Lenguajes Regulares

Departamento de Informática e Ingeniería de Sistemas
C.P.S. Universidad de Zaragoza

Última revisión: Feb. 2003

Índice

- Problema de especificación de lenguajes
- Lenguajes regulares
- Expresiones regulares

Especificación de lenguajes: Introducción

- Hasta ahora ha sido sencillo especificar qué cadenas pertenecen a un determinado lenguajes sobre algún alfabeto Σ.

$$L_{dígitos} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$$

- Objetivo:

Encontrar formalismos que me permitan definir lenguajes más complejos.

- ¿Cuántos lenguajes diferentes puedo definir sobre un determinado alfabeto Σ?
  - Cálculo del número de palabras del lenguaje universal Σ*.
  - Cálculo del número de subconjuntos, en este caso sublenguajes, que puedo formar con las palabras del lenguaje universal Σ*.

Especificación de lenguajes (II)

- Teorema: Para todo Σ, Σ* es infinito numerable.
  - Consideremos el alfabeto Σ = {a, b}
    - Orden lexicográfico:

      ε, a, b, aa, ab, ba, bb, aaa, ...
    - Numeramos las palabras por orden lexicográfico:

ε	0
a	1
b	2
aa	3
ab	4
ba	5
...	

  - Conclusión (informal): dado que hemos construído una función biyectiva de Σ* a N, se tiene que Σ* es infinito numerable.

Especificación de lenguajes (III)

- Teorema: El conjunto de todos los lenguaje sobre Σ no es numerable.

  Demostración: Supongamos que el conjunto de todos los lenguajes sobre Σ (L) es numerable. Si es numerable $L = \{A_0, A_1, A_2,...\}$

  Como Σ^* es numerable, $\Sigma^*=\{w_0, w_1, w_2,...\}$

  Sea $B = \{w_i \mid w_i \notin A_i\}$, palabras que no pertenecen al lenguaje con su mismo índice. B es un lenguaje sobre Σ luego $B = A_k$ para algún k.

  Si $w_k \in B$, entonces $w_k \notin A_k$ (= B), luego $w_k \notin B$ 

  Si $w_k \notin B$, entonces $w_k \in A_k$ (= B), luego $w_k \in B$. 

  Conclusión: La suposición de que el conjunto de todos los lenguajes sobre Σ es numerable es falsa. Luego el conjunto es no numerable.

Especificación de lenguajes: Conclusiones

¿ Cuál es la magnitud del problema de especificar lenguajes sobre un Σ ?

¿ Existe algún método suficientemente expresivo capaz de especificar todos los lenguajes sobre un Σ ?

¿ Con qué método podemos especificar todos esos lenguajes ?

Lenguajes regulares

- Sea el alfabeto Σ. El conjunto de lenguajes regulares se define de manera recursiva como sigue:

  1) ϕ es un lenguaje regular.
  2) $\{\varepsilon\}$ es un lenguaje regular.
  3) Para todo $a \in \Sigma$, $\{a\}$ es un lenguaje regular.
  4) Si A y B son lenguaje regulares, entonces $A \cup B$, $A \cdot B$ y A^* son lenguajes regulares.
  5) Ningún otro lenguaje sobre Σ es regular

Lenguajes regulares: ejemplos.

- Dado $\Sigma = \{a,b\}$, las siguientes afirmaciones son ciertas:

  ϕ y $\{\varepsilon\}$ son lenguajes regulares

  $\{a\}$ y $\{b\}$ son lenguajes regulares

  $\{a, b\}$ es un lenguaje regular

  $\{ab\}$ es un lenguaje regular

  $\{a, ab, b\}$ es un lenguaje regular

  $\{a^i \mid i \geq 0\}$ es un lenguaje regular

  $\{a^i b^j \mid i \geq 0 \text{ y } j \geq 0\}$ es un lenguaje regular

  $\{(ab)^i \mid i \geq 0\}$ es un lenguaje regular

Lenguajes regulares: ejemplos (II)

- Dado $\Sigma = \{a,b,c\}$, ¿El lenguaje de todas las cadenas sobre el Σ que no contiene ninguna subcadena *ac* es regular?

  $\{c\}$, $\{a\}$, $\{b\}$ son L.R. (3)

  $\{c\}^*$ es un L.R. (4, *)

  $\{b\}\{c\}^*$ es un L.R. (4, ·)

  $\{a\}\cup\{b\}\{c\}^*$ es un L.R. (4, $\cup$)

  $(\{a\}\cup\{b\}\{c\}^*)^*$ es un L.R. (4, *)

  $\{c\}^*(\{a\}\cup\{b\}\{c\}^*)^*$ es un L.R. (4, ·)

  ¿Es este lenguaje el que buscábamos?

- Resulta engorroso construir y especificar lenguajes regulares.

Expresiones regulares

- Las expresiones regulares permiten representar de manera simplificada un lenguaje regular.
- Las expresiones regulares sobre un Σ y el lenguaje que denotan se define recursivamente como:

1) ϕ es una e.r y denota el lenguaje vacío
2) ε es una e.r y denota el lenguaje $\{\varepsilon\}$
3) Para cada $a \in \Sigma$, a es una e.r y denota el lenguaje $\{a\}$
4) Si α, β son dos e.r que denotan los lenguajes A y B, entonces:
   1) $\alpha + \beta$ es una e.r que denota $A \cup B$
   2) $\alpha\beta$ es una e.r que denota $A \cdot B$
   3) α^* es una e.r que denota A^*
5) Orden de precedencia operadores

   * · + (se pueden usar paréntesis)

Expresiones regulares: Ejemplos

- Dado el alfabeto $\Sigma = \{a,b,c\}$:
  - Lenguaje Universal Σ^*

    $(a+b+c)^*$
  - L = { w | |w| es múltiplo de 3}

    $((a+b+c)\ (a+b+c)\ (a+b+c))^*$
  - L = { w | $|w|_a$ es par}

    $((b+c)^*\ a\ (b+c)^*\ a\ (b+c)^*)^* + (b+c)^*$
  - L = { w | w contiene la subpalabra 'abc'}

    $(a+b+c)^*\ abc\ (a+b+c)^*$

- Nota: Usaremos la notación $|w|_a$ para denotar el número de ocurrencias del símbolo a en la palabra w

Expresiones regulares: Equivalencia

- Cuando sea necesario diferenciar entre expresión regular *r* y el lenguaje denotado por la misma, usaremos L(r) para denotar el lenguaje.
- Dos expresiones regulares *r* y *s* sobre un mismo Σ se dice que son equivalentes, si $L(r) = L(s)$

  Ejemplo:

  r = (a*b)*

  s = ε+(a+b)*b

  L(*r*) = L(*s*) = {palabras con 0 o más a'es y b'es que son la palabra vacía o tienen una b al final}

Expresiones Regulares: Ejemplo de Equivalencia

- Sean r, s dos e.r, ¿Son equivalentes las siguientes expresiones regulares?

$$r\,(sr)^* = (rs)^*\,r$$

- Conclusión: Se podrá simplificar expresiones regulares reemplazándolas por otras por otras equivalentes pero menos complejas.

Expresiones regulares: Simplificación

- Teorema: Sean r, s, t expresiones regulares sobre un mismo alfabeto Σ. Entonces:

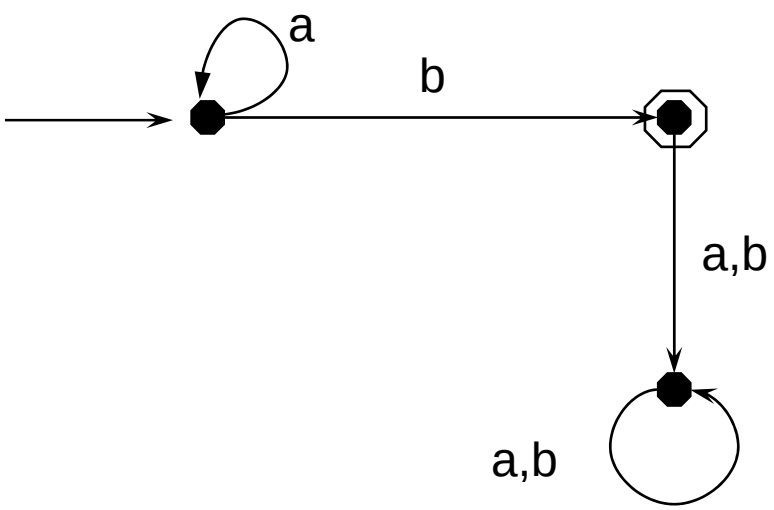
1. $r + s = s + r$
2. $r + \phi = r = \phi + r$
3. $r + r = r$
4. $(r + s) + t = r + (s + t)$
5. $r\varepsilon = \varepsilon r = r$
6. $r\phi = \phi r = \phi$
7. $(rs)t = r(st)$
8. $r(s+t) = rs+rt$, y $(r+s)t = rt+st$
9. $r^* = r^{**} = r^*r^* = (\varepsilon+r)^* = r^*(r+\varepsilon) = (r+\varepsilon)r^* = \varepsilon+rr^*$
10. $(r+s)^* = (r^*+s^*)^* = (r^*s^*)^* = (r^*s)^*r^* = r^*(sr^*)^*$
11. $r(sr)^* = (rs)^*r$
12. $(r^*s)^* = \varepsilon+(r+s)^*s$
13. $(rs^*)^* = \varepsilon+r(r+s)^*$
14. $s(r+\varepsilon)^*(r+\varepsilon)+s = sr^*$
15. $rr^* = r^*r$

Reconocimiento de Palabras

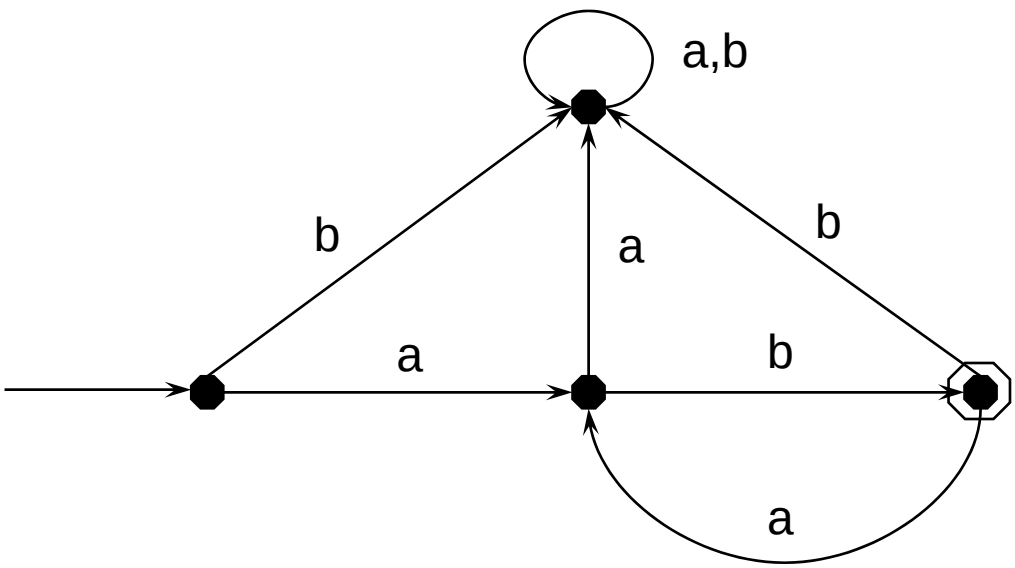
- Objetivo: Construir diagramas que nos ayuden a determinar si una palabra w pertenece a un lenguaje L.
- Diagramas de Transición:
  - Tienen forma de grafo dirigido con información adicional.
  - Dos tipos de elementos:
    - Nodos (estados): representan hasta que lugar ha sido reconocida la cadena.
    - Aristas (transiciones) etiquetadas con símbolos del Σ: si el siguiente carácter de la palabra corresponde con el símbolo de alguna de las aristas de salida del nodo actual, nos desplazamos al estado al que nos lleva esa arista.
  - Dos tipos de Nodos (estados) especiales:
    - Nodo Inicial: donde comenzamos el reconocimiento de la palabra.
    - Nodos Final o de Aceptación: aquellos que determinan que una palabra es “legal”.

Reconocimiento de Palabras: Ejemplos

- $L = \{a^k b \mid k \geq 0\}$ sobre $\Sigma = \{a,b\}$

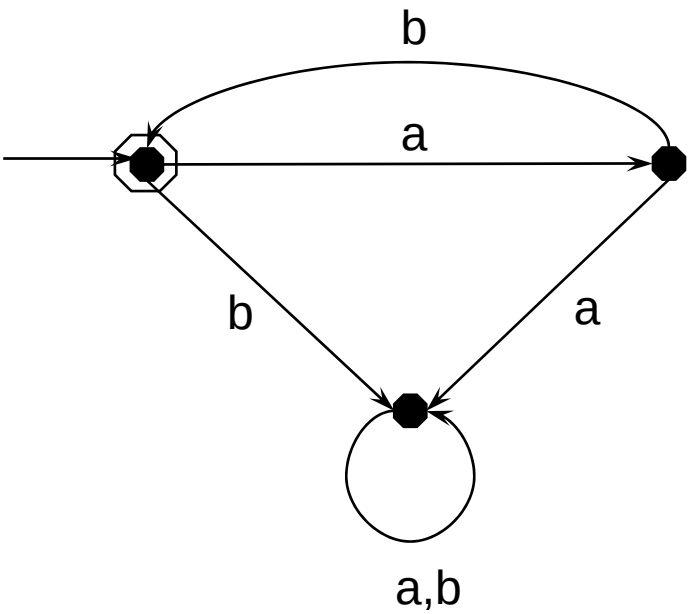


- $L = \{(ab)^i \mid i \geq 1\}$ sobre $\Sigma = \{a,b\}$



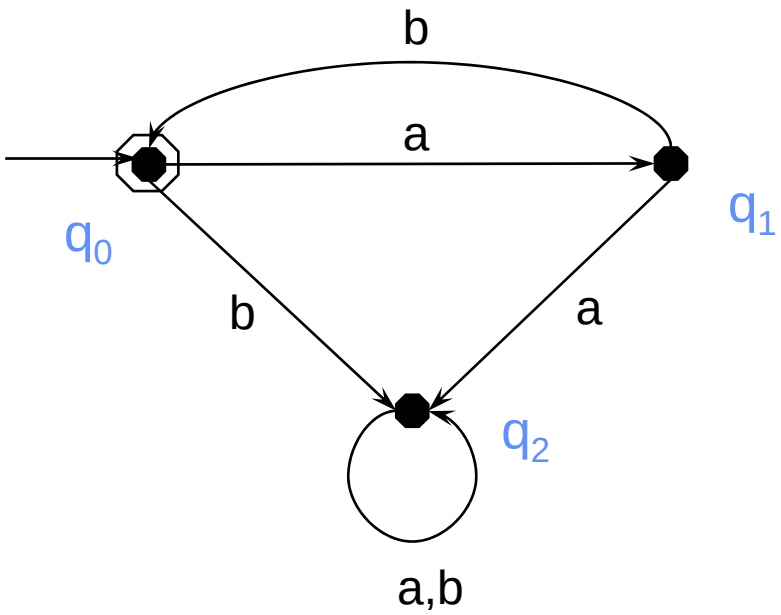
Reconocimiento de Palabras: Ejemplos(II)

- $L = \{(ab)^i \mid i \geq 0\}$ sobre $\Sigma = \{a,b\}$



Reconocimiento de Palabras: Ejemplos(II)

- $L = \{(ab)^i \mid i \geq 0\}$ sobre $\Sigma = \{a,b\}$



Podemos representar el Diagrama de Transición por medio de una tabla:

Estado Actual	Entrada: a	Entrada: b
q_0	q_1	q_2
q_1	q_2	q_0
q_2	q_2	q_2

Autómata Finito Determinista

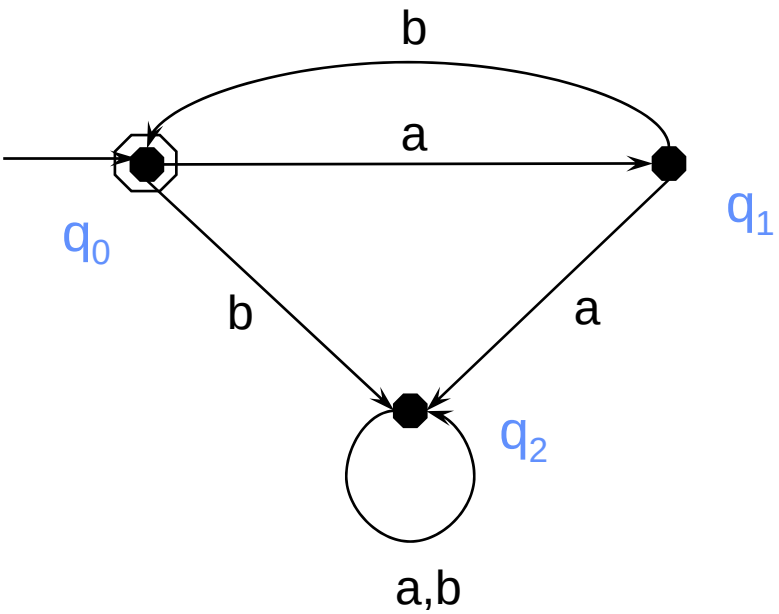
- Un Autómata Finito Determinista ***M*** es una colección de cinco elementos:
  - Un alfabeto Σ de entrada.
  - Una colección Q finita de estados.
  - Un estado inicial s ($s \in Q$).
  - Una colección F de estados finales o de aceptación ($F \subseteq Q$).
  - Una función δ: $Q \times \Sigma \rightarrow Q$ que determina el ÚNICO estado siguiente para el par (q_i, σ) correspondiente al estado actual y a la entrada.

- Abreviatura de un AFD:

$$M = (\Sigma, Q, s, F, \delta)$$

Autómata Finito Determinista: Ejemplos

- $L = \{(ab)^i \mid i \geq 0\}$ sobre $\Sigma = \{a,b\}$



$M = (\Sigma, Q, s, F, \delta)$

$\Sigma = \{a,b\}$

$Q = \{q_0, q_1, q_2\}$

$s = q_0$

$F = \{q_0\}$

δ	a	b
q_0	q_1	q_2
q_1	q_2	q_0
q_2	q_2	q_2

Autómata Finito Determinista: Ejemplos (II)

- Sea M = (Σ, Q, s, F, δ),

  Σ = {a, b}

  Q = {q_0, q_1}

  s = q_0

  F = {q_0}

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_0

- Sea M = (Σ, Q, s, F, δ),

  Σ = {a, b}

  Q = {q_0, q_1 , q_2, q_3}

  s = q_0

  F = {q_0, q_1 , q_2}

δ	a	b
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_0	q_3
q_3	q_3	q_3

¿Cuáles son sus diagramas de transición?

¿Qué lenguaje acepta cada autómata?

Autómatas Finitos Deterministas y Lenguajes: Definiciones

- Sea M un AFD, el lenguaje aceptado por M es,

$$L(M) = \{w \in \Sigma^* \mid w \text{ es aceptada por } M\}$$

es decir, el conjunto de cadenas que hace que M pase de su estado inicial a uno de sus estados de aceptación.

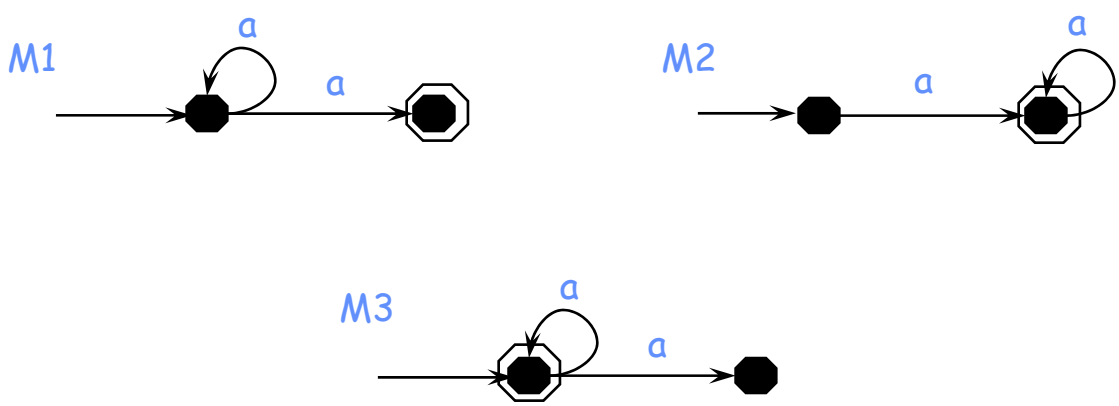
- Ejemplo (AFD ejemplo anterior):

$L(M) = \{w \in \{a,b\}^* \mid w \text{ no contiene tres b's consecutivas}\}$

- Sean M1 y M2 dos AFD, se dice que son equivalentes, si

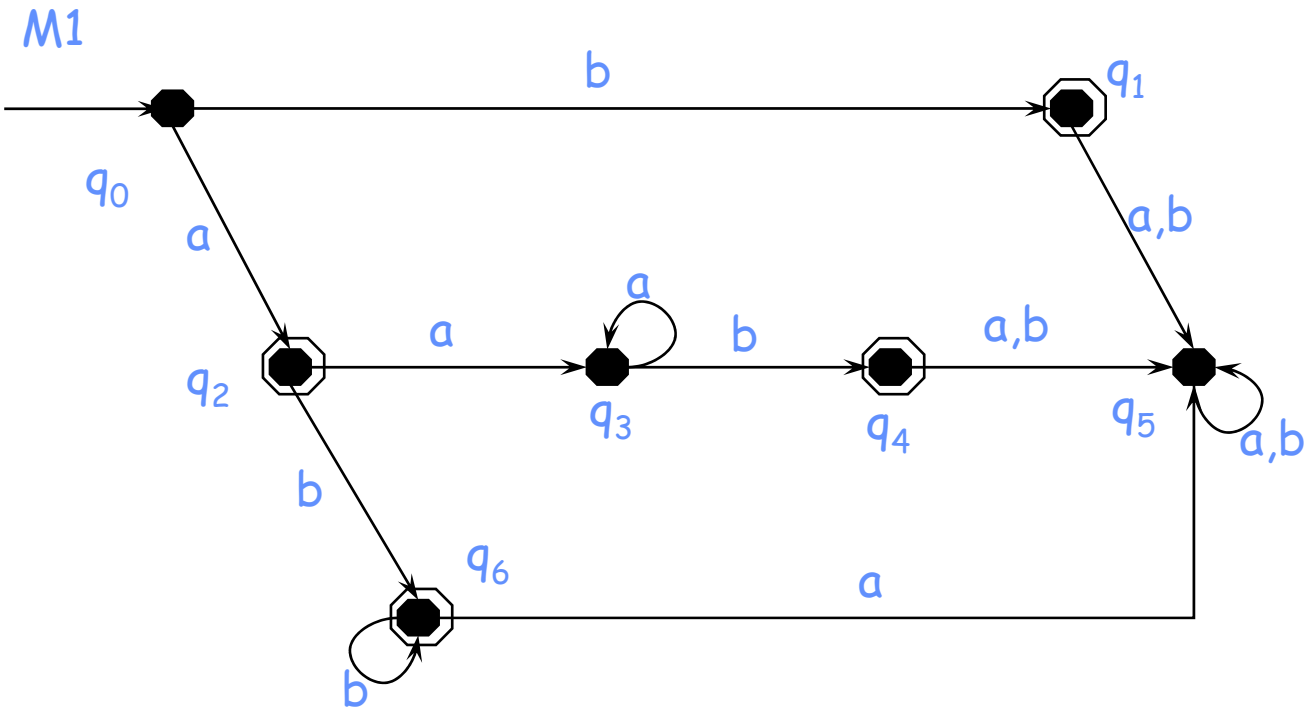
$$L(M1) = L(M2)$$

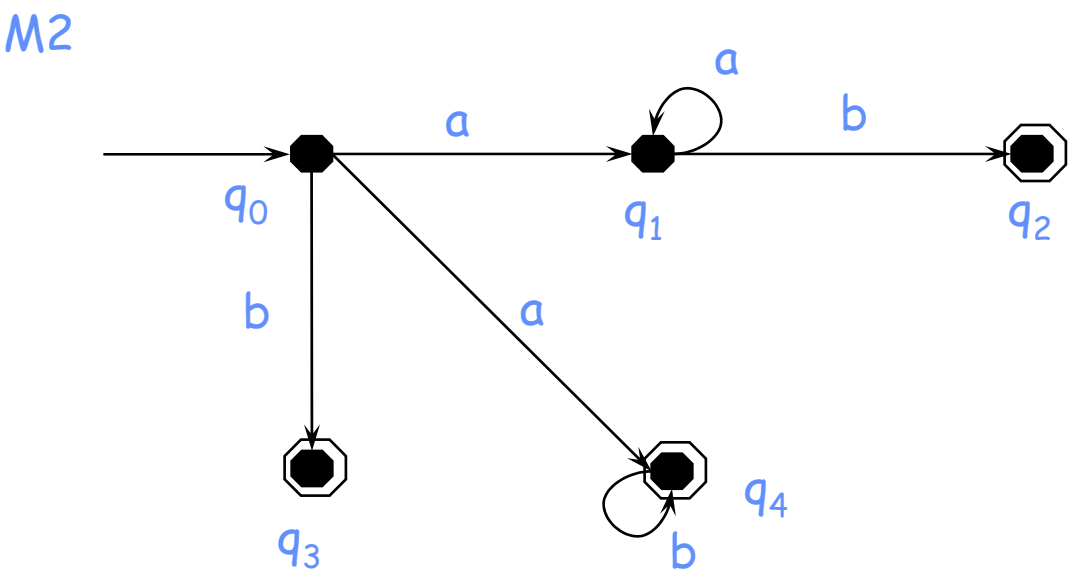
- Ejemplo: Sea $\Sigma = \{a\}$, ¿son equivalentes Mi?



Autómata Finito No Determinista

- ¿Los autómatas M1 y M2 reconocer el mismo lenguaje L?





Autómata Finito No Determinista

- Ambos autómatas reconocen el lenguaje

  a*b + ab*

- Entonces, ¿cuáles son las diferencias entre M1 y M2 ?:
  - Cada estado de M1 tiene una única transición para cada símbolo del Σ (δ es una función).
  - Cada estado de M2 tiene cero, una o más transiciones para cada símbolo del Σ (regla de transición) ⇒ Indeterminismo.

- En ocasiones resulta más conveniente diseñar autómatas finitos No deterministas (como por ejemplo, M2)

Autómata Finito No Determinista

- Un Autómata Finito No Determinista ***M*** es una colección de cinco elementos:
  - Un alfabeto Σ de entrada.
  - Una colección Q finita de estados.
  - Un estado inicial s ($s \in Q$).
  - Una colección F de estados finales o de aceptación ($F \subseteq Q$).
  - Una relación Δ: $Q \times \Sigma \rightarrow Q$ que se llama relación de transición,

$$\Delta(q_i, \sigma) \subseteq Q, \text{ tal que } q_i \in Q \text{ y } \sigma \in \Sigma$$

- Abreviatura de un AFN:

$$M = (\Sigma, Q, s, F, \Delta)$$

Autómata Finito No Determinista: Ejemplos

- Sea el AFN $M = (\Sigma, Q, s, F, \Delta)$:

$\Sigma = \{a, b\}$

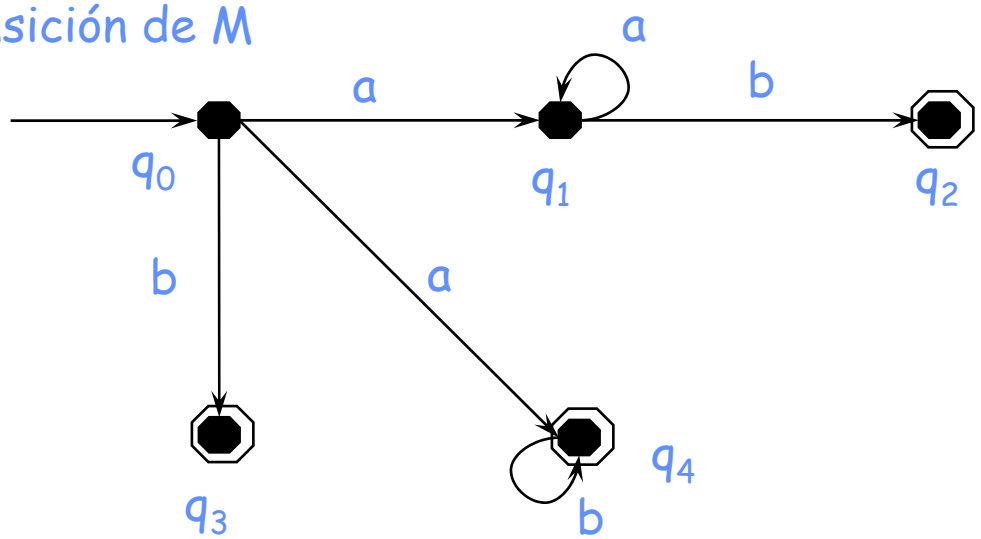
$Q = \{q_0, q_1, q_2, q_3, q_4\}$

$s = q_0$

$F = \{q_2, q_3, q_4\}$

Δ	a	b
q_0	$\{q_1, q_4\}$	$\{q_3\}$
q_1	$\{q_1\}$	$\{q_2\}$
q_2	ϕ	ϕ
q_3	ϕ	ϕ
q_4	ϕ	$\{q_4\}$

Diagrama de Transición de M



Autómata Finito No Determinista: Ejemplos (II)

- Sea el AFN *M* = (Σ, Q, s, F, Δ):

  $\Sigma = \{a, b\}$

  $Q = \{q_0, q_1, q_2\}$

  $s = q_0$

  $F = \{q_0\}$

Δ	a	b
q_0	$\{q_1\}$	ϕ
q_1	ϕ	$\{q_0, q_2\}$
q_2	$\{q_0\}$	ϕ

- ¿Cuál es el diagrama de transición que representa el AFN *M*?
- ¿Cuál es lenguaje regular que acepta el AFN *M*?
- ¿Pertenece la palabra 'aba' al lenguaje regular aceptado por el AFN *M*?

Autómatas Finitos No Deterministas y Lenguajes: Definiciones

- Sea M un AFN, el lenguaje aceptado por M es,

$$L(M) = \{w \in \Sigma^* \mid w \text{ es aceptada por } M\}$$

,es decir, el conjunto de cadenas que hace que M pase de su estado inicial a uno de sus estados de aceptación.

- Dado su carácter indeterminista, para afirmar que una cadena w no está en L(M) debemos agotar todas las formas posibles de recorrer el diagrama para dicha cadena.

Equivalencia entre AFN y AFD

- Dados dos autómatas finitos M y M', independientemente si son AFD o AFN, se dice que son equivalentes si L(M) = L(M').
- Los AFN no son más potentes que los AFD respecto a los lenguajes que aceptan.
- Objetivo: Demostrar que todo lenguaje aceptado por un AFN es también aceptado por un AFD equivalente.
  - Subobjetivo: A partir de un AFN, obtener el AFD equivalente que reconoce el mismo lenguaje.

Equivalencia entre AFN y AFD

- Sea $M=(\Sigma, Q, s, F, \Delta)$ un AFN. Definimos un AFD $M'=(\Sigma', Q', s', F', \delta)$ equivalente tal que:
  - $\Sigma' = \Sigma$
  - $Q' = 2^Q$ (colección de todos los subconjuntos de Q)
  - $s' = \{s\}$
  - F' = colección de todos los subconjunto de Q con algún estado en F
  - Definimos δ como,

    $\delta(\{q_{i1}, q_{i2}, ..., q_{in}\}, \sigma) = \{p_1, p_2, ..., p_k\}$

    tal que,

    $\Delta(\{q_{i1}, q_{i2}, ..., q_{in}\}, \sigma) = \{p_1, p_2, ..., p_k\}$

    $$\delta: Q' \times \Sigma \rightarrow Q'$$

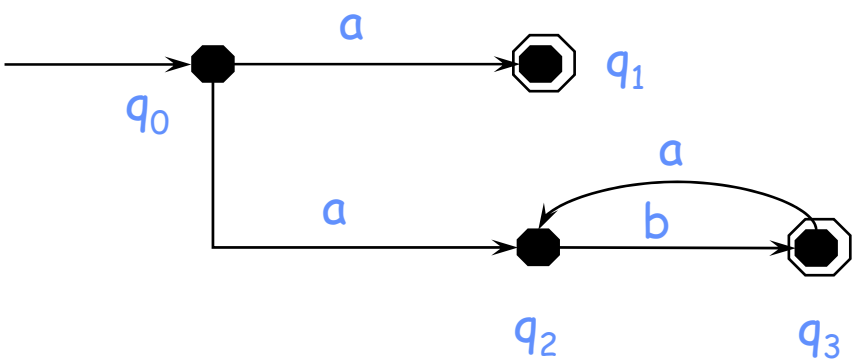
    Notad que tenemos que ampliar la definición de Δ:

    si $X \subseteq Q$, y $\sigma \in \Sigma$, $\Delta(X, \sigma) = \cup_{(q \in X)} \Delta(q, \sigma)$

- En la práctica, no todos los posibles subconjuntos de Q son estados accesibles desde el estado inicial. Añadir sólo aquellos que sean el resultado de una transición desde un estado previamente añadido.

Equivalencia entre AFN y AFD: Ejemplo

- Ejemplo: Sea el AFN M con el siguiente diagrama de transición:



$$L(M) = a + (ab)^+$$

- Para el AFN tenemos,

$\Delta(q_0, a) = \{q_1, q_2\}$

$\Delta(q_0, b) = \phi$

$\Delta(\{q_1,q_2\}, a) = \phi$

$\Delta(\{q_1,q_2\}, b) = \{q_3\}$

$\Delta(\phi, a) = \Delta(\phi, b) = \phi$

$\Delta(\{q_3\}, a) = \{q_2\}$

$\Delta(\{q_3\}, b) = \phi$

$\Delta(\{q_2\}, a) = \phi$

$\Delta(\{q_2\}, b) = \{q_3\}$

Equivalencia entre AFN y AFD: Ejemplo

- A partir de Δ definimos el AFD equivalente,

  $\Sigma' = \Sigma = \{a, b\}$

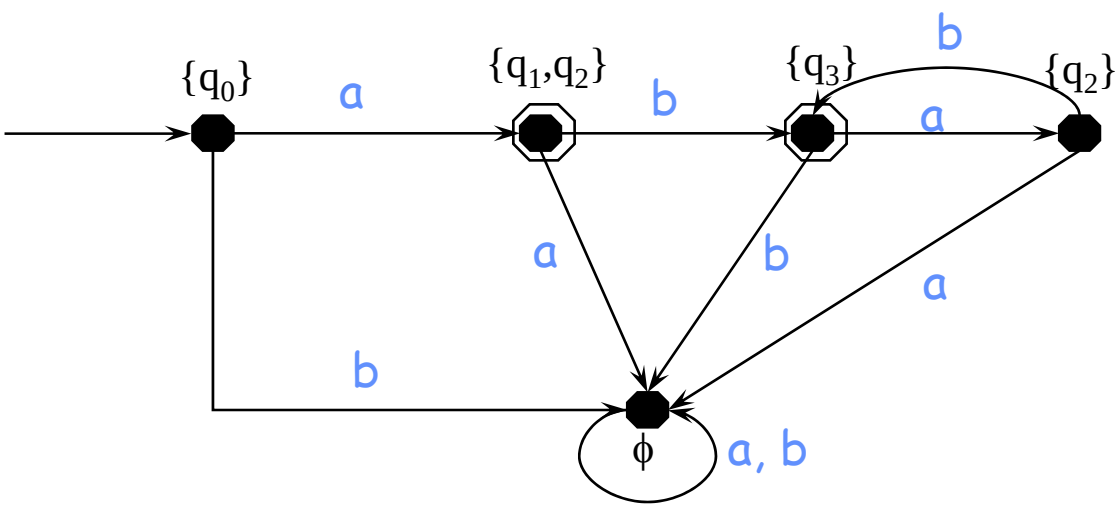
  $Q' = \{\phi, \{q_0\}, \{q_1,q_2\}, \{q_2\},\{q_3\}\}$

  $s' = \{q_0\}$

  $F' = \{\{q_1,q_2\}, \{q_3\}\}$

  δ definida tal que,

δ	**a**	**b**
ϕ	ϕ	ϕ
$\{q_0\}$	$\{q_1,q_2\}$	ϕ
$\{q_1,q_2\}$	ϕ	$\{q_3\}$
$\{q_2\}$	ϕ	$\{q_3\}$
$\{q_3\}$	$\{q_2\}$	ϕ



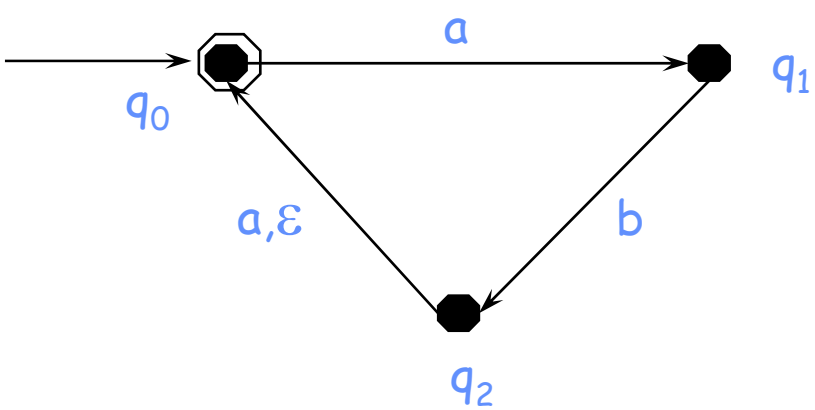
Equivalencia entre AFN y AFD

- Teorema: Sea $M=(\Sigma, Q, s, F, \Delta)$ un AFN. Entonces, existe un AFD $M'=(\Sigma', Q', s', F', \delta)$ que es equivalente a M.

  Idea de la demostración: Construir un AFD a partir de un AFN aplicando el método de transformación visto, probar que reconoce las mismas palabras y verificar que es un AFD.

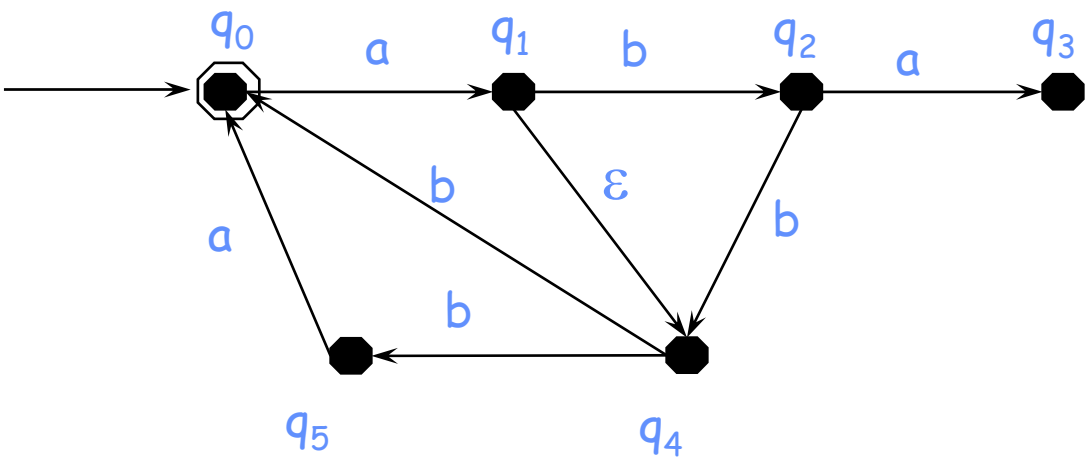
ε-Transiciones

- Podemos ampliar la definición de los AFN añadiendo nuevos elementos no deterministas.
- Las ε-Transiciones son un tipo de transiciones entre dos estados que no consumen ningún símbolo de entrada.
- La decisión de elegir una ε-Transiciones se realiza de la misma forma que la elección de una transición con elección múltiple para un símbolo de entrada.
  - Basándose en algo que no determina el modelo.



ε-Transiciones

- Problema: Dado un AFN con ε-Transiciones ¿Cómo calcular el conjunto de los estados siguientes?
  - Hay que tener en cuenta las ε-Transiciones "anteriores" y "posteriores" a la transición etiquetada con el símbolo de entrada σ.
- Ejemplo:



$\Delta(q_0, a) = \{q_1, q_4\}$

$\Delta(q_1, b) = \{q_0, q_2, q_5\}$

$\Delta(q_0, ababbb) = ???$

ε-Transiciones

- Objetivo: Sistematizar el proceso de determinar el conjunto de los siguientes estados de un AFN con ε-Transiciones.
- Definimos la ε-Cerradura de un estado q como,

  ε-c(q)={p| p es accesible desde q sin consumir nada en la entrada}

  podemos ampliar la definición,

  $$\varepsilon\text{-c}(\{q_{i1},q_{i2},...,q_{in}\}) = \cup\ \varepsilon\text{-c}(q_{ik}) \text{ para } k=1..n$$

- Además, para $q \in Q$, $\sigma \in \Sigma$ se define,

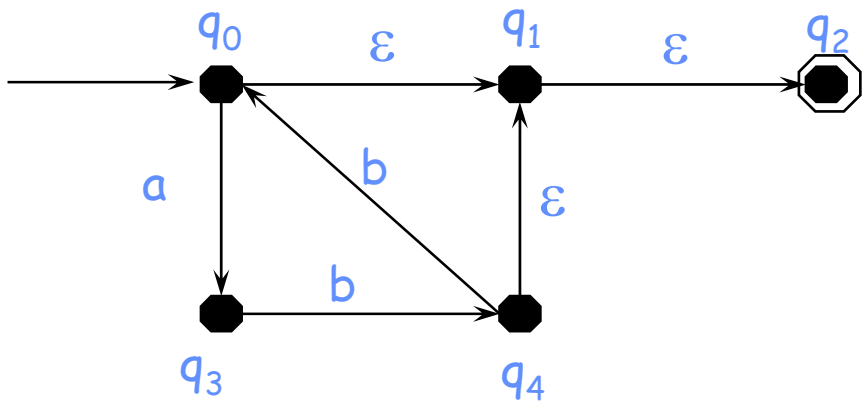
  d(q,σ)={p| hay una transición de q a p etiquetada con σ}

  podemos ampliar la definición,

  $$d(\{q_{i1},q_{i2},...,q_{in}\},\sigma) = \cup\ d(q_{ik},\sigma) \text{ para } k=1..n$$

ε-Transiciones: Ejemplo

- Ejemplo: Sea el AFN,



entonces,

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

$\varepsilon\text{-c}(q_4)=\{q_1, q_2, q_4\}$

$d(q_0, a)=\{q_3\}$

$d(q_0, b)= \phi$

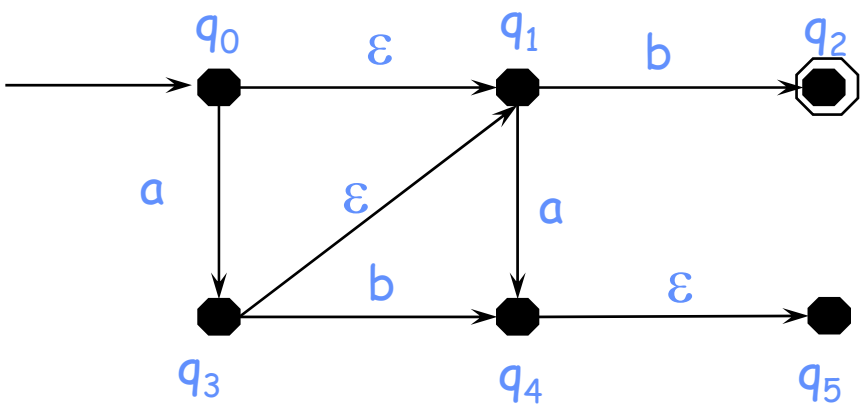
$d(\{q_3,q_4\}, b)=\{q_0, q_4\}$

ε-Transiciones

- En base a las definiciones previas, dado un AFN con ε-Transiciones, el conjunto de siguientes estado a q mediante el símbolo de entrada σ se define como:

$$\varepsilon\text{-c}(d(\varepsilon\text{-c}(q), \sigma))$$

- Ejemplo: Sea el AFN con ε-Transiciones,



conjunto de siguientes estados para q_0 para el símbolo de entrada a,

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

$$d(\varepsilon\text{-c}(q_0), a)=\{q_3, q_4\}$$

$$\varepsilon\text{-c}(d(\varepsilon\text{-c}(q_0), a))=\{q_1, q_3, q_4, q_5\}$$

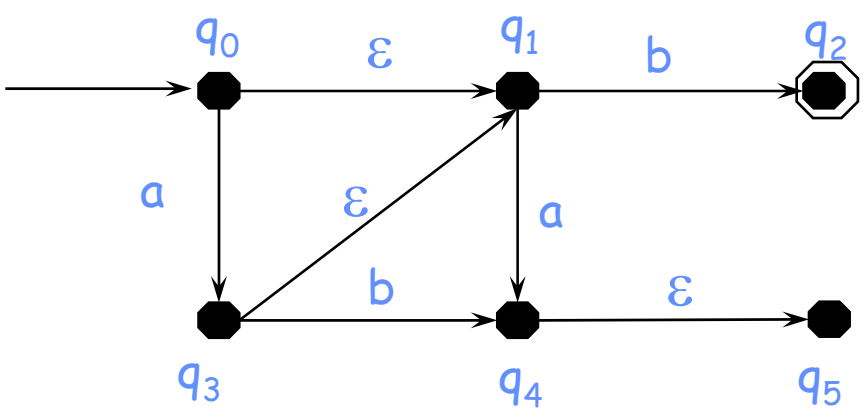
ε-Transiciones: Equivalencia

- A partir de un AFN con ε-Transiciones $M=(\Sigma,Q,s,F,\Delta)$ se puede construir un AFN sin ε-Transiciones $M'=(\Sigma,Q,s,F',\Delta')$ que acepte el mismo lenguaje, siendo

$$F' = F \cup \{q \mid \varepsilon\text{-c}(q) \cap F \neq \phi\}$$

$$\Delta'(q, \sigma) = \varepsilon\text{-c}\,(d(\varepsilon\text{-c}(q), \sigma))$$

- La colección de lenguajes aceptada por AFN con ε-Transiciones es la misma que la aceptada por AFN sin ε-Transiciones.
- Ejemplo: Transformar AFN con ε-Transiciones a un AFN sin ε-Transiciones.

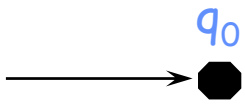


A.F. y Expresiones Regulares

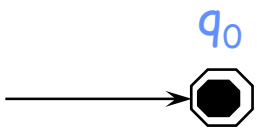
- Objetivos:
  - Dada una Expresión Regular construir un Autómata Finito que la reconozca.
  - Dado un Autómata Finito calcular la Expresión Regular que reconoce.
- Repaso: Las expresiones regulares sobre un Σ y el lenguaje que denotan se define recursivamente como:
  1) ϕ es una e.r y denota el lenguaje vacío
  2) ε es una e.r y denota el lenguaje $\{\varepsilon\}$
  3) Para cada $a \in \Sigma$, a es una e.r y denota el lenguaje $\{a\}$
  4) Si α, β son dos e.r que denotan los lenguajes A y B, entonces:
     1) $\alpha + \beta$ es una e.r que denota $A \cup B$
     2) $\alpha\beta$ es una e.r que denota $A \cdot B$
     3) α^* es una e.r que denota A^*

A.F. y Expresiones Regulares

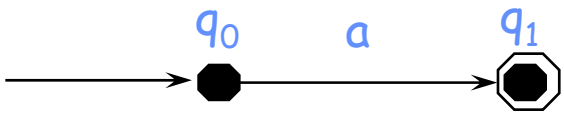
- Para un determinado Σ es posible construir AFN que acepten:
  - Lenguaje Vacío (ϕ):



  - Lenguaje {ε}:



  - Lenguajes unitarios de Σ:



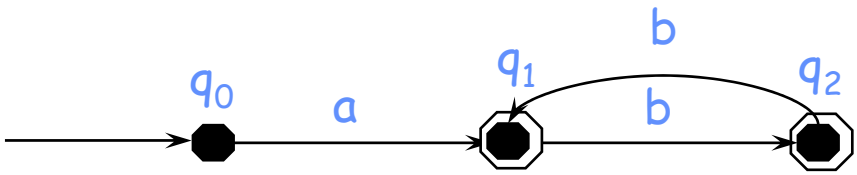
A.F. y Expresiones Regulares

- Dados dos AFN $M_1=(\Sigma_1, Q_1, s_1, F_1, \Delta_1)$ y $M_2= (\Sigma_2, Q_2, s_2, F_2, \Delta_2)$. Podemos **UNIR** M_1 y M_2 para crear un AFN que reconozca el lenguaje $L(M_1) \cup L(M_2)$.
- La construcción formal del nuevo autómata viene dada por $M=(\Sigma,Q,s,F,\Delta)$ tal que:
  - $\Sigma = \Sigma_1 \cup \Sigma_2$
  - $Q = Q_1 \cup Q_2 \cup \{s\}$
  - s = es un nuevo estado inicial añadido.
  - $F = F_1 \cup F_2$
  - Δ se construye para que incluya las transiciones de Δ_1 y de Δ_2, y además, se incluyen dos ε-transiciones desde el nuevo estado s a los estados iniciales de M_1 y M_2.

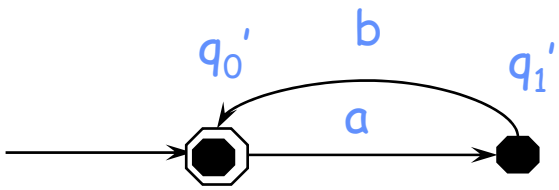
$$\Delta = \Delta_1 \cup \Delta_2 \cup \{(s, \varepsilon, s_1), (s, \varepsilon, s_2)\}$$

A.F. y Expresiones Regulares: Ejemplo.

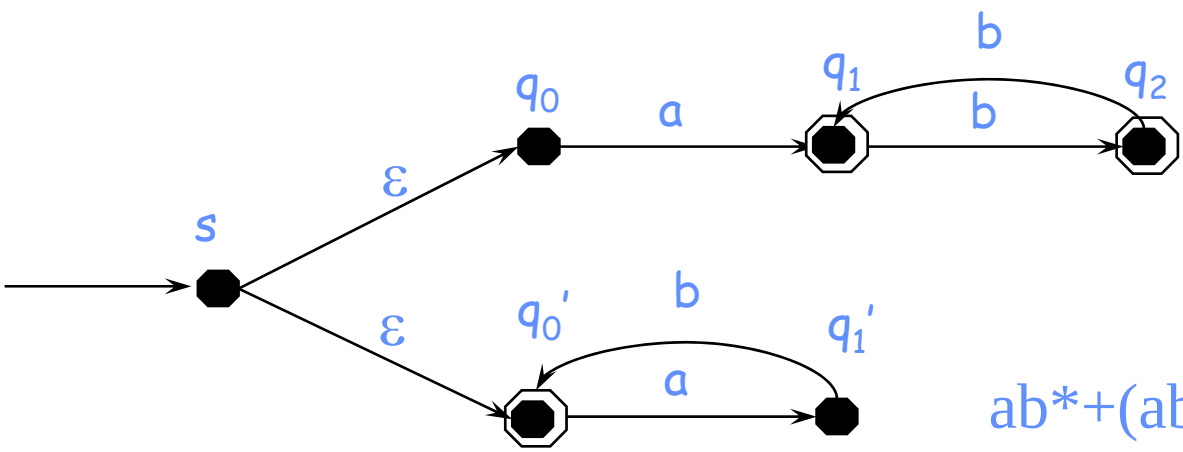
- Ejemplo: Tenemos los AFN que aceptan ab* y (ab)*, y queremos construir un AFN que acepte ab* + (ab)*.



ab*



(ab)*



ab*+(ab)*

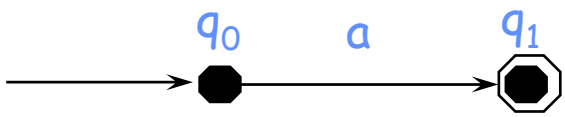
A.F. y Expresiones Regulares

- Dados dos AFN $M_1=(\Sigma_1, Q_1, s_1, F_1, \Delta_1)$ y $M_2= (\Sigma_2, Q_2, s_2, F_2, \Delta_2)$. Podemos **CONCATENAR** M_1 y M_2 para crear un AFN que reconozca el lenguaje $L(M_1)L(M_2)$.
- La construcción formal del nuevo autómata viene dada por $M=(\Sigma,Q,s,F,\Delta)$ tal que:
  - $\Sigma = \Sigma_1 \cup \Sigma_2$
  - $Q = Q_1 \cup Q_2$
  - $s = s1$
  - $F = F_2$
  - $\Delta = \Delta_1 \cup \Delta_2 \cup \{ F_1 \times \{\varepsilon\} \times \{s_2\} \}$

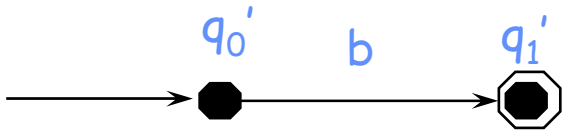
    $\forall q \in F_1, (q, \varepsilon, s_2)$

A.F. y Expresiones Regulares: Ejemplo.

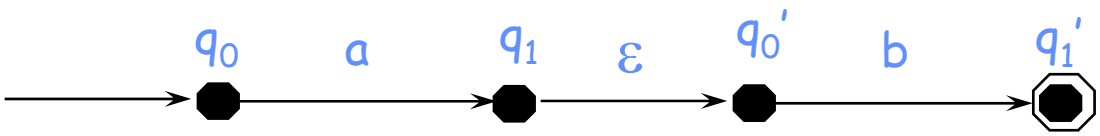
- Ejemplo: Tenemos los AFN que aceptan a y b, y queremos construir un AFN que acepte ab.



a



b



ab

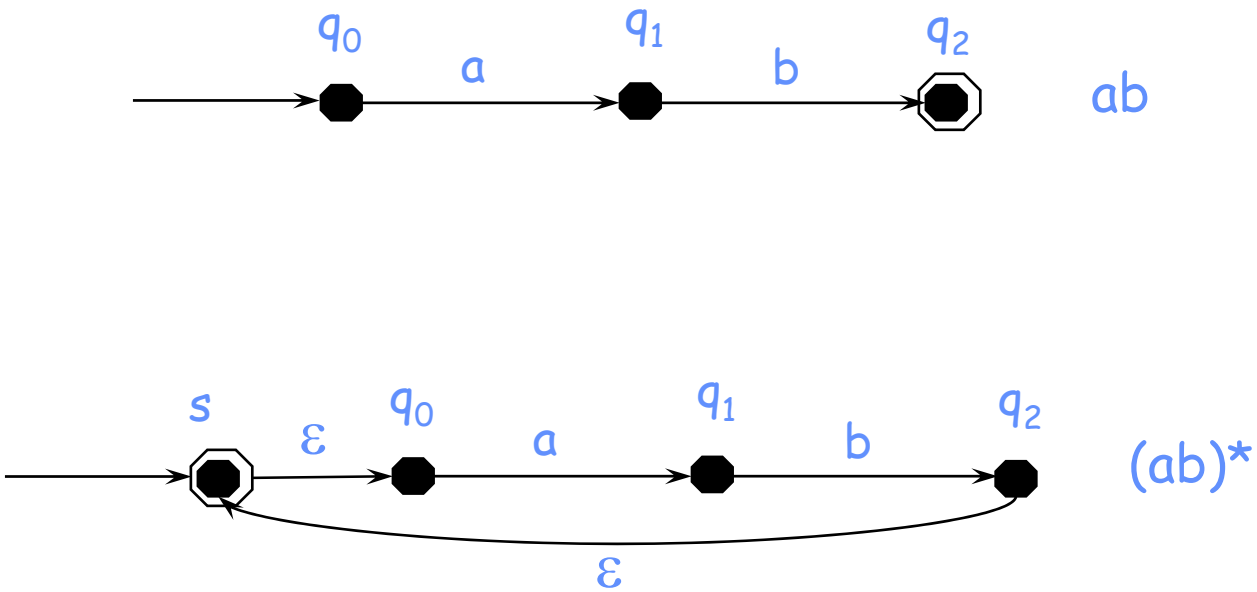
A.F. y Expresiones Regulares

- Dado un AFN $M_1=(\Sigma_1, Q_1, s_1, F_1, \Delta_1)$. Podemos crear un AFN que reconozca el lenguaje $L(M_1)^*$.
- La construcción formal del nuevo autómata viene dada por $M=(\Sigma,Q,s,F,\Delta)$ tal que:
  - $\Sigma = \Sigma_1$
  - $Q = Q_1 \cup \{s\}$
  - s = nuevo estado inicial del AFN M.
  - $F = \{s\}$
  - $\Delta = \Delta_1 \cup \{s, \varepsilon, s_1\} \cup \{ F_1 \times \{\varepsilon\} \times \{s\} \}$

    $\forall q \in F_1, (q, \varepsilon, s)$

A.F. y Expresiones Regulares: Ejemplo.

- Ejemplo: Tenemos un AFN que acepta ab, y queremos construir un AFN que acepte (ab)*.



A.F. y Expresiones Regulares

- De la discusión previa se deduce, que el conjunto de los lenguajes aceptados por un AF sobre un Σ, contiene los lenguajes de construcción básica. Además, este conjunto es cerrado respecto de la unión, de la concatenación y la estrella de Kleene.
- Es decir, aplicando las técnicas previas, somos capaces de construir a partir de una expresión regular el AF que la reconoce (objetivo 1).

A.F. y Expresiones Regulares

- El objetivo ahora es a partir de un autómata finito deducir la expresión regular que define el lenguaje que reconoce.
- Sea $M=(\Sigma,Q,s,F,\Delta)$ un A.F. Suponemos que $s=q_0$. Entonces, para todo estado q_i, sea

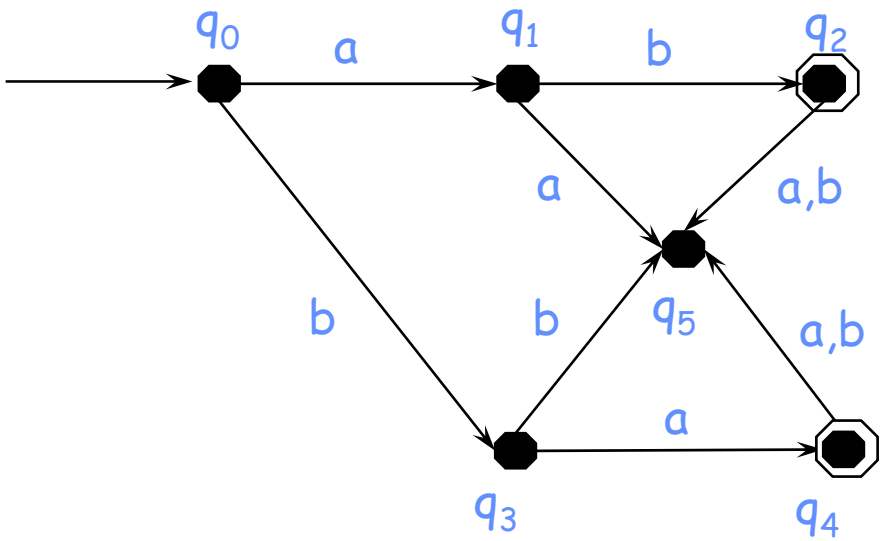
$$A_i=\{w\in\Sigma^* \mid \Delta(q_i, w) \cap F \neq\phi\}$$

el conjunto de todas las cadenas que hacen que M pase de q_i a un estado de aceptación.

  - $A_0 = L(M)$, por suponer que $s = q_0$
  - Es posible que $A_i = \phi$, por ejemplo, un estado sumidero.
  - $q_i \in F$, entonces $\varepsilon\in A_i$

A.F. y Expresiones Regulares: Ejemplo

- Ejemplo: Dado el siguiente A.F,



Entonces,

$A_5 = \phi$ $A_2 = \varepsilon$

$A_4 = \varepsilon$ $A_1 = b$

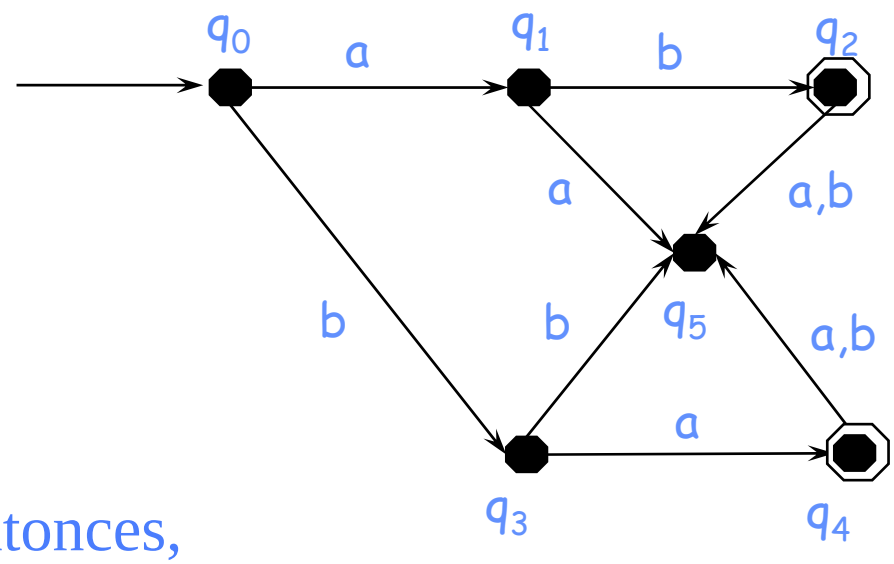
$A_3 = a$ $A_0 = ab + ba$

A.F. y Expresiones Regulares

- Si $q_j \in \Delta(q_i, \sigma)$. Entonces A_i contiene σA_j. De hecho, se tiene que

$$A_i = \cup \{\sigma A_j \mid q_j \in \Delta(q_i, \sigma)\}$$

- Ejemplo: Dado el siguiente A.F,



entonces,

$A_0 = aA_1 + bA_3$ $\quad$ $A_3 = aA_4 + bA_5$

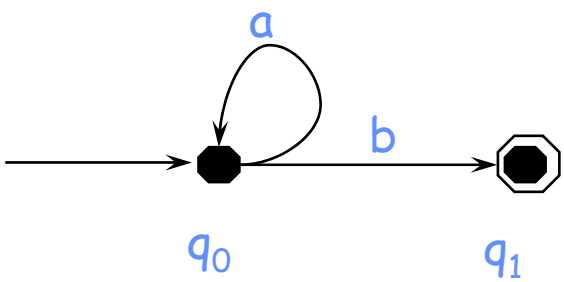
$A_1 = aA_5 + bA_2$ $\quad$ $A_4 = \varepsilon + aA_5 + bA_5$

$A_2 = \varepsilon + aA_5 + bA_5$ $\quad$ $A_5 = \phi$

$$A_0 = L(M) = ab + ba$$

A.F. y Expresiones Regulares

- Ejemplo: Dado el siguiente A.F,



entonces,

$A_0 = aA_0 + bA_1$

$A_1 = \varepsilon$

Resolviendo por sustitución queda,

$$A_0 = aA_0 + b$$

- Lema de Arden: Una ecuación de la forma X=AX+B donde $\varepsilon \notin A$, tiene como solución única X=A*B.

Por tanto, la solución será $A_0 = a^*b$

A.F. y Expresiones Regulares

- Sea M un AF entonces existe una expresión regular r tal que L(M)=L(r).
- Teorema de Kleene: Un lenguaje es regular si y sólo si es aceptado por un A.F.
- Es decir, aplicando las técnicas previas, somos capaces de construir a partir de un AF la expresión regular que especifica el lenguaje que reconoce (objetivo 2).

Propiedades de los Lenguajes Regulares

- Dado un lenguaje L, ¿L es regular?:
  - L es finito $\Rightarrow$ L es regular.
  - L es infinito $\Rightarrow$??

- Objetivo: Encontrar propiedades que sean compartidas por todos los lenguajes regulares infinitos y que no estén presentes en los lenguajes no regulares.

Propiedades de los Lenguajes Regulares

- Suponemos que tenemos un lenguaje L que es regular, aceptado por un AFD $M=(\Sigma,Q,s,F,\delta)$ donde Q tiene n estados.

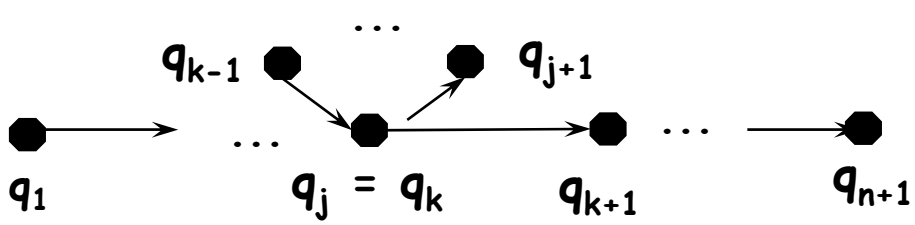
  Si L es infinito entonces existirá $w=a_1a_2...a_{n+1}$. Si tuviéramos sucesivamente,

  $$\delta(s,a_1) = q_1$$

  $$\delta(q_1,a_2) = q_2$$

  ...

  pasaríamos por n+1 estados q_i. Q sólo tiene n estados, con lo cual no todos ellos son distintos. Entonces, para $1 \leq j < k \leq n+1$, $q_j = q_k$



  Dado que $j < k$, el "ciclo" tiene al menos longitud 1. Entonces, $w'=a_1a_2...a_ja_{k+1}...a_{n+1} \in L$, y $w''= a_1a_2...a_j\,(a_{j+1}...a_k)^m\,a_{k+1}...a_{n+1} \in L$.

  Puedo "bombear" cero o más veces el "ciclo" obteniendo palabras que pertenecen a L.

Propiedades de los Lenguajes Regulares

- Lema de Bombeo: Sea L un lenguaje regular infinito. Entonces, hay una constante n de forma que, si w es una cadena de L cuya longitud es mayor o igual que n, se tiene que w=uvx, siendo, $uv^ix \in L$ para todo $i \geq 0$, con $|v| \geq 1$ y $|uv| \leq n$.
  - Nos permite demostrar cuando un lenguaje no es regular
  - Para demostrar que no es regular, para cualquier valor de n lo bastante grande, se tendrá al menos una cadena que falle al ser "bombeada".
  - Todos los lenguajes regulares cumplen el L.B. ¡y algunos no regulares, también! p.ej. $L=\{a^ib^jc^k \mid i = 0 \text{ ó } j = k\}$ no es regular, pero cumple el L.B.

Propiedades de los Lenguajes Regulares

- Ejemplos: ¿Son regulares los siguientes lenguajes?
  - $L = \{a^{i^2} \mid i \geq 1\}$
  - $L = \{a^m b^m \mid m \geq 0\}$

Utilidades: Equivalencia de AFD

- ¿Son equivalentes dos AFD M y M'?

```
algoritmo equivalentesAFD (M, M': AFD)
principio
    Construimos una tabla de comparación;
    {#columnas = 1 (pares estados) + # símbolos Σ}
    Insertar fila en tabla (s, s');
    Mientas que existan filas en tabla hacer
    {fila (q, q') }
        Para cada símbolo σ hacer
            Insertar columna símbolo (p,p')
             {p∈Q, p'∈Q' y δ(q, σ)=p, δ(q', σ)=p'}
             Si (p∈F and p'∉F') or
                (p∉F and p'∈F')
              entonces return(No_Equivalentes);
              sino Insertar fila en tabla (p,p') si no está ya;
             fsi;
        fpara;
    fMientrasQue:
    return (Si_Equivalentes);
fin;
```

Utilidades: Equivalencia de AFD

```
algoritmo equivalentesAFD (M, M': AFD)
T : TablaComparacion [][1 + # σ ∈ Σ] := TablaVacía;
{n filas, una columna 1 y una columna por cada σ ∈ Σ}
principio
    i := 1;
    Si (s∈F and s'∉F') or (s∉F and s'∈F')
    entonces return (No_Equivalentes);
    T[i][1] = (s, s');
    Mientras que T[i][1] no es nulo  hacer
        (q, q') = T[i][1];
        Para cada σ ∈ Σ hacer
            p := δ(q, σ);     p' := δ'(q', σ);
            T[i][σ ] := (p, p');
            Si (p∈F and p'∉F') or
               (p∉F and p'∈F')
            entonces return (No_Equivalentes);
            {Añadir (p,p') a T[][1]
              si no estaba ya}
            fsi;
        fpara;
        i := i + 1;
    fMientrasQue:
    return (Si_Equivalentes);
fin;
```

Utilidades: AFD Mínimo

- Dado un AFD, dos estados p y q son distinguibles si para alguna $w \in \Sigma^*$, entonces $\delta(p, w) \in F$ y $\delta(q, w) \notin F$ o viceversa. Dos estados no distinguibles se dicen equivalentes.
- Un AFD mínimo no tiene estados redundantes, es decir, todos sus estados son distinguibles.
- ¿Cómo obtener un AFD Mínimo?

  Eliminar la redundancia sustituyendo los conjuntos de estados no distinguibles por un único estado.

Utilidades: AFD Mínimo

- Construir el AFD Mínimo:

algoritmo AFDMínimo (M:AFD)
principio
    Eliminar estados no alcanzables de M;
    Construcción tabla [Q,Q]
    {Con la mitad es suficiente, las celdas simétricas respecto la diagonal representan los mismos pares de estados}
    Marcamos Distinguibles los pares [$q \in F$, $q \notin F$] ;
    Para (p,q) no conocidos como Distinguibles hacer
        Para cada símbolo $\sigma \in \Sigma$ hacer
            Calcular (p_σ, q_σ) tq $\delta(p,\sigma) = p_\sigma$ y $\delta(q,\sigma) = q_\sigma$
            Si (p_σ, q_σ) son Distinguibles
                entonces (p,q) Distinguibles y Break;
                sino (p,q) insertar lista asociada (p_σ, q_σ)
            fsi;
        fpara;
    fpara;
    Para todas las listas asociadas hacer
        Si (p_σ, q_σ) son Distinguibles
            entonces Todos sus (p,q) en su lista Distinguibles;
        fsi;
    fpara;
    Extraer conjunto de estados Distinguibles;
    Establecer transiciones para el nuevo AFD Mínimo;
fin.