Ø	Ø	{q1}
q1	{q0}	{q0}	Ø

Autómata Finito No Determinista - ε

Ejemplos dado {a,b}

2)



δ	a	b	ε
q0	Ø	Ø	{q1,q2}
q1	{q0,q2}	{q0}	Ø
q2	Ø	Ø	{q0,q1}

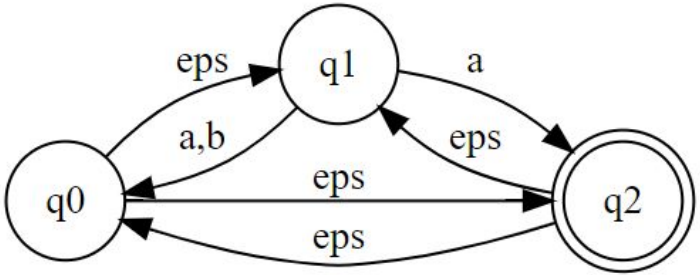
Epsilon-clausura

Definición:

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

~~etiqueta~~tados exclusivamente con ε } $q \in Q$

$q \in \varepsilon$-clausura(q)



En el ejemplo anterior…

ε-clausura(q_0) = {q_0,q_1,q_2}

ε-clausura(q_1) = {q_1}

ε-clausura(q_2) = {q_0,q_1,q_2}

Epsilon-clausura

Definición:

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

Propiedad:

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

Autómata Finito No Determinista - ε

Extensión para manejar strings

$\delta\hat{}: Q \times \Sigma^* \rightarrow 2^Q$

$\delta\hat{}(q,\varepsilon) = \varepsilon\text{-clausura}(q) \quad \forall\ q \in Q$

$\delta\hat{}(q,wa) = \varepsilon\text{-clausura}(\delta\tilde{}(\delta\hat{}(q,w),a)) \quad \forall\ q \in Q \quad a \in \Sigma \quad w \in \Sigma^*$

Autómata Finito No Determinista - ε

Definición:

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

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

Lenguaje Regular

Definición (4):

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

otra forma

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

$$L = L(M)$$

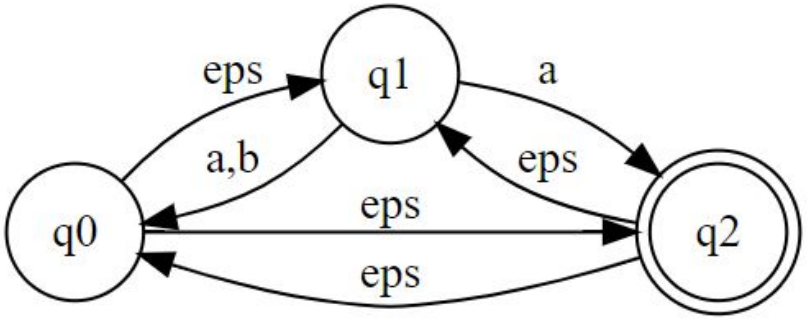
Equivalencia de AFND y AFND- ε

1) AFND $\Rightarrow$ AFND- ε
2) AFND- ε $\Rightarrow$ 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 ε-clausura$(q_0) \cap F \neq \varnothing$)
   - $\delta'(q,a) = \varepsilon$-clausura $(\delta^{\sim}(\varepsilon$-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 ε-clausura(q) $\forall$ q $\in$ Q

ε-clausura(q_0) = {q_0,q_1,q_2}
ε-clausura(q_1) = {q_1}
ε-clausura(q_2) = {q_0,q_1,q_2}

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

Aplicación (cont.)

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

$\delta'(q_0,a) = \varepsilon\text{-clausura}(\delta^{\sim}(\{q_0,q_1,q_2\},a)) =$

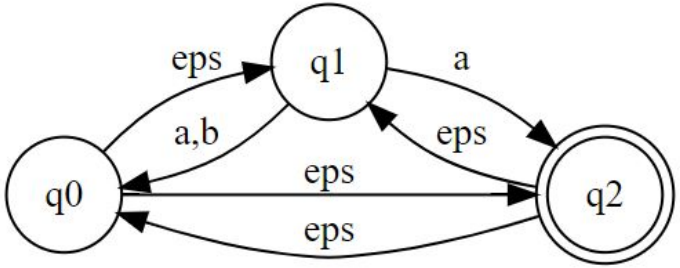
$= \varepsilon\text{-clausura}(\{\delta(q_0,a) \cup \{\delta(q_1,a) \cup \{\delta(q_2,a)\}) =$

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

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

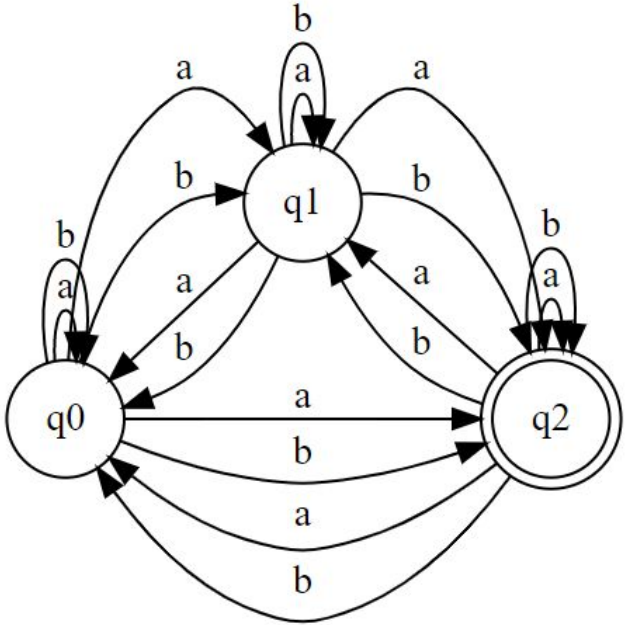
$= \varepsilon\text{-clausura}(\{q_0\}) = \{q_0,q_1,q_2\}$

...



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

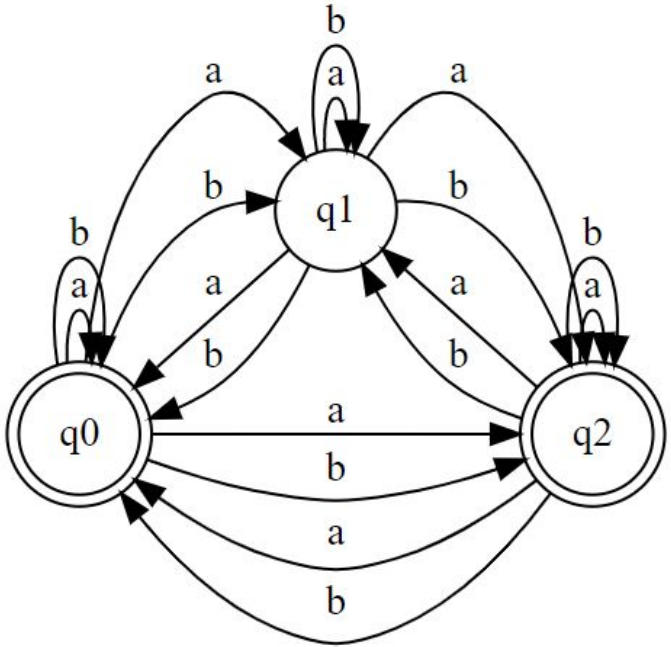
Aplicación (cont.)



δ	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.)

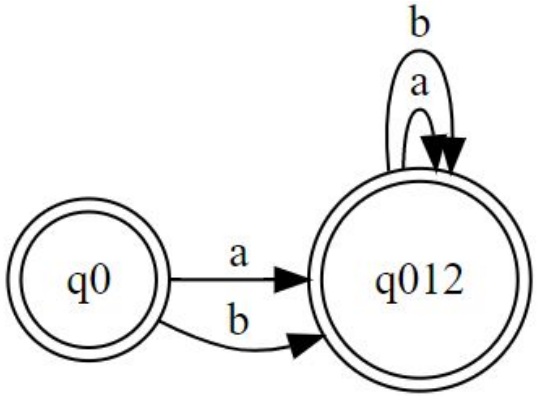


δ	a	b
q0	{q0,q1,q2}	{q0,q1,q2}
q1	{q0,q1,q2}	{q0,q1,q2}
q2	{q0,q1,q2}	{q0,q1,q2}

Hay que agregar q_0 como final

Aplicación (cont.)

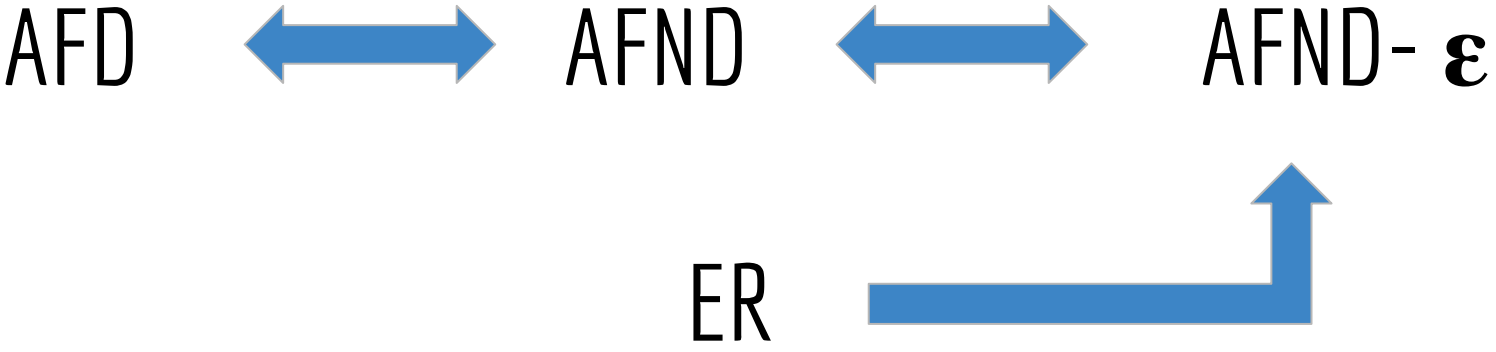
AFND → AFD



δ	a	b
q0	{q0,q1,q2}	{q0,q1,q2}
q1	{q0,q1,q2}	{q0,q1,q2}
q2	{q0,q1,q2}	{q0,q1,q2}

δ’	a	b
[q0]	[q012]	[q012]
[q012]	[q012]	[q012]

Equivalencia de AF y ER

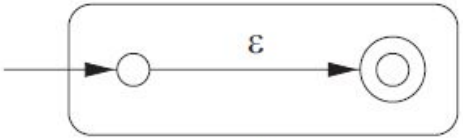


Equivalencia de AF y ER

1) ER ⇒ AFND- ε

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

Paso Base:



(a)

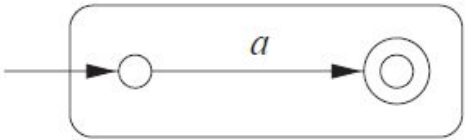
r = ε

L(r) = {ε}

(b)

r = ∅

L(r) = ∅



(c)

r = a ∀ a ∈ ∑

L(r) = {a}

Equivalencia de AF y ER (cont.)

Paso Inductivo:

Existen *r* y *s* ER / para cada una existe un AFND-ε de forma que

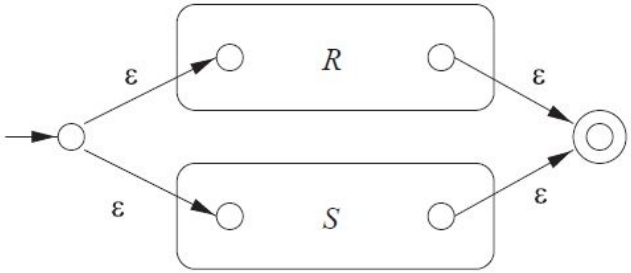
$$M_1 : (Q_1, \Sigma_1, \delta_1, q_1, \{q_{f1}\}) \;\; / \;\; L(r) = L(M_1)$$

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

Asumimos $Q_1 \cap Q_2 = \varnothing$

Se construyen los siguientes autómatas…

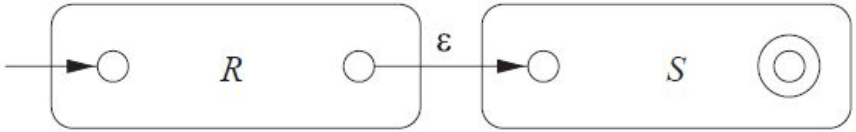
Equivalencia de AF y ER (cont.)



(a)

e = *r* | *s*

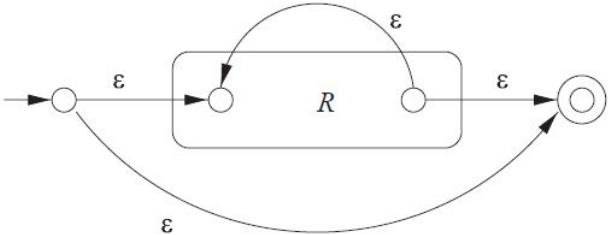
L(e) = R ∪ S



(b)

e = *r* . *s*

L(e) = R . S



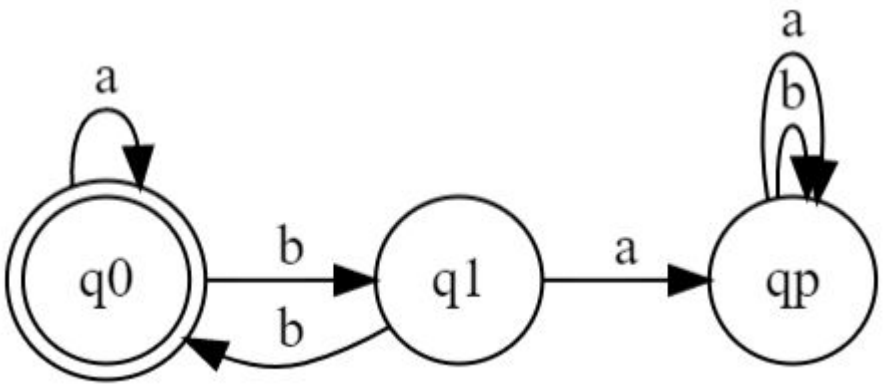
(c)

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} {yylval = yytext[]; return(IDENTIF)}
...
%%

main ()
{
        yyex();
}
```