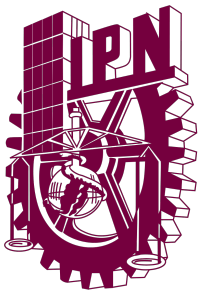


Instituto Politécnico Nacional

Escuela Superior de Cómputo

Teoría Computacional

Práctica 5

Alumno: Meléndez Padilla Mauricio

Profesor: Rosas Trigueros Jorge Luis

2CV4

Fecha de realización de la práctica

20/SEP/18

Fecha de entrega del reporte

26/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.

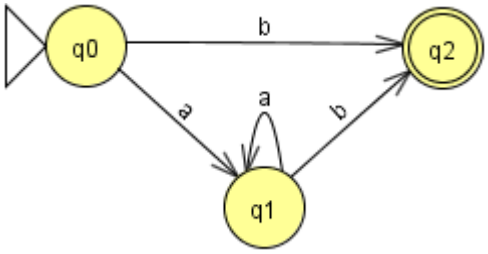
Representación o diagrama de un AFD

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).



Autómata Finito No Determinista AFND

Es un autómata finito que, a diferencia de los autómatas finitos deterministas (AFD), posee al menos un estado $q \in Q$, tal que para un símbolo $a \in \Sigma$ del alfabeto, existe más de una transición $\delta(q,a)$ posible.

En un AFND puede darse cualquiera de estos dos casos:

- Que existan 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 $\delta(q, \varepsilon)$, siendo q un estado no-final, o bien un estado final pero con transiciones hacia otros estados.

Cuando se cumple el segundo caso, se dice que el autómata es un autómata finito no determinista con transiciones vacías o transiciones ε (abreviado AFND-ε). Estas transiciones permiten al autómata cambiar de

estado sin procesar ningún símbolo de entrada. Considérese una modificación al modelo del autómata finito para permitirle ninguna, una o más transiciones de un estado sobre el mismo símbolo de entrada.

Definición formal de un AFND

Formalmente, si bien un autómata finito determinista se define como una 5-tupla (Q, Σ, q_0, δ, F) donde:

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

en un AFND la función de transición se define como:

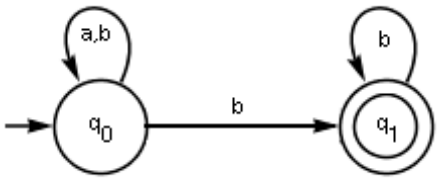
$$\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$$

Para el caso de los AFND-ε, se suele expresar la función de transición de la forma:

$$\delta : Q \times \{\Sigma \cup \epsilon\} \rightarrow \mathcal{P}(Q)$$

donde *P(Q)* es el conjunto potencia de *Q*. Esto significa que los autómatas finitos deterministas son un caso particular de los no deterministas, puesto que *Q* pertenece al conjunto *P(Q)*.

La interpretación que se suele hacer en el cómputo de un AFND es que el autómata puede pasar por varios estados a la vez, generándose una ramificación de las *configuraciones* existentes en un momento dado. Asimismo, en un autómata finito no determinista podemos aceptar la existencia de más de un nodo inicial.



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 convierta un AFND en un AFD.

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

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

DELTA={ ('q0', 'a') : ['q0', 'q1'],
        ('q0', 'b') : ['q1'],
        ('q1', 'b') : ['q0', 'q1']
        }

from itertools import chain, combinations
def powerset(iterable):
    s=list(iterable)
    return chain.from_iterable(combinations(s,r) for r in range(len(s)+1))

Qprima=list(powerset(Q))

sprima=(s,)

Fprima=[]

print 'Qprima'
print Qprima
```

```
#En Fprima van los miembros de Qprima que contengan los estados en F

for q in Qprima:
    for x in q:
        if x in F:
            Fprima.append(q)

print 'Fprima'
print Fprima

#Construir delta

delta={}

print 'Trancisiones'
for q in Qprima:
    for s in Sigma:
        P=[]
        for x in q:
            if (x,s) in DELTA.keys():
                for r in DELTA[(x,s)]:
                    if r not in P:
                        P.append(r)
        P.sort()
        print '(',q,',',s,') -> ',tuple(P)
        delta [(q,s)]=tuple(P)
print 'delta'
print delta

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='aaaab'
estado=sprima
for sigma in w:
    estado, STATUS=transicion(estado, sigma)
    if(not STATUS):
```

```
        break
if((STATUS) and (estado in Fprima)):
    print w, "sí está en el lenguaje"
else:
    print w, "no está en el lenguaje"
```

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

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

Diagramas, gráficas y pantallas

```
MacPro:Prac5 berry$ python afnd_to_afd.py
Qprima
[(), ('q0',), ('q1',), ('q0', 'q1')]
Fprima
[('q1',), ('q0', 'q1')]
Trancisiones
( () , a ) ->  ()
( () , b ) ->  ()
( ('q0',) , a ) ->  ('q0', 'q1')
( ('q0',) , b ) ->  ('q1',)
( ('q1',) , a ) ->  ()
( ('q1',) , b ) ->  ('q0', 'q1')
( ('q0', 'q1') , a ) ->  ('q0', 'q1')
( ('q0', 'q1') , b ) ->  ('q0', 'q1')
delta
{(('q1',), 'b'): ('q0', 'q1'), (('q1',), 'a'): (), (('q0', 'q1'), 'a'): ('q0', 'q1'), (('
q0', 'q1'), 'b'): ('q0', 'q1'), (('q0',), 'a'): ('q0', 'q1'), (('q0',), 'b'): ('q1',), ((
), 'b'): (), ((), 'a'): ()}
Transición( ('q0',) , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , b ) ('q0', 'q1')
aaab sí está en el lenguaje
MacPro:Prac5 berry$ python afnd_to_afd.py
Qprima
[(), ('q0',), ('q1',), ('q0', 'q1')]
Fprima
[('q1',), ('q0', 'q1')]
Trancisiones
( () , a ) ->  ()
( () , b ) ->  ()
( ('q0',) , a ) ->  ('q0', 'q1')
( ('q0',) , b ) ->  ('q1',)
( ('q1',) , a ) ->  ()
( ('q1',) , b ) ->  ('q0', 'q1')
( ('q0', 'q1') , a ) ->  ('q0', 'q1')
( ('q0', 'q1') , b ) ->  ('q0', 'q1')
delta
{(('q1',), 'b'): ('q0', 'q1'), (('q1',), 'a'): (), (('q0', 'q1'), 'a'): ('q0', 'q1'), (('
q0', 'q1'), 'b'): ('q0', 'q1'), (('q0',), 'a'): ('q0', 'q1'), (('q0',), 'b'): ('q1',), ((
), 'b'): (), ((), 'a'): ()}
Transición( ('q0',) , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , b ) ('q0', 'q1')
aaaab sí está en el lenguaje
```

Imagen 5.1 Ejercicio 2

```
MacPro:Prac5 berry$ python afnd_to_afd.py
Qprima
[(), ('q0',), ('q1',), ('q0', 'q1')]
Fprima
[('q1',), ('q0', 'q1')]
Trancisiones
( () , a ) ->  ()
( () , b ) ->  ()
( ('q0',) , a ) ->  ('q0', 'q1')
( ('q0',) , b ) ->  ('q1',)
( ('q1',) , a ) ->  ()
( ('q1',) , b ) ->  ('q0', 'q1')
( ('q0', 'q1') , a ) ->  ('q0', 'q1')
( ('q0', 'q1') , b ) ->  ('q0', 'q1')
delta
{(('q1',), 'b'): ('q0', 'q1'), (('q1',), 'a'): (), (('q0', 'q1'), 'a'): ('q0', 'q1'), (('
q0', 'q1'), 'b'): ('q0', 'q1'), (('q0',), 'a'): ('q0', 'q1'), (('q0',), 'b'): ('q1',), ((
), 'b'): (), ((), 'a'): ()}
Transición( ('q0',) , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , b ) ('q0', 'q1')
aaaaaab sí está en el lenguaje
MacPro:Prac5 berry$ python afnd_to_afd.py
Qprima
[(), ('q0',), ('q1',), ('q0', 'q1')]
Fprima
[('q1',), ('q0', 'q1')]
Trancisiones
( () , a ) ->  ()
( () , b ) ->  ()
( ('q0',) , a ) ->  ('q0', 'q1')
( ('q0',) , b ) ->  ('q1',)
( ('q1',) , a ) ->  ()
( ('q1',) , b ) ->  ('q0', 'q1')
( ('q0', 'q1') , a ) ->  ('q0', 'q1')
( ('q0', 'q1') , b ) ->  ('q0', 'q1')
delta
{(('q1',), 'b'): ('q0', 'q1'), (('q1',), 'a'): (), (('q0', 'q1'), 'a'): ('q0', 'q1'), (('
q0', 'q1'), 'b'): ('q0', 'q1'), (('q0',), 'a'): ('q0', 'q1'), (('q0',), 'b'): ('q1',), ((
), 'b'): (), ((), 'a'): ()}
Transición( ('q0',) , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , b ) ('q0', 'q1')
ab sí está en el lenguaje
```

Imagen 5.2 Ejercicio 2

```
MacPro:Prac5 berry$ python afnd_to_afd.py
Qprima
[(), ('q0',), ('q1',), ('q0', 'q1')]
Fprima
[('q1',), ('q0', 'q1')]
Trancisiones
( () , a ) ->  ()
( () , b ) ->  ()
( ('q0',) , a ) ->  ('q0', 'q1')
( ('q0',) , b ) ->  ('q1',)
( ('q1',) , a ) ->  ()
( ('q1',) , b ) ->  ('q0', 'q1')
( ('q0', 'q1') , a ) ->  ('q0', 'q1')
( ('q0', 'q1') , b ) ->  ('q0', 'q1')
delta
{(('q1',), 'b'): ('q0', 'q1'), (('q1',), 'a'): (), (('q0', 'q1'), 'a'): ('q0', 'q1'), (('
q0', 'q1'), 'b'): ('q0', 'q1'), (('q0',), 'a'): ('q0', 'q1'), (('q0',), 'b'): ('q1',), ((
), 'b'): (), ((), 'a'): ()}
Transición( ('q0',) , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , b ) ('q0', 'q1')
aaaaaaaaaaab sí está en el lenguaje
```

Imagen 5.3 Ejercicio 2

```
MacPro:Prac5 berry$ python afnd_to_afd.py
Qprima
[(), ('q0',), ('q1',), ('q0', 'q1')]
Fprima
[('q1',), ('q0', 'q1')]
Trancisiones
( () , a ) ->  ()
( () , b ) ->  ()
( ('q0',) , a ) ->  ('q0', 'q1')
( ('q0',) , b ) ->  ('q1',)
( ('q1',) , a ) ->  ()
( ('q1',) , b ) ->  ('q0', 'q1')
( ('q0', 'q1') , a ) ->  ('q0', 'q1')
( ('q0', 'q1') , b ) ->  ('q0', 'q1')
delta
{(('q1',), 'b'): ('q0', 'q1'), (('q1',), 'a'): (), (('q0', 'q1'), 'a'): ('q0', 'q1'), (('
q0', 'q1'), 'b'): ('q0', 'q1'), (('q0',), 'a'): ('q0', 'q1'), (('q0',), 'b'): ('q1',), ((
), 'b'): (), ((), 'a'): ()}
Transición( ('q0',) , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , b ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , b ) ('q0', 'q1')
abab sí está en el lenguaje
MacPro:Prac5 berry$ python afnd_to_afd.py
Qprima
[(), ('q0',), ('q1',), ('q0', 'q1')]
Fprima
[('q1',), ('q0', 'q1')]
Trancisiones
( () , a ) ->  ()
( () , b ) ->  ()
( ('q0',) , a ) ->  ('q0', 'q1')
( ('q0',) , b ) ->  ('q1',)
( ('q1',) , a ) ->  ()
( ('q1',) , b ) ->  ('q0', 'q1')
( ('q0', 'q1') , a ) ->  ('q0', 'q1')
( ('q0', 'q1') , b ) ->  ('q0', 'q1')
delta
{(('q1',), 'b'): ('q0', 'q1'), (('q1',), 'a'): (), (('q0', 'q1'), 'a'): ('q0', 'q1'), (('
q0', 'q1'), 'b'): ('q0', 'q1'), (('q0',), 'a'): ('q0', 'q1'), (('q0',), 'b'): ('q1',), ((
), 'b'): (), ((), 'a'): ()}
Transición( ('q0',) , b ) ('q1',)
Transición( ('q1',) , a ) ()
Transición( () , b ) ()
Transición( () , a ) ()
baba no está en el lenguaje
```

Imagen 5.4 Ejercicio 3

```
MacPro:Prac5 berry$ python afnd_to_afd.py
Qprima
[(), ('q0',), ('q1',), ('q0', 'q1')]
Fprima
[('q1',), ('q0', 'q1')]
Trancisiones
( () , a ) ->  ()
( () , b ) ->  ()
( ('q0',) , a ) ->  ('q0', 'q1')
( ('q0',) , b ) ->  ('q1',)
( ('q1',) , a ) ->  ()
( ('q1',) , b ) ->  ('q0', 'q1')
( ('q0', 'q1') , a ) ->  ('q0', 'q1')
( ('q0', 'q1') , b ) ->  ('q0', 'q1')
delta
{(('q1',), 'b'): ('q0', 'q1'), (('q1',), 'a'): (), (('q0', 'q1'), 'a'): ('q0', 'q1'), (('
q0', 'q1'), 'b'): ('q0', 'q1'), (('q0',), 'a'): ('q0', 'q1'), (('q0',), 'b'): ('q1',), ((
), 'b'): (), ((), 'a'): ()}
Transición( ('q0',) , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , b ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , a ) ('q0', 'q1')
Transición( ('q0', 'q1') , b ) ('q0', 'q1')
aabaab sí está en el lenguaje
MacPro:Prac5 berry$ python afnd_to_afd.py
Qprima
[(), ('q0',), ('q1',), ('q0', 'q1')]
Fprima
[('q1',), ('q0', 'q1')]
Trancisiones
( () , a ) ->  ()
( () , b ) ->  ()
( ('q0',) , a ) ->  ('q0', 'q1')
( ('q0',) , b ) ->  ('q1',)
( ('q1',) , a ) ->  ()
( ('q1',) , b ) ->  ('q0', 'q1')
( ('q0', 'q1') , a ) ->  ('q0', 'q1')
( ('q0', 'q1') , b ) ->  ('q0', 'q1')
delta
{(('q1',), 'b'): ('q0', 'q1'), (('q1',), 'a'): (), (('q0', 'q1'), 'a'): ('q0', 'q1'), (('
q0', 'q1'), 'b'): ('q0', 'q1'), (('q0',), 'a'): ('q0', 'q1'), (('q0',), 'b'): ('q1',), ((
), 'b'): (), ((), 'a'): ()}
Transición( ('q0',) , b ) ('q1',)
Transición( ('q1',) , a ) ()
Transición( () , b ) ()
bab no está en el lenguaje
```

Imagen 5.5 Ejercicio 3

```
MacPro:Prac5 berry$ python afnd_to_afd.py
Qprima
[(), ('q0',), ('q1',), ('q0', 'q1')]
Fprima
[('q1',), ('q0', 'q1')]
Trancisiones
( () , a ) ->  ()
( () , b ) ->  ()
( ('q0',) , a ) ->  ('q0', 'q1')
( ('q0',) , b ) ->  ('q1',)
( ('q1',) , a ) ->  ()
( ('q1',) , b ) ->  ('q0', 'q1')
( ('q0', 'q1') , a ) ->  ('q0', 'q1')
( ('q0', 'q1') , b ) ->  ('q0', 'q1')
delta
{(('q1',), 'b'): ('q0', 'q1'), (('q1',), 'a'): (), (('q0', 'q1'), 'a'): ('q0', 'q1'), (('
q0', 'q1'), 'b'): ('q0', 'q1'), (('q0',), 'a'): ('q0', 'q1'), (('q0',), 'b'): ('q1',), ((
), 'b'): (), ((), 'a'): ()}
Transición( ('q0',) , b ) ('q1',)
Transición( ('q1',) , a ) ()
Transición( () , a ) ()
Transición( () , a ) ()
Transición( () , a ) ()
baaaa no está en el lenguaje
```

Imagen 5.6 Ejercicio 3

Conclusiones y recomendaciones

El establecimiento de esta equivalencia es útil porque a veces la construcción de un AFND para reconocer un lenguaje determinado es más fácil que construir un AFD para dicho lenguaje. También es importante porque el AFND se puede utilizar para reducir la complejidad del trabajo matemático necesario para establecer muchas propiedades importantes en la teoría de la computación. Por ejemplo, es mucho más fácil demostrar las siguientes propiedades utilizando un AFND que un AFD:

- La unión de dos lenguajes regulares es regular.
- La concatenación de dos lenguajes regulares es regular.
- La Clausura de Kleene en un Lenguaje regular es regular.

Bibliografía

- [1]"Autómata finito no determinista En.wikipedia.org, 2018. [Online]. Available: https://es.wikipedia.org/wiki/Aut%C3%B3mata_finito_no_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].