INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO (ESCOM)

TEORIA COMPUTACIONAL

NOMBRE Y NÚMERO DE LA PRÁCTICA:

- PRÁCTICA 4. AFD

NOMBRE DEL ALUMNO:

- SANTOS MÉNDEZ ULISES JESÚS

NOMBRE DEL MAESTRO:

- JORGE LUIS ROSAS TRIGUEROS

FECHA DE REALIZACIÓN:

- 20/11/2020

FECHA DE ENTREGA:

- 27/11/2020

Marco Teórico:

Autómata Finito Determinista

Un autómata finito determinista, que es aquel que sólo puede estar en un único estado después de leer cualquier secuencia de entradas. 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 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 $\sum$.
3. Una función de transición que toma como argumento un estado y un símbolo de entrada y devuelve un estado. La función de transición se designa habitualmente como: δ.
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.

A menudo se hace referencia a un autómata finito determinista mediante su acrónimo: AFD. La representación más sucinta de un AFD consiste en un listado de los componentes generales.

Normalmente, en las demostraciones, definiremos un AFD utilizando la notación de “quíntupla” siguiente:

$$A = (Q, \sum, \delta, q0, F)$$

Notaciones más simples para los AFD

Especificar un AFD utilizando una quíntupla con una descripción detallada de la función de transición δ resulta bastante tedioso y complicado de leer. Ha y disponibles dos notaciones más cómodas para describir los autómatas:

1. Un diagrama de transiciones, que es un grafo.
2. Una tabla de transiciones, que es una ordenación tabular de la función δ, la cual especifica el conjunto de estados y el alfabeto de entrada.

Diagramas de transiciones

Un diagrama de transiciones de un AFD $A = (Q, \sum, \delta, q0, F)$ es un grafo definido como:

a) Para cada estado de Q, existe un nodo.
b) Existe una flecha dirigida al estado inicial q0, etiquetada como Inicio. Esta flecha o tiene origen en ningún nodo.
c) Los nodos correspondientes a los estados de aceptación (los que pertenecen a F) están marcados con un doble círculo. Los estados que ni pertenecen a F tienen un círculo simple.

Material y Equipo:

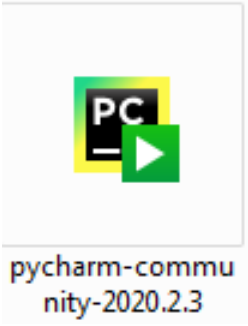
-PC (véase figura 1)[1]

(Figura 1)

-Python 3 (Python 3.8.6) (véase figura. 2)[2]

(Figura 2)

-IDE (Pycharm) (véase figura 3)[3]



(Figura 3)

[1] Figura 1: PC o cualquier computadora para el desarrollo de la práctica.
[2] Ejecutable para instalar Python 3.8.6.
[3] Ejecutable de IDE Pycharm .

Desarrollo:

1) Programa de ejemplo para conocer cómo definir el AFD y saber si acepta un lenguaje propuesto perteneciente a ∑.(véase figura 4)[4].

AFD.py

```
# AFD Ejemplo clase
def ejemplo():
    Q = ['q0', 'q1']
    Sigma = ['a', 'b']
    s = 'q0'
    F = ['q0']

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

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

Despúes se definio la función que se ercargó de hacer las transiciones de un estado a otro almacenando su resultado y comparando si cumple o no cumple las condiciones (véase figura 5)[5].

```
    print("\n---------------AFD de ejemplo----------------------\n")

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


ejemplo()
```

[4] Fragmento de código que describe al AFD.
[5] Fragmento de código que describe las transiciones del AFD.

Ejecución del programa comprobando que el autómata acepta el lenguaje y subcadenas que lo contienen, así como indica las subcadenas que no son aceptadas por el autómata (véase figura 6)[6].

```
---------------AFD de ejemplo-----------------------

 es aceptada
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  b
estado_siguiente =  q0 sigma =  b
bb es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  b
estado_siguiente =  q0 sigma =  b
abb es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
aaa es aceptada
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
b no es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  a
estado_siguiente =  q1 sigma =  a
aba no es aceptada
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  a
estado_siguiente =  q1 sigma =  a
ba no es aceptada
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
 estado =  q1 sigma =  b
 estado_siguiente =  q0 sigma =  b
 estado =  q0 sigma =  b
 estado_siguiente =  q1 sigma =  b
 bbb no es aceptada
```

[6] Ejecución del programa mostrando estados aceptados y estados no aceptados.

Se hizo como ejercicio inicial el hacer un autómata que determine si acepta los lenguajes pertenecientes a (véase figura 7)[7]:

$$L(M) = \{a^n b^m\}$$

AFD.py

```
def autdos():
    # L(M)= Cadena que tengan a*b*
    # conjunto de estados con conjunto de caracteres
    Q = ['q0', 'q1', 'q2']
    sigma = ['a', 'b']
    # estado inicial
    s = 'q0'
    # Estado de aceptacion
    F = ['q0', 'q1', ]
    # delta como diccionario donde las llaves sean duplas
    delta = {('q0', 'a'): 'q0',
             ('q0', 'b'): 'q1',
             ('q1', 'b'): 'q1',
             ('q1', 'a'): 'q2',
             ('q2', 'b'): 'q2',
             ('q2', 'a'): 'q2'}

    Ejemplos_L = ['', 'a', 'b', 'aa', 'bb', 'aabb', ]
    Ejemplos_Lc = ['ba', 'aba', 'baa', 'ababa']
```

Parte del programa encargada de hacer las transiciones necesarias con las condiciones establecidas así como de comparar si cumple o no cumple con pertenecer al lenguaje (véase figura 8)[8].

```
    print("\n-----------------AFD {a^n b^m}---------------------------\n\n")

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


autdos()
```

[7] Fragmento de código que describe el AFD del lenguaje pedido
[8] Fragmento de código que describe los cambios de estado.

La ejecución del programa nos indica que autómata estamos utilizando así como las distintas combinaciones que hace el autómata, afirmando si pertenece al lenguaje o no, en este caso dice si lo acepta o no lo acepta (véase figura 9)[9].

```
-----------------AFD {a^n b^m}--------------------------


 es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
a es aceptada
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
b es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
aa es aceptada
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  b
estado_siguiente =  q1 sigma =  b
bb es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  b
estado_siguiente =  q1 sigma =  b
aabb es aceptada
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  a
estado_siguiente =  q2 sigma =  a
ba no es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  a
estado_siguiente =  q2 sigma =  a
```

[9] Ejecución del programa mostrando las transiciones y su aceptación o rechazo.

```
aba no es aceptada
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  a
estado_siguiente =  q2 sigma =  a
estado =  q2 sigma =  a
estado_siguiente =  q2 sigma =  a
baa no es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  a
estado_siguiente =  q2 sigma =  a
estado =  q2 sigma =  b
estado_siguiente =  q2 sigma =  b
estado =  q2 sigma =  a
estado_siguiente =  q2 sigma =  a
ababa no es aceptada
```

Definición del AFD encargado de aceptar el lenguaje propuesto para a impar y b par (véase figura 10)[10].

AFD.py

```
103 def autparimpar():
104     # L(M)= Cadena que tengan un numero impar de a's y un numero par de b's
105     # conjunto de estados con conjunto de caracteres
106     Q = ['q0', 'q1', 'q2', 'q3']
107     sigma = ['a', 'b']
108     # estado inicial
109     s = 'q0'
110     # Estado de aceptacion
111     F = ['q1']
112     # delta como diccionario donde las llaves sean duplas
113     delta = {('q0', 'a'): 'q1',
114              ('q0', 'b'): 'q3',
115              ('q1', 'a'): 'q0',
116              ('q1', 'b'): 'q2',
117              ('q2', 'a'): 'q3',
118              ('q2', 'b'): 'q1',
119              ('q3', 'a'): 'q2',
120              ('q3', 'b'): 'q0',}
121
122     Ejemplos_L = ['a', 'aaa', 'abbaa', 'aabbbba']
123     Ejemplos_Lc = ['','b', 'aabbbaa', 'bbbaa', 'abbba']
```

[10] Fragmento de código que describe el AFD del lenguaje que nos piden.

Parte del programa para aceptar el lenguaje propuesto con su condición (véase figura 11)[11].

```
    print("\n-----------------AFD {a impar y b par}-------------------------\n\n")

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


autparimpar()
```

[11] Fragmento de código que describe el recorrido en las distintas transiciones del AFD.

Ejecución del programa con la aceptación del lenguaje y de lo que no pertenece fue no aceptado (véase figura 12)[12].

```
-----------------AFD {a impar y b par}--------------------------


estado =  q0 sigma =  a
estado_siguiente =  q1 sigma =  a
a es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q1 sigma =  a
estado =  q1 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  a
estado_siguiente =  q1 sigma =  a
aaa es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q1 sigma =  a
estado =  q1 sigma =  b
estado_siguiente =  q2 sigma =  b
estado =  q2 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  a
estado_siguiente =  q1 sigma =  a
abbaa es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q1 sigma =  a
estado =  q1 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
estado_siguiente =  q3 sigma =  b
estado =  q3 sigma =  b
estado_siguiente =  q0 sigma =  b
estado =  q0 sigma =  b
estado_siguiente =  q3 sigma =  b
estado =  q3 sigma =  b
estado_siguiente =  q0 sigma =  b
estado =  q0 sigma =  a
estado_siguiente =  q1 sigma =  a
aabbbba es aceptada
 no es aceptada
```

[12] Ejecución del programa mostrando la aceptación y rechazo de las subcadenas.

```
estado_siguiente =  q3 sigma =  b
b no es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q1 sigma =  a
estado =  q1 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
estado_siguiente =  q3 sigma =  b
estado =  q3 sigma =  b
estado_siguiente =  q0 sigma =  b
estado =  q0 sigma =  b
estado_siguiente =  q3 sigma =  b
estado =  q3 sigma =  a
estado_siguiente =  q2 sigma =  a
estado =  q2 sigma =  a
estado_siguiente =  q3 sigma =  a
aabbbaa no es aceptada
estado =  q0 sigma =  b
estado_siguiente =  q3 sigma =  b
estado =  q3 sigma =  b
estado_siguiente =  q0 sigma =  b
estado =  q0 sigma =  b
estado_siguiente =  q3 sigma =  b
estado =  q3 sigma =  a
estado_siguiente =  q2 sigma =  a
estado =  q2 sigma =  a
estado_siguiente =  q3 sigma =  a
bbbaa no es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q1 sigma =  a
estado =  q1 sigma =  b
estado_siguiente =  q2 sigma =  b
estado =  q2 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  b
estado_siguiente =  q2 sigma =  b
estado =  q2 sigma =  a
estado_siguiente =  q3 sigma =  a
abbba no es aceptada
```

Parte del programa donde se nos dio la premisa de que lenguaje queremos que acepte el AFD (véase figura 13)[13].

```
157  def terminacon():
158      # L(M)= Cadenas que terminan ya sea con la subcadena ab o con la subcadena aa
159      # conjunto de estados con conjunto de caracteres
160      Q = ['q0', 'q1', 'q2', 'q3', 'q4']
161      sigma = ['a', 'b']
162      # estado inicial
163      s = 'q0'
164      # Estado de aceptacion
165      F = ['q0', 'q1']
166      # delta como diccionario donde las llaves sean duplas
167      delta = {('q0', 'a'): 'q0',
168               ('q0', 'b'): 'q1',
169               ('q1', 'a'): 'q2',
170               ('q1', 'b'): 'q4',
171               ('q2', 'a'): 'q1',
172               ('q2', 'b'): 'q3',
173               ('q3', 'a'): 'q3',
174               ('q3', 'b'): 'q3',
175               ('q4', 'a'): 'q0',
176               ('q4', 'b'): 'q4'}
177
178      Ejemplos_L = ['ab', 'aa', 'abbab', 'abaa', 'abaabbbab']
179      Ejemplos_Lc = ['', 'ba', 'aaba', 'ababa', 'aababa']
180
181      print("\n-----------------AFD {terminan con ab y aa}-------------------------\n\n")
```

[13] Fragmento de código que describe al AFD para el lenguaje que se pide.

Parte del programa encargado de hacer los ciclos y combinaciones necesarias para que se puedan validar o rechazar las distintas subcadenas (véase figura 14)[14].

```
183     # funcion
184     def transicion(estado, sigma):
185         print("estado = ", estado, "sigma = ", sigma)
186         estado_siguiente = delta[(estado, sigma)]
187         print("estado_siguiente = ", estado_siguiente, "sigma = ", sigma)
188         return estado_siguiente
189 
190     for w in Ejemplos_L:
191         estado = s
192         for sigma in w:
193             estado = transicion(estado, sigma)
194 
195         if estado in F:
196             print(w, "es aceptada")
197         else:
198             print(w, "no es aceptada")
199 
200     for w in Ejemplos_Lc:
201         estado = s
202         for sigma in w:
203             estado = transicion(estado, sigma)
204 
205         if estado in F:
206             print(w, "es aceptada")
207         else:
208             print(w, "no es aceptada")
209 
210 
211 terminacon()
```

[14] Fragmento de código que describe a las transiciones que tiene el autómata en su recorrido.

En la ejecución del programa se logró ver cuales subcadenas acepto y cuales no acepto haciendo énfasis en las que pertenecen al lenguaje y las que no pertenecen al lenguaje (véase figura 15)[15].

```
-----------------AFD {terminan con ab y aa}---------------


estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
ab es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
aa es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  b
estado_siguiente =  q4 sigma =  b
estado =  q4 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
abbab es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  a
estado_siguiente =  q2 sigma =  a
estado =  q2 sigma =  a
estado_siguiente =  q1 sigma =  a
abaa es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
```

[15] Ejecución del programa mostrando las subcadenas aceptadas y rechazadas.

```
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  a
estado_siguiente =  q2 sigma =  a
estado =  q2 sigma =  a
estado_siguiente =  q1 sigma =  a
estado =  q1 sigma =  b
estado_siguiente =  q4 sigma =  b
estado =  q4 sigma =  b
estado_siguiente =  q4 sigma =  b
estado =  q4 sigma =  b
estado_siguiente =  q4 sigma =  b
estado =  q4 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
abaabbbab es aceptada
 es aceptada
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  a
estado_siguiente =  q2 sigma =  a
ba no es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  a
estado_siguiente =  q2 sigma =  a
aaba no es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  a
```

```
estado_siguiente =  q2 sigma =  a
estado =  q2 sigma =  b
estado_siguiente =  q3 sigma =  b
estado =  q3 sigma =  a
estado_siguiente =  q3 sigma =  a
ababa no es aceptada
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  a
estado_siguiente =  q0 sigma =  a
estado =  q0 sigma =  b
estado_siguiente =  q1 sigma =  b
estado =  q1 sigma =  a
estado_siguiente =  q2 sigma =  a
estado =  q2 sigma =  b
estado_siguiente =  q3 sigma =  b
estado =  q3 sigma =  a
estado_siguiente =  q3 sigma =  a
aababa no es aceptada
```

Conclusiones:

En conclusión esta práctica nos enseñó a como modelar un AFD en Python así como determinar si el Lenguaje asignado es aceptado por el autómata, se comprobó que determinar el AFD en Python es muy similar a como se hace teóricamente.

Bibliografía:

- *Practique la teoría de autómatas y lenguajes formales*, Leonardo Alonso Hernández Rodríguez, 2010.
- *Python para todos*, Raúl González Duque, 2007.
- *Introducción a la programación en Python*, Andrés Marzal, Isabel García, 2010.
- *Teoría de autómatas y lenguajes formales*, Serafín Moral.
- *Teorías de Autómatas y lenguajes formales*, Elena Jurado Málaga, 2008.
- *Teoría de la computación; Lenguajes formales, autómatas y complejidad,* J. Glenn Brookshear.
- *Teoría de autómatas, lenguajes y computación,* Jeffrey Ullman, 2007.