Máquinas de Estados Finitos
Breve Introducción

Jorge Alejandro Gutiérrez Orozco
Escuela Superior de Cómputo

22 de agosto de 2008

Resumen

Hablaremos de algunas de las más comunes Máquinas de Estados Fintos, una breve descripción sobre ellas, su definición formal y sus relaciones entre ellas. En la última sección se da una sencilla definición de una *Máquina de Turing*. Existen varios temas que se deben de conocer antes de abordar Máquina de Turing, pero dado que la explicación que se propone no es del todo profunda, no es necesario tener amplias bases teóricas. El presente va dirigido a alumnos de Nivel Superior como material de apoyo en el estudio de Máquinas de estados finitos. Se ofrece un enfoque diferente con explicaciones sumamente sencillas.

Se necesitan conocimientos previos antes abordar los siguientes temas, así que explicaremos **brevemente** algunos conceptos que utilizaremos.

- *Conjunto*: Es un colección de objetos relacionados entre ellos. Un conjunto solo indica los elementos que lo componen, por tanto no es necesario tener un orden y no se repiten los elementos dentro de un conjunto. Un *Conjunto finito* es aquel donde podemos contar sus elemntos, por ejemplo: $C = \{\heartsuit, \clubsuit, \diamondsuit, \spadesuit\}$ claramente podemos notar que son cuatro elementos. En un *Conjunto infinito* encontramos de dos tipos: los *Conjuntos Infinitos no Numerables* donde no podemos contar todos los elementos, un buen ejemplo es el conjunto de todos los números reales $\Re$ y el conjunto de los *Infiniftos Numerables* como el caso de los números naturales $\mathbf{N}$, donde podemos numerar todos los elementos aunque haya un infinito de ellos.

- *Subconjunto*: Es una colección de elementos que a su vez pertenece a otro conjunto de igual o mayor número de miembros. Por ejemplo: consideremos los conjuntos $A = \{a, b\}$ y $B = \{z, a, c, b, d\}$ se dice que A es un subconjunto propio de B y se escribe como $A \subset B$, si en algún momento el conjunto A pudiese llegar a tener los mismos elementos que B pero sin dejar de ser subconjunto se deberá escribir $A \subseteq B$. Dos conjuntos con los mismos elementos son iguales.

- *Símbolo*: Es la representación abstracta de un objeto, puede ser un dígito o una letra. En general es cualquier caracter que nos represente algún elemento. Algunos ejemplos son: a, 1, π, etc.

- *Alfabeto*: También llamado *Vocabulario*, es un conjunto finito de símbolos. Debe existir al menos un símbolo en el alfabeto, es decir el alfabeto no puede ser un conjunto vacío. Comunmente se le denomina Σ a este conjunto.

- *Cadena*: Es una secuencia de símbolos hecha con los elementos de un Alfabeto, por eso se dice que una cadena ω sobre un alfabeto Σ es un elemento del alfabeto universal Σ^*, es decir $\omega \in \Sigma^*$, donde Σ^* es un conjunto formado de todas las posibles cadenas que se puedan hacer con los elementos de Σ. Una cadena no es un subconjunto del Alfabeto universal, es alguno de sus elementos. Una cadena no puede ser infinita. También se le conoce como *frase* o *palabra*. Por ejemplo consideremos un alfabeto $\Sigma = \{1, 0\}$, entonces $\Sigma^* = \{1, 0, 00, 01, 10, 11, 000, ...\}$ una cadena ω sobre Σ podría ser 101011010, esta cadena es un elemento de Σ^*.

- *Lenguaje*: Es un conjunto de cadenas, las cuales deben estar formadas con los símbolos de un Alfabeto Σ, entonces decimos que el Lenguaje L está sobre el Alfabeto Σ. Por ejemplo: el lenguaje $L = \{100, 001, 00, 1111\}$ se forma con los elemetos de $\Sigma = \{1, 0\}$, la cadena $\sigma = 1010$ se forma también con los elemntos de Σ (*i.e.* $\sigma \in \Sigma$) pero no pertenece al lenguaje L y se denota $\sigma \notin L$.

- *Estado*: Es la situación o las condiciones en que se halla un objeto en algún momento, dicho objeto no puede estar en más de un estado al mismo tiempo. En el caso de una máquina son las características que posee en un momento dado.

- *Algoritmo*: Es una secuencia finita de instrucciones bien definidas. Un algoritmo está compuesto por una sucesión de pasos que llevan siempre a un mismo resultado.
- *Computación*: Es la aplicación de un algoritmo sobre un conjunto de datos de entrada, obteniendo como resultado otro conjunto de datos de tal proceso.

1. Introducción

Las Máquinas de estados Finitos conocidas como *Finite State Machines* por su traducción al Inglés, nos sirven para realizar procesos bien definidos en un tiempo discreto. Reciben una entrada, hacen un proceso y nos entregan una salida. Notemos que éstas máquinas hacen una *computación*.

En otras palabras, imaginemos una máquina capaz de seguir una secuencia finita de pasos al introducir un conjunto de datos en ella, solo se puede leer un dato en cada paso que se realize, por tanto el número de pasos a seguir está dado por el número de datos a introducir. Cada entrada diferente genera una salida diferente, pero siempre el mismo resultado con los mismos datos de entrada.

Por lo tanto una computación es capaz de resolver un problema, sí y solo sí tiene una solución algorítmica, es decir, puede ser descrito mediante una secuencia finita de pasos bien definidos.

Mediante una computación podemos encontrar soluciones a problemas que teóricamente tienen una representación algorítmica, pero que pueden necesitar tal cantidad de recursos (factores como el tiempo y el espacio de almacenamiento) que desde el punto de vista práctico no se puede llegar a la solución.

2. Máquinas de Estados Finitos

Una máquina de estados finitos en un modelo abstracto para la manipulación de símbolos, nos permiten saber si una cadena pertence a un lenguaje o nos pueden generar otro conjutno de símbolos como resultado.

Llamaremos una Máquina de Estados Finitos como *Autómata Finito*, el hecho es que un Autómata y una Máquina de Estados Finitos son lo mismo, podemos utilizar ambos términos de forma indistinta.

Los Autómatas se caracterizan por tener un *Estado inicial*, reciben una *cadena* de símbolos, cambian de estado por cada elemento leído o pueden permanecer en el mismo estado. También tienen un conjunto de *Estados Finales o Aceptables* que nos indican si una cadena pertenece al lenguaje al final de una lectura.

Los Autómatas se clasifican en 2 tipos:

- Autómata Finito Determinísta.
- Autómata Finito no Determinísta.

Siempre llamamos un Autómata como *Autómata Finito*, esto nos puede llevar a pensar que existe algun tipo de *Autómata Infinito*, lo cual no tiene mucho sentido pensar en un tipo de Máquina que tiene un conjunto infinito de estados, pero aún se discute su utilidad para propósitos prácticos. Un "Autómata Infinito" tiene cintas infinitas o registros de almacenamiento de capacidad ilimitada, esto le da el carácter de infinito [5].

Autómatas Finitos Determinístas. Un Autómata recibe secuencialmente una cadena de símbolos y cambia de estado por cada símbolo leído o también puede permanecer en el mismo estado. Al final de la lectura el estado del Autómata nos indica si la cadena es aceptada o mejor dicho pertenece al Lenguaje que describe nuestra máquina. Si al final de leer todos los símbolos de entrada la máquina está en alguno de los estados Finales entonces esa cadena es aceptada, si el estado no es final entonces la cadena no pertenece al lenguaje.

Las partes que componen una Autómata son 5 y se pueden definir:

$$A = \{Q, q_0, F, \Sigma, \delta\}$$

donde:

Q: Conjunto finito de estados.

q_0: Estado inicial donde $q_0 \in Q$. Debe haber uno y sólo un estado incial.

F: Conjunto de estados finales $F \subseteq Q$. El estado q_0 también puede ser final.

Σ: Alfabeto finito de entrada.

δ: Función de Transición $Q \times \Sigma \to Q$.

Supongamos que el Autómata se encuentra en el estado q_i donde $q_i \in Q$, también tenemos el símbolo a donde $a \in \Sigma$. Una entrada a causa que el Autómata cambie del estado q_i al estado q_k. La función δ, llamada *función de transición*, describe este cambio de la forma $\delta(q_i, a) \to q_k$ de esta forma obtenemos un nuevo estado. Se entiende por *transición* como el proceso que hace un Autómata al cambiar de estado.

La forma mas fácil de imaginarnos un Autómata es mediante un *diagrama de transición*. Un diagrama de transición es un digrafo etiquetado con los elementos de un Autómata para este caso, pero de hecho se puede representar cualquier Máquina de Etados Finitos por medio de un diagrama de transición, es la forma más común de hacerlo por ejemplo (ver figura 1):

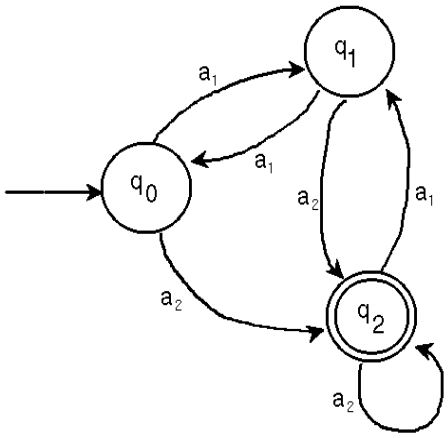

Figura 1: Diagrama de Transición de un Autómata

En un diagrama de transición existe un nodo por cada estado q_i de Q. Los estados finales estań encerrados en un círculo doble. El estado incial q_0 es apuntado por una flecha que no proviene de ningún otro estado. Para cada estado q_i y un símbolo a, hay exactamente una y sólo una flecha que inicia en q_i y termina en $\delta(q_i, a)$, es decir en q_k, la flecha es etiquetada como a. Si q_k pertenece a F decimos que la entrada es aceptada.

Debe haber exactamente una flecha saliendo de cada estado por cada símbolo $a_0, a_1, a_2...a_n$, por tanto todos los estados tienen el mismo número de flechas saliendo de cada uno de ellos. Con esto garantizamos que nuestro Autómata pueda ser llamado *Determinísta*. No importa el estado ni el símbolo leído, siempre hay una transición definida.

Para describir por completo una función de transición δ ocupamos una *Tabla de Transición*. Las columnas se etiquetan con los símbolos de entrada, la filas son etiquetadas con los estados y en las intersecciones se colocan los nuevos estados $\delta(q_i, a)$, suponinedo que $q_i \in Q$ es la columna y $a \in \Sigma$ la fila que lo intersecta. La tabla de transición de la figura 1 es:

	a_1	a_2
$\rightarrow q_0$	q_1	q_2
q_1	q_0	q_2
$\leftarrow q_2$	q_1	q_2

El estado incial tiene una flecha que apunta a él, los estados finales tienen una flecha que sale de ellos y los estados que no son finales y no son el inicial no tienen flecha. En caso de que nuestro estado inicial también sea un estado final, se apuntará con una flecha doble $\leftrightarrow$.

Una tabla de transición representa una función δ la cual recibe un símbolo y un estado, si queremos introducir una cadena ω donde $\omega \in \Sigma^*$, donde Σ^* es la

cerradura de Σ, [1] en lugar de un solo símbolo debemos usar δ^* conocida como *Función de Transición Extendida* que nos permite manejar una cadena dado que es una función de $Q \times \Sigma^*$ y cumple con:

- $\delta^*(q_i, a) \rightarrow \delta(q_i, a)$ donde $q_i \in Q$ y $a \in \Sigma$
- $\delta^*(q_i, \varepsilon) \rightarrow \delta(q_i, a)$ donde $\varepsilon \in \Sigma$ es el elmento vacío.
- $\delta^*(q_i, aw) \rightarrow \delta^*(\delta(q_i, a), w)$ donde $a \in \Sigma$ y $w \in \Sigma^*$

Si evaluamos δ^* con un estado $q_i \in Q$ y con un símbolo $a \in \Sigma$ se comporta de la misma forma que δ.

El primer elemento de Σ^* generalmente es el elemento vacío ε puede ser el único elemento de nuestro lenguaje $L = \{\varepsilon\}$ o podemos suponer que $\varepsilon \in (\Sigma^* - \Sigma^+)$.
El autómata permanece en el mismo estado al introducir ε, es decir no cambia el estado, se comporta como un símbolo *neutro* para δ^*.
El elemento ε puede ser aceptado sí y solo sí el estado inicial q_0 también pertenece al conjunto de estados finales $q_0 \in F$. El Autómata que solo acepta el elemento vacío es:

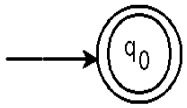

Figura 2: Autómata que acepta el símbolo ε

La última propiedad de δ^* nos define como evaluar cadenas en una forma recursiva. Se toma el primer símbolo de la cadena y se evalúa con δ, el estado resultante es evaluado con el segundo símbolo para obtener un nuevo estado que a su vez será usado con el tercer símbolo y así sucesivamente. De esta forma pasamos por los estados $\delta(q_0, a_0), \delta(\delta(q_0, a_0), a_1)...$ hasta terminar de evaluar la cadena, si el Autómata se encuentra en un estado final f se entiende que $\delta^*(q_0, \omega) = f$ y la cadena es aceptada.

No importa si en algún momento de la lectura de símbolos llegamos a un estado final, si no se ha terminado de leer la cadena, el autómata continúa recibiendo símbolos y cambiando de estado.

[1] También conocido como lenguaje universal, consta de todas las posibles cadenas que se puedan formar con los símbolos de Σ [6]

Formalmente se dice que un Lenguaje L es aceptado por un Autómata A sí y solo sí $\exists\ \omega \in \Sigma^*$ y cumple con $\delta^*(q_0, \omega) = f$ donde $q_0 \in Q$ y $f \in F$. Nuestro Lenguaje es aceptado y se compone de todos los ω de la siguiente forma:

$$L(A) = \{\omega \in \Sigma^* | \delta^*(q_0, \omega) \in F\}$$

Autómatas Finitos No Determinístas A diferencia de los Autómatas Finitos Determinístas, donde existe una única forma de llegar de un estado a otro con una entrada y se tiene solo un estado incial, los Autómatas Finitos No Determinístas no cuentan con estas virtudes, pero son una herramienta de mucha ayuda cuando queremos diseñar un Autómata Determinísta. Para cada Autómata No Determinísta existe un Autómata Determinísta que lo representa y que acepta el mismo lenguaje.

Podemos definir un Autómata Finito No Determinísta como:

$$A = \{Q, I, F, \Sigma, \delta\}$$

donde:

Q: Conjunto finito de estados.

I: Conjunto de estados inicales donde $I \in Q$.

F: Conjunto de estados finales $F \subseteq Q$.

Σ: Alfabeto finito de entrada.

δ: Función de Transición $Q \times \Sigma \rightarrow S$ donde $S \subseteq Q$.

Aparentemente es muy similar a un Autómata Finito Determinísta, pero se perciben algunas diferencias. Puede existir más de un Estado inicial [8] y la función de transición δ ahora nos entrega un conjunto, tal vez vacío, de posibles estados. Precisamente esta es la diferencia entre un Autómata Determinísta y uno No Determinísta. Cuando todas las transiciones están *determinadas* en un Autómata, es decir para cada par de (*estado,símbolo*) existe uno y sólo un estado correspondiente, se tiene un Autómata Determinísta. Si se tiene al menos una transición *no definida* o *indeterminada* entonces tenemos un Autómata No Determinísta.

En la función de transición extendida también hay algunos cambios, se puede comenzar en cualquiera de los estados iniciales q_I (donde $q_I \in I$) y suma todas las posibles transiciones t que inicien en el mismo estado pero que lleven a estados diferentes a pesar que el símbolo leído a es el mismo. Definimos cualquier transición como $t = \delta(q, a)$ donde $q \in Q$ es un estado y $a \in \Sigma$ es un símbolo. Entonces podemos definir:

$$\delta^*(q_I, aw) = \sum_{i=0}^{n} \delta^*(t_i, w)$$

El número máximo de transiciones para ese par (estado,símbolo) está dado por n, es decir todas las flechas que salen de cualquier estado q y que están etiquetadas por el mismo símbolo a. Para comenzar a evaluar la cadena, donde $w \in \Sigma^*$, partimos de cualquier estado inicial q_I, evaluamos el primer símbolo a y sumamos todas las transiciones posibles, notese que ahora δ^* nos entrega un conjunto de posibles estados, es decir $\delta^*(q_I, aw) \rightarrow S$ donde $S \subseteq Q$ se pueden dar los siguientes casos:

- Si se evalua un solo símbolo $\delta^*(q, a) = \delta(q, a)$ el conjunto S que obetenemos tiene un solo elemento que es el estado siguiente.
- Si el símbolo a evaluar es el vacío $\delta^*(q_i, \varepsilon) = q_i$ entonces no existe transición alguna para ese par (q, a), es decir ε se comporta como un valor neutro para el operador δ^*.
- Si existen n transiciones para los mismos valores $(\delta^*(q, a) = q_1)+(\delta^*(q, a) = q_2) + ... + (\delta^*(q, a) = q_n)$ entonces el conjunto S contiene los valores $q_1, q_2, ..., q_n$.

Se dice que una cadena es aceptada si $\delta^*(q_I, aw) = q_f$ donde $q_f \in F$.

Un ejemplo de un Autómata Finito No Determinísta es la figura 3:

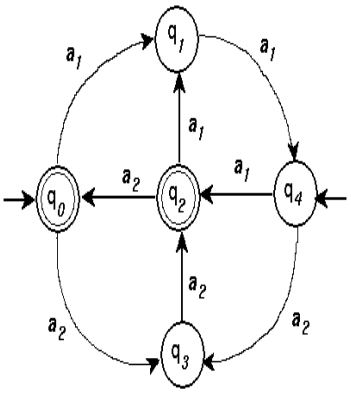

Figura 3: Diagrama de transición de un Autómata Finito No Determinísta

Se puede observar que existe más de un estado inicial y las transiciones $\delta(q_1, a_2)$ y $\delta(q_3, a_1)$ no están determinadas, basta solo una de las condiciones anteriores para considerar al Autómata como No Determinísta.

Su tabla de transición de la figura 3 es:

	a_1	a_2
$\leftrightarrow q_0$	q_1	q_3
q_1	q_4	$\emptyset$
$\leftarrow q_2$	q_1	q_0
q_3	$\emptyset$	q_2
$\rightarrow q_4$	q_2	q_3

Equivalencia entre Autómatas Finitos Las transiciones que no están determinadas en el diagrama de la figura 3 se relacionan con el símbolo $\emptyset$, éste símbolo nos representa un conjunto vacío. Por lo tanto podemos decir que las transiciones indeterminadas llevan a un estado $\emptyset$. Si incluímos el estado *vacío* para que forme parte de nuestro autómata no determinísta de la forma $\emptyset \in Q$ se puede completar la tabla de transición (como la tabla anterior), con esto quedan definidas todas las transiciones del autómata o por lo menos garantizamos que cualquier símbolo de entrada nos lleve a otro u otros estados, tal vez al mismo.

Aunque se tenga completa la tabla de transición, es decir se tiene una respuesta para toda $\delta(q, a)$, un autómata podría tener más de un estado inicial, razón suficiente para llamarse *no determinísta*. Entonces se necesita crear un estado que represente todos los posibles estados iniciales el cual será nuestro nuevo estado incial, también necesitamos crear un estado $\emptyset$ para las transiciones no determinadas.

Una herramineta útil para encontrar tales estados es el *conjunto potencia* P que representa todas las posibles combinaciones que se pueden hacer con un conjunto (incluyendo el $\emptyset$) y generalmete contiene 2^n elementos, donde n es el total de elementos del conjunto evaluado[4]. Por ejemplo el conjunto $W = \{1, 0\}$ contiene 2 elementos, el conjunto potencia $P(W)$ tiene $(2)^2 = 4$ elementos y es de la forma $P(W) = \{\emptyset, 0, 1, \{0, 1\}\}$.

Para poder encontrar un autómata finito determinísta equivalente a otro autómata finito no determinísta, necesitamos dirigir a un estado $\emptyset$ aquellas transiciones que no estén determinadas, reducir el conjunto de estados inciales a un único estado que los represente a todos y encontrar los estados que son alcanzados por la misma transición para crear únicos y difernetes estados que los representen. Así obtenemos una forma de convertir cualquier Autómata No Determinísta en uno Determinísta.

Para ilustrar mejor el proceso de crear un *Autómata Equivalente* a otro, es decir un autómata que acepte el mismo lenguaje L, vamos a dar un ejemplo:

Dado el Autómata No determinísta $A_N = \{Q_N, \Sigma, I, F_N, \delta\}$ donde $Q_N = \{q_0, q_1, q_2\}$ y $\Sigma = \{a_1, a_2\}$, vamos a convertirlo en un Autómata Determinísta A_D que acepte el mismo legunaje, es decir $L(A_N) = L(A_D)$ con el mismo alfabeto de entrada Σ, el diagrama de transición es:

ht

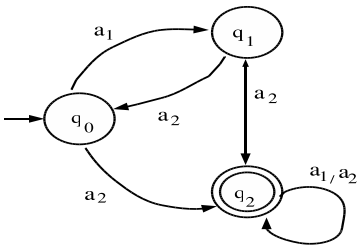

Figura 4: Ejemplo de Autómata Finito No Determinísta

La tabla de transición del autómata es:

	a_1	a_2
q_0	q_1	q_2
q_1	$\emptyset$	$\{q_0, q2\}$
q_2	q_2	q_2

Definimos el conjunto potencia $P(A_N) = \{\emptyset, q_0, q_1, q_2, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$ el total de elementos es $(2)^3 = 8$ dado que $|Q_N| = 3$. Por comodidad vamos a escribir los estados que nos *representan* la combinación de otros estados de la forma $q_3 = \{q_0, q_1\}$, $q_4 = \{q_0, q_2\}$, $q_5 = \{q_1, q_2\}$ y $q_6 = \{q_0, q_1, q_2\}$, ahora dibujemos el diagrama:

ht

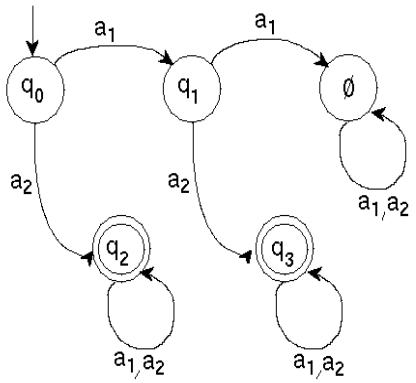

Figura 5: Autómata Finito Determinísta

Notamos que el estado incial sigue siendo uno. Todos los estados tienen determinadas todas las transiciones, las transiciones etiquetadas con dos símbolos son equivalentes a tener dos transiciones con una etiqueta. Tenemos dos estados más, el vacío $\emptyset$ y el estado q_4 el cual es un estado final porque se obtuvo a partir de otros estados de los cuales uno ya era final. Los estados q_3, q_5 y q_6 no aparecen en el diagrama porque no existe transición alguna que permita llegar a ellos. Cualquier cadena que pertenezca al Autómata de la figura 4 también pertenecerá al Autómata de la figura 5.

2.1. Máquinas de Estados Finitos Transductoras

Una derivación de los Autómatas son la *Máquinas de Estados Finitos Transductoras*, la diferencia es que las Transductoras nos entregan como resultado un conjunto de símbolos que pertenecen al lenguaje, las aceptadoras nos indican si un conjunto de símbolos pertenece o no al lenguaje. La diferencia radica en cambiar el conjunto de estados finales por algún tipo de función que nos arroje valores.

Por su **funcionamiento** podemos definir a las Máquinas de Estados Finitos en dos tipos:

1. Máquinas de Estados Finitos Transductoras.

2. Máquinas de Estados Finitos Aceptadoras.

Ambos tipos de máquinas siguen siendo Autómatas, las transductoras son una variación de las Aceptadoras.

En estas máquinas existe una función (la llameremos función de salida) que puede tomar como parámetro el estado actual o la transición de nuestra máquina y arroja un elemento del conjunto de símbolos de salida. Existe otra función (la llamermos función estado) que nos indicará el estado siguiente que deberá adoptar nuestra máquina según el caracter leído de la cadena de entrada y el estado

actual. Observamos que la función de salida puede tomar como parámetro el estado o la transición, por tanto tenemos 2 tipos de máquinas, las que entregan un valor al llegar a un estado y las que entegan un valor al momento de cambiar de estado (transición).

Máquina de Moore Se definie como la 6-tupla $\{Q, \Sigma, S, \delta, \lambda, q_0\}$ donde:

Q: Es el conjunto finito de estados.
Σ: Es el alfabeto de entrada.
S: Es el alfabeto de salida.
δ: Función de transición $Q \times \Sigma \to Q$.
λ: Función de Q a S, dado q nos arroja una s donde $s \in S$ y $q \in Q$.
q_0: Estado inicial.

Nuestra máquina comienza en su estado inicial q_0, la función de salida λ nos entrega un símbolo s cada vez que llegamos a un estado, la función de transición δ lee un elemento de la cadena de entrada Σ e indica el nuevo estado que adoptaremos, puede ser el mismo, depende del símbolo leído y del estado actual.

Un bonito ejemplo es:

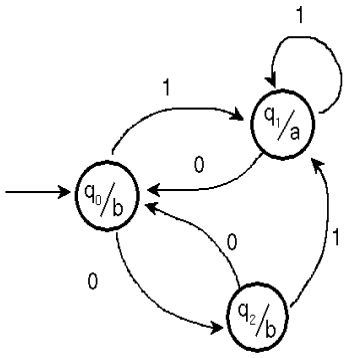

Figura 6: Ejemplo de una Máquina de Moore

Si introducimos la cadena 11001 la máquina arroja la secuencia de datos *baabba*. El primer símbolo simpre será el mismo ya que siempre que comenzamos a analizar una cadena partimos del estado inicial.

Máquina de Mealy También se define como una 6-tupla $\{Q, \Sigma, S, \delta, \lambda, q_0\}$ donde:

Q: Es el conjunto finito de estados.
Σ: Es el alfabeto de entrada.
S: Es el alfabeto de salida.
δ: Función de transición $Q \times \Sigma \to Q$.
λ: Función de salida $Q \times \Sigma \to S$, $\lambda(q_i, a) \to s$ donde $s \in S$, $q \in Q$ y $a \in \Sigma$.

q_0: Estado inicial.

Aparentemente es muy similar a la Máquina de Moore, notemos que solo cambia en la función de salida λ, ahora es una función donde necesitamos saber un elemento de la cadena de entrada. Ahora nos entregará un resultado en cada transición o y no al llegar a cada estado como en las Máquinas Moore. Supongamos que existe una Máquina de Mealy y una Máquina de Moore, ambas con el mismo conjunto de estados, el mismo estado incial, el mismo alfabeto de entrada y la misma función de transición. La Máquina de Moore nos arroja un símbolo desde que entramos al estado inicial, mientras la Máquina de Mealy hasta que realizamos la primera transición. Esto da como resultado que la Máquina de Moore tenga un elemento más en la salida que la Máquina de Mealy, siempre y cuando se tengan los mismos componentes en ambas máquinas.

Un ejemplo de una Máquina de Mealy es:

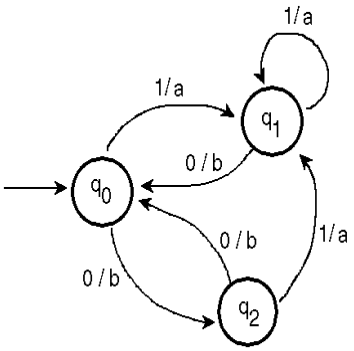

Figura 7: Máquina de Mealy

A diferencia de la Máquina de Moore se excluye el primer elemento en todas las cadenas de salida, es decir, si introducimos la misma cadena que en el ejemplo anterior 11001 obtenemos como resultado *aabba* y podemos observar la equivalencia de ambas máquinas.

3. Máquina de Turing

La idea de una Máquina que pudiera seguir un conjunto de pasos que describen cualquier sentencia matemática, es decir que puede ser descrita por un algoritmo fué introducida por primera vez por Alan Turing en 1936 en su trabajo *on computable numbers, with an application to the entscheidungsproblem.*

Una Máquina de Turing es solamente un concepto abstracto y no realmente una máquina como posiblemente la imaginemos (engranes, circuitos, etc.), pero sí podemos darnos una idea según sus propiedades básicas.

Imaginemos que nuestra máquina tiene un conjunto finito de estados (Q), tiene uno y sólo un estado inicial (q_0) el cual siempre es el primero cuando la máquina comienza a trabajar, puede tener uno o más estados finales (F) que le indican a la máquina cuando parar . Cambia de un estado a otro y regresa a ellos cuantas veces es necesario hasta llegar a uno de los estados finales y haber recorrido todos los elementos de una cadena de entrada (E). También tiene un cabezal de lectura y escritura con el cual puede obtener datos y escribirlos en una cinta que idealmente es infinitamete larga, la cual está dividida en casillas o celdas donde solo puede contener uno y sólo uno de los símbolos de entrada. Su forma de trabajo es la siguiente:
Según el estado de la máquina y el valor leído de la cadena de entrada escrita en la cinta (un símbolo por celda), escribe un nuevo símbolo en esa casilla, mueve la cinta a la derecha o a la izquierda (una celda a la vez) y cambia de estado. Nuevamente lee el símbolo de la celda actual y según su nuevo estado repetirá el mismo proceso hasta recorrer todos los símbolos de entrada o situarse en un estado terminal.
Existe una función de trancisíon (δ) o tabla de transición que nos indica, según el estado de la máquina y el valor leído, el símbolo a ser escrito y el nuevo estado que se debe adoptar así como el movimiento de la cinta (derecha o izquierda) que se va a realizar.

Ahora podemos definir formalmente una Máquina de Turing (MT) como una 7-tupla de la forma: $MT = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$

Q: Es el conjunto finifto de estados, son todos y cada uno de los posibles estados en que puede estar nuestra máquina.

Σ: Es un conjunto finito de símbolos de entrada. Son todos aquellos elementos que va a recibir nuestra máquia para su lectura.

Γ: Es el conjunto finito de símbolos que puede reconocer nuestra máquina, es decir, el alfabeto, donde $\Sigma \in \Gamma$.

δ: Es la función de transición $\delta(q_i, \sigma) \rightarrow (q_k, \gamma, R|L)$, también puede ser respresentada por una tabla llamada Tabla de transición. Y nos indica la función de nuestra máquina:

- La función δ recibe como parámetros el estado $q_i(q_i \in Q)$ en el que se encuentra actualmente la máquina y un símbolo $\sigma(\sigma \in \Sigma)$ leído de la cadena de entrada.
- Tal combinación de parámetros nos va a dar un nuevo estado (q_k) y escribirá un símbolo $\gamma(\gamma \in \Gamma)$ en la celda donde recién acaba de leer el símbolo.
- Por último se moverá la izquierda (R) o a la derecha (R) una sola celda a la vez y se repetirá el mismo proceso ahora con los nuevos valores.

B: Es el símbolo en *blanco*, indica que la celda está vacía.

F: Es el conjunto de estados finales.

A simple vista parece un modelo sencillo, pero su poder es tal que cualquier modelo computacional o algoritmo existente puede ser modelado por una Máquina de Turing.

Podemos decir que cualquier solución a un problema que pueda ser representada por una Máquina de Turing está haciendo una computación. Si una máquina imprime dos tipos tipos de símbolos, donde el primer tipo (llamadas figuras) consiste integramante en 1 y 0 y el segundo tipo serán llamados símbolos de la segunda especie entonces podemos llamar a esa máquina *computadora* [7].

Referencias

[1] TRAKHTENBROT B.A. *Algorítmos y Computadoras.* Limusa, 1973.

[2] HOPCROFT and ULLMAN. *Formal Languages and their Relation to Automata.* Addison-Wesley, 1969.

[3] MOTWANI HOPCROFT and ULLMAN. *Introduction to Automata Theory, Languages and Computation.* Pearson Education, 2001.

[4] RICHARD JHONSONBAUGH. *Matemáticas Discretas.* Iberoamérica, 1988.

[5] MINSKY Marvin L. *Computation: Finite and Infinite Machines.* Prentice-Hall, 1967.

[6] PABLO G. PADILLA. *Lenguajes y álgebra de eventos regulares.* Escuela Superior de Cómputo, 2006.

[7] A.M. TURING. *on computable numbers, with an application to the entscheidungsproblem.* London Math, Soc., 1936.

[8] LAWSON Mark V. *Finite Automata.* Chapman and Hall-CRC, 2004.