Teoría de la Computación para Ingeniería de Sistemas: un enfoque práctico

Prof. Hilda Contreras

6 de febrero de 2018

Índice general

1. Lenguajes Regulares y Autómatas de Estados Finitos **5**
1.1. Autómatas de estados finitos 5
1.1.1. Autómatas finitos determinista (AFD) 5
1.1.2. Autómatas finitos no determinista (AFND) 9
1.1.3. Autómatas finitos no determinista con transiciones nulas (AFND-λ) 13
1.1.4. Herramientas, implementación y uso de Lenguajes regulares 18
1.1.5. Preguntas y respuestas, ejercicios resueltos y propuestos 21

Capítulo 1

Lenguajes Regulares y Autómatas de Estados Finitos

Los lenguajes regulares, de tipo 3 según la jerarquía de Chomsky, son aquellos que son **reconocidos** por *autómatas de estados finitos*, son **denotados** por *expresiones regulares* y **generados** por *gramáticas regulares*. Estos lenguajes contienen a todos los lenguajes finitos generados a partir de cualquier alfabeto. Los lenguajes infinitos tipificados como regulares poseen ciertas propiedades que lo caracterizan y distinguen de otros lenguajes más complejos (ver tema sobre las propiedades y algoritmos de decisión de los lenguajes regulares).

1.1. Autómatas de estados finitos

Los Autómatas de Estados Finitos (AF): Describen, reconocen y definen a los Lenguajes Regulares. Es el modelo matemático de un sistema con estados y salidas discretas. Los Tipos de Autómatas Finitos son:

1. Autómata Finito Determinista (AFD). No pueden estar en más de un estado simultáneamente.
2. Autómata Finito No Determinista (AFN). Puede estar en varios estados al mismo tiempo.

1.1.1. Autómatas finitos determinista (AFD)

Un AFD es una quíntupla A = (Q , Σ, δ, q_0, F), siendo:

Q = conjunto finito de estados.

Σ = conjunto finito de símbolos del alfabeto.

q_0 = es el estado inicial (denotado con flecha $\rightarrow$ a inicio)

F = conjunto de estados finales (o estados de aceptación), F $\subseteq$ Q

δ = La función de transición entre estados, δ: Q x $\Sigma \rightarrow$ Q.

En los AFD debe cumplirse que: Para toda a que esta en Σ existe exactamente una transición de salida de cada estado, es decir la función de transición dado un estado y un valor de entrada se obtiene como resultado siempre un solo valor (estado).
Las siguientes son las tres formas para expresar y representar la función de transición, las cuales son equivalentes entre si, cualquiera de ellas es suficiente para completar el diseño del AFD:

1. **Diagrama de Transición**: representado a través de un *grafo*. Un grafo G = (V,E) consiste de un conjunto finito de vértices (o nodos) V y un conjunto finito de pares de vértices (o aristas) E llamados enlaces (los cuales son bidireccionales).
   Un camino en un grafo es una secuencia de vértices V_1, V_2, ... , V_k, con k $\geq$ 1, tal que hay un enlace (V_i,V_{i+1}) con 1 $\leq$ i $\leq$ k. La longitud del camino es k-1. Si V_1 = V_k entonces el camino es un ciclo.
   Una clase de grafo dirigido Dígrafo G =(V,E), consiste en un conjunto finito de vértices V (o nodos) y un conjunto finito de pares ordenados de vértices E llamados arcos (una sola dirección). Un arco de v a w se denota como v $\rightarrow$ w.
   Un camino en un dígrafo es una secuencia de vértices V_1, V_2, ... , V_k, con k $\geq$ 1, tal que $V_i \rightarrow V_{i+1}$ es un arco con 1 $\leq$ i $\leq$ k. La longitud del camino es k-1. El camino es de V_1 a V_k (importa la dirección). Si v $\rightarrow$ w es un arco entonces v es predecesor de w y w es sucesor de v.
   Ejemplo: Dado el Grafo G = ({1,2,3,4}, {i $\rightarrow$ j | i $<$ j })
   1,2,3,4 es un camino de longitud 3 de 1 a 4.
   1,4 es un camino de longitud 1 de 1 a 4.
   3,3 no es un camino.
   3 es un camino de longitud 0 de 3 a 3?
   Por notación, dado un AFD si p y q estan en Q y a en Σ, se define la transición δ(q,a) = p. como un arco (enlace) etiquetado a entre el estado (vértice) q y el estado (vértice) p. Los estados o vértices de aceptación se denotan gráficamente con un doble círculo. Por ejemplo si r esta en F debe ser denotado con doble círculo como se observa en la figura 1.1.

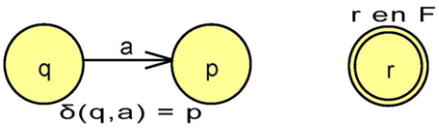

Figura 1.1: Ejemplo de transición en AFD

2. **Función de transición**: representada a través de una *función*. Una función asigna a cada elemento de un conjunto (Conjunto Dominio) un elemento de otro conjunto (Conjunto Rango o Codominio). Así la función f se denota f : A $\rightarrow$ B, siendo A el dominio y B el rango. Por ejemplo, la siguiente función f(x)= x^2, se define f: $\mathbb{N} \rightarrow \mathbb{N}$, donde $\mathbb{N}$ es el conjunto de los números naturales.
   Los tipos de funciones son:

   - Inyectiva: Si $a_1 \neq a_2$, entonces $f(a_1) \neq f(a_2)$
   - Sobreyectiva: Si todo elemento de B es imagen de alguno de A, es decir la función inversa [1] $f^{-1}(b) \neq \phi$, para todo b en conjunto Rango o Codominio.

[1] Dada f:A $\rightarrow$ B, la función inversa f^{-1}:B $\rightarrow$ A, se define como $f^{-1}(b)$ = {x | x en A y f(x) = b } para todo b en B

Cuadro 1.1: Estructura de la Tabla de Transición

δ	Símbolos de Σ
Estados	Estados

La función de transición de un AFD esta definida como δ: Q x $\Sigma \rightarrow$ Q, es decir tiene como dominio cualquier par del producto cartesiano (Q x Σ) del conjunto de estados Q y el alfabeto de entrada Σ, y da como resultado un elemento del conjunto rango de estados Q. En la figura 1.1 se muestra un ejemplo de una función de transición δ(q,a) = p.

3. **Tabla de Transiciones**: representada a través de una matriz. La tabla tiene en las columnas todas las entradas del alfabeto Σ y en las filas los estados de Q, por lo que el par FilaxColumna representa el dominio de la función de transición (Q x Σ). Por notación, en la tabla se escriben los estados de aceptación o estados finales con asterisco al lado del estado y el estado inicial con una flecha. En cada celda interna de la tabla se muestra el resultado de la transición (codominio) es decir el estado Q al cual se realiza la transición dado la fila y la columna de la posición de la celda respectiva. En los AFD cada celda de la tabla debe tener un estado, no se permiten celdas vacias. Ver cuadro 2.1.

Ejemplo: Dado Σ = { 0, 1 } y el lenguaje L = { x | x en Σ^* y x comienza con 0 }, es decir todas las cadenas binarias que comienzan con 0. El AFD que lo reconoce, en sus tres representaciones, es el siguiente:
M = (Q, Σ, δ, q_0, F)
Donde Q = {q_0, q_1, q_2}, Σ= {0,1} y F = {q_1}

1. Diagrama de transición: se observa en la figura 1.2.

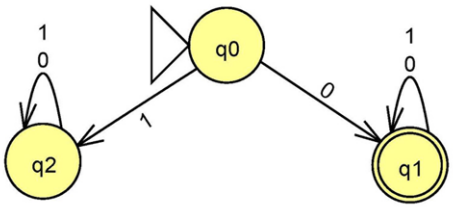

Figura 1.2: Ejemplo de AFD: cadenas binarias que comienzan con cero

2. Función de transición δ
$\delta(q_0, 0) = q_1$
$\delta(q_0, 1) = q_2$
$\delta(q_1, 0) = q_1$
$\delta(q_1, 1) = q_1$
$\delta(q_2, 0) = q_2$

Cuadro 1.2: Ejemplo de Tabla de Transición

δ	0	1
$\rightarrow q_0$	q_1	q_2
$^*q_1$	q_1	q_1
q_2	q_2	q_2

$\delta(q_2, 1) = q_2$

3. Tabla de transición: se observa en el cuadro 1.2

Las tres representaciones del AFD son equivalentes y cada una de ellas es suficiente para expresar el modelo. Sin embargo, debe enterderse la relación entre ellas aunque se seleccione solo una de ellas para representar al autómata.

Ejecución de un Autómatas de estados finitos determinista (AFD)

Para la ejecución formal de un AFD se utiliza la definición de la función de transición extendida sobre cadenas $\widehat{\delta}$. La definición de su dominio y rango es: $\widehat{\delta}$ = Q x Σ^* $\rightarrow$ Q. La definición recursiva de la función $\widehat{\delta}$ se define:

1. Caso base para la cadena vacía λ: Para todo q en Q, $\widehat{\delta}$(q,λ) = q. Esto quiere decir que desde cualquier estado la cadena vacía λ no cambia el estado del AFD.

2. Caso recursivo para cadenas de cardinalidad mayor que 1: Para todo w en Σ^* y a en Σ, se cumple que $\widehat{\delta}$(q,wa) = $\delta(\widehat{\delta}$(q,w),a).

Por ejemplo: Dado el AFD anterior que reconoce el lenguaje sobre Σ = {0, 1} y L = {x | x en Σ^* y x comienza con 0}, pruebe si las cadenas w_1 = 010 y w_2 = 101 están en el lenguaje L:

1. $\widehat{\delta}$(q$_0$,010)
   = $\delta(\widehat{\delta}(q_0$,01),0)
   = $\delta(\delta(\widehat{\delta}(q_0$,0),1),0)
   = $\delta(\delta(\delta(\widehat{\delta}(q_0, \lambda)$,0),1),0)
   $\rightarrow \delta(\delta(\delta(q_0, 0), 1), 0) \rightarrow (\delta(q_1, 1), 0) \rightarrow \delta(q_1, 0) = q_1 \rightarrow$ Se acepta al terminar de procesar la cadena 010 y quedar en un estado de aceptación $\widehat{\delta}$(q$_0$,010) = q_1 (q_1 está en F)

2. $\widehat{\delta}$(q$_0$,101)
   = $\delta(\widehat{\delta}(q_0$,10),1)
   = $\delta(\delta(\widehat{\delta}(q_0$,1),0),1)
   = $\delta(\delta(\delta(\widehat{\delta}(q_0, \lambda)$,1),0),1)
   $\rightarrow \delta(\delta(\delta(q_0, 1), 0), 1) \rightarrow (\delta(q_2, 1), 0) \rightarrow \delta(q_2, 0) = q_2 \rightarrow$ No se acepta al terminar de procesar la cadena 101 y quedar en un estado que no es de aceptación $\widehat{\delta}$(q$_0$,101) = q_2 (q_2 no está en F)

Lenguaje aceptado por un AFD

Dado M = (Q, Σ, δ, q_0, F) un AFD, el lenguaje L aceptado por M está definido por:

$$\boxed{\text{L(M)= \{ w | w en } \Sigma^* \text{ y } \widehat{\delta}(q_0,w) \text{ pertenece a F \}}}$$

Es decir, el lenguaje de M, es un conjunto de cadenas w que llevan al autómata M desde q_0 a un estado de aceptación. Si L es un L(M) para algún AFD M, entonces decimos que L es un *Lenguaje Regular*.
Nota importante: Para poder verificar si la cadena w está (es aceptada) o no está (no es aceptada) en L se debe terminar de procesarse la cadena w (desde el principio hasta el final).

1.1.2. Autómatas finitos no determinista (AFND)

Son autómatas de estados finitos que tienen la capacidad de estar en más de un estado simultáneamente. No hay que considerar todos los casos en cada estado, ya que permiten cero, una o más transiciones de salida de un estado para el mismo símbolo del alfabeto de entrada. Ellos tienen la ventaja de ser más compactos, flexibles y fáciles de diseñar que los AFD.
Esta compuesta por la misma tupla de elementos que un AFD A = (Q, Σ, δ, q_0, F), la diferencia es la definición de la función $\widehat{\delta}$ y el rango de la función de transición, δ: Q x $\Sigma \rightarrow 2^Q$ ó también δ: Q x $\Sigma \rightarrow Q^*$
En la figura 1.3 se muestra el diagrama de transición δ(q,a) = {p,r}, donde el AFND dado el estado q y el símbolo a se mueve a dos estados (un conjunto de estados) simultáneamente. Como δ: Q x $\Sigma \rightarrow Q^*$, entonces δ(q,a) = R, donde R $\subseteq$ Q.

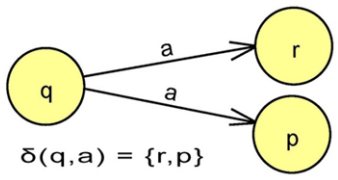

Figura 1.3: Ejemplo de transición en AFND

Ejecución de un Autómatas de estados finitos determinista AFND

Para la ejecución formal de un AFND se utiliza la definición de la función de transición extendida sobre cadenas $\widehat{\delta}$: Q x $\Sigma^* \rightarrow Q^*$, definida recursivamente así:

1. Caso base para la cadena vacía λ: $\widehat{\delta}$(q,λ) = {q}. Es desde cualquier estado del autómata la cadanea vacía λ no cambia de estado.

2. Paso recursivo para cadenas de cardinalidad mayor que 1: $\widehat{\delta}$(q,xa) = { r | para algún p en $\widehat{\delta}$(q,x), δ(p,a) = r }. Sea w de la forma w = xa donde a es el símbolo final (sufijo de longitud 1) de w y x el resto de w.
   $\widehat{\delta}$(q,x) = $\{p_1, p_2, ..., p_n\}$ sea $\bigcup_{i=1}^{k} \delta(p_i,a) = \{r_1, r_2, ..., r_n\}$
   Entonces $\widehat{\delta}$(q,x) = $\delta(\widehat{\delta}(q,x),a) = \{r_1, r_2, ..., r_n\}$, ver figura 1.4.
   La extensión para $\widehat{\delta}$ de $Q^* x \Sigma$:
   $\widehat{\delta}$((P,w) = $\bigcup_{qenP} \widehat{\delta}$(q,w), para todo conjunto de estados P en Q^*

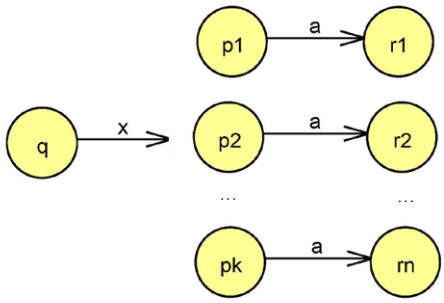

Figura 1.4: Definición de $\widehat{\delta}$ en un AFND

Cuadro 1.3: Tabla de transición: Lenguaje de las cadenas binarias que terminan en 01

δ	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
$*q_2$	ϕ	ϕ

Ejemplo: Dado el alfabeto Σ = {0,1} y el lenguaje de todas las cadenas binarias que terminan en 01, L= { x | x en Σ^* y x termina en 01 }, hallar el AFND que lo reconozca
L = { 01, 0001, 101, 1001, 10101, ... } A = (Q, Σ, δ, q_0, F), donde:
Q = $\{q_0, q_1, q_2\}$, Σ = {0,1} y F = $\{q_2\}$
En la figura 1.5 y el cuadro 1.3 se observa el AF que reconoce el lenguaje. Para las cadenas w_1 = 01

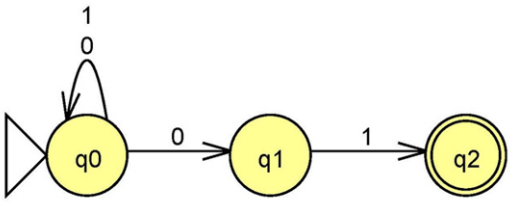

Figura 1.5: Ejemplo de AFND: cadenas binarias que terminan en 01

y w_2 = 10101 se muestra la ejecución formal a partir de la función de transición extendida $\widehat{\delta}$:

1. $\widehat{\delta}(q_0,01)$
   = $\delta(\widehat{\delta}(q_0,0),1)$
   = $\delta(\delta(\widehat{\delta}(q_0,\lambda),0),1)$
   $\rightarrow$ $\delta(\delta(\{q_0\},0),1)$
   $\rightarrow$ $\delta(\{q_0,q_1\},1) = \delta(q_0,1) \bigcup \delta(q_1,1) = \{q_0\} \bigcup \{q_2\} = \{q_0,q_2\}$, $\rightarrow$ Acepto porque $\{q_0,q_2\} \bigcap \{q_2\} \neq \phi$, es decir al menos uno de los estados del autómata es de aceptación (pertenece a F) luego de procesar la cadena.
2. $\widehat{\delta}(q_0,10101) = \delta(\widehat{\delta}(q_0,1010),1) = \delta(\delta(\delta(\delta(\widehat{\delta}(q_0,\lambda),1),0),1),0),1)$
   $\rightarrow$ $\delta(\delta(\delta(\delta(\{q_0\},1),0),1),0),1)$, $\delta(\{q_0\},1) = \{q_0\}$

$\rightarrow \delta(\delta(\delta(\delta(\{q_0\},0),1),0),1)$, $\delta(\{q_0\},0) = \{q_0, q_1\}$
$\rightarrow \delta(\delta(\delta(\{q_0, q_1\},1),0),1)$, $\delta(\{q_0, q_1\},1) = \delta(q_0,1) \bigcup \delta(q_1,1) = \{q_0\} \bigcup \{q_2\} = \{q_0, q_2\}$
$\rightarrow \delta(\delta(\{q_0, q_2\},0),1)$, $\delta(\{q_0, q_2\},0) = \delta(q_0,0) \bigcup \delta(q_2,0) = \{q_0, q_1\} \bigcup \phi = \{q_0, q_1\}$
$\rightarrow \delta(\{q_0, q_1\},1)$, $\delta(\{q_0, q_1\},1) = \delta(q_0,1) \bigcup \delta(q_1,1) = \{q_0\} \bigcup \{q_2\} = \{q_0, q_2\}$
Acepto porque $\{q_0, q_2\} \bigcap \{q_2\} \neq \phi$, es decir al menos uno de los estados es de aceptación (pertenece a F).

Lenguaje aceptado por un AFND

Dado M = (Q, Σ, δ, q_0, F) un AFND, el lenguaje L aceptado por M está definido por:

$$\boxed{\text{L(M) = \{ w | w en } \Sigma^* \text{ y } \widehat{\delta}(q_0,\text{w}) \bigcap F \neq \phi \text{ \}}}$$

Es el conjunto de cadenas pertenecientes a Σ^* tal que ejecutar el AFND desde el estado inicial q_0 con la cadena w, la función de transición extendida $\widehat{\delta}(q_0,\text{w})$ contiene al menos un estado de aceptación (es diferente del vacío la intersección con F).

Equivalencia entre AFD y AFND

Todo Lenguaje regular puede describirse con ambos tipos de autómatas AFND y AFD. Generalmente un AFND es más fácil de diseñar (capturar su modelo) y un AFD es mas fácil de implementar (ejecutar para el conocimiento de cadenas). El AFD más grande puede tener 2^n estados y el AFND más pequeño solo tiene n estados.

Teorema 1. *Si L es aceptado por un AFND entonces existe un AFD que acepta L (es decir, ambos aceptan la misma clase de lenguaje: lenguajes regulares).*

Algoritmo para la Construcción de Subconjuntos:
Este algoritmo permite a partir de un AFND obtener un AFD equivalente que reconoce el mismo lenguaje. Su nombre se debe a que se intenta construir todos los subconjuntos del conjunto de estados del AFND.
AFN N = (Q_N, Σ, δ_N, q_0 , F_N)
AFD D = (Q_D, Σ, δ_D, $\{q_0\}$, F_D)
Tal que L(N) = L(D) (el lenguaje reconocido por el autómata N sea el mismo que el lenguaje reconocido por el autómata D, por tanto sean equivalentes)
Algoritmo:

1. Σ son iguales para N y D.
2. Q_D es el conjunto de los subconjuntos de Q_N, (Conjunto potencia $Q_D = Q_N^*$). No todos los estados de Q_D son accesibles desde q_0, los estados inalcanzables desde el estado inicial serán eliminados.
3. F_D es el conjunto S de subconjuntos de Q_N (S $\subseteq$ Q_N) tal que S $\bigcap F_N \neq \phi$. Es decir F_D contiene todos los conjuntos de estados de N que incluyen al menos un estado de aceptación de N.
4. Para todo S $\subseteq$ Q_N y a en Σ, entonces δ_D(S,a) = $\bigcup_{p \epsilon S} \delta_N$(p,a).

Cuadro 1.4: Tabla de transición δ_N: Lenguaje de las cadenas binarias que terminan en 01

δ_N	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
$*q_2$	ϕ	ϕ

Cuadro 1.5: Tabla de transición δ_D a través de la construcción de subconjuntos

	δ_D	0	1
q_0	$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
q_2	$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$

Ejemplo: Dado el AFDN de la figura 1.5 y el cuadro 1.4 denotado como N = (Q_N, Σ, δ_N, q_0, F_N), con Σ= {0,1} y $Q_N = \{q_0, q_1, q_2\}$, $F_N = \{q_2\}$. Obtener D un AFD tal que L(N) = L(D) a través del algoritmo de construcción de subconjuntos.

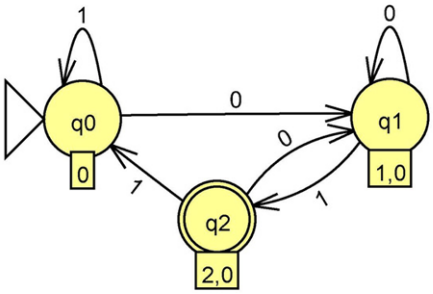

Figura 1.6: Relación AFDN y AFD: construcción de subconjunto sobre el AFND de la figura 1.5

1. Σ = {0, 1}
2. D = (Q_D, Σ, δ_D, $\{q_0\}$, F_D)
3. Q_D = $\{\phi, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$ (Conjunto de todos los subconjuntos de Q)
4. F_D = $\{\{q_2\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$
5. Generar δ_D, ver cuadro 1.5 y figura 1.6 (se renombran los estados $\{q_0\} = q_0$, $\{q_0, q_1\} = q_1$, $\{q_0, q_2\} = q_2$)
   a) $\delta_D(\{q_0\},0) = \delta_N(q_0,0) = \{q_0, q_1\}$

b) $\delta_D(\{q_0\},1) = \delta_N(q_0,1) = \{q_0\}$

c) $\delta_D(\{q_0,q_1\},0) = \delta_N(q_0,0) \bigcup \delta_N(q_1,0) = \{q_0,q_1\} \bigcup \phi = \{q_0,q_1\}$

d) $\delta_D(\{q_0,q_1\},1) = \delta_N(q_0,1) \bigcup \delta_N(q_1,1) = \{q_0\} \bigcup \{q_2\} = \{q_0,q_2\}$

e) $\delta_D(\{q_0,q_2\},0) = \delta_N(q_0,0) \bigcup \delta_N(q_2,0) = \{q_0,q_1\} \bigcup \phi = \{q_0,q_1\}$

f) $\delta_D(\{q_0,q_2\},1) = \delta_N(q_0,1) \bigcup \delta_N(q_2,1) = \{q_0\} \bigcup \phi = \{q_0\}$

A continuación se realiza la ejecución de AFD resultante para evaluar la cadena w=10101:

- $\widehat{\delta_D}(\{q_0\},10101) =$
  $\delta_D(\widehat{\delta_D}(\{q_0\},1010),1) =$
  $\delta_D(\delta_D(\delta_D(\delta_D(\widehat{\delta_D}(\{q_0\},\lambda),1),0),1),0),1)$

- $\delta_D(\delta_D(\delta_D(\delta_D(\{q_0\},1),0),1),0),1)$
  $= \delta_D(\delta_D(\delta_D(\delta_D(\delta_D(\{q_0\},1),0),1),0),1)$
  $= \delta_D(\delta_D(\delta_D(\delta_D(\{q_0\},0),1),0),1)$
  $= \delta_D(\delta_D(\delta_D(\{q_0,q_1\},1),0),1)$
  $= \delta_D(\delta_D(\{q_0,q_2\},0),1) = \delta_D(\{q_0,q_1\},1) = \{q_0,q_2\}$

- Se acepta la cadena porque $\{q_0,q_2\}$ esta en F_D

La demostración de este algoritmo se encuentra en el libro de Hopcroft [2] : Se demuestra por inducción sobre el tamaño de w, donde el caso base $\widehat{\delta_N}(q_0,\lambda) = \widehat{\delta_N}(\{q_0\},\lambda)$ y para el caso inductivo $\widehat{\delta_N}(q_0,\text{w}) = \widehat{\delta_N}(\{q_0\},\text{w})$.

1.1.3. Autómatas finitos no determinista con transiciones nulas (AFND-λ)

Es un AFND que permite transiciones para la cadena vacía λ. Antes los AF no se movían con la cadena vacía, es decir $\widehat{\delta_N}(\text{q},\lambda) = \{\text{q}\}$ y $\widehat{\delta_D}(\text{q},\lambda) = \text{q}$, los AF se quedaban en el mismo estado cuando procesan a la cadena vacía. Ahora con AFND-λ se tienen transiciones espontáneas, es decir sin recibir o procesar ningún símbolo de la entrada. Se aplica si λ no pertenece al conjunto de símbolos de Σ. Su utilidad:

- Facilidad para modelar (autómatas más sencillos y con menos estados y transiciones).
- Relación directa con las expresiones regulares (equivalencia demostrada formalmente)

Un AFND-λ es una quíntupla: A=(Q, Σ, δ, q_0, F), siendo:

Q = conjunto finito de estados.

Σ = conjunto finito de símbolos del alfabeto.

q_0 = es el estado inicial (denotado con flecha $\rightarrow$) a inicio

F = conjunto de estados finales (o estados de aceptación). F $\subseteq$ Q

δ = la función de transición entre estados δ: Q x $\Sigma \bigcup \{\lambda\} \rightarrow Q^*$.

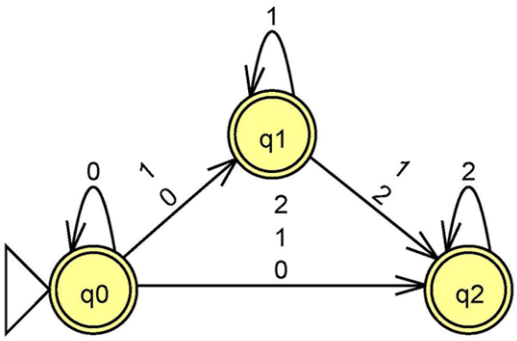

Figura 1.7: AFND para el lenguaje $0^n1^m2^p$

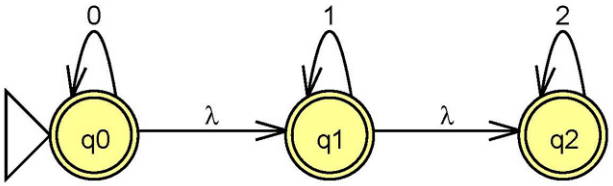

Figura 1.8: AFND-λ para el lenguaje $0^n1^m2^p$

Ejemplo: Dado L = $\{0^n1^m2^p \mid n,m,p \geq 0\}$, el lenguaje de todas las cadenas de 0, 1 o más digitos cero (0^n) seguidas de 0, 1 o más dígitos uno (1^m), y seguidas de 0, 1 o más digitos dos (2^p). Los siguientes autómatas de las figuras 1.7 y 1.8 reconocen a L:

En la tabla de transición 1.6, del AFND-λ puede o no asumirse la transición espontánea de la cadena vacía λ. Es decir, $\widehat{\delta}$(q,λ) = {q}. Por ejemplo en el estado q_2, puede ser λ pero sin embargo también es $\phi \bigcup \{q_2\}$.

Ejecución de un Autómatas de estados finitos no determinista AFND-λ

Dada una cadena w y un AFND-λ, para reconocer dicha cadena se debe aplicar la función extendida para cadenas $\widehat{\delta}$: Q x $\Sigma^* \rightarrow Q^*$. Donde $\widehat{\delta}$(q,w) es el conjunto de todos los estados p_i tal que se puede ir de q a cada p_i por un camino etiquetado w que puede incluir arcos etiquetados λ. Para calcular este camino es necesario definir el término de la λ-clausura(q) de un estado q (q en Q) como, el conjunto de todos los estados a los cuales se puede llegar a partir de q siguiendo un camino de arcos únicamente

Cuadro 1.6: Tabla de transición de un AFND-λ

δ	0	1	2	λ
$\rightarrow q_0$	$\{q_0\}$	ϕ	ϕ	$\{q_1\}$
q_1	ϕ	$\{q_1\}$	ϕ	$\{q_2\}$
* q_2	ϕ	ϕ	$\{q_2\}$	ϕ

etiquetados con λ.
La definición recursiva de λ-clausura(q):

- Base: λ-clausura(q) = {q} , es decir por definición $\widehat{\delta}$(q,λ) = {q}, podemos imaginarnos un bucle de un estado a si mismo etiquetado λ. Ver figura 1.9.
- Paso inductivo: Si $\widehat{\delta}$ es la función de transición de AFND-λ y p está en la λ-clausura(q), es decir $\widehat{\delta}$(q, λ) = {p} entonces la λ-clausura(q) también contiene los estados $\widehat{\delta}$(p,λ). Ver figura 1.9.

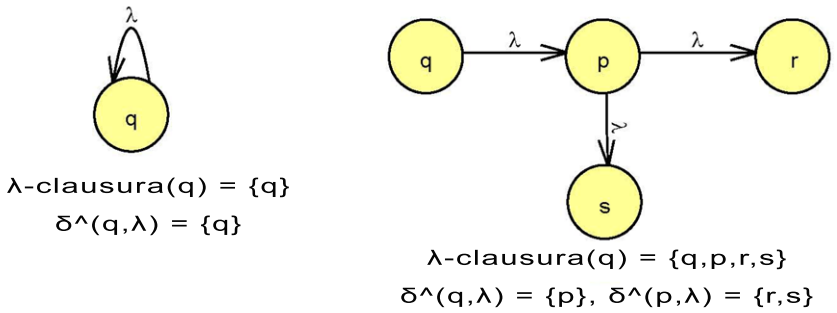

Figura 1.9: Ejemplo de la definición de λ-clausura en un AFND-λ

La definición de la función de transición extendida $\widehat{\delta}$ de los AFND-λ, con la que se pueden ejecutar este tipo de autómata, es la siguiente:

1. $\widehat{\delta}(q_0, \lambda)$ = λ-clausura(q_0)
2. Para todo w en Σ^* y a en Σ, $\widehat{\delta}$(q,wa) = λ-clausura(P) con P $\subseteq$ Q, Donde:
   - λ-clausura(P) = { p | para algún r en $\widehat{\delta}$(q,w), p en δ(r,a) } , Aplicar λ-clausura a cada p en P.

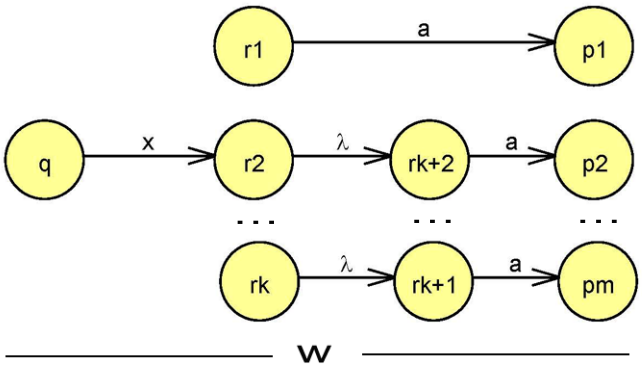

Figura 1.10: Ejemplo de la definición de λ-clausura en un AFND-λ en caso inductivo

3. Como aplicar la función δ a un conjunto de estados R. δ(R,a) = $\bigcup_{qenR}$ δ(q,a), R $\subseteq$ Q

4. Como aplicar la función $\widehat{\delta}$ a un conjunto de estados R. $\widehat{\delta}$(R,a) = $\bigcup_{qenR} \widehat{\delta}$(q,a), R $\subseteq$ Q.
   Entonces se tienen las siguientes definiciones $\widehat{\delta}$(q,w) = $\widehat{\delta}$(q,x) = $\{r_1, r_2, ..., r_k\}$ con w = xa.
   $\bigcup_{i=1}^{k} \delta(r_i$,a) = $\{p_1, p_2, ..., p_m\}$ y $\widehat{\delta}$(q,w) = $\bigcup_{j=1}^{m}$ λ-clausura(p_j). Ver figura 1.10.

Ejemplo: Dada la cadena w = 02 determinar si pertenece o no al lenguaje $0^n1^m2^p$ reconocido por el AFND-λ de la figura 1.8
Definición de la λ-clausura de los estados del autómata:

- λ-clausura(q_0) = $\{q_0, q_1, q_2\}$
- λ-clausura(q_1) = $\{q_1, q_2\}$
- λ-clausura(q_2) = $\{q_2\}$

Aplicando la función de transición extendida $\widehat{\delta}$:

- $\widehat{\delta}(q_0$,02) = λ-clausura($\delta(\widehat{\delta}(q_0$,0),2)) = λ-clausura(δ(λ-clausura($\delta(\widehat{\delta}$(**q_0,λ**),0)),2))
- λ-clausura(δ(λ-clausura(δ(**{**q_0, q_1, q_2**},0)**),2)) $\rightarrow$ $\widehat{\delta}(q_0, \lambda)$ = λ-clausura(q_0) = $\{q_0, q_1, q_2\}$
- λ-clausura(δ(λ-**clausura(**{q_0}),2)) $\rightarrow$ $\delta(\{q_0, q_1, q_2\}$,0) = $\{q_0\} \bigcup \phi \bigcup \phi = \{q_0\}$
- = λ-clausura(δ(**{**q_0, q_1, q_2**},2)**) $\rightarrow$ λ-clausura($\{q_0\}$) = λ-clausura(q_0) = $\{q_0, q_1, q_2\}$
- λ-**clausura(**q_2**)** $\rightarrow$ $\delta(\{q_0, q_1, q_2\}$,2) = $\phi \bigcup \phi \bigcup \{q_2\} = \{q_2\}$
- $\{q_2\}$ Acepto la cadena w = 02 ya que $\widehat{\delta}(q_0$,02) $\bigcap$ F = $\{q_2\} \neq \phi$.

Lenguaje aceptado por un AFND-λ

Dado un AFND-λ E = (Q, Σ, δ, q_0, F) el L(E) se define como:

$$\boxed{\text{L(E)= \{ w | w en } \Sigma^*, \widehat{\delta}(q_0,\text{w}) \bigcap \text{F} \neq \phi \}}$$

El lenguaje L(E), el reconocido por el autómata E, es el conjunto de cadenas de Σ^* que al aplicar la función de transición extendida $\widehat{\delta}$ se obtiene un conjunto de estados de Q donde al menos uno pertenece al conjunto de estados de aceptación (F).

Equivalencia entre AFD y AFND-λ (construcción de subconjuntos de λ)

Teorema 2. *Si L es aceptado por un AFND con λ-transiciones entonces L es aceptado por un AFND sin λ-transiciones.*

Teorema 3. *Un lenguaje regular L es aceptado por un AFND con λ-transiciones si y sólo si es aceptado por un AFD.*

A partir de estos teoremas se deriva un algoritmo para pasar de un AFND-λ a un AFD, es decir para eliminar las transiciones λ y obtener un autómata equivalente que reconozca el mismo lenguaje de forma deterministica.
Dado E un AFND-λ se quiere lograr que L(E) = L(A), con y A un AFD, es decir $\widehat{\delta_E}(q_{0E}$,w) = $\widehat{\delta_D}(q_{0D}$,w) para todo w en L:

- E (AFND-λ) = $(Q_E, \Sigma, \delta_E, q_{0E}, F_E)$
- D (AFD) = $(Q_D, \Sigma, \delta_D, q_{0D}, F_D)$

Algoritmo:

1. q_{0D} = λ-clausura(q_{0E})
2. $Q_D = Q_E^*$, es decir el conjunto de subconjuntos de Q_E
3. F_D = { s | s en Q_D y {s} $\bigcap$ $F_E \neq \phi$ }
4. δ_D(S,a) para todo a en Σ y S en Q_D, se cumple:
   δ_D(S,a) = $\bigcup_{j=1}^{n}$ λ-clausura(r_j),
   $\{r_1, r_2, ..., r_n\}$ = $\bigcup_{i=1}^{k}$ δ_E(p_i,a) con S = $\{p_1, p_2, ..., p_k\}$. Ver figura 1.11

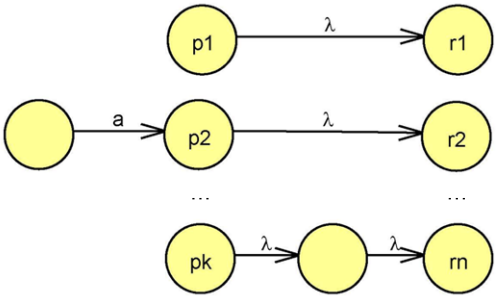

Figura 1.11: Construcción de subconjuntos λ en un AFND-λ

Ejemplo: Dado el AFND-λ E de la figura 1.8, encontrar un AFD equivalente que reconozca el mismo lenguaje.
E = (Q, Σ, δ, q_0, F), donde Q = $\{q_0, q_1, q_2\}$, Σ = {0, 1, 2} y F = $\{q_2\}$. Se obtiene el AFD D = (Q_D, Σ, δ_D, q_{0D}, F_D), aplicando el algorimos de construcción de subconjuntos de λ:

1. Σ = {0, 1, 2}
2. q_{0D} = λ-clausura(q_0) = $\{q_0, q_1, q_2\}$
3. Q_D = $\{\phi, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_1, q_2\}, \{q_0, q_2\}, \{q_0, q_1, q_2\}\}$
4. F_D = $\{\{q_2\}, \{q_1, q_2\}, \{q_0, q_2\}, q_0, q_1, q_2\}$
5. Cálculo de δ_D: (ver cuadro 1.7)
   - Para el estado inicial $\{q_0, q_1, q_2\}$ se aplica la definición para todos los elementos de Σ:
     * $\delta_D(\{q_0, q_1, q_2\},0)$ = λ-clausura($\delta_E(q_0, 0) \bigcup \delta_E(q_1, 0) \bigcup \delta_E(q_2, 0)$) = λ-clausura($\{q_0\} \bigcup \phi \bigcup \phi$)
     = λ-clausura($\{q_0\}$) = λ-clausura(q_0) = $\{q_0, q_1, q_2\}$
     * $\delta_D(\{q_0, q_1, q_2\},1)$ = λ-clausura($\delta_E(q_0, 1) \bigcup \delta_E(q_1, 1) \bigcup \delta_E(q_2, 1)$) = λ-clausura($\phi \bigcup \{q_1\} \bigcup \phi$)
     = λ-clausura($\{q_1\}$) = λ-clausura(q_1) = $\{q_1, q_2\}$
     * $\delta_D(\{q_0, q_1, q_2\},2)$ = λ-clausura($\delta_E(q_0, 2) \bigcup \delta_E(q_1, 2) \bigcup \delta_E(q_2, 2)$) = λ-clausura($\phi \bigcup \phi \bigcup \{q_2\}$)
     = λ-clausura($\{q_2\}$) = λ-clausura(q_2) = $\{q_2\}$

Cuadro 1.7: Tabla de transición δ_D a través de la construcción de sunconjuntos de λ

	δ_D	0	1	2
q_0	$\rightarrow \{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	$\{q_1, q_2\}$	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
q_2	$*\{q_2\}$	ϕ	ϕ	$\{q_2\}$

- Para el estado $\{q_1, q_2\}$ se aplica la definición para todos los elementos de Σ:
  * $\delta_D(\{q_1, q_2\},0)$ = λ-clausura($\delta_E(q_1, 0) \bigcup \delta_E(q_2, 0)$) = λ-clausura($\phi \bigcup \phi$) = λ-clausura(ϕ) = ϕ
  * $\delta_D(\{q_1, q_2\},1)$ = λ-clausura($\delta_E(q_1, 1) \bigcup \delta_E(q_2, 1)$) = λ-clausura($\{q_1\} \bigcup \phi$) = λ-clausura($\{q_1\}$) = λ-clausura(q_1) = $\{q_1, q_2\}$
  * $\delta_D(\{q_1, q_2\},2)$ = λ-clausura($\delta_E(q_1, 2) \bigcup \delta_E(q_2, 2)$) = λ-clausura($\phi \bigcup \{q_2\}$) = λ-clausura($\{q_2\}$) = λ-clausura(q_2) = $\{q_2\}$
- Para el estado $\{q_2\}$ se aplica la definición para todos los elementos de Σ:
  * $\delta_D(\{q_2\},0)$ = λ-clausura($\delta_E(q_2, 0)$) = λ-clausura(ϕ) = ϕ
  * $\delta_D(\{q_2\},1)$ = λ-clausura($\delta_E(q_2, 1)$) = λ-clausura(ϕ) = ϕ
  * $\delta_D(\{q_2\},2)$ = λ-clausura($\delta_E(q_2, 2)$) = λ-clausura($\{q_2\}$) = λ-clausura(q_2) = $\{q_2\}$

Hay dos formas de interpretar la construcción de subconjuntos-λ, una para crear un AFD y otra AFND. La idea es llegar a un AF más fácil de implementar (es decir obtener un AFD).

- AFD: Se toma cada estado como un conjunto de etiquetas. Ver figura 1.12
- AFND: Se interpreta el contenido de la etiqueta de cada estado. Ver figura 1.13

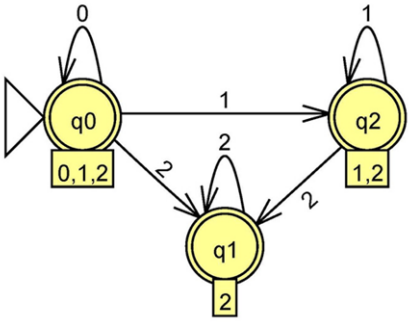

Figura 1.12: AFD equivalente del AFND-λ de la figura 1.8 a través de la Construcción de subconjuntos λ

1.1.4. Herramientas, implementación y uso de Lenguajes regulares

Implementación de un AFD: programación basada en autómatas

La simulación de un AFD resulta un enfoque de programación que permite aumentar el nivel de abstracción ante la solución de un problema. El siguiente algoritmo (no recursivo) expresado en

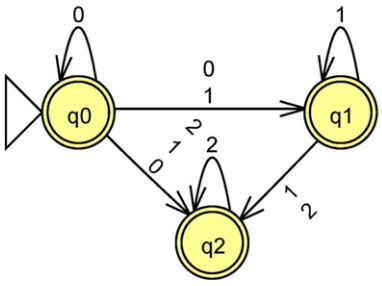

Figura 1.13: AFND equivalente del AFND-λ de la figura 1.8 a través de la Construcción de subconjuntos λ

pseudocódigo muestra el funcionamiento del AFD de la figura 1.2:

INICIO

```
Conjunto Q[Estado] = {q0, q1, q2}
Conjunto Σ[Simbolo]= {0,1}
Conjunto F[Estado] = {q1}
Estado Actual = q0
Simbolo A
A = LeerSimboloCadena()
Mientras a ≠ λ entonces
   Actual = transicionδ(Actual, A)
   A = LeerSimboloCadena()
fin Mientras
Si Pertenece(Actual,F) entonces
   Imprimir("Se acepta la cadena")
sino
   Imprimir("No se acepta la cadena")
fin Si
```

FIN

```
Función transicionδ(Estado q, Símbolo a): Estado
```

```
Estado S
Selección
```

- q = q_0 y a = 0: S = q_1
- q = q_0 y a = 1: S = q_2
- q = q_1 y a = 0: S = q_1
- q = q_1 y a = 1: S = q_1
- q = q_2 y a = 0: S = q_2
- q = q_2 y a = 1: S = q_2

```
        fin Selección
        Retornar S

    fin Función
```

Donde se asume que se tiene los siguientes tipos de datos: `Conjunto` (paramétrico por el tipo de elemento), `Estado`, `Simbolo`. Para ellos se tiene las siguientes operaciones:

- `LeerSimboloCadena()` que permite obtener desde la entrada estándar un `Símbolo` de la cadena de entrada w (desde el comienzo, secuencialmente uno a uno)
- `Imprimir(CadenaCaracter)` función que permite imprimir la salida del autómata, si acepta o no la cadena de entrada
- `Pertenece(Estado A, Conjunto B[Estado])`, es una operación del tipo de dato Conjunto que determina si el elemento A (del tipo Estado) pertenece al Conjunto de Estados B.

Una posible implementación del algoritmo anterior en lenguaje C se muestra a continuación:

```
#include <iostream>
#include <string>
#include <stdio.h>
using namespace std;
enum estados { q0, q1, q2 };
void transicion(enum estados *estado, char a) // funci\'on de transici\'on
{
switch(*estado) {
case q0:
if(a == '0') {
putchar(a);
*estado = q1;
}
else {
if(a == '1') {
putchar(a);
*estado = q2;
}
}
break;
case q1:
if(a == '0' || a == '1') {
putchar(a);
*estado = q1;
}
break;
case q2:
if(a == '0' || a == '1') {
```

```
putchar(a);
*estado = q2;
}
break;
}
}
int main()
{
char c;
string cad;
enum estados state = q0;
cout << "AFD que reconoce todas las cadena binarias que comienzan con 0" << endl;
cout << "Estado inicial: " << state << endl;
c = getchar();
cad = c;
while((c == '0') || (c == '1')) {
transicion(&state, c);
cout << " Estado actual: " << state << endl;
c = getchar();
if ((c == '0') || (c == '1')) cad = cad + c;
}
if (state == q1) cout << "La cadena " << cad << " SI esta en el lenguaje" << endl;
else cout << "La cadena " << cad << " NO esta en el lenguaje" << endl;
return 0;
}
```

1.1.5. Preguntas y respuestas, ejercicios resueltos y propuestos

Preguntas y respuestas

1. ¿Qué es un autómata?. Es una máquina abstracta, dispositivo teórico o modelo matemático que procesa una secuencia finita de símbolos de un alfabeto, cambiando su estado de acuerdo al símbolo que está leyendo o procesando, lo que permite saber si una cadena se encuentra dentro o no de un lenguaje dado. Tiene un conjunto finitos de estados posibles y muestra las posibles transiciones entre los estados.

2. ¿Por qué en los AF se tiene siempre un sólo estado de inicial y uno o más estados finales?. El objetivo de una AF es aceptar cadenas de un lenguaje y para eso necesita estados de aceptacón, por tanto el conjunto F no puede ser vacío, pero si puede tener uno o más caminos para reconocer esas cadenas (tener mas de un estado en F). Con respecto al estado inicial se necesita sólo uno, pues el AF necesita saber de forma precisa por donde comenzar el funcionamiento de la máquina. Si se tiene más de un estado inicial el AF debe seleccionar por donde comenzar, lo cual resultaría complicado para un modelo matemático sencillo. De forma análoga ocurre en los lenguajes de programación donde sólo hay un inicio para el programa (main) pero puede tener mas de un final (exit).

3. ¿Por qué un AF debe terminar de procesar la cadena de entrada para saber si está o no en el lenguaje?. La mayoría de los autómatas o máquinas de reconocimiento (excepto la Máquina de

Turing) tienen esta restricción que les permite simplificar el funcionamiento de su modelo. Si bien es cierto que para algunos lenguajes no hace falta recorrer toda la cadena pues su definición no lo requiere (por ejemplo puede estar relacionada sólo con el prefijo de las cadenas), hacer que estos casos funcionen como los demás es lo más simple y general en un modelo matemático.

4. ¿Por qué el conjunto de estados de un AF es finito? Una de las características de los autómatas finitos es que el conjunto de estados es finito. Esto se debe a una propiedad matemática sobre los lenguajes regulares (relación de equivalencia) que establece particiones finitas sobre el lenguaje (aunque el lenguaje puede ser infinito). Esta propiedad es diferente en otros tipos de lenguajes más complejos pues las particiones se hacen infinitas y por ende el número de estados para representarlas.

5. ¿Puede obtener un AFND algorítmicamente a partir de un AFD?, ¿por qué? El procedimiento de la construcción de subconjuntos permite llevar de un AFND a AFD, es decir eliminar el no determinismo. El no determinismo implica el manejo de opciones o alternativas en cada paso de transición generando instancias simultáneas del Autómata en cada alternativa. El caso contrario parte de un AFD y se quiere obtener un AFND, es decir agregar no determinismo al autómata. Este proceso no tiene un algortimo, pero si dicho algoritmo pudiera existir tendría que ser en si no determinístico, es decir aplicar alternativas finitas en cada paso de transición que incluso podría no tener sentido, sería muy complejo e incluso dificil de aplicar en algunos lenguajes regulares.

6. ¿Qué significa la equivalencia de autómatas?. Dos autómatas son equivalente cuando reconocen exactamente el mismo lenguaje. Es necesario distinguir entre ser iguales y ser equivalente. Dos autómatas diferentes pueden tener diferente cantidad de estados, con diferentes transiciones e incluso diferente cantidad de estados de aceptaciión, es decir son diferentes en todo sentido y por tanto no son iguales. Sin embargo, estos autómatas diferentes, $M_1 \neq M_2$, son capaces de procesar cadenas de Σ^* y reconocer el mismo lenguaje L (todas y cada una de las cadenas que forman a L, ni una más ni una menos), en ese caso son llamados autómatas equivalentes, es decir $L(M_1) = L(M_2)$.

Ejercicios resueltos

1. Determine las diferencias entre un AFD y un AFND. Ver el cuadro 1.8

2. Aplicaciones de la vida real que pueden ser vistos como AF. Algunas aplicaciones interesantes de los AF están referidas al almacenamiento de diccionarios como los usados en los celulares, en la que se utilizan sus características para hacer búsquedas, inserciones o modificaciones rápidas. Así como también para mejorar el ordenamiento y búsqueda en otros tipos de aplicaciones como los circuitos digitales, texto, secuencias lógicas de funcionamiento (como la maquinaria industrial, robótica) y otras. También son útiles para el modelado de aquellos casos en los que a partir de una acción anterior se puede llegar a un conjunto determinado de destinos, pero no se tiene certeza anticipada de cuál de ellos es el indicado, utilizados para modelar protocolos de comunicación, sistemas distribuidos, planificación, etc.

3. Modelar un AF que reconozca un identificador válido en el lenguaje de programación C.
Un identificador es un nombre que define a una variable, una función o un tipo de datos. Un

Cuadro 1.8: Tabla comparativa de AFD y AFND

Característica	AFD	AFND
Función de transición δ	δ: Q x $\Sigma \rightarrow$ Q, rango es un elemento de Q	δ: Q x $\Sigma \rightarrow Q^*$, rango es un subconjunto de Q
Definición del Lenguaje que reconoce	L(M)={w|w ϵ Σ^* y $\widehat{\delta}$(q$_0$,w) ϵ F }, Lenguaje regular L reconocido por el AFD M	L(M)={w|w ϵ Σ^* y $\widehat{\delta}(q_0$,w) $\bigcap F \neq \phi$}, Lenguaje regular L reconocido por el AFND M
Estado actual	Un solo estado actual en cada paso del autómata	El estado actual puede ser un conjunto de estados (subconjunto de Q), es deir múltiples estados simultaneamente
Condición del diagrama de transición	Desde cada estado existe una transición de salida por cada símbolo de Σ	No hay restricciones con respecto a las transiciones, un símbolo de Σ puede tener 0, 1 o más transciones de salida desde el mismo estado
Condición de la tabla de transición	Cada posición de la celda de la tabla debe tener sólo un estado (estar completa)	Las posiciones de las celdas pueden tener cualquier subconjunto de Q (incluyendo el ϕ).
Utilidad	Los AFD son los que se implementan para el reconocimiento de lenguajes regulares, se usan para simular AFND	Los AFND se utilizan en lenguajes regulares con alfabetos grandes y muchos estados, pues es más sencillo de modelar, tiene relación directa con las expresiones regulares y son utiles para aplicar operaciones sobre autómatas
Facilidad de modelado y programación	El modelo de un AFD deben considerar todo Σ en cada estado, esto lo hace exhaustivo y difícil de diseñar. En cambio su programación o implementación es un algortimo determinístico (tiene definido que hacer en cada caso) eficiente	El modelo de un AFND es sencillo pues no exige evaluar todo Σ en cada estado, al contrario puede considerar solo las opciones que aceptan las cadenas y muestra un diseño más sencillo. Por el contrario, su implementación requiere simular las alternativas simultáneas en cada paso de transición (búsqueda en un árbol realizando backtracking) y puede hacer complejo y poco eficiente su programación e implementación

identificador válido ha de empezar por una letra o por el carácer de subrayado _, seguido de cualquier cantidad de letras, dígitos o subrayados. Además: No debe contener caracteres especiales, tales como @, $, #, no debe haber espacios en blanco en los identificadores.
El AF que reconoce este lenguaje es el siguiente:
A = (Q, Σ, δ, q_0, F), donde Q = $\{q_0, q_1\}$, Σ son todos los caracteres alfanuméricos y caracteres especiales permitidos (tales como -,_,.) , F = $\{q_1\}$, y δ se expresa en el diagrama de transición de la figura 1.14 , donde se observa dos conjuntos de caracteres:

- A = conjunto de caracteres alfabéticos
- B = conjunto de caracteres alfabéticos, numéricos y subrayados

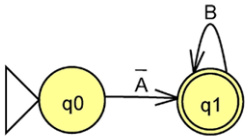

Figura 1.14: AF que reconoce un identificador en el lenguaje C

Existen palabras propias del lenguaje (palabras reservadas) que no pueden ser usadas como identificadores ej: if, do, while, etc. Cómo puede modificar el AF anterior para no considerar como identificadores a las palabras reservadas del lenguaje.

4. Modelar un AF que reconozca el conjunto de enteros y números en punto flotante o también llamadas "literales numéricas".
   En el lenguaje C, las siguientes expresiones son *literales numéricas* válidas: 3, +0, -2E+7, 13., -01, -14.4, 41.16, -.4E-7, 0.2E-20, 00E0, 01E-06. La letra E se refiere a un exponente y su presencia hace que el número subsiguiente sea un entero. Se supone que no existe límite en el número de dígitos consecutivos en ninguna parte de la expresión (no hay el problema de la precisión).
   El AF que reconoce este lenguajs puede ser el siguiente: B = (Q, Σ, δ, q_0, F), donde Q = $\{q_i \mid 0 \leq i \leq 7\}$, Σ son todos los caracteres numéricos y caracteres especiales permitidos (tales como .,+,-,E), F = $\{q_1, q_4, q_5, q_6\}$, y δ se expresa en el diagrama de transición de la figura 1.15, donde se observa el conjunto de caracteres N = {0,1,2,3,4,5,6,7,8,9}.

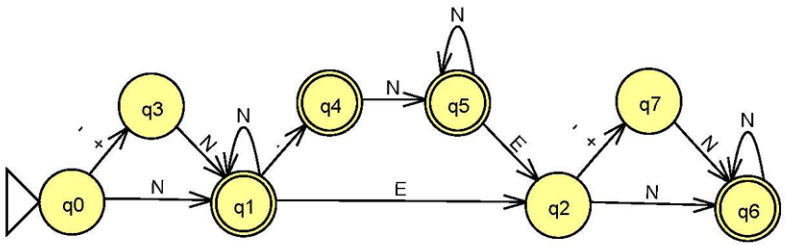

Figura 1.15: AF que reconoce un literal numérico en el lenguaje C

Se plantea como ejercicio pensar cómo puede modificar el AF de la figura 1.15 para reconocer lo siguiente: cambiar el uso del punto para que en su lugar la coma (,) se utilice para separar enteros y decimales, y el punto (.) se emplee como separador de las unidades decimales (comenzando desde el lado derecho de tres en tres). Por ejemplo, partes enteras de las literales numéricas como: 1.300 o 35.413 o 38 o 23.300.080, etc.

Ejercicios propuestos

1. Para Σ = {0,1} construya un AFD que acepte los siguientes lenguajes:

   a) El conjunto de cadenas que tiene 011 como subcadena.

   b) El conjunto de todas las cadenas cuyo tercer elemento desde el extremo izquierdo sea un 1.

   c) El conjunto de cadenas en las que el número de ceros es divisible por 3.

   d) El conjunto de cadenas que terminan con 01.

   e) El conjunto vacío.

   f) El conjunto de la cadena vacía.

2. Constuya un AF para los siguientes lenguajes:

   a) El conjunto de cadenas basadas en Σ={a,b,c} cuyo último símbolo no haya aparecido antes en la misma entrada.

   b) Conjunto de 0 y 1 que contenga 2 ceros separados por un número de posiciones múltiplo de 3. Nótese que 0 es un múltiplo de 3 válido.

3. Obtenga un autómata finito (de cualquier tipo) para reconocer los siguientes lenguajes apartir del alfabeto binario:

   a) cadenas acabadas en 00

   b) cadenas con dos unos consecutivos

   c) cadenas que no contengan dos unos consecutivos

   d) cadenas con dos ceros consecutivos o dos unos consecutivos

   e) cadenas con dos ceros consecutivos y dos unos consecutivos

   f) cadenas acabadas en 00 o 11

   g) cadenas con un 1 en la antepenúltima posición

   h) cadenas de longitud 4

4. Construir un AFD equivalente a los siguientes autómatas:

   a) M_1 = ({p, q, r, s}, {0, 1}, δ, p, {s}) donde δ está dada por el cuadro 1.9

   b) M_2 = ({p, q, r, s}, {0, 1}, δ, p, {q, s}) donde δ está dada por el cuadro 1.10

   c) M_3 = ({p, q, r}, {a, b, c}, δ, p, {r}) donde δ está dada por el cuadro 1.11

   d) M_4 = ({q_0, q_1, q_2, q_3, q_4}, {a, b}, δ, q_0, {q_4}) donde δ está dada por el cuadro 1.12

Cuadro 1.9: Tabla de transición δ

δ	0	1
$\rightarrow$ p	{p, q}	{p}
q	{r}	{r}
r	{s}	ϕ
*s	{s}	{s}

Cuadro 1.10: Tabla de transición δ

δ	0	1
$\rightarrow$ p	{q, s}	{q}
*q	{r}	{q, r}
r	{s}	{p}
*s	ϕ	{p}

Cuadro 1.11: Tabla de transición δ

δ	λ	a	b	c
$\rightarrow$ p	ϕ	{p}	{q}	{r}
q	{p}	{q}	{r}	ϕ
*r	{q}	{r}	ϕ	{p}

Cuadro 1.12: Tabla de transición δ

δ	a	b	λ
$\rightarrow q_0$	$\{q_1, q_2\}$	$\{q_1, q_3\}$	ϕ
q_1	ϕ	ϕ	$\{q_3\}$
q_2	$\{q_4\}$	ϕ	ϕ
q_3	ϕ	$\{q_4\}$	ϕ
q_4	ϕ	ϕ	ϕ

Bibliografía

[1] Alonzo Church: A note on the Entscheidungsproblem. Journal of Symbolic Logic. 1 (1936). pp 40 - 41.

[2] Hopcroft, Rajeev Motwani, Jeffrey D. Ullman. Introducción a la Teoría de Autómatas y Lenguajes Formales. PrenticeHall, 2002.

[3] Chacín, M. y Padrón, J. (1994) Qué es teoría. Investigación y Docencia. Caracas: Publicaciones del Decanato de Postgrado, USR.

[4] Seymour Lipschutz (1970) Teoría de Conjuntos y Temas afines. macGraw-Hill