2 Autómatas Finitos

2.1 AUTÓMATAS FINITOS DETERMINISTAS (AFD).

2.1.1 DEFINICIÓN DE AFD.

Un autómata finito determinista es una quíntupla que denotaremos de manera genérica por M=(Q,Σ,q_0,δ,F) donde:

Q es un conjunto finito cuyos elementos llamaremos *estados*.
Σ es un alfabeto que llamamos *alfabeto de entrada*.
$q_0 \in Q$ es un estado señalado que llamamos *estado inicial*.
F es un subconjunto de Q no vacío, cuyos elementos llamamos *estados finales*.
δ es una aplicación de $Q\times\Sigma\rightarrow Q$, que llamamos *función de transición*.

La función de transición es la verdadera clave de la máquina. Obsérvese que es una aplicación, así cada pareja posible formada por un estado y un símbolo del alfabeto debe tener una imagen y sólo una, es decir $\delta(q,a)\in Q$, cualquiera que sean $q\in Q$ y $a\in\Sigma$.

Ejemplos:

Sea $M_1 = (Q, \Sigma, \delta, q_0, F)$ donde Q={p,q,r}, Σ={a,b}, Sea p el estado inicial, F={r} y δ definida como sigue:

δ(p,a)=q	δ(p,b)=r
δ(q,a)=p	δ(q,b)=q
δ(r,a)=r	δ(r,b)=r

Tabla 2.1. Función de transición de M_1.

Según nuestra definición M_1 es un AFD.

Para visualizarlo de alguna forma imaginemos una especie de circuito electrico con tantas bombillas como estados, las correspondientes a los estados finales de color verde, las demás amarillas. Sobre una cinta de entrada escribimos una palabra con símbolos del alfabeto de entrada. Al poner a funcionar la máquina se enciende la bombilla correspondiente al estado inicial. A partir de ese momento se procesa el símbolo actual en la cinta de entrada transitando al estado definido en cada momento por la función de transición hasta que la palabra de la entrada haya sido leido completa.

Si la palabra a procesar fuese *aabbab*, se enciende el estado *p* inicial y a continuación *qprrrr*. El estado que queda encendido es *r* que es final. Si la palabra a procesar fuese *abbb* la secuencia de estados sería *pqqqq*.

2.1.2 REPRESENTACIÓN DE UN AFD.

Tenemos dos maneras de representar un AFD

- Con una tabla:
  Se ponen tantas filas como estados, y tantas columnas como símbolos forman el alfabeto. Marcamos el estado inicial con una flecha de entrada y cada uno de los estados finales con un asterisco. En el cruce de la fila marcada con el estado q y la columna marcada con el símbolo a del alfabeto ponemos el estado δ(q,a).

- Con un diagrama:
  Cada estado no final se representa con un círculo; cada estado final se representa con un doble círculo; se señala el estado inicial con una flecha entrando, sin

etiqueta; por cada transición δ(q,a)=t se dibuja una flecha dirigida del estado de partida q al de llegada llegada t etiquetada a

Ejemplos:

La máquina M_1 del ejemplo se representa con una tabla

δ	a	b
→p	q	r
q	p	q
*r	r	r

Tabla 2.2. Autómata finito determinista M_1.

O bien se representa con un diagrama

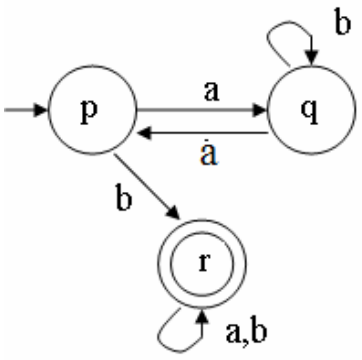


Fig. 2.1 Autómata finito determinista M_1.

Observemos que cada una de ambas representaciones contiene toda la información del autómata.

2.1.3 DIAGRAMAS INCOMPLETOS: ESTADO DE ABSORCIÓN.

Con frecuencia nos encontramos AFDs como el M_2 siguiente.

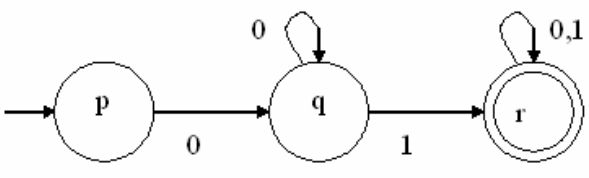


Fig. 2.2 autómata finito determinista M_2.

Si observamos con un poco de atención vemos que la transición δ(p,1) no está representada, lo que contradice la definición de AFD puesto que hemos afirmado que δ es una aplicación. Lo que ocurre en realidad es que la máquina no ha sido completamente dibujada, por comodidad y claridad. Debemos entender que todas las transiciones que falten en el diagrama de un AFD van a un único estado no final, que llamamos genéricamente *estado de absorción*. Así el diagrama completo de M_2 es:

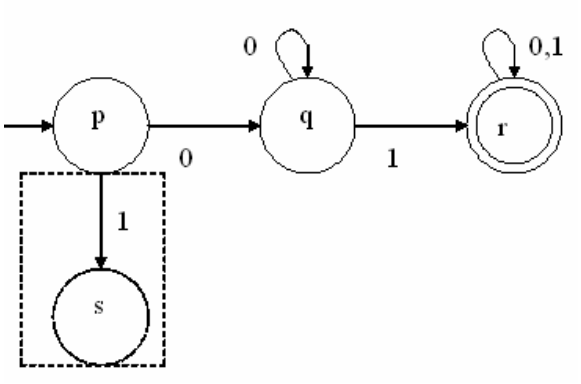


Fig.2.3. Autómata finito determinista M_2 con el estado de absorción.

El estado s es el estado de absorción y con frecuencia desaparece del gráfico del autómata, pero no debemos olvidar que está ahí aunque no lo dibujemos.

2.1.4 EXTENSIÓN DE LA FUNCIÓN DE TRANSICIÓN

Se trata de definir una función que describa qué estado se alcanza desde un estado q si a continuación en vez de entrar un sólo símbolo (en cuyo caso se alcanzaría el estado descrito por δ(q,a)), entrara una palabra ω∈Σ*.

Definimos $\hat{\delta} : Q \times \Sigma^* \to Q$ por recurrencia sobre la longitud de la palabra ω.

- Si |ω| = 0 entonces $\omega = \lambda$ y definimos $\forall q \in Q,\ \hat{\delta}(q,\lambda) = q$ (si no hay entrada no hay cambio de estado).
- Supongamos definido $\hat{\delta}(q,x)$ para cada $x \in \Sigma^*$, tal que $|x| \leq n$.
- Sea $\omega \in \Sigma^*$ tal que $|\omega|=n+1$, entonces ω se puede escribir $\omega = xa$ con $|x|=n$ y $a \in \Sigma$. Ahora definimos $\forall q \in Q,\ \hat{\delta}(q,\omega) = \hat{\delta}(q,xa) = \delta\left(\hat{\delta}(q,x),a\right)$.

Ejemplo:

En nuestra máquina M_1 sería

$$\hat{\delta}(q,ba) = \delta\left(\hat{\delta}(q,b),a\right) = \delta(q,a) = p$$
$$\hat{\delta}(q,bab) = \delta\left(\hat{\delta}(q,ba),b\right) = \delta(p,b) = r$$

En el diagrama se pueden seguir los arcos desde *q*, con los símbolos *bab* consecutivamente y se observa que terminamos en el estado *r*.

2.1.5 PALABRA ACEPTADA POR UN AFD.

Decimos que $\omega \in \Sigma^*$ es una palabra aceptada por el autómata finito determinista M si desde el estado inicial entrando consecutivamente de izquierda a derecha los símbolos de la palabra en cuestión, se alcanza un estado final. Expresado con símbolos si $\hat{\delta}(q_0,\omega) \in F$.

Ejemplos:

Las cadenas abb, baa, b, abbaaa son palabras aceptadas por M_1. Las cadenas 01, 0001, 001001 son palabras aceptadas por M_2.

2.1.6 LENGUAJE ACEPTADO POR UN AFD.

Es el conjunto de todas las palabras aceptadas, es decir:

$$L(M) = \left\{\omega \in \Sigma^* : \hat{\delta}(q_0,\omega) \in F\right\}$$

Ejemplos:

El lenguaje de M_2 son todas las cadenas de ceros y unos que empiezan por cero y tiene al menos un uno

$$L(M) = \left\{x1y : x, y \in \Sigma^*\right\}$$

2.1.7 EQUIVALENCIA DE AFDS.

Sean M y M′ dos AFDs, decimos que son equivalentes si L(M)=L(M`).

Según lo que hemos dicho hasta el momento los AFDs se caracterizan por las palabras que aceptan y es menos importante la estructura interna de la máquina, el número de

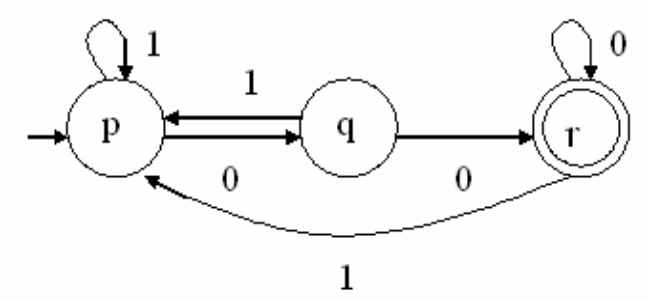
estados, cuántos estados finales tenga, etc… Digamos que un AFD es principalmente una especie de filtro de todas las cadenas posibles sobre el alfabeto de entrada. Los AFDs tienen muchas utilidades, en particular a nosotros nos interesa especialmente su utilidad como analizadores léxicos. Cuando, por ejemplo, en un programa necesitamos una expresión algebraica del tipo (a+b)*c/(d+e) debe estar correctamente escrita y de comprobarlo se encarga precisamente un analizador construido a partir de un autómata. Lo más normal es que tengamos un lenguaje de interés L sobre un alfabeto Σ y construyamos un AFD cuyo lenguaje sea precisamente L.

Ejemplo:

Construir un AFD sobre el alfabeto Σ={0,1} cuyo lenguaje sea

L={cadenas que terminan en 00}

Un AFD para este lenguaje es



Fig, 2.4. AFD cuyo lenguaje es L={cadenas que terminan en 00}.

Antes de terminar esta sección damos alguna definición más

2.1.8 ESTADOS INACCESIBLES.

Sea M=(Q,Σ,δ,q_0,F) un AFD. Un estado q∈Q decimos que es inaccesible, si no existe ninguna palabra sobre el alfabeto de entrada que partiendo desde q_0 llegue a q. Con símbolos será

$$\text{q inaccesible si } \forall \omega \in \Sigma^*,\ \hat{\delta}(q_0,\omega) \neq q$$

Los estados que no son inaccesibles decimos que son accesibles. Si eliminamos los estados inaccesibles y todas sus transiciones el AFD obtenido es equivalente al dado.

Ejemplo:

En el AFD M_3 de la figura, el estado r es inaccesible por que no se puede alcanzar a partir del estado inicial

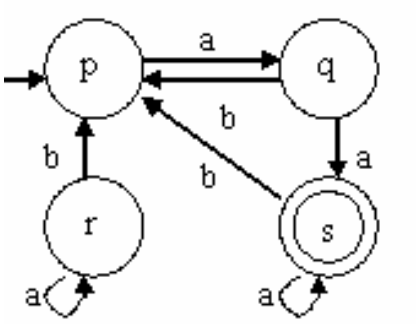


Fig. 2.5. M_3 con el estado r inaccesible.

2.1.9 AUTÓMATA CONEXO.

Un AFD decimos que es conexo si no tiene estados inaccesibles. Si un AFD no es conexo basta eliminar los estados inaccesibles y todas sus transiciones (las de entrada y las de salida) para obtener un nuevo AFD conexo equivalente al de partida.

Ejemplo:

El AFD M_4 obtenido quitando a M_3 el estado r y sus transiciones, es conexo

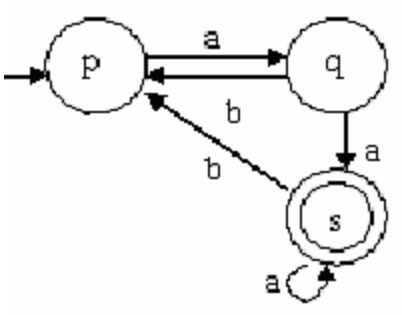


Fig. 2.6. M_4 que es conexo y equivalente a M_3.

2.1.10 CONSTRUCCIÓN DE UN ANALIZADOR LÉXICO A PARTIR DE UN AFD.

Daremos a continuación un método genérico de construcción de un trozo de programa que nos vale como analizador de las palabras del lenguaje de un AFD dado, M=(Q, Σ, q_0, δ, F).

Como datos de entrada se introducen el alfabeto de entrada de la máquina y una tabla M=(m_{ij}) de orden r×c siendo r=|Q| y c=|Σ| +1 donde la filas se rotulan con el nombre de los estados del autómata, las c primeras columnas se rotulan con los símbolos del alfabeto de entrada y la última con 'fdc' ('fin de cadena'); en la posición (i,j) se escribe $m_{ij} = \delta(q_i, a_j)$ para i=1,…,r y j=1,…,(c-1), es decir las (c-1) primeras columnas son exactamente la tabla del autómata, sin marcar el estado inicial ni los finales, y la última columna se rellena de la siguiente manera: si la fila corresponde a un estado final $q_i \in F$, entonces m_{ic}="aceptar", pero si $q_i \notin F$, entonces m_{ic}="e".

Un algoritmo genérico, debe tratar de emular el funcionamiento del AFD, para ello se inicializa una variable 'estado' con el estado inicial de la máquina concreta que tengamos y a partir de ahí se manda leer de la entrada, clasificar el símbolo leído, y en caso de que sea válido cambiar de estado según lo que diga la matriz introducida. Si cuando se ha acabado de leer la palabra completa, es decir cuando el símbolo actual en la cinta de entrada es la marca de 'fin de cadena', no se ha alcanzado un estado final, la palabra leída no es del lenguaje del AFD y en la tabla se lee 'e' (recuérdese como se construyó la columna de la matriz de entrada rotulada con 'fdc') y debe dar un mensaje de error; si la palabra ha logrado llegar a un estado final el estado que se alcance será 'Aceptar'. Este modelo es válido para cualquier AFD. Pasamos estas ideas a un algoritmo genérico:

```
ESTADO:=q0
REPETIR
   LEER siguiente símbolo
    CASO símbolo DE
        SI símbolo∈Σ, ENTRADA:=símbolo
        SI símbolo es la marca de fin de cadena, ENTRADA:="fdc"
        SI símbolo no es ninguno de los anteriores, salir a la rutina de ERROR
    FIN CASO
   ESTADO:= Tabla(ESTADO, ENTRADA)
   SI ESTADO:="e" ENTONCES salir a la rutina de ERROR
```

```
        FIN SI
    HASTA ESTADO:="aceptar"
```

Ejemplo:

La matriz correspondiente a nuestra máquina M_1 es:

	a	b	fdc
p	q	r	e
q	p	q	e
r	r	r	aceptar

Tabla 2.2. Tabla del analizador léxico del autómata M_1.

en la rutina anterior basta sustituir la inicialización de la variable ESTADO en la primera línea del algoritmo por ESTADO:=p, puesto que en nuestro autómata el estado inicial es p.

Si nuestro AFD está incompleto en la matriz de entrada del analizador podemos suprimir la fila correspondiente al estado de absorción y en las casillas correspondientes a transiciones que lleguen a dicho estado escribimos una "e".

Ejemplo:

La matriz asociada a M_3 es

	0	1	fdc
p	q	s	e
q	r	q	e
r	s	s	aceptar
s	s	s	e

Tabla 2.2. Tabla del analizador léxico del autómata M_3 completo.

o bien

	0	1	fdc
p	q	e	e
q	r	q	e
r	e	e	aceptar

Tabla 2.3. Tabla del analizador léxico del autómata M_3 sin el estado de absorción.

2.1.11 MINIMIZACIÓN DE UN AFD.

Dado un AFD se trata de construir otro equivalente pero que sea mínimo en cuanto al número de estados.

Sea $(Q, \Sigma, q_0, \delta, F)$ un AFD. Definimos una relación de equivalencia en Q: Dos estados $p,q \in Q$ son equivalentes,

$$pEq, \text{ si } \forall \omega \in \Sigma^*, \text{ se tiene } (\delta(p,\omega) \in F \Leftrightarrow \delta(q,\omega) \in F)$$

El conjunto de estados del AFD mínimo es el conjunto cociente de esta relación de equivalencia de estados Q/E.

Para el cálculo de este conjunto cociente definimos:

Dos estados $p,q \in Q$ son equivalentes de orden r ($r \geq 0$),

$$pE_rq, \text{ si } \forall \omega \in \Sigma^* \text{ tal que } |\omega| \leq r \text{ se tiene } (\delta(p,\omega) \in F \Leftrightarrow \delta(q,\omega) \in F)$$

Para cada r la relación E_r es de equivalencia. De las definiciones anteriores se tiene

$$pEq \Leftrightarrow pE_rq, \ \forall r \geq 0.$$

Para obtener Q/E se va calculando Q/E_0, Q/E_1,... hasta que $Q/E_r = Q/E_{r+1} = Q/E$.

Lo hacemos por recurrencia sobre r.

- Para r=0, como $\delta(p,\lambda)=p$, dos estados p y q pueden ser equivalentes de orden 0 sólo si ambos son estados finales ($p,q \in F$) o si ambos son estados no finales ($p,q \notin F$) así el conjunto cociente de la relación de equivalencia E_0 será

  $$Q/E_0 = \{F, \overline{F}\}.$$

- Supongamos que hemos calculado el conjunto cociente para la relación de equivalencia de estados de orden r,

  $$Q/E_r = \{C_1, C_2, \ldots, C_i, \ldots, C_k\}.$$

- Queremos construir Q/E_{r+1}.

  Para cada $Ci \in Q/E_r$ ocurre uno de los dos casos **excluyentes** siguientes:

  i) Para todo $a \in \Sigma$, existe un *j* tal que $\delta(C_i, a) \subseteq C_j$, entonces incluimos C_i en Q/E_{r+1}.

  ii) Existe $a \in \Sigma$ tal que para todo $j=1,\ldots,k$, $\delta(C_i,a) \not\subset C_j$, entonces hacemos $C_i = C_{i1} \cup C_{i2}$ de manera que para cada uno de los dos subconjuntos creados

existan j y k tales que $\delta(C_{i1},a) \subseteq C_j$ y $\delta(C_{i2}, a) \subseteq C_k$ e incluimos C_{i1} y C_{i2} en Q/E_{r+1} eliminando C_i.

Ejemplo:

Sea el AFD M_4 dado por la tabla

	a	b
→p	r	q
q	r	q
*r	s	t
*s	r	t
t	t	q
u	u	p

Tabla 2.4. Tabla del autómata M_4.

Vamos a minimizarlo. Primero observamos que no es conexo y que 'u' es un estado inaccesible. Eliminamos dicho estado y sus transiciones y nos queda

	a	b
→p	r	q
q	r	q
*r	s	t
*s	r	t
t	t	q

Tabla 2.5. Tabla del autómata M_4 sin el estdo inaccesible *u*.

Cuidamos de que estén todas las transiciones (el AFD debe estar completo antes de minimizarlo).

- Consideramos ahora el cociente

$$Q/E_0=\{F, \overline{F}\}=\{[r,s],[p,q,t]\}=\{C_{01}, C_{02}\}.$$

- Veamos si los estados r y s son equivalentes de orden 1. Consideramos

$$\delta(C_{01}, a) = \delta([r,s],a) = \{r,s\} \subseteq C_{01}$$

$$\delta(C_{01,}b) = \delta([r,s],b) = \{t\} \subseteq C_{02}$$

vemos que se ajusta al punto i) anterior, luego esa clase entera pasa a ser una clase también en Q/E_1.

- Veamos ahora lo que ocurre con la clase C_{02} de Q/E_0. Consideramos

$$\delta(C_{02}, a) = \delta([p,q,t],a) = \{r,t\}$$
$$\delta(C_{02}, b) = \delta([p,q,t],b) = \{q\} \subseteq C_2$$

como $\delta(C_{02},a) \not\subset C_{01}$ y $\delta(C_2,a) \not\subset C_{02}$, estos tres estados no son equivalentes de orden 1. Estamos en el caso descrito en el punto ii) por lo que tenemos que dividir esa clase en dos subclases que sí verifiquen la propiedad mencionada. La división correcta es

$$C_{02} = C_{021} \cup C_{022} \text{ con } C_{021}=\{p,q\} \text{ y } C_{022}=\{t\}$$

de manera que $\delta(C_{021}, a)=\{r\} \subseteq C_1$ y $\delta(C_{022}, a)=\{t\} \subseteq C_2$. Tenemos ahora el conjunto cociente $Q/E_1 = \{C_{01}, C_{021}, C_{022}\}$ que renombramos $C_{11} = \{r,s\}, C_{12} = \{p,q\}$ y $C_{13} = \{t\}$ (con el primer subíndice indicamos siempre a que cociente pertenece, en este caso a Q/E_1).

- Estudiamos ahora el conjunto Q/E_2. La clase C_{11} pasa tal cual (comprobarlo), la clase C_{13} también porque es una clase unitaria y no se puede subdividir más. En cuanto a la clase $C_{12} = \{p,q\}$ será

$$\delta(C_{12},a) = \{r\} \subseteq C_1$$
$$\delta(C_{12},b) = \{q\} \subseteq C_2$$

luego el conjunto cociente es $Q/E_1=Q/E_2$ y hemos terminado.

Este proceso se suele escribir ordenadamente en una tabla cuyas filas son clases y cuyas columnas van rotuladas con los símbolos del alfabeto de entrada. En el cruce de fila y columna se pone el conjunto imagen de la clase completa con el símbolo correspondiente. Si está contenido en alguna de las clases actuales se mantiene y si no está contenido en ninguna se subdivide y se vuelve a empezar. En nuestro caso quedaría algo así

	a	b
[r,s]	$\{r,s\}\subseteq C_1$	$q\subseteq C_2$
[p,q,t]	$\{r,t\}\not\subset C_1,C_2$	
[p,q]	$\{r\}\subseteq C_1$	$\{q\}\subseteq C_{21}$
[t]	$\{t\}\subseteq C_{22}$	$\{q\}\subseteq C_{21}$

Tabla 2.6. Tabla para la minimización de M_4.

El AFDmín equivalente tiene sólo tres estados y su tabla es

	a	b
$*C_1$	C_1	C_3
$\rightarrow C_2$	C_1	C_2
C_3	C_3	C_2

O bien

	a	b
$\rightarrow$[p,q	[r,s]	[p,q]
*[r,s]	[r,s]	[t]
[t]	[t]	[p,q]

Tabla 2.7. AFD mínimo equivalente a M_4.

Gráficamente la maquina M_4 y su equivalente mínimo son

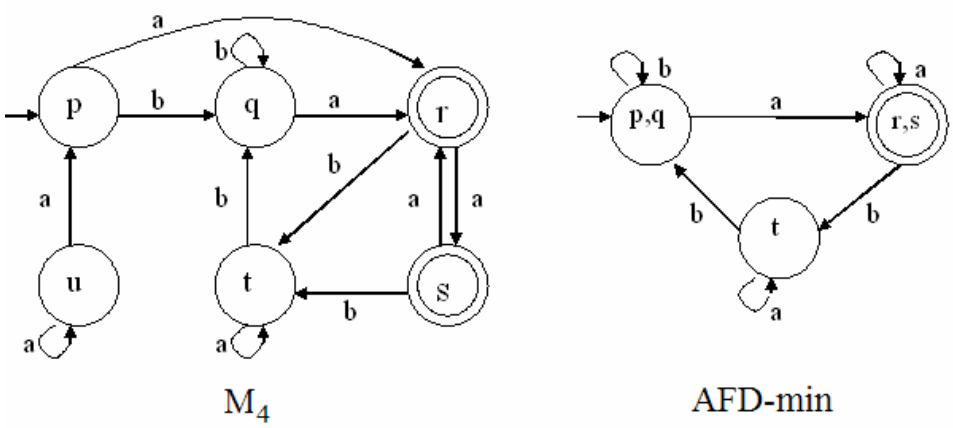


Fig 2.7. Diagrama de M_4 y de su equivalente mínimo.

2.2 AUTÓMATA FINITO NO DETERMINISTA (AFND).

2.2.1 DEFINICIÓN DE AFND.

El tipo de autómata que vamos a definir a continuación es esencialmente muy parecido al que definimos como AFD. La diferencia fundamental es que ahora permitiremos que desde un estado, con un símbolo del alfabeto se alcance no un único estado, sino un

subconjunto de estados (incluido el vacío). Formalmente un autómata finito no determinista M es una quíntupla $(Q, \Sigma, q_0, \delta, F)$ donde

Q es un conjunto finito a cuyos elementos llamamos *estados*
Σ es un alfabeto que llamamos *alfabeto de entrada*
q_0 es un estado señalado que llamamos *estado inicial*
F es un subconjunto de Q, no vacío, cuyos elementos llamamos *estados finales*
δ es una aplicación de $Q\times\Sigma\rightarrow P(Q)$, que llamamos *función de transición*

Obsérvese que ahora, desde un estado, con un símbolo de entrada se alcanza un conjunto de estados. Con símbolos $\delta(q,a) = A \in Q$.

Ejemplo:

Sea $M_5= (Q,\Sigma,q_0,\delta,F)$ siendo $Q=\{p,q,r,s\}$, $\Sigma=\{0,1,2\}$, sea p el estado inicial, $F=\{q,s\}$ y la función de transición δ dada por

$$\delta(p,0)=\{q,r\} \quad \delta(p,1)=\varnothing$$
$$\delta(q,0)=\{q\} \quad \delta(q,1)=\{q,s\}$$
$$\delta(r,0)=\varnothing \quad \delta(r,1)=\{s\}$$
$$\delta(s,0)=\{p\} \quad \delta(s,1)=\varnothing$$

Según nuestra definición M_5 es un AFND.

2.2.2 REPRESENTACIÓN DE UN AFND.

Tenemos dos maneras de representar un AFND

- Con una tabla: se ponen tantas filas como estados, y tantas columnas como símbolos forman el alfabeto. Marcamos el estado inicial con una flecha de entrada y cada uno de los estados finales con un asterisco. En cruce de la fila marcada con el estado q y la columna marcada con el símbolo a del alfabeto ponemos el conjunto de estados dado por δ(q,a).

- Con un diagrama: Cada estado no final se representa con un circulo; cada estado final se representa con un doble circulo; se señala el estado inicial con una flecha

entrando, sin etiqueta; si $\delta(q,a)=A$, por cada estado $r \in A$ se dibuja una flecha dirigida del estado de partida q al de llegada r etiquetada a.

Ejemplo:

La máquina M_5 del ejemplo se representa con una tabla

δ	0	1
→p	q,r	∅
*q	q	q,s
r	∅	s
*s	p	∅

Tabla 2.8. Tabla de transiciones de M_5.

Y con un diagrama

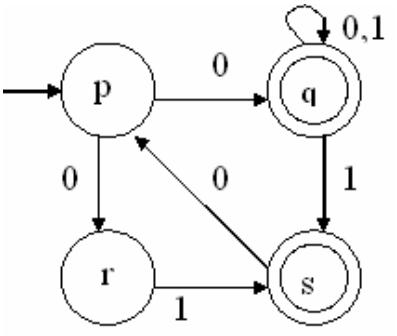


Fig. 2.8. Diagrama de M_5.

Observemos que este AFND presenta dos puntos de no determinismo: uno en 'p' con 0 que pasa a 'q' y a 'r', y otro en 'q' con uno que pasa a 'q' y a 's'. Observemos también que tanto la tabla como el diagrama contienen toda la información del autómata.

Podemos visualizarlo casi como un AFD: bombillas que se verán encendidas cuando el estado en cuestión esté activo, las bombillas de estado final de un color diferente-digamos, verde- y una palabra que será procesada símbolo a símbolo en la cinta de entrada. Al comenzar se enciende sólo el estado inicial; a partir de ese momento y a diferencia de lo que sucede en los AFDs, en cada momento podría verse encedido un solo estado, varios o incluso apagarse todos, precisamente debido a que las transiciones

ahora son conjuntos de estados. (En los AFDs en todo momento se encuentra exactamente un estado activo).

2.2.3 EXTENSIÓN DE LA FUNCIÓN DE TRANSICIÓN.

Se trata, como ya hicimos con los AFD, de definir una función que describa qué estados se alcanzan desde un estado *q* si a continuación en vez de entrar un sólo símbolo (en cuyo caso se alcanzarían los estados descritos por δ(q,a)), entrara una palabra ω∈Σ*. Antes de continuar convenimos que para A⊆Q escribiremos

$$\delta(A,a) = \underset{r\in A}{\cup} \delta(r,a)$$

Definimos $Q \times \Sigma^* \xrightarrow{\hat{\delta}} P(Q)$, por recurrencia sobre la longitud de la palabra ω.

- Si |ω|=0 entonces ω=λ y definimos $\hat{\delta}(q,\lambda) = q$ para cada estado (si no hay entrada no hay cambio de estado).
- Supongamos definido $\hat{\delta}(q,x)$ para cada x∈Σ* tal que |x|≤n.
- Sea ω∈Σ* tal que |ω|=n+1, entonces ω se puede escribir ω=xa con |x|=n y a∈∑. Ahora definimos ∀q∈Q ,

$$\hat{\delta}(q,xa) = \delta\left(\hat{\delta}(q,x),a\right) = \underset{r\in\hat{\delta}(q,x)}{\cup} \delta(r,a)$$

Para las palabras de longitud 1 se tiene

$$\hat{\delta}(q,a) = \hat{\delta}(q,\lambda a) = \delta\left(\hat{\delta}(q,\lambda),a\right) = \delta(q,a)$$

es decir la función de extensión a palabras y la función de transición original coinciden para palabras de longitud 1.

2.2.4 PALABRA ACEPTADA POR UN AFND.

Sea ω∈Σ* decimos que ω es una palabra aceptada por el AFND si partiendo del estado inicial logra alcanzar *alguno* de los estados finales, es decir si

$$\hat{\delta}(q_0,\omega) \cap F \neq \varnothing$$

Ejemplo:

Las palabras 0, 01, 010101 son aceptadas por M_5.

2.2.5 LENGUAJE ACEPTADO POR UN AFND.

Es el conjunto de todas las palabras aceptadas es decir

$$L(M)=\{\omega\in\Sigma^* : \delta(q_0,\omega) \cap F \neq \varnothing\}$$

Los autómatas finitos deterministas son un caso particular de los no deterministas en que la imagen de cada par $\delta(q,a)$ es un conjunto unitario.

2.2.6 CONSTRUCCIÓN DE UN AFD EQUIVALENTE A UN AFND DADO.

Sea M = (Q, Σ, q_0, δ, F) un AFND que supondremos conocido. Queremos construir un AFD, M´, tal que L(M)=L(M′). Antes de continuar debemos recordar que los autómatas se caracterizan por su lenguaje asociado y que la técnica que vamos a presentar garantiza la existencia de un AFD equivalente, pero en ningún caso se da la unicidad, es decir hay muchos AFDs equivalentes a M. Nosotros construiremos uno de ellos.

La idea para la construcción de la nueva máquina es que al ser Q un conjunto finito, el número de sus subconjuntos también lo es. Si tenemos que para cierto estado q∈Q y para cierto símbolo del alfabeto a∈Σ, δ(q,a) = A∈P(Q), a fin de cuentas ese subconjunto podría ser representado por un único estado en la nueva máquina que queremos construir. De modo que como conjunto de estados para M′ tomamos Q′ = P(Q), como alfabeto de entrada tomamos por supuesto el mismo Σ, como estado inicial tomamos $\{q_0\}$, como conjunto de estados finales tomamos todos los subconjuntos de Q que contengan algún estado final, formalmente

$$F' = \{B\in P(Q) : B\cap F \neq \varnothing\}$$

y definimos la nueva función de transición $\delta':Q'\times\Sigma\rightarrow Q'$ como sigue:

$$\delta'(A,a)=\delta(A,a)=\cup_{q\in A}\delta(q,a)$$

es decir la unión de todos los estados que se alcanzan en la máquina dada desde los estados que están en A, con el símbolo a. Ahora $M'=(Q', \Sigma, \{q_0\}, \delta', F')$ es un AFD y afirmamos que $L(M)=L(M')$.

Con esta construcción queda demostrado que el no determinismo no aporta nada nuevo al conjunto de los lenguajes que son abarcables por medio de los AFDs, es decir

$$\{L\subseteq\Sigma^* : \exists M=AFD, L=L(M)\}=\{ L\subseteq\Sigma^* : \exists M =AFND, L=L(M)\}.$$

Ejemplo:

Vamos a construir un AFD equivalente a M_5. Aunque hemos dicho de forma genérica que el conjunto de estados para la máquina nueva es P(Q), la mayoría de las veces muchos de estos nuevos estados van a ser inaccesibles por lo que podremos eliminarlos. Lo que se hace es ir añadiendo estados a partir del estado inicial que en este caso es {p}, según vayan apareciendo al definir la nueva tabla de transiciones. Así, al comenzar, es una tabla abierta, no sabemos cuántos de los nuevos estados serán accesibles. (Por comodidad no escribimos las llaves de conjunto).

Las tablas de M_5 y M'_5 son respectivamente

δ	0	1
→p	q,r	∅
*q	q	q,s
r	∅	s
*s	p	∅

δ´	0	1
→p	q,r	∅
*q,r	q	q,s
∅	∅	∅
q	q	q,s
*q,s	p,q	q,s
*p,q	q,r	q,s

Tabla 2.9. M_5 (izquierda) y M_5' (derecha), AFD equivalente a M_5.

Y gráficamente la nueva máquina equivalente es

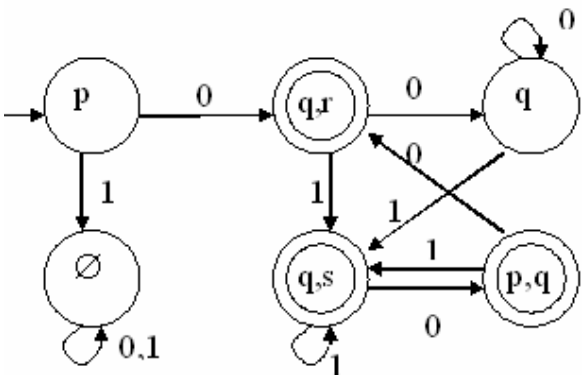
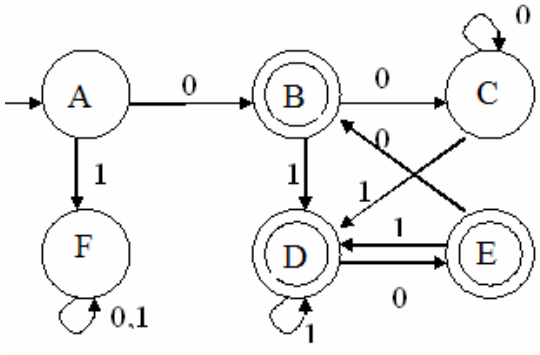


Fig.2.9. Diagrama de M_5'.

Debemos hacer un par de observaciones. El estado marcado ∅ refleja la situación de todos los estados inactivos en la máquina original -todas las bombillas apagadas- y es el estado de absorción, por tanto si queremos dibujar el diagrama correspondiente no hace falta que lo incluyamos. Por otra parte es un estado en sí mismo y debemos recordar que si ahora queremos minimizar el AFD obtenido es uno de los estados no finales y no debemos olvidarnos de incluirlo. Lo más cómodo es renombrar en este punto los estados.



Fig, 2.10. M_5' con los estados renombrados

2.3 AUTÓMATA FINITO NO DETERMINISTA CON λ-TRANSICIONES (AFND-λ)

Una de las características de los autómatas vistos hasta ahora es que sin lectura de un símbolo del alfabeto no hay transición. El tipo de máquina que vamos a definir a continuación elimina esta restricción permitiendo cambios de estado sin lectura.

2.3.1 DEFINICIÓN DE AFND-λ.

Un autómata finito no determinista con λ-transiciones es una quíntupla $M=(Q,\Sigma,q_0,\delta,F)$ donde todos los elementos están definidos como en los autómatas ya vistos, salvo la función de transición $\delta:Q\times(\Sigma\cup\{\lambda\})\rightarrow P(Q)$. Es decir ahora no sólo hay transiciones del tipo $\delta(q,a)$ donde $q\in Q$ y $a\in\Sigma$, sino que también se definen transiciones del tipo $\delta(q,\lambda)=A\in Q$.

Ejemplo:

Si la máquina M_5 añadimos las transiciones

$\delta(p,\lambda)=\{r\}$ $\delta(q,\lambda)=\{p\}$ $\delta(r,\lambda)=\varnothing$ $\delta(s,\lambda)=\{q\}$

obtenemos una nueva máquina M_6.

2.3.2 REPRESENTACIÓN DE UN AFND-λ

La forma de representación es similar a lo visto hasta ahora

1. Con una tabla: Se ponen tantas filas como estados rotuladas con los nombres de los estados; tantas columnas como símbolos tenga el alfabeto de entrada rotuladas con los nombres de los símbolos y se añade una columna rotulada λ. En el cruce de fila y columna se escribe el conjunto δ(q,a), como es habitual.

2. Mediante un diagrama: Se dibuja un círculo por cada estado no final y un doble círculo por cada estado final; se marca el estado inicial con una flecha que entra sin etiquetar; si $r\in\delta(q,a)$ se dibuja una flecha con origen en *q* y final en *r*, rotulada *a* (donde *a* es un símbolo del alfabeto o λ).

Ejemplo:

Para la máquina M_6 la tabla es

δ	0	1	λ
→p	q,r	∅	r
*q	q	q,s	p

r	∅	s	∅
*s	p	∅	∅

Tabla 2.10. Tabla para el AFND-λ,M_6.

Y el diagrama

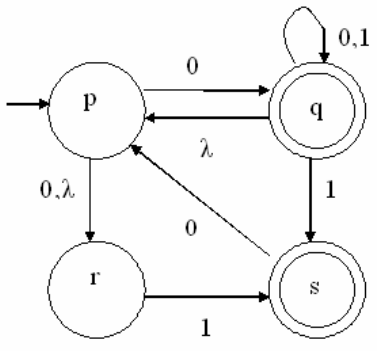


Fig. 2.11. Diagrama del AFND-λ,M_6.

2.3.3 λ-CLAUSURA DE UN ESTADO

La λ-clausura de un estado *p* es el conjunto de estados a los que se puede llegar desde *p* sólo por caminos etiquetados *λ*.

Se debe observar que siempre $p \in \lambda\text{-cl}(p)$ y por tanto $\lambda\text{-cl}(p) \neq \emptyset$.

Ejemplo:

Vamos a escribir las λ-clausuras de cada uno de los estados de M_6.

$$\lambda\text{-cl}(p)=\{p,r\} \qquad \lambda\text{-cl}(q)=\{p,q,r\} \qquad \lambda\text{-cl}(r)=\{r\} \qquad \lambda\text{-cl}(s)=\{q,s\}$$

Como ya hicimos anteriormente con los AFND para un subconjunto A de estados escribimos

$$\delta(A,a)=\cup_{r\in A}\delta(r,a)$$

Es sencillo demostrar que $\lambda\text{-cl}(\lambda\text{-cl}(A))=\lambda\text{-cl}(A)$.

2.3.4 EXTENSIÓN DE LA FUNCIÓN DE TRANSICIÓN.

Se trata de nuevo de definir una función que describa qué estados se alcanzan desde un estado *q* si en vez de entrar un sólo símbolo, entrara una palabra ω∈Σ*.

Será ahora $\hat{\delta}$:Q×Σ*→P(Q) y definimos como siempre $\hat{\delta}$ (q,ω) por recurrencia sobre la longitud de ω.

- Si |ω|=0 entonces ω=λ, definimos ∀q∈Q, δ(q,λ)=λ-cl(q) (desde q, sin entrada de ningún símbolo, se alcanzan todos los estados de la λ-cl(q)).
- Supongamos definido $\hat{\delta}(q,x)$ para cada palabra x tal que |x|≤n.
- Sea ω∈Σ* tal que |ω|=n+1, entonces ω se puede escribir ω=xa con |x|=n. Ahora definimos ∀q∈Q

$$\hat{\delta}(q,xa)=\lambda\text{-}cl(\delta(\hat{\delta}(q,x),a)).$$

Para las palabras de longitud 1 se tiene

$$\hat{\delta}(q,a)=\hat{\delta}(q,\lambda a)=\lambda-cl\left(\delta\left(\hat{\delta}(q,\lambda),a\right)\right)=\lambda-cl(\delta(\lambda-cl(q),a))$$

es decir la función de extensión a palabras y la función de transición original no tienen por qué coincidir para palabras de longitud 1, es decir los símbolos del alfabeto.

2.3.5 PALABRA ACEPTADA POR UN AFND-λ.

Sea ω∈Σ* decimos que ω es una palabra aceptada por el AFND-λ si desde el estado inicial y leyendo la palabra completa alguno de los estados que se alcanzan es final, es decir si

$$\hat{\delta}(q_0,\omega)\cap F\neq\varnothing$$

2.3.6 LENGUAJE ACEPTADO POR UN AFND-λ.

Es el conjunto de todas las palabras aceptadas es decir

$$L(M)=\left\{\omega\in\Sigma^*:\hat{\delta}(q_0,\omega)\cap F\neq\varnothing\right\}$$

Los AFND son un caso particular de AFND-λ en que todos las imágenes δ(q,λ) son el conjunto vacío.

2.3.7 CONSTRUCCIÓN DE UN AFND EQUIVALENTE A UN AFND-λ DADO.

Sea M=(Q,Σ,q_0,δ,F) un AFND-λ que supondremos conocido. Queremos construir un AFND M′ tal que L(M) = L(M′).

Antes de continuar debemos de nuevo recordar que esta construcción garantiza la existencia de un AFND equivalente, pero en ningún caso se garantiza la unicidad, es decir hay muchos AFNDs equivalentes a M; nosotros construiremos uno de ellos.

La idea para la construcción de la nueva máquina es que desde cualquier estado *q* de la máquina original, cuando se lee en cinta un símbolo *a* del alfabeto, se alcanzan los nuevos estados en tres tiempos: los que se alcanzan desde *q* por λ transiciones (la *λ*-clausura de *q*); los que se alcanzan por la lectura del símbolo leído, que son

$\delta(\lambda\text{-cl}(q),a)$ y los que se alcanzan a partir de éstos por las transiciones λ, que son $\lambda\text{-cl}(\delta(\lambda\text{ –cl}(q),a))$ en definitiva un subconjunto de estados, concretamente $\hat{\delta}(q,a)$. Además si ocurre que desde q_0 se llega a algún estado final por caminos etiquetados λ, o lo que es lo mismo si $\lambda\text{-cl}(q_0)\cap F\neq\varnothing$, una de las palabras aceptadas es λ; en tal caso, para que la nueva máquina acepte λ el estado inicial debe ser final. De modo que tomamos M′=(Q,Σ, q_0,δ′,F′), siendo

$$F' = \begin{cases} F & si\ \ \lambda - cl(q_0) \cap F = \varnothing \\ F \cup \{q_0\} & si\ \ \lambda - cl(q_0) \cap F \neq \varnothing \end{cases}$$

y la nueva función de transición como sigue: δ′:Q×Σ→P(Q),

$$\delta'(q,a) = \hat{\delta}(q,a) = \lambda - cl(\delta(\lambda - cl(q),a))$$

Ahora M′ es un AFND y afirmamos que L(M)=L(M′).

Con esta construcción queda demostrado que las transiciones λ no aportan nada nuevo al conjunto de los lenguajes que son abarcables por medio de los AFNDs-λ, es decir

$$\{L \subseteq \Sigma^* : \exists M = AFND - \lambda, L = L(M)\}$$
$$= \{L \subseteq \Sigma^* : \exists M = AFND, L = L(M)\}$$
$$= \{L \subseteq \Sigma^* : \exists M = AFD, L = L(M)\}$$

Los tres tipos de Autómatas vistos, a los que a partir de ahora llamaremos simplemente Autómatas Finitos, son equivalentes en cuanto a los lenguajes que pueden abarcar.

Ejemplo:

Construimos un AFND equivalente a M_6

δ	0	1	λ
→p	q,r	∅	r
*q	q	q,s	p
r	∅	s	∅
*s	p	∅	∅

λ-clausura
p, r
p, q, r
r
s

δ'	0	1
→p	p,q,r	s
*q	p,q,r	p,q,r,s
r	∅	s
*s	p,r	∅

Tabla 2.11. Autómata M_5(izquierda) y su equivalente sin λ-transiciones (derecha).

2.4 EQUIVALENCIA DE AUTÓMATAS FINITOS Y GRAMÁTICAS REGULARES.

Recordemos que una gramática decimos que es regular lineal por la derecha si sus reglas son de una de las tres formas siguientes S:=λ o A:=*a* o A:=*a*B siendo A,B y S símbolos no terminales, S el axioma y *a* un símbolo terminal.

Lo que veremos a continuación es que para cualquier autómata finito existe una gramática regular lineal a la derecha equivalente y, recíprocamente, que dada una gramática regular lineal a la derecha existe un autómata finito equivalente. O lo que es lo mismo

$$\{L \subseteq \Sigma^* : \exists AF, L = L(AF)\} = \{L \subseteq \Sigma^* : \exists G_3,\ L = L(G_3)\}$$

Se demuestra por separado cada uno de los términos de la doble inclusión.

2.4.1 GRAMÁTICA REGULAR EQUIVALENTE A UN AF DADO.

Sea M =(Q,Σ,q_0,δ,F) un autómata conocido (debe ser un autómata sin λ-transiciones). Queremos construir una gramática G = (Σ_T,Σ_N,S,P) tal que L(M) = L(G).

Tomamos como alfabeto de terminales el alfabeto de entrada del autómata, Σ; como conjunto de símbolos no terminales tomamos Q; como axioma para la gramática tomamos q_0, el estado inicial del autómata; por último traducimos las transiciones del autómata en reglas para la gramática de la forma siguiente:

si δ(p,a) = q y q∉F añadimos la regla p:=aq

si δ(p,a) = q y q∈F añadimos las reglas p:=aq y p:=a

si q_0∈F añadimos la regla q_0:=λ.

Ejemplo:

Sea M_7 el AFND dado por la tabla

δ	0	1
→p	q,r	∅
*q	q	q,s
r	∅	s
*s	p	∅

Tabla 2.12. Tabla de M_7.

Una gramática equivalente es G ={{0,1},{p,q,r,s},p,*P*}, siendo *P* las reglas

p:= 0q / 0 / 0r

q:= 0q / 0 /1q / 1 / 1s

r:= 1s / 1

s:= 1p

Por la transición δ(p,0) = q, introducimos las reglas p:= 0q / 0, por ser q estado final. Por la transición δ(p,0)=r, introducimos la regla p:= 0r, por ser r un estado no final.
Y así sucesivamente.

2.4.2 AUTÓMATA FINITO EQUIVALENTE A UNA GRAMÁTICA DADA.

Suponemos que la gramática $G=(\Sigma_T,\Sigma_N,S,P)$ es conocida (debe ser una gramática regular a la derecha). Queremos construir un autómata $M = (Q,\Sigma,q_0,\delta,F)$ tal que $L(G)= L(M)$.

Como conjunto de estados tomamos $\Sigma_N \cup \{f\}$ donde f es un símbolo que no pertenece al alfabeto de no terminales; como alfabeto de entrada se toma el alfabeto de terminales de la gramática, Σ_T; como estado inicial se toma el estado representado por el axioma S de la gramática; como conjunto de estados finales tomamos F={f}; por último definimos las transiciones traduciendo las reglas de la gramática de la forma siguiente:

Si A:=aB es una regla, definimos la transición $\delta(A,a)=B$.

Si A:=a es una regla, definimos la transición $\delta(A,a)=f$.

Si S:=λ es una regla, definimos la transición $\delta(S,\lambda)=f$.

Ejemplo:

Sea la gramática $G_1=\{\{0,1\},\{A,B,C,D\},A,P\}$, siendo P las reglas

A:= 0B / 0 / 0C

B:= 0B / 0 /1B / 1 / 1D

C:= 1D / 1

D:= 1A

Para construir un AF equivalente tomamos Q={A,B,C,D,f}, Σ={0,1}, estado inicial A, estado final f y la tabla

	0	1
→A	B,C,f	∅
B	B,f	B,D,f
C	∅	D,f
D	∅	A
*f	∅	∅

Tabla 2.12. Autómata equivalente a la gramática G_1.

Por la regla A:=0B hemos puesto la transición $\delta(A,0)=B$. Por la regla A:=0 hemos puesto la transición $\delta(A,0)=f$. Y así sucesivamente.

2.5 EXPRESIONES REGULARES

Las expresiones regulares son una manera resumida de representar ciertos lenguajes sobre un alfabeto. Sea Σ un alfabeto, cada expresión regular representa un cierto lenguaje sobre Σ.

Son expresiones regulares

- $\varnothing$ es una expresión regular que representa al lenguaje $\varnothing$.
- λ es una expresión regular y representa al lenguaje $\{\lambda\}$.
- $\forall a \in \Sigma$, a es una expresión regular y representa a $\{a\}$.
- Si α y β son expresiones regulares $\alpha+\beta$ es una expresión regular y representa a
- $L(\alpha) \cup L(\beta)$.
- Si α y β son expresiones regulares $\alpha\beta$ es una expresión regular y representa a $L(\alpha)L(\beta)$.
- Si α es una expresión regular, α^* es una expresión regular y representa a $L(\alpha)^*$.
- Si α es una expresión regular, α^+ es una expresión regular y representa a $L(\alpha)^+$.
- Si α es una expresión regular, (α) es una expresión regular y representa a $L(\alpha)$.
- Cualquier combinación finita de expresiones regulares unidas mediante los operadores definidos (concatenación, unión, clausura positiva y cierre de Kleene) es una expresión regular.

Ejemplos:

$r_1 = (01)^* = \{(01)^k : k \geq 0\}$

$r_2 = (01)^*+0+10 = \{(01)^k : k \geq 0\} \cup \{0,10\}$

$r_3 = (ab^*+b^*a)ab^+ = \{aab, aabb, aabbb...., baab, bbab....\}$

2.6 EQUIVALENCIA DE EXPRESIONES REGULARES Y AUTÓMATAS FINITOS.

Lo que vamos a demostrar a continuación es que los Lenguajes que pueden ser expresados mediante una expresión regular, son *todos y los únicos* lenguajes que son abarcables por los Autómatas Finitos.

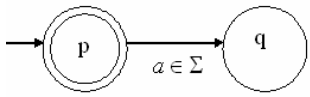
2.6.1 AUTÓMATA FINITO EQUIVALENTE A UNA EXPRESIÓN REGULAR DADA.

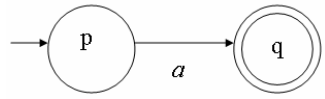
Lo que se quiere comprobar principalmente es que dada una expresión regular existe siempre un autómata cuyo lenguaje es el denotado por la expresión regular, es decir que $L(ER) \subseteq L(AF)$. La demostración es constructiva, el autómata obtenido es con λ-transiciones y normalmente tiene muchos estados. Es conveniente combinar la intuición con el método que se explica a continuación.

La construcción se hace por recurrencia sobre el número *k* de operadores de la expresión regular y demostrando que dada una expresión regular cualquiera siempre existe un AF, *con sólo un estado final*, cuyo lenguaje es el dado por la expresión regular.

- Si k=0 la expresión regular sólo puede ser de uno de los tres siguientes tipos:

  i) $r = \varnothing$, un AF cuyo lenguaje sea r es 

  ii) $r = \lambda$ un AF cuyo lenguaje sea r es 

  iii) $r = a \in \Sigma$, un AF cuyo lenguaje sea r es 

- Supongamos, por hipótesis de recurrencia, que si la expresión regular contiene un número menor o igual a *k* de operadores, existe un AF con un solo estado final para el lenguaje denotado por la expresión regular.

- Sea r una expresión regular con *k*+1 operadores. La construcción del AF la haremos según cual sea el último operador que interviene en r.

a) Si el último operador es la concatenación entonces podemos escribir $r = r_1r_2$ donde r_1 y r_2 tienen menos de k+1 operadores. Entonces, por hipótesis de inducción, existen dos AF, M_1y M_2 tales que $L(M_1) = r_1$ y $L(M_2) = r_2$.

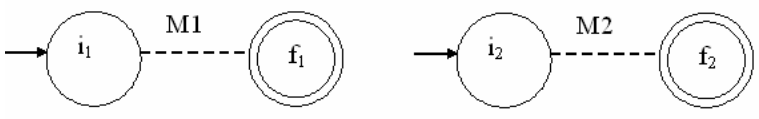


Fig. 2.12. Autómatas tales que $L(M_1)=r_1$ y $L(M_2)=r_2$.

Un autómata que acepta L(r) sería

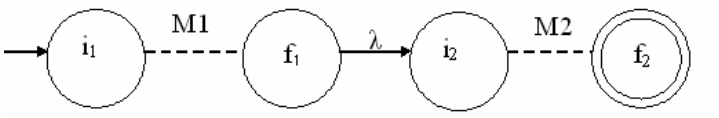


Fig 2.13. AF cuyo lenguaje es r_1r_2

b) Si el último operador de r es una unión $r = r_1+r_2$, un autómata que acepta L(r) será

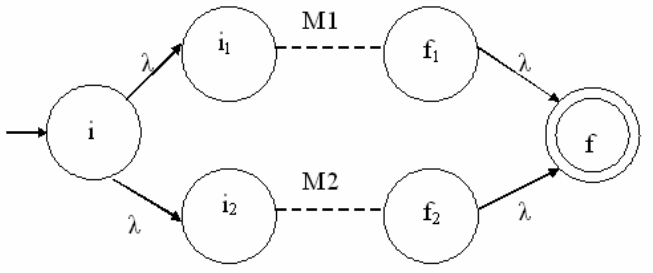


Fig 2.14. AF cuyo lenguaje es r_1+r_2

c) Si el último operador de r es un cierre de Kleene $r = (r_1)^*$ un autómata que acepta L(r) será:

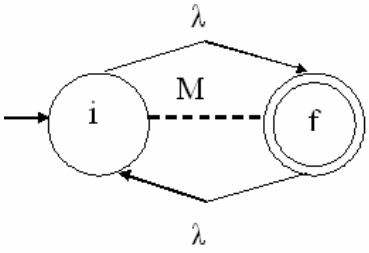


Fig 2.15. AF cuyo lenguaje es $(r_1)^*$.

d) Si el último operador de r es una clausura positiva $r=(r_1)^+$ un autómata que acepta L(r) será:

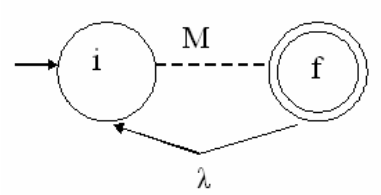


Fig 2.16. AF cuyo lenguaje es $(r_1)^+$.

Ejemplo:

Sea $\Sigma=\{0,1\}$ y sea $r = 0^*+1^+0$ y queremos construir un autómata cuyo lenguaje sea exactamente el definido por la expresión regular r. El último operador que interviene es la suma

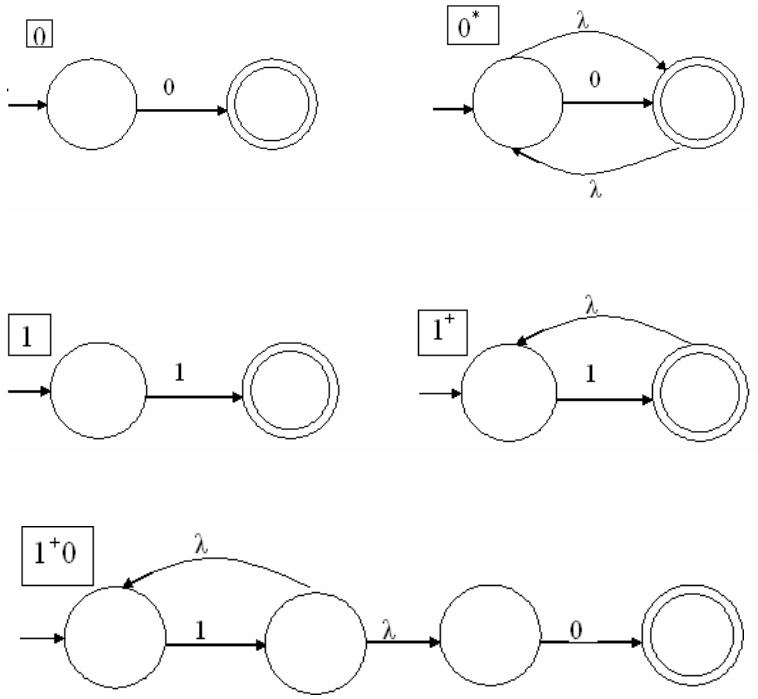


Fig. 2.17. Construcción de un AF cuyo lenguaje es $r = 0^*+1^+0$.

Por último un AF cuyo lenguaje es L(r) es

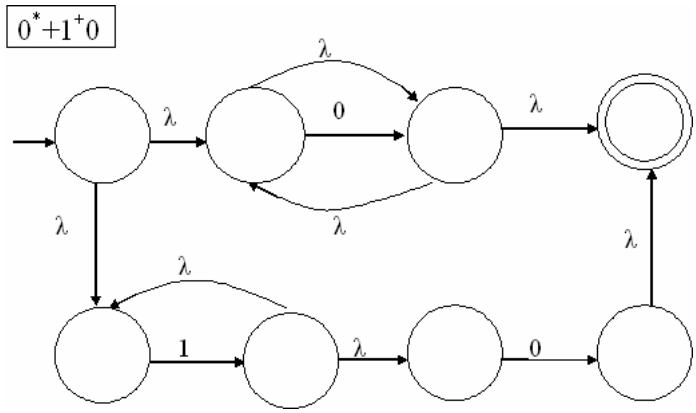


Fig. 2.18. Un AF cuyo lenguaje es r = 0*+1+0.

Como puede verse el autómata construido aplicando directamente el método estudiado tiene muchos estados y muchas λ-transiciones, aún cuando la expresión regular sea relativamente simple. Normalmente se puede construir otro autómata equivalente más simple, por ejemplo

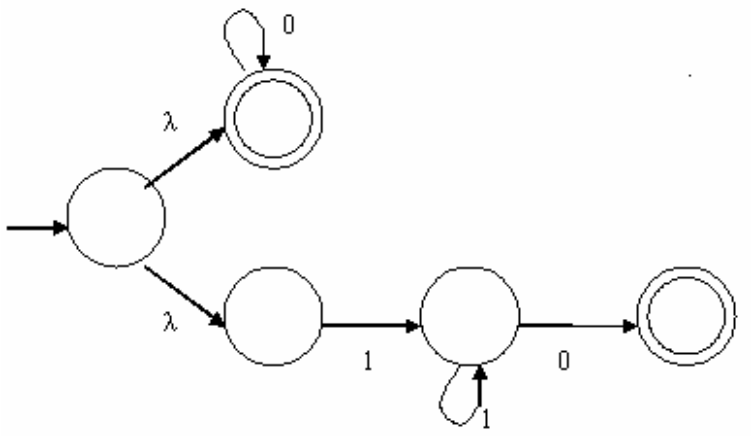


Fig. 2.19. Otro AF con menos estados cuyo lenguaje es también r = 0*+1+0

2.6.2 EXPRESIÓN REGULAR EQUIVALENTE A UN AUTÓMATA FINITO DADO.

Suponemos que M es un autómata finito dado, *sin transiciones* λ. Buscamos una expresión regular r tal que L(M) = r.

Para cada estado q∈Q definimos

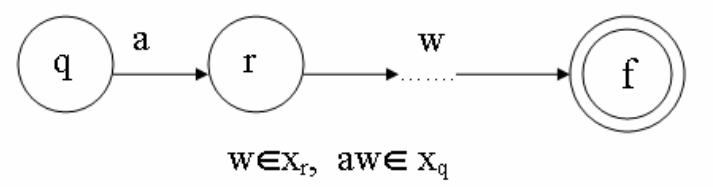
$$x_q = \left\{\omega \in \Sigma^* : \hat{\delta}(q,\omega) \cap F \neq \varnothing\right\}$$

es decir el conjunto de todas las palabras que desde *q* alcanzan algún estado final.

Queremos escribir de manera más explicita x_q, para cada estado del autómata dado. Observemos que

i) $\lambda \in x_q$ si y sólo si $q \in F$

ii) Si $r \in \delta(q,a)$ y $w \in x_r$ entonces $aw \in x_q$ o lo que es lo mismo si $r \in \delta(q,a)$ entonces $ax_r \subseteq x_q$.



Con estas dos propiedades en mente escribimos lo que llamamos *la ecuación lineal del estado q*

Si $q \notin F$ escribimos $\quad x_q = \sum_{a \in \Sigma} \sum_{r \in \delta(q,a)} a x_r$

Si $q \in F$ escribimos $\quad x_q = \sum_{a \in \Sigma} \sum_{r \in \delta(q,a)} a x_r + \lambda$

donde las sumas representan uniones como en las expresiones regulares.

Cuando tengamos una ecuación del tipo

$$x_q = A x_q + B$$

donde A es el conjunto de todas las palabras que partiendo de q llegan de nuevo a q y B es el conjunto de todas las palabras que van de q a un estado final sin volver a pasar por q. Entonces podemos escribir

$$x_q = A^* B$$

Debemos resolver el sistema usando las propiedades de las operaciones con lenguajes hasta despejar $x_{q_0} = L(M)$.

Ejemplo:

Sea el AFND M_7 cuya tabla era

δ	0	1
→p	q,r	∅
*q	q	q,s
r	∅	s
*s	p	∅

Tabla 2.13. Tabla de M_7.

las ecuaciones lineales M_3 de son

$$x_p = 0x_q + 0\ x_r$$
$$x_q = 0\ x_q + 1\ x_q + 1\ x_s + \lambda$$
$$x_r = 1\ x_s$$
$$x_s = 0x_p + \lambda$$

Para resolver el sistema tenemos que proceder aplicando las propiedades de las operaciones con lenguajes y lo que se acaba de explicar

$$x_r = 1(0\ x_p + \lambda) = 10\ x_p + 1$$
$$x_q = (0+1)\ x_q + 1(0\ x_p + \lambda) + \lambda = (0+1)\ x_q + 10\ x_p + 1 + \lambda =$$
$$= (0+1)^*(10\ x_p + 1 + \lambda)$$
$$x_p = 0(0+1)^*(10\ x_p + 1 + \lambda) + 0(10\ x_p + 1)$$
$$= [0(0+1)^*10 + 010]\ x_p + 0(0+1)^*(1+\lambda) + 01$$
$$= [0(0+1)^*10 + 010]^*[0(0+1)^*(1+\lambda) + 01]$$

ésta última es una expresión regular para el lenguaje del autómata dado.

Hemos demostrado que las clases de las gramáticas regulares, los autómatas finitos y las expresiones regulares son equivalentes y hemos explicado técnicas para construir objetos de cada una de las clases equivalentes entre sí. Queremos volver a señalar que los objetos se caracterizan por los lenguajes que llevan asociados y que en ninguna de las tres clases se da la unicidad. Presentamos un gráfico que nos muestra estas relaciones

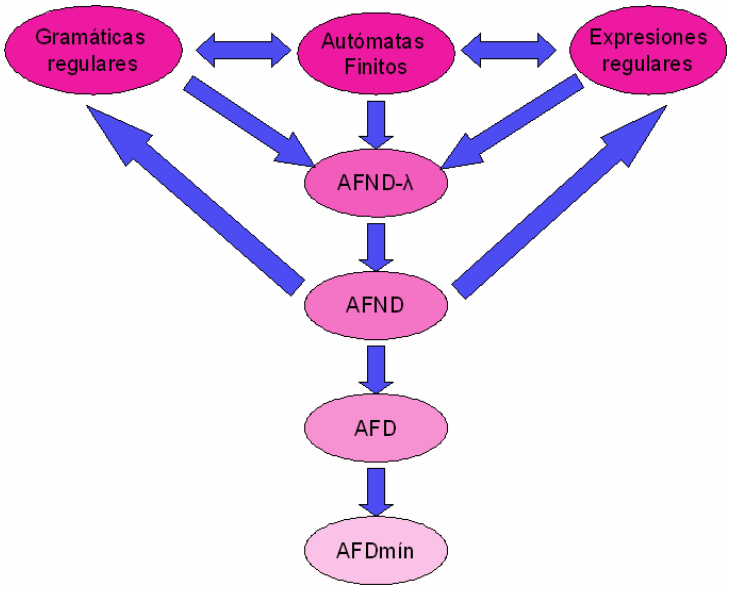


Fig. 2.20. Relación entre Gramáticas regulares, Autómatas Finitos y Expresiones regulares.