INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

MATERIA: Teoría Computacional.

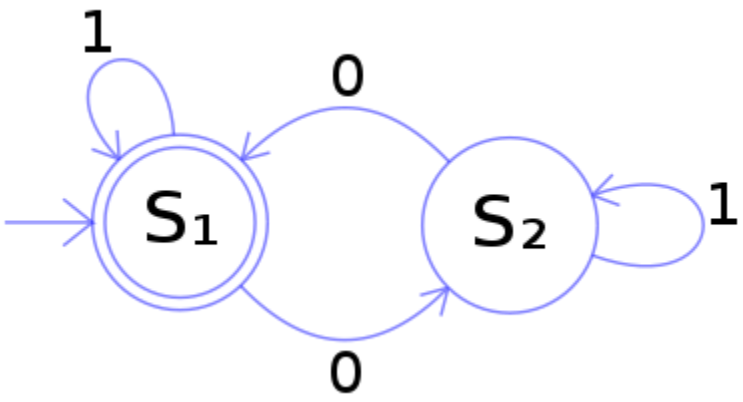
PRÁCTICA: Práctica 4. Autómata Finito Determinista.

ALUMNO: Meza Vargas Brandon David.

PROFESOR: Jorge Luis Rosas Trigueros.

FECHA PRÁCTICA: 20-nov-2020

FECHA DE ENTREGA: 27-nov-2020



MARCO TEÓRICO

Autómatas Finitos Deterministas (AFD)

Un AFD tiene un conjunto finito de estados y un conjunto finito de símbolos de entrada. El término "determinista" hace referencia al hecho de que para cada entrada sólo existe uno y sólo un estado al que el autómata puede hacer la transición a partir de su estado actual. Un estado se diseña para que sea el estado inicial, y cero o más estados para que sean estados de aceptación. Una función de transición determina cómo cambia el estado cada vez que se procesa un símbolo de entrada.

Un autómata finito determinista consta de:

1. Un conjunto finito de estados, a menudo designado como Q.
2. Un conjunto finito de símbolos de entrada, a menudo designado como Σ
3. Una función de transición que toma como argumentos un estado y un símbolo de entrada y devuelve un estado. la función de transición se designa habitualmente como δ. En nuestra representación gráfica informal del autómata, δ se ha representa mediante arcos entre los estados y las etiquetas sobre los arcos. Si q es un estado y a es un símbolo de entrada, entonces δ(q,a) es el estado p tal que existe un arco etiquetado a que va desde q hasta p.
4. Un estado inicial, uno de los estados de Q.
5. Un conjunto de estados finales o de aceptación F. El conjunto F es un subconjunto de Q.

La representación más sucinta de un AFD consiste en un listado de los cinco componentes anteriores. Normalmente, en las demostraciones, definiremos un AFD utilizando la notación de "quíntupla" siguiente: $A = (Q, \Sigma, \delta, q_0, F)$

Donde:

A: es el nombre del AFD.

Q: es un conjunto finito de estados.

∑: son los símbolos de entrada.

q0: es el estado inicial.

F: es el conjunto de estados finales.

δ: es la función de transición.

Tablas de transiciones

- Cada fila corresponde a un estado q ∈ Q
- El estado inicial se procede del símbolo →
- Cada estado final se procede del símbolo #
- Cada columna corresponde a un símbolo de entrada $a \in V$
- En la posición (q, a) está el estado que determine $\delta(q, a)$

Una tabla de transiciones es una representación tabular convencional de una función, como por ejemplo δ, que toma dos argumentos y devuelve un valor. Las filas de la tabla corresponden a los estados y las columnas a las entradas. La entrada para la fila correspondiente al estado q y la columna correspondiente a la entrada a es el estado δ (q,a).

Ejemplo:

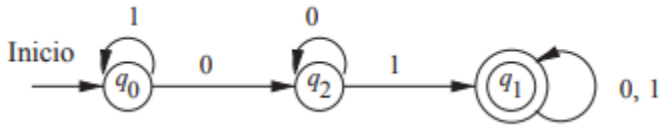


Figura 1. Diagrama de transiciones de un AFD que acepta todas las cadenas que contienen la subcadena 01.

	0	1
→ q_0	q_2	q_0
*q_1	q_1	q_1
q_2	q_2	q_1

Figura 2. Tabla de transiciones del AFD de la figura 1.

La tabla de transiciones correspondiente a la función δ del ejemplo se muestra en la Figura 2. También pueden verse otras dos características de una tabla de transiciones. El estado inicial se marca mediante una flecha y los estados de aceptación mediante un asterisco. Dado que es posible deducir los conjuntos de estados y los símbolos de entrada fijándose en los encabezamientos de las filas y las columnas, podemos obtener a partir de la tabla de transiciones toda la información necesaria para especificar el autómata finito de forma unívoca.

MATERIAL Y EQUIPO

Para la realización de esta práctica son necesarias las siguientes herramientas:

- Un equipo de cómputo que cumpla con los requerimientos para el uso del lenguaje de programación Python.
- Tener instalado el lenguaje de programación Python.
- Contar con un IDE para programar con Python, cualquiera es útil.

DESARROLLO

El desarrollo de esta práctica consistió en diseñar AFD's y codificarlos en el lenguaje de programación Python.

En primer lugar, el profesor dio un ejemplo;

L(M)= {cadenas que tengan un #par de b's}

El AFD para este lenguaje ya lo habíamos revisado en clase, el cual es el mostrado en la figura 3;

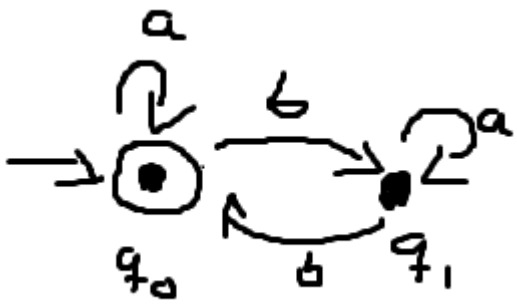


Figura 3. AFD que acepta #par de b's.

Una vez sabiendo cual es el diagrama de transiciones correspondiente al lenguaje, podemos codificarlo, iremos parte por parte explicando el código, esto solo para este ejemplo, pues los ejercicios propuestos son muy similares y se explicará de forma general.

```
Q=['q0','q1']
sigma=['a','b']
s='q0'
F=['q0']
```

Figura 4. Elementos del AFD.

En la figura 4 se muestra la parte del código donde definimos los elementos del AFD, donde:

Q: conjunto finito de estados, en este caso q0 y q1.

sigma: conjunto finito de símbolos, en este caso a y b.

s: es el estado de inicio, en este caso q0.

F: estado de aceptación, en este caso q0.

```
Ejemplos_L=['', 'bb', 'abb', 'aaa']
Ejemplos_Lc=['b', 'ab', 'ba', 'bbb']
```

Figura 5. Ejemplos de cadenas aceptadas y no aceptadas.

En la figura 5 vemos ejemplos de cadenas que caen en el lenguaje, es decir, aquellas cadenas que son aceptadas, y vemos ejemplos del complemento, es decir, aquellas cadenas que no son aceptadas, estos ejemplos nos servirán más adelante.

```
def transicion(estado,sigma):
    #print("estado=",estado,"sigma",sigma)
    estado_siguiente=delta[(estado,sigma)]
    #print("estado siguiente=",estado_siguiente)
    return estado_siguiente
```

Figura 6. función que hace las transiciones entre estados.

En la figura 6 vemos una función que es la encargada de hacer las transiciones entre los distintos estados de nuestro AFD, vemos dos líneas comentadas, estas nos ayudan a llevar un control de que se está haciendo.

```
for w in Ejemplos_L:
    estado=s
    for sigma in w:
        estado=transicion(estado,sigma)

    if estado in F:
        print(w, "es aceptada")
    else:
        print(w, "no es aceptada")
```

Figura 7. recorrido para determinar cadenas aceptadas y rechazadas.

En la figura 7 podemos ver que se recorren las cadenas dentro de Ejemplos_L y se van realizando las transiciones gracias a la función codificada en la figura 6, de esta forma podemos verificar que si el estado al que se llegó pertenece al estado de aceptación definido en F para saber si la cadena es aceptada, si el estado al que se llegó no es de aceptación, entonces la cadena no es aceptada.

```
for w in Ejemplos_Lc:
    estado=s
    for sigma in w:
        estado=transicion(estado,sigma)

    if estado in F:
        print(w, "es aceptada")
    else:
        print(w, "no es aceptada")
```

Figura 8. recorrido para determinar cadenas aceptadas y rechazadas.

En la figura 8 vemos que se hace el mismo procedimiento que lo que vemos en la figura 7, pero ahora tomamos Ejemplos_Lc.

En la figura 9, podemos ver el código completo, y en la figura 10, la salida que nos arroja.

```
#AFD
#L(M)={cadenas que tengan un #par de b's}

Q=['q0','ql']
sigma=['a','b']
s='q0'
F=['q0']

delta= { ('q0' , 'a'):'q0',
         ('q0' , 'b'):'ql',
         ('ql' , 'a'):'ql',
         ('ql' , 'b'):'q0'
        }

Ejemplos_L=['', 'bb', 'abb', 'aaa']
Ejemplos_Lc=['b', 'ab', 'ba', 'bbb']

def transicion(estado,sigma):
    #print("estado=",estado,"sigma",sigma)
    estado_siguiente=delta[(estado,sigma)]
    #print("estado siguiente=",estado_siguiente)
    return estado_siguiente

#w='baab'

for w in Ejemplos_L:
    estado=s
    for sigma in w:
        estado=transicion(estado,sigma)

    if estado in F:
        print(w, "es aceptada")
    else:
        print(w, "no es aceptada")

for w in Ejemplos_Lc:
    estado=s
    for sigma in w:
        estado=transicion(estado,sigma)

    if estado in F:
        print(w, "es aceptada")
    else:
        print(w, "no es aceptada")
```

Figura 9. Código completo.

```
 es aceptada
bb es aceptada
abb es aceptada
aaa es aceptada
b no es aceptada
ab no es aceptada
ba no es aceptada
bbb no es aceptada
>>>
```

Figura 10. salida del código.

Ejercicio 1

El primer ejercicio es generar un AFD que acepte las cadenas generadas por la expresión regular a*b*.

Siendo el diagrama de transiciones el mostrado en la figura 11:

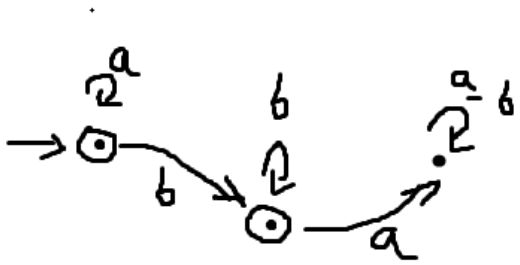


*Figura 11. Diagrama de transiciones de a*b*.*

En la figura siguiente (figura 12), podemos ver el código de este AFD, es bastante similar al código del ejemplo realizado por el profesor, solo que aquí cambiamos nuestros estados, así como los estados de aceptación, nuestro delta y las cadenas de ejemplo del lenguaje y su complemento.

```
#AFD
#a*b*

Q=['q0','q1','q2']
sigma=['a','b']
s='q0'
F=['q0','q1']

delta= { ('q0' , 'a'):'q0',
         ('q0' , 'b'):'q1',
         ('q1' , 'a'):'q2',
         ('q1' , 'b'):'q1',
         ('q2' , 'a'):'q2',
         ('q2' , 'b'):'q2'
       }

Ejemplos_L=['', 'a', 'aaa', 'ab', 'bbb']
Ejemplos_Lc=['ba', 'baa', 'aba', 'ababa']

def transicion(estado,sigma):
    #print("estado=",estado,"sigma",sigma)
    estado_siguiente=delta[(estado,sigma)]
    #print("estado siguiente=",estado_siguiente)
    return estado_siguiente


#w='baab'

for w in Ejemplos_L:
    estado=s
    for sigma in w:
        estado=transicion(estado,sigma)

    if estado in F:
        print(w, "es aceptada")
    else:
        print(w, "no es aceptada")

for w in Ejemplos_Lc:
    estado=s
    for sigma in w:
        estado=transicion(estado,sigma)

    if estado in F:
        print(w, "es aceptada")
    else:
        print(w, "no es aceptada")
```

Figura 12. Código del ejercicio 1.

En la figura 13 tenemos la salida del código de la figura 12;

```
 es aceptada
a es aceptada
aaa es aceptada
ab es aceptada
bbb es aceptada
ba no es aceptada
baa no es aceptada
aba no es aceptada
ababa no es aceptada
>>>
```

Figura 13. salida del código del ejercicio 1.

Ejercicio 2

El segundo ejercicio consta de generar el AFD que acepte las cadenas en el siguiente lenguaje; L= {cadenas que tienen un numero impar de a's y un numero par de b's}

Siendo el diagrama de transiciones el mostrado en la figura 14:

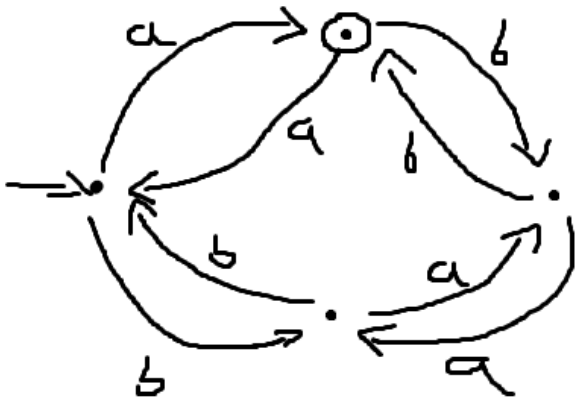


Figura 14. diagrama de transiciones del ejercicio 2.

Ahora aquí tenemos 4 estados, por lo tanto, una función de transición más amplia, el código de este ejercicio lo vemos en la figura 15;

```
#AFD
#L={Cadenas que tiene un numero impar de a´s y un numero par de b's}

Q=['q0','q1','q2', 'q3']
sigma=['a','b']
s='q0'
F=['q1']

delta= { ('q0' , 'a'):'q1',
         ('q0' , 'b'):'q3',
         ('q1' , 'a'):'q0',
         ('q1' , 'b'):'q2',
         ('q2' , 'a'):'q3',
         ('q2' , 'b'):'q1',
         ('q3' , 'a'):'q2',
         ('q3' , 'b'):'q0'
        }

Ejemplos_L=['a', 'aaa', 'abb', 'abbbb', 'aaabb']
Ejemplos_Lc=['', 'aa', 'aba', 'bbba']

def transicion(estado,sigma):
    #print("estado=",estado,"sigma",sigma)
    estado_siguiente=delta[(estado,sigma)]
    #print("estado siguiente=",estado_siguiente)
    return estado_siguiente

for w in Ejemplos_L:
    estado=s
    for sigma in w:
        estado=transicion(estado,sigma)

    if estado in F:
        print(w, "es aceptada")
    else:
        print(w, "no es aceptada")

for w in Ejemplos_Lc:
    estado=s
    for sigma in w:
        estado=transicion(estado,sigma)

    if estado in F:
        print(w, "es aceptada")
    else:
        print(w, "no es aceptada")
```

Figura 15. código del ejercicio 2.

En la figura 16 tenemos la salida del código de la figura 15;

```
a es aceptada
aaa es aceptada
abb es aceptada
abbbb es aceptada
aaabb es aceptada
 no es aceptada
aa no es aceptada
aba no es aceptada
bbba no es aceptada
>>>
```

Figura 16. salida del código del ejercicio 2.

Ejercicio 3

Por último, debemos generar el AFD que acepte las cadenas del siguiente lenguaje;
L= {Cadenas que terminan ya sea con la subcadena ab o con la subcadena aa}

Siendo el diagrama de transiciones el mostrado en la figura 17:

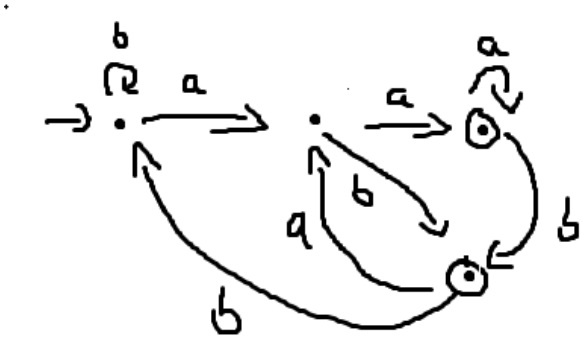


Figura 17. diagrama de transiciones del ejercicio 3.

El código de este ejercicio lo vemos en la figura 18;

```
#AFD
#L={Cadenas que terminan ya sea con la subcadena ab o con la subcadena aa}

Q=['q0','q1','q2', 'q3']
sigma=['a','b']
s='q0'
F=['q2','q3']

delta= { ('q0' , 'a'):'q1',
         ('q0' , 'b'):'q0',
         ('q1' , 'a'):'q2',
         ('q1' , 'b'):'q3',
         ('q2' , 'a'):'q2',
         ('q2' , 'b'):'q3',
         ('q3' , 'a'):'q1',
         ('q3' , 'b'):'q0'
        }

Ejemplos_L=['ab', 'aa', 'abababb', 'bababababaa', 'bbaa','aaa']
Ejemplos_Lc=['', 'a', 'b', 'ba','baaba','aababbb']

def transicion(estado,sigma):
    #print("estado=",estado,"sigma",sigma)
    estado_siguiente=delta[(estado,sigma)]
    #print("estado siguiente=",estado_siguiente)
    return estado_siguiente


#w='baab'

for w in Ejemplos_L:
    estado=s
    for sigma in w:
        estado=transicion(estado,sigma)

    if estado in F:
        print(w, "es aceptada")
    else:
        print(w, "no es aceptada")

for w in Ejemplos_Lc:
    estado=s
    for sigma in w:
        estado=transicion(estado,sigma)

    if estado in F:
        print(w, "es aceptada")
    else:
        print(w, "no es aceptada")
```

Figura 18. código del ejercicio 3.

En la figura 19 tenemos la salida del código de la figura 18;

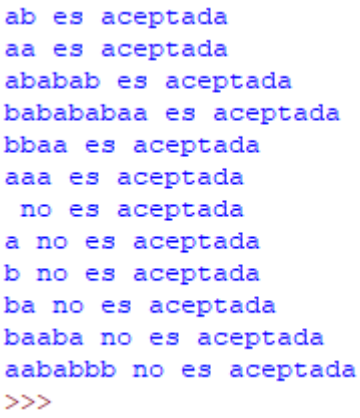


Figura 19. salida del código del ejercicio 3

CONCLUSIONES Y RECOMENDACIONES

A partir de los ejercicios realizados de AFD, pude comprender de mejor manera lo revisado en las clases teóricas sobre este tema, además de llevarlo a código en el lenguaje de programación Python.

Al inicio pensé que sería algo complicado llevar los AFD's a código, pero al final no fue tan complicado como lo pensaba. Las principales dificultades que se me presentaron fueron en el último AFD, ya que se me dificulto un poco hacer su diagrama de transiciones, pero al final, creo que lo resolví bien.

Una vez más considero que la realización de la práctica fue llevada de manera correcta y de una forma clara.

BIBLIOGRAFÍA

García, P., Pérez, T., Ruiz, J., Segarra, E., Sempere, J. M., & de Parga, M. V. (2001). *Teoría de autómatas y lenguajes formales* (pp. 32-44). Alfaomega.

Rodríguez Gustavo y López Aurelio. (2010). "Introducción a la Teoría de Autómatas, Lenguajes y Computación". Obtenido de: https://ccc.inaoep.mx/~grodrig/Descargas/AutomatasLenguajesCapitulo2.pdf