Unidad III: Autómatas finitos

Un autómata finito (AF) o máquina de estado finito es un modelo computacional que realiza cómputos en forma automática sobre una entrada para producir una salida.

Este modelo está conformado por un alfabeto, un conjunto de estados y un conjunto de transiciones entre dichos estados. Su funcionamiento se basa en una función de transición, que recibe a partir de un *estado inicial* una cadena de caracteres pertenecientes al alfabeto (la entrada), y que va leyendo dicha cadena a medida que el autómata se desplaza de un estado a otro, para finalmente detenerse en un *estado final* o *de aceptación*, que representa la salida.

La finalidad de los autómatas finitos es la de reconocer lenguajes regulares, que corresponden a los lenguajes formales más simples según la Jerarquía de Chomsky.

3.1 Definición formal

Formalmente, un autómata finito es una 5-tupla (Q, Σ, q_0, δ, F) donde:[6]

- Q es un conjunto finito de estados;
- Σ es un alfabeto finito;
- $q_0 \in Q$ es el estado inicial;
- $\delta: Q \times \Sigma \to Q$ es una función de transición;
- $F \subseteq Q$ es un conjunto de estados finales o de aceptación.

En el comienzo del proceso de reconocimiento de una cadena de entrada, el autómata finito se encuentra en el *estado inicial* y a medida que procesa cada símbolo de la cadena va cambiando de estado de acuerdo a lo determinado por la *función de transición*. Cuando se ha procesado el último de los símbolos de la cadena de entrada, el autómata se detiene en el estado final del proceso. Si el estado final en el que se detuvo es un estado *de aceptación*, entonces la cadena

pertenece al lenguaje reconocido por el autómata; en caso contrario, la cadena no pertenece a dicho lenguaje.

Note que el estado inicial q_0 de un autómata finito siempre es único, en tanto que los estados finales pueden ser más de uno, es decir, el conjunto F puede contener más de un elemento. También puede darse el caso de que un estado final corresponda al mismo estado inicial.

3.2 Clasificación de AF

Los autómatas se pueden clasificar en:

- Deterministas; Cada combinación (estado, símbolo de entrada) produce un solo estado.
- No Deterministas; Cada combinación (estado, símbolo de entrada) produce varios estados y además son posibles las transiciones con λ.

3.3 Conversión de un AFND a AFD

Para convertir un AFD en un AFN que reconozca el mismo lenguaje. Este algoritmo, a menudo es llamado *construcción de subconjuntos*, es útil para simular un AFN por medio de un programa de computadora.

En la tabla de transiciones de un AFN, cada entrada es un conjunto de estados; en la tabla de transiciones de un AFD, cada entrada es tan solo un estado. La idea general tras la construcción AFN a AFD es que cada estado de AFD corresponde a un conjunto de estados del AFN. El AFD utiliza un estado para localizar todos los posibles estados en los que puede estar el AFN después de leer cada símbolo de la entrada. Es decir, después de leer la entrara a_1, a_2,. ..a_n, el AFD se encuentra en un estado que representa al subconjunto T de los estados del AFN alcanzables desde el estado de inicio del AFN a lo largo de algún camino etiquetado con a_1,

a_2,. ..a_n. El número de estados de AFD puede ser exponencialmente en el número de estados del AFN pero en la práctica este peor caso ocurre raramente.

Algoritmo *(Construcción de subconjuntos)* Construcción de un AFD a partir de un AFN.

Entrada. Un AFN N

Salida. Un AFD D que acepta el mismo lenguaje

Método. El algoritmo construye una tabla de transiciones tranD para D. Cada estado del AFD es un conjunto de estados del AFN y se construye tranD de modo que D simulará "en paralelo" todos los posibles movimientos que N puede realizar con una determinada cadena de entrada.

Se utilizan las operaciones de la siguiente tabla para localizar los conjuntos de los estados del AFN (s representa un estado del AFN, y T un conjunto de estados del AFN).

Operación	Descripción
Cerradura-□ (s)	Conjunto de estados del AFN alcanzables desde el estado s del AFN con transiciones □ solamente
Cerradura-□ (T)	Conjunto de estados del AFN alcanzables desde algún estado s en T con transiciones □ solamente.
mueve(T, a)	Conjunto de estados del AFN hacia los cuales hay una transición con el símbolo de entrada a desde algún estado s en T del AFN

Antes de detectar el primer símbolo de entrada N se puede encontrar en cualquiera de los estados del conjunto *cerradura*-□ (S_0), donde S_0 es el estado de inicio de N. Supóngase que exactamente los estados del conjunto T son alcanzables desde S_0 con una secuencia de símbolos de entrada, y sea a el

siguiente símbolo de entrada. Al ver a, N puede trasladarse a cualquiera de los estados del conjunto *mueve*(T, a). Cuando se permiten transiciones-□ , N puede encontrarse cualquiera de los estados de *cerradura*-□ (T, a), después de ver la a.

Se construyen estadosD, el conjunto de estados de D, y tranD, la tabla de transiciones de D, de la siguiente forma. Cada estado de D corresponde a un conjunto de estados de AFN en los que podría estar N después de leer alguna secuencia de símbolos de entrada, incluidas todas las posibles transiciones-□ anteriores o posteriores a la lectura de símbolos. El estado de inicio de D es *cerradura-□ (*S_0). Se añaden los estados y las transiciones a D utilizando el algoritmo siguiente. Un estado de D es un estado de aceptación si es un conjunto de estados de AFN que contenga al menos un estado de aceptación de N.

al inicio, cerradura-□ (S_0*) es el único estado dentro de estadosD y no está marcado;*

while haya un estado no marcado T en estadosD do begin

marcar T;

for cada símbolo de entrada a do begin

U:= cerradura-□ (mueve(T,a));

if U no está en estadosD then

añadir U como estado no marcado a estadosD;

tranD[T, a]:= U;

end;

end;

El cálculo de cerradura- (T) es un proceso típico de búsqueda en un grafo de nodos alcanzables desde un conjunto dado de nodos. En este caso, los estados de T son el conjunto dado de nodos. En este caso, los estados de T son el conjunto dado de nodos, y el grafo está compuesto solamente por las aristas del AFN etiquetadas por. Ejemplo:

La siguiente figura muestra otro AFN N aceptando el lenguaje (a | b)*abb. Se aplica el algoritmo anterior a N.

El estado de inicio del AFD equivalente es *cerradura-□ (0)*, que es A = {0, 1 , 2, 4, 7 }, puesto que estos son exactamente los estados alcanzables desde el estado 0 por un camino en el que todas las aristas están etiquetadas por □ . Obsérvese que un camino puede no tener aristas, de modo que 0 es alcanzado desde sí mismo por dicho camino.

Aquí, el alfabeto de símbolos de entrada es {a, b}. El algoritmo anterior indica que hay que marcar A y después calcular *cerradura-□ (mueve(A, a))*.

Primero se calcula *mueve (A, a)*, el conjunto de estado de N que tiene transiciones en a desde miembros de A. Entre los estados 0,1,2,4 y 7 sólo 2 y 7 tienen dichas transiciones, a 3 y a 8, de modo que:

cerradura-□ (mueve({0,1,2,4,7}.a)) = cerradura-□ (3,8})= {1,2,3,4,6,7,8}

Este conjunto se denominará B. Así, tranD[A,a] = B.

Entre los estados de A, solo 4 tienen una transición en b a 5, de modo que el AFD tiene una transición en b desde A a

C = *cerradura-□ ({5}) = {1,2,4,5,6,7}.*

Por tanto, tranD[A,b] = C

Si se continua este proceso con los conjuntos B y C, ahora sin marcar, finalmente, se llegará al punto en que todos los conjuntos que son estados del AFD estén marcados. Esto es cierto porque "solo" hay 2^{11} subconjuntos distintos de un conjunto de 11 estados, y un conjunto, una vez marcado, queda marcado para siempre.

Los cinco conjuntos de estados realmente construidos son:

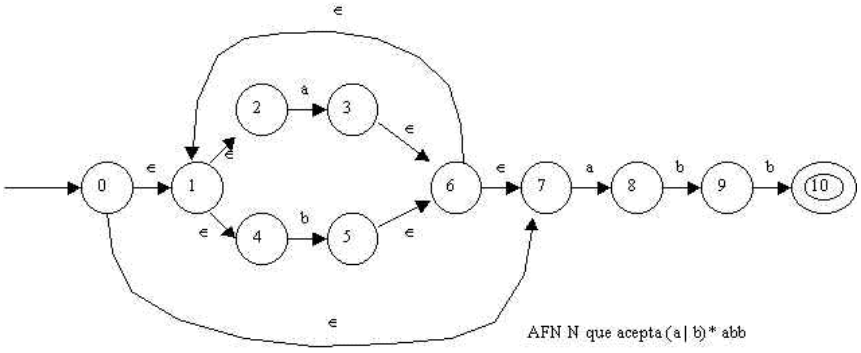
A = {0,1,2,4,7 }

B = {1,2,3,4,6,7,8 }

C = {1,2,4,5,6,7 }

D= {1,2,4,5,6,7,9 }

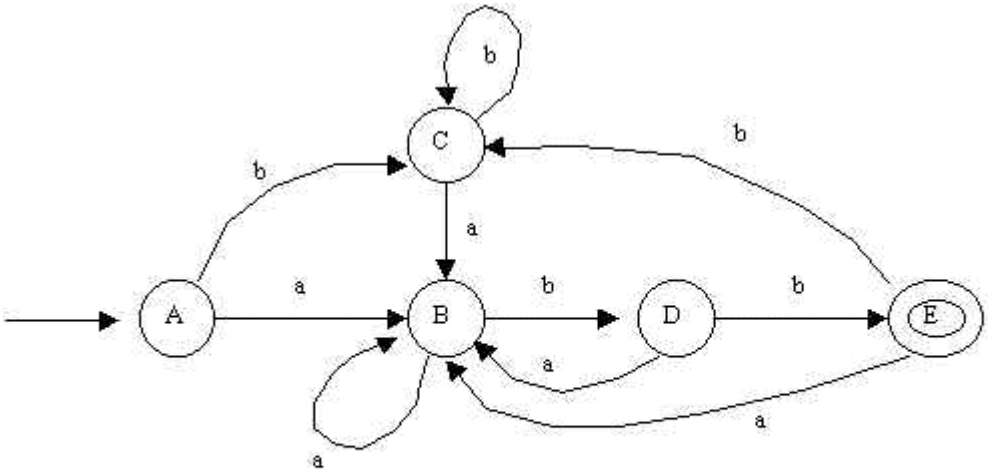
E= {1,2,4,5,6,7,10 }



El estado A es el estado de inicio, y el estado E es el único estado de aceptación. La tabla de transiciones completa tranD se muestra a continuación.

Estado	*Símbolo de entrada*	
	a	*b*
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

En la siguiente figura, se muestra un grafo de transiciones para el AFD resultante.



AFD resultante de aplicar el algoritmo anterior

3.4 Representación de ER usando AFND

3.5 Minimización de estados en un AF

3.6 Aplicaciones (definición de un caso de estudio)