TeoriaComputacion_5

December 16, 2019

Teoría de la Computación

Expresiones Regulares y

Gramáticas Regulares

Prof. Wladimir Rodriguez

wladimir.rodriguez@outlook.com

Departamento de Computación

0.1 Expresiones Regulares:

- Si A es un alfabeto, una expresión regular sobre este alfabeto se define de la siguiente forma:
  - $\emptyset$ es una expresión regular que denota el lenguaje vacío.
  - ϵ es una expresión regular que denota el lenguaje $\{\epsilon\}$
  - Si $a \in A$, a es una expresión regular que denota el lenguaje $\{a\}$
- Si r y s son expresiones regulares denotando los lenguajes R y S entonces definimos las siguientes operaciones:
  - *Unión*: $(r + s)$ es una expresión regular que denota el lenguaje $R \cup S$
  - *Concatenación*: (rs) es una expresión regular que denota el lenguaje RS
  - *Clausura*: r^* es una expresión regular que denota el lenguaje R^*.

0.2 Ejemplos sobre $A = \{0, 1\}$

- 00 El Conjunto $\{00\}$
- $01^* + 0$ Conjunto de palabras que comienzan en 0 y después tienen una sucesión de cero o más unos
- $(1 + 10)^*$ Conjunto de palabras en las que los ceros están precedidos siempre por unos
- $(0 + 1)^*011$ Conjunto de palabras que terminan en 011
- 0^*1^* Conjunto de palabras formadas por una sucesión de ceros seguida de una sucesión de unos. Ambas sucesiones pueden ser nulas
- 00^*11^* Conjunto de palabras formadas por una sucesión de ceros seguida de una sucesión de unos. Ninguna de las sucesiones puede ser vacía
- A rr^* se le denota r^+ por lo que la anterior expresión se puede escribir 0^+1^+

0.3 Ejercicios - Alfabeto $\{0, 1\}$

- Construir una expresión para las palabras con un número par de ceros

$$1^*(01^*01^*)^*$$

- Construir una expresión regular para las palabras que contengan a 0110 como subcadena

$$(0+1)^*0110(0+1)^*$$

- Construir una expresión regular para el conjunto de palabras que empiezan por 000 y después no aparece mas esta cadena.

$$(000)(1+10+100)^*$$

- Construir una expresión regular para el conjunto de palabras que tienen a 000 o a 101 como subcadena

$$(0+1)^*(000+101)(0+1)^*$$

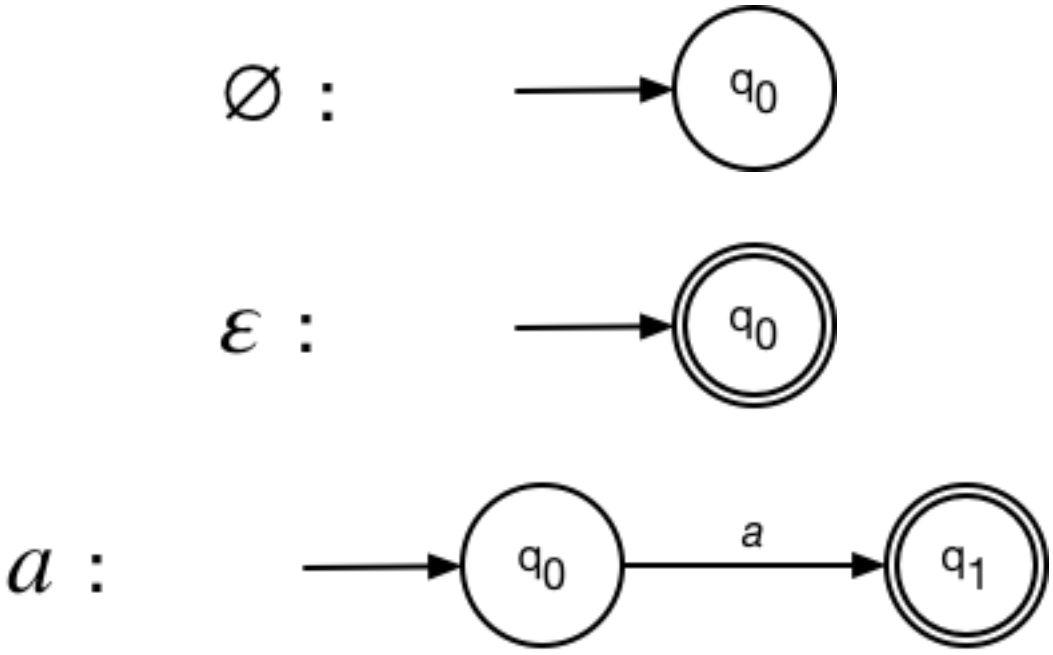
0.4 Propiedades de las Expresiones Regulares

1. $r_1 + r_2 = r_2 + r_1$	9. $(r_1 + r_2)\, r_3 = r_1 r_3 + r_2 r_3$
2. $r_1 + (r_2 + r_3) = (r_1 + r_2) + r_3$	10. $r^+ + \varepsilon = r^*$
3. $r_1(r_2 r_3) = (r_1 r_2) r_3$	11. $r^* + \varepsilon = r^*$
4. $r\varepsilon = r$	12. $(r + \varepsilon)^* = r^*$
5. $r\varnothing = \varnothing$	13. $(r + \varepsilon)^+ = r^*$
6. $r + \varnothing = r$	14. $({r_1}^* + {r_2}^*)^* = (r_1 + r_2)^*$
7. $\varepsilon^* = \varepsilon$	15. $({r_1}^* {r_2}^*)^* = (r_1 + r_2)^*$
8. $r_1(r_2 + r_3) = r_1 r_2 + r_1 r_3$	

0.5 Expresiones Regulares ⇒ Autómata

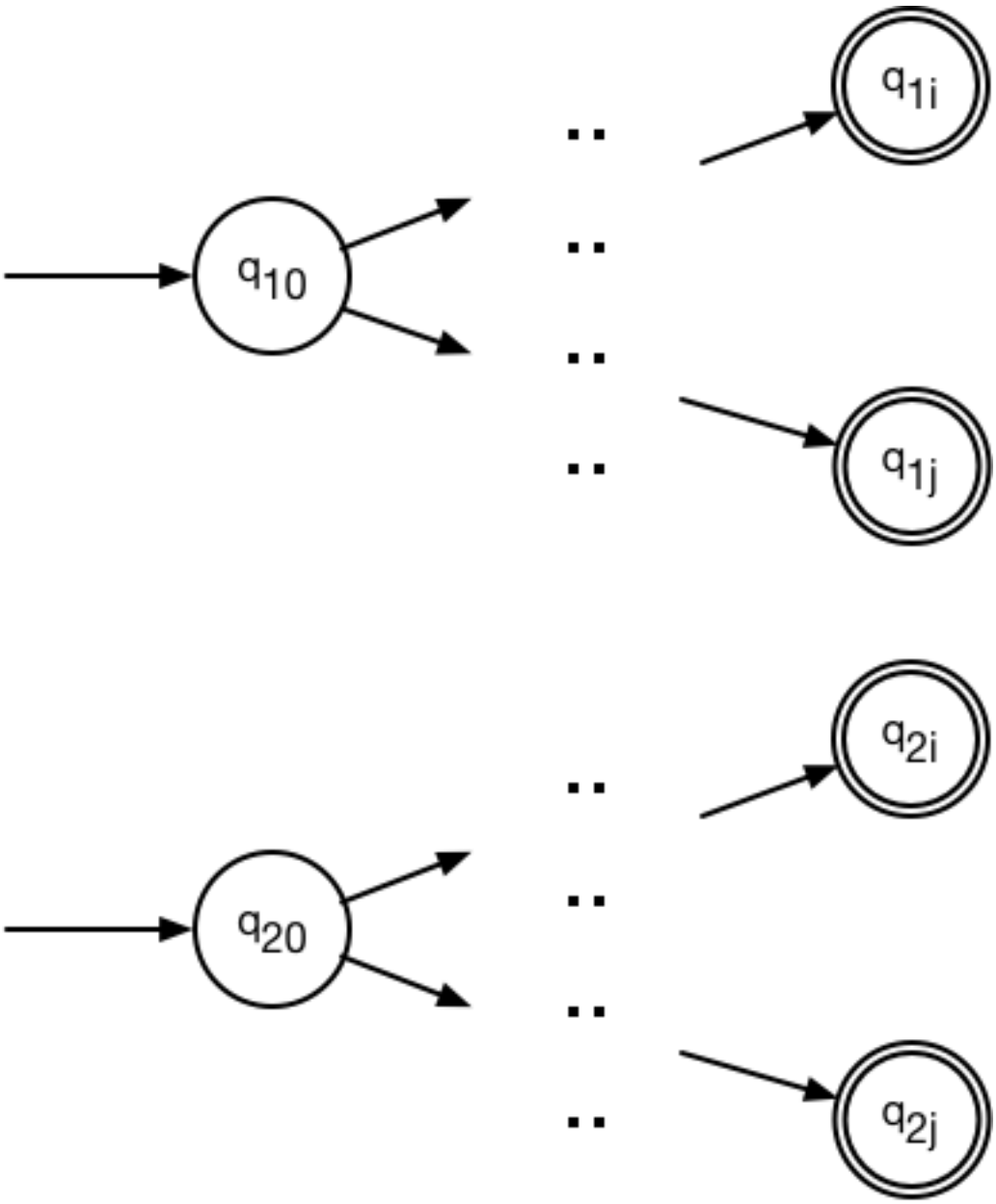
- Dada una expresión regular existe un autómata finito que acepta el lenguaje asociado a esta expresión regular.
- Vamos a demostrar que existe un AFND con transiciones nulas. A partir de él se podría construir el autómata determinista asociado.
- La construcción del autómata va a ser recursiva.

0.5.1 Para las expresiones regulares iniciales tenemos los siguiente autómatas:

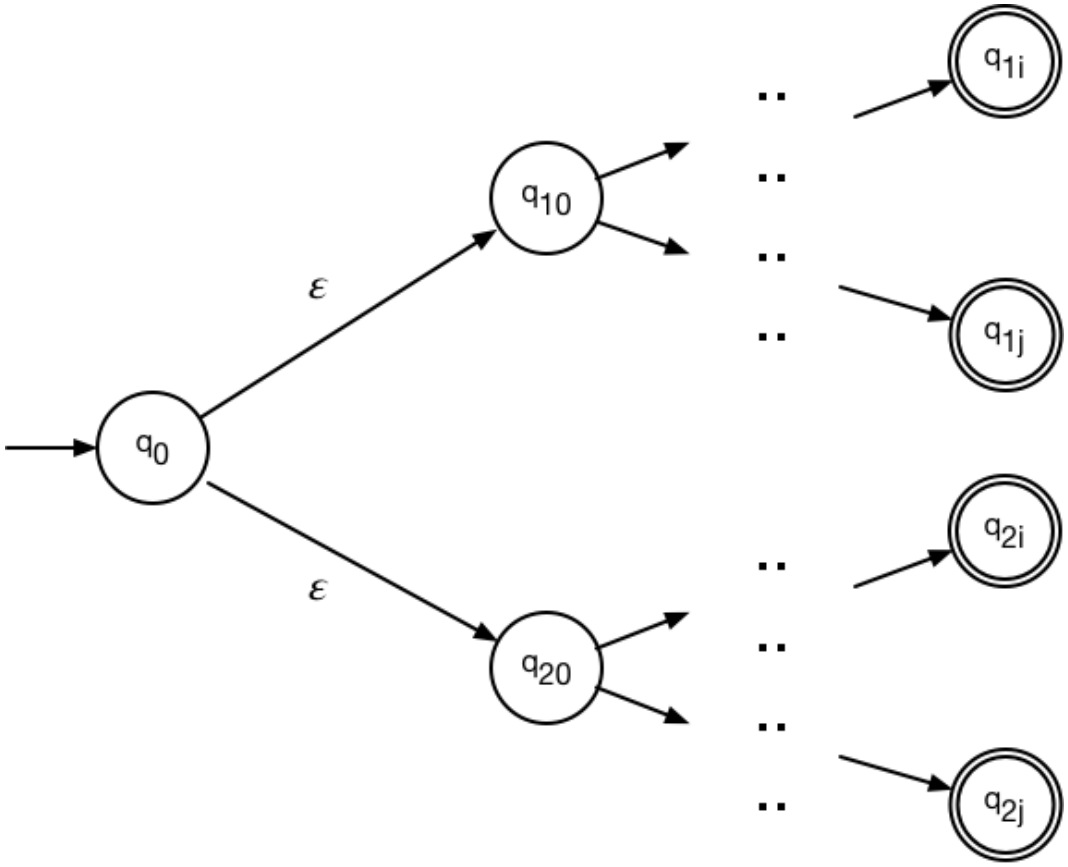


0.5.2 Autómatas Compuestos: Unión

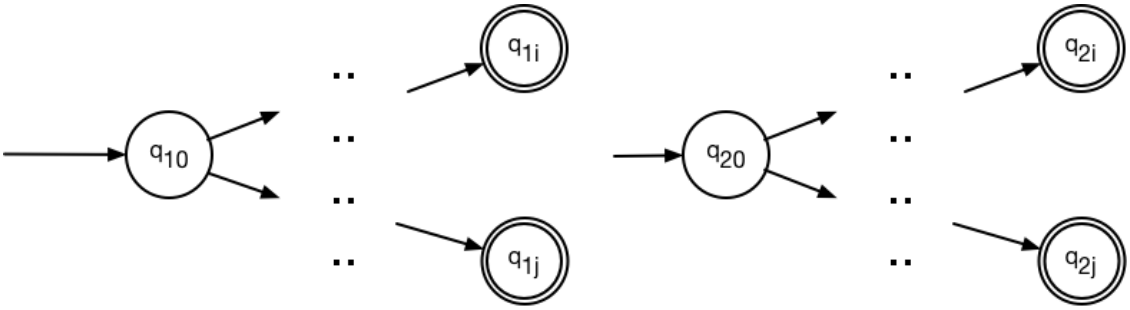
- Si M_1 es el autómata que acepta el mismo lenguaje que el representado por r_1 y M_2 el que acepta el mismo lenguaje que el de r_2 , entonces $Union(r_1 + r_2)$



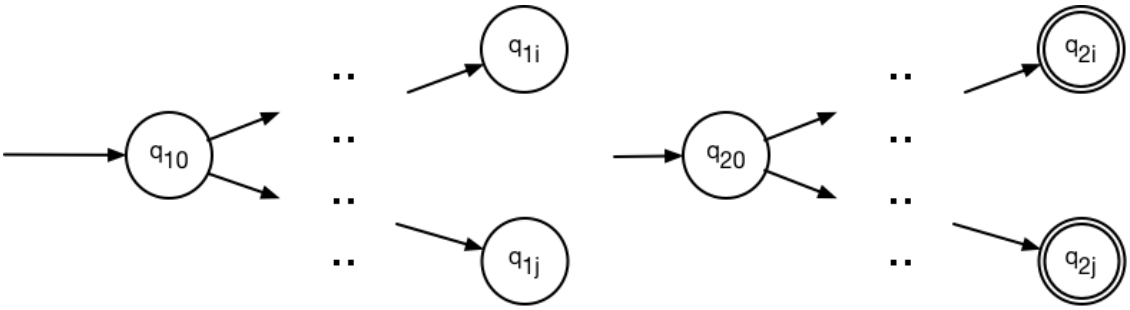
- Agregar un nuevo estado inicial y colcar transiciones nulas a los estados iniciales de los dos autómatas. $Union(r_1 + r_2) = $



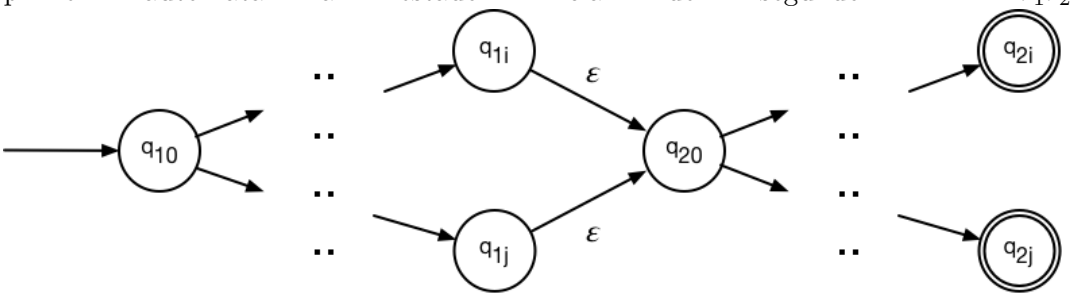
0.5.3 Concatenación: El autómata para la expresión $(r_1 r_2)$ es:



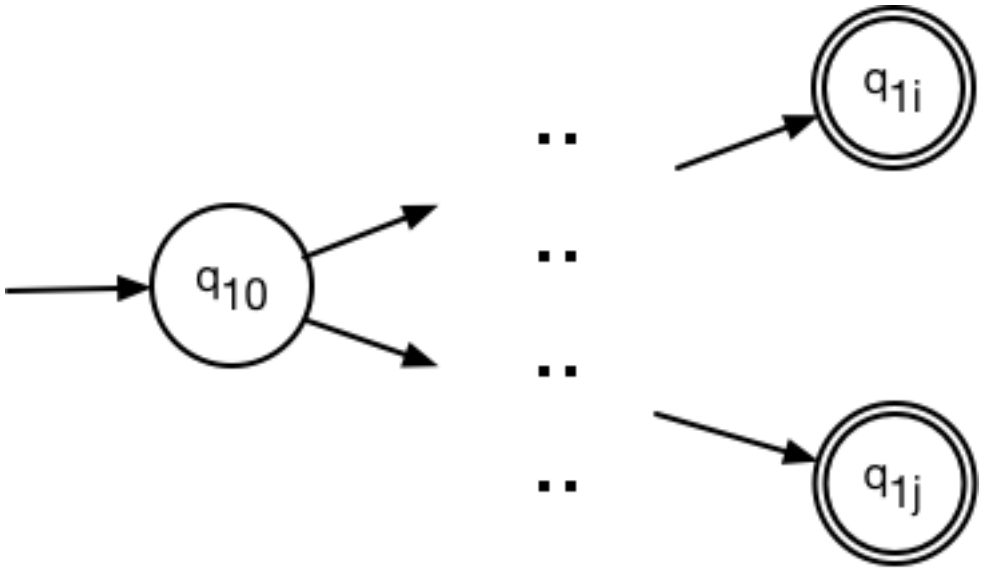
- Hacer no finales a los estados finales del primer autómata:



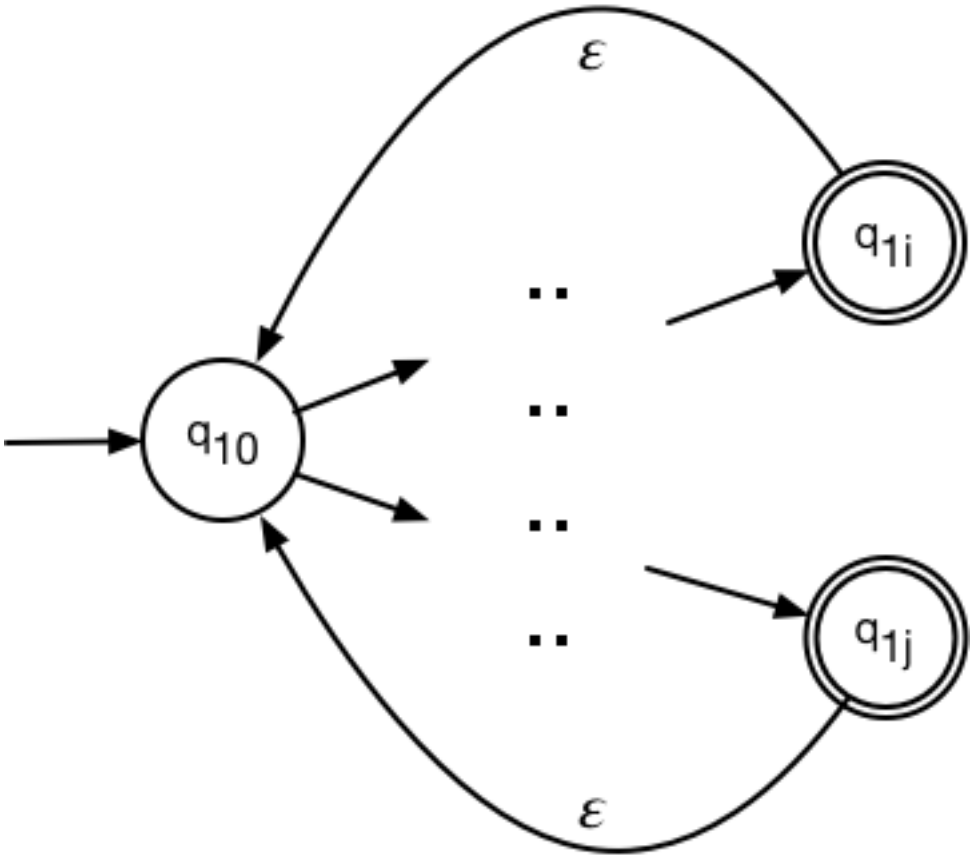
- Colocar transiciones nulas de los antiguos estados finales del primer autómata al estado inicial del segundo. $r_1 r_2$ =



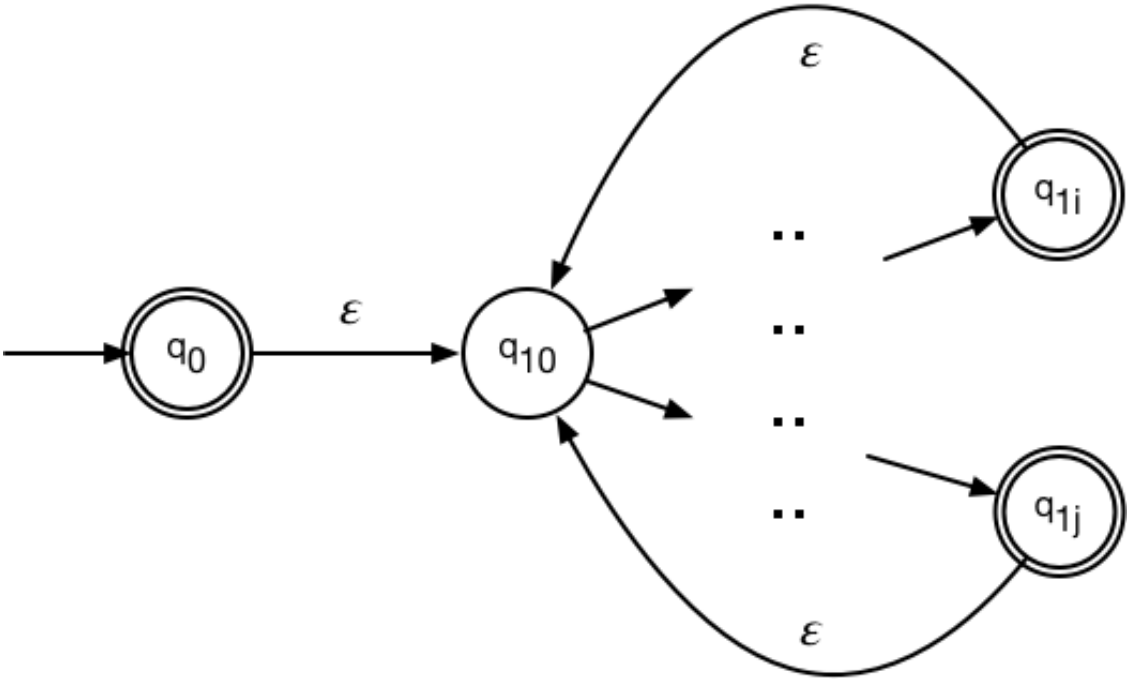
0.5.4 Clausura: El autómata para r_1^* es:



- Colocar transiciones nulas desde los estados finales al estado inicial:

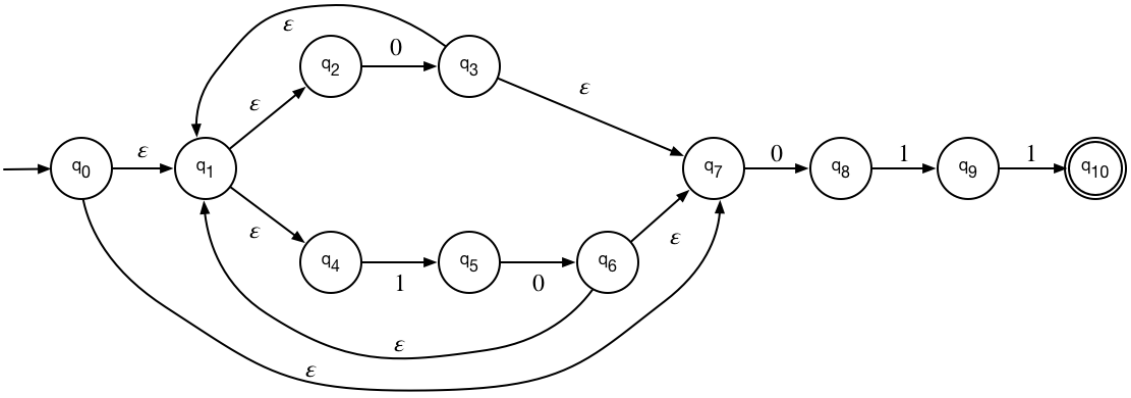


- Crear nuevo estado inicial/final con transición nula al antiguo estado unicial: $r_1^* =$



0.6 Ejercicio:

- Encontrar un autómata que acepte el mismo lenguaje que el asociado a la expresión regular $(0 + 10)^*011$
- $(0 + 10)^*011 =$

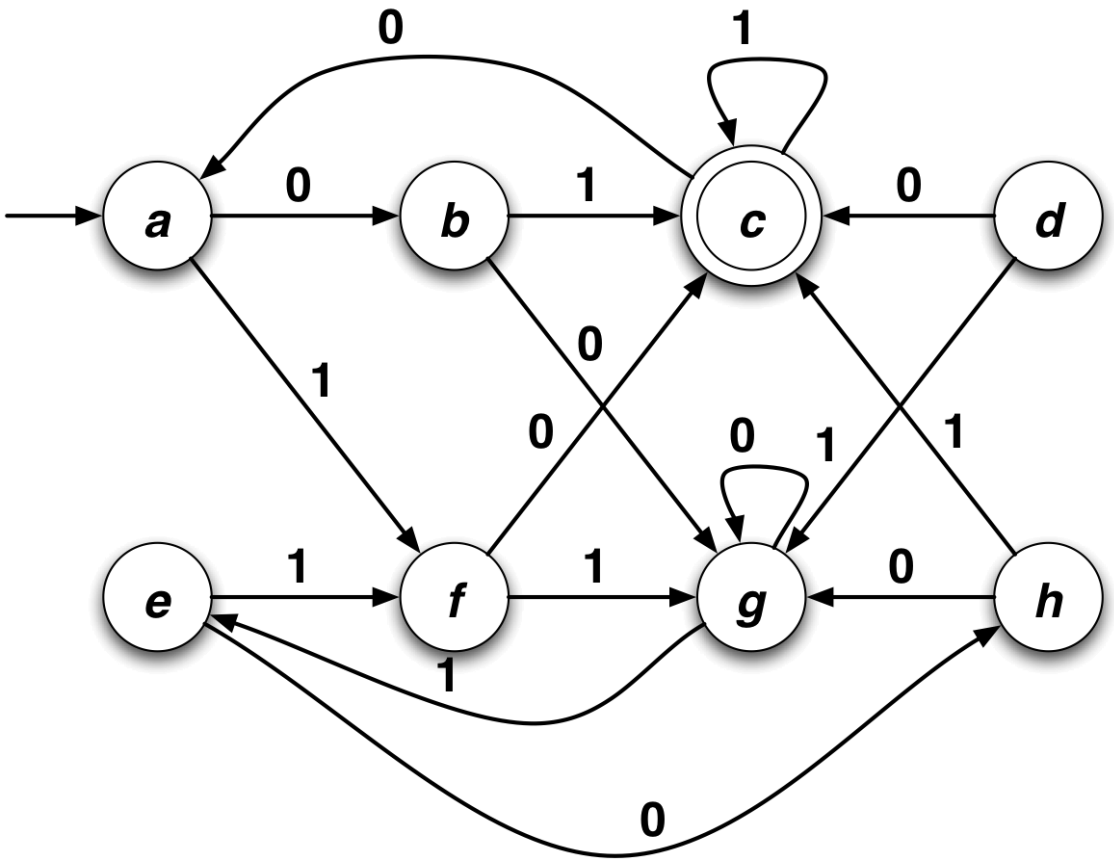


0.7 Minimización de Autómatas

- Existe un método simple para encontrar el AFD con número mínimo de estados M' equivalente a un AFD $M = (Q, A, \delta, q_0, F)$. Sea $\equiv$ la relación de equivalencia de los estados de M tal que $p \equiv q$ si y sólo si para cada entrada $x, \delta(p, x)$ es un estado de aceptación si y sólo si $\delta(q, x)$ es un estado de aceptación.
- Si $p \equiv q$, decimos que p es equivalente a q. Decimos que p es distinguible de q si existe un x tal que $\delta(p, x) \in F$ y $\delta(q, x) \notin F$, o viceversa.

0.7.1 Ejemplo:

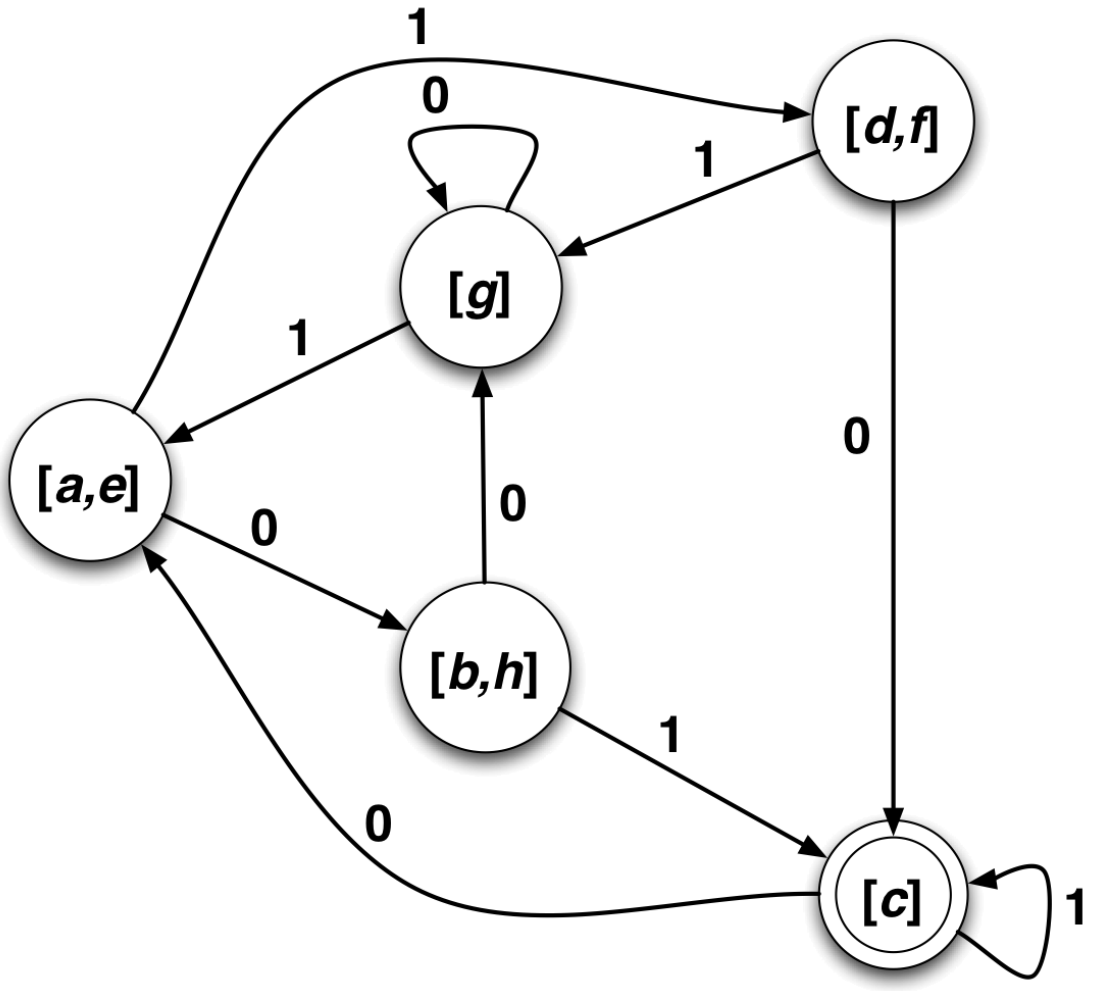
- Sea M el siguiente autómata



- Se tiene que construir una tabla con una entrada para cada par de estados. Se coloca una X en la tabla cada vez que un par de estados no son equivalentes. Inicialmente se coloca una X en cada entrada correspondiente a un estado final y un estado no final. En el ejemplo, colocamos una X en las entradas $(a, c), (b, c), (c, d), (c, e), (c, f), (c, g)$ y (c, h).
- Para cada par de estados p y q que no se han identificado como distinguibles, consideramos el par de estados $r = \delta(p, a)$ y $s = \delta(q, a)$ para cada entrada a.
- Si se demuestra que los estados s y r son distinguibles para alguna cadena x entonces p y q son distinguibles para cualquier cadena ax.
- Así si la entrada (r, s) en la tabla tiene una X, se coloca una X en la entrada (p, q).
- Si la entrada (r, s) no tiene X, entonces el par (p, q) es colocado en una lista asociada con la entrada (r, s).
- Continuando se tiene que si la entrada (r, s) recibe una X entonces cada par en la lista asociada con la entrada (r, s) también recibe una X.
- En el ejemplo, colocamos una X en la entrada (r, s), porque la entrada $(\delta(b, 1), \delta(a, 1)) = (c, f)$ ya tiene una X. Similarmente, la entrada (a, d) recibe una X. Ahora consideramos la entrada (a, e) que con la entrada 0 va a dar el par (b, h), así (a, e) es colocado en la lista asociada con (b, h). Observe que con la entrada 1, a y e van al mismo estado f y por lo tanto no hay cadena con 1 que pueda distinguir a de e.

b	X						
c	X	X					
d	X	X	X				
e		X	X	X			
f	X	X	X		X		
g	X	X	X	X	X	X	
h	X		X	X	X	X	X
	a	b	c	d	e	f	g

- Se concluye que los estados equivalentes son $a \equiv e$, $b \equiv h$, y $d \equiv f$, y el autómata con número de estados es el siguiente:



0.8 Gramáticas Regulares

- Los autómatas finitos son reconocedores de lenguajes y las gramáticas regulares son generadores
- Vamos a ver que ambos tratan con la misma clases de lenguajes: los lenguajes regulares.
  - Primero vamos a ver un teorema ($AF \rightarrow GR$) cuya demostración nos proporciona un método para obtener una GR a partir de un AF
  - Luego presentamos el teorema ($GR \rightarrow AF$) para obtener un AF a partir de una GR

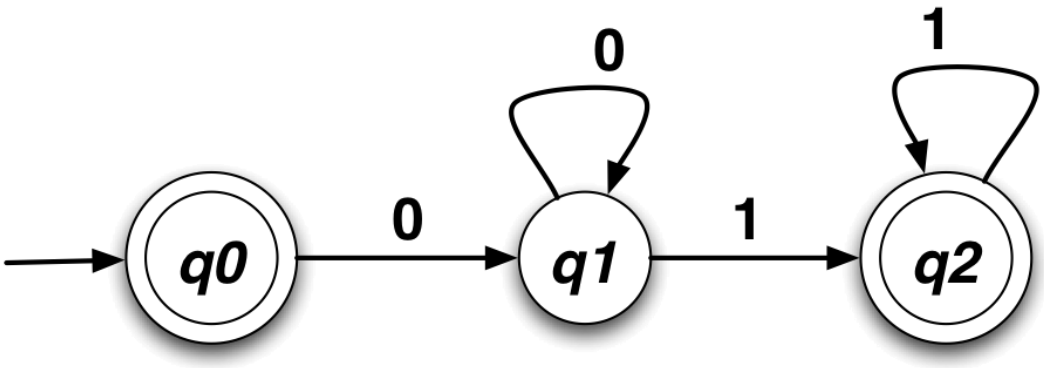
0.8.1 Equivalencia ($AF \rightarrow GR$)

- ($AF \rightarrow GR$) Si L es un lenguaje aceptado por una autómata finito M, entonces existe una gramática regular G tal que $L = L(M) = L(G)$.
- Suponemos que $M = (Q, A, \delta, q_0, F)$ es un AF sin λ-transiciones.
- Podemos obtener la gramática $G = (Q, A, q_0, P)$ a partir del diagrama de transición del AF con el siguiente método:
  - Si tenemos el arco $q \rightarrow p$ entonces añadimos a P la regla $q \rightarrow ap$

- Si q_f F añadimos la regla $q_f \rightarrow \lambda$

0.8.2 Ejemplo:

- Dado el siguiente AF, obtener la gramática correspondiente y la expresión regular.



- La gramática es:
  - $G = (\{q_0, q_1, q_2\}, \{0, 1\}, q_0, \{q_0 \rightarrow 0q_1|\lambda, q_1 \rightarrow 0q_1|1q_2, q_2 \rightarrow 1q_2|\lambda\})$

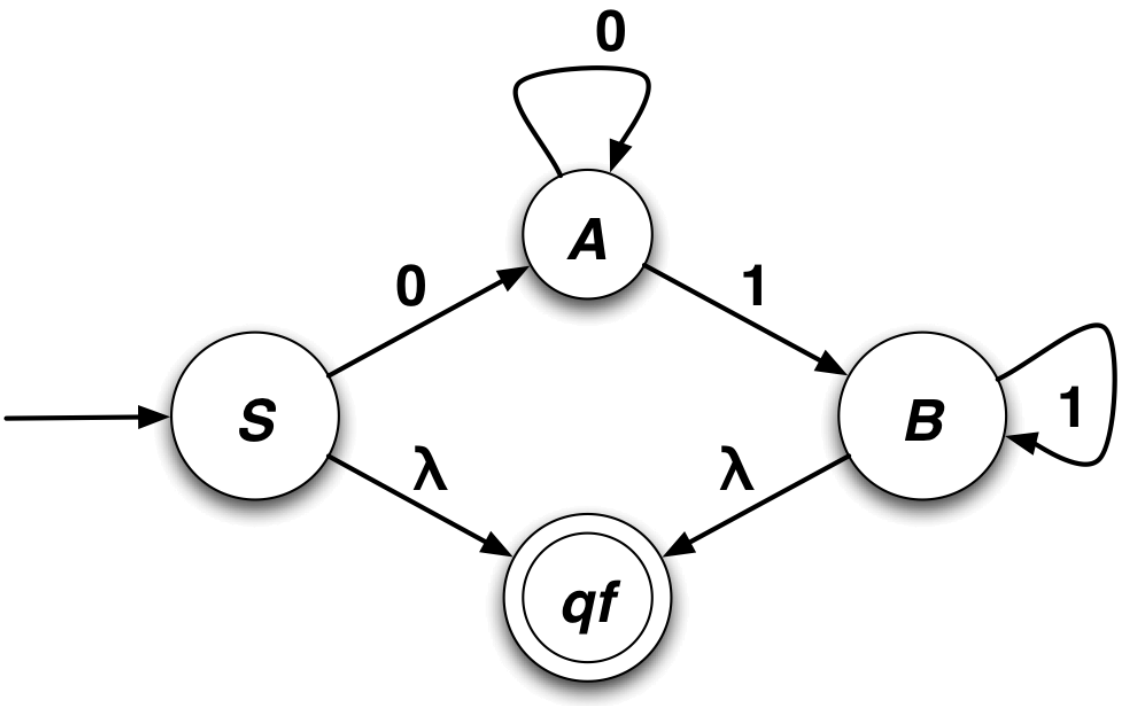
0.8.3 Equivalencia $(GR \rightarrow AF)$

- $(GR \rightarrow AF)$ Si L es un lenguaje generado por una gramática regular G, entonces existe un autómata finito M tal que $L = L(G) = L(M)$.
- Podemos suponer que $G = (V_n, V_t, S, P)$ es una gramática lineal derecha.
- Obtenemos el diagrama del autómata finito $M = (V_n \cup q_f, V_t, \delta, S, q_f)$ a partir de la gramática con el siguiente método:

- Si la regla $A \rightarrow aB \in P$ entonces añadimos el arco $A \rightarrow B$
- Si la regla $A \rightarrow a \in P$ añadimos el arco $A \rightarrow q_f$
- Si la regla $A \rightarrow \lambda \in P$ añadimos el arco $A \rightarrow q_f$

0.8.4 Ejemplo:

- Dada la gramática con reglas: $P = \{S \rightarrow 0A|\lambda, A \rightarrow 0A|1B, B \rightarrow 1B|\lambda\}$, el AF que reconoce el lenguaje generado por esta gramática es:

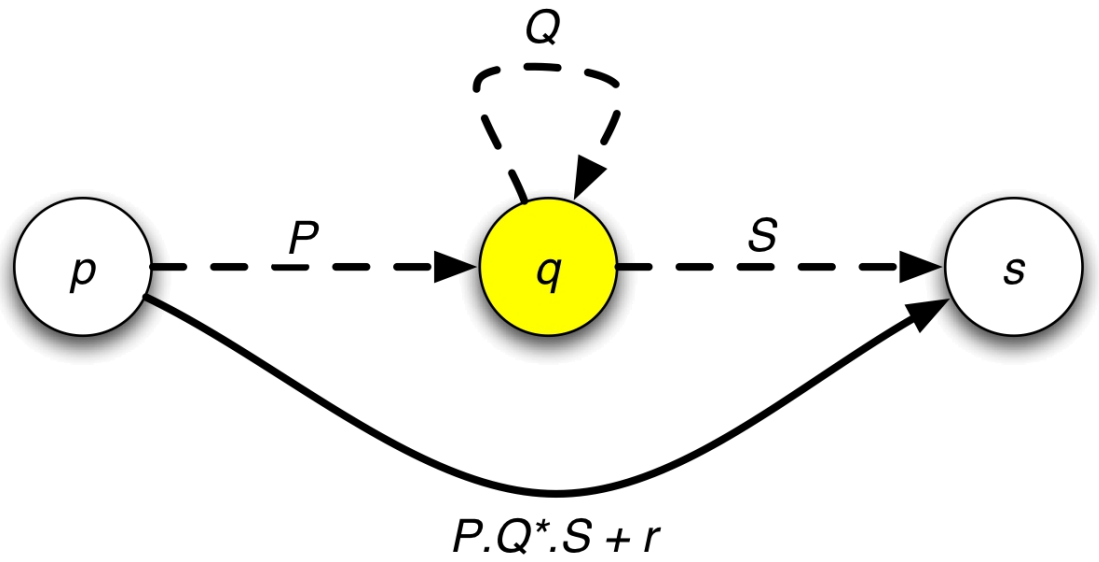


0.8.5 Equivalencia ($AF \rightarrow ER$)

- Eliminación de estados
- Ecuación fundamental

0.8.6 Eliminación de estado

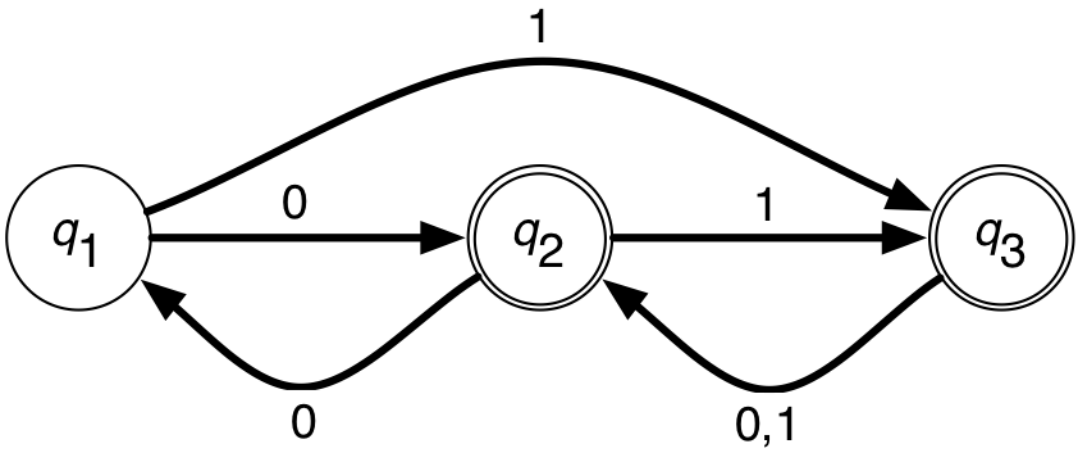
- Sea $q \notin \{q_0, q_f\}$ el estado a eliminar.
- Sea p un estado predecesor de q (hay un arco de p a q)
- Sea s un estado sucesor de q (hay un arco de q a s)
- Al eliminar q se rompe el camino existente entre p y s, entonces para que no se modifique $L(M)$ se añade un arco entre p y s etiquetado con las cadenas que recorrían ese camino.
- Gráficamente:



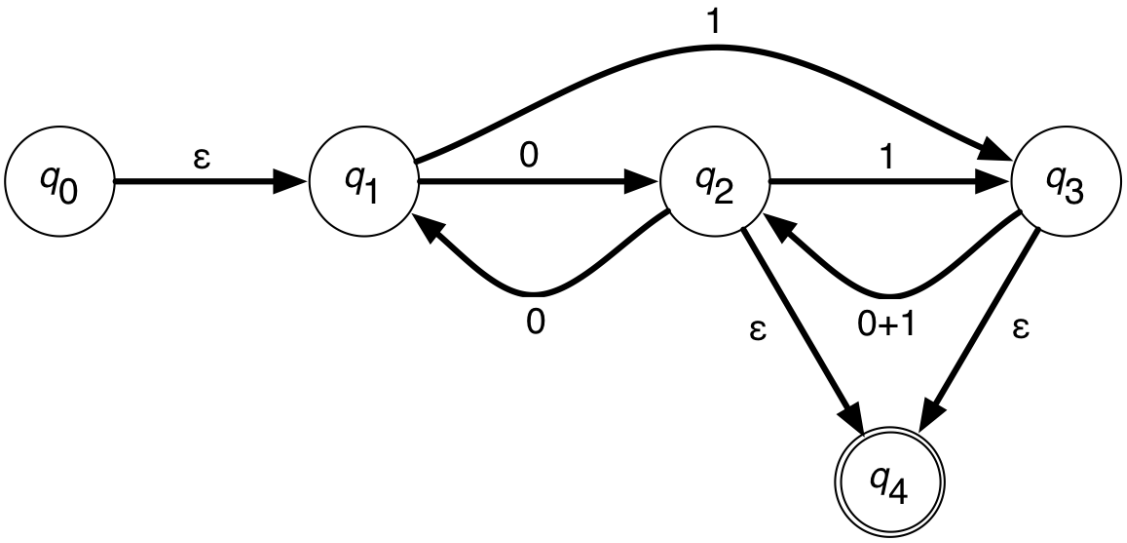
- *Eliminar*(q):
  1. $pred(q) = \{p | p \in Q - \{q\}$ y $\exists p \to q\}$
  2. $suc(q) = \{s | s \in Q - \{q\}$ y $\exists q \to s\}$
  3. Para todo par (p_i, s_j) tal que $p_i \in pred(q)$ y $s_j \in suc(q)$, hacer: $etiqueta'(p_i \to s_j) = P\ Q*\ S + r$, con
     - $P = etiqueta(p_i \to q)$
     - $S = etiqueta(q \to s_j)$
     - $Q = etiqueta(q \to q)$ ó $Q = \epsilon$ si no existe $q \to q$
     - $r = etiqueta(p_i \to s_j)$ ó $r = \emptyset$ si no existe $p_i \to s_j$
  4. Eliminar q y todas sus transiciones del autómata
  5. Añadir todas las nuevas *etiqueta*' al autómata
- $Obtener - ER(AFDM)$:
  1. Construir un AFNDG M' a partir de M
  2. Mientras $|Q'| > 2$, hacer
     - Elegir un estado q de Q'–$\{q_{inicial}, q_{final}\}$
     - $Eliminar(q)$

0.8.7 Autómata Finito No Determinista Generalizado

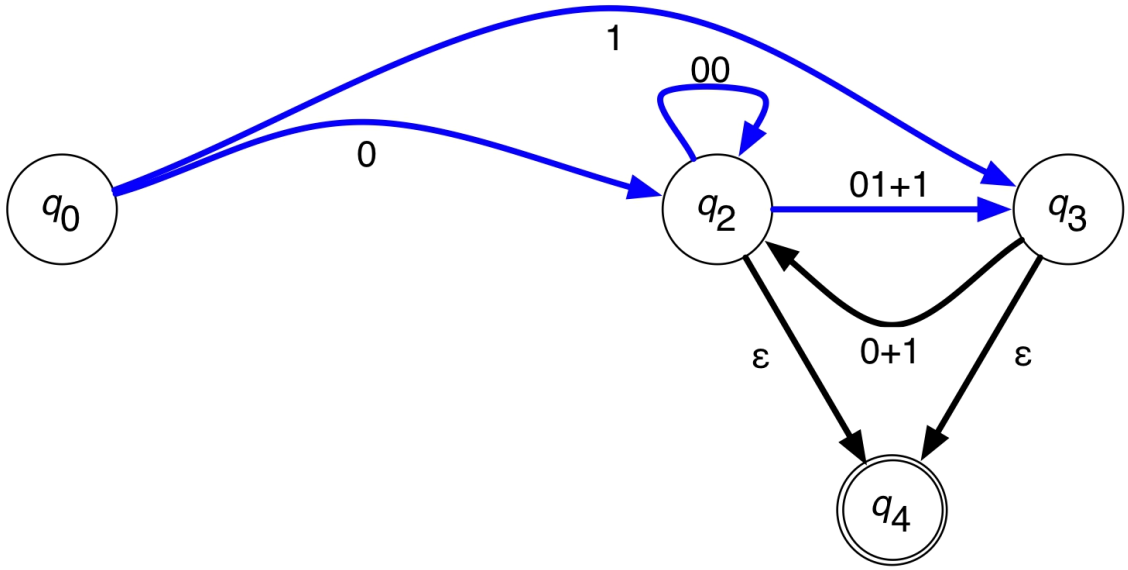
- AFD:



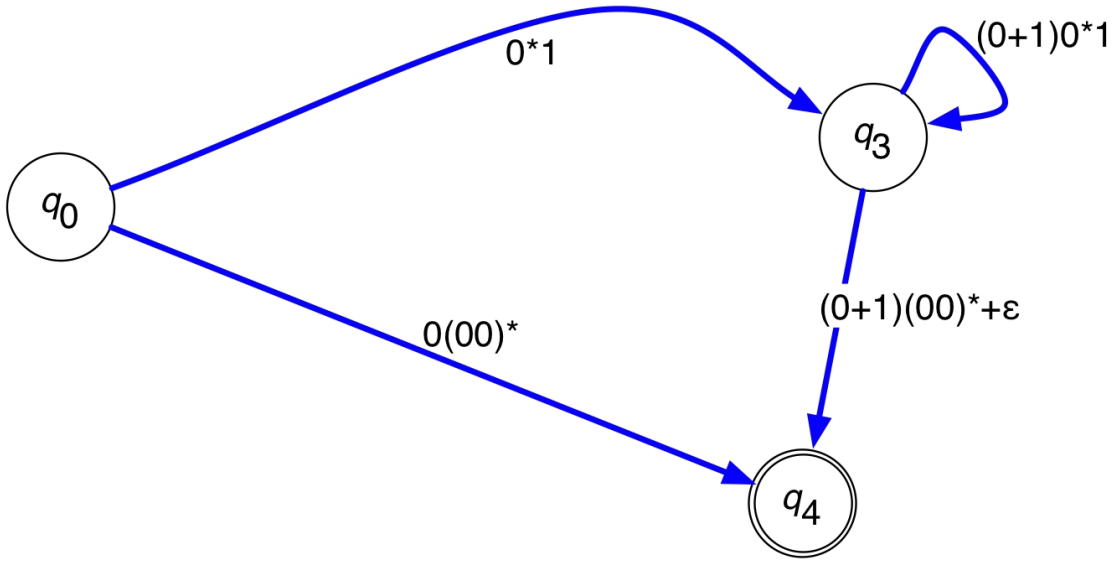
- AFNDG:



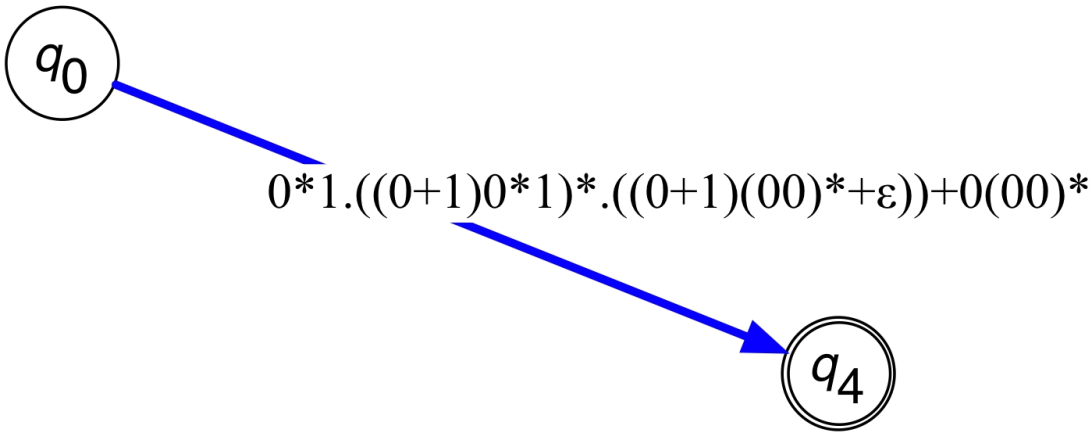
- Eliminar q_1:
  - $pred(q_1) = \{q_0, q_2\}$ $\quad$ $suc(q_1) = \{q_2, q_3\}$
  - $etiqueta'(q_0 \rightarrow q_2) = \epsilon.\epsilon^*.0 + \ = 0$
  - $etiqueta'(q_0 \rightarrow q_3) = \epsilon.\epsilon^*.1 + \ = 1$
  - $etiqueta'(q_2 \rightarrow q_2) = 0.\epsilon^*.0 + \ = 00$
  - $etiqueta'(q_2 \rightarrow q_3) = 0.\epsilon^*.1 + 1 = 01 + 1$



- Eliminar q_2
  - $pred(q_2) = \{q_0, q_3\}$ $\quad suc(q_2) = \{q_3, q_4\}$
  - $etiqueta'(q_0 \to q_3) = 0.(00)^*.(01+1) + 1 = 0^*1$
  - $etiqueta'(q_0 \to q_4) = 0.(00)^*.\epsilon + \ = 0(00)^*$
  - $etiqueta'(q_3 \to q_3) = (0+1).(00)^*.(01+1) + \ = (0+1)0^*1$
  - $etiqueta'(q_3 \to q_4) = (0+1).(00)^*.\epsilon + \epsilon = (0+1)(00)^* + \epsilon$



- Eliminar q_3
  - $pred(q_3) = \{q_0\}$ $\quad suc(q_3) = \{q_4\}$
  - $etiqueta'(q_0 \to q_4) = 0^*1.((0+1)0^*1)^*.((0+1)(00)^* + \epsilon)) + 0(00)^*$



0.8.8 Ecuación Fundamental

- Planteamiento de la ecuación
  - Sea $M = (\{q_1, \cdots, q_n\}, \sum, \delta, q_1, F)$ un AF.
  - Encontrar una expresión regular α tal que $L(\alpha) = L(M)$.
  - Definimos el siguiente conjunto:

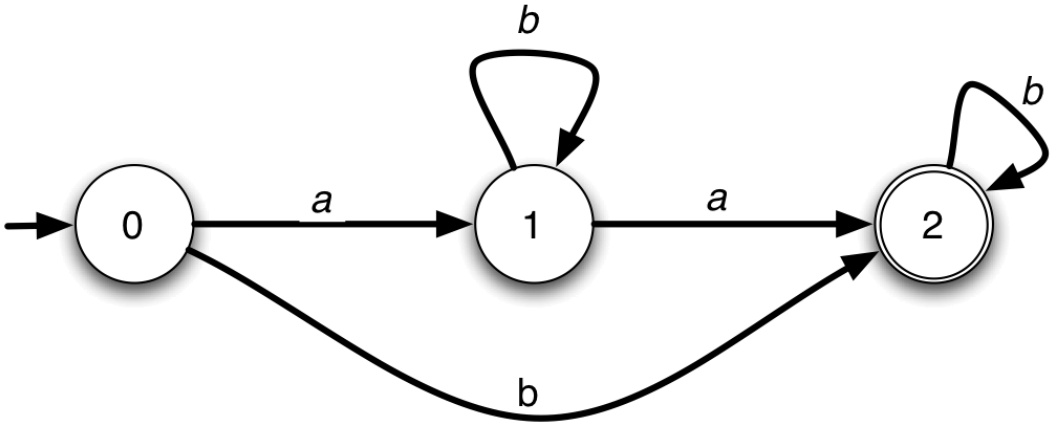
$$X_i = \{x \in \sum^{*} | \delta^*(q_i, x) \in F\}$$

  - Sea $\{a_{ij}^1, \cdots, a_{ij}^{pj}\}$ el conjunto de símbolos pertenecientes a $\sum \cup \{\epsilon\}$, tales que existe una transición desde q_i a q_j etiquetada con cada uno de ellos.
- Entonces, el conjunto X_i puede describirse de la siguiente forma:

$$X_i = \begin{cases} (a_{i0}^1 + \cdots + a_{i0}^{p0})X_0 + \cdots + (a_{in}^1 + \cdots + a_{in}^{pn})X_n & \text{si } q_i \notin F \\ (a_{i0}^1 + \cdots + a_{i0}^{p0})X_0 + \cdots + (a_{in}^1 + \cdots + a_{in}^{pn})X_n + \varepsilon & \text{si } q_i \in F \end{cases}$$

Sistemas de equaciones:

- Para cada estado q_i se plantea una ecuación X_i.



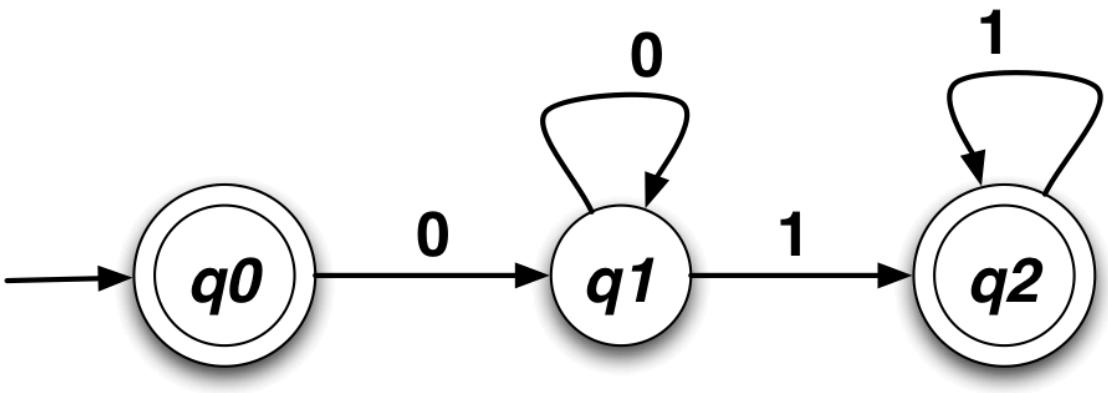
- Ejemplo:

- Ecuaciones obtenidas:
  - $X_0 = aX_1 + bX_2$
  - $X_1 = bX_1 + aX_2$
  - $X_2 = bX_2 + \epsilon$
- Dado que por definición $X_0 = L(M)$, nuestro objetivo es resolver el sistema de ecuaciones y obtener la solución para X_0.

Resolución del sistema de ecuaciones

- *Lema de Arden.* Una ecuación $X = AX + B$, tiene como solución $X = A^*B$. Si $\epsilon \notin A$, entonces $X = A^*B$ es la única solución.

- $X_0 = aX_1 + bX_2$
- $X_1 = bX_1 + aX_2$
- $X_2 = bX_2 + \epsilon$
- $X_2 = b^*(\epsilon) = b^*$
- $X_1 = bX_1 + ab^* = b^*ab^*$
- $X_0 = ab^*ab^* + bb^*$



Ejemplo 2:

- $q_0 = 0q_1 + \epsilon$
- $q_1 = 0q_1 + 1q_2$
- $q_2 = 1q_2 + \epsilon$
- $q_2 = 1^*$
- $q_1 = 0q_1 + 1.1^*$
- $q_1 = 0^*11^*$
- $q0 = 00^*11^* + \epsilon$

0.9 Lema del Bombeo

- El lema de bombeo para lenguajes regulares enuncia una propiedad que cumplen todos los lenguajes regulares infinitos (y también algunos lenguajes que no son regulares). Gracias a este lema podremos demostrar que ciertos lenguaje infinitos no son regulares. Es importante

hacer notar que el lema de bombeo es una herramienta adecuada para demostrar que un lenguaje no es regular, pero no lo será para demostrar que un lenguaje si es regular (por el hecho de que existen algunos lenguajes no regulares que la cumplen). Por tanto, si un lenguaje no cumple el lema de bombeo no es regular, pero si lo cumple no podremos decir si es o no regular.

0.9.1 Enunciado del Lema de Bombeo

- Para todo lenguaje regular infinito L, existe una constante n, dependiente de ese lenguaje, de forma que si w es una cadena de L con $|w| \geq n$, podemos partir w en tres cadenas, x, y, z, de forma que:
  - $w = xyz$,
  - $y \neq \epsilon$ (o dicho de otro modo, que $|y| \geq 1$),
  - $|xy| \leq n$
  - Para cualquier $k \geq 0$, la cadena $xykz$ pertenece a L.

Más formalmente:

- $\forall$ lenguaje regular infinito L sobre un alfabeto $\sum$
- $\exists n \in N$ /
- $\forall w \in L$ / $|w| \geq n$
- $\exists x, y, z \in {}^{*}$ / $w = xyz, y \neq \epsilon, |xy| \leq n$,
- $\forall k \geq 0, xykz \in L$

- O sea que para cualquier cadena de L lo bastante larga, siempre podremos encontrar una partición en tres subcadenas, con una no vacía en el medio (la y) que no está demasiado lejos del comienzo de la palabra, que podremos "bombear"; es decir, que si se repite la subcadena y cualquier número de veces, la cadena resultante también pertenecerá a L.

0.9.2 Demostrar que un Lenguaje no es Regular

- Dado que para todo lenguaje regular infinito se cumple el lema de bombeo, si nos dan un lenguaje infinito y demostramos que para él no se cumple, habremos demostrado que no es un lenguaje regular. Como el lema de bombeo es una propiedad que se cumple para todas las cadenas de longitud mayor o igual a cierta n, bastará encontrar una cadena de ese lenguaje, de longitud mayor o igual a esa n, que no se pueda "bombear" para demostrar que el lenguaje no es regular

Con esta idea en mente, los pasos a dar para demostrar que un lenguaje dado no es regular son los siguientes: 1. Elegir una palabra w que pertenezca al lenguaje dado. Podemos elegir cualquier palabra del lenguaje, pero debe ser una cuya longitud sea mayor o igual que una constante n que desconocemos (la constante del lema de bombeo). Como desconocemos n, lo habitual será elegir una palabra en función de un n cualquiera y cuya longitud sea mayor o igual que n.

2. El lema de bombeo dice que si el lenguaje fuera regular, podríamos encontrar una forma de partir esa palabra w en tres, cumpliendo ciertas restricciones, y que esa partición sería bombeable. Como queremos demostrar que el lenguaje no es regular, tendremos que demostrar que

no hay ninguna forma de partir la palabra en tres cumpliendo las restricciones del lema, y que después se pueda bombear siempre.
3. Finalmente bastará con encontrar una constante $k \geq 0$ que haga que ninguna de las particiones posibles de w sea bombeable.

Más formalmente, para demostrar que un lenguaje L sobre un alfabeto Σ no es regular habrá que demostrar que:

- $\forall n \in N$
- $\exists w \in L \ / \ |w| \geq n$,
- $\forall x, y, z \in \sum^* \ / \ w = xyz, y \neq \epsilon, |xy| \leq n$,
- $\exists k \geq 0 \ / \ xy^k z \notin L$

0.9.3 Ejemplo:

Sea el lenguaje $L = \{a^{2n}b^n \mid n \geq 0\}$. Demostrar que L no es regular

1. Voy a suponer que el lenguaje es regular. Si lo es, y como es infinito, para él se cumplirá el lema de bombeo. Sea por tanto $n \in N$ la constante del lema de bombeo para L (constante que no conozco). Elijo una palabra que pertenezca a L y de longitud mayor o igual a $n : w = a^{2n}b^n$, tenemos que $w \in L$ y $|w| = 3n$ y por tanto $|w| \geq n$, sea cual sea n.
2. Tengo que encontrar todas las formas de partir la palabra elegida w en tres xyz que cumplan las restricciones del lema de bombeo:
   - $w = xyz$
   - $y \neq \epsilon$
   - $|xy| \leq n$ (siendo n la constante del lema)
3. Si me fijo en la palabra w que he elegido, cualquier x, y, z que cumplan las condiciones (restricciones) del lema serán de la siguiente forma:
   - $x = a^i$
   - $y = a^j$
   - (con $j \geq 1$ puesto que $y \neq \epsilon$, con $i + j \leq n$ puesto que $|xy| \leq n$)
   - $z = a^{2n-i-j}b^n$

- Se que la x y la y estarán formadas sólo por aes porque la palabra w que he elegido, y que estoy partiendo, tiene $2n$ aes al principio y la longitud de xy es menor o igual que n. También se las restricciones que cumplen sus índices (i, j) porque me las impone el lema de bombeo. Se puede ver además que obviamente $w = xyz$, porque $xyz = a^i a^j a^{2n-i-j} b^n = a^{2n}b^n = w$.

4. Debo encontrar ahora una constante $k \geq 0$ con la que ninguna de las posibles particiones de w que hemos encontrado en el punto anterior sea bombeable. Si elijo $k = 2$ y bombeo las x, y, z encontradas en el punto anterior para esa constante, tendré que:
   - $xy^2z = a^i a^{2j} a^{2n-i-j} b^n = a^{2n+j}b^n$ (para cualquier i y j, o sea para cualquiera de las particiones "legales" de la w elegida según el lema de bombeo).

- Pero como $j \geq 1$ (ver punto 2, es una de las restricciones del lema) tengo que $xy^2z = a^{2n+j}b^n$, es una palabra que no pertenece al lenguaje porque tiene más del doble de aes que de bs (al menos una a más).
- He llegado por tanto a una contradicción (una palabra que no es bombeable de ninguna forma para al menos una constante k en un lenguaje supuestamente regular) que viene de suponer

precisamente que el lenguaje L es regular, luego L no es regular.