



4. Propiedades de los Lenguajes Regulares

4.1. Lenguajes no regulares. Lema de bombeo.

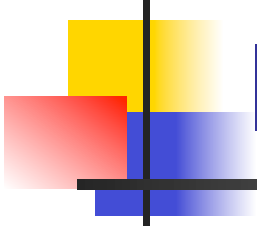
Fernando Rosa Velardo

Traducción y adaptación de transparencias de Ananth Kalyanaraman
(<http://www.eecs.wsu.edu/~ananth/>)



Algunos lenguajes no son regulares

- Un lenguaje es regular si podemos construir para él:
 - AFD o AFN o ε -AFN o expresión regular
- ¿Se puede construir un autómata para todos los lenguajes?
- Si probamos que ningún AFD acepta un lenguaje dado, entonces ese lenguaje NO es regular



¿Cómo se prueba que un lenguaje *no* es regular?

Si no se nos ocurre ningún autómeta...

- ¿Es que el lenguaje no es regular?
- ¿No hemos abordado el problema correctamente?
- ¿Cómo probamos sin lugar a dudas que no existe tal autómeta?


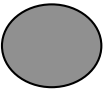


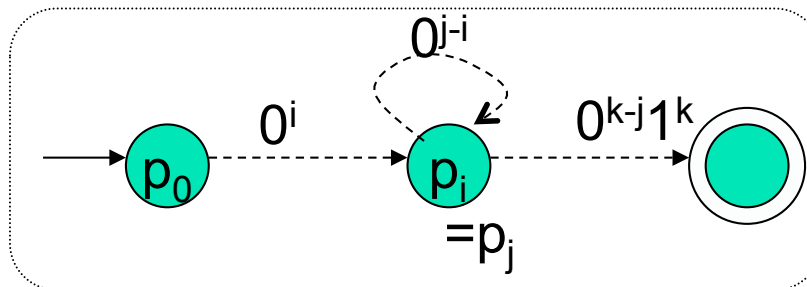
Ejemplo de lenguaje no regular

$$L = \{0^n 1^n \mid n \geq 0\}$$

- Hipótesis: *L no es regular*
- Idea intuitiva: ¿Cómo contamos en un autómata?
- Idea más formal:
 - Por contradicción, si L es regular entonces existe un AFD “A” para L. Sea k = número de estados de A.
 - Consideremos la palabra $w = 0^k 1^k$
 - A está en algún estado p_i tras consumir cada uno de los k+1 prefijos de 0^k : $\{\epsilon, 0^1, 0^2, \dots, 0^k\}$

Idea...

- Sea $\{p_0, p_1, \dots, p_k\}$ esa sucesión de estados
- El AFD sólo tiene k estados
- \implies al menos dos estados en esa secuencia son el mismo estado (principio del palomar:  + )
- \implies así que existen i, j con $i < j$ tales que $p_i = p_j$
- \implies el AFD también acepta $0^{k-(j-i)}1^k$
- \implies contradicción con el hecho de que el AFD es correcto





Lema de Bombeo para Lenguajes Regulares

Sea L un lenguaje regular

Entonces existe una constante n tal que toda palabra $w \in L$ con $|w| \geq n$, se puede descomponer en tres partes, $w = xyz$, tal que:

1. $|xy| \leq n$
2. $y \neq \varepsilon$
3. Para todo $k \geq 0$, la palabra $xy^kz \in L$



Lema de bombeo: demostración

- L regular \Rightarrow hay un AFD que acepta L
 - Sea A ese AFD;
 - Tomamos $n = \text{número de estados de } A$
- Consideramos las palabras $w \in L$ con $|w| \geq n$
 $w = a_1 a_2 \dots a_m$, con $m \geq n$
- Conjuntos atravesados al leer los primeros n símbolos de w : $\{p_0, p_1, \dots, p_n\}$
 - \Rightarrow Hay $n+1$ estados, mientras que el AFD tiene sólo n estados
 - \Rightarrow al menos un estado se repite, es decir, $p_i = p_j$ con $0 \leq i < j \leq n$ (principio de palomar)

Lema de bombeo: demostración...

➤ \Rightarrow Descomponemos $w = xyz$, de la siguiente forma:

➤ $x = a_1 a_2 \dots a_i$; $y = a_{i+1} a_{i+2} \dots a_j$; $z = a_{j+1} a_{j+2} \dots a_m$

➤ x : camino $p_0 \dots p_i$

➤ y : camino $p_i p_{i+1} \dots p_j$ (como $p_i = p_j$ hay un ciclo)

➤ z : camino $p_j p_{j+1} \dots p_m$

➤ Consideremos cualquier $w_2 = xy^k z$

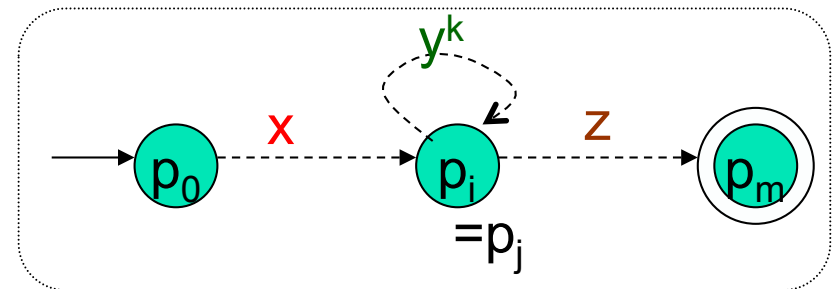
➤ Caso $k=0$

➤ A alcanza el estado p_m

➤ Caso $k>0$

➤ A cicla en y^k , y finalmente alcanza el estado p_m

➤ En cualquier caso, $w_2 \in L$

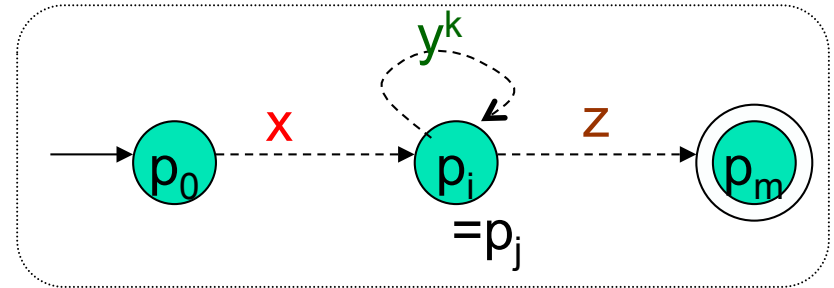


Esto prueba la parte (3) del lema

Lema de bombeo: demostración...

- Para la parte (2):

- Como $i < j$, $y \neq \varepsilon$



- Para la parte (1):

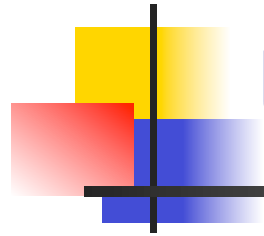
- Por el principio del palomar, los estados tienen que repetirse en los primeros n símbolos
- $\implies |xy| \leq n$

□



Utilidad del Lema de Bombeo para Lenguajes Regulares

- Probar que algunos lenguajes *no pueden* ser regulares.



Uso del lema de bombeo

Juego entre dos personas

- Jugador 1 (el malo): afirma que el lenguaje es regular
- Jugador 2 (el bueno): su adversario, que quiere probar que eso no puede ser verdad

Uso del lema de bombeo

El bueno (nosotros)

Construye $w \in L$ con $|w| \geq n$ usando N (sin usar un valor particular para N)

Intenta violar la tercera condición del lema, sean cuales sean $\{x,y,z\}$

- primero lo intenta eliminando y ($k=0$)
- si eso no funciona, lo intenta añadiendo y 's ($k \geq 2$)

El malo

Afirma L es regular

\Rightarrow Conoce N pero no lo revela

Descompone w en $\{x,y,z\}$
con $y \neq \varepsilon$, $|xy| \leq n$
(de nuevo, no revela xyz)



Ejemplo de uso del lema de bombo para probar que un lenguaje no es regular

Sea $L_{eq} = \{w \mid w \text{ cadena binaria con el mismo número de 0s que de 1s}\}$

- Hipótesis: L_{eq} no es regular

- Demostración:

- L_{eq} es regular. n = constante del lema de bombeo

MALO

- Consideramos $w = 0^n 1^n$

BUENO

- Se puede descomponer $w = xyz$, tal que:

MALO

- 1) $y \neq \varepsilon$

- 2) $|xy| \leq n$

- 3) Para todo $k \geq 0$, la palabra xy^kz también está en L

Demostración... $w = 0^n 1^n$

- Como $|xy| \leq n$, xy sólo contiene 0s
 - (y como $y \neq \varepsilon$, $y = 0^l$ con $l > 0$)
- Además, los n 1s están en z
- Por (3), las cadenas $xy^kz \in L_{eq}$ para todo $k \geq 0$
- Caso $k=0$: xz tiene $n-1$ 0s y n 1s
- Por lo tanto, $xy^0z \notin L_{eq}$ (contradicción) ↯

BUENO





Ejemplo 2

$L = \{0^n 1 0^n \mid n \geq 1\}$ no es regular

- Obs: No confundir ese n con la constante del lema de bombeo.
 - $L = \{0^k 1 0^k \mid k \geq 1\}$ no es regular



Ejemplo 3

Hip: $L = \{ 0^i \mid i \text{ cuadrado perfecto} \}$ no es regular

■ Demostración:

- Suponemos que L es regular.
- Entonces se cumple el lema de bombeo
- n = constante del lema de bombeo
- Consideramos $w = 0^{n^2}$
- Sea $w = xyz$ que satisface las tres condiciones
- Por (1) y (2), y tiene entre 1 y n 0s
- Por (3), las palabras xy^kz están en L para todo $k \geq 0$
- Caso $k=0$:

$$\text{\#ceros}(xy^0z) = \text{\#ceros}(xyz) - \text{\#ceros}(y)$$

$$\text{\#ceros}(xy^0z) \leq n^2 - n$$

$$(n-1)^2 < n^2 - n \leq \text{\#ceros}(xy^0z) \leq n^2 - 1 < n^2$$

$$xy^0z \notin L$$




4. Propiedades de los Lenguajes Regulares

4.2. Propiedades de clausura de los Lenguajes Regulares

Fernando Rosa Velardo

Traducción y adaptación de transparencias de Ananth Kalyanaraman
(<http://www.eecs.wsu.edu/~ananth/>)



Propiedades de clausura de lenguajes regulares

Nada que ver con la clausura de Kleene

- Propiedad de clausura:
 - Si un conjunto de lenguajes regulares se combinan usando una operación, el lenguaje resultado también es regular
- Los lenguajes regulares son cerrados para:
 - Unión
 - Intersección, complemento, diferencia
 - Reflexión
 - Concatenación
 - Clausura de Kleene



Unión

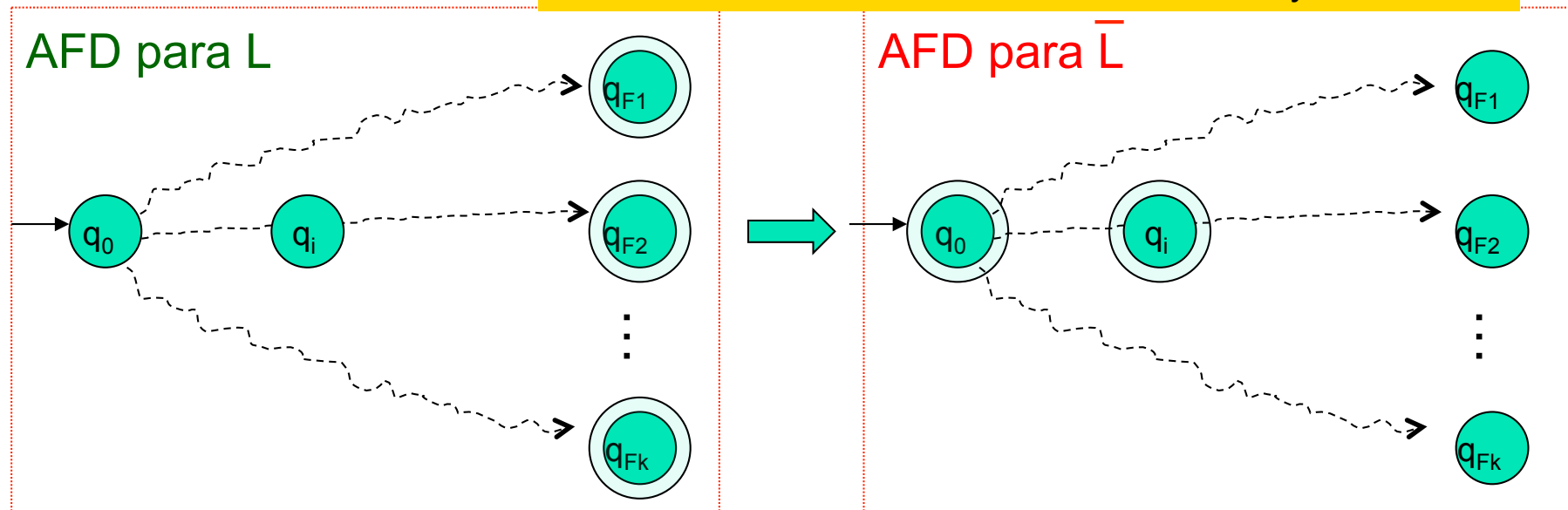
- Si L y M son lenguajes regulares:
 - Existen expresiones regulares R y S que generan L y M , respectivamente
 - La expresión regular $R+S$ genera $L \cup M$
 - Por lo tanto, $(L \cup M)$ también es regular

¿Se puede probar usando autómatas?

Complemento

- Si L es un lenguaje regular sobre Σ entonces
 $\bar{L} = \Sigma^* - L$
- L es regular, mediante la construcción...

Convertimos estados finales en no finales, y viceversa





Intersección

- Demostración rápida e indirecta:
 - Por las leyes de DeMorgan:
 - $L \cap M = \overline{(\bar{L} \cup \bar{M})}$
 - Como los lenguajes regulares son cerrados para la unión y el complemento, también lo son para la intersección
- Más directa: construcción de un autómata finito para $L \cap M$

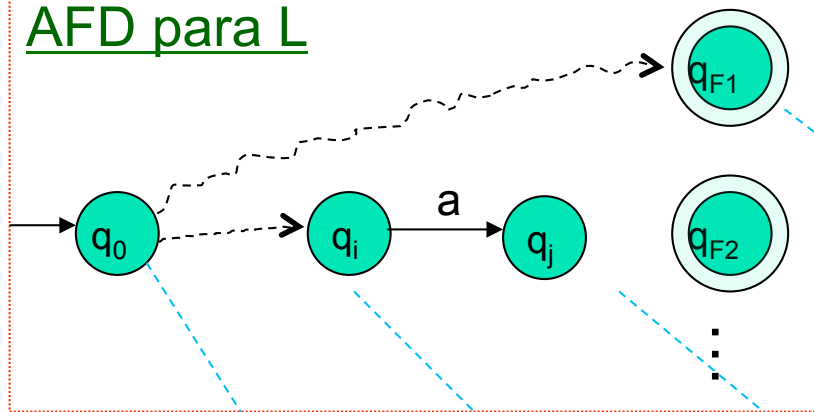


AFD para $L \cap M$

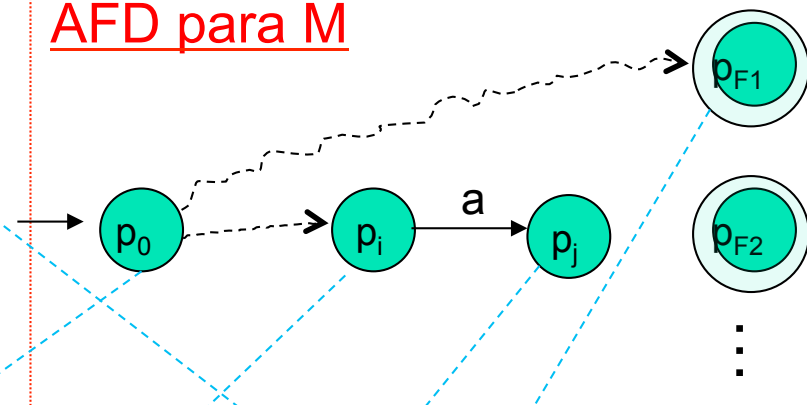
- $A_L = \text{AFD para } L = \{Q_L, \Sigma, q_L, F_L, \delta_L\}$
- $A_M = \text{AFD para } M = \{Q_M, \Sigma, q_M, F_M, \delta_M\}$
- $A_{L \cap M} = \{Q_L \times Q_M, \Sigma, (q_L, q_M), F_L \times F_M, \delta\}$ tal que:
 - $\delta((p, q), a) = (\delta_L(p, a), \delta_M(q, a))$, con p en Q_L , y q en Q_M
- Se acepta w si y sólo si w alcanza un estado que es final en ambos autómatas

AFD para $L \cap M$

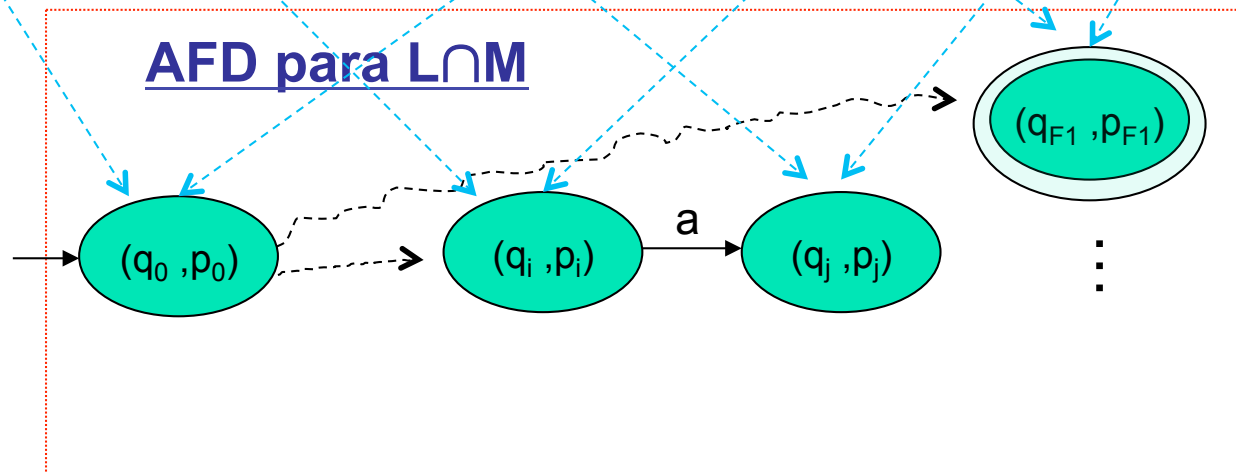
AFD para L



AFD para M



AFD para $L \cap M$





Diferencia

- Observamos que:

- $L - M = L \cap \overline{M}$

Cerrados para la intersección

Cerrados para el
complemento

- Por lo tanto, $L - M$ también es regular



Reflexión

w^R = reflexión de la cadena

- Por ejemplo, $w=00111$, $w^R=11100$

Reflexión de un lenguaje:

- L^R = lenguaje generado por reflexiones de cadenas de L

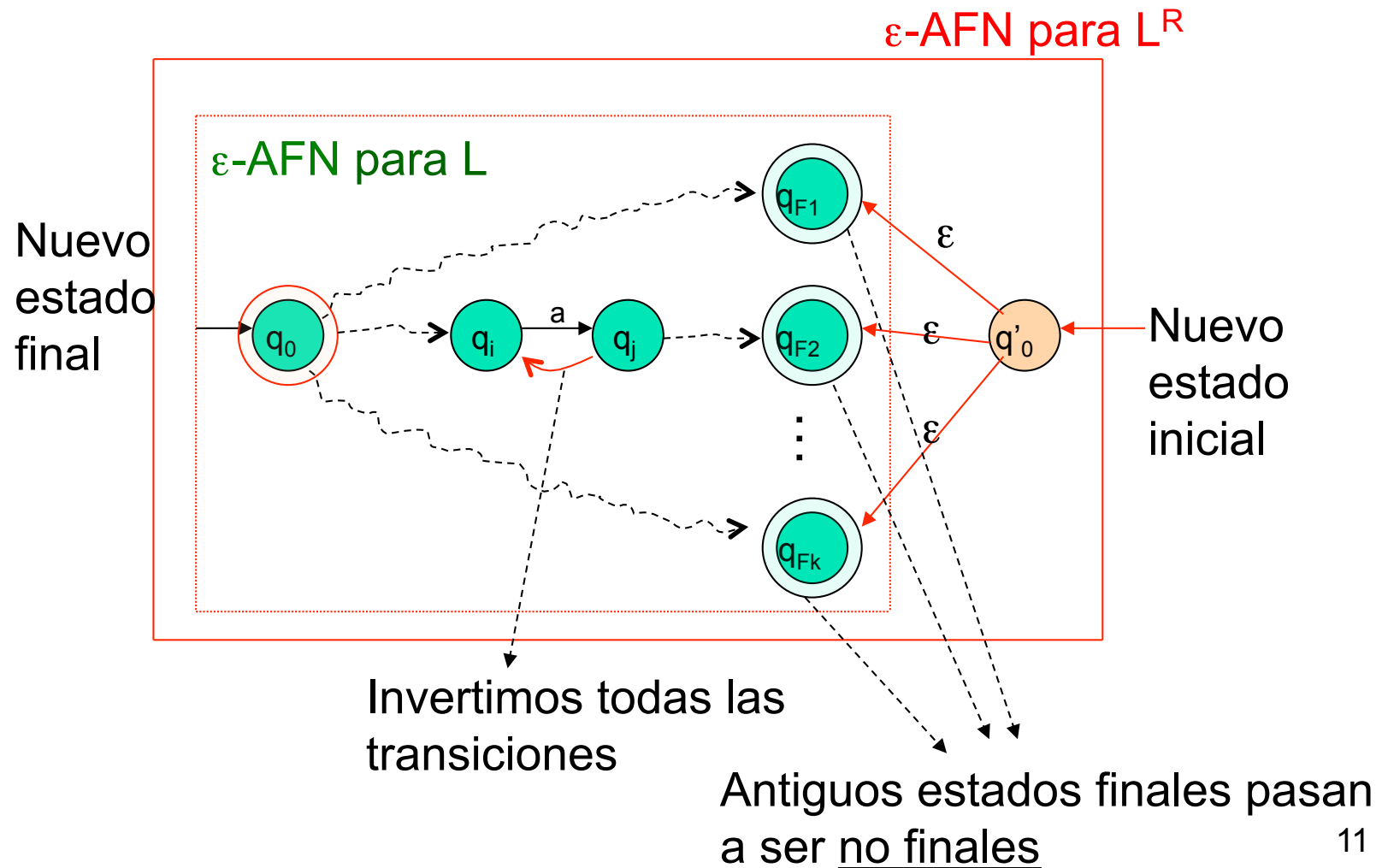
Teorema: si L es regular también lo es L^R



Si L es regular L^R es regular (usando expresiones regulares)

- Sea E una expresión regular que genera L
- Construimos E^R por inducción en el número de operadores que hay en E :
- Base: Si $E = \varepsilon, \emptyset$, o a , entonces $E^R = E$
- Inducción: E puede ser de una de las siguientes formas:
 1. $E = E_1 + E_2$
 - $E^R = E_1^R + E_2^R$
 2. $E = E_1 E_2$
 - $E^R = E_2^R E_1^R$
 3. $E = (E_1)^*$
 - $(E^R)^* = (E_1^R)^*$

Construcción de ϵ -AFN para L^R





4. Propiedades de los Lenguajes Regulares

4.3. Propiedades de decisión de los Lenguajes Regulares

Fernando Rosa Velardo

Traducción y adaptación de transparencias de Ananth Kalyanaraman
(<http://www.eecs.wsu.edu/~ananth/>)



Pertenencia

- Entrada: w y una representación de L
- Salida: respuesta a ¿está w en L ?
- Algoritmo:
 1. Construye un AFD para L
 2. Ejecuta el AFD sobre w
 3. Si el AFD acepta devolvemos true; si no, devolvemos false.



Test de vacío

- Entrada: una representación de L
- Salida: respuesta a ¿ $L=\emptyset$?
- Algoritmo:
 1. Construye un AF para L
 2. Ejecuta un test de alcanzabilidad, que devuelve:
 1. false: si existe un estado final alcanzable desde el estado inicial
 2. true: en otro caso

¿Cómo se implementa ese test de alcanzabilidad?



Finitud

- Entrada: una representación de L
- Salida: ¿es L finito?
- Algoritmo:
 1. Construye un AFN para L
 2. Elimina todos los estados inalcanzables desde el estado inicial
 3. Elimina todos los estados desde los que no se puede alcanzar ningún estado final
 4. Busca ciclos en el AFN resultante
 5. L es finito si y sólo si no hay ciclos

¿Cómo se implementan los pasos 2 y 3?



Inclusión

- Entrada: representaciones de L y M
- Salida: ¿L contenido en M?
- Algoritmo:
 1. Construye un AFD para $L \cap \bar{M}$
 2. Ejecuta el test de vacío sobre ese autómata
 3. L contenido en M si y sólo si el test devuelve cierto



Equivalencia

- Entrada: representaciones de L y M
- Salida: ¿ $L = M$?
- Algoritmo:
 1. $L=M$ si y sólo si L contenido en M y M contenido en L



4. Propiedades de los Lenguajes Regulares

4.4. Equivalencia y minimización de autómatas

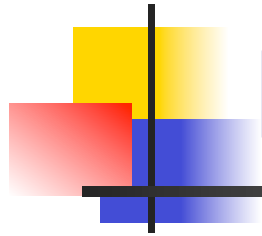
Fernando Rosa Velardo

Traducción y adaptación de transparencias de Ananth Kalyanaraman
(<http://www.eecs.wsu.edu/~ananth/>)



Objetivos

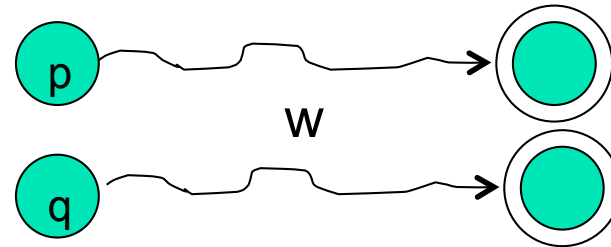
- Minimización de AFDs:
 - Construcción de un AFD equivalente a uno dado con el menor número de estados posible
- Aplicación
 - ¿ $L(\text{AFD}_1) = L(\text{AFD}_2)$?



Estados equivalentes en AFD

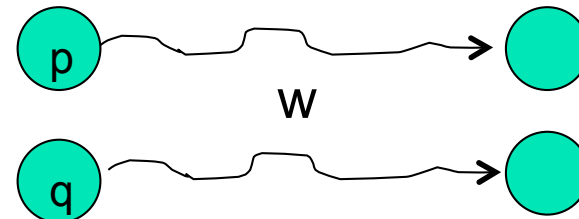
Decimos que dos estados p y q son *equivalentes* si:

- i) Cualquier w aceptada desde p también es aceptada desde q ;



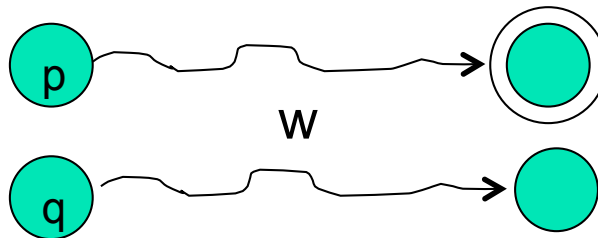
Y

- ii) Cualquier w rechazada desde p también es rechazada desde q .



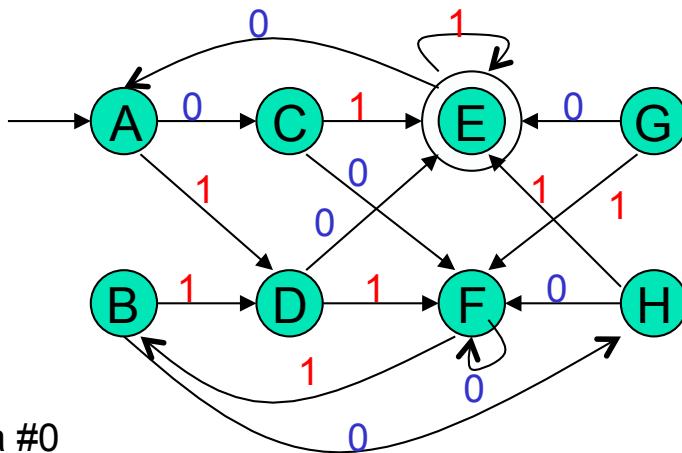
Estados equivalentes en AFD

Equivalentemente, p y q NO son *equivalentes* si existe w aceptada desde p y rechazada desde q (o viceversa):



Cálculo de estados equivalentes en un AFD

Algoritmo de llenado de tabla



Pasada #0

1. Marcamos pares de estado final y no final

Pasada #1

1. Comparamos cada par de estados no marcados
2. Distinguimos mediante palabras de longitud 1

Pasada #2

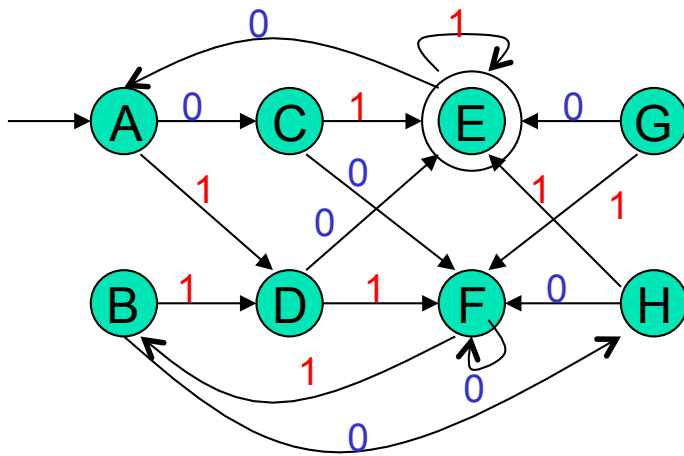
1. Comparamos cada par de estados no marcados
2. Distinguimos mediante palabras de longitud hasta 2

....

(hasta que no se pueden poner más marcas)

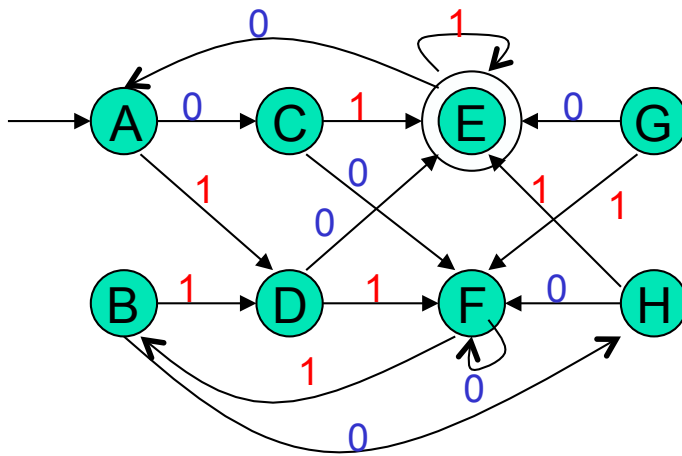
A								
B								
C	x	x						
D	x	x	x					
E	x	x	x	x				
F	x	x	x	x	x			
G	x	x	x		x	x		
H	x	x		x	x	x	x	
	A	B	C	D	E	F	G	H

Algoritmo de llenado de tabla, paso a paso



A								
B								
C								
D								
E								
F								
G								
H								
	A	B	C	D	E	F	G	H

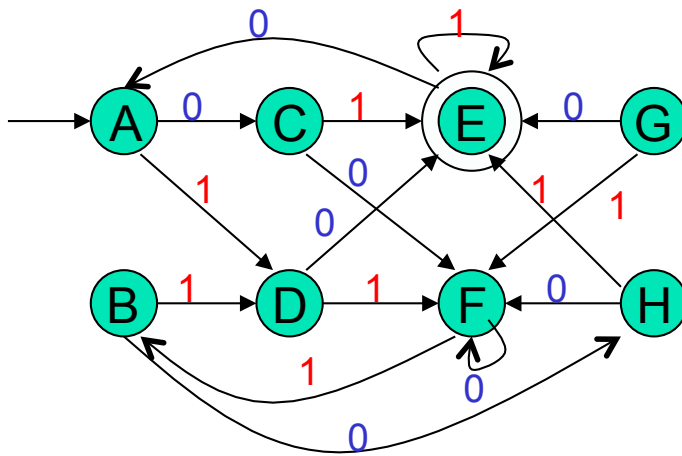
Algoritmo de llenado de tabla, paso a paso



1. Marcamos pares final/no final

A								
B								
C								
D								
E	X	X	X	X				
F					X			
G					X			
H					X			
	A	B	C	D	E	F	G	H

Algoritmo de llenado de tabla, paso a paso

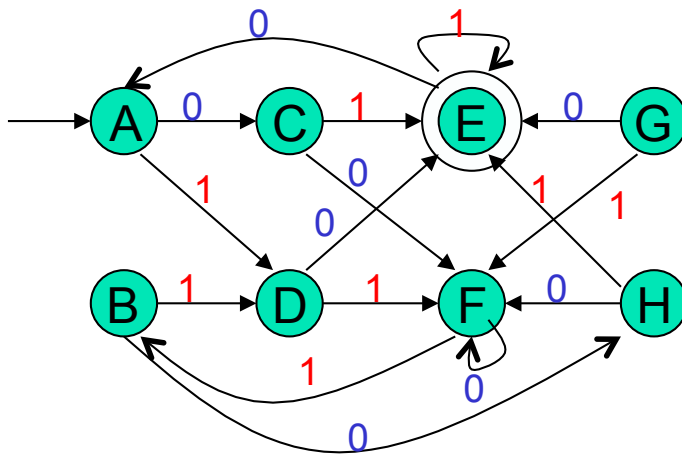


1. Marcamos pares final/no final
2. Buscamos pares distinguibles mediante símbolos

A								
B								
C	X							
D	X							
E	X	X	X	X				
F					X			
G	X				X			
H	X				X			
	A	B	C	D	E	F	G	H

↑

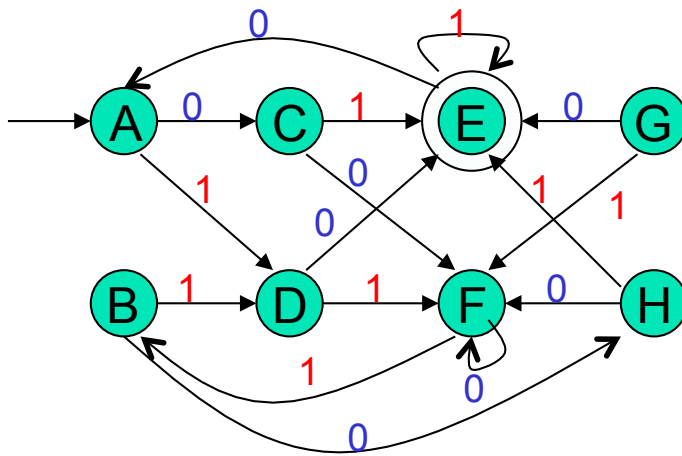
Algoritmo de llenado de tabla, paso a paso



1. Marcamos pares final/no final
2. Buscamos pares distinguibles mediante símbolos

A								
B								
C	X	X						
D	X	X						
E	X	X	X	X				
F					X			
G	X	X			X			
H	X	X			X			
	A	B	C	D	E	F	G	H

Algoritmo de llenado de tabla, paso a paso

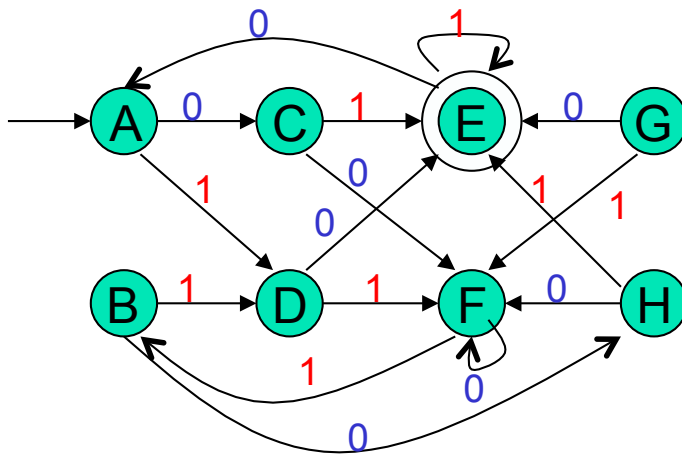


1. Marcamos pares final/no final
2. Buscamos pares distinguibles mediante símbolos

A								
B								
C	X	X						
D	X	X	X					
E	X	X	X	X				
F			X		X			
G	X	X	X		X			
H	X	X			X			
	A	B	C	D	E	F	G	H

↑

Algoritmo de llenado de tabla, paso a paso

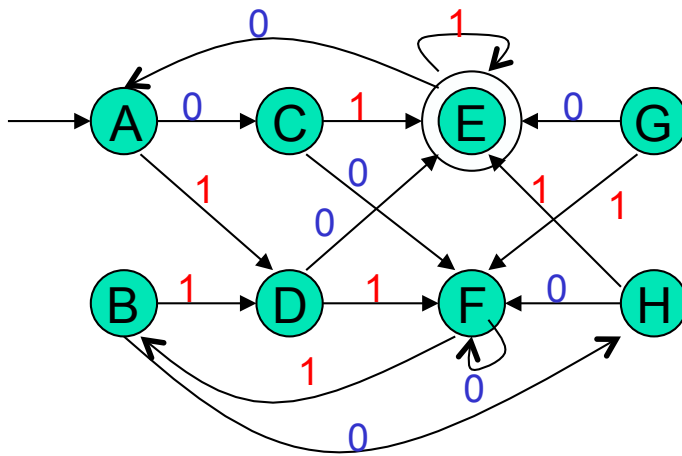


1. Marcamos pares final/no final
2. Buscamos pares distinguibles mediante símbolos

A								
B								
C	X	X						
D	X	X	X					
E	X	X	X	X				
F			X	X	X			
G	X	X	X		X			
H	X	X		X	X			
	A	B	C	D	E	F	G	H

↑

Algoritmo de llenado de tabla, paso a paso

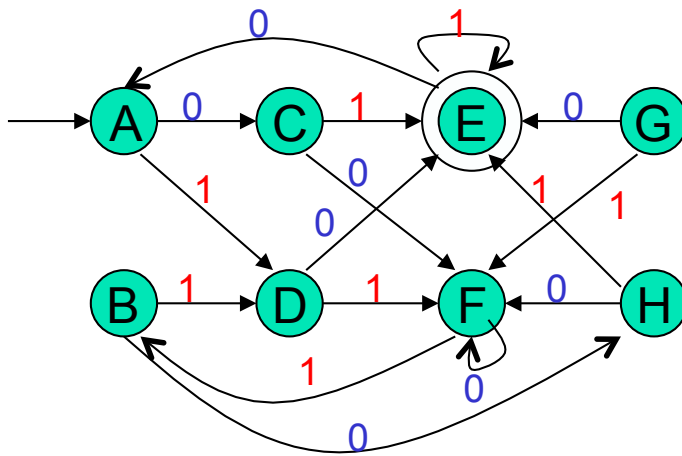


1. Marcamos pares final/no final
2. Buscamos pares distinguibles mediante símbolos

A								
B								
C	X	X						
D	X	X	X					
E	X	X	X	X				
F			X	X	X			
G	X	X	X		X	X		
H	X	X		X	X	X		
	A	B	C	D	E	F	G	H



Algoritmo de llenado de tabla, paso a paso

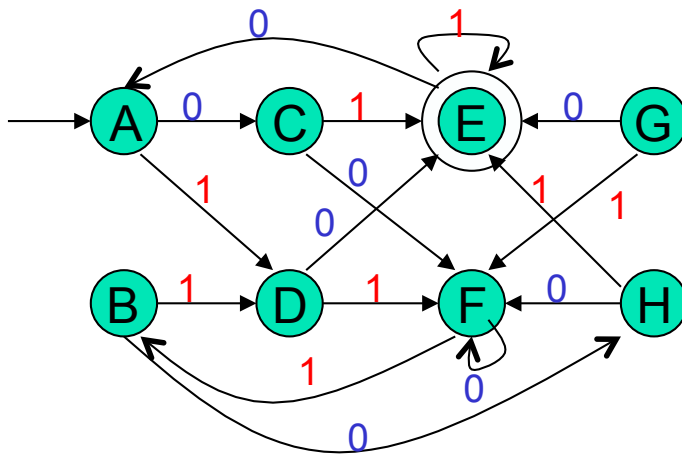


1. Marcamos pares final/no final
2. Buscamos pares distinguibles mediante símbolos

A								
B								
C	X	X						
D	X	X	X					
E	X	X	X	X				
F			X	X	X			
G	X	X	X		X	X		
H	X	X		X	X	X	X	
	A	B	C	D	E	F	G	H



Algoritmo de llenado de tabla, paso a paso



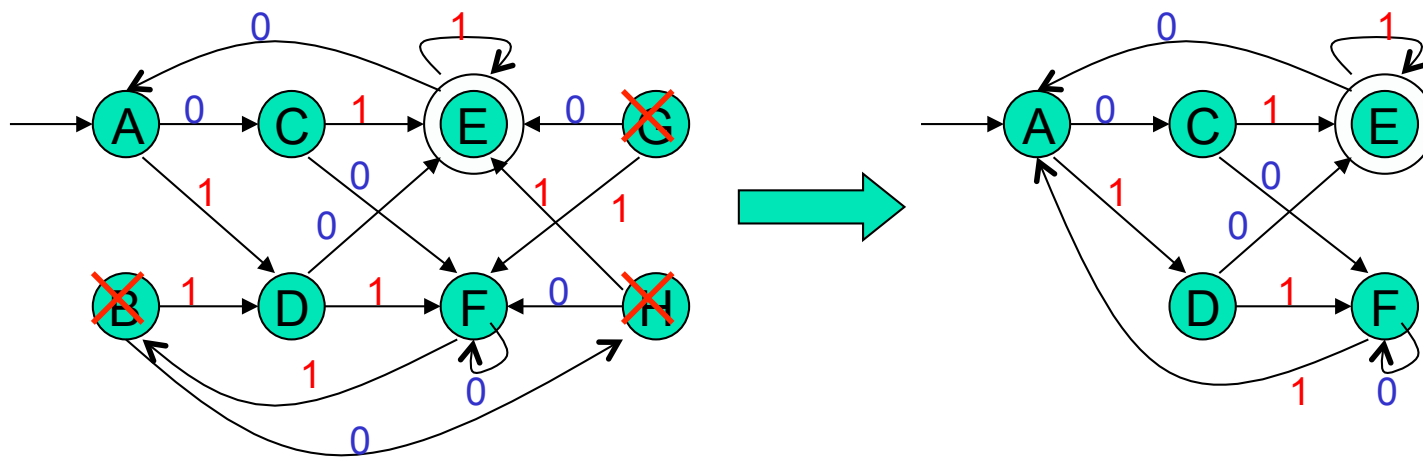
1. Marcamos pares final/no final
2. Buscamos pares distinguibles mediante símbolos
3. Buscamos pares distinguibles mediante palabras de longitud 2

A								
B								
C	X	X						
D	X	X	X					
E	X	X	X	X				
F	X	X	X	X	X			
G	X	X	X		X	X		
H	X	X		X	X	X	X	
	A	B	C	D	E	F	G	H

Equivalencias:

- A=B
- C=H
- D=G

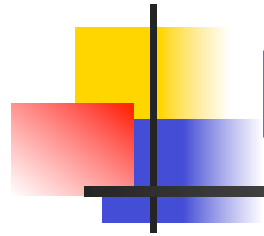
Algoritmo de llenado de tabla, paso a paso



Unificación de estados equivalentes

Equivalencias:

- A=B
- C=H
- D=G



Minimización de AFD

- Objetivo: Minimizar número de estados de un AFD

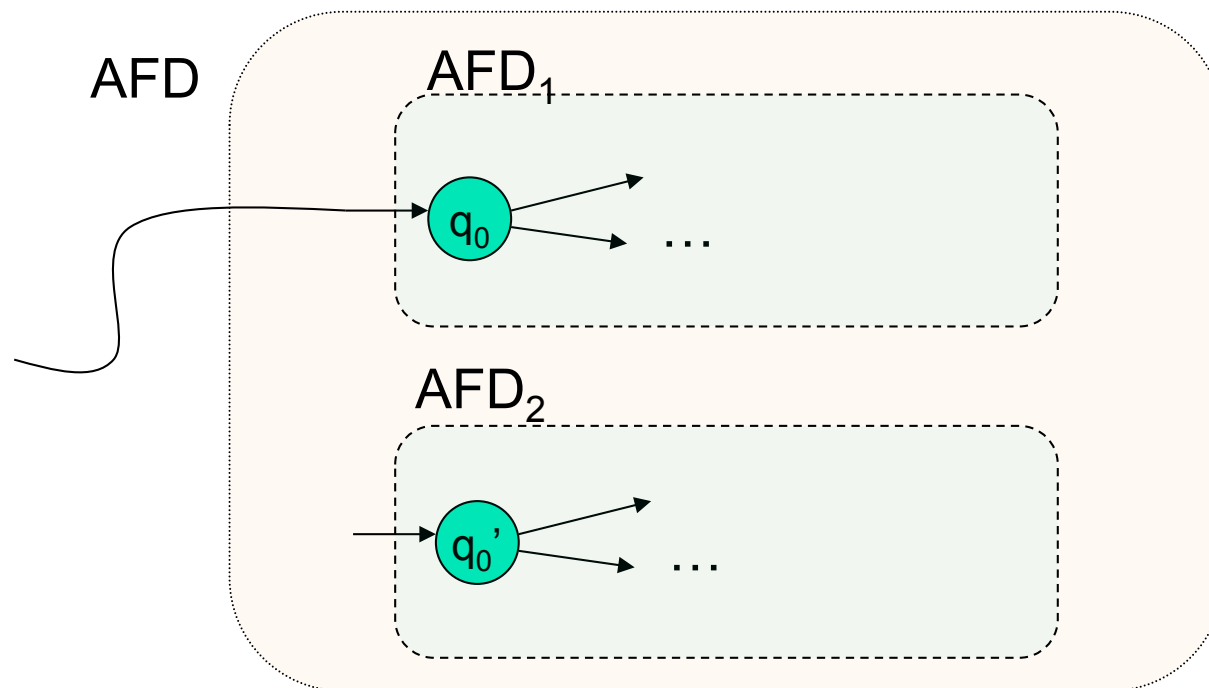
- Algoritmo:

Test de alcanzabilidad

1. Eliminar estados inalcanzables desde el estado inicial
2. Identificar y eliminar estados equivalentes

Llenado de tabla

Equivalencia de AFDs



¿ q_0 y q_0'
equiv.?

1. Construye nuevo AFD, simplemente juntando ambos
2. Ejecuta el algoritmo de llenado de tabla sobre el nuevo AFD
3. *IF* (q_0 y q_0' equivalentes)
 THEN: AFD₁ y AFD₂ equivalentes
 ELSE: no equivalentes