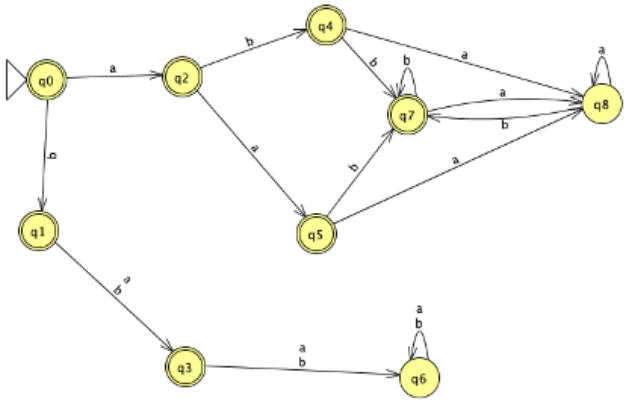
LENGUAJES FORMALES

Tarea No.: *6*	Fecha: 07/09/21
Alumno: Cordero Hernández Marco Ricardo	Id alumno: 727272
Palabras clave: Programación, autómatas, diseño, AFD.	

Objetivo general.

Simular en específico el siguiente autómata finito determinista (AFD).



Programa

Entrada: cadena de entrada w. Ejemplo : w= abb

Salida: mostrar la secuencia de estados visitados para procesar la cadena e indicar si es aceptada o no por el autómata. Ejemplo : q0/q2/q4/q7.Cadena aceptada.

Descripción algoritmo.

Se crea una clase contenedora del autómata, en donde el inicializador contiene los estados, su valor (normal o final) y los estados a los que pasará dependiendo del carácter que se ingrese; también, se define el alfabeto en la misma sección.

Enseguida, se declara la función privada de validación para la cadena introducida, en donde se verifica si la cadena introducida es válida para el autómata. Esta función es invocada únicamente dentro de la siguiente función.

Finalmente, se declara la función para probar la cadena introducida, en donde se construye el resultado de manera iterativa haciendo uso de los estados definidos inicialmente.

La manera en la que esto se integra es mediante una solicitud al usuario, el cual ingresará una cadena de texto; todas las validaciones y procedimientos necesarios, como ya se detalló, están contenidos en la clase.

Casos de prueba.

1. **abbbbb**

```
Introduce tu cadena: abbbbb
q0/q2/q4/q7/q7/q7/q7. Cadena aceptada
```

2. **aba**

```
Introduce tu cadena: aba
q0/q2/q4/q8. Cadena rechazada
```

3. **bababbb**

```
Introduce tu cadena: bababbb
q0/q1/q3/q6/q6/q6/q6/q6/q6. Cadena rechazada
```

4. **baaabab**

```
Introduce tu cadena: baaabab
q0/q1/q3/q6/q6/q6/q6/q6. Cadena rechazada
```

Código

```python
# IS727272 - Cordero Hernández Marco Ricardo - Tarea 6
class automata:
    def __init__(self):
        self.q0 = (True,"q2","q1")
        self.q1 = (True,"q3","q3")
        self.q2 = (True,"q5","q4")
        self.q3 = (True,"q6","q6")
        self.q4 = (True,"q8","q7")
        self.q5 = (True,"q8","q7")
        self.q6 = (False,"q6","q6")
        self.q7 = (True,"q8","q7")
        self.q8 = (False,"q8","q7")

        self.l = "ab"

    def __validar(self,w):
        for i in w:
            if(i not in self.l):
                return False
        else:
            return True

    def test(self, w):
        if(not self.validar(w)):
```

```
            return -1

        estado = self.q0
        estados = "q0"

        for i in range(len(w)):
            if(w[i] == "a"):
                estados += "/" + estado[1]
                estado = eval("self." + estado[1])
            elif(w[i] == "b"):
                estados += "/" + estado[2]
                estado = eval("self." + estado[2])

        return ("%s. Cadena %s" % (estados, "aceptada" if(estado[0]) else "rechazada")
)

print(automata().test(input("Introduce tu cadena: ")))
```

Conclusiones.

Al desarrollar un autómata predefinido con estados “mapeados” previamente, se puede percatar como realizar estas pruebas manuales sin la ayuda de herramientas como JFLAP puede llegar a ser muy tedioso. Habría que pensar en un diseño en donde se permita introducir estados variables y demás funcionalidades, no necesariamente con interfaz gráfica.
También, hay que tener en cuenta que los conocimientos que la mayoría del curso poseemos, por más básicos que sean, han permitido el desarrollo de aplicaciones como la de esta tarea, algo que posiblemente no sería posible para alguien interesado únicamente en teoría matemática o fundamentos de ciencias computacionales.

A manera de prueba, a continuación se demuestra la cadena vacía y su permanencia en el estado inicial:

```
Introduce tu cadena:
q0. Cadena aceptada
```

En este caso, la cadena vacía es aceptada puesto que permanece en q0, es decir y como ya se había mencionado, el estado inicial.

Bibliografía.

Para esta tarea no se hizo uso de ninguna referencia bibliográfica.