



Instituto Politécnico Nacional
Escuela Superior de Cómputo

Teoría Computacional

Practica 3: Automata Finito Determinista

Alumna: Ramírez Galindo Karina

Profesor: Rosas Trigueros Jorge Luis

Fecha de realización: 23-09-2019

Fecha de entrega: 30-09-2019



Marco Teórico

Teoría de Autómatas

Autómata: [1]

- Modelo matemático de computación.
- Dispositivo abstracto con capacidad de computación.

Teoría de Autómatas: [1]

- Abstracción de cualquier tipo de computador y/o lenguaje de programación.
- Desglose en sus elementos básicos (Entrada, Estado, Transición, Salidas y elementos auxiliares)

La Teoría de Autómatas es una rama de la Teoría de la Computación que estudia las máquinas teóricas llamadas autómatas. Estas máquinas son modelos matemáticos. [2]

Un Autómata está formado por un conjunto de estados, uno de los cuales es el estado en el que la máquina se encuentra inicialmente. Recibe como entrada una palabra (una concatenación de símbolos del alfabeto del autómata) y según esta, la máquina puede cambiar de estados. [2] [3]

Los Autómatas se clasifican según el número de estados (finito o no), la forma en que se realiza el cambio de estado (determinista o no), si acepta o no el símbolo vacío ϵ , si tiene o no una pila, etc. [4]

Los Autómatas están estrechamente relacionados con la máquina de Turing (1936), de gran importancia en la Teoría de la Computación. Esto se debe a que una máquina de Turing puede simular el almacenamiento y la unidad de control de una computadora. Tenemos certeza de que lo que no puede ser resuelto por una máquina de Turing no puede ser resuelto por una computadora real. [5]

Los autómatas finitos se utilizan como modelos para: [2] ´

- Software para diseñar circuitos digitales
- Analizador léxico de un compilador
- Buscar palabras clave en un archivo o en el web
- Software para verificar sistemas de estados finitos, como protocolos de comunicaciones

Autómata Finito Determinista

Un autómata finito determinista M es una colección de cinco elementos [4]

$$M=(Q,\Sigma,\delta,s,F)$$

Donde:

- Q es un conjunto finito de estados, denotados por $q_0, q_1, q_2, \dots, q_0, q_1, q_2, \dots$
- Σ es el alfabeto, es decir, un conjunto finito de símbolos que formaran palabras o cadenas. El conjunto de palabras que se pueden formar concatenando los símbolos de Σ se denota por Σ^* . La palabra vacía, que no está formada por ningún símbolo, forma parte de Σ^* .
- δ es la función de transición. Determina el comportamiento del autómata.

$$\delta(q_i, a) = q_j \quad \delta(q_i, a) = q_j$$

Significa que si en el estado q_i de Q el autómata recibe el símbolo de entrada a de Σ , entonces pasa al estado q_j de Q .

- s es el estado inicial, el estado en qué el autómata se encuentra inicialmente.
- F es el subconjunto de Q (por tanto, finito) que contiene los estados de aceptación (o finales), que son los estados que provocan la parada del autómata.

Representación de un Autómata Finito Determinista [5]

- ❖ Representaremos los estados del AFD mediante círculos que encierran el nombre del estado (q_0, q_1, \dots).
- ❖ La posible transición

$$\delta(q_i, x) = q_j \quad \delta(q_i, x) = q_j$$

- ❖ Se representa mediante una flecha que empieza en q_i y termina en q_j con la etiqueta "x".
- ❖ Los círculos de los estados de aceptación tienen el borde doble.
- ❖ El estado inicial, q_0 , se representa con una flecha que termina en dicho estado (pero no empieza en ningún estado).

Material y equipo

- 🌐 Sistema Operativo Ubuntu
- 🌐 Python 3

Desarrollo de la práctica

Objetivo:

Reconocer lenguajes regulares, realizar autómatas finitos deterministas, comprender su funcionamiento así como su aplicación

Ejercicios:

- 1) Desarrollar un autómata que reciban la siguiente expresión regular: a^*b (Desarrollado en clase)

El autómata queda representado de la siguiente forma:

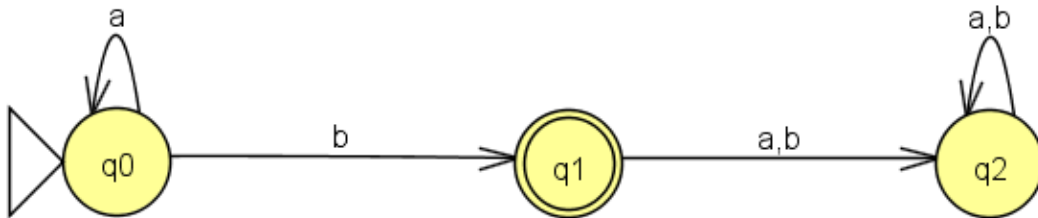


Imagen 1

AFD Ejercicio 1

El cual está conformado por:

$$M: \{Q, \Sigma, \delta, s, F\}$$

Donde:

$$Q = \{q_0, q_1, q_2\}$$

$$S = q_0$$

$$\Sigma = \{a, b\}$$

$$F = \{q_1\}$$

$$\delta = (q_0, a) = q_0$$

$$\delta = (q_1, a) = q_2$$

$$\delta = (q_2, a) = q_2$$

$$\delta = (q_0, b) = q_1$$

$$\delta = (q_1, b) = q_2$$

$$\delta = (q_2, b) = q_2$$

En Python queda de la siguiente manera:



```

1.py
~/Documentos
Guardar

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

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

print(Q,Sigma,F,s,delta)

w=['b','aaaab','ab','ba','aba','bb']

def transicion(estado,sigma):
    global Sigma,delta
    STATUS=True
    if sigma not in Sigma:
        STATUS=False
        return '',STATUS

    if (estado,sigma) not in delta.keys():
        STATUS=False
        return '',STATUS

    estado_siguiete = delta[(estado,sigma)]
    print("Transicion(",estado,",",sigma,")->",estado_siguiete)
    return estado_siguiete,STATUS

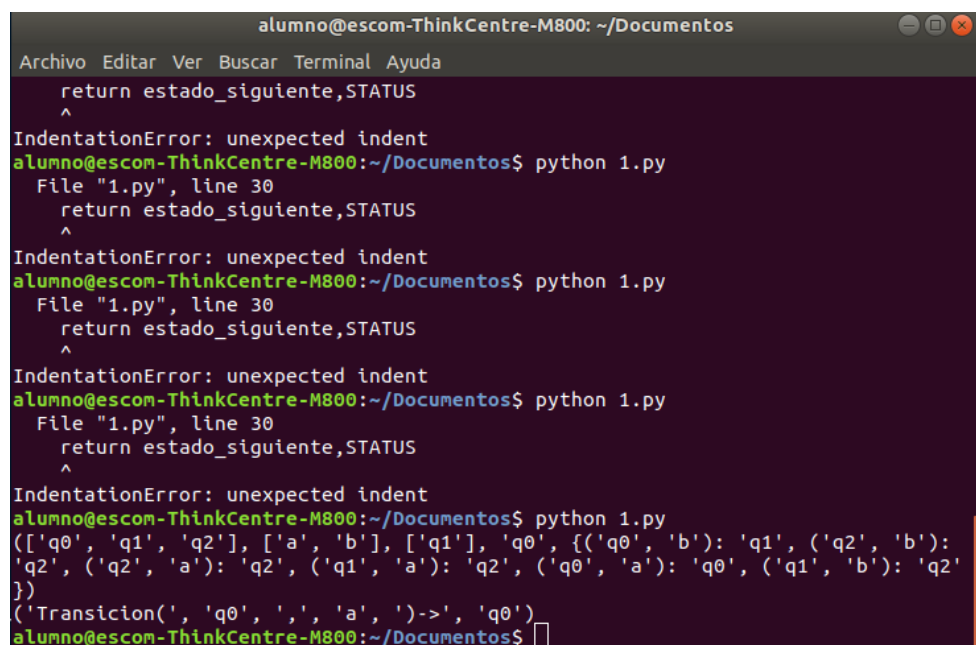
transicion(s,'a')

```

Imagen 2

Código en Python AFD1

Podemos probar la transición con cualquier elemento del alfabeto:



```

alumno@escom-ThinkCentre-M800: ~/Documentos
Archivo Editar Ver Buscar Terminal Ayuda
return estado_siguiete,STATUS
^
IndentationError: unexpected indent
alumno@escom-ThinkCentre-M800:~/Documentos$ python 1.py
File "1.py", line 30
    return estado_siguiete,STATUS
    ^
IndentationError: unexpected indent
alumno@escom-ThinkCentre-M800:~/Documentos$ python 1.py
File "1.py", line 30
    return estado_siguiete,STATUS
    ^
IndentationError: unexpected indent
alumno@escom-ThinkCentre-M800:~/Documentos$ python 1.py
File "1.py", line 30
    return estado_siguiete,STATUS
    ^
IndentationError: unexpected indent
alumno@escom-ThinkCentre-M800:~/Documentos$ python 1.py
([('q0', 'q1', 'q2'], ['a', 'b'], ['q1'], 'q0', {('q0', 'b'): 'q1', ('q2', 'b'): 'q2', ('q2', 'a'): 'q2', ('q1', 'a'): 'q2', ('q0', 'a'): 'q0', ('q1', 'b'): 'q2'})
('Transicion(', 'q0', ' ', 'a', ') -> ', 'q0')
alumno@escom-ThinkCentre-M800:~/Documentos$

```

Imagen 3

SALIDA:AFD1

Ahora, comprobaremos si se aceptan o no las cadenas anteriormente escritas en “w”

```

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

print(Q,Sigma,F,s,delta)

w=['b','aaaab','ab','ba','aba','bb']

def transicion(estados,sigma):
    global Sigma,delta
    STATUS=True
    if sigma not in Sigma:
        STATUS=False
        return '',STATUS

    if (estados,sigma) not in delta.keys():
        STATUS=False
        return '',STATUS

    estado_siguiente = delta[(estados,sigma)]
    print("Transicion(",estados," ",sigma,"->",estado_siguiente)
    return estado_siguiente,STATUS

#modificable
transicion(s,'a')

for cadena in w:
    estado = s
    for c in cadena:
        estado,STATUS=transicion(estado,c)
        if not STATUS:
            break
    if STATUS and estado in F:
        print(cadena,"si esta en el lenguaje")
    else:
        print(cadena,"no esta en el lenguaje")

```

Imagen 4

Condiciones para las cadenas de “w”

```

SyntaxError: invalid syntax
alumno@escom-ThinkCentre-M800:~/Documentos$ python 1.py
([('q0','q1','q2'],[('a','b'],[('q1','q0'],[('q0','b'):'q1',('q2','b'):'q2',('q2','a'):'q2',('q1','a'):'q2',('q0','a'):'q0',('q1','b'):'q2'})
('Transicion(', 'q0', ' ', 'a', ' )->', 'q0')
('Transicion(', 'q0', ' ', 'b', ' )->', 'q1')
('b', 'si esta en el lenguaje')
('Transicion(', 'q0', ' ', 'a', ' )->', 'q0')
('Transicion(', 'q0', ' ', 'a', ' )->', 'q0')
('Transicion(', 'q0', ' ', 'a', ' )->', 'q0')
('Transicion(', 'q0', ' ', 'a', ' )->', 'q0')
('Transicion(', 'q0', ' ', 'b', ' )->', 'q1')
('aaaab', 'si esta en el lenguaje')
('Transicion(', 'q0', ' ', 'a', ' )->', 'q0')
('Transicion(', 'q0', ' ', 'b', ' )->', 'q1')
('ab', 'si esta en el lenguaje')
('Transicion(', 'q0', ' ', 'b', ' )->', 'q1')
('Transicion(', 'q1', ' ', 'a', ' )->', 'q2')
('ba', 'no esta en el lenguaje')
('Transicion(', 'q0', ' ', 'a', ' )->', 'q0')
('Transicion(', 'q0', ' ', 'b', ' )->', 'q1')
('Transicion(', 'q1', ' ', 'a', ' )->', 'q2')
('aba', 'no esta en el lenguaje')
('Transicion(', 'q0', ' ', 'b', ' )->', 'q1')
('Transicion(', 'q1', ' ', 'b', ' )->', 'q2')
('bb', 'no esta en el lenguaje')
alumno@escom-ThinkCentre-M800:~/Documentos$

```

Imagen 5

Salido: Cadenas de “w”

2) $\{w \in \{a,b\}^* \mid w \text{ no contiene la subcadena } aa\}$ (Tarea)

El autómata queda representado de la siguiente forma:

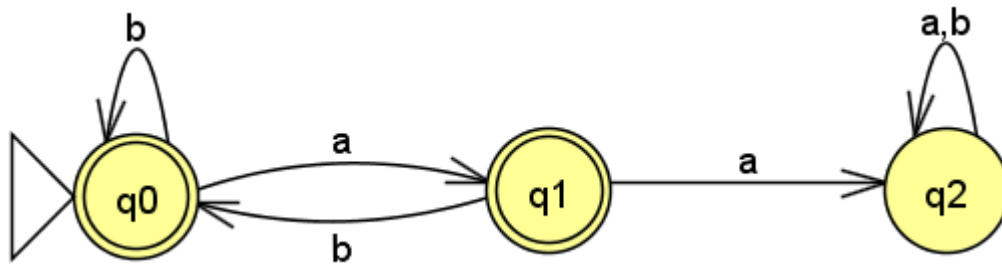


Imagen 6

AFD Ejercicio 2

El cual está conformado por:

$$M: \{Q, \Sigma, \delta, s, F\}$$

Donde:

$$Q = \{q0, q1, q2\}$$

$$S = q0$$

$$\Sigma = \{a, b\}$$

$$F = \{q0, q1\}$$

$$\delta = (q0, a) = q1$$

$$\delta = (q1, a) = q2$$


$$\delta = (q2, a) = q2$$

$$\delta = (q0, b) = q0$$

$$\delta = (q1, b) = q0$$

$$\delta = (q2, b) = q2$$

En Python queda de la siguiente manera:



```
Abrir AFD.py Guardar
Q=['q0','q1','q2']
Sigma=['a','b']
F=['q0','q1']
s='q0'

delta={ ('q0','a'):'q1',
        ('q0','b'):'q0',
        ('q1','a'):'q2',
        ('q1','b'):'q0',
        ('q2','a'):'q2',
        ('q2','b'):'q2' }
print(Q,Sigma,F,s,delta)

w=['a','b','ab','ababab','aab','abaab','abaaaab']

def transicion(estado,sigma):
    global Sigma,delta
    STATUS=True
    if sigma not in Sigma:
        STATUS=False
        return '',STATUS

    if(estado,sigma) not in delta.keys():
        STATUS=False
        return '',STATUS

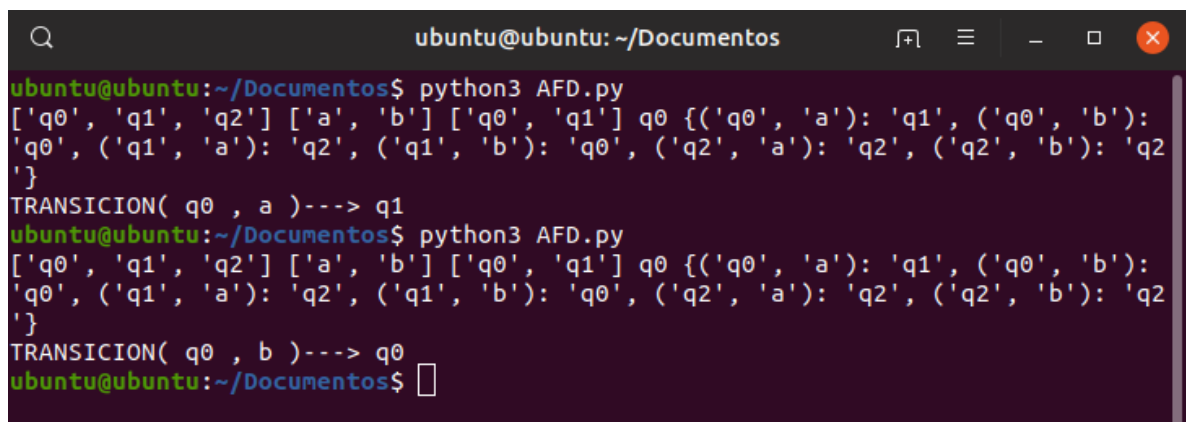
    estado_siguiente=delta[(estado,sigma)]
    print("TRANSICION(",estado,",",sigma,")--->",estado_siguiente)
    return estado_siguiente,STATUS

transicion(s,'a')
```

Imagen 7

código AFD2

Probamos la transición con cualquier elemento del alfabeto

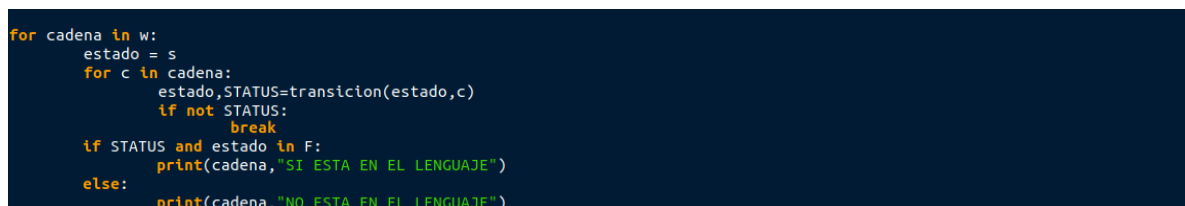


```
ubuntu@ubuntu: ~/Documentos
ubuntu@ubuntu:~/Documentos$ python3 AFD.py
['q0', 'q1', 'q2'] ['a', 'b'] ['q0', 'q1'] q0 {('q0', 'a'): 'q1', ('q0', 'b'): 'q0', ('q1', 'a'): 'q2', ('q1', 'b'): 'q0', ('q2', 'a'): 'q2', ('q2', 'b'): 'q2'}
TRANSICION( q0 , a )---> q1
ubuntu@ubuntu:~/Documentos$ python3 AFD.py
['q0', 'q1', 'q2'] ['a', 'b'] ['q0', 'q1'] q0 {('q0', 'a'): 'q1', ('q0', 'b'): 'q0', ('q1', 'a'): 'q2', ('q1', 'b'): 'q0', ('q2', 'a'): 'q2', ('q2', 'b'): 'q2'}
TRANSICION( q0 , b )---> q0
ubuntu@ubuntu:~/Documentos$
```

Imagen 8

Transición AFD2

Añadimos las condiciones para validar las cadenas de “w”



```
for cadena in w:
    estado = s
    for c in cadena:
        estado,STATUS=transicion(estado,c)
        if not STATUS:
            break
    if STATUS and estado in F:
        print(cadena,"SI ESTA EN EL LENGUAJE")
    else:
        print(cadena,"NO ESTA EN EL LENGUAJE")
```

Imagen 9

Validación de cadenas de “w”

```
ubuntu@ubuntu: ~/Documentos
python3 AFD.py
['q0', 'q1', 'q2'] ['a', 'b'] ['q0', 'q1'] q0 [('q0', 'a'): 'q1', ('q0', 'b'): 'q0', ('q1', 'a'): 'q2', ('q1', 'b'): 'q0', ('q2', 'a'): 'q2', ('q2', 'b'): 'q2']
TRANSICION( q0 , b )--> q0
TRANSICION( q0 , a )--> q1
a SI ESTA EN EL LENGUAJE
TRANSICION( q0 , b )--> q0
b SI ESTA EN EL LENGUAJE
TRANSICION( q0 , a )--> q1
TRANSICION( q1 , b )--> q0
ab SI ESTA EN EL LENGUAJE
TRANSICION( q0 , a )--> q1
TRANSICION( q1 , b )--> q0
TRANSICION( q0 , a )--> q1
TRANSICION( q1 , b )--> q0
TRANSICION( q0 , a )--> q1
TRANSICION( q1 , b )--> q0
TRANSICION( q0 , a )--> q1
abababa SI ESTA EN EL LENGUAJE
TRANSICION( q0 , a )--> q1
TRANSICION( q1 , a )--> q2
TRANSICION( q2 , b )--> q2
aab NO ESTA EN EL LENGUAJE
TRANSICION( q0 , a )--> q1
TRANSICION( q1 , b )--> q0
TRANSICION( q0 , a )--> q1
TRANSICION( q1 , a )--> q2
TRANSICION( q2 , b )--> q2
abaab NO ESTA EN EL LENGUAJE
TRANSICION( q0 , a )--> q1
TRANSICION( q1 , b )--> q0
TRANSICION( q0 , a )--> q1
TRANSICION( q1 , a )--> q2
TRANSICION( q2 , a )--> q2
TRANSICION( q2 , a )--> q2
TRANSICION( q2 , b )--> q2
abaaaaab NO ESTA EN EL LENGUAJE
ubuntu@ubuntu:~/Documentos$
```

Imagen 10 Salida: Cadenas de w

Como se observa en la imagen, las primeras 4 cadenas se aceptan, y las restantes no debido a que no cumplen con la representación del autómata.

Conclusiones

En esta práctica, aprendí a representar un autómata finito determinista en código y hacer que este vaya dando paso por paso el recorrido de cada cadena y al final diga si es parte o no del autómata, además de que con el marco teórico presentado anteriormente, pude ver más allá de una representación matemática en los autómatas, y me di cuenta de cuan importantes son el la computación, la representación más importante en mi opinión es la máquina de Turing ya que es una herramienta muy poderosa para nosotros como futuro ingenieros en sistemas.

Referencias

- [1] A. L. E. A. I. M. G. J. M. A. W. Araceli Sanchis de Miguel, «Teoría de Autómatas,» [En línea]. Available: <http://ocw.uc3m.es/ingenieria-informatica/teoria-de-automatas-y-lenguajes-formales/material-de-clase-1/tema-2-teoria-de-automatas>.
- [2] INAOE, «Teoria de Automatas y Lenguajes Formales,» [En línea]. Available: <https://ccc.inaoep.mx/~emorales/Cursos/Automatas/Introduccion.pdf>.
- [3] G. Morales-Luna, «Teoría de Autómatas,» 27 06 2000. [En línea]. Available: <http://delta.cs.cinvestav.mx/~gmorales/TeoriaDeAutomatas.pdf>.
- [4] D. G. P. Luis Migel Pardo Vasallo, «Teoria de Automatas Finitos,» [En línea]. Available: https://ocw.unican.es/pluginfile.php/1516/course/section/1946/2-1_Introduccion.pdf.
- [5] D. R. Soto, «Teoría de Autómatas y Compiladores,» Escuela de Ingeniería Informática Pontificia Universidad Católica de Valparaíso, Marzo 2010. [En línea]. Available: http://zeus.inf.ucv.cl/~rsoto/cursos/ICI445/Cap2_ICI445.pdf.