



Tema
Autómatas de Estado Finito

Facultad: Ingeniería
Escuela: Computación
Asignatura: Teoría de
Lenguajes de Programación

I. OBJETIVOS

- Conocer las características básicas de un Autómata de Estado Finito Determinista.
- Crear algunas secuencias de cadenas evaluadas por un FDA (Autómata Finito Determinista).

II. INTRODUCCIÓN

Autómatas de estado finito

Alfabeto.- conjunto de símbolos finito no vacío a partir del cual estarán construidas las cadenas.

Flujo de entrada.- Cada cadena se va a presentar como una **secuencia de símbolos** analizándose de uno en uno y **de izquierda a derecha**. Esta secuencia puede llegar a ser infinita.

Cuando llega un símbolo del flujo de entrada, este se reconoce si se cambia de un estado a otro o bien se permanece en el mismo, si la transición (arco etiquetado) es posible con ese símbolo. **El nuevo estado dependerá únicamente del estado actual y del símbolo que se recibe.**

Un **autómata finito determinista** consiste en un dispositivo que puede estar en un estado de entre un número finito de los mismos; uno de ellos será el estado inicial y por lo menos uno será estado de aceptación. Tiene un flujo de entrada por el cual llegan los símbolos de una cadena que pertenecen a un alfabeto determinado. Se detecta el símbolo y dependiendo de este y del estado en que se encuentre hará **una transición a otro estado o permanece en el mismo**. El mecanismo de control (programa) es que determina cual es la transición a realizar.

La palabra **finito** se refiere a que hay un **número finito de estados**.

La palabra **determinista** es porque el mecanismo de control (programa) no debe tener ambigüedades, es decir, en **cada estado** solo se puede dar **una y solo una** (ni dos ni ninguna) **transición para cada símbolo** posible. El autómata **acepta la cadena** de entrada **si** la máquina **cambia a un estado de aceptación después** de leer el **último símbolo de la cadena**. **Si** después del último símbolo la máquina **no queda en estado de aceptación, se ha rechazado la cadena**.

Si la máquina llega al final de su entrada antes de leer algún símbolo la entrada es una **cadena vacía** (cadena que no contiene símbolos) y la representaremos con ϕ . Solo aceptará ϕ si su estado inicial es de aceptación.

Definición formal

Un autómata finito determinista consiste en una quintupla (S, Σ, d, i, F) donde:

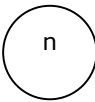

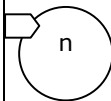
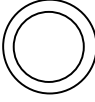
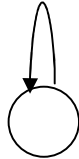
- S es un conjunto finito de estados
- Σ es el alfabeto de la máquina
- d es una **función** (llamada de **transición**) de $S \times S$ en S ($d: (S \times S) \rightarrow S$)
- i (un elemento de S) es el estado inicial
- F (un **subconjunto de S**) es el conjunto de los estados de aceptación.

Dados dos estados, p y q , de S y cualquier elemento del alfabeto, x de S , la función transición es $d(p, x) = q$ si y solo si se puede pasar de un estado p a uno q al leer el símbolo x . Más formalmente, el autómata (S, S, d, i, F) acepta la cadena no vacía de la forma $x_1x_2...x_n$ si y solo si existe un camino a través de una serie de estados $s_0, s_1, ..., s_n$ tal que $s_0 = i$ (uno es inicial) y $s_n \in F$ (al menos uno es de aceptación) y para cada entero j de 1 a n , y cada símbolo x del alfabeto, se da la función $d(s_{j-1}, x_j) = s_j$.

Diagrama de transiciones determinista

Estará caracterizado porque debe estar totalmente definido: **para cada estado** solo debe salir **un arco** y solo uno **para cada símbolo** (el autómata no puede determinar la transición en el caso de que haya dos arcos con el mismo símbolo o no haya ninguno).

Símbolos

Conceptualización	Símbolo
Un estado se representa con	
Para indicar la transición de un estado a otro se utiliza	
El estado inicial de una cadena o del autómata se representa con	
Un estado final o de aceptación se representa con	
Si una transición retorna al mismo estado (transición que va de un estado al mismo estado) se representa como	

Ejemplo

El siguiente autómata (Fig. 1), el cual, para el alfabeto $S = \{a, b, c\}$ reconoce la cadena c , la cadena a , las cadenas que empiezan por a y acaban en a o en b y las que empiezan por a , seguidas de una serie de a ó de b y acaban en c .

Su diagrama será:

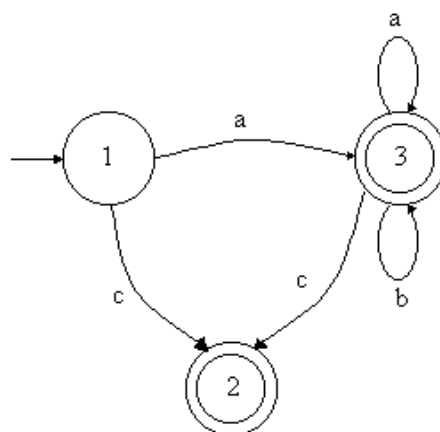


Fig. 1

III. REQUERIMIENTOS:

1. Materiales y equipo a utilizar

DESCRIPCIÓN	CANTIDAD
Guía de laboratorio	1
JFLAP	1
Compilador de Visual C++	1
Disco Flexible (1.44 Mb)	2

2. Estudiar la guía, para evaluación al iniciar el laboratorio.

IV. PROCEDIMIENTO

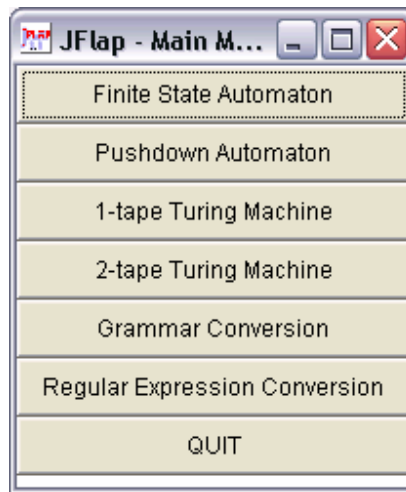
PARTE I (Creación de un Autómata Simple)

- Ingrese a JFLAP. Utilizar el acceso directo ubicado en el escritorio. El resultado es la siguiente ventana:



Acceso directo
a jflap.bat

El resultado es la siguiente



- Clic al botón Finite State Automaton. Y se deberá cargar la siguiente ventana (Fig. 2):



Fig. 2

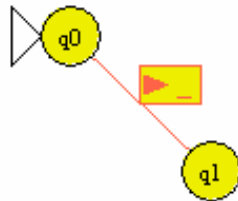
- Cree el diagrama (autómata) del ejemplo presentado arriba (figura 1) siguiendo las siguientes instrucciones:

1. Para crear un estado presione la tecla **shift** y el botón izquierdo del mouse. Por defecto el primer estado creado es el estado inicial.




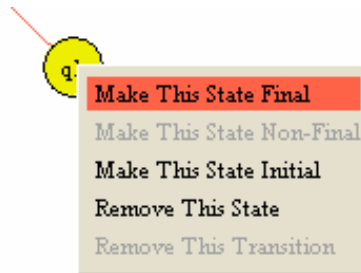
Estado inicial

2. Para crear una transición de un estado a otro, ubíquese en un estado, presione el botón izquierdo del mouse, y sin soltar el botón arrastre el puntero del mouse hasta el siguiente estado.

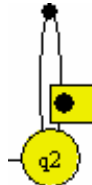


Transición entre estados

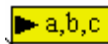
- Para ingresar un valor (alfabeto) entre un estado y otro, colóquese en  y digite el contenido.
- Para establecer un nodo como terminal (estado final), ubíquese en el estado y presione el botón derecho del mouse y sin soltarlo arrastre el puntero del mouse sobre el menú colgante hasta donde dice **make this state final**.



- Para crear una transición de un estado hacia el mismo, ubíquese en el estado y haga un clic con el botón izquierdo del mouse.

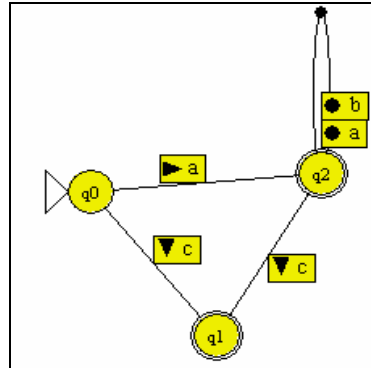


- Para ingresar más de un valor (alfabeto) entre un estado y otro, digite los valores separados por comas.



- Para remover (quitar) un estado, presione el botón derecho del mouse y seleccione **Remove this state** del menú colgante
- Para mover un estado, presione la tecla ALT y luego con el puntero del mouse movilice el estado a la nueva posición.
- Para guardar el autómata, clic al menú **File – Save**, en la ventana indicar la carpeta de ubicación y luego el nombre del archivo. La extensión la aplicación la asigna (la extensión es FA).

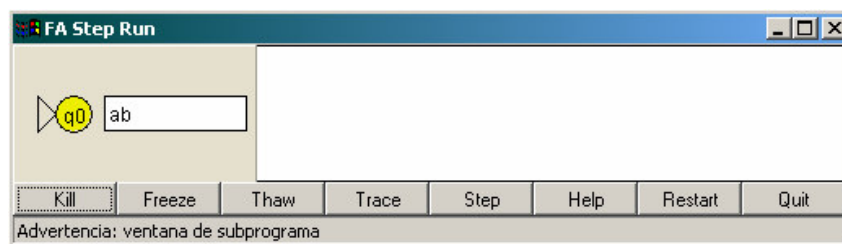
- El autómata deberá quedar de la siguiente forma:



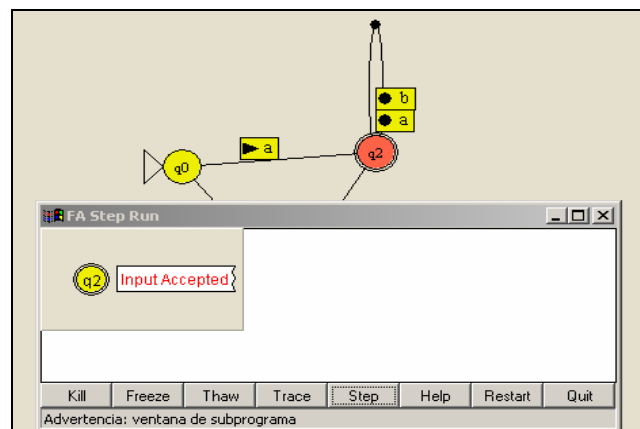
- Para probar el autómata digite una cadena como “ab” en la caja de texto **Input String** tal como se muestra a continuación:

Input String:

- A continuación seleccione **Step Run** del menú **Run** para evaluar carácter por carácter de la cadena digitada.
- Presione **Step** para analizar la cadena de estado a estado (carácter por carácter).



- Dicha cadena deberá ser “aceptada” si se llega al estado final cumpliendo con las reglas del alfabeto (gramática) definida por el autómata (de estado a estado). Si un carácter no forma parte del alfabeto del autómata (si no es reconocido) o no cumple con las reglas entre un estado y otro, entonces la cadena es “rechazada”.

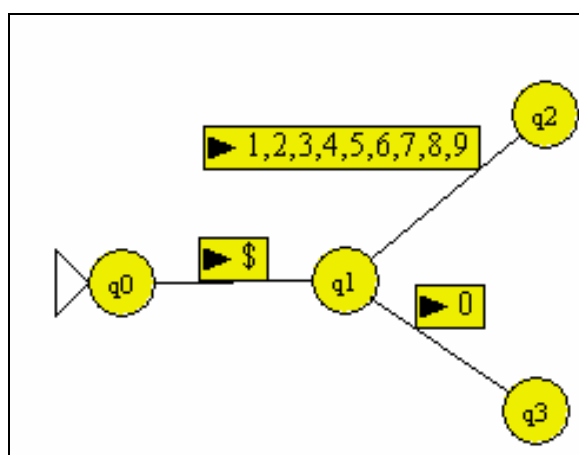


Actividad 1

Cree un autómata que contenga en su alfabeto $S = \{ \$, ., 0, 1, 2, 3 \dots 9 \}$, el cual reconoce la cadena con el siguiente formato: $\$xx.xx$ y $\$0.xx$, es decir, una cantidad en dólares que comience con \$, seguido de dos dígitos con dos decimales separados por un punto, o bien, una cantidad que comience con \$, seguido de un cero con dos decimales separados por un punto. ($\$10.99$ o $\$0.45$ respectivamente).

- Ayuda:**

Su autómata deberá comenzar con la siguiente secuencia:



Actividad 2

En base al autómata creado anteriormente, generalice la expresión $\$xx.xx$ de tal forma que sea aceptada una cadena con cualquier cantidad en dólares, ya sea una cantidad entera o una cantidad entera con dos decimales $\$xxxxxxx\dots n.xx$.

Es decir el autómata aceptará una cantidad entera de dólares, por ejemplo **\$100000** y también una cantidad con dos decimales, por ejemplo **\$900000.99**. Además su autómata siempre deberá permitir el formato $\$0.xx$ (**\$0.95**).

- Ayuda:**

Su autómata debe incluir 2 estados finales ($q2$ y $q6$), uno para el formato con números enteros (cantidad entera de dólares) y otro para el formato con decimales (cantidad entera con dos decimales de dólares).

El autómata debe tener como máximo 7 estados ($q0$ a $q6$), incluyendo los 2 estados finales y el inicial.

Actividad 3

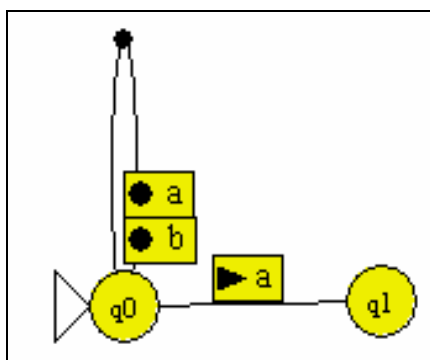
Construya un autómata que cumpla con las características presentadas por la tabla de transiciones siguiente:

Estado	Símbolo de Entrada	
	a	b
0	{0,1}	{0}
1	-	{2}
2	-	{3}

Donde {0,1} indica el estado 0 y 1, y estado final es {3}.

- Ayuda:**

El autómata deberá comenzar con la siguiente secuencia:



El autómata creado se le llama “autómata finito no determinista”, la única diferencia con los anteriores está en que en la transición **en un estado** determinado puede haber, **para un mismo símbolo, más de un arco o no haber ninguno** (Para q0 existen 3 arcos o transiciones).

Para hacer un programa de un autómata no determinista se podría hacer probando todas las rutas una a una hasta que se terminasen o se aceptase la cadena, lo cual no es eficiente por la cantidad de memoria necesaria (crece exponencialmente con el número de estados).

Para analizar una cadena en un autómata finito no determinista **vamos a seguir todas las transiciones posibles en paralelo** (atravesamos todas las rutas a la vez); la cadena **se aceptará** si alguna de las **rutas acaba en un estado de aceptación**. Nuestro **estado en un momento dado** ya no **se hallará determinado** por un estado del diagrama de transiciones, sino **por la colección de los estados actuales de todas las rutas posibles**.

El autómata finito no determinista creado anteriormente aceptaría las cadenas de entrada:
abb, aabb, babb, aaabb

PARTE II (Simulación de un Automata Simple en Lenguaje C/C++)

En esta parte usted simulara el autómata finito resuelto en la Actividad 2, mediante la creación de un pequeño programa en C/C++.

A continuación se presenta el pseudocódigo asociado al autómata creado en la Actividad 2:

```
/* Este pseudocodigo asume que la funcion recibe la cadena a evaluar por
el automata*/
```

Variables:

```
estado:entero;    -- estado puede ser S0, S1, S2, S3, S4, S5, S6
next_car: caracter; -- char next_car
lexema : string;  -- lexema almacena caracter por caracter de la
                  -- cadena a evaluar ( char lexema[80] )
```

Funcion Cantidad_en_Dolar(Cadena) --funcion void

```
estado = S0;
```

Repetir

Conseguir el siguiente carácter --next_car=Cadena[]

Agregar al lexema el siguiente carácter -- lexema[] = next_car

En CASO de (estado) sea

S0: --estado inicial

Si next_car es "\$" entonces

estado = S1;

sino

exit_Repetir; -- salir de Lazo Repetir

fin_si

S1: --"0" o "1-9"

Si next_car es un digito entonces --isdigit(nextcar)

Si es cero entonces

estado = S3;

sino

estado = S2;

sino

exit_Repetir;

fin_si

S2: --"1-9"+digito+

Si next_car es un digito entonces

estado=S2;

sino Si next_car es "." entonces

estado=S4;

Sino

Exit_Repetir;

fin_si

```
S3: --"0"+
    Si next_car es "." entonces
        estado=S4;
    sino
        exit_Repetir;
    fin_si

S4: --"0" + "." + ; "1-9"+ digito + "."+
    Si next_car es un digito entonces
        estado=S5;
    sino
        exit_Repetir;
    fin_si

S5: --"0" + "." + digito+ ; "1-9"+ digito+ "." + digito+
    Si next_car es un digito entonces
        estado=S6;
    sino
        exit_Repetir;
    fin_si

S6: --estado final
    Si no es fin de cadena entonces -- next_car!=='\0'
        estado=Se; --estado de error
        exit_Repetir;
    fin_si
Fin_CASO
Repetir_hasta_que (next_car sea carácter nulo) --next_car=='\0' o fin de cadena

Si estado es S2 o S6 entonces
    Agregar al lexema el carácter nulo --lexema[]='\0'
    Imprimir "Cadena Aceptada"
sino
    Imprimir "Cadena Rechazada"
fin_si

Fin_Funcion
```

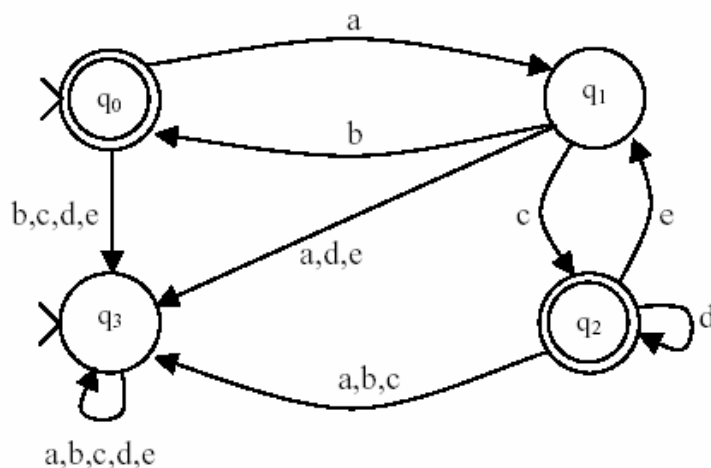
Actividad

Inicie Visual C++ y cree un programa que permita simular el autómata de la Actividad 2, utilizando como referencia el pseudocódigo anterior.

- **Ayuda:**
Utilice como referencia algunas de las **funciones descritas en la guía No.2 por ejemplo**: isdigit(), getchar() y getc().

V. INVESTIGACIÓN Y EJERCICIOS COMPLEMENTARIOS

- Investigar sobre otra(s) herramienta(s) para evaluar autómatas. Características y Requerimientos para su funcionamiento.
- Definición de autómatas: Finito Determinista, Finito no Determinista, Autómata de pilas
- En qué consiste el análisis léxico. Y escriba algunas de las funciones del analizador léxico
- Describa en forma general las características del analizador léxico de C.
- Probar el siguiente autómata (e indique las cadenas de prueba que utilizo), definir la tabla de transiciones y que tipo de autómata es.



VI. FUENTES DE CONSULTA

- ☒ Fundamentos de Compiladores. Cómo Traducir al Lenguaje de Computadora. Karen A. Lemone. Editorial Continental.
- ☒ <http://www.cs.duke.edu/~rodger/tools/tools.html>
- ☒ <http://www-csi.mty.itesm.mx/~sconant/CB00841/Material.html>