Conceptos fundamentales

Alfabetos

Conjunto de símbolos **finito y no vacío**.

Convencionalmente, utilizamos el símbolo ∑ para designar un alfabeto.

Cadenas de caracteres

Una cadena de caracteres es una secuencia finita de símbolos seleccionados de algún alfabeto.

La cadena vacía

Representa cero apariciones de símbolos. Esta cadena **ε** es una cadena que puede construirse en cualquier alfabeto.

Longitud de una cadena

La notación estándar para indicar la longitud de una cadena w es $|w|$.
Por ejemplo, $|011| = 3$

Potencias de un alfabeto

Definimos para que sea el conjunto de las cadenas de longitud k, tales que cada uno de los símbolos de las mismas pertenece a $\sum$

Observe que $\sum^0 = \varepsilon$ independientemente de cuál sea el alfabeto $\sum$
Es decir, ε es la única cadena cuya longitud es 0.

Si $\sum = \{0,1\}$, entonces:

$\sum^0 = \varepsilon$
$\sum^1 = \{0,1\}$
$\sum^2 = \{00,01,10,11\}$
$\sum^3 = \{000,001,010,011,100,101,110,111\}$

Todas las cadenas de un alfabeto $\sum$ se designa mediante

$\sum^*$: Por ejemplo:

$$\sum^* = \{ \varepsilon ,0,1,00,01,10,11,000, \ldots \}$$

De otra forma:

$$\sum^* = \sum^0 \cup \sum^1 \cup \sum^2 \cup \cdots$$

Si desearemos excluir la cadena vacía **ε** del conjunto de cadenas. Este conjunto designa como $\sum+$.

$$\sum+ = \sum^1 \cup \sum^2 \cup \cdots$$
$$\sum^* = \sum+ \cup \{ \boldsymbol{\varepsilon} \}.$$

Concatenación de cadenas

Si x e y son dos cadenas. Entonces, xy es la concatenación de x e y, es decir, la cadena formada por una copia de x seguida de una copia de y.

- $x = a_1a_2 \cdot \cdot \cdot a_i$ e $y = b_1b_2 \cdot \cdot \cdot b_j$, entonces xy es la cadena de longitud
- $i + j$: $xy = a_1a_2 \cdot \cdot \cdot a_ib_1b_2 \cdot \cdot \cdot b_j$.

Potencia (cadenas)

$$w^n = \begin{cases} \varepsilon & n = 0 \\ w w^{n-1} & n > 0 \end{cases}$$

Si w = 122 sobre el alfabeto Σ = { 1, 2 }

$w^0 = \varepsilon$

$w^1 = 122$

$w^2 = 122122$

$w^3 = 122122122$

LENGUAJES

Un conjunto de cadenas, todas ellas seleccionadas de un Σ^* , donde $\sum$ es un determinado alfabeto se denomina lenguaje.

Si $\sum$ es un alfabeto y L pertenece a Σ^* , entonces L es un lenguaje de $\sum$

Un ejemplo es el lenguaje C, o cualquier otro lenguaje de programación, donde los programas correctos son un subconjunto de las posibles cadenas que pueden formarse a partir del alfabeto del lenguaje

Existen también otros muchos lenguajes

Algunos ejemplos son los siguientes:

- El lenguaje de todas las cadenas que constan de n ceros seguidos de n unos para cualquier n ≥ 0:

  {∑,01,0011,000111, . . .}.
- El conjunto de cadenas formadas por el mismo número de ceros que de unos:

  {∑,01,10,0011,0101,1001, . . .}
- El conjunto de números binarios cuyo valor es un número primo:

  {10,11,101,111,1011, . . .}
- Σ* es un lenguaje para cualquier alfabeto ∑.
- φ, el lenguaje vacío, es un lenguaje de cualquier alfabeto.
- { ε }, el lenguaje que consta sólo de la cadena vacía, también es un lenguaje de cualquier alfabeto.

φ ≠ { ε }, el primero no contiene ninguna cadena y el segundo sólo tiene una cadena.

Operaciones con Lenguajes

Sean A y B dos lenguajes se pueden definir las siguientes operaciones:

- Unión:

$$A \cup B = \{ x \mid x \text{ está en A o x está en B } \}$$

- Intersección:

$$A \cap B = \{ x \mid x \text{ está en A y x está en B simultáneamente } \}$$

- Concatenación:

$$A.B = \{ w.x \mid w \text{ está en A y x está en B } \}$$

Si A es un lenguaje sobre Σ_1
y B es un lenguaje sobre Σ_2
A.B sera un lenguaje sobre $\Sigma_1 \cup \Sigma_2$

- Potencia:

$$A^n = \begin{cases} \varepsilon & n = 0 \\ A.A^{n-1} & n >= 1 \end{cases}$$

Si A = {ab}

$A^0 = \varepsilon$

$A^1 = A = \{ab\}$

$A^2 = A.\ A^1 = \{abab\}$

$A^3 = A.\ A^2 = \{ababab\}$

- Cerradura de Kleene:

Es el lenguaje **A** compuesto por todas las cadena sobre un alfabeto dado **Σ**

Ejemplo: Σ = {1}

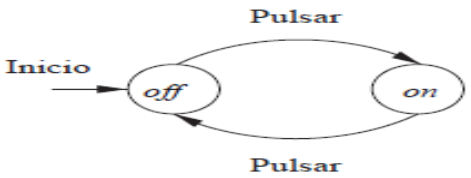
A* = { ε, 1, 11, 111, 1111, …. }

A* es infinito

$$A^* = \sum_{0}^{\infty} A^n$$

Autómatas Finitos

- Los autómatas finitos, o máquinas de estados finitos, son una manera matemática para describir clases particulares de algoritmos (o "máquinas").



- Se pueden utilizar para describir el proceso de reconocimiento de patrones en cadenas de entrada, y de este modo construir analizadores léxicos.

- Existe una fuerte relación entre los autómatas finitos y las expresiones regulares

- Antes de comenzar nuestro estudio de los autómatas finitos, veamos unos ejemplos

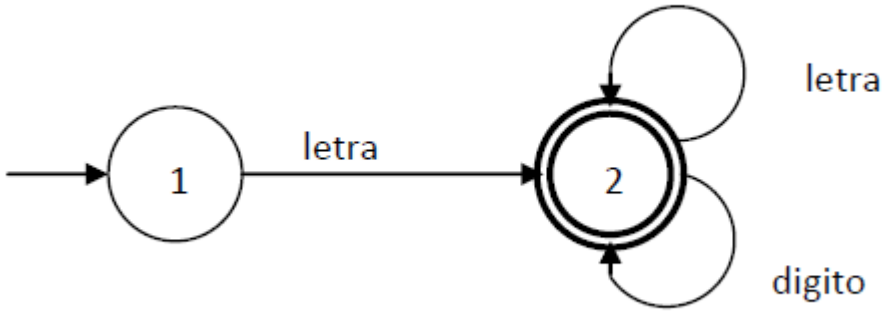
El patrón para ***identificadores*** como se define comúnmente en los lenguajes de programación está dado por la siguiente definición regular (supondremos que letra y dígito ya se definieron):

identificador = letra(letra | dígito)*

Esto representa una cadena que comienza con una letra y continúa con cualquier secuencia de letras y/o dígitos.

Esto se puede describir mediante el diagrama de la figura

En este diagrama los círculos numerados 1 y 2 representan estados. Las líneas con flechas representan transiciones que registran un cambio de un estado a otro en una coincidencia del carácter o caracteres mediante los cuales son etiquetadas.



Otro ejemplo de un problema real, cuya solución emplea autómatas finitos que desempeñan un importante papel.

Vamos a investigar protocolos que ayudan a gestionar el “dinero electrónico” (los archivos que un cliente puede utilizar para realizar pagos por bienes a través de Internet, y que el vendedor puede recibir con la seguridad de que el “dinero” es real).

Veamos un ejemplo muy simple de un (pobre) protocolo de dinero electrónico, modelado mediante un autómata finito y vamos a mostrar cómo pueden utilizarse las construcciones sobre autómatas para verificar los protocolos (o, como en este caso, para descubrir que el protocolo tiene un error).

- Tenemos tres participantes: **el cliente, la tienda y el banco**. Para simplificar, suponemos que sólo existe un archivo de "dinero electrónico".

- El cliente puede decidir transferir este archivo a la tienda, la cual lo reenviará al banco (es decir, solicita al banco que emita un nuevo archivo que refleje que el dinero pertenece a la tienda en lugar de al cliente) y suministra los bienes al cliente.

- Además, el cliente tiene la opción de cancelar el archivo; es decir, el cliente puede pedir al banco que devuelva el dinero a su cuenta, anulando la posibilidad de gastarlo.

La interacción entre los tres participantes se limita por tanto a cinco sucesos:

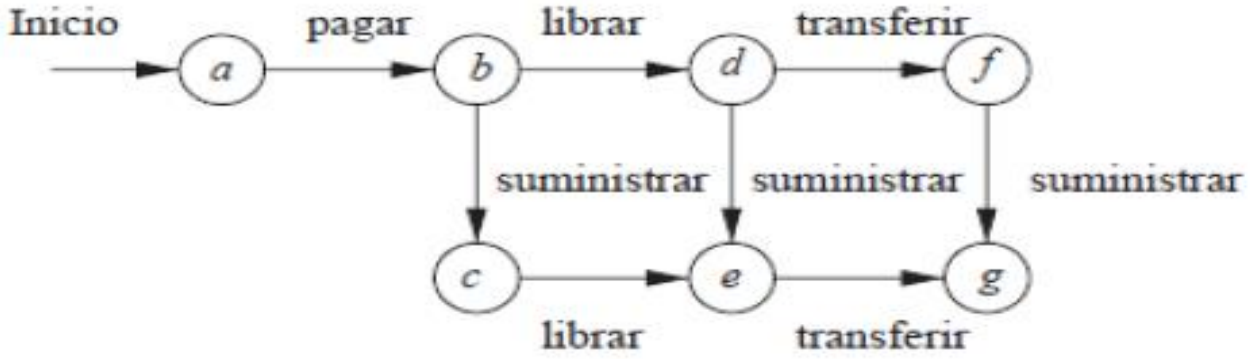
- 1. El cliente decide pagar. Es decir, el cliente envía el dinero a la tienda.
- 2. El cliente decide cancelar el pago.
- 3. La tienda suministra los bienes al cliente.
- 4. La tienda libra el dinero. Es decir, el dinero se envía al banco con la solicitud de que su valor se asigne a la cuenta de la tienda.
- 5. El banco transfiere el dinero creando un nuevo archivo de dinero electrónico cifrado y se lo envía a la tienda.

Los protocolos de este tipo pueden representarse mediante un autómata finito.

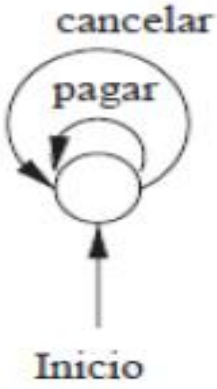
Cada estado representa una situación en la que puede encontrarse uno de los participantes. Es decir, el estado "recuerda" qué sucesos importantes han ocurrido y qué sucesos todavía no han tenido lugar.

Las transiciones entre estados se producen cuando tiene lugar uno de los cinco sucesos descritos anteriormente.

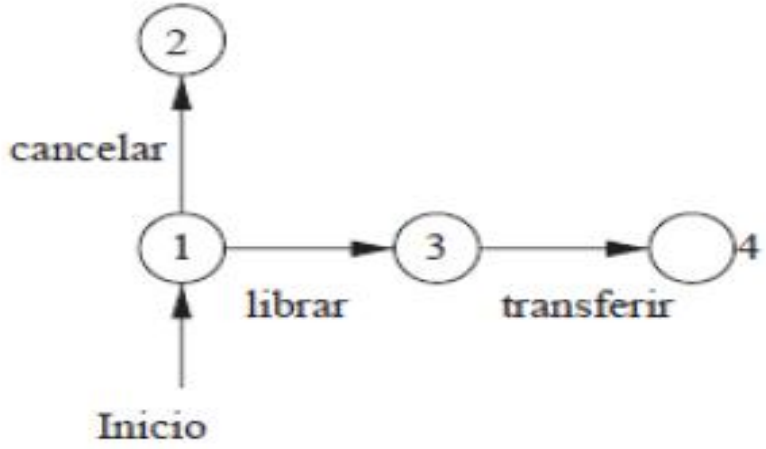
Supondremos que estos sucesos son "externos" al autómata que representa a los tres participantes, aunque cada participante sea responsable de iniciar uno o más de los sucesos. Lo importante en este problema es qué secuencias pueden ocurrir y no quién las inicia.



(a) Tienda



(b) Cliente



(c) Banco

Definición de los autómatas finitos determinísticos AFD

- Los diagramas son descripciones útiles de los autómatas finitos porque nos permiten visualizar fácilmente las acciones del algoritmo.

- Sin embargo, es necesario tener una descripción más formal de un autómata finito, y por ello procederemos ahora a proporcionar una definición matemática.

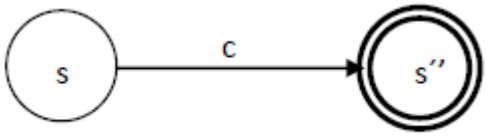
- La mayor parte del tiempo, no necesitaremos una visión tan abstracta como ésta, y describiremos la mayoría de los ejemplos en términos del diagrama.

- Otras descripciones de autómatas finitos también son posibles, particularmente las tablas, y éstas serán útiles para convertir los algoritmos en código de trabajo.

- Un AFD (Autómata Finito Determinista **AFD**) M se compone de:
- Un alfabeto Σ
- Un conjunto de estados S
- Una función de transición ∂: $S_{n-1} \times \Sigma \rightarrow S_n$
- Un estadio inicial S_0 que pertenece a S
- Un conjunto de estados de aceptación A que pertenecen a S

El lenguaje aceptado por M, escrito como L(M), se define como el conjunto de cadenas $c_1, c_2, c_3, \ldots, c_n$, con cada $c_i \in \Sigma$, tal que existan estados $s_1 = \partial(s_0, c_1)$, $s_2 = \partial(s_1, c_2)$, ..., $s_n = \partial(s_{n-1}, c_n)$ con s_n como un elemento de A

Para una transición del estado s al estado s’ etiquetada con c, se tendrá un diagrama del siguiente aspecto:



- El doble círculo del estado s’, indica que este es un estado de aceptación que pertenece al conjunto **A**.
- Un **AFD** para reconocer el lenguaje de los identificadores se verá como:

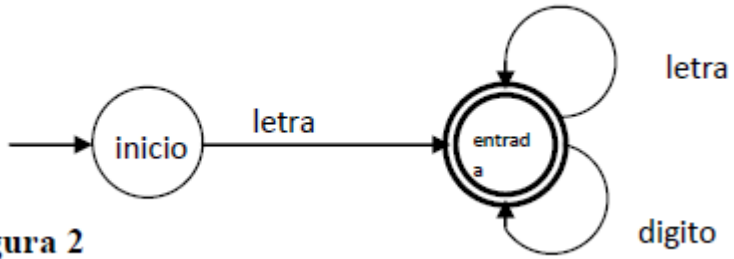


Figura 2

- Debemos destacar que en este caso hemos usado nombres para los estados, en lugar de números, lo que no contradice a la definición dada.
- Otra cosa a tener en cuenta es que no se han etiquetado las aristas con símbolos del alfabeto, sino que se han etiquetado con nombres que representan conjuntos de símbolos:

letra → [a-zA-Z]

digito → [0-9]

Tal como se los ha definido, un AFD debe tener una transición ð (s,c) para cada estado s y para cada símbolo c del alfabeto Σ.

Sin embargo en el diagrama de la figura 2, en el estado inicio, está definida la transición ð (inicio, c), solo si c es una letra.

Las transiciones que no se han definido, representan errores para el lenguaje que reconoce el autómata, y en general no se dibujan en los diagramas de estado, pero deben ser tenidas en cuenta al momento de la programación del autómata.

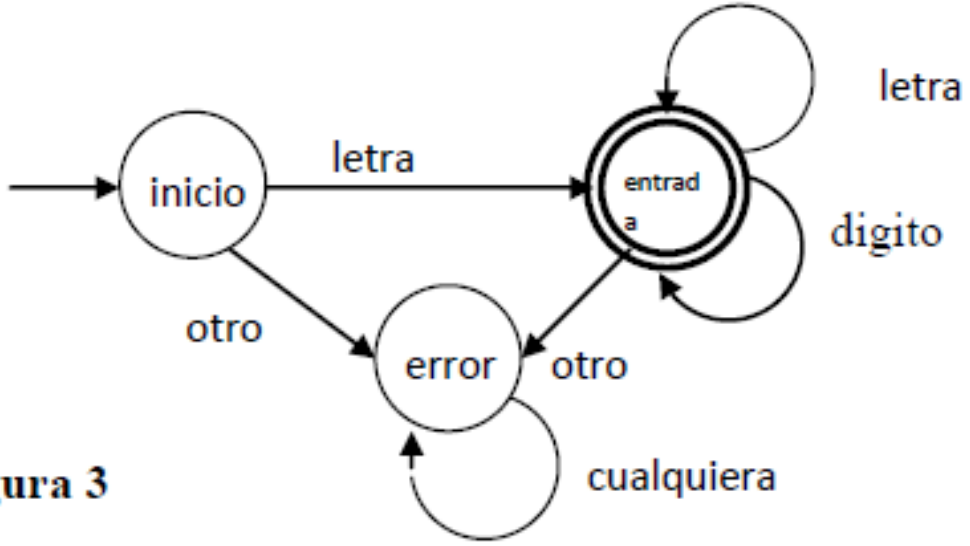


Figura 3

Otro ejemplo de Autómata, es el que reconoce el lenguaje de los números, ya sean estos enteros, con signo o sin él y también números en formato exponencial:

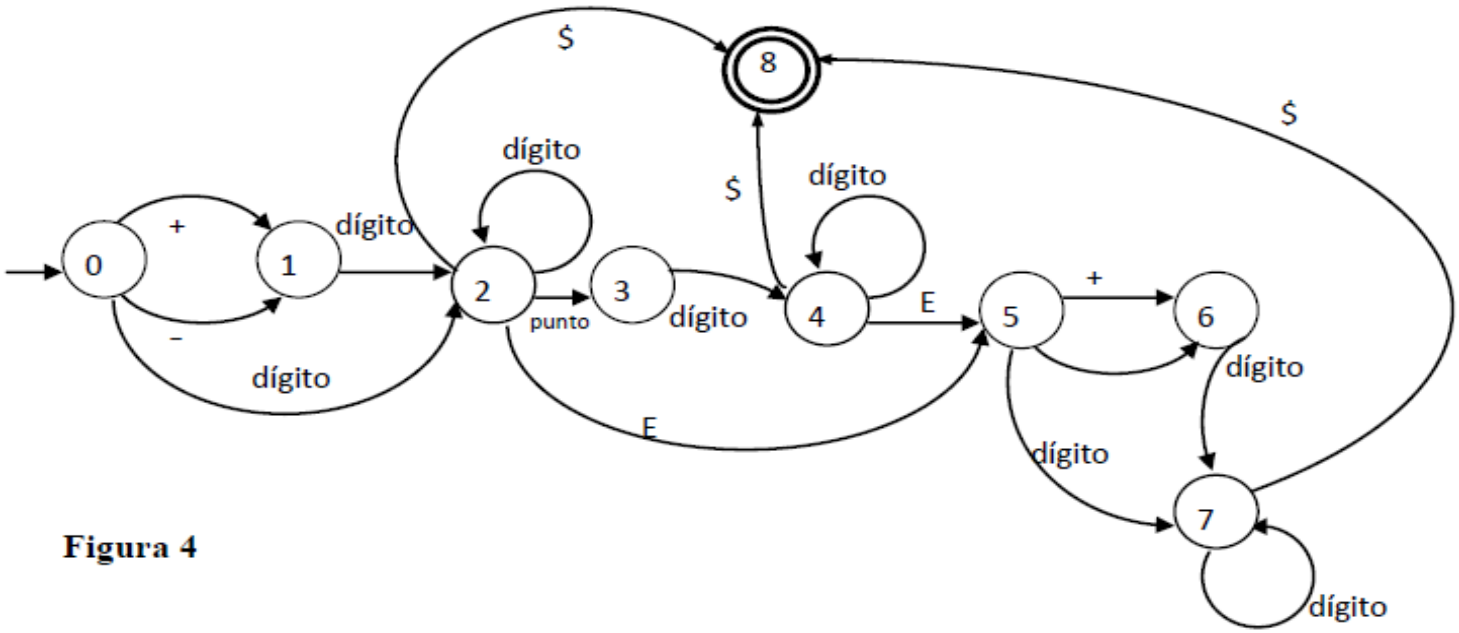


Figura 4

123, +25, -120, 3.14, 12E-3, etc.

Hay otras notaciones más cómodas para describir los autómatas como ser una Tabla de Transiciones.

Estados	Alfabeto					
	dígito	+	-	punto	E	$
0	2	1	1			
1	2					
2	2			3	5	8
3	4					
4	4				5	8
5	7	6	6			
6	7					
7	7					8
8	acepta					

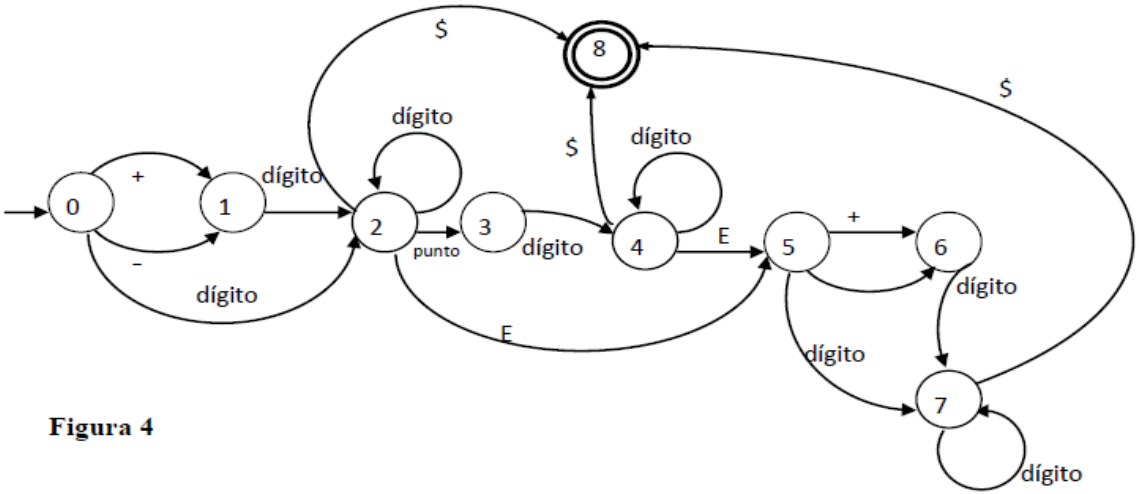


Figura 4

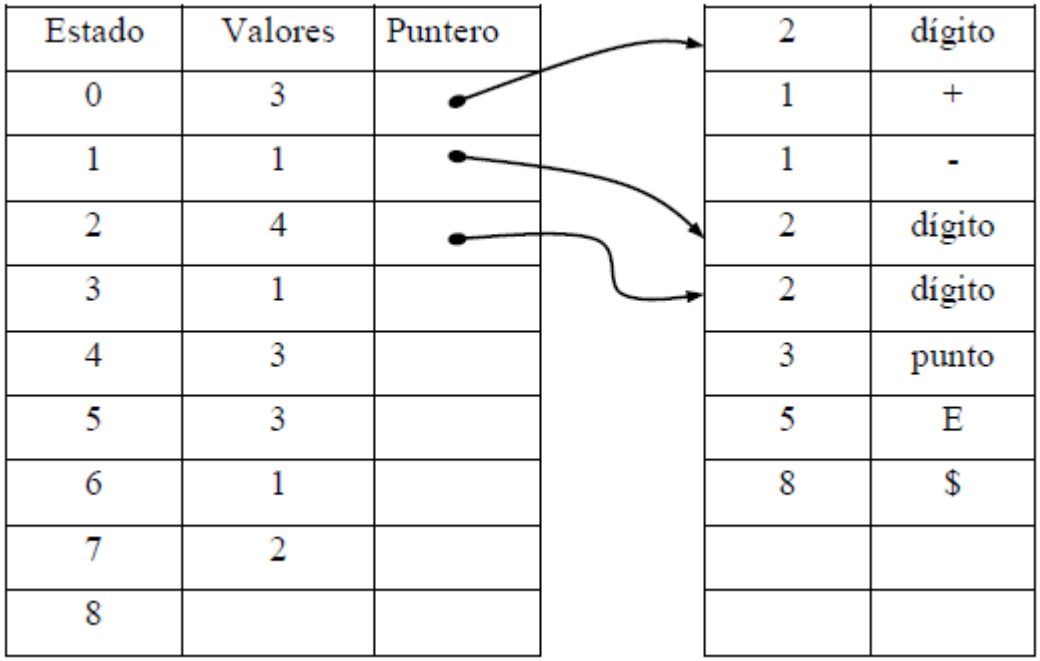
Estados	Alfabeto					
	dígito	+	-	punto	E	$
0	2	1	1			
1	2					
2	2			3	5	8
3	4					
4	4				5	8
5	7	6	6			
6	7					
7	7					8
8	acepta					

Implementación de Autómatas Finitos en Código

Estados	Alfabeto					
	dígito	+	-	punto	E	$
0	2	1	1			
1	2					
2	2			3	5	8
3	4					
4	4				5	8
5	7	6	6			
6	7					
7	7					8
8	acepta					

```
estado =0;
while (estado != 8 && estado != error )
{
        ch = siguiente carácter de entrada;
        estado = T[estado][ch];
}
if (estado = =8) aceptar();
else error();
```

- Un analizador típico puede tener unos 100 estados y manejar unos 100 símbolos de entrada, lo que daría lugar a una matriz de transición de 10000 elementos, y a lo sumo con 100 o 200 elementos ocupados.
- Para solucionar este problema, solo se almacenan los elementos no nulos de la matriz en una tabla, y se genera otra tabla que contiene para cada estado (fila) un puntero al primer valor de ese estado, y el número de valores para ese estado:



Estado	Valores	Puntero
0	3	
1	1	
2	4	
3	1	
4	3	
5	3	
6	1	
7	2	
8		

2	dígito
1	+
1	-
2	dígito
2	dígito
3	punto
5	E
8	$

Estados	Alfabeto					
	dígito	+	-	punto	E	$
0	2	1	1			
1	2					
2	2			3	5	8
3	4					
4	4				5	8
5	7	6	6			
6	7					
7	7					8
8	acepta					

Estado	Valores	Puntero
0	3	
1	1	
2	4	
3	1	
4	3	
5	3	
6	1	
7	2	
8		

2	dígito
1	+
1	-
2	dígito
2	dígito
3	punto
5	E
8	$

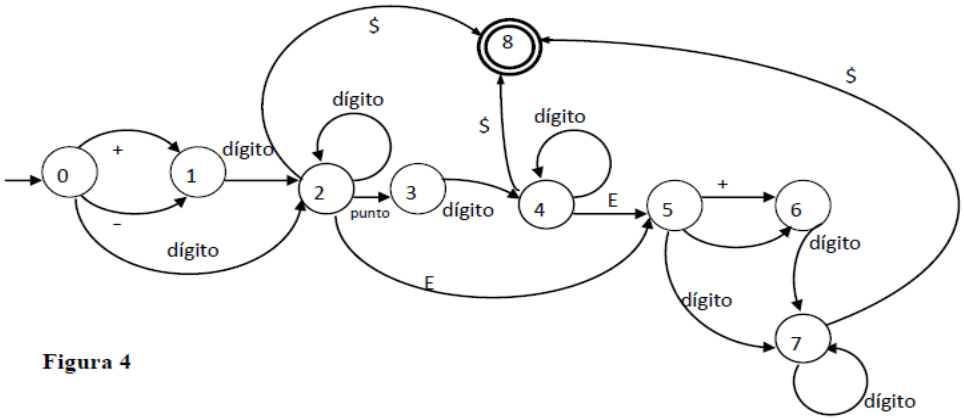


Figura 4

Autómatas Finitos No Deterministas AFN

Un autómata finito “no determinista” (AFN) tiene la capacidad de estar en varios estados a la vez.

Igual que el AFD, un AFN tiene un conjunto finito de estados, un conjunto finito de símbolos de entrada, un estado inicial y un conjunto de estados de aceptación.

- También dispone de una función de transición ð .

La diferencia entre los AFD y los AFN se encuentra en la **función ð** .

En los AFN, ð es una función que toma un estado y símbolos de entrada como argumentos, pero devuelve un conjunto de cero, uno o más estados (en lugar de devolver exactamente 1 como en un AFD)

Definición de autómata finito no determinista

Un AFN se representa esencialmente como un AFD:

$$A = (Q, \Sigma, \partial, q0, F)$$

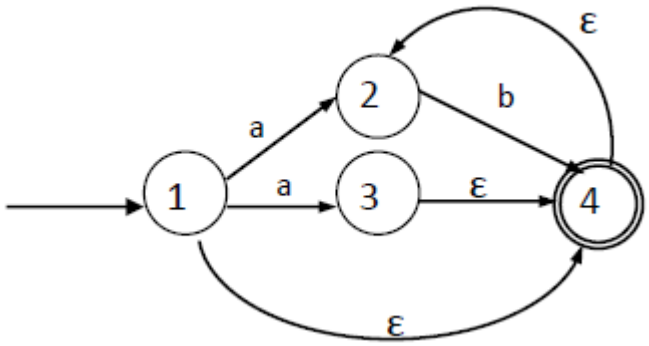
donde:

- 1. Q es un conjunto finito de estados.
- 2. Σ es un conjunto finito de símbolos de entrada.
- 3. q0, un elemento de Q, es el estado inicial.
- 4. F, un subconjunto de Q, es el conjunto de estados finales (o de aceptación).
- 5. ∂, la función de transición, es una función que toma como argumentos un estado de Q y un símbolo de entrada de Σ y devuelve un subconjunto de Q.

La única diferencia entre un AFN y un AFD se encuentra en el tipo de valor que devuelve ∂ : un conjunto de estados en el caso de un AFN y un único estado en el caso de un AFD.

Ejemplo de un AFN:

Consideremos el siguiente autómata:



La cadena **abb** puede ser aceptada por cualquiera de las siguientes secuencias de transiciones:

$$\xrightarrow{a} 1 \xrightarrow{b} 2 \xrightarrow{\varepsilon} 4 \xrightarrow{b} 2 \xrightarrow{} 4$$

$$\xrightarrow{a} 1 \xrightarrow{\varepsilon} 3 \xrightarrow{\varepsilon} 4 \xrightarrow{b} 2 \xrightarrow{\varepsilon} 4 \xrightarrow{b} 2 \xrightarrow{} 4$$

Los AFN se pueden implementar de maneras similares a los AFD, excepto que como los AFN son no determinísticos, tienen muchas secuencias diferentes de transiciones en potencia que se deben probar.

Un programa que simula un AFN debe almacenar transiciones que todavía no se han probado y realizar el **retroseguimiento** a ellas en caso de falla.

Los algoritmos que realizan una gran cantidad de **retroseguimientos** tienden a ser poco eficientes

Se puede resolver el problema de simular un AFN por medio del método que estudiaremos más adelante, el cual convierte un AFN en un AFD.