Autómatas y Lenguajes Formales
Nota 03. Autómatas de estados finitos*

Noé Salomón Hernández S.

La teoría de la computación comienza con una pregunta: ¿qué es una computadora? Las computadoras reales son bastante complicadas, en su lugar usamos una computadora idealizada llamada **modelo computacional**. Un modelo computacional puede ser preciso en ciertas maneras y tal vez impreciso en otras. Empezamos con el modelo más sencillo, llamado *autómata de estados finito*.

1. Autómata de estados finito

Un autómata de estados finito es un modelo para computadoras con una memoria limitada. ¿Qué puede hacer una computadora con tan poca memoria? Muchas cosas, de hecho interactuamos con tales computadoras todo el tiempo, pues se encuentran en varios dispositivos electromecánicos.

El control de una puerta automática es un ejemplo. La puerta se abre hacia adentro cuando detecta a una persona que se aproxima y desea ingresar al interior del edificio o centro comercial. La puerta automática tiene una almohadilla enfrente para detectar la presencia de una persona que está por entrar y cruzar la puerta. Otra almohadilla se encuentra detrás de la puerta para que el controlador pueda mantener la puerta abierta por el tiempo suficiente para que la persona pase por completo, y también para que la puerta no se abra y golpee a alguien que está detrás de la puerta. Esta configuración se muestra en la siguiente figura:

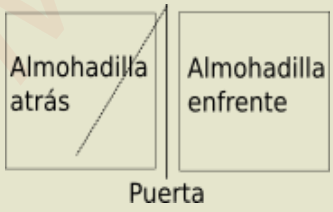


Figura 1: Puerta automática vista desde arriba. La puerta abre hacia adentro.

El controlador de la puerta está en dos estados: ABIERTO o CERRADO. Como se aprecia en las siguientes figuras, hay cuatro posibles condiciones de entrada: ENFRENTE (hay una persona en la almohadilla de enfrente), ATRÁS (hay una persona en la almohadilla de atrás), AMBAS (hay personas en ambas almohadillas), y NINGUNO (no hay personas en las almohadillas).

El controlador se mueve de un estado a otro, dependiendo de la entrada. Cuando está en el estado CERRADO y recibe como entrada NINGUNO o ATRÁS, el controlador permanece en el estado CERRADO. Si la entrada AMBAS se recibe, el controlador permanece en el estado CERRADO porque

*Esta nota se basa en libro de M. Sipser. *Introduction to the Theory of Computation* y en los textos del profesor Rajeev Motwani, los cuales pueden encontrar aquí.

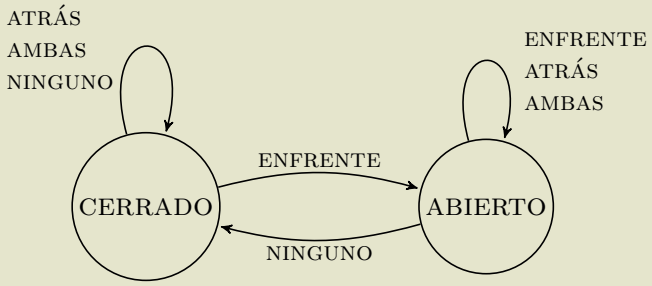


Figura 2: Diagrama de estados del controlador de una puerta automática.

	NINGUNO	ENFRENTE	ATRÁS	AMBAS
CERRADO	CERRADO	ABIERTO	CERRADO	CERRADO
ABIERTO	CERRADO	ABIERTO	ABIERTO	ABIERTO

Figura 3: Tabla de transición entre estados para el controlador de la puerta automática.

al abrir la puerta se corre el riesgo de golpear a alguien que esté en la almohadilla trasera. Si se tiene la entrada ENFRENTE, el controlador pasa al estado ABIERTO. En el estado ABIERTO, si la entrada ENFRENTE, ATRÁS, o AMBAS llega, entonces permanece ABIERTO. Si se tiene la entrada NINGUNO, el controlador regresa al estado CERRADO.

Por ejemplo, el controlador puede iniciar en el estado CERRADO y recibir la secuencia de entradas ENFRENTE, ATRÁS, NINGUNO, ENFRENTE, AMBAS, NINGUNO, ATRÁS y NINGUNO. Entonces se tendrá que visitar la siguiente secuencia de estados CERRADO (inicial), ABIERTO, ABIERTO, CERRADO, ABIERTO, ABIERTO, CERRADO, CERRADO y CERRADO, respectivamente. Este controlador es una computadora que tiene un sólo bit de memoria. En un controlador para un elevador un estado puede representar el piso en el que se encuentra y las entradas podrían ser las señales recibidas mediante los botones.

Algunas aplicaciones para los autómatas de estados finitos son:

1. software para el diseño y verificación de circuitos digitales,
2. analizadores léxicos para compiladores,
3. búsquedas en grandes cuerpos de texto (motor de búsquedas web, `grep`, ...),
4. diseño, verificación e implementación de sistemas de software que involucran interacción (protocolos de red, comercio electrónico, ...).

Necesitamos desarrollar una definición precisa, así que comenzaremos describiendo la teoría matemática de los autómatas finitos sin hacer referencia a una aplicación particular. La siguiente figura representa un autómata de estados finito llamado M_1.

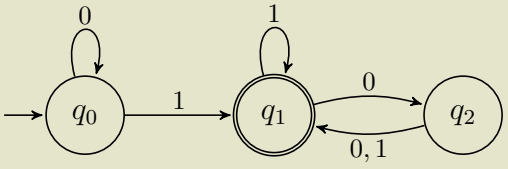


Figura 4: Un autómata de estados finito llamado M_1 con tres estados.

La figura anterior es llamada el *diagrama de estados* de M_1. Tiene tres **estados**, etiquetados con q_0, q_1 y q_2. El **estado inicial** q_0, se indica por una flecha que no tiene origen. El **estado final** o de **aceptación**, q_1, es el que tiene doble círculo. Las flechas yendo de un estado a otro son llamadas **transiciones**. Cuando un autómata recibe una cadena de entrada, la procesa para producir una salida. La salida es **aceptar** o **rechazar**. El procesamiento comienza con M_1 en el estado inicial. El autómata recibe los símbolos de la cadena de entrada uno por uno de izquierda a derecha. Depués de leer cada símbolo, M_1 se mueve de un estado al otro a lo largo de una transición que tiene a ese símbolo como su etiqueta. Cuando lee el último símbolo, M_1 produce su salida. La salida es *aceptar* si M_1 está en un estado final y *rechazar* si no. Para el autómata M_1 de la figura anterior se acepta cualquier cadena que termine con 1. Además, M_1 acepta cualquier cadena que termina con un número par de 0s que figuran después del último 1.

En resumen, los autómatas finitos tienen las siguientes características:

- son los modelos más simples de computación,
- describen la clase de programas llamada *regulares*,
- siempre operan en un estado, de los cuales hay un número finito; el cambio de estado es en respuesta a un símbolo de entrada; y la entrada original se acepta si al terminar se llega a un estado final o de aceptación.

2. Autómatas finitos deterministas

Definición 2.1 Formalmente, un autómata finito determinista (AFD) M es una tupla $M = (Q, \Sigma, \delta, q_0, F)$ donde:

- Q - es un conjunto finito de estados;
- Σ - es el alfabeto de entrada;
- q_0 - es el estado inicial, con $q_0 \in Q$;
- F - es el conjunto de estados finales, $F \subseteq Q$;
- $\delta : Q \times \Sigma \to Q$ - es la función de transición entre estados, donde $\delta(q, a) = p$ se representa en el diagrama de estados como la transición $(q) \xrightarrow{a} (p)$. δ puede ser dada como un diagrama de estados o como una tabla de transición. Por ejemplo, el autómata de la Fig. 4 tiene como tabla de transición:

		0	1
$\to$	q_0	q_0	q_1
F	q_1	q_2	q_1
	q_2	q_1	q_1

  donde $\to$ indica el estado inicial y con **F** señalamos a los estados finales.

2.1. Función de transición extendida

Buscamos ahora extender δ de modo que podamos realizar transiciones múltiples en un autómata finito determinista. Hasta ahora δ efectúa una sola transición sobre el símbolo de entrada $a \in \Sigma$, deseamos encontrar una $\widehat{\delta}$ que efectúe una secuencia de transiciones sobre la subcadena $x \in \Sigma^*$. Así $\widehat{\delta}(q, x) = p$ denota que iniciando en el estado q, la porción x de la cadena de entrada hará que el autómata finito llegue al estado p. Para el autómata de la Figura 4, nos gustaría tener

$$\begin{aligned} \widehat{\delta}(q_0, 00) &= q_0 & \qquad \widehat{\delta}(q_0, 101) &= q_1 \\ \widehat{\delta}(q_0, 1000) &= q_2 & \qquad \widehat{\delta}(q_0, 10100) &= q_1 \end{aligned}$$

Definición 2.2 La función de transición extendida, $\widehat{\delta} : Q \times \Sigma^* \to Q$, se define inductivamente en términos de δ como sigue. Sea $q \in Q$, $a \in \Sigma$ y $x \in \Sigma^*$, entonces

$$\widehat{\delta}(q, \varepsilon) = q, \tag{2.1.1}$$

$$\widehat{\delta}(q, xa) = \delta(\widehat{\delta}(q, x), a). \tag{2.1.2}$$

Ejemplo 2.3 Tomando al autómata M_1 de la Figura 4, deseamos saber a donde nos lleva al procesar la cadena 101 iniciando en el estado q_0. Tenemos, entonces:

$$\begin{aligned} \widehat{\delta}(q_0, \varepsilon) &= q_0 \\ \widehat{\delta}(q_0, 1) &= \delta(\widehat{\delta}(q_0, \varepsilon), 1) = \delta(q_0, 1) = q_1 \\ \widehat{\delta}(q_0, 10) &= \delta(\widehat{\delta}(q_0, 1), 0) = \delta(q_1, 0) = q_2 \\ \widehat{\delta}(q_0, 101) &= \delta(\widehat{\delta}(q_0, 10), 1) = \delta(q_2, 1) = q_1 \end{aligned}$$

Partiendo del estado q_0, al terminar de procesar la cadena 101 el autómata M_1 llega al estado q_1, el cual es final por lo que se acepta dicha cadena.

Proposición 2.4 Para todo $a \in \Sigma$ y para todo $q \in Q$, la función de transición extendida para los autómatas finitos deterministas cumple que:

$$\widehat{\delta}(q, a) = \delta(q, a).$$

Demostración.

$$\begin{aligned} \widehat{\delta}(q, a) &= \delta(\widehat{\delta}(q, \varepsilon), a) \\ &= \delta(q, a). \end{aligned}$$

⊣

Así, $\widehat{\delta}$ y δ coinciden en cadenas de longitud uno, y δ sólo está definida sobre cadenas de dicha longitud. Por consiguiente, establecemos la convención de llamar a $\widehat{\delta}$ como δ sin haber confusión alguna.

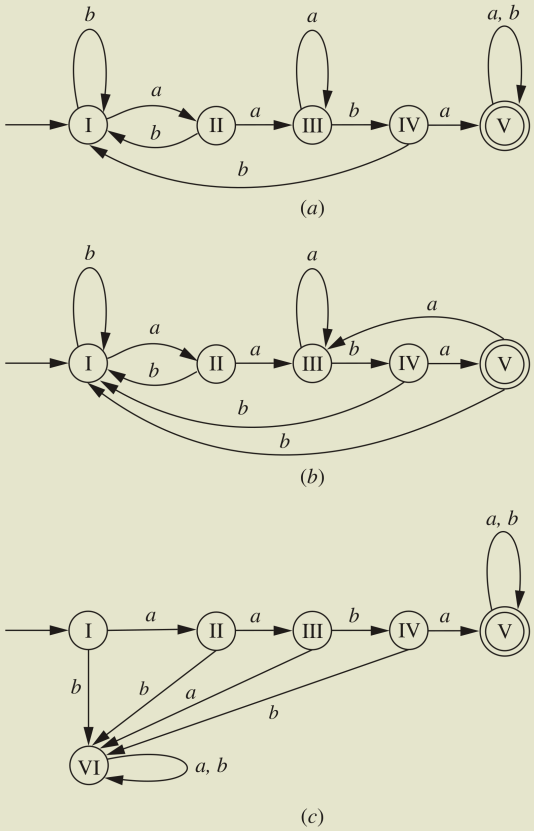
Definición 2.5 Lenguaje de un autómata finito determinista. Sean $M = (Q, \Sigma, \delta, q_0, F)$ un autómata finito determinista y $L(M)$ el conjunto de todas las cadenas aceptadas por M. $L(M)$ se conoce como el lenguaje de M. Tenemos que

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

Ejercicios 2.6 Diseñe autómatas finitos deterministas para los lenguajes siguientes:

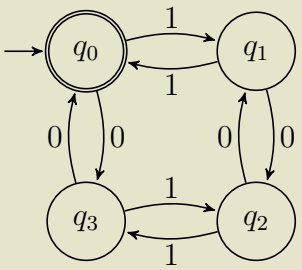
1. Representaciones binarias de enteros divisibles entre 3. Acepte representaciones únicas para todo número, por ejemplo: acepte 1001 pero rechace 001001.
2. $\{w \in \{a,b\}^* \,|\, w$ tiene exactamente dos a's y al menos dos b's$\}$.
3. $\{w \in \{a,b\}^* \,|\, w$ no es a ni es $b\}$.
4. $\{w \in \{a,b\}^* \,|\, w$ contiene tanto a bb como a aba como subcadenas$\}$.
5. $\{w \in \{0,1\}^* \,|\, w$ contiene al menos dos 0's y a lo más un 1$\}$.
6. $B_n = \{a^k \,|\,$ donde k es un múltiplo de $n\}$, con $n \geq 1$.

Ejercicios 2.7 Describa en español el lenguaje que generan los siguientes AFD.



(a) (b) (c)

Ejercicios 2.8 Describa en español el lenguaje que reconoce el siguiente AFD.



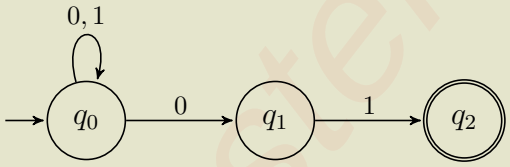
3. Autómatas finitos no-deterministas

En un autómata finito determinista la elección de $\delta(q, a)$ es única. Cada estado q tiene una flecha que sale por cada $a \in \Sigma$. Lo que significa que para una cadena de entrada específica w, la ejecución sobre el autómata es totalmente predecible y repetible.

En los autómatas finitos no-deterministas (AFN) se tiene que:

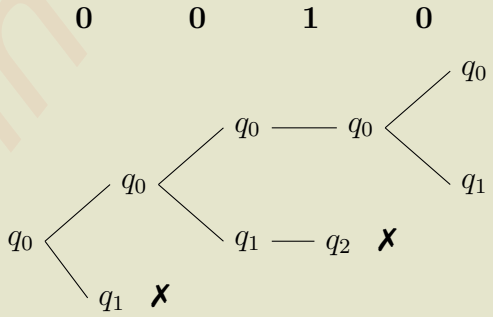
- $\delta(q, a) = \{q_i, \ldots, q_j\}$, así la elección de $\delta(q, a)$ se toma de un conjunto de estados. Por lo que es posible tener más de un estado, y también no tener ningún estado, de donde elegir.
- Así, un estado q puede tener múltiples flechas que salgan de él etiquetadas con $a \in \Sigma$. O bien, puede no tener ninguna flecha con etiqueta $a \in \Sigma$, en cuyo caso el estado se encuentra "atascado".
- El tener múltiples opciones permite al AFN "adivinar" la acción correcta, en lugar de tenerla definida de antemano.

Ejemplo 3.1 Diseñe un AFN N_{01} para el lenguaje $L_{01} = \{w \in \{0,1\}^* \mid w \text{ termina en } 01\}$. La idea es que N_{01} use el no determinismo para identificar correctamente el penúltimo símbolo, que debe ser 0, y entonces reconozca el 1. El autómata es



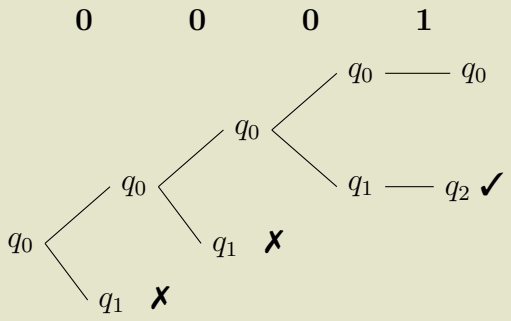
Observemos que el no determinismo implica que en lugar de tener una única trayectoria de ejecución, como ocurre en un AFD, tenemos un árbol de posibles trayectorias de ejecución.

Ejemplo 3.2 El árbol de trayectorias de ejecución que obtiene el autómata N_{01} del Ejemplo 3.1 al procesar la cadena $w = 0010$ es:



Los nodos marcados con ✗ son trayectorias estancadas. Al terminar de procesar w en su totalidad, llegamos a los estados q_0 y q_1, como no son finales la cadena w se rechaza.

Ejemplo 3.3 El árbol de trayectorias de ejecución que obtiene el autómata N_{01} al procesar la cadena $w = 0001$ es:



La cadena w se acepta pues entre los estados a los que llega el autómata al terminar de procesar w está el estado final q_2.

Aceptación. Cuando existe al menos una trayectoria de ejecución que termina en un estado final.

Rechazo. Cuando todas las posibles trayectorias de ejecución se estancan o terminan en un estado que no es final.

Intuitivamente podemos interpretar el no determinismo en un AFN N como:

- si N siempre tomara las *elecciones correctas* para asegurarse de elegir la trayectoria de aceptación, si es que existe; o bien, como
- si N explorara múltiples trayectorias en paralelo.

Observación. Un AFN N que reconoce las cadenas de un lenguaje L debe asegurarse que:

- Para toda cadena $x \notin L$ todas las trayectorias de ejecución son de rechazo.
- Para toda cadena $x \in L$ al menos hay una trayectoria de aceptación.

Así, mientras N es libre de elegir la trayectoria, debemos verificar que las elecciones que toma nos llevan a la salida correcta.

Definición 3.4 Formalmente, un autómata finito no determinista (AFN) N es una tupla $N = (Q, \Sigma, \delta, q_0, F)$ donde:

- Q - es un conjunto finito de estados;
- Σ - es el alfabeto de entrada;
- q_0 - es el estado inicial, con $q_0 \in Q$;
- F - es el conjunto de estados finales, $F \subseteq Q$;
- $\delta : Q \times \Sigma \to 2^Q$, es decir, $\delta(q, a)$ es un subconjunto de Q.

3.1. Función de transición extendida

Buscamos extender δ a $\widehat{\delta}$ de modo que $\widehat{\delta}(q, w)$ representa el conjunto de estados que son alcanzables desde el estado q al procesar la cadena w.

Definición 3.5 La función de transición extendida para los autómatas finitos no deterministas se define inductivamente como sigue. Sea $q \in Q$, $x \in \Sigma^*$ y $a \in \Sigma$, entonces

$$\widehat{\delta}(q, \varepsilon) = \{q\}, \tag{3.1.1}$$

$$\widehat{\delta}(q, xa) = \bigcup_{p_i \in \widehat{\delta}(q,x)} \delta(p_i, a). \tag{3.1.2}$$

Proposición 3.6 Para todo $a \in \Sigma$ y para todo $q \in Q$, la función de transición extendida para los autómatas finitos no deterministas cumple que:

$$\widehat{\delta}(q, a) = \delta(q, a).$$

Vimos que esto sucede también para los autómatas finitos deterministas. Así, denotaremos a $\widehat{\delta}$ como δ sin haber ambigüedad.

Ejemplo 3.7 Usando la función de transición extendida, encontremos los estados a los que el autómata N_{01} llega al procesar la cadena 010 iniciando en el estado q_0.

$$\begin{aligned}
\widehat{\delta}(q_0, \varepsilon) &= \{q_0\} \\
\widehat{\delta}(q_0, 0) &= \bigcup_{p_i \in \widehat{\delta}(q_0,\varepsilon)} \delta(p_i, 0) = \delta(q_0, 0) = \{q_0, q_1\} \\
\widehat{\delta}(q_0, 01) &= \bigcup_{p_i \in \widehat{\delta}(q_0,0)} \delta(p_i, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) \\
&\qquad = \{q_0\} \cup \{q_2\} = \{q_0, q_2\} \\
\widehat{\delta}(q_0, 010) &= \bigcup_{p_i \in \widehat{\delta}(q_0,01)} \delta(p_i, 0) = \delta(q_0, 0) \cup \delta(q_2, 0) \\
&\qquad = \{q_0, q_1\} \cup \varnothing = \{q_0, q_1\}
\end{aligned}$$

Al terminar de procesar la cadena 010, el autómata N_{01} llega a los estados q_0 y q_1. Como ninguno de estos estados es final, se rechaza dicha cadena.

Definición 3.8 El lenguaje de un autómata finito no determinista. Sean $N = (Q, \Sigma, \delta, q_0, F)$ un autómata finito no determinista y $L(N)$ el conjunto de todas las cadenas aceptadas por N. Tenemos que

$$L(N) = \{w \in \Sigma^* \mid \widehat{\delta}(q_0, w) \cap F \neq \varnothing\}.$$

Ejercicios 3.9

- Encuentre un AFN para el lenguaje L_{123} de cadenas $w \in \{1, 2, 3\}^*$ tal que el último símbolo en w figura previamente, sin que en medio de estas dos presencias haya un símbolo más grande. Por ejemplo: las cadenas 21311, 2312112, 12123123 pertenecen a L_{123}.

- Sea $L \subseteq \{0,1\}^*$ el lenguaje de todas las cadenas binarias tales que hay dos 0's separados por un número de posiciones que es un múltiplo de 5 mayor a cero. Por ejemplo, $1001110 \notin L$ y $10110110 \in L$. Construya un AFN para L.
- Diseñe un AFN para el lenguaje $A = \{w \in \{0,1\}^* \mid w \text{ contiene un 1 en la antepenúltima posición}\}$.
- Diseñe un AFN para el lenguaje de las cadenas binarias cuyo cuarto símbolo es diferente del antepenúltimo símbolo.
- Dé un AFN para los siguientes lenguajes, donde el alfabeto es $\{0,1\}$:
  i. Lenguaje 0^* con un estado.
  ii. Lenguaje $\{0\}$ con dos estados.
  iii. Lenguaje $\{\varepsilon\}$ con un estado.

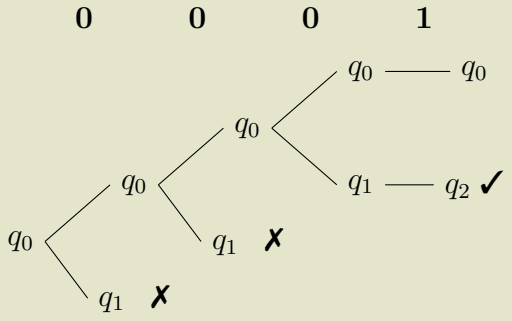
4. Comparando la expresividad de los AFDs y los AFNs

La expresividad o el poder de un autómata se mide en términos de la habilidad para aceptar lenguajes. Observemos que un AFD es un caso especial de un AFN con $|\delta(q,a)| = 1$, de modo que todo lo que un AFD puede hacer, un AFN también lo puede hacer. También veremos que para cada AFN N existe un AFD M el cual acepta el mismo lenguaje. Por lo tanto, el poder de un AFN es el mismo que el de un AFD. ¿Para qué estudiamos AFNs, los cuales no podemos implementar en la vida real? El convertir un AFN N con k estados a un AFD M requiere en algunas instancias un crecimiento exponencial (2^k) en el número de estados.

Es importante mencionar que los AFNs son más sencillos de construir, especificar y comprender, debido a que son concisos. Mientras que los AFDs pueden ser implementados en la vida real. Por lo tanto, usamos AFNs para capturar patrones en procesadores de texto – pero optamos por los AFDs cuando deseamos encontrar dichos patrones.

Teorema 4.1 *Para cada AFN N, existe un AFD M con $L(N) = L(M)$.*

La idea es que dado N, M simulará el árbol de ejecución completo de N al procesar una cadena. El truco es que un estado en M corresponderá a un subconjunto de los estados de N, es por esto que esta técnica se conoce como *construcción por subconjuntos*. Para ilustrar esta idea, recordemos el AFN N_{01} y el árbol de trayectorias de ejecución del Ejemplo 3.3 obtenido al procesar la cadena de entrada 0001:



El AFD M_{01} construido a partir de N_{01} debe seguir la trayectoria de ejecución para la cadena 0001 que se muestra a continuación:

$$\{q_0\} \overset{\mathbf{0}}{\longrightarrow} \{q_0, q_1\} \overset{\mathbf{0}}{\longrightarrow} \{q_0, q_1\} \overset{\mathbf{0}}{\longrightarrow} \{q_0, q_1\} \overset{\mathbf{1}}{\longrightarrow} \{q_0, q_2\}$$

Formalmente, dado un AFN $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ construimos un AFD $M = (Q_M, \Sigma, \delta_M, \{q_0\}, F_M)$ tal que

- $Q_M = 2^{Q_N}$, es decir, el conjunto potencia de Q_N,
- $\delta_M : 2^{Q_N} \times \Sigma \to 2^{Q_N}$, donde

$$\delta_M(S, a) = \bigcup_{p_i \in S} \delta_N(p_i, a). \qquad (\spadesuit \text{ - } 1)$$

  Así, $\delta_M(S, a)$ es el conjunto de estados de N alcanzables desde estados $p_i \in S$ procesando el símbolo de entrada a.

- Finalmente, el conjunto de estados finales de M se define como

$$F_M = \{S \subseteq Q_N \mid S \cap F_N \neq \varnothing\}. \qquad (\spadesuit \text{ - } 2)$$

Para demostrar que el autómata M que acabamos de construir cumple que $L(N) = L(M)$ necesitamos del siguiente lema.

Lema 4.2 *Para todo $q \in Q_N$ y para toda cadena $w \in \Sigma^*$ se cumple que*

$$\widehat{\delta_M}(\{q\}, w) = \widehat{\delta_N}(q, w).$$

Demostración. Por inducción estructural sobre w.

Caso base. Se cumple para $w = \varepsilon$, ya que $\widehat{\delta_M}(\{q\}, \varepsilon) \overset{\text{Def. 2.1.1}}{=} \{q\} \overset{\text{Def. 3.1.1}}{=} \widehat{\delta_N}(q, \varepsilon)$.

Hipótesis de inducción. Suponemos que se cumple para $w = x$, es decir, $\widehat{\delta_M}(\{q\}, x) = \widehat{\delta_N}(q, x)$.

Paso inductivo. Por demostrar que se cumple para $w = xa$. Hay que llegar a

$$\widehat{\delta_M}(\{q\}, xa) = \widehat{\delta_N}(q, xa).$$

Tenemos que,

$$\begin{aligned}
\widehat{\delta_M}(\{q\}, xa) &\overset{\text{Def. 2.1.2}}{=} \delta_M\big(\widehat{\delta_M}(\{q\}, x), a\big) \\
&\overset{\text{HI}}{=} \delta_M\big(\widehat{\delta_N}(q, x), a\big) \\
&\overset{\text{Def. } \spadesuit \text{ - } 1}{=} \bigcup_{p_i \in \widehat{\delta_N}(q,x)} \delta_N(p_i, a) \\
&\overset{\text{Def. 3.1.2}}{=} \widehat{\delta_N}(q, xa)
\end{aligned}$$

⊣

Teorema 4.3 *Dado un AFN N, siguiendo la construcción anterior, podemos producir un AFD M tal que $L(N) = L(M)$.*

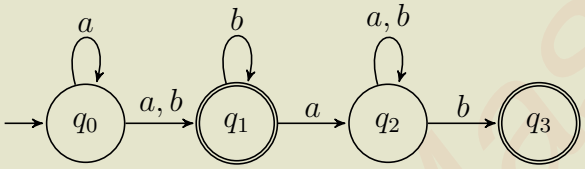
Demostración.

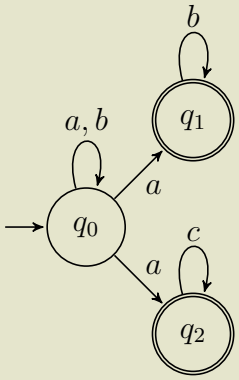
$$
\begin{aligned}
L(N) &\overset{\text{Def. 3.8}}{=} \{w \in \Sigma^* \mid \widehat{\delta_N}(q_0, w) \cap F_N \neq \varnothing\} \\
&\overset{\text{Lema 4.2}}{=} \{w \in \Sigma^* \mid \widehat{\delta_M}(\{q_0\}, w) \cap F_N \neq \varnothing\} \\
&\overset{\text{Def. ♠ - 2}}{=} \{w \in \Sigma^* \mid \widehat{\delta_M}(\{q_0\}, w) \in F_M\} \\
&\overset{\text{Def. 2.5}}{=} L(M)
\end{aligned}
$$

⊣

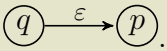
Ejercicios 4.4

- Transforme el AFN del Ejemplo 3.1 a un AFD usando la construcción por subconjuntos.
- Mediante la construcción por subconjunos transforme los siguientes AFNs a AFDs. ¿Qué lenguajes reconocen?



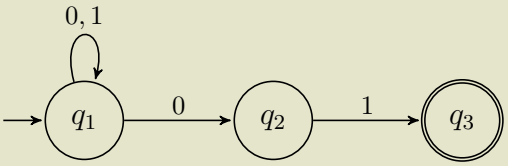


5. Autómatas finitos no deterministas con transiciones ε

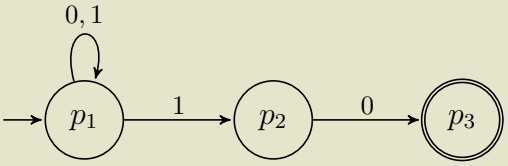
Hasta ahora hemos visto que añadir la característica de no determinismo no incrementa la expresividad ni la capacidad de reconocimiento de lenguajes por parte de los AFDs. Añadimos ahora otra característica llamada transición o movimiento ε, la cual se representa como: 

. Esto significa que el autómata se mueve sin consumir un símbolo de entrada. Intuitivamente, un autómata finito no determinista con transiciones ε (AFN-ε) es un AFN, excepto que ahora se permiten movimientos ε.

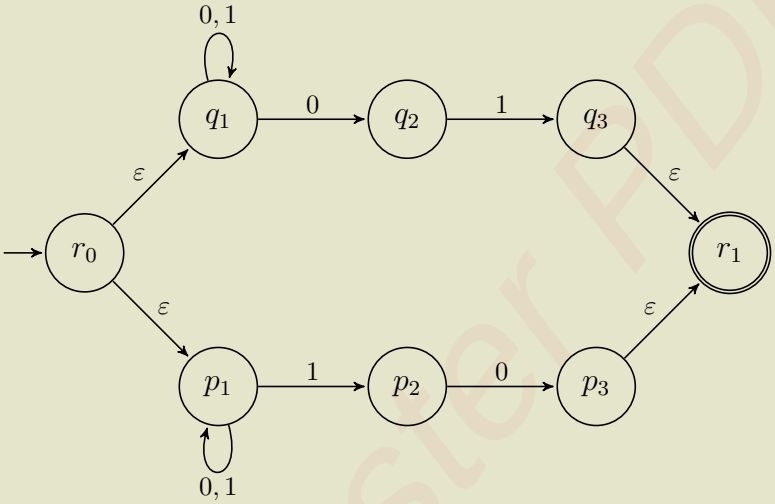
Ejemplo 5.1 Recordemos el AFN N_{01} que reconoce el lenguaje de cadenas binarias que terminan en 01,



Tengamos en cuenta también al AFN N_{10} que reconoce el lenguaje de cadenas binarias que terminan en 10,



Buscamos un AFN-ε N para el lenguaje $L = \{w \in \{0,1\} \mid w \text{ termina en } 01 \text{ ó } 10\}$. Simplemente combinamos los autómatas N_{01} y N_{10} anteriores con transiciones ε como sigue:



¿Para qué movimientos ε?

- son una herramienta descriptiva útil,
- son ideales para componer o combinar AFNs, y
- un AFN-ε puede ser convertido a un AFD e implementarse.

Definición 5.2 Formalmente, un autómata finito no determinista con transiciones ε, N, es una tupla $N = (Q, \Sigma, \delta, q_0, F)$ donde:

- Q - es un conjunto finito de estados;
- Σ - es el alfabeto de entrada;
- q_0 - es el estado inicial, con $q_0 \in Q$;
- F - es el conjunto de estados finales, $F \subseteq Q$;
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$, así podemos tener, por ejemplo, $\delta(q, \varepsilon) = \{q_2, q_3, q_4\}$.

Definición 5.3 Cerradura ε. Sea $q \in Q$. La cerradura ε de q, denotada ECLOSURE(q), es el conjunto de todos los estados alcanzables desde q usando cualquier secuencia de transiciones ε, esto incluye a la secuencia de longitud cero. ECLOSURE(q) se encuentra a través del procedimiento siguiente:

Encontrar ECLOSURE(q). Para este fin, usaremos la versión al tiempo t: $\text{ECLOSURE}_t(q)$.

Base. $\text{ECLOSURE}_0(q) := \{q\}$.

Inducción. $\text{ECLOSURE}_{t+1}(q) := \text{ECLOSURE}_t(q) \cup \{r \in \delta(p, \varepsilon) \mid p \in \text{ECLOSURE}_t(q) \text{ y } r \notin \text{ECLOSURE}_t(q)\}$.

Término. Cuando $\text{ECLOSURE}_{t+1}(q) = \text{ECLOSURE}_t(q)$. El procedimiento regresa $\text{ECLOSURE}_t(q)$ como la cerradura ε de q.

Sea $S \subseteq Q$. La cerradura ε del conjunto S se define como $\text{ECLOSURE}(S) = \bigcup_{r \in S} \text{ECLOSURE}(r)$.

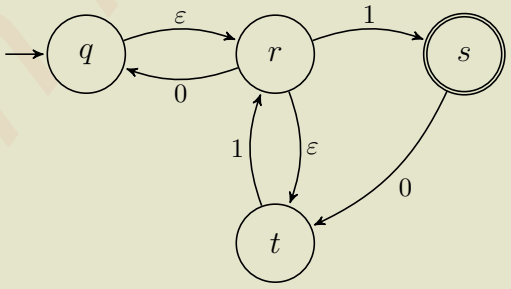
Vemos que para encontrar ECLOSURE(q) los pasos de **Inducción** y **Término** realizan una exploración exhaustiva de todos los estados que pueden alcanzarse mediante secuencias de transiciones ε con origen q. Así, para cualquier estado $p \in \text{ECLOSURE}(q)$, los estados alcanzables partiendo de p con secuencias de transiciones ε también estarán en ECLOSURE(q). Por consiguiente, tenemos el siguiente lema.

Lema 5.4 *Sean p y q estados, con $p \in \text{ECLOSURE}(q)$, entonces* $\text{ECLOSURE}(p) \subseteq \text{ECLOSURE}(q)$.

El lema anterior tiene la siguiente extensión.

Lema 5.5 *Sean p un estado y S un conjunto de estados, con $p \in \text{ECLOSURE}(S)$, se tiene entonces que* $\text{ECLOSURE}(p) \subseteq \text{ECLOSURE}(S)$.

Ejemplo 5.6 Consideremos el siguiente AFN-ε N:



Observemos que

- ECLOSURE(q) = $\{q, r, t\}$
- ECLOSURE(r) = $\{r, t\}$
- ECLOSURE(s) = $\{s\}$
- ECLOSURE(t) = $\{t\}$

5.1. Función de transición extendida

Deseamos extender la función de transición de los AFN-ε para que se puedan procesar cadenas. La definición formal es la siguiente:

Definición 5.7 La función de transición extendida para los autómatas finitos no deterministas con transiciones ε se define inductivamente como sigue. Sea $q \in Q$, $x \in \Sigma^*$ y $a \in \Sigma$, entonces

$$\begin{aligned} \widehat{\delta}(q, \varepsilon) &= \text{ECLOSURE}(q), \\ \widehat{\delta}(q, xa) &= \bigcup_{p_i \in \widehat{\delta}(q,x)} \text{ECLOSURE}(\delta(p_i, a)). \end{aligned}$$

Así, $\widehat{\delta}(q, w)$ es el conjunto de todos los estados alcanzables partiendo del estado q siguiendo transiciones cuyas etiquetas produzcan x, ignorando las etiquetas ε presentes en la trayectoria. Notemos dos cosas (i) $q \in \widehat{\delta}(q, \varepsilon)$ y (ii) $\delta(q, a) \neq \widehat{\delta}(q, a)$ con $a \in \Sigma$, a diferencia de los AFDs y AFNs. De hecho

$$\widehat{\delta}(q, a) = \bigcup_{p_i \in \text{ECLOSURE}(q)} \text{ECLOSURE}(\delta(p_i, a)). \tag{5.1.1}$$

Ejemplo 5.8 Utilicemos la función de transición extendida recién definida para el AFN-ε del Ejemplo 5.6. Encontremos:

- $\widehat{\delta}(q, \varepsilon) = \{q, r, t\}$,
- $\delta(q, \varepsilon) = \{r\}$,
- $\widehat{\delta}(q, 1) = \{r, s, t\}$,
- $\widehat{\delta}(q, 10) = \{q, r, t\}$,

Lema 5.9 *Si $p \in \widehat{\delta}(q, w)$, entonces* $\text{ECLOSURE}(p) \subseteq \widehat{\delta}(q, w)$.

Demostración. Analizaremos los posibles casos para la cadena w.

- $w = \varepsilon$. Así, $\widehat{\delta}(q, \varepsilon) = \text{ECLOSURE}(q)$. Nuestro supuesto es que $p \in \widehat{\delta}(q, \varepsilon)$, de manera que tenemos $p \in \text{ECLOSURE}(q)$. Por el Lema 5.4, llegamos a $\text{ECLOSURE}(p) \subseteq \text{ECLOSURE}(q)$. Luego,
$$\text{ECLOSURE}(p) \subseteq \widehat{\delta}(q, \varepsilon).$$
- $w = xa$. Tenemos,

$$\begin{aligned} p \in \widehat{\delta}(q, xa) &\Rightarrow p \in \bigcup_{p_i \in \widehat{\delta}(q,x)} \text{ECLOSURE}(\delta(p_i, a)) \\ &\Rightarrow p \in \text{ECLOSURE}(\delta(p_k, a)), \text{ para algún } p_k \in \widehat{\delta}(q, x) \\ &\Rightarrow \text{ECLOSURE}(p) \subseteq \text{ECLOSURE}(\delta(p_k, a)), \text{ por el Lema 5.5} \\ &\Rightarrow \text{ECLOSURE}(p) \subseteq \widehat{\delta}(q, xa), \text{ pues } \text{ECLOSURE}(\delta(p_k, a)) \subseteq \widehat{\delta}(q, xa). \dashv \end{aligned}$$

Teorema 5.10 *Para todo AFN-ε N, existe un AFN M con $L(N) = L(M)$.*

Demostración. Sea $N = (Q, \Sigma, \delta_N, q_0, F)$ una AFN-ε. Vamos a construir un AFN $M = (Q, \Sigma, \delta_M, q_0, F)$ donde agregaremos nuevas transiciones para capturar el efecto de los movimientos ε presentes en N.

Formalmente, en M todo es igual que en N, excepto la función de transición. Sea $q \in Q$ y $a \in \Sigma$, definimos δ_M como

$$\delta_M(q, a) = \widehat{\delta}_N(q, a) = \bigcup_{p_i \in \text{ECLOSURE}(q)} \text{ECLOSURE}(\delta_N(p_i, a)). \tag{5.1.2}$$

Así $\delta_M(q, \varepsilon)$ queda indefinida. Es importante señalar que si $\varepsilon \in L(N)$ o $\text{ECLOSURE}(q_0) \cap F \neq \varnothing$ en N, entonces agregamos q_0 a F en la construcción de M.

Se sigue $L(N) = L(M)$ del siguiente lema.

Lema 5.11 *Para todo $q \in Q$ y $w \in \Sigma^* - \{\varepsilon\}$ se cumple*

$$\widehat{\delta}_N(q, w) = \widehat{\delta}_M(q, w).$$

Demostración. Por inducción sobre la cadena w.

Base ($w = a$) Tenemos que $\widehat{\delta}_N(q, a) = \delta_M(q, a) = \widehat{\delta}_M(q, a)$ por 5.1.2 y porque para los AFN se tiene $\widehat{\delta}_M(q, a) = \delta_M(q, a)$. El caso $w = \varepsilon$ no se toma en cuenta porque estamos tomando $w \in \Sigma^* - \{\varepsilon\}$.

Hipótesis de inducción ($w = x$) Suponemos $\widehat{\delta}_N(q, x) = \widehat{\delta}_M(q, x)$.

Paso inductivo ($w = xa$) Por la Definición 5.7 tenemos que

$$\widehat{\delta}_N(q, w) = \widehat{\delta}_N(q, xa) = \bigcup_{p_i \in \widehat{\delta_N}(q,x)} \text{ECLOSURE}(\delta_N(p_i, a)).$$

Observemos que $\text{ECLOSURE}(p_i) \subseteq \widehat{\delta}_N(q, x)$ por el Lema 5.9. Así, los estados r_j que son miembros de $\text{ECLOSURE}(p_i)$ lo son también de $\widehat{\delta}_N(q, x)$. Luego,

$$\bigcup_{p_i \in \widehat{\delta_N}(q,x)} \text{ECLOSURE}(\delta_N(p_i, a)) = \bigcup_{p_i \in \widehat{\delta_N}(q,x)} \bigcup_{r_j \in \text{ECLOSURE}(p_i)} \text{ECLOSURE}(\delta_N(r_j, a)).$$

ya que del lado derecho, las r_js simplemente están generando ECLOSUREs repetidas que ya están contempladas en los p_is. Como el resultado final es un conjunto, no se pueden tener repeticiones; por consiguiente, la igualdad se satisface.

Con las observaciones previas estamos en condiciones de dar un razonamiento ecuacional que nos permita llegar al resultado deseado. Tenemos entonces que:

$$
\begin{aligned}
\widehat{\delta}_N(q,w) = \widehat{\delta}_N(q,xa) &= \bigcup_{p_i \in \widehat{\delta_N}(q,x)} \text{ECLOSURE}(\delta_N(p_i,a)) \\
&= \bigcup_{p_i \in \widehat{\delta_N}(q,x)} \bigcup_{r_j \in \text{ECLOSURE}(p_i)} \text{ECLOSURE}(\delta_N(r_j,a)) \\
&= \bigcup_{p_i \in \widehat{\delta_N}(q,x)} \widehat{\delta_N}(p_i,a) \ \text{(ver Ecuación 5.1.1)} \\
&= \bigcup_{p_i \in \widehat{\delta_N}(q,x)} \widehat{\delta_N}(p_i,a) \ \text{(Hipótesis de inducción)} \\
&= \bigcup_{p_i \in \widehat{\delta_M}(q,x)} \delta_M(p_i,a) \ \text{(ver Ecuación 5.1.2)} \\
&= \widehat{\delta_M}(q,xa) \ \text{(ver Definición 3.5)} \\
&= \widehat{\delta_M}(q,w)
\end{aligned}
$$

⊣

Es menester mencionar que la construcción anterior de un AFN-ε a un AFN, junto con la construcción por subconjuntos de un AFN a un AFD, nos permite convertir cualquier AFN-ε directamente a un AFD.

Ejercicios 5.12

- Diseñe un AFN-ε para el lenguaje $L = \{w \in \{0,1\}^* \mid w$ contiene un número par de 0s, o contiene exactamente dos 1s$\}$ con seis estados.
- Mediante el procedimiento descrito en la demostración del Teorema 5.10 encuentre un AFN equivalente al AFN-ε del Ejemplo 5.6.
- Quite las transiciones ε de los siguientes tres AFN-ε para obtener un AFN equivalente.