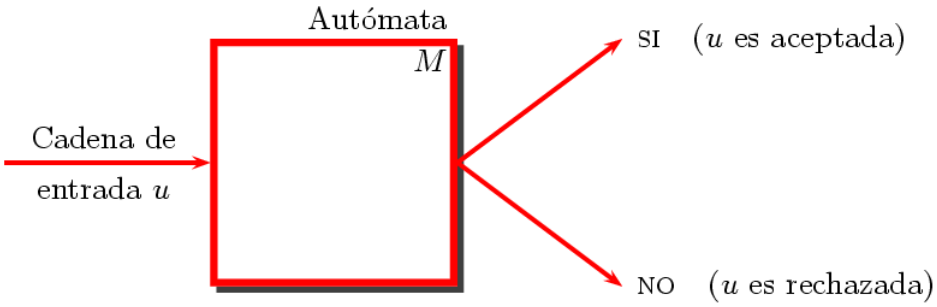
Capítulo 2

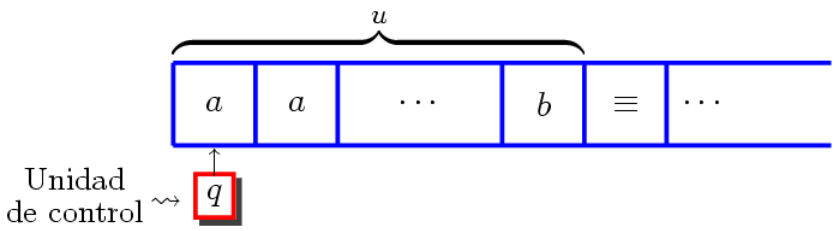
Autómatas finitos

2.1. Autómatas finitos deterministas (AFD)

Los **autómatas finitos** son máquinas abstractas que procesan cadenas, las cuales son aceptadas o rechazadas:



El autómata actúa leyendo los símbolos escritos sobre una cinta semi-infinita, dividida en celdas o casillas, sobre la cual se escribe una cadena de entrada u, un símbolo por casilla. El autómata posee una **unidad de control** que inicialmente "escanea" o lee la casilla del extremo izquierdo de la cinta.



La unidad de control (también llamada **control finito** o **cabeza lectora**) del autómata posee un cierto número (finito) de configuraciones internas, llamadas **estados del autómata**. Entre los estados de un autómata se destacan el **estado inicial** y los **estados finales** o **estados de aceptación**.

Formalmente, un autómata finito M está definido por cinco parámetros, $M = (\Sigma, Q, q_0, F, \delta)$, a saber:

1. Un alfabeto Σ, llamado alfabeto de cinta. Todas las cadenas que procesa M pertenecen a Σ^*.
2. $Q = \{q_0, q_1, \ldots, q_n\}$, conjunto de estados del autómata.
3. $q_0 \in Q$, estado inicial.
4. $F \subseteq Q$, conjunto de estados finales o de aceptación. $F \neq \varnothing$.
5. La función de transición del autómata

$$\begin{array}{rcl} \delta : Q \times \Sigma & \longrightarrow & Q \\ (q, a) & \longmapsto & \delta(q, a) \end{array}$$

Toda cadena de entrada es procesada completamente, hasta que la unidad de control encuentra la primera casilla vacía.

Ejemplo

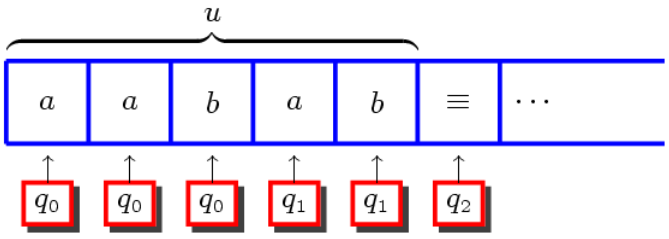
$\Sigma = \{a, b\}$.
$Q = \{q_0, q_1, q_2\}$.
q_0 : estado inicial.
$F = \{q_0, q_2\}$, estados de aceptación.
Función de transición δ:

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

$$\begin{array}{ll} \delta(q_0, a) = q_0 & \delta(q_0, b) = q_1 \\ \delta(q_1, a) = q_1 & \delta(q_1, b) = q_2 \\ \delta(q_2, a) = q_1 & \delta(q_2, b) = q_1. \end{array}$$

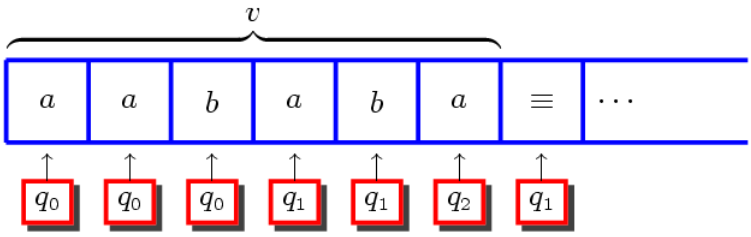
Ilustración del procesamiento de dos cadenas de entrada:

1. $u = aabab$.



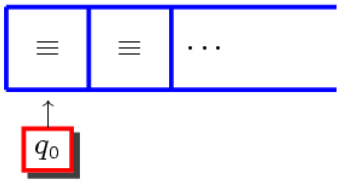
Como q_2 es un estado de aceptación, la cadena de entrada u es aceptada.

2. $v = aababa$.



Puesto que q_1 no es un estado de aceptación, la cadena de entrada v es rechazada.

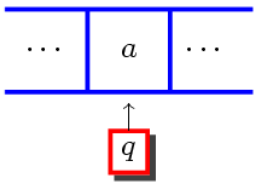
Caso especial: la cadena λ es la cadena de entrada.



Como q_0 es un estado de aceptación, la cadena λ es aceptada.

En general se tiene lo siguiente: la cadena vacía λ es aceptada por un autómata M si y solamente si el estado inicial q_0 de M también es un estado de aceptación.

Los autómatas finitos descritos anteriormente se denominan **autómatas finitos deterministas** (AFD) ya que para cada estado q y para cada símbolo $a \in \Sigma$, la función de transición $\delta(q, a)$ siempre está definida. Es decir, la función de transición δ *determina completa y unívocamente* la acción que el autómata realiza cuando la unidad de control se encuentra en un estado q leyendo un símbolo a sobre la cinta:



Dado un autómata M, el **lenguaje aceptado o reconocido** por M se denota $L(M)$ y se define por

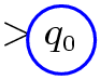
$$L(M) := \{u \in \Sigma^* : M \text{ termina el procesamiento de la cadena de entrada } u \text{ en un estado } q \in F\}.$$

2.2. Diagrama de estados de un autómata finito

Un autómata finito se puede representar por medio de un grafo dirigido y etiquetado. Recuérdese que un **grafo** es un conjunto de vértices o nodos unidos por arcos o conectores; si los arcos tienen tanto dirección como etiquetas, el grafo se denomina **grafo dirigido y etiquetado** o **digrafo etiquetado**.

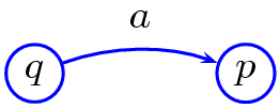
El grafo de un autómata se obtiene siguiendo las siguientes convenciones:

- Los vértices o nodos son los estados del autómata.
- El estado q se representa por: 

- El estado inicial q_0 se representa por: 

- Un estado final q se representa por: 

- La transición $\delta = (q, a) = p$ se representa en la forma



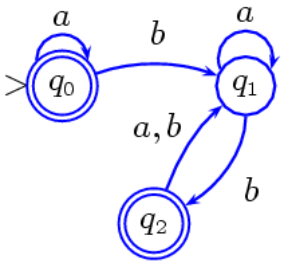
Dicho grafo se denomina el **diagrama de estados del autómata**.

Ejemplo Diagrama de estados del autómata presentado en la sección anterior.

$\Sigma = \{a, b\}$.
$Q = \{q_0, q_1, q_2\}$.
q_0 : estado inicial.
$F = \{q_0, q_2\}$, estados de aceptación.
Función de transición δ:

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

$\delta(q_0, a) = q_0$ $\quad$ $\delta(q_0, b) = q_1$
$\delta(q_1, a) = q_1$ $\quad$ $\delta(q_1, b) = q_2$
$\delta(q_2, a) = q_1$ $\quad$ $\delta(q_2, b) = q_1$



2.3. Diseño de autómatas

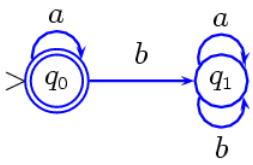
Para autómatas deterministas se adopta la siguiente convención adicional con respecto a los diagramas de estados: se supone los arcos no dibujadas explícitamente conducen a un estado "limbo" de no-aceptación. Es decir, en el diagrama de estados se indican únicamente los arcos que conduzcan a trayectorias de aceptación. Esto permite simplificar considerablemente los diagramas.

En este capítulo abordaremos dos tipos de problemas:

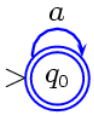
1. Dado un lenguaje regular L diseñar un autómata finito M que acepte o reconozca a L, es decir, tal que $L(M) = L$.
2. Dado un autómata M determinar el lenguaje aceptado por M.

Más adelante se demostrará, en toda su generalidad, que estos problemas *siempre* tienen solución. Consideremos inicialmente problemas del primer tipo.

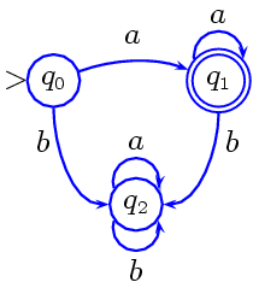
Ejemplo $\Sigma = \{a, b\}$. $L = a^* = \{\lambda, a, a^2, a^3, \ldots\}$. AFD M tal que $L(M) = L$:



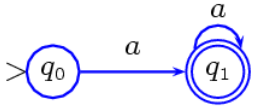
Versión simplificada:



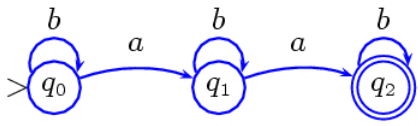
Ejemplo $\Sigma = \{a, b\}$. $L = a^+ = \{a, a^2, a^3, \ldots\}$. AFD M tal que $L(M) = L$:



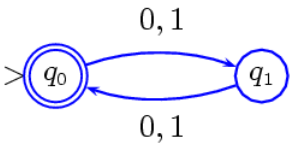
Versión simplificada:



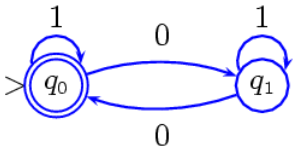
Ejemplo $\Sigma = \{a, b\}$. $L =$ lenguaje de las cadenas que contienen exactamente dos $a$es $= b^*ab^*ab^*$. AFD M tal que $L(M) = L$:



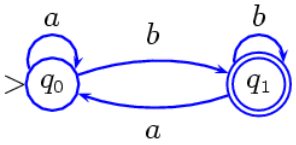
Ejemplo $\Sigma = \{0, 1\}$. $L =$ lenguaje de las cadenas sobre Σ que tienen un número par de símbolos (cadenas de longitud par). AFD M tal que $L(M) = L$:



Ejemplo $\Sigma = \{0, 1\}$. $L =$ lenguaje de las cadenas sobre Σ que contienen un número par de ceros. AFD M tal que $L(M) = L$:



Ejemplo $\Sigma = \{a, b\}$. $L =$ lenguaje de las cadenas sobre Σ que terminan en b. AFD M tal que $L(M) = L$:



Ejercicios Diseñar autómatas finitos deterministas que acepten los siguientes lenguajes:

1. $\Sigma = \{0, 1\}$. $L =$ lenguaje de las cadenas sobre Σ de longitud impar.
2. $\Sigma = \{0, 1\}$. $L =$ lenguaje de las cadenas sobre Σ que contienen un número impar de unos.
3. $\Sigma = \{a, b\}$. $L = ab^+$.
4. $\Sigma = \{a, b\}$. $L = ab^* \cup ab^*a$.

5. $\Sigma = \{0, 1\}$. $L = (0 \cup 10)^*$.

6. $\Sigma = \{0, 1\}$. $L = (01 \cup 10)^*$.

7. $\Sigma = \{0, 1\}$. Lenguaje de todas las cadenas que no contienen dos unos consecutivos.

8. $\Sigma = \{a, b\}$. $L =$ lenguaje de las cadenas sobre Σ que contienen un número par de $a$es y un número par de $b$es. Ayuda: utilizar 4 estados.

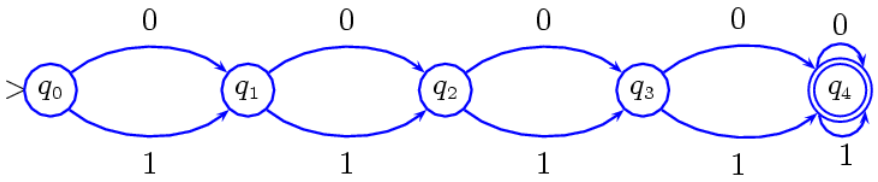
9. $\Sigma = \{a, b\}$. Para cada combinación de las condiciones "par" e "impar" y de las conectivas "o" e "y", diseñar un AFD que acepte el lenguaje L donde:

   $L =$ lenguaje de las cadenas con un número par/impar de $a$es
   y/o un número par/impar de $b$es.

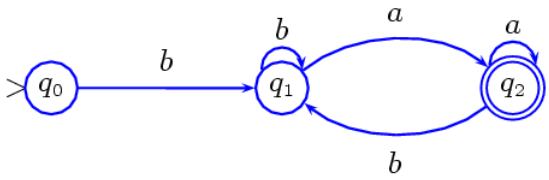
   Ayuda: utilizar el autómata de 4 estados diseñado en el ejercicio anterior, modificando adecuadamente el conjunto de estados finales.

Ejercicios Determinar los lenguajes aceptados por los siguientes AFD. Describir los lenguajes ya sea por medio de una propiedad característica o de una expresión regular.

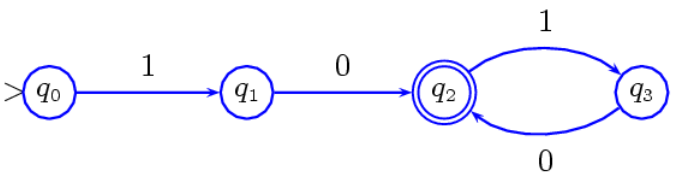
1.



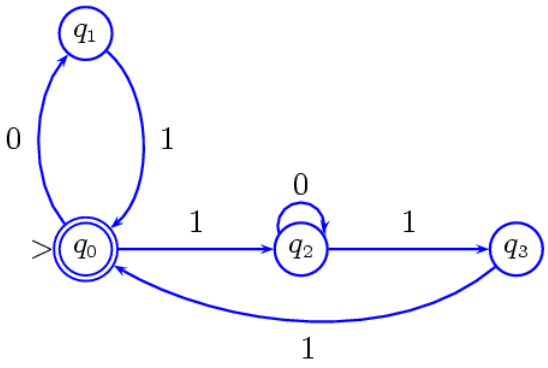
2.



3.



4.



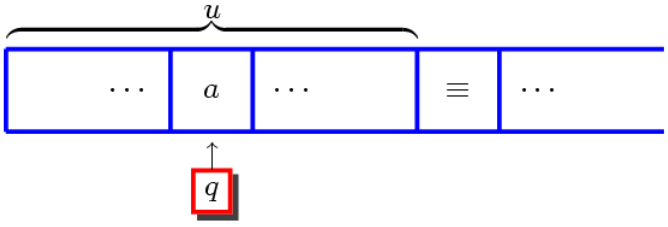
2.4. Autómatas finitos no deterministas (AFN)

Los **autómatas finitos no-deterministas** (AFN) se asemejan a los AFD, excepto por el hecho de que para cada estado $q \in Q$ y cada $a \in \Sigma$, la transición $\delta(q, a)$ puede consistir en más de un estado o puede no estar definida. Concretamente, un AFN está definido por $M = (\Sigma, Q, q_0, F, \Delta)$ donde:

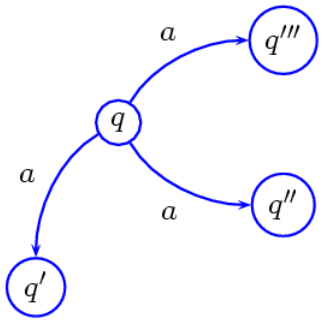
1. Σ es el alfabeto de cinta.
2. Q es un conjunto (finito) de estados.
3. $q_0 \in Q$ es el estado inicial.
4. $\varnothing \neq F \subseteq Q$ es el conjunto de estados finales o estados de aceptación.
5. $$\begin{array}{rcl} \Delta : Q \times \Sigma & \longrightarrow & \wp(Q) \\ (q, a) & \longmapsto & \Delta(q, a) = \{q_{i_1}, q_{i_2}, \ldots, q_{i_k}\} \end{array}$$
   donde $\wp(Q)$ es el conjunto de subconjunto de Q.

Puede suceder que $\Delta(q, a) = \varnothing$, lo cual significa que, si durante el procesamiento de una cadena de entrada u, M ingresa al estado q leyendo sobre la cinta el símbolo a, el cómputo se aborta.

Cómputo abortado:



La noción de diagrama de estados para un AFN se define de manera análoga al caso AFD, pero puede suceder que desde un mismo nodo (estado) salgan dos o más arcos con la misma etiqueta:



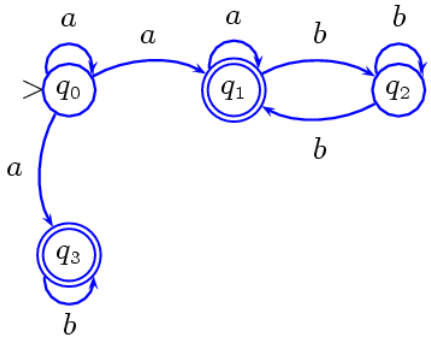
Un AFN M puede procesar una cadena de entrada $u \in \Sigma^*$ de varias maneras. Sobre el diagrama de estados del autómata, esto significa que pueden existir varias trayectorias etiquetadas con los símbolos de u.

La siguiente es la noción de aceptación para autómatas no deterministas:

$$\begin{array}{rcl} L(M) & = & \text{lenguaje aceptado o reconocido por } M \\ & = & \{u \in \Sigma^* : \text{existe por lo menos un cómputo completo} \\ & & \qquad \text{de } u \text{ que termina en un estado } q \in F\} \end{array}$$

Es decir, para que una cadena u sea aceptada, debe existir por lo menos un cómputo en el que u sea procesada completamente y que finalice estando M en un estado de aceptación.

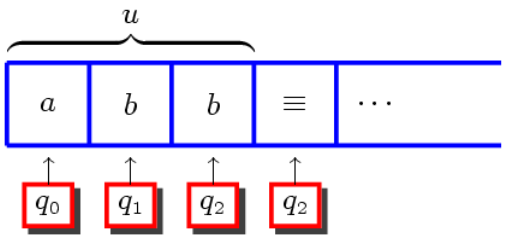
Ejemplo Sea M el siguiente AFN:



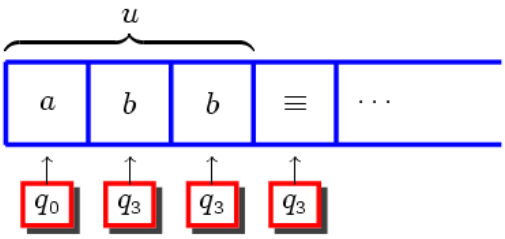
Δ	a	b
q_0	$\{q_0, q_1, q_3\}$	$\varnothing$
q_1	$\{q_1\}$	$\{q_2\}$
q_2	$\varnothing$	$\{q_1, q_2\}$
q_3	$\varnothing$	$\{q_3\}$

Para la cadena de entrada $u = abb$, existen cómputos que conducen al rechazo, cómputos abortados y cómputos que terminan en estados de aceptación. Según la definición de lenguaje aceptado, $u \in L(M)$.

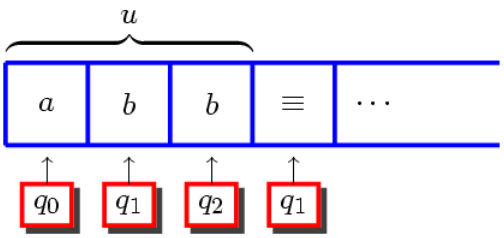
Cómputo de rechazo:



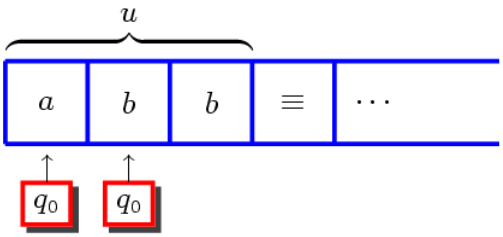
Cómputo de aceptación:



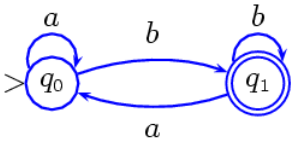
Otro cómputo de aceptación:



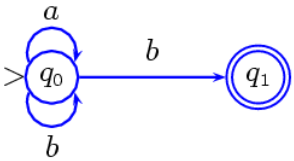
Cómputo abortado:



Ejemplo En el último ejemplo de la sección 2.3 se diseñó el AFD que acepta el lenguaje de las cadenas sobre $\Sigma = \{a, b\}$ que terminan en b:

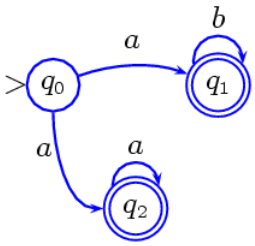


Un AFN que acepta el mismo lenguaje y que es, tal vez, más fácil de concebir, es el siguiente:

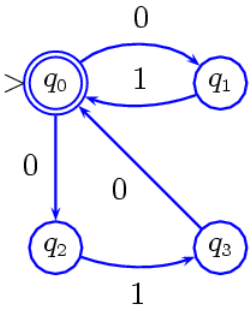


Este autómata se asemeja a la expresión regular $(a \cup b)^* b$.

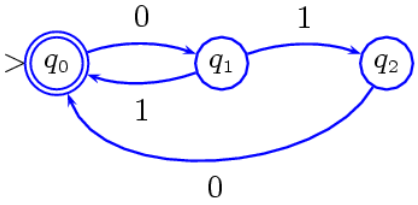
Ejemplo Considérese el lenguaje $L = ab^* \cup a^+$ sobre el alfabeto $\Sigma = \{a, b\}$. El siguiente AFN M satisface $L(M) = L$.



Ejemplo $\Sigma = \{0, 1\}$, $L = (01 \cup 010)^*$. El siguiente AFN acepta a L.



Otro AFN que acepta el mismo lenguaje y que tiene sólo tres estados es el siguiente:



Ejercicios Diseñar autómatas AFD o AFN que acepten los siguientes lenguajes:

1. $\Sigma = \{a, b\}$. $L = ab^+a^*$.
2. $\Sigma = \{a, b\}$. $L = a(a \cup ab)^*$.
3. $\Sigma = \{a, b, c\}$. $L = a^*b^*c^*$.

4. $\Sigma = \{0, 1, 2\}$. L = lenguaje de las cadenas sobre Σ que comienzan con 0 y terminan con 2.

5. $\Sigma = \{0, 1\}$. Lenguaje de las cadenas de longitud par ≥ 2 formadas por ceros y unos alternados.

6. $\Sigma = \{0, 1\}$. Lenguaje de las cadenas que tienen a lo sumo dos ceros consecutivos.

7. $\Sigma = \{0, 1\}$. Lenguaje de las cadenas de longitud impar que tienen unos en todas las posiciones impares y únicamente en las posiciones impares.

8. $\Sigma = \{a, b, c\}$. L = lenguaje de las cadenas sobre Σ que contienen la cadena bc.

9. $\Sigma = \{a, b, c\}$. L = lenguaje de las cadenas sobre Σ que *no* contienen la cadena bc. En el último ejemplo de la sección 1.14 se presentaron dos expresiones regulares para L. Nota: ¡se puede construir un AFD con sólo dos estados para aceptar este lenguaje!

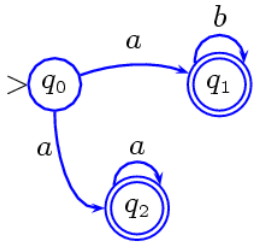
2.5. Equivalencia computacional entre los AFD y los AFN

En esta sección se mostrará que los modelos AFD y AFN son computacionalmente equivalentes. En primer lugar, es fácil ver que un AFD $M = (\Sigma, Q, q_0, F, \delta)$ puede ser considerado como un AFN $M' = (\Sigma, Q, q_0, F, \Delta)$ definiendo $\Delta(q, a) = \{\delta(q, a)\}$ para cada $q \in Q$ y cada $a \in \Sigma$. Para la afirmación recíproca tenemos el siguiente teorema.

2.5.1 Teorema. *Dado un AFN $M = (\Sigma, Q, q_0, F, \Delta)$ se puede construir un AFD M'* ***equivalente a*** *M, es decir, tal que $L(M) = L(M')$.*

Este teorema, cuya demostración se dará en detalle más adelante, establece que el no-determinismo se puede eliminar. Dicho de otra manera, los autómatas deterministas y los no deterministas aceptan los mismos lenguajes. La idea de la demostración consiste en considerar cada conjunto de estados $\{p_1, \ldots, p_j\}$, que aparezca en la tabla de la función Δ del autómata no-determinista, como un *único* estado del nuevo autómata determinista. La tabla de Δ se completa hasta que no aparezcan nuevas combinaciones de estados. Los estados de aceptación del nuevo autómata son los conjuntos de estados en los que aparece *por lo menos* un estado de aceptación del autómata original. El siguiente ejemplo ilustra el procedimiento.

Ejemplo Consideremos el AFN M, presentado en la sección 2.4, que acepta el lenguaje $L(M) = ab^* \cup a^+$ sobre $\Sigma = \{a, b\}$:

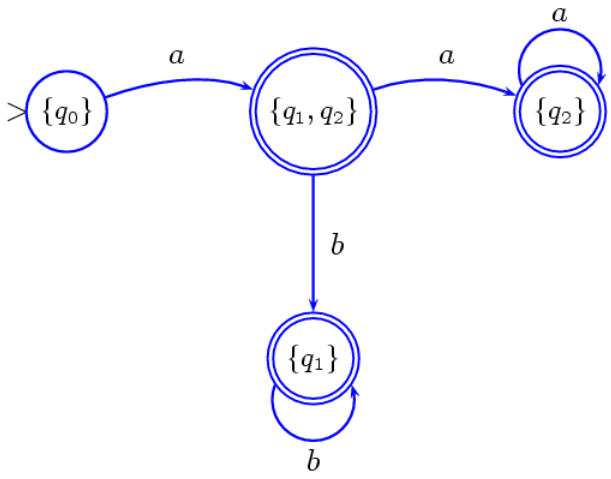


Δ	a	b
q_0	$\{q_1, q_2\}$	$\varnothing$
q_1	$\varnothing$	$\{q_1\}$
q_2	$\{q_2\}$	$\varnothing$

El nuevo AFD M' construido a partir de M tiene un estado más, $\{q_1, q_2\}$, y su función de transición δ tiene el siguiente aspecto:

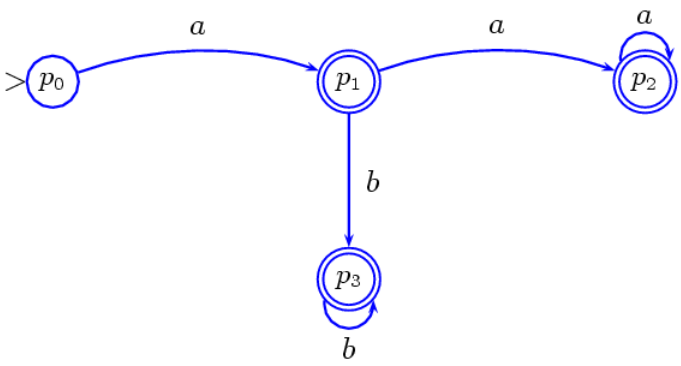
δ	a	b
q_0	$\{q_1, q_2\}$	$\varnothing$
q_1	$\varnothing$	$\{q_1\}$
q_2	$\{q_2\}$	$\varnothing$
$\{q_1, q_2\}$	$\{q_2\}$	$\{q_1\}$

El diagrama de estados de este autómata es:

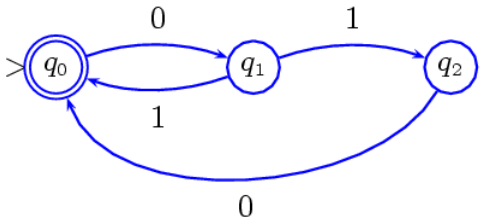


Los estados de aceptación son aquéllos en los que aparezcan q_1 ó q_2, que son los estados de aceptación del autómata original.

Para mayor simplicidad, podemos cambiar los nombres de los estados del nuevo autómata:



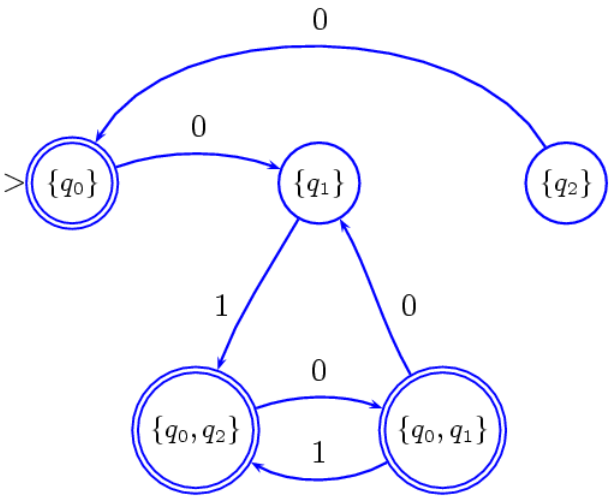
Ejemplo El siguiente AFN M, presentado en la la sección 2.4, acepta el lenguaje $L = (01 \cup 010)^*$ sobre $\Sigma = \{0, 1\}$.



La tabla de la función de transición de M se extiende para completar la función δ del nuevo AFN:

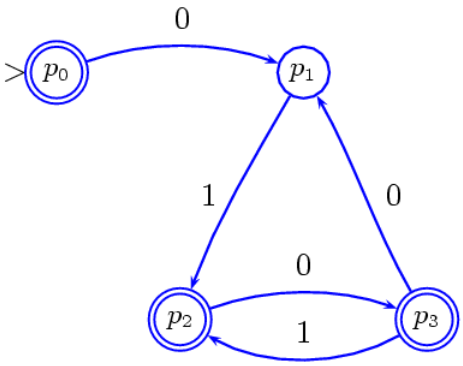
δ	0	1
q_0	$\{q_1\}$	$\varnothing$
q_1	$\varnothing$	$\{q_0, q_2\}$
q_2	$\{q_0\}$	$\varnothing$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\varnothing$
$\{q_0, q_1\}$	$\{q_1\}$	$\{q_0, q_2\}$

El diagrama de estados del nuevo autómata es:



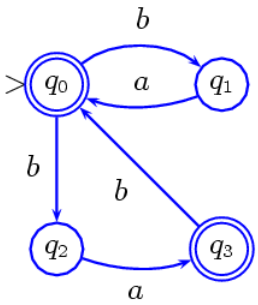
Los estados de aceptación son aquéllos en los que aparezca q_0 ya que q_0 es el único estado de aceptación del autómata original.

Puesto que el nuevo estado $\{q_2\}$ no interviene en el lenguaje aceptado, el nuevo autómata se puede simplificar en la siguiente forma:

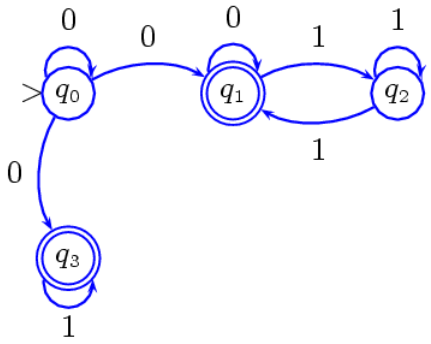


Ejercicios Diseñar AFD equivalentes a los siguientes AFN:

1.



2.



Para la demostración del teorema 2.5.1, conviene extender la definición de la función de transición, tanto de los autómatas deterministas como de los no-deterministas.

2.5.2 Definición. Sea $M = (\Sigma, Q, q_0, F, \delta)$ un AFD. La función de transición $\delta : Q \times \Sigma \longrightarrow Q$ se extiende a una función $\widehat{\delta} : Q \times \Sigma^* \longrightarrow Q$ por medio de la siguiente definición recursiva:

$$\begin{cases} \widehat{\delta}(q, \lambda) = q, \quad q \in Q, \\ \widehat{\delta}(q, a) = \delta(q, a), \quad q \in Q,\ a \in \Sigma, \\ \widehat{\delta}(q, wa) = \delta(\widehat{\delta}(q, w), a), \quad q \in Q,\ a \in \Sigma,\ w \in \Sigma^*. \end{cases}$$

Según esta definición, para una cadena de entrada $w \in \Sigma^*$, $\widehat{\delta}(q_0, w)$ es el estado en el que el autómata termina el procesamiento de w. Por lo tanto, podemos describir el lenguaje aceptado por M de la siguiente forma:

$$L(M) = \{w \in \Sigma^* : \widehat{\delta}(q_0, w) \in F\}.$$

Notación. Sin peligro de ambigüedad, la función extendida $\widehat{\delta}(q, w)$ se notará simplemente $\delta(q, w)$.

2.5.3 Definición. Sea $M = (\Sigma, Q, q_0, F, \Delta)$ un AFN. La función de transición $\Delta : Q \times \Sigma \longrightarrow \wp(Q)$ se extiende inicialmente a conjuntos de estados. Para $a \in \Sigma$ y $S \subseteq F$ se define

$$\Delta(S, a) := \bigcup_{q \in S} \Delta(q, a).$$

Luego se extiende Δ a una función $\widehat{\Delta} : Q \times \Sigma^* \longrightarrow \wp(Q)$, de manera similar a como se hace para los AFD. Recursivamente,

$$\begin{cases} \widehat{\Delta}(q, \lambda) = \{q\}, \quad q \in Q, \\ \widehat{\Delta}(q, a) = \Delta(q, a), \quad q \in Q,\ a \in \Sigma, \\ \widehat{\Delta}(q, wa) = \Delta(\widehat{\Delta}(q, w), a) = \displaystyle\bigcup_{p \in \widehat{\Delta}(q,w)} \Delta(p, a), \quad q \in Q,\ a \in \Sigma,\ w \in \Sigma^*. \end{cases}$$

Según esta definición, para una cadena de entrada $w \in \Sigma^*$, $\widehat{\Delta}(q_0, w)$ es el conjunto de los posibles estados en los que terminan los cómputos *completos* de w. Si el cómputo se aborta durante el procesamiento de w, se tendría $\widehat{\Delta}(q_0, w) = \varnothing$. Usando la función extendida $\widehat{\Delta}$, el lenguaje aceptado por M se puede describir de la siguiente forma:

$$L(M) = \{w \in \Sigma^* : \widehat{\Delta}(q_0, w) \text{ contiene un estado de aceptación}\}.$$

Notación. Sin peligro de ambigüedad, la función extendida $\widehat{\Delta}(q, w)$ se notará simplemente $\Delta(q, w)$.

Demostración del Teorema 2.5.1:

Dado el AFN $M = (\Sigma, Q, q_0, F, \Delta)$, construimos el AFD M' así:

$$M' = (\Sigma, \wp(Q), \{q_0\}, F', \delta)$$

donde

$$\begin{array}{rcl} \delta : \wp(Q) \times \Sigma & \longrightarrow & \wp(Q) \\ (S, a) & \longmapsto & \delta(S, a) := \Delta(S, a). \end{array}$$

$$F' = \{S \subseteq \wp(Q) : S \text{ contiene al menos un estado de aceptación}\}.$$

Se demostrará que $L(M) = L(M')$ probando que, para toda cadena $w \in \Sigma^*$, $\delta(\{q_0\}, w) = \Delta(q_0, w)$. Esta igualdad se demostrará por inducción sobre w.

Para $w = \lambda$, claramente se tiene $\delta(\{q_0\}, \lambda) = \Delta(q_0, \lambda) = \{q_0\}$.

Para $w = a$, $a \in \Sigma$, se tiene

$$\delta(\{q_0\}, a) = \Delta(\{q_0\}, a) = \Delta(q_0, a).$$

Supóngase (hipótesis de inducción) que $\delta(\{q_0\}, w) = \Delta(q_0, w)$, y que $a \in \Sigma$.

$$\begin{array}{rll} \delta(\{q_0\}, wa) = \delta(\delta(\{q_0\}, w), a) & & \text{(definición de la extensión de } \delta) \\ = \delta(\Delta(\{q_0\}, w), a) & & \text{(hipótesis de inducción)} \\ = \Delta(\Delta(\{q_0\}, w), a) & & \text{(definición de } \delta) \\ = \Delta(\{q_0\}, wa) & & \text{(definición de la extensión de } \Delta) \\ = \Delta(q_0, wa) & & \text{(definición de la extensión de } \Delta) \end{array}$$

Esto demuestra el teorema. □

2.6. Autómatas con transiciones λ (AFN-λ)

Un **autómata finito con transiciones λ** (AFN-λ) es un autómata no-determinista $M = (\Sigma, Q, q_0, F, \Delta)$ en el que la función de transición está definida como:

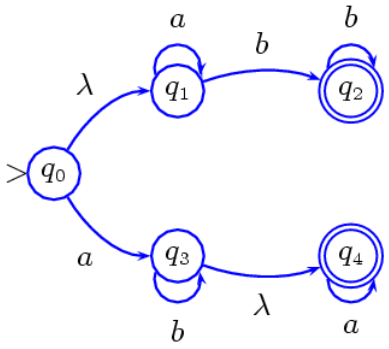
$$\Delta : Q \times (\Sigma \cup \{\lambda\}) \to \wp(Q).$$

La transición $\Delta(q, \lambda) = \{p_{i_1}, \ldots, p_{i_n}\}$, llamada **transición λ** o **transición nula**, tiene el siguiente significado computacional: estando en el estado q, el autómata puede cambiar a uno cualquiera de los estados $p_{i_1}, \ldots, p_{i_n}$, independientemente del símbolo leído y sin mover la unidad de control. Dicho de otra manera, las transiciones λ permiten al autómata cambiar internamente de estado sin procesar o "consumir" el símbolo leído sobre la cinta.

En el diagrama de estados, las transiciones λ dan lugar a arcos con etiquetas λ. Una cadena de entrada w es aceptada por un AFN-λ si existe por lo menos una trayectoria cuyas etiquetas son exactamente los símbolos de w, intercalados con cero, uno o más λs.

El modelo AFN-λ, al igual que el AFN, permite múltiples cómputos para una misma cadena de entrada, así como cómputos abortados. Pero, a diferencia de los AFD y los AFN, en los AFN-λ pueden existir "cómputos infinitos", es decir cómputos que nunca terminan.

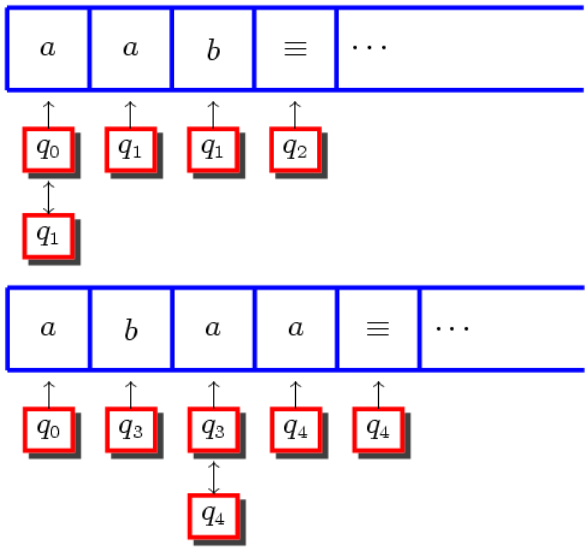
Ejemplo M:



Ejemplos de cadenas aceptadas por M:

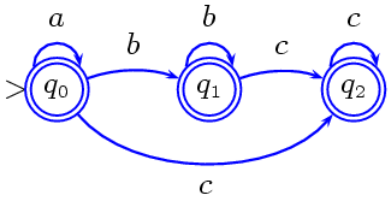
$u = aab$
$v = abaa$
$w = abbaa$

Cómputos de aceptación de $u = aab$ y $v = abaa$:

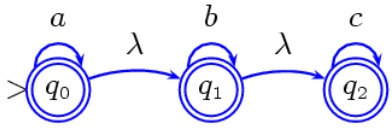


Los AFN-λ permiten aún más libertad en el diseño de autómatas, especialmente cuando hay numerosas concatenaciones.

Ejemplo $\Sigma = \{a, b, c\}$. $L = a^*b^*c^*$. AFD que acepta a L:



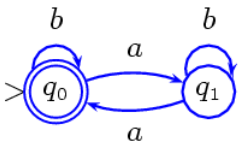
AFN-λ que acepta a L:



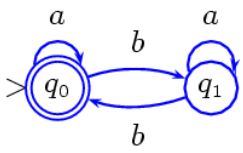
Este autómata se asemeja a la expresión regular $a^*b^*c^*$: las concatenaciones han sido reemplazadas por transiciones λ.

Ejemplo $\Sigma = \{a, b\}$. $L =$ lenguaje de todas las cadenas sobre Σ que tienen un número par de $a$es o un número par de $b$es.

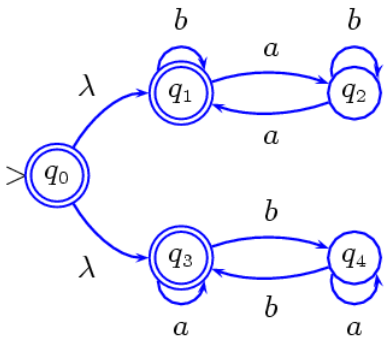
AFD que acepta el lenguaje de las cadenas con un número par de $a$es:



AFD que acepta el lenguaje de las cadenas con un número par de $b$es:



AFN-λ que acepta el lenguaje de las cadenas con un número par de $a$es o un número par de $b$es:



Ejercicios Diseñar AFN-λ que acepten los siguientes lenguajes:

1. $(ab \cup b)^* ab^*$, sobre $\Sigma = \{a, b\}$.
2. $a(a \cup c)^* b^+$, sobre $\Sigma = \{a, b, c\}$.
3. $ab^* \cup ba^* \cup b(ab \cup ba)^*$, sobre $\Sigma = \{a, b\}$.
4. $ab^* ba^* b(ab \cup ba)^*$, sobre $\Sigma = \{a, b\}$.
5. $(0 \cup 010)^* 0^* (01 \cup 10)^*$, sobre $\Sigma = \{0, 1\}$.
6. $0^+ 1(010)^* (01 \cup 10)^* 1^+$, sobre $\Sigma = \{0, 1\}$.
7. $\Sigma = \{a, b\}$. $L =$ lenguaje de todas las cadenas sobre Σ que tienen un número par de $a$es y un número par de $b$es.

2.7. Equivalencia computacional entre los AFN-λ y los AFN

En esta sección se mostrará que el modelo AFN-λ es computacionalmente equivalente al modelo AFN. O dicho más gráficamente, las transiciones λ se pueden eliminar, añadiendo transiciones que las simulen, sin alterar el lenguaje aceptado.

En primer lugar, un AFN $M = (\Sigma, Q, q_0, F, \Delta)$ puede ser considerado como un AFN-λ en el que, simplemente, hay *cero* transiciones λ. Para la afirmación recíproca tenemos el siguiente teorema:

2.7.1 Teorema. *Dado un AFN-λ $M = (\Sigma, Q, q_0, F, \Delta)$, se puede construir un AFN M' equivalente a M, es decir, tal que $L(M) = L(M')$.*

Bosquejo de la demostración. Para construir M' a partir de M se requiere la noción de **λ-clausura de un estado**. Para un estado $q \in Q$, la λ-clausura de q, notada $\lambda[q]$, es el conjunto de estados de M a los que se puede llegar desde q por 0, 1 o más transiciones λ. Nótese que, en general, $\lambda[q] \neq \Delta(q, \lambda)$. Por definición, $q \in \lambda[q]$. La λ-clausura de un conjunto de estados $\{q_1, \ldots, q_k\}$ se define por:

$$\lambda[\{q_1, \ldots, q_k\}] := \lambda[q_1] \cup \cdots \cup \lambda[q_k].$$

Además, $\lambda[\varnothing] := \varnothing$. Sea $M' = (\Sigma, Q, q_0, F', \Delta')$ donde

$$\begin{array}{rcl} \Delta' : Q \times \Sigma & \longrightarrow & \wp(Q) \\ (q, a) & \longmapsto & \Delta'(q, a) := \lambda\big[\Delta(\lambda[q], a)\big]. \end{array}$$

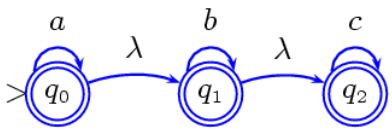
M' simula así las transiciones λ de M teniendo en cuenta todas las posibles trayectorias. F' se define como:

$$F' = \{q \in Q : \lambda[q] \text{ contiene al menos un estado de aceptación}\}.$$

Es decir, los estados de aceptación de M' incluyen los estados de aceptación de M y aquellos estados desde los cuales se puede llegar a un estado de aceptación por medio de una o más transiciones λ. □

Como se puede apreciar, la construcción de M' a partir de M es puramente algorítmica.

Ejemplo Vamos a ilustrar el anterior algoritmo con el AFN-λ M, presentado en el segundo ejemplo de la sección 2.6.



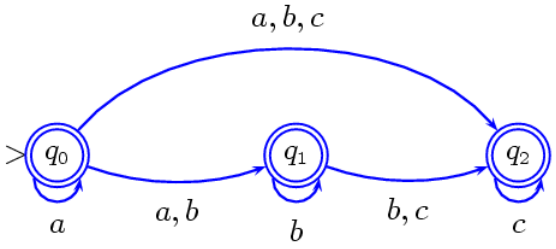
$L(M) = a^*b^*c^*$. Las λ-clausuras de los estados vienen dadas por:

$$\begin{array}{rcl} \lambda[q_0] & = & \{q_0, q_1, q_2\}. \\ \lambda[q_1] & = & \{q_1, q_2\}. \\ \lambda[q_2] & = & \{q_2\}. \end{array}$$

La función de transición $\Delta' : Q \times \{a, b, c\} \to \wp(\{q_0, q_1, q_2\})$ es:

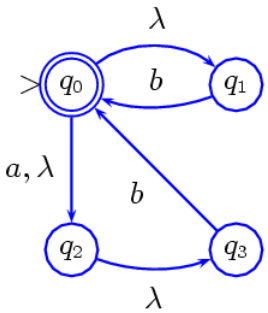
$$\begin{array}{rcl} \Delta'(q_0, a) & = & \lambda\left[\Delta(\lambda[q_0], a)\right] = \lambda\left[\Delta(\{q_0, q_1, q_2\}, a)\right] = \lambda[\{q_0\}] = \{q_0, q_1, q_2\}. \\ \Delta'(q_0, b) & = & \lambda\left[\Delta(\lambda[q_0], b)\right] = \lambda\left[\Delta(\{q_0, q_1, q_2\}, b)\right] = \lambda[\{q_1\}] = \{q_1, q_2\}. \\ \Delta'(q_0, c) & = & \lambda\left[\Delta(\lambda[q_0], c)\right] = \lambda\left[\Delta(\{q_0, q_1, q_2\}, c)\right] = \lambda[\{q_2\}] = \{q_2\}. \\ \Delta'(q_1, a) & = & \lambda\left[\Delta(\lambda[q_1], a)\right] = \lambda\left[\Delta(\{q_1, q_2\}, a)\right] = \lambda[\varnothing] = \varnothing. \\ \Delta'(q_1, b) & = & \lambda\left[\Delta(\lambda[q_1], b)\right] = \lambda\left[\Delta(\{q_1, q_2\}, b)\right] = \lambda[\{q_1\}] = \{q_1, q_2\}. \\ \Delta'(q_1, c) & = & \lambda\left[\Delta(\lambda[q_1], c)\right] = \lambda\left[\Delta(\{q_1, q_2\}, c)\right] = \lambda[\{q_2\}] = \{q_2\}. \\ \Delta'(q_2, a) & = & \lambda\left[\Delta(\lambda[q_2], a)\right] = \lambda\left[\Delta(\{q_2\}, a)\right] = \lambda[\varnothing] = \varnothing. \\ \Delta'(q_2, b) & = & \lambda\left[\Delta(\lambda[q_2], b)\right] = \lambda\left[\Delta(\{q_2\}, b)\right] = \lambda[\varnothing] = \varnothing. \\ \Delta'(q_2, c) & = & \lambda\left[\Delta(\lambda[q_2], c)\right] = \lambda\left[\Delta(\{q_2\}, c)\right] = \lambda[\{q_2\}] = \{q_2\}. \end{array}$$

El autómata M' así obtenido es el siguiente:

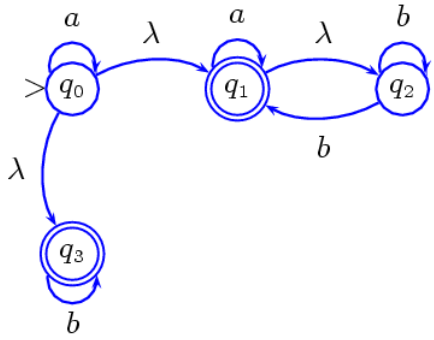


Ejercicios Construir AFN equivalentes a los siguientes AFN-λ:

1.



2.



2.8. Teorema de Kleene. Parte I

En las secciones anteriores se ha mostrado la equivalencia computacional de los modelos AFD, AFN y AFN-λ, lo cual puede ser descrito en la forma:

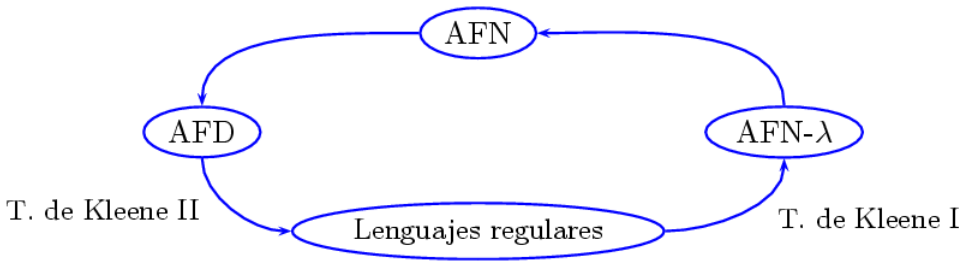
$$\boxed{\text{AFD} \equiv \text{AFN} \equiv \text{AFN-}\lambda}$$

Esto quiere decir que para cada autómata de uno de estos tres modelos se pueden construir autómatas equivalentes en los otros modelos. Por lo tanto, los modelos AFD, AFN y AFN-λ aceptan exactamente los mismos lenguajes. El teorema de Kleene establece que tales lenguajes son precisamente los lenguajes regulares.

2.8.1. ***Teorema de Kleene.*** *Un lenguaje es regular si y sólo si es aceptado por un autómata finito (AFD o AFN o AFN-λ).*

Para demostrar el teorema consideraremos las dos direcciones por separado. Primero demostraremos que para un lenguaje regular L dado existe un AFN-λ tal que $L(M) = L$. En la sección 2.11 demostraremos que, a partir de un AFD M, se puede encontrar una expresión regular R tal que $L(M) = R$. En ambas direcciones las demostraciones son constructivas.

Las construcciones de este capítulo se pueden presentar así:



Parte I. Dada una expresión regular R sobre un alfabeto Σ, se puede construir un AFN-λ M tal que el lenguaje aceptado por M sea exactamente el lenguaje representado por R.

Expresión regular R $\xrightarrow[\text{algorítmico}]{\text{Procedimiento}}$ AFN-λ M tal que $L(M) = R$.

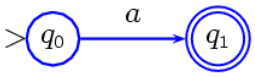
Demostración: Puesto que la definición de las expresiones regulares se hace recursivamente, la demostración se lleva a cabo razonando por inducción sobre R. Para las expresiones regulares básicas, podemos construir fácilmente autómatas que acepten los lenguajes representados. Así, el autómata

acepta el lenguaje $\varnothing$, es decir, el lenguaje representado por la expresión regular $R = \varnothing$.

El autómata

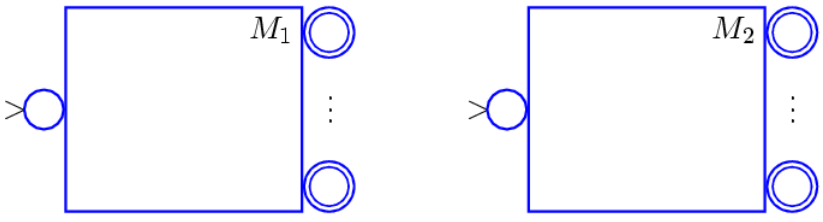
acepta el lenguaje $\{\lambda\}$, es decir, el lenguaje representado por la expresión regular $R = \lambda$.

El autómata



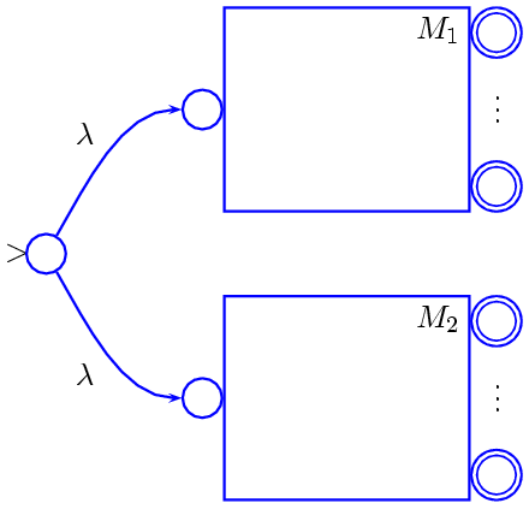
acepta el lenguaje $\{a\}$, $a \in \Sigma$, es decir, el lenguaje representado por la expresión regular $R = a$.

Paso inductivo: supóngase que para las expresiones regulares R y S existen AFN-λ M_1 y M_2 tales que $L(M_1) = R$ y $L(M_2) = S$. Esquemáticamente vamos a presentar los autómatas M_1 y M_2 en la siguiente forma:

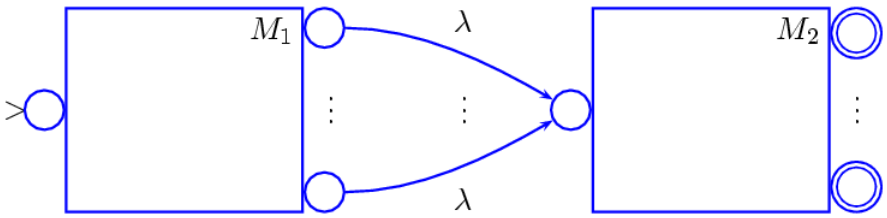


Los estados finales o de aceptación se dibujan a la derecha, pero cabe advertir que el estado inicial puede ser también un estado de aceptación. Obviando ese detalle, podemos ahora obtener AFN-λ que acepten los lenguajes $R \cup S$, RS y R^*.

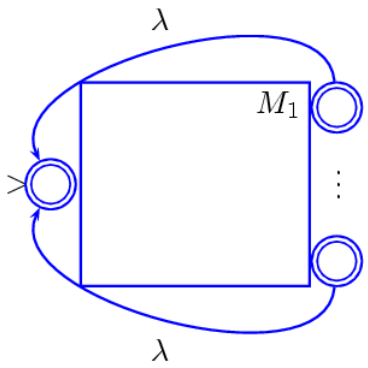
- Autómata que acepta $R \cup S$. Los autómatas M_1 y M_2 se conectan en paralelo y los estados finales del nuevo autómata son los estados finales de M_1 junto con los de M_2:



- Autómata que acepta RS. Los autómatas M_1 y M_2 se conectan en serie y los estados finales del nuevo autómata son únicamente los estados finales de M_2:



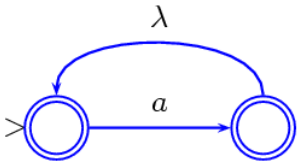
- Autómata que acepta R^*. Los estados finales del nuevo autómata son los estados finales de M_1 junto con el estado inicial.



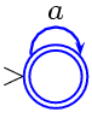
Esto concluye la demostración de la parte I del teorema de Kleene. En la siguiente sección se presentan ejemplos concretos del procedimiento utilizado en la demostración. □

2.9. Ejemplos de la parte I del Teorema de Kleene

De acuerdo con las construcciones presentadas en la demostración de la parte I del Teorema de Kleene, un AFN-λ que acepta el lenguaje a^* es:

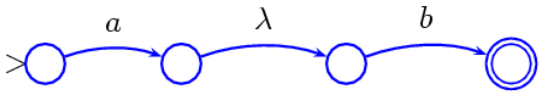


Para simplificar las próximas construcciones utilizaremos, en su lugar, el autómata:

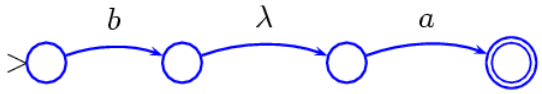


Ejemplo Vamos a utilizar el procedimiento del teorema para construir un AFN-λ que acepte el lenguaje $a^*(ab \cup ba)^* \ \cup \ a(b \cup a^*)$ sobre el alfabeto $\{a, b\}$.

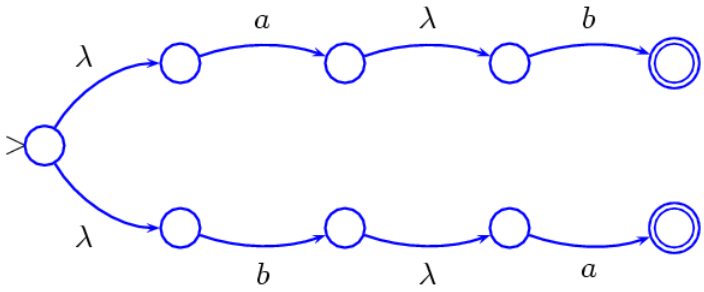
Autómata que acepta ab:



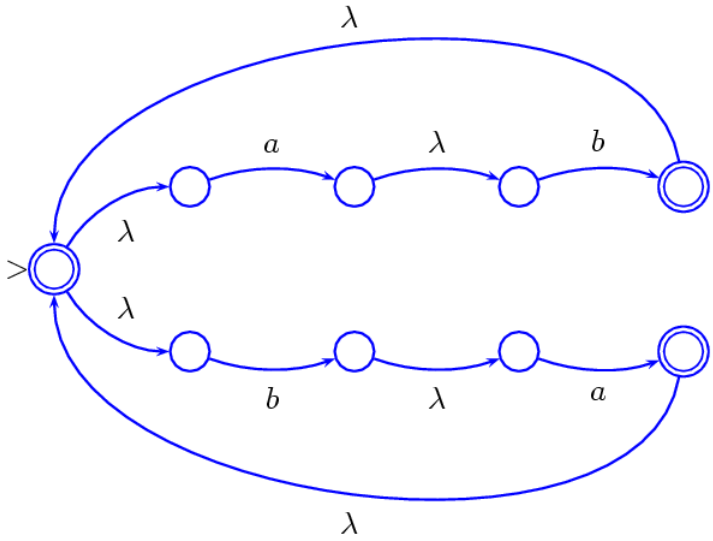
Autómata que acepta ba:



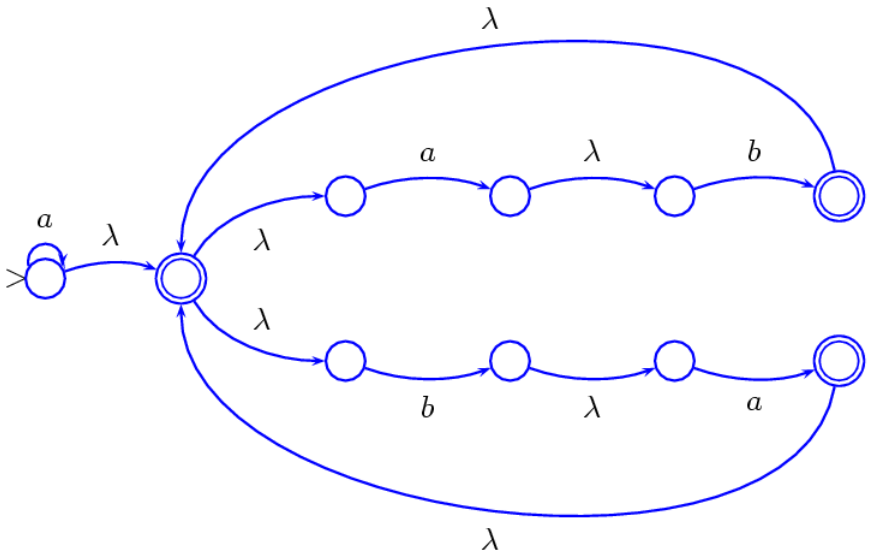
Autómata que acepta $ab \cup ba$:



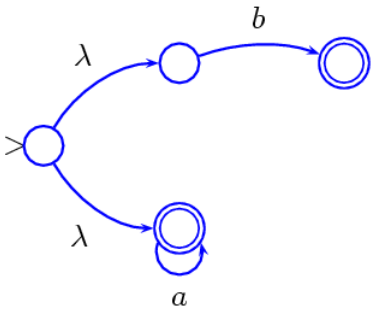
Autómata que acepta $(ab \cup ba)^*$:



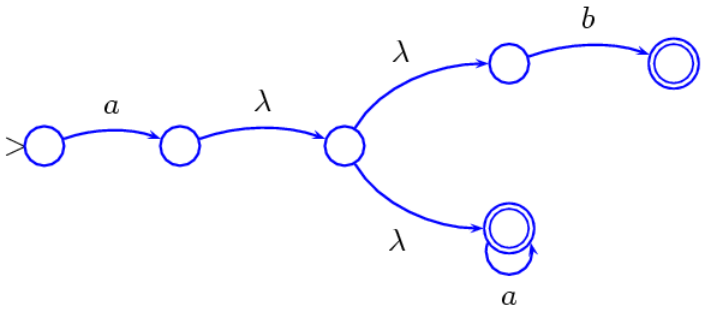
Autómata que acepta $a^*(ab \cup ba)^*$:



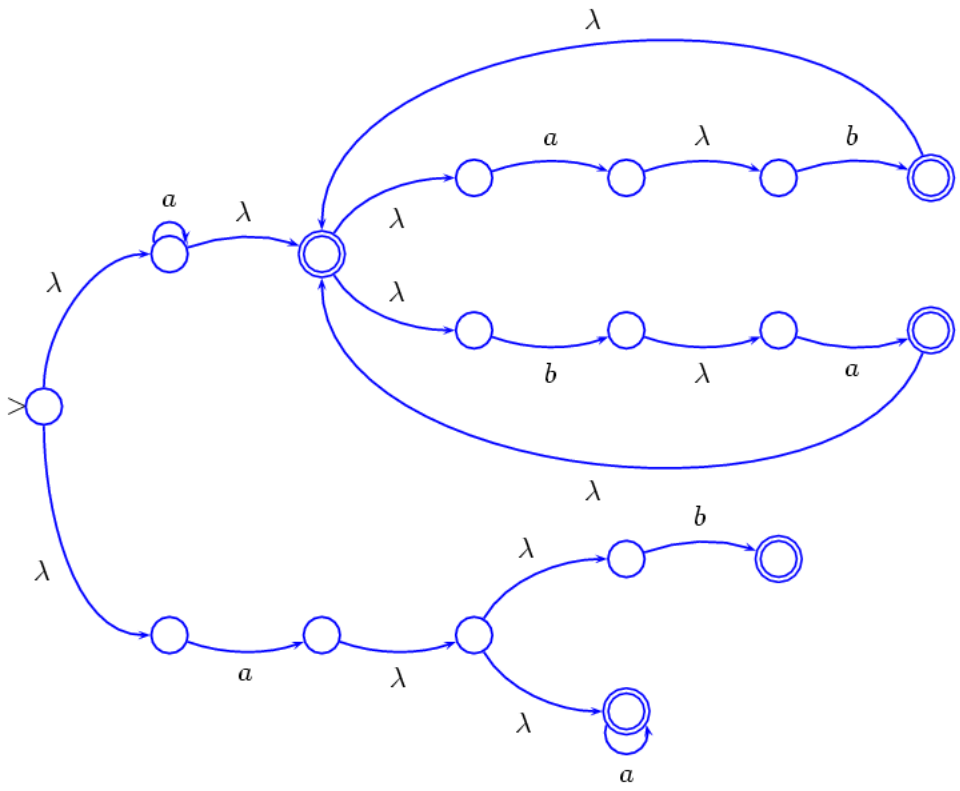
Autómata que acepta $b \cup a^*$:



Autómata que acepta $a(b \cup a^*)$:



Autómata que acepta $a^*(ab \cup ba)^* \cup a(b \cup a^*)$:



Ejercicios Diseñar autómatas AFN-λ que acepte los siguientes lenguajes sobre $\Sigma = \{a, b, c\}$:

1. $a^*(b \cup ab^* \cup ab^*a)c^* \cup (a \cup b)(a \cup ac)^*$.
2. $c^*a(a \cup ba)^*(abc)^* \cup c^*(a \cup cb^*c)$.
3. $(ac)^* \cup a(a \cup ab^*a) \cup (abc)^*(cba)^* \cup (c \cup ab \cup ba \cup ca)^*(ca \cup cb)^*$.

2.10. Lema de Arden

Vamos a utilizar el siguiente resultado, conocido como "lema de Arden", para demostrar la segunda parte del teorema de Kleene.

2.10.1. ***Lema de Arden.*** *Si A y B son lenguajes sobre un alfabeto Σ y $\lambda \notin A$, entonces la ecuación $X = AX \cup B$ tiene una única solución dada por $X = A^*B$.*

Demostración: Si X es una solución de $X = AX \cup B$, entonces $B \subseteq AX \cup B = X$. También se tiene $AX \subseteq X$; a partir de esta contenencia y usando inducción sobre n, se puede demostrar que $A^n X \subseteq X$ para todo $n \in \mathbb{N}$. Por lo tanto

$$A^n B \subseteq A^n X \subseteq X$$

para todo $n \in \mathbb{N}$. Así que

$$A^* B = \left(\bigcup_{n \geq 0} A^n \right) B = \bigcup_{n \geq 0} A^n B \subseteq X.$$

Esto muestra que toda solución de $X = AX \cup B$ contiene a A^*B y es fácil verificar que, de hecho, A^*B es una solución:

$$A(A^*B) \cup B = A^+ B \cup B = (A^+ \cup \lambda)B = A^* B.$$

Para la unicidad, demostraremos que si $A^*B \cup C$, con $C \cap A^*B = \varnothing$, es una solución de la ecuación, entonces $C = \varnothing$.

$$\begin{aligned} A^*B \cup C &= A(A^*B \cup C) \cup B \\ &= A^+B \cup AC \cup B \\ &= (A^+ \cup \lambda)B \cup AC \\ &= A^*B \cup AC. \end{aligned}$$

Intersectando con C ambos lados de la anterior igualdad, se tiene:

$$\begin{aligned} (A^*B \cap C) \cup C &= (A^*B \cap C) \cup (AC \cap C), \\ C &= AC \cap C. \end{aligned}$$

Por lo tanto, $C \subseteq AC$. Si se tuviera $C \neq \varnothing$, existiría una cadena $u \in C$ de longitud mínima. Entonces $u = vw$, con $v \in A$, $w \in C$. Como $\lambda \notin A$, $v \neq \lambda$; por consiguiente $|w| < |u|$. Esta contradicción muestra que necesariamente $C = \varnothing$, tal como se quería. □

Ejemplo La ecuación $X = aX \cup b^*ab$ tiene solución única $X = a^*b^*ab$.

Ejemplo La ecuación $X = a^2X \cup b^+X \cup ab$ se puede escribir en la forma $X = (a^2 \cup b^+)X \cup ab$. Por el lema de Arden la ecuación tiene solución única $X = (a^2 \cup b^+)^*ab$.

Ejemplo La ecuación $X = ab^2X \cup aX \cup a^*b \cup b^*a$ se puede escribir como $X = (ab^2 \cup a)X \cup (a^*b \cup b^*a)$. Por lema de Arden la ecuación tiene solución única $X = (ab^2 \cup a)^*(a^*b \cup b^*a)$.

Ejercicios Encontrar las soluciones (únicas) de las siguientes ecuaciones.

1. $X = aX \cup bX$.
2. $X = aX \cup b^*ab \cup bX \cup a^*$.

Ejercicio Demostrar de si $\lambda \in A$, entonces Y es una solución de la ecuación $X = AX \cup B$ si y solo si $Y = A^*(B \cup D)$ para algún $D \subseteq \Sigma^*$.

2.11. Teorema de Kleene. Parte II

En esta sección demostraremos que para todo AFN $M = (\Sigma, Q, q_0, F, \Delta)$ existe una expresión regular R tal que $L(M) = R$.

Un autómata tiene un único estado inicial pero cambiando dicho estado surgen nuevos autómatas. Para cada $q_i \in Q$, sea M_i el autómata que coincide con M pero con estado inicial q_i; más precisamente, $M_i = (\Sigma, Q, q_i, F, \Delta)$. Al lenguaje aceptado por M_i lo denotaremos A_i; es decir, $L(M_i) = A_i$. En particular, $A_0 = L(M)$. Puesto que los estados de aceptación no se han alterado, se tiene que

$$A_i = \{w \in \Sigma^* : \Delta(q_i, w) \cap F \neq \varnothing\}.$$

Cada A_i se puede escribir como

$$A_i = \begin{cases} \bigcup\limits_{a\in\Sigma} \{aA_j : q_j \in \Delta(q_i, a)\}, & \text{si } q_i \notin F, \\ \bigcup\limits_{a\in\Sigma} \{aA_j : q_j \in \Delta(q_i, a)\} \cup \lambda. & \text{si } q_i \in F. \end{cases} \tag{2.1}$$

Si $Q = \{q_0, q_1, \ldots, q_n\}$, las igualdades de la forma (2.1) dan lugar a un sistema de $n+1$ ecuaciones con $n+1$ incógnitas (los A_i):

$$\begin{cases} A_0 &= C_{01}A_0 \cup C_{02}A_1 \cup \cdots \cup C_{0n}A_n & (\cup\lambda) \\ A_1 &= C_{11}A_0 \cup C_{12}A_1 \cup \cdots \cup C_{1n}A_n & (\cup\lambda) \\ \vdots & \vdots & \vdots \\ A_n &= C_{n1}A_0 \cup C_{n2}A_1 \cup \cdots \cup C_{nn}A_n & (\cup\lambda) \end{cases}$$

donde cada coeficiente C_{ij} o es $\varnothing$ o es un símbolo de Σ. El término λ se añade a una ecuación solamente si el estado correspondiente es un estado de aceptación.

Utilizando sucesivas veces el lema de Arden, se puede mostrar que este sistema de ecuaciones siempre se puede solucionar y su solución es única. En efecto,

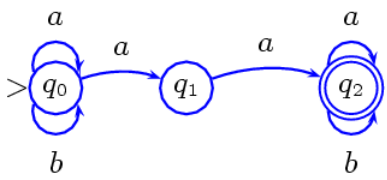
comenzando con la última ecuación, se escribe A_n en términos de los demás A_i, para lo cual se usa el lema de Arden si es necesario. Este valor de A_n se reemplaza en las demás ecuaciones y el sistema se reduce a n ecuaciones con n incógnitas. Similarmente, A_{n-1} se escribe en términos de los demás A_i, usando el lema de Arden si es necesario, y tal valor se reemplaza en las ecuaciones restantes. Prosiguiendo de esta manera, el sistema original se reduce a una sola ecuación cuya incógnita es precisamente A_0. Esta ecuación se soluciona recurriendo una vez más al lema de Arden. Puesto que los coeficientes C_{ij} diferentes de $\varnothing$ son símbolos de Σ, se obtiene una expresión regular R tal que $L(M) = A_0 = R$.

Ejercicio ¿Por qué la anterior demostración no es válida para autómatas con transiciones λ?

2.12. Ejemplos de la parte II del Teorema de Kleene

A continuación ilustraremos el procedimiento de la sección 2.11 para encontrar $L(M)$ a partir de un AFN $M = (\Sigma, Q, q_0, F, \Delta)$ dado.

Ejemplo Considérese el siguiente AFN M:



Por simple inspección sabemos que $L(M) = (a \cup b)^* a^2 (a \cup b)^*$, pero utilizaremos el método descrito para encontrar explícitamente $L(M)$.

El sistema de ecuaciones asociado con el autómata M es:

$$\begin{cases} (1) & A_0 = aA_0 \cup bA_0 \cup aA_1 \\ (2) & A_1 = aA_2 \\ (3) & A_2 = aA_2 \cup bA_2 \cup \lambda. \end{cases}$$

La ecuación (3) se puede escribir como

(4) $\quad A_2 = (a \cup b)A_2 \cup \lambda.$

Aplicando el Lema de Arden en (4):

(5) $\quad A_2 = (a \cup b)^* \lambda = (a \cup b)^*.$

Reemplazando (5) en (2):

(6) $A_1 = a(a \cup b)^*$.

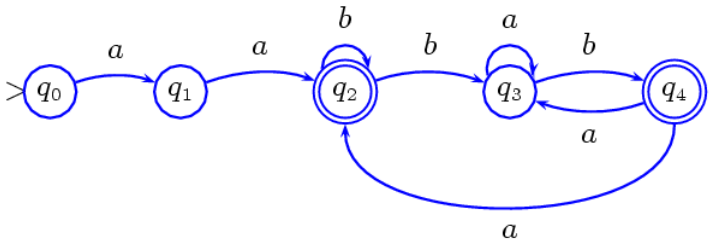
Reemplazando (6) en (1):

(7) $A_0 = (a \cup b)A_0 \cup a^2(a \cup b)^*$.

Aplicando el Lema de Arden en (7) concluimos:

$$\boxed{A_0 = (a \cup b)^* a^2 (a \cup b)^*}$$

Ejemplo Encontrar una expresión regular para el lenguaje aceptado por el siguiente AFN M:



El sistema de ecuaciones asociado con el autómata M es:

$$\begin{cases} (1) & A_0 = aA_1 \\ (2) & A_1 = aA_2 \\ (3) & A_2 = bA_2 \cup bA_3 \cup \lambda \\ (4) & A_3 = aA_3 \cup bA_4 \\ (5) & A_4 = aA_2 \cup aA_3 \cup \lambda \end{cases}$$

Reemplazando (5) en (4):

(6) $A_3 = aA_3 \cup baA_2 \cup baA_3 \cup b = (a \cup ba)A_3 \cup baA_2 \cup b$.

Aplicando el Lema de Arden en (6):

(7) $A_3 = (a \cup ba)^*(baA_2 \cup b) = (a \cup ba)^* baA_2 \cup (a \cup ba)^* b$.

Reemplazando (7) en (3):

(8) $A_2 = bA_2 \cup b(a \cup ba)^* baA_2 \cup b(a \cup ba)^* b \cup \lambda$.

El sistema original de cinco ecuaciones y cinco incógnitas se reduce al sistema de tres ecuaciones y tres incógnitas formado por (1), (2) y (8).

La ecuación (8) se puede escribir como

(9) $A_2 = \big[b \cup b(a \cup ba)^* ba\big] A_2 \cup b(a \cup ba)^* \cup \lambda$.

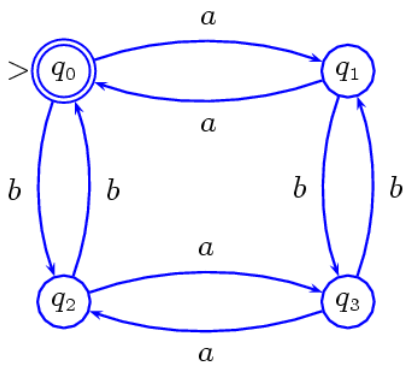
Aplicando el Lema de Arden en (9):

(10) $A_2 = \big[b \cup b(a \cup ba)^* ba\big]^* \big[b(a \cup ba)^* b \cup \lambda\big]$.

Si se sustituye (10) en (2) y luego el valor de A_1 obtenido se sustituye en (1), se obtiene finalmente:

$$\boxed{A_0 = a^2 \big[b \cup b(a \cup ba)^* ba\big]^* \big[b(a \cup ba)^* b \cup \lambda\big]}$$

Ejemplo Encontrar una expresión regular para el lenguaje L de todas las cadenas sobre $\Sigma = \{a, b\}$ que tienen un número par de $a$es y un número par de $b$es. El siguiente autómata acepta el lenguaje L:



Este autómata da lugar al siguiente sistema de ecuaciones:

$$\begin{cases} (1) & A_0 = aA_1 \cup bA_2 \cup \lambda \\ (2) & A_1 = aA_0 \cup bA_3 \\ (3) & A_2 = aA_3 \cup bA_0 \\ (4) & A_3 = aA_2 \cup bA_1 \end{cases}$$

Reemplazando (4) en (3):

(5) $A_2 = a^2 A_2 \cup abA_1 \cup bA_0$.

Reemplazando (4) en (2):

(6) $A_1 = aA_0 \cup baA_2 \cup b^2 A_1$.

El sistema original de cuatro ecuaciones y cuatro incógnitas se reduce a un sistema de tres ecuaciones y tres incógnitas, a saber:

$$\begin{cases} (1) & A_0 = aA_1 \cup bA_2 \cup \lambda \\ (6) & A_1 = aA_0 \cup baA_2 \cup b^2 A_1 \\ (5) & A_2 = a^2 A_2 \cup abA_1 \cup bA_0 \end{cases}$$

Aplicando el Lema de Arden en (5):

(7) $A_2 = (a^2)^*(abA_1 \cup bA_0) = (a^2)^*abA_1 \cup (a^2)^*bA_0.$

Reemplazando (7) en (6):

(8) $A_1 = aA_0 \cup ba(a^2)^*abA_1 \cup ba(a^2)^*bA_0 \cup b^2A_1.$

Reemplazando (7) en (1):

(9) $A_0 = aA_1 \cup b(a^2)^*abA_1 \cup b(a^2)^*bA_0 \cup \lambda.$

El sistema se reduce ahora a dos ecuaciones:

$$\begin{cases} (9) \quad A_0 = aA_1 \cup b(a^2)^*abA_1 \cup b(a^2)^*bA_0 \cup \lambda \\ (8) \quad A_1 = aA_0 \cup ba(a^2)^*abA_1 \cup ba(a^2)^*bA_0 \cup b^2A_1 \\ \qquad\quad = \big(ba(a^2)^*ab \cup b^2\big)A_1 \cup aA_0 \cup ba(a^2)^*bA_0. \end{cases}$$

Aplicando el Lema de Arden en (8):

$$\begin{aligned} A_1 &= \big(ba(a^2)^*ab \cup b^2\big)^*\big(aA_0 \cup ba(a^2)^*bA_0\big) \\ &= \big(ba(a^2)^*ab \cup b^2\big)^*aA_0 \cup \big(ba(a^2)ab \cup b^2\big)^*ba(a^2)^*bA_0. \end{aligned} \tag{10}$$

Haciendo $R = \big(ba(a^2)^*ab \cup b^2\big)^*$, (10) se puede escribir como

(11) $A_1 = RaA_0 \cup Rba(a^2)^*bA_0.$

Aplicando el Lema de Arden en (9):

$$\begin{aligned} A_0 &= \big(b(a^2)^*b\big)^*\big(aA_1 \cup b(a^2)^*abA_1 \cup \lambda\big) \\ &= \big(b(a^2)^*b\big)^*aA_1 \cup \big(b(a^2)^*b\big)^*b(a^2)^*abA_1 \cup \big(b(a^2)^*b\big)^*. \end{aligned} \tag{12}$$

Haciendo $S = \big(b(a^2)^*b\big)^*$, (12) se puede escribir como:

(13) $A_0 = SaA_1 \cup Sb(a^2)^*abA_1 \cup S.$

Al sustituir (11) en (13), el sistema original se reduce a una sola ecuación:

(14) $A_0 = Sa\big[RaA_0 \cup Rba(a^2)^*bA_0\big] \cup Sb(a^2)^*ab\big[RaA_0 \cup Rba(a^2)^*bA_0\big] \cup S.$

Agrupando los términos en los que aparece A_0 y factorizando, se obtiene

(15) $A_0 = \big[SaRa \cup SaRba(a^2)^*b \cup Sb(a^2)^abRa \cup Sb(a^2)^*abRba(a^2)^*b\big]A_0 \cup S.$

Aplicando Lema de Arden en (15):

(16) $A_0 = \big[SaRa \cup SaRba(a^2)^*b \cup Sb(a^2)^*abRa \cup Sb(a^2)^*abRba(a^2)^*b\big]^*S.$

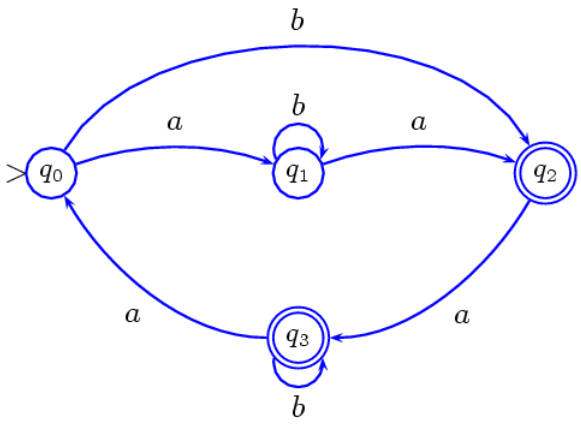
Si sustituimos R y S en (16) obtenemos una expresión regular para L.

Ejercicios Utilizando el lema de Arden, encontrar expresiones regulares para los siguientes lenguajes sobre $\Sigma = \{a, b\}$:

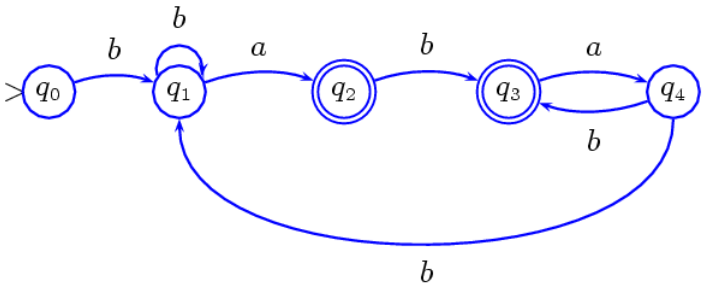
1. El lenguaje L de todas las cadenas que tienen un número par de $a$es y un número impar de $b$es.
2. El lenguaje L de todas las cadenas que tienen un número par de $a$es o un número impar de $b$es.

Ejercicios Utilizando el lema de Arden, encontrar expresiones regulares para los lenguajes aceptados por los siguientes AFN:

1.



2.



3.