INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

TEORÍA COMPUTACIONAL

PRÁCTICA 05

ALUMNO: HERNÁNDEZ CASTELLANOS CÉSAR URIEL

PROFESOR: ROSAS TRIGUEROS JORGE LUIS

FECHA DE REALIZACIÓN: 13 DE SEPTIEMBRE DEL 2017

FECHA DE ENTREGA: 17 DE SEPTIEMBRE DEL 2017

¿Pero en que puedo usar un autómata?

Los autómatas son modelos de computadoras y su creador Alan Turing padre de la computación, desde hace años (30) estudió una maquina abstracta que poseía la misma capacidad de las computadoras actuales. Su objetivo era determinar la frontera entre lo que puede y no puede hacer una computadora, y aun cuando sus estudios están basados en estas maquinas abstractas son aplicables hoy en día a nuestros Pcs.

Imagen 1.0 Alan Turing

Autómatas finitos deterministas (AFD)

Un autómata finito determinista (AFD) es una quíntupla

$$M = (\Sigma, Q, \delta, q_0, F)$$

donde

- Σ es un alfabeto (sabemos $\epsilon \notin \Sigma$)
- Q es un conjunto finito no vacío de estados, es decir, $0 < |Q| < \infty$.
- δ es una *función* de transición:

$$\delta : Q \times \Sigma \longrightarrow Q \; ; \; \delta(q, \sigma) = p$$

es decir, si el autómata se encuentra en el estado q y `lee' el símbolo σ va al estado

R.

- $q_0 \in Q$ es el estado inicial.
- $F \subset Q$ es el conjunto de estados finales.

Podemos pensar de un autómata como un dispositivo que lee desde una cinta con símbolos y que realiza cambios de estados internamente:

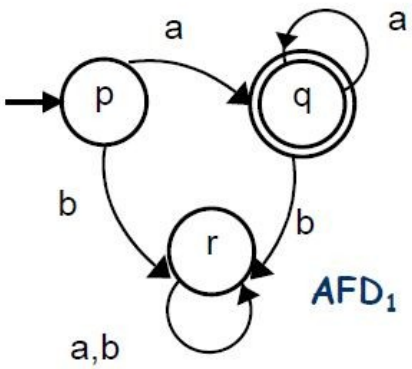

Imagen 1.1 Autómata finito determinista.

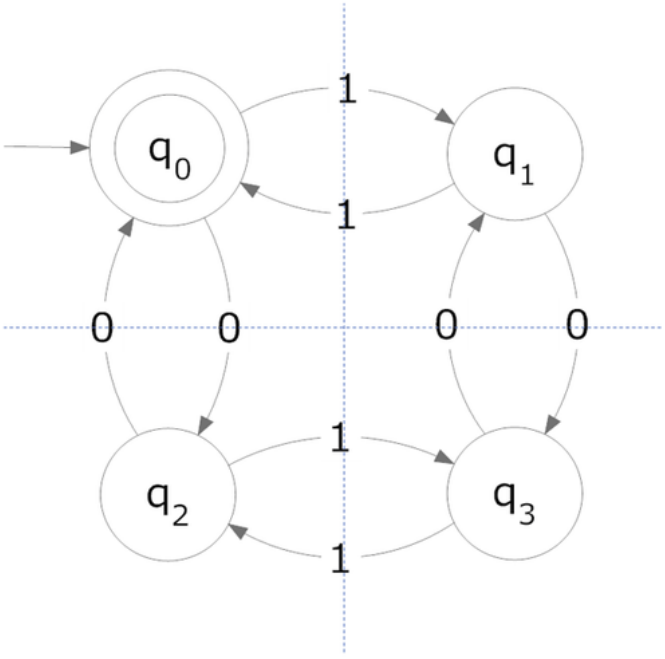

Imagen 1.2 AFD que reconocer cadenas con un número par de ceros y un números par de unos.

Material y equipo.

El material utilizado en la práctica es el siguiente:

Herramientas de software:

* Oracle VM VirtualBox
* Ubuntu 17.04
* Python 3.6.2
* eric6 Web Browser (QtWebKit)
* JFLAP8

Herramientas de hardware:

* Computadora personal

Desarrollo de la práctica

En la práctica numero cinco de teoría computacional, se dió una introducción a los autómatas, especificamente a los autómatas finitos deterministas.

Las actividades que se realizaron en el laboratorio, fue la codificación de un autómata que reconociera cadenas que no contienen la cadena “AC”, para esto se tuvo que realizar en primera instancia un dibujo, de nuestro posible autómata, para posteriormente implementarlo en python.

Las dificultades que se presentaron en la práctica, fue el desconocimiento sobre el tema, además de que en lo personal me resultó difícil poder interpretar el autómata, más aun poder hacer uno, sin que se escapará algún caso particular.

Para disminuir las complicaciones que se presentarón, se buscó una herramienta que mostrará de manera gráfica el como opera un AFD, esto ayudo a mejorar la comprensión sobre el tema en cuestión.

Los resultados obtenidos en la práctica se muestran a continuación.

Diagramas, gráficas y pantallas.

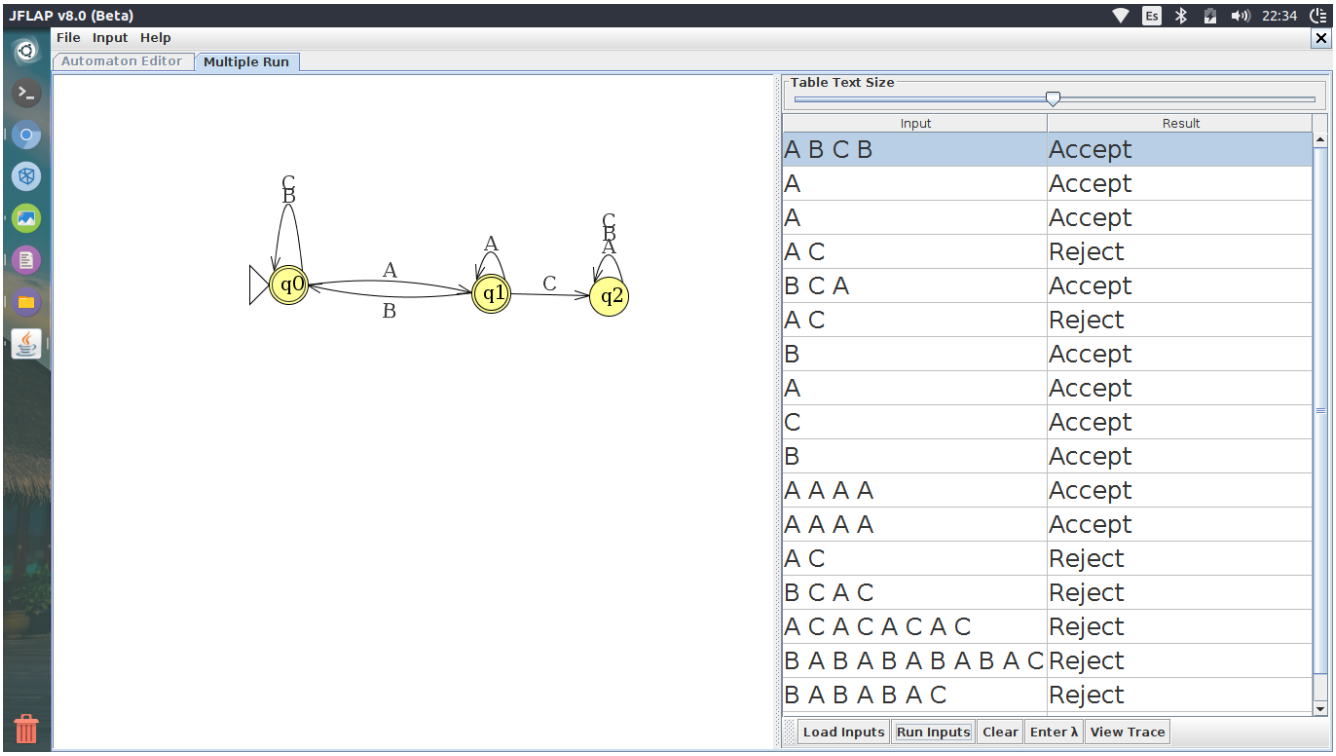

Imagen 1.3 AFD en JFLAP

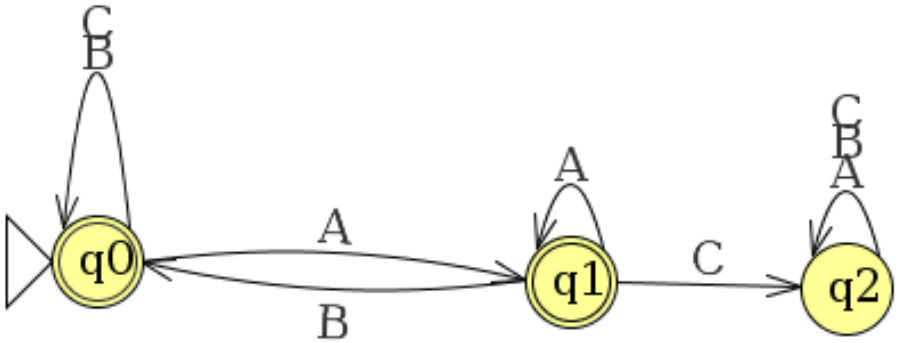

Imagen 1.4 AFD que reconocer cadenas que no contiene “AC”

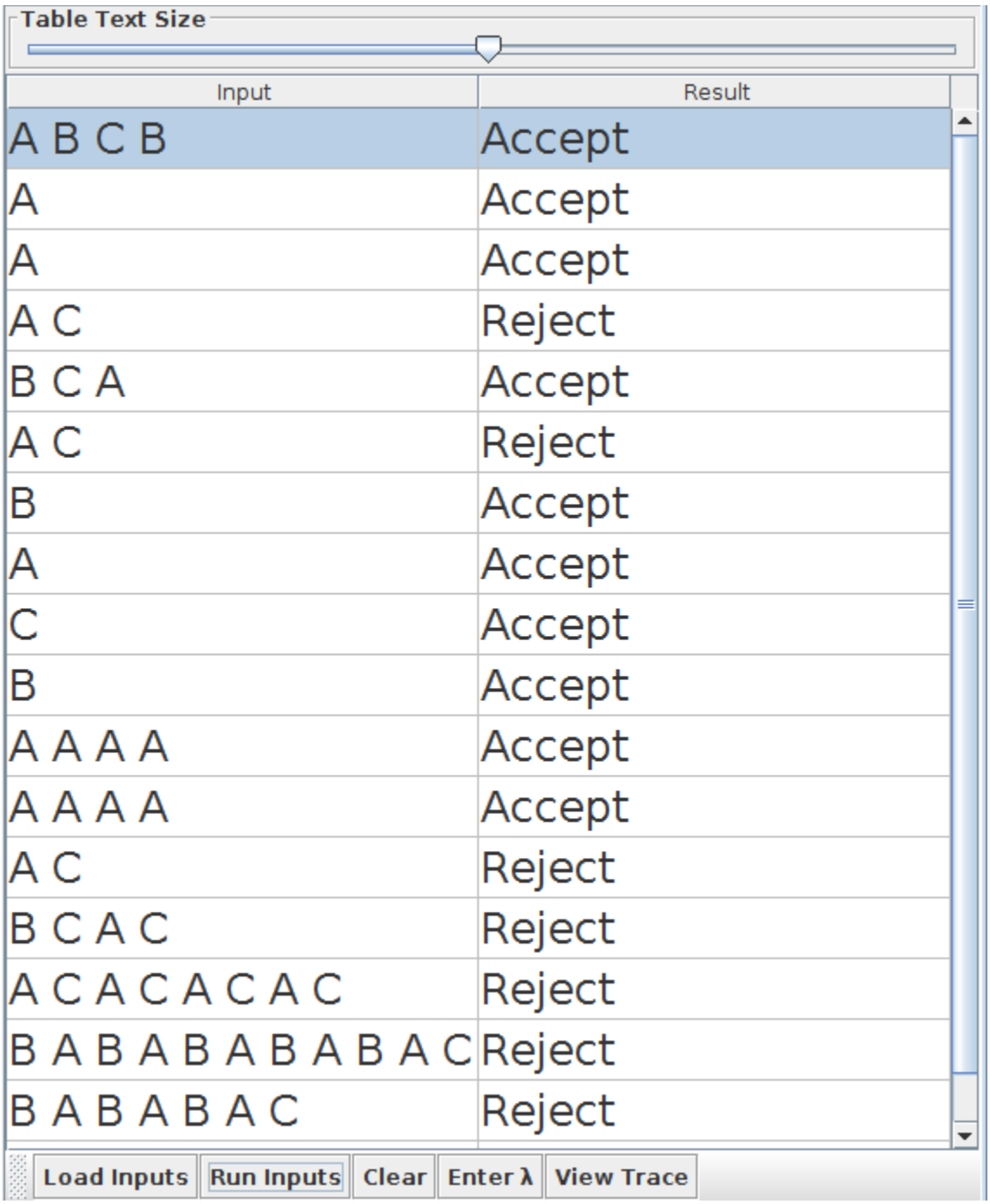

Imagen 1.5 – Pruebas con algunas cadenas.

```
Q=['q0','q1','q2']
S='q0'
Sigma = ['A','B','C']
F=['q0','q1']
delta = {
        ('q0', 'B'):'q0',
        ('q0', 'C'):'q0',
        ('q0', 'A'):'q1',
        ('q1', 'B'):'q1',
        ('q1', 'A'):'q1',
        ('q1', 'C'):'q2',
        ('q2', 'A'):'q2',
        ('q2', 'B'):'q2',
        ('q2', 'C'):'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 ('Transicion(', estado, ',' , sigma ,')->', estado_siguiente)
        return estado_siguiente, STATUS
w = 'AB'
estado = S
for sigma in w:
        estado, STATUS = transicion(estado, sigma)
        if not STATUS:
                break
if (STATUS and (estado in F)):
        print (w, 'si esta en el lenguaje')
else:
        print (w, 'no esta en el lenguaje')
```

Imagen 1.6 – Implementación en python.

```
Q=['q0','q1','q2']
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 ('Transicion(', estado, ',' , sigma ,')->', estado_siguiente)
        return estado_siguiente, STATUS

w = 'ab'
estado = S
for sigma in w:
        estado, STATUS = transicion(estado, sigma)
        if not STATUS:
                break
if (STATUS and (estado in F)):
        print (w, 'si esta en el lenguaje')
else:
        print (w, 'no esta en el lenguaje')
```

Imagen 1.7 – AFD de ejemplo en clase.

Conclusiones y recomendaciones.

Como conclusión se podría decir que usando los autómatas finitos es posible construir una máquina para cualquier tipo de expresión regular el cual describa un lenguaje o conjunto regular, por lo que es posible concluir que la famlia de lenguajes aceptados por una máquina o autómata finito son del tipo regular.

La recomendación que daría para esta práctica seria el de utilizar alguna herramienta gráfica, con el fin de observar el comportamiento de un autómata, pienso que ayudaría a la mejor comprensión del tema, sobre todo para los que son nuevos en los autómatas.

Referencias

[1]D. Kelley, *Automata and formal languages*. Upper Saddle River, NJ: Prentice Hall, 2002.

[2]"Autómata finito determinista", *Es.wikipedia.org*, 2017. [Online]. Available: https://es.wikipedia.org/wiki/Aut%C3%B3mata_finito_determinista. [Accessed: 17- Sep- 2017].

[3] Autómatas finitos deterministas, *Matesfacil.com*, 2017. [Online]. Available: https://www.matesfacil.com/automatas-lenguajes/automata-finito-y-su-lenguaje.html. [Accessed: 17- Sep- 2017].

[4]A. determinista, "AFD en Python - Automata finito determinista", *Pythondiario.com*, 2017. [Online]. Available: http://www.pythondiario.com/2015/06/afd-en-python-automata-finito.html. [Accessed: 17- Sep- 2017].

[5]"Alan Turing", *Es.wikipedia.org*, 2017. [Online]. Available: https://es.wikipedia.org/wiki/Alan_Turing. [Accessed: 17- Sep- 2017].