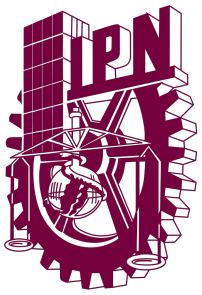


Instituto Politécnico Nacional

Escuela Superior de Cómputo

Teoría Computacional

Práctica 4

Alumno: Meléndez Padilla Mauricio

Profesor: Rosas Trigueros Jorge Luis

2CV4

Fecha de realización de la práctica

13/SEP/18

Fecha de entrega del reporte

18/SEP/18

Marco Teórico

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.
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 palabra la máquina puede cambiar de estados.
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.
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.

Autómata finito determinista AFD

Un autómata finito determinista (abreviado AFD) es un autómata finito que además es un sistema determinista; es decir, para cada estado en que se encuentre el autómata, y con cualquier símbolo del alfabeto leído, existe siempre no más de una transición posible desde ese estado y con ese símbolo.

Definición formal

Formalmente, se define como una 5-tupla $(Q, \Sigma, q_0, \delta, F)$ donde:[1]

- Q es un conjunto de estados;
- Σ es un alfabeto;
- $q_0 \in Q$ es el estado inicial;
- $\delta: Q \times \Sigma \rightarrow Q$ es una función de transición;
- $F \subseteq Q$ es un conjunto de estados finales o de aceptación.

En un AFD no pueden darse ninguno de estos dos casos:

- Que existan dos transiciones del tipo $\delta(q,a)=q_1$ y $\delta(q,a)=q_2$, siendo $q_1 \neq q_2$;
- Que existan transiciones del tipo $\delta(q, \varepsilon)$, donde ε es la cadena vacía, salvo que q sea un estado final, sin transiciones hacia otros estados.

Representación o diagrama de un AFD

Representaremos los estados del AFD mediante círculos que encierran el nombre del estado (q_0, q_1,...).

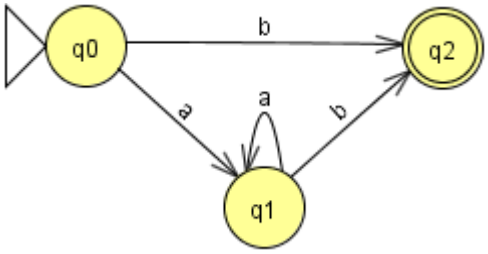
La posible transición

$$\delta(qi,x)=qj\delta(qi,x)=qj$$

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.

El material utilizado en la práctica es el siguiente:

Herramientas de software:

- Mac OS X 10.13.6
- Python 2.7.15
- VIM - Vi IMproved 8.1
- Terminal

Herramientas de hardware:

- Computadora personal.

Desarrollo de la práctica.

1. Cree un programa en Python que sea un AFD.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

Q=['q0', 'qi', 'q2']
#Estado inicial
S='q0'
Sigma=['a','b']
F=['q1']
delta={
    ('q0','a'):'q0',
    ('q0','b'):'q1',
    ('q1','a'):'q2',
    ('q1','b'):'q2',
    ('q2','a'):'q2',
    ('q2','b'):'q2'
}
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 'Transición(',estado,',',sigma,')',estado_siguiente
    return estado_siguiente, STATUS
#Prueba
w='aaab'
estado=S
for sigma in w:
    estado, STATUS=transicion(estado, sigma)
    if(not STATUS):
        break
if((STATUS) and (estado in F)):
    print w, "sí está en el lenguaje"
else:
    print w, "no está en el lenguaje"

def powerset(Q):
    from itertools import chain, combinations
    return chain.from_iterable(
        combinations(Q,r) for r in range(len(Q)+1)
    )
```

2. Pruebe 5 cadenas distintas que pertenezcan al lenguaje dado:
   - aaab
   - aaaab
   - aaaaaab
   - ab
   - aaaaaaaaaab

3. Pruebe 5 cadenas distintas que no pertenezcan al lenguaje dado:
   - abab
   - baba
   - aabaab
   - bab
   - aaaa

Diagramas, gráficas y pantallas

```
MacPro:/ root# python /Users/berry/Library/Mobile\ Documents/com~apple~CloudDocs/Documents/prac4.py
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , b ) q1
aaab sí está en el lenguaje
MacPro:/ root# python /Users/berry/Library/Mobile\ Documents/com~apple~CloudDocs/Documents/prac4.py
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , b ) q1
aaaab sí está en el lenguaje
MacPro:/ root# python /Users/berry/Library/Mobile\ Documents/com~apple~CloudDocs/Documents/prac4.py
Transición( q0 , a ) q0
Transición( q0 , b ) q1
Transición( q1 , a ) q2
Transición( q2 , b ) q2
abab no está en el lenguaje
MacPro:/ root# python /Users/berry/Library/Mobile\ Documents/com~apple~CloudDocs/Documents/prac4.py
Transición( q0 , b ) q1
Transición( q1 , a ) q2
Transición( q2 , b ) q2
Transición( q2 , a ) q2
baba no está en el lenguaje
MacPro:/ root# python /Users/berry/Library/Mobile\ Documents/com~apple~CloudDocs/Documents/prac4.py
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , b ) q1
Transición( q1 , a ) q2
Transición( q2 , a ) q2
Transición( q2 , b ) q2
aabaab no está en el lenguaje
MacPro:/ root# python /Users/berry/Library/Mobile\ Documents/com~apple~CloudDocs/Documents/prac4.py
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , b ) q1
aaaaaab sí está en el lenguaje
MacPro:/ root# python /Users/berry/Library/Mobile\ Documents/com~apple~CloudDocs/Documents/prac4.py
Transición( q0 , a ) q0
Transición( q0 , b ) q1
ab sí está en el lenguaje
MacPro:/ root# python /Users/berry/Library/Mobile\ Documents/com~apple~CloudDocs/Documents/prac4.py
Transición( q0 , b ) q1
Transición( q1 , a ) q2
Transición( q2 , b ) q2
bab no está en el lenguaje
```

Imagen 4.1 Ejercicio 1

```
MacPro:/ root# python /Users/berry/Library/Mobile\ Documents/com~apple~CloudDocs/Documents/prac4.py
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
aaaa no está en el lenguaje
```

Imagen 4.2 Ejercicio 2

```
MacPro:/ root# python /Users/berry/Library/Mobile\ Documents/com~apple~CloudDocs/Documents/prac4.py
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , a ) q0
Transición( q0 , b ) q1
aaaaaaaaaaab sí está en el lenguaje
```

Imagen 4.3 Ejercicio 3

Conclusiones y recomendaciones

Las expresiones regulares en Linux, así como en los diversos lenguajes de programación son muy útiles a la hora de encontrar ocurrencias con una dicha búsqueda, no siempre se puede buscar algo en concreto, sin embargo las expresiones regulares nos permiten todo tipo de cosas y posibilidades a la hora de las búsquedas.

Bibliografía

- [1]"Autómata finito determinista En.wikipedia.org, 2018. [Online]. Available: https://es.wikipedia.org/wiki/Aut%C3%B3mata_finito_determinista [Accessed: 18- SEP- 2018].
- [2]"Autómata Finito Determinista En.matesfacil.com, 2017 [Online]. Available: https://www.matesfacil.com/automatas-lenguajes/automata-finito-y-su-lenguaje.html [Accessed: 18-SEP-2018].