Instituto Politécnico Nacional

Escuela Superior de Cómputo

Teoría Computacional

Practica 4:Conversión de un AFN a un AFD

Alumna: Ramírez Galindo Karina

Profesor:Rosas Trigueros Jorge Luis

Fecha de realización:30-09-2019

Fecha de entrega: 14-10-2019

Marco Teórico

AUTOMATAS FINITOS

Un autómata finito tiene un conjunto de estados y su "control" pasa de un estado a otro en respuesta a las "entradas" externas. Una de las diferencias fundamentales entre las clases de autómatas finitos es si dicho control es "determinista", lo que quiere decir que el autómata no puede encontrarse en más de un estado a un mismo tiempo, o "no determinista", lo que significa que sí puede estar en varios estados a la vez. [1] [2]

AUTOMATA FINITO DETERMINISTA (AFD)

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. [2]

Un autómata finito determinista M es una colección de cinco elementos [3]

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

Donde:

- Q el un conjunto finito de estados**,** denotados por q0,q1,q2,...q0,q1,q2,...
- 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.

$$\delta(q_i,a)=q_j \delta(qi,a)=qj$$

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 [4]

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

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

- 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).

Existen dos notaciones más cómodas para describir los autómatas: [2]

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.**

- **EJEMPLO:**

La siguiente figura muestra el diagrama de transiciones de un AFD. En este diagrama podemos ver los tres nodos correspondientes a los tres estados. Hay una flecha etiquetada como Inicio que entra en el estado inicial, q0, y un estado de aceptación, q1, representado mediante un doble círculo. De cada estado sale un arco etiquetado con 0 y otro con 1 (aunque los dos arcos se han combinado en uno con una doble etiqueta en el caso de q1). [2]

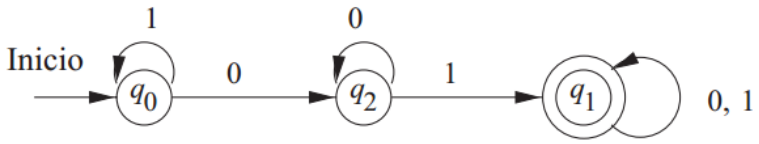

Imagen 1 *Diagrama de transiciones del AFN que acepta todas las cadenas que contienen la subcadena 01.*

	0	1
$\rightarrow q_0$	q_2	q_0
$*q_1$	q_1	q_1
q_2	q_2	q_1

Tabla 1 *Transiciones del AFD*

AUTOMATA FINITO NO DETERMINISTA (AFN)

Un autómata finito "no determinista" (AFN) tiene la capacidad de estar en varios estados a la vez. Esta capacidad a menudo se expresa como la posibilidad de que el autómata "conjeture" algo acerca de su entrada. [2]

Un autómata finito determinista M es una colección de objetos [3]

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

Donde:

- Q el un conjunto finito de estados
- Σ es el alfabeto, es decir, un conjunto finito de símbolos que formaran palabras o cadenas.
- Δ es una relación sobre (QXΣ)XQ y se llama relativa de transición.
- S es el estado inicial, el estado en qué el autómata se encuentra inicialmente.
- F es una colección de estados de aceptación o finales.

Una cadena w es aceptada por un AFN si es posible que:

$$q_{f\,\in\Delta(s,w)} \wedge\ q_f\ \in F$$

Si x⊆Q, vamos a interpretar Δ(X,σ) como el conjunto de estados {p|q∈X y p∈Δ(q,σ)}

Por tanto Δ(X,σ) es el conjunto de todos los estados siguientes a los que se puede llegar desde X con la entrada σ.

$$\Delta(X,\sigma)=Uq\in X\ \Delta(q,\sigma)$$

- **EJEMPLO:**

El siguiente autómata consiste en aceptar todas y sólo las cadenas formadas por ceros y unos que terminan en 01. [2]

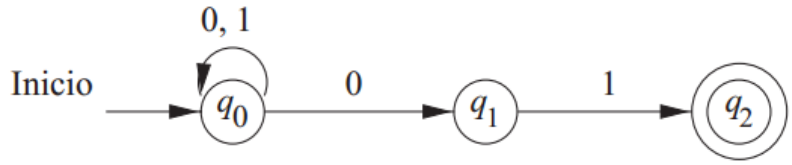

Imagen 2 *AFN que acepta todas las cadenas que terminan en 01*

Procesamiento de entradas del AFN con la cadena 00101:

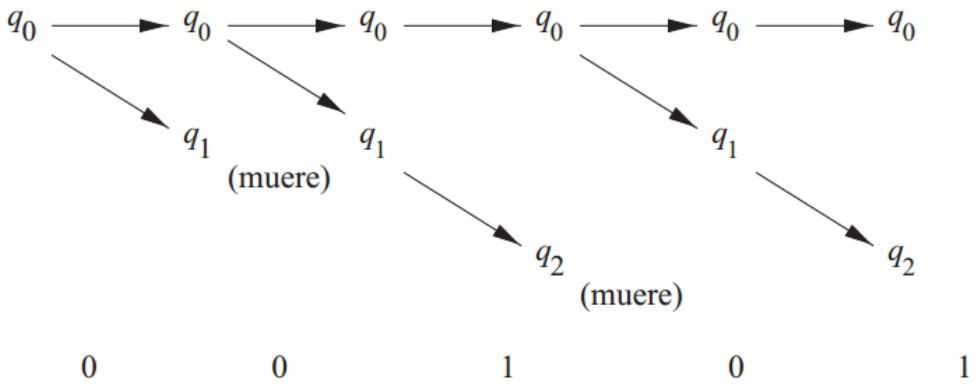

Imagen 3 *Estados de un AFN durante el procesamiento de la secuencia de entrada 00101*

EQUIVALENCIA DE AUTOMATAS FINITOS DETERMINISTAS Y NO DETERMINISTAS

La diferencia entre los AFD y los AFN se encuentra en el tipo de función δ. En los AFN, δ es una función que toma un estado y símbolos de entrada como argumentos (al igual que la función de transición del AFD), pero devuelve un conjunto de cero, uno o más estados (en lugar de devolver exactamente un estado, como lo hacen los AFD). [2]

La demostración de que los AFD pueden hacer lo que hacen los AFN implica una "construcción" importante conocida como *construcción de subconjuntos*, porque exige construir todos los subconjuntos del conjunto de estados del AFN. [4] [2]

Sea M=(Q,Σ,Δ,s,F) un AFN, entonces existe un AFD M'=(Q',Σ',δ',s',F') que es equivalente a M'

Definimos M'=(Q',Σ',δ',s',F') como:

- $Q'=2^Q$

- Σ'=Σ
- s'={s}
- F'={X ∈ Q' | q ∈ F ∧ q ∈ X}

Para cada conjunto de Q' y cada σ ∈ Σ.

δ ($\{q_{i1}, q_{i2}, \ldots\ldots\ldots, q_{in}\}$,σ)=$\{P_1, P_2, \ldots\ldots\ldots, P_K\}$

Si y solo si

Δ ($\{q_{i1}, q_{i2}, \ldots\ldots\ldots, q_{in}\}$,σ)=$\{P_1, P_2, \ldots\ldots\ldots, P_K\}$

- **EJEMPLO:**

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$*\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$*\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

Imagen 4 *Construcción del subconjunto completo de la imagen 2*

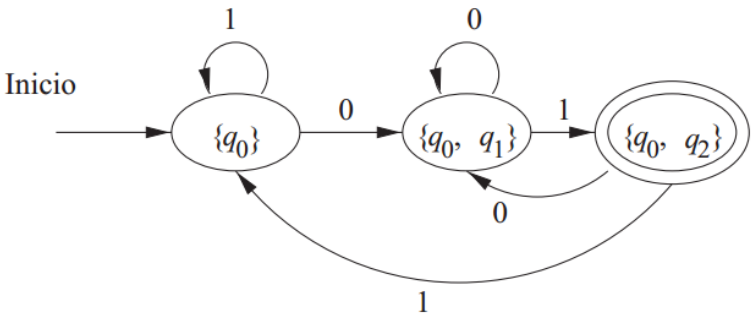

Imagen 5 *AFD construido a partir del AFN de la imagen 2*

Material y equipo

- Sistema Operativo Ubuntu
- Python 3

Desarrollo de la práctica

Objetivo: programar un AFD a partir de un AFN

Ejercicios:

1) **Hacer un programa en python para convertir el siguiente AFN en un AFD**

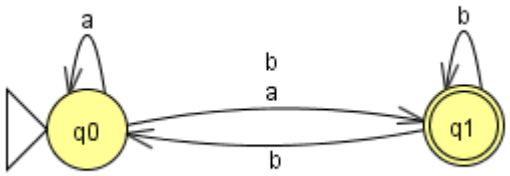

Imagen 6 AFN del ejercicio 1

El código es el siguiente:

```
Abrir   p4.py ~/Documentos   Guardar

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))

Q=['q0','q1']

Qprima=list(powerset(Q))

print("Qprima:",Qprima)

s='q0'

F=['q1']

Sigma=['a','b']

sprima=(s,)

print("sprima:",sprima)

Sigmaprima=Sigma

print("Sigmaprima:",Sigmaprima)

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

print("Construyendo Fprima:")
Fprima=[]
for x in Qprima:
        print(x)
        for q in x:
                print(q)

Python   Anchura del tabulador: 8   Ln 41, Col 14   INS
```

Imagen 7 código AFN ejercicio 1

En la imagen 7 se muestra la definición de las partes que conforman el AFN a convertir y empieza la equivalencia a AFD

```
Abrir   p4.py ~/Documentos   Guardar

                print(q)
                if q in F:
                        Fprima.append(x)
print("Fprima:")
print(Fprima)

print("Construyendo delta: ")
delta={}

for x in Qprima:
        for s in Sigmaprima:
                estados_siguientes=[]
                for q in x:
                        estados_q_s=DELTA[(q,s)]
                        for m in estados_q_s:
                                if not (m in estados_siguientes):
                                        estados_siguientes.append(m)
                estados_siguientes.sort()
                delta[(x,s)]=tuple(estados_siguientes)

print("DELTA:")
print(delta)

#transiciones
def transicion(x,s):
        global delta,Sigmaprima
        if not (s in Sigmaprima):
                print(s,"NO ESTA EN EL ALFABETO")
                return False,''
        estado_siguiente=delta[(x,s)]
        return True,estado_siguiente
```

Imagen 8 *Código AFN-AFD*

En la imagen 8 se muestra la equivalencia de los componentes restantes de la conversión de AFN a AFD y se establecen las condiciones para las transiciones del autómata.

```
w=['aaab','','b','ba']

for c in w:
        estado=sprima
        for l in c:
                y,estado=transicion(estado,l)
        if y==True and (estado in Fprima):
                print(c,"SI ESTA EN EL LENGUAJE")
        else:
                print(c,"NO ESTA EN LE LENGUAJE")
```

Imagen 9 *Condiciones para que el autómata reconozca cadenas*

En la imagen 9 se definen algunas cadenas que estén o no dentro del lenguaje para comprobar el funcionamiento del autómata

El resultado es el siguiente:

```
ubuntu@ubuntu: ~/Documentos
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ubuntu:~$ cd Documentos;
ubuntu@ubuntu:~/Documentos$ python3 p4.py
Qprima: [(), ('q0',), ('q1',), ('q0', 'q1')]
sprima: ('q0',)
Sigmaprima: ['a', 'b']
Construyendo Fprima:
()
('q0',)
q0
('q1',)
q1
('q0', 'q1')
q0
q1
Fprima:
[('q1',), ('q0', 'q1')]
Construyendo delta:
DELTA:
{((), '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')}
aaab SI ESTA EN EL LENGUAJE
 NO ESTA EN LE LENGUAJE
b SI ESTA EN EL LENGUAJE
ba NO ESTA EN LE LENGUAJE
ubuntu@ubuntu:~/Documentos$
```

Imagen 10 *Resultado de la conversión de autómatas*

2) Hacer un programa en python para convertir el siguiente AFN en un AFD

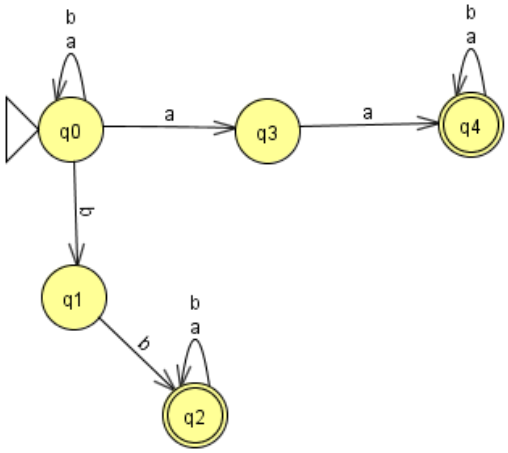

Imagen 11 *AFN del ejercicio 2*

El código es el siguiente:

```
Abrir   practica4.py ~/Documentos   Guardar

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))

Q=['q0','q1','q2','q3','q4']

Qprima=list(powerset(Q))

print("\tQprima:")
print("==================================")
print(Qprima)
print("\n")

s='q0'

F=['q2','q4']

Sigma=['a','b']

sprima=(s,)

print("\tsprima:")
print("==================================")
print(sprima)
print("\n")

Sigmaprima=Sigma

print("\tSigmaprima:")
print("==================================")
print(Sigmaprima)
print("\n")

DELTA = { ('q0','a'): ['q0','q3'],
          ('q0','b'): ['q0','q1'],

Python   Anchura del tabulador: 8   Ln 103, Col 36   INS
```

Imagen 13 *Código AFN ejercicio 1*

En la imagen 13 se muestra la definición de las partes que conforman el AFN a convertir y empieza la equivalencia a AFD

Abrir — practica4.py ~/Documentos — Guardar

```
DELTA = { ('q0','a'): ['q0','q3'],
          ('q0','b'): ['q0','q1'],
          ('q1','a'): [],
          ('q1','b'): ['q2'],
          ('q2','a'): ['q2'],
          ('q2','b'): ['q2'],
          ('q3','a'): ['q4'],
          ('q3','b'): [],
          ('q4','a'): ['q4'],
          ('q4','b'): ['q4']}

print("\tConstruyendo Fprima:")
print("==================================")
Fprima=[]
for x in Qprima:
        print(x)
        for q in x:
                print(q)
                if q in F:
                        Fprima.append(x)
print("\n")
print("\tFprima:")
print("==================================")
print(Fprima)
print("\n")
#print("\tConstruyendo delta: ")
#print("==================================")
delta={}

for x in Qprima:
        for s in Sigmaprima:
                estados_siguientes=[]
                for q in x:
                        estados_q_s=DELTA[(q,s)]
                        for m in estados_q_s:
                                if not (m in estados_siguientes):
                                        estados_siguientes.append(m)
```

Python — Anchura del tabulador: 8 — Ln 103, Col 36 — INS

Imagen 15 *Código AFN-AFD ejercicio 1*

En la imagen 14 se muestra la equivalencia de los componentes restantes de la conversión de AFN a AFD y se establecen las condiciones para las transiciones del autómata.

Abrir — practica4.py ~/Documentos — Guardar

```
                for q in x:
                        estados_q_s=DELTA[(q,s)]
                        for m in estados_q_s:
                                if not (m in estados_siguientes):
                                        estados_siguientes.append(m)
                estados_siguientes.sort()
                delta[(x,s)]=tuple(estados_siguientes)
print("\n")
print("\tDELTA:")
print("==================================")
print(delta)
print("\n")
#transiciones

print("\tTRANSICIONES")
print("==================================")
print("\n")

def transicion(x,s):
        global delta,Sigmaprima
        if not (s in Sigmaprima):
                print(s,"NO ESTA EN EL ALFABETO")
                return False,''
        estado_siguiente=delta[(x,s)]
        print("Transición: \t(", estado,",",Sigmaprima,") --->",estado_siguiente)
        return True,estado_siguiente
w =['bb','aa','abbaba','','a','b']

for c in w:
        estado=sprima
        for l in c:
                y,estado=transicion(estado,l)
        if y==True and (estado in Fprima):
                print("\tLa cadena:",c,"SI ESTA EN EL LENGUAJE\n")
        else:
                print("\tLa cadena:",c,"NO ESTA EN EL LENGUAJE\n")
```

Python — Anchura del tabulador: 8 — Ln 103, Col 36 — INS

Imagen 14 *Condiciones para que el autómata reconozca cadenas*

En la imagen 14 se definen algunas cadenas que estén o no dentro del lenguaje para comprobar el funcionamiento del autómata

El resultado es el siguiente:

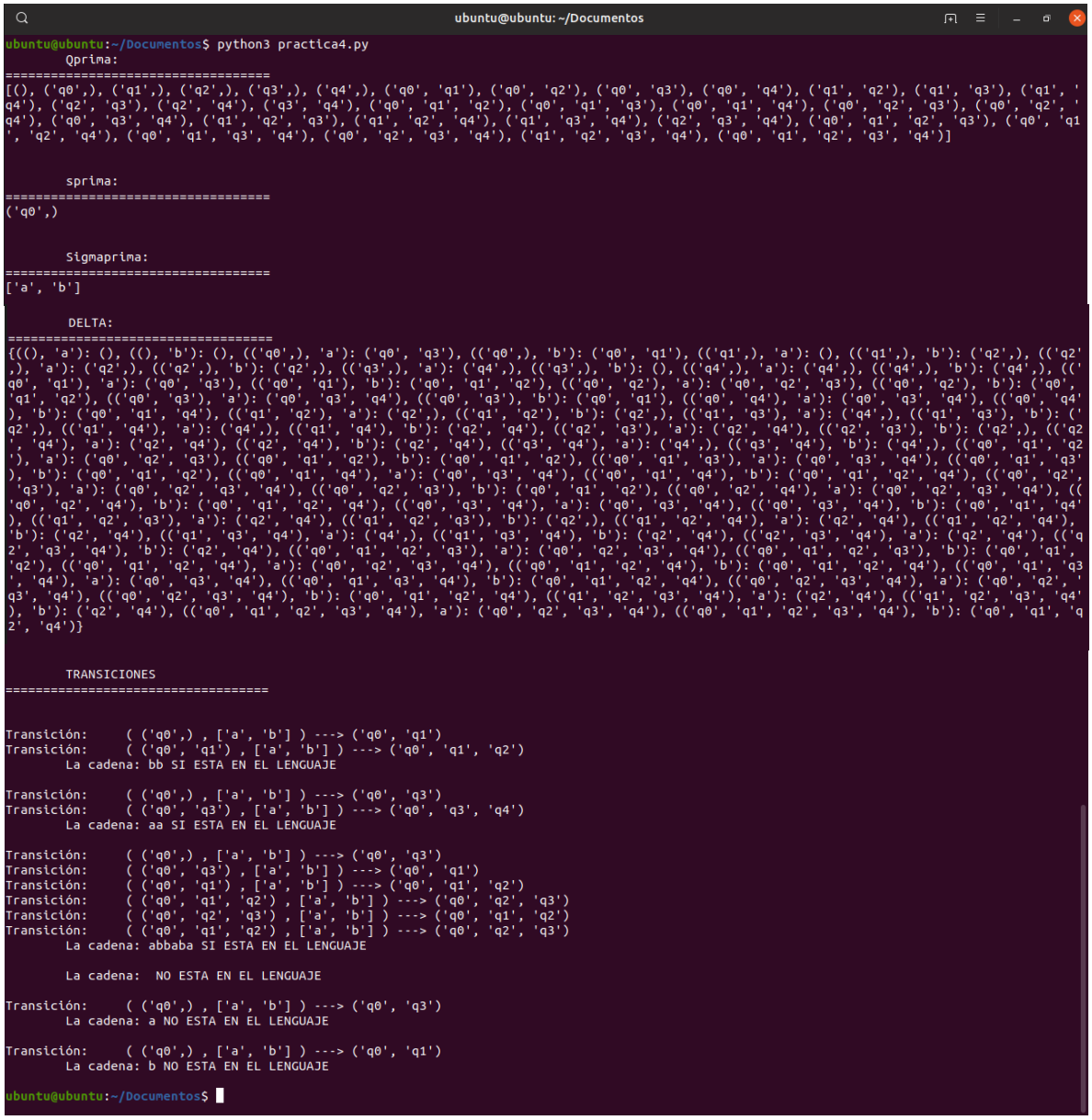

Imagen 15 *Resultado de la conversión de autómatas*

Conclusiones

En la presente práctica, aprendí como hacer un AFN a un AFD, así como a realizar un programa en python que reconozca la conversión de autómatas, este programa facilita dicha conversión ya que podemos modificar dicho programa agregando, quitando estados o haciendo alguna otra modificación.

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] J. E. Hopcroft, R. Motwani y J. D. Ullman, Introducción a la teoría de autómatas, lenguajes y computación, Madrid: PEARSON EDUCACIÓN S.A, 2007.

[3] 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.

[4] D. Kelley, Teoría de autómatas y lenguajes formales, Madrid: PEARSON EDUCACIÓN, S. A, 1995.

[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.