UNIDAD 6: AUTÓMATAS FINITOS.

(86:44) Técnica Digital Avanzada- Unidad 6.

Profesor: Ing. Miguel Antonio Martínez.

Máquinas de Estados Finitos (FSM).

En esta sección desarrollaremos varios temas y daremos a conocer definiciones que nos permitan entender el funcionamiento de una máquina de estados finitos y su importancia en tratamiento de circuitos digitales. Primero definiremos temas como que es un alfabeto, que es una palabra, que es un autómata de estado finito (FSA) y su diferencia con una FSM.

La teoría de autómatas es el estudio de dispositivos de cálculo abstractos, es decir, de las "máquinas". Antes de que existieran las computadoras, en la década de los años treinta, A. Turing estudió una máquina abstracta que tenía todas las capacidades de las computadoras de hoy día, al menos en lo que respecta a lo que podían calcular. El objetivo de Turing era describir de forma precisa los límites entre lo que una máquina de cálculo podía y no podía hacer; estas conclusiones no sólo se aplican a las máquinas abstractas de Turing, sino a todas las máquinas reales actuales. En las décadas de los años cuarenta y cincuenta, una serie de investigadores estudiaron las máquinas más simples, las cuales todavía hoy denominamos "autómatas finitos". Originalmente, estos autómatas se propusieron para modelar el funcionamiento del cerebro y, posteriormente, resultaron extremadamente útiles para muchos otros propósitos. También a finales de la década de los cincuenta, el lingüista N. Chomsky inició el estudio de las "gramáticas" formales. Aunque no son máquinas estrictamente, estas gramáticas están estrechamente relacionadas con los autómatas abstractos y sirven actualmente como base de algunos importantes componentes de software, entre los que se incluyen componentes de los compiladores. En 1969, S. Cook amplió el estudio realizado por Turing sobre lo que se podía y no se podía calcular. Cook fue capaz de separar aquellos problemas que se podían resolver de forma eficiente mediante computadora de aquellos problemas que, en principio, pueden resolverse, pero que en la práctica consumen tanto tiempo que las computadoras resultan inútiles para todo excepto para casos muy simples del problema. Este último tipo de problemas se denominan "insolubles" o "NP-difíciles". Es extremadamente improbable que incluso la mejora de carácter exponencial en la velocidad de cálculo que el hardware de computadora ha experimentado ("Ley de Moore") tenga un impacto significativo sobre nuestra capacidad para resolver casos complejos de problemas insolubles. Todos estos desarrollos teóricos afectan directamente a lo que los expertos en computadoras hacen. Algunos de los conceptos, como el de autómata finito y determinados tipos de gramáticas formales, se emplean en el diseño y la construcción de importantes clases de software. Otros conceptos, como la máquina de Turing, nos ayudan a comprender lo que podemos esperar de nuestro software. En particular, la teoría de los problemas intratables nos permite deducir si podremos enfrentarnos a un problema y escribir un programa para resolverlo (porque no pertenece a la clase de problemas intratables) o si tenemos que hallar alguna forma de salvar dicho problema: hallar una aproximación, emplear un método heurístico o algún otro método para limitar el tiempo que el programa invertirá en resolver el problema. En este capítulo de introducción, vamos a abordar la teoría de autómatas desde un punto de vista de alto nivel, así como sus usos. Gran parte del capítulo es una introducción a las técnicas de demostración y a los trucos

que permiten llevar a cabo dichas demostraciones. Obviamos las demostraciones deductivas, la reformulación de proposiciones, las demostraciones por reducción al absurdo, las demostraciones por inducción y otros importantes conceptos. También presentamos los conceptos que dominan la teoría de autómatas: alfabetos, cadenas de caracteres y lenguajes. ¿Por qué estudiar la teoría de autómatas? Son varias las razones por las que el estudio de los autómatas y de la complejidad de cálculo constituye una parte importante del núcleo de la Ciencias de la Computación. Esta sección presenta al lector estas razones, e introduce los temas más importantes que se cubren en este apunte. Los autómatas finitos constituyen un modelo útil para muchos tipos de hardware y software. Algunos ejemplos importantes:

1. Software para diseñar y probar el comportamiento de circuitos digitales.

2. El "analizador léxico" de un compilador típico, es decir, el componente del compilador que separa el texto de entrada en unidades lógicas, tal como identificadores, palabras clave y signos de puntuación.

3. Software para explorar cuerpos de texto largos, como colecciones de páginas web, o para determinar el número de apariciones de palabras, frases u otros patrones.

4. Software para verificar sistemas de todo tipo que tengan un número finito de estados diferentes, tales como protocolos de comunicaciones o protocolos para el intercambio seguro de información.

Empezaremos definiendo algunos conceptos:

Consideramos un conjunto **A** de símbolos no vacío, en el que una palabra o cadena **w** sobre el conjunto **A** es una secuencia finita de sus elementos. Por ejemplo, supongamos que A = {a, b, c}. Entonces las siguientes secuencias son **palabras de A**:

u = ababb v = accbaaa

Cuando se analizan palabras de A, a menudo a **A** se denomina **alfabeto** y a sus elementos, **letras**.

La secuencia vacía de letras, que se denota con λ o con $\in$, también se considera una palabra sobre **A**, que se denomina **palabra vacía**. El conjunto de todas las palabras sobre **A** se denota por $\mathbf{A}^*$ (se traduce con A estrella).

La **longitud** de una palabra **u** se escribe **l (u)**, es el número de elementos en su secuencia de letras. Para las palabras del ejemplo anterior tenemos **l (u) = 5** y **l (v) = 7**. Siguiendo con nuestro ejemplo **l (λ) = 0**, donde λ es la palabra vacía.

A menos que digamos otra cosa, el alfabeto **A** es finito, los símbolos **u, v, w** se reservan para palabras sobre A y los elementos de A provienen de las letras **a, b, c.**

Un lenguaje **L** sobre un alfabeto **A** es una colección de **palabras sobre A**. Recuerde que $\mathbf{A}^*$ denota todas las palabras sobre A. Así, un lenguaje L es simplemente un subconjunto de A^*.

Ejemplo: Sea A = {a, b}. Algunos lenguajes sobre A son los siguientes:

a) $L_1 = \{a, ab, ab^2, ...\}$ c) $L_3 = \{ a^m b^m / m > 0 \}$

b) $L_2 = \{ a^m b^n / m > 0 , n > 0 \}$ d) $L_4 = b^m a b^n / m \geq 0, n \geq 0 \}$

La descripción verbal de estos lenguajes es:

a) L_1 consta de todas las palabras que empiezan con una **a** seguida de cero o más **b.**
b) L_2 consta de todas las palabras que empiezan con una o más **a** seguidas de una o más **b.**
c) L_3 consta de todas las palabras que empiezan con una o más **a** seguidas por el mismo número de **b**.
d) L_4 consta de todas las palabras que tienen exactamente una **a**.

AUTÓMATAS DE ESTADO FINITO (FSA)

Un autómata de estado finito (finite state automation, FSA) o, simplemente, un autómata M consta de cinco partes:

1) Un conjunto finito (alfabeto) **A** de datos de entrada.
2) Un conjunto finito **S** de estados (internos).
3) Un subconjunto **Y** de **S** (que se denomina estados de aceptación o estados "sí").
4) Un estado inicial $\mathbf{s_0}$ en **S**.
5) Una función **F** de estado siguiente de **S x A** en **S**.

Un autómata M así se denota por M = (A, S, Y, s_0, F) cuando se quieren indicar sus cinco partes.

Ejemplo:

A continuación, definiremos un autómata M con dos símbolos de entrada y tres estados:

1) $A = \{a, b\}$, símbolos de entrada.
2) $S = \{s_0, s_1, s_2\}$, estados internos.
3) $Y = \{s_0, s_1\}$, estados "sí".
4) s_0, estado inicial.
5) La función de estado siguiente $F: S \times A \rightarrow S$ que se define explícitamente en la figura 12-1*a*) o en la tabla de la figura 12-1*b*).

$$F(s_0, a) = s_0,\ F(s_1, a) = s_0,\ F(s_2, a) = s_2$$
$$F(s_0, b) = s_1,\ F(s_1, b) = s_2,\ F(s_2, b) = s_2$$

a)

F	a	b
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_2	s_2

b)

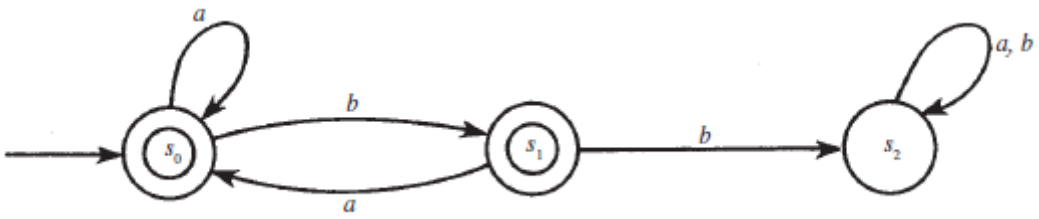
Veremos que hay una forma más práctica de definir un autómata sin enumerar sus cinco partes, es el llamado **diagrama de estado**. O sea, un autómata **M** se puede definir por su diagrama de estado **D = D(M)** en lugar de enumera sus cinco partes constitutivas. El diagrama **D** es una gráfica dirigida etiquetada como sigue:

1) Los vértices de **D(M)** son los estados en **S** y un estado de aceptación se denota por medio de un círculo doble.
2) Hay una flecha (arista dirigida) en **D(M)** del estado $\mathbf{s_j}$ al estado $\mathbf{s_k}$ identificada por una entrada **a** si $\mathbf{F(s_j, a) = s_k}$.

3) El estado inicial s_0 se indica por medio de una flecha especial que termina en s_0 pero que, en cambio, no tiene vértice inicial.

Para cada vértice s_j y cada letra **a** en el alfabeto **A**, hay una flecha identificada por **a** que sale de s_j, por lo tanto, el grado de salida de cada vértice es igual al número de elementos de **A**. Por conveniencia en la notación, una sola flecha identifica todas las entradas que ocasionan el mismo cambio de estado, en lugar de tener una flecha para cada una de tales entradas.

Ahora realizaremos el diagrama de estado del ejemplo anterior:



Obsérvese que tanto **a** como **b** identifican la flecha que va de s_2 a s_2 puesto que $F(s_2, a) = s_2$ y $F(s_2, b) = s_2$, también que el grado de salida de cada vértice es 2, el número de elementos de **A.**

EL AUTÓMATA M DETERMINA EL LENGUAJE L(M).

Cada autómata **M** con alfabeto de entrada **A** define un lenguaje sobre **A**, que se denota con **L(M)** como sigue:

Sea $w = a_1\, a_2 \ldots .\, a_m$ una palabra sobre **A**. Entonces, **w** determina la siguiente ruta en la gráfica del diagrama de estado **D(M)**, donde s_0 es el estado inicial y $F(s_{i-1}, a_i) = s_i$ para $i \geq 1$:

$$P = (s_0, a_1, s_1, a_2, s_2, \ldots\ldots, a_m, s_m)$$

Se dice que **M** reconoce la palabra **w** si el estado final s_m es un estado de aceptación en **Y**. El lenguaje **L (M)** de **M** es la colección de todas las palabras de **A** que **M** acepta.

Ejemplo:

Ahora determinaremos con el diagrama de estado del ejemplo anterior, si el mismo acepta o no las siguientes palabras:

$$w_1 = ababba; \quad w_2 = baab; \quad w_3 = \lambda \text{ la palabra vacía.}$$

Para resolver este dilema recorremos el diagrama de estado con las palabras dadas:

$$P_1 = s_0 \xrightarrow{a} s_0 \xrightarrow{b} s_1 \xrightarrow{a} s_0 \xrightarrow{b} s_1 \xrightarrow{b} s_2 \xrightarrow{a} s_2$$

El estado final en P_1 es s_2 que no está en **Y**, por lo tanto, **w_1 no es aceptada por M.** Veremos ahora que pasa con w_2. Recorriendo el diagrama vemos que:

$$P_2 = s_0 \xrightarrow{b} s_1 \xrightarrow{a} s_0 \xrightarrow{a} s_0 \xrightarrow{b} s_1$$

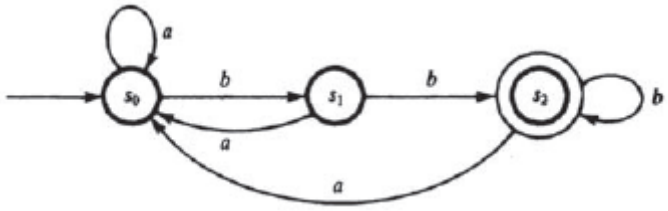
El estado final en P_2 es s_1, que está en **Y**, por lo tanto, **w_2 es aceptada por M.**

Por último, el estado final determinado por w_3 es el instante inicial s_0, puesto que $w_3 = \lambda$ es la palabra vacía. Así, w_3 es aceptada por **M** puesto que $s_0 \in Y$.

Por último, podríamos tratar de dilucidar el lenguaje del autómata visto en el ejemplo anterior. Podemos decir que el lenguaje $L(M)$ consta de todas las palabras de A que no tienen dos **b** consecutivas. Esto se debe a los siguientes hechos:

1) El estado s_2 se introduce si y solo sí hay dos **b** consecutivas.
2) Nunca es posible dejar s_2.
3) El estado s_2 es el único estado de rechazo (no aceptación).

Si ahora queremos hacer un diagrama que solo acepte las palabras de **A** que terminen en dos **b**, tendríamos una gráfica como la siguiente:



Máquinas de Estados Finitos (FSM):

Hasta ahora vimos **autómatas de estados finitos (FSA).** Ahora definiremos realmente como es una **Máquina de Estados finitos (FSM).** La diferencia que esta última nos da un **alfabeto de salida** que puede ser distinto al alfabeto de entrada.

Una **máquina de estados finitos (o maquina secuencial completa) M** consta de seis partes. Recordar que una FSA se divide en cinco partes.

1) Un conjunto finito **A** de símbolos de entrada.
2) Un conjunto finito **S** de estados "internos".
3) Un conjunto finito **Z** de símbolos de salida.
4) Un estado inicial s_0 en **S.**
5) Una función **f** de estado siguiente de **S x A** en **S.**
6) Una función **g** de salida de **S x A** en **Z.**

Cuando se indican las seis partes de una maquina **M** se denota por **M = m(A, S,Z,s_0,f,g).**

Ejemplo: A continuación, se define una máquina de estados finitos M con dos símbolos de entrada, tres estados internos y tres símbolos de salida. Estos datos se muestran en la siguiente figura:

1) $A = \{a, b\}$, 2) $S = \{s_0, s_1, s_2\}$, 3) $Z = \{x, y, z\}$, 4) Estado inicial s_0.

5) Función de estado siguiente $f : S \times A \to S$ que se define con:

$$f(s_0, a) = s_1, \quad f(s_1, a) = s_2, \quad f(s_2, a) = s_0$$
$$f(s_0, b) = s_2, \quad f(s_1, b) = s_1, \quad f(s_2, b) = s_1$$

6) Función de salida $g : S \times A \rightarrow S$ que se define con:

$$g(s_0, a) = x, \quad g(s_1, a) = x, \quad g(s_2, a) = z$$
$$g(s_0, b) = y, \quad g(s_1, b) = z, \quad g(s_2, b) = y$$

Hay dos formas de representar una máquina de estados finitos M en forma breve. Una es mediante **Tabla de estados** de la maquina M y la otra es por medio de una gráfica dirigida etiquetada que se denomina **diagrama de estado** de la maquina M.

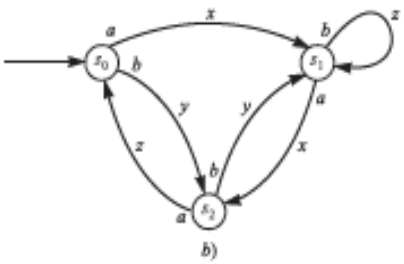
La tabla de estado combina la función **f** de estado siguiente y la función **g** de salida en una sola tabla que representa la función **F : S x A → S x Z** que se define:

$$F(s_i, a_j) = [f(s_i, a_j), g(s_i, a_j)]$$

Por ejemplo, la tabla de estado de la maquina M se muestra en la siguiente figura (a). Los estados se enumeraron a la izquierda de la tabla con el estado inicial primero y los símbolos de salida se muestran en la parte superior de la tabla. Una entrada en la tabla es un para (sk, z_i), donde $s_k = f(s_i, a_j)$ es el siguiente estado y $z_i = g(s_i, a_j)$ es el símbolo de salida. Se supone que no hay símbolos de salida distintos a los que aparecen en la tabla.

F	a	b
s_0	s_1, x	s_2, y
s_1	s_2, x	s_1, y
s_2	s_2, x	s_1, y

a)



b)

El diagrama de estados D = D(M) de una máquina de estados finitos M = M(A, S,Z,s_0,f,g) es una gráfica dirigida etiquetada. Los vértices de D son los estados de M. Además, si:

$$F(s_i, a_j) = (s_k, z_r), \quad \text{o, en forma equivalente,} \quad f(s_i, a_j) = s_k \text{ y } g(s_i, a_j) = z_r$$

Entonces hay un arco (flecha) de s_i a s_k que se identifica por el para a_j , z_i . Suele acostumbrarse a escribir el símbolo de entrada a_i cerca de la base de la flecha (cerca de s_i) y el símbolo z_i, cerca del centro de la flecha. El estado inicial s_0 también se identifica al trazar una flecha adicional hacia s_0. El diagrama de estados de la maquina M del ejemplo se muestra en la figuran (b).

El análisis anterior de una máquina de estados finitos M no muestra la calidad dinámica de M. Suponga que M es una palabra de símbolos de entrada, por ejemplo:

$$u = a_1\ a_2\ \ldots\ldots\ a_m$$

La máquina M "lee" estos símbolos de entrada uno por uno y, en forma simultánea, cambia a través de una secuencia de estados:

$$v = s_0\ s_1\ s_2\ \ldots\ldots\ s_m$$

donde s_0 es el estado inicial, mientras "imprime" una cadena (palabras) de símbolos de salida:

$$w = z_1 z_2 \ldots\ldots z_m$$

En términos formales, el estado inicial s_0 y la cadena de entrada "u" determinan las cadenas "v2 y "w" como sigue, donde i = 1, 2, …., m:

$$s_i = f(s_{i-1}, a_i) \quad y \quad z_i = g(s_{i-1}, a_i)$$

Ejemplo. Consideremos la máquina M de la figura anterior. Suponemos que la entrada es la palabra:

u = a b a a b

La secuencia "v" de estados y la palaba "w" de salida se calculan a partir del diagrama de estado, se empieza en el estado inicial s_0 y se siguen las flechas que están identificadas por los símbolos de entrada como sigue:

$$s_0 \xrightarrow{a,x} s_1 \xrightarrow{b,z} s_1 \xrightarrow{a,x} s_2 \xrightarrow{a,z} s_0 \xrightarrow{b,y} s_2$$

Así se obtienen la siguiente secuencia "v" de estados y la palabra "w" de salida:

$$v = s_0 s_1 s_1 s_2 s_0 s_2 \quad y \quad w = xzxzy$$

Ejemplo: Queremos describir una máquina de estados finitos M capaz de efectuar una suma binaria. Suponemos que ambos números tienen la misma cantidad de dígitos. Si introducimos la siguiente entrada:

$$\begin{array}{r} 1101011 \\ +0111011 \\ \hline \end{array}$$

Entonces se desea que la salida sea la suma binaria 10100110. En este caso, la entrada es la cadena de pares de dígitos a sumar:

$$11, \quad 11, \quad 00, \quad 11, \quad 01, \quad 11, \quad 10, \quad b$$

Donde "b" denota espacios en blanco y la salida debe ser la cadena:

$$0, \quad 1, \quad 1, \quad 0, \quad 0, \quad 1, \quad 0, \quad 1$$

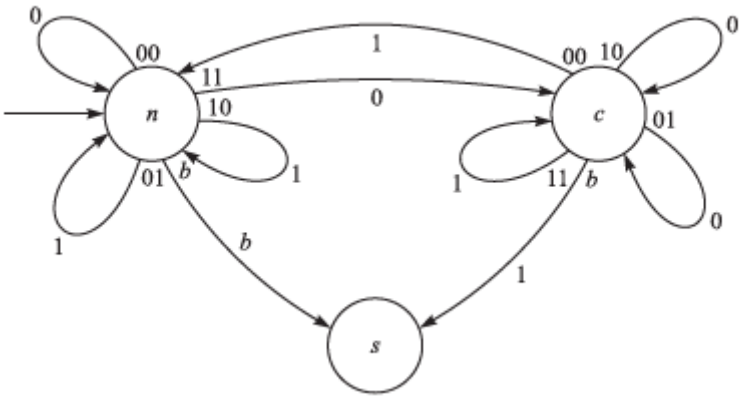
También se desea introducir a la maquina un estado denominado "alto" (stop) cuando la maquina termine la adición.
Los símbolos de entrada y salida son, respectivamente:

$$A = \{00, 01, 10, 11, b\} \quad y \quad Z = \{0, 1, b\}$$

La máquina M que "se construye" tiene tres estados:

$$S = \{\text{llevar } (c), \text{ no llevar } (n), \text{ alto } (s)\}$$

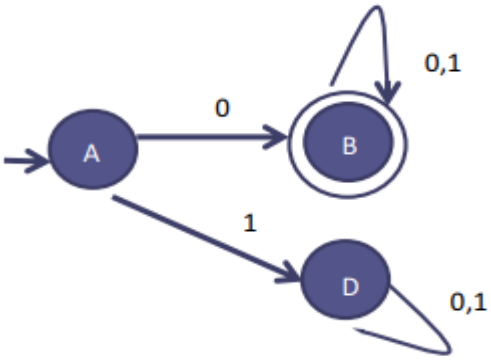
Aquí el estado inicial es "n". La máquina se observa en la siguiente figura:



Algunos ejemplos:

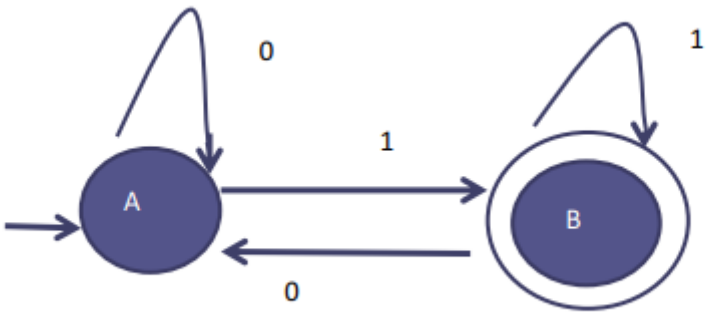
1) Obtener un Autómata Finito que dado el lenguaje definido en el alfabeto ∑ = {0, 1}, indique el conjunto de cadenas que empiezan en “0”.

El diagrama de este autómata es el siguiente:



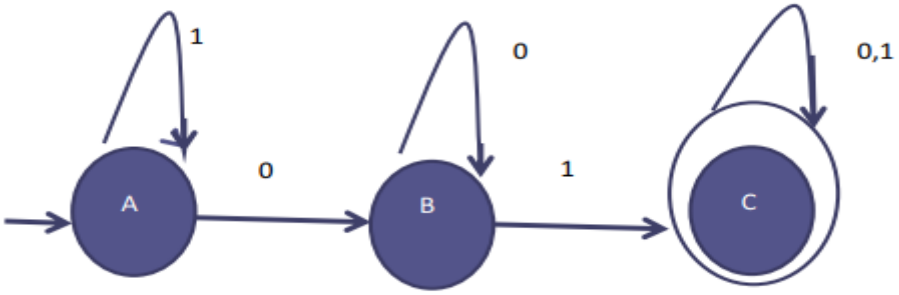
2) Ídem anterior, pero para las cadenas que terminan en “1”.

Solución:



3) Con los mismos datos que los problemas anteriores, ahora queremos detectar las cadenas que contienen a la sub-cadena “01”.

Solución:



4) Con los mismos datos que los problemas anteriores, ahora queremos detectar las cadenas que no contienen a la sub-cadena “01”.

Solución: