

Instituto Politécnico Nacional

ESCUELA SUPERIOR DE COMPUTACIÓN

TEORÍA DE LA COMPUTACIÓN

PRÁCTICA 3

Protocolo mediante un AFD

Profesor: Juarez Martinez Genaro

Alumno: Jimenez Luna Rodrigo Efren

Email: jimenez.luna.efren@gmail.com

Grupo: 4CM2

Índice

1. Introducción	2
2. Marco Teórico	2
2.1. Autómatas	2
3. Desarrollo	2
3.1. Programa 3 Protocolo mediante un AFD	2
3.2. Planteamiento y solución	3
3.3. Código	3
3.3.1. Código de la función principal (main.py)	3
3.3.2. Código de los métodos (functions.py)	4
4. Resultados	7
5. Conclusión	9

1. Introducción

Los autómatas fueron una gran herramienta durante mucho tiempo para resolver diversos tipos de problemas, a pesar de que hoy en día existen otro tipo de herramientas, estos siguen siendo bastante útiles, como se analizará más adelante, estos se pueden implementar para resolver diferentes tipos de problemas, en este caso por ejemplo confirmar si un mensaje enviado en forma binaria contiene algún tipo de error utilizando la paridad.

2. Marco Teórico

2.1. Autómatas

En informática, se entiende por autómata a un modelo matemático creado para una máquina el cuál tiene finitos estados. Este recibe una serie de símbolos y cambia de estado mediante algún tipo de "función". Los autómatas suelen recibir como entrada un alfabeto con el cuál trabajan [1].

Existen diferentes tipos de autómatas, entre ellos están los autómatas finitos deterministas o por sus siglas AFD, lo que significa que solo existe una única posibilidad para el cambio de estado [1].

Formalmente se puede definir a un AFD como una tupla de 5 valores $(Q, \Sigma, q_0, \delta, F)$ donde Q representa todo el conjunto de posibles estados del autómata; Σ como su notación lo indica es el alfabeto que recibe como entrada el AFD; q_0 representa un estado inicial ($q_0 \in Q$); δ se define como la función de transición, la cual dependiendo el estado actual y un valor de entrada nos llevará al siguiente estado y por último $F \subseteq Q$ es un subconjunto de estados finales también conocidos como estados de aceptación [2].

3. Desarrollo

3.1. Programa 3 Protocolo mediante un AFD

En esta práctica se plantea el siguiente problema: Desarrollar un programa que simule el funcionamiento de un protocolo mediante un AFD de paridad. Por lo que el programa deberá saber cuando finalizar, ejecutarse y cuando seguir de forma automática. Al comenzar con la paridad, el programa generará 10^6 cadenas binarias y comprobará si cumplen con la paridad o no, si cumplen, serán ingresadas en un archivo de texto y las que sean rechazadas se ingresarán en otro archivo de texto.

3.2. Planteamiento y solución

Para este problema lo unico que se necesita hacer es crear el autómata que va a ejecutar la paridad. Tal como se observa en la figura 1, se puede observar los pasos por los que pasa el protocolo.

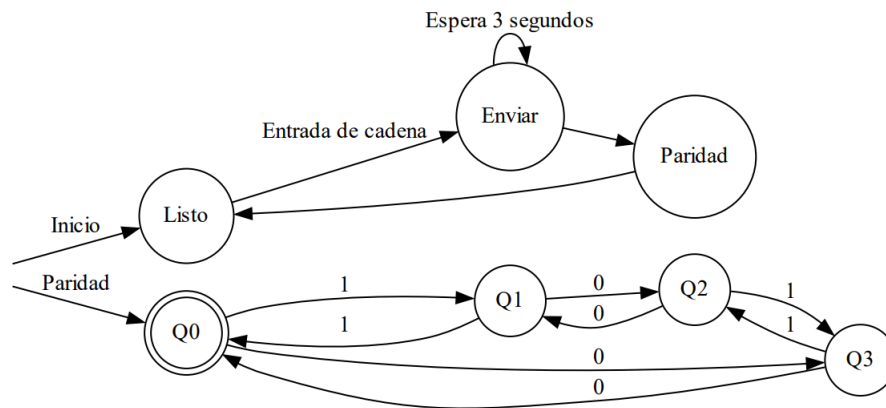


Figura 1. Diagrama del protocolo junto con el autómata de paridad. Fuente: Elaboración propia.

3.3. Código

A partir de la solución planteada en la sección anterior, se muestra el código implementado en el lenguaje de programación python, en este caso se separo en dos archivos diferentes, el primero dedicado para la función principal y el segundo para implementar los métodos:

3.3.1. Código de la función principal (main.py)

```
# IPN – Escuela Superior de Computacion
# Autor: Jimenez Luna Rodrigo Efren
# Grupo: 4CM2

from functions import *
import os

if __name__ == "__main__":
    while True:
        option = 0

        #Menu
        #####
```

```

print("Elija una de las siguientes tres opciones del menu:")
print("1) Iniciar protocolo")
print("2) Graficar Automata")
print("3) Salir")

option = int(input())

if option == 1:
    start_protocol()
elif option == 2:
    show_automata()
elif option == 3:
    break
elif option != 0:
    os.system("cls")
    print("La opción no es valida, vuelva a intentarlo!\n")
#####

```

3.3.2. Código de los métodos (functions.py)

```

# IPN – Escuela Superior de Computacion
# Autor: Jimenez Luna Rodrigo Efren
# Grupo: 4CM2

import random
import os
import time
import graphviz
from tqdm import tqdm

# Automata que comprueba si la cadena cumple con la paridad
#####
def parity_DFA():
    string = ""
    state = "q0"

    for i in range(64):
        c = str(random.getrandbits(1))
        string += c

        if state == "q0":
            if c == "1":
                state = "q1"
            else:
                state = "q3"

```

```

        elif state == "q1":
            if c == "1":
                state = "q0"
            else:
                state = "q2"
        elif state == "q2":
            if c == "1":
                state = "q3"
            else:
                state = "q1"
        elif state == "q3":
            if c == "1":
                state = "q2"
            else:
                state = "q0"

    if state == "q0":
        return True, string
    else:
        return False, string
#####

# Inicia el procotolo ya sea encendido o apagado de forma aleatoria
def start_protocol():
    state = random.getrandbits(1)
    executions = 0

    if state == 0:
        os.system("cls")
        print("El protocolo se encuentra apagado.\n".format(executions))
        return

    while state:
        state = random.getrandbits(1)

        os.system("cls")
        print("Generando cadenas\n")
        time.sleep(1)
        os.system("cls")
        print("Generando cadenas.\n")
        time.sleep(1)
        os.system("cls")
        print("Generando cadenas..\n")
        time.sleep(1)
        os.system("cls")
        print("Generando cadenas...\n")

```

```

print("Iniciando analisis de cadenas:\n")

strings_file = open("strings.txt", "a+")
accepted_file = open("approved_strings.txt", "a+")
rejected_file = open("rejected_strings.txt", "a+")

# Ejecuta el AFD de paridad para las 10^6 cadenas binarias
for i in tqdm(range(10**6)):
    result, string = parity_DFA()

    strings_file.write(string + "\n")

    if result:
        accepted_file.write(string + "\n")
    else:
        rejected_file.write(string + "\n")

accepted_file.close()
rejected_file.close()
strings_file.close()

executions += 1

os.system("cls")
print("El protocolo se ejecuto {} veces, ahora se encuentra apagado.\n".format(
    executions))

def show_automata():
    os.system("cls")

    DFA_graph = graphviz.Digraph('AFD de Paridad')
    DFA_graph.attr(rankdir='LR')

    # Inicia diagrama del protocolo
    DFA_graph.attr('node', shape='circle')
    DFA_graph.node('Listo')

    DFA_graph.attr('node', shape='plaintext')
    DFA_graph.edge('', 'Listo', label='Inicio')

    DFA_graph.attr('node', shape='circle')
    DFA_graph.edge('Listo', 'Enviar', label='Entrada de cadena')
    DFA_graph.edge('Enviar', 'Enviar', label='Espera 3 segundos')
    DFA_graph.edge('Enviar', 'Paridad', label='')
    DFA_graph.edge('Paridad', 'Listo', label='')

```

```

# Inicia diagrama del AFD
DFA_graph.attr('node', shape='doublecircle ')
DFA_graph.node('Q0')
DFA_graph.node('Paridad ')

DFA_graph.attr('node', shape='plaintext ')
DFA_graph.edge('', 'Q0', label='Paridad ')

DFA_graph.attr('node', shape='circle ')
DFA_graph.edge('Q0', 'Q1', label='1')
DFA_graph.edge('Q0', 'Q3', label='0')
DFA_graph.edge('Q1', 'Q0', label='1')
DFA_graph.edge('Q1', 'Q2', label='0')
DFA_graph.edge('Q2', 'Q1', label='0')
DFA_graph.edge('Q2', 'Q3', label='1')
DFA_graph.edge('Q3', 'Q2', label='1')
DFA_graph.edge('Q3', 'Q0', label='0')

DFA_graph.render('Automata', view=True)

```

4. Resultados

Para confirmar su funcionamiento, se desarrollo un tercer programa el cuál funcionará para comprobar si el AFD de paridad funcionó de forma correcta. Esto comprobando que las cadenas aceptadas cumplan las características y por el contrario las cadenas rechazadas ninguna cumpla la paridad. A continuación se muestra el código utilizado para tal propósito.

```

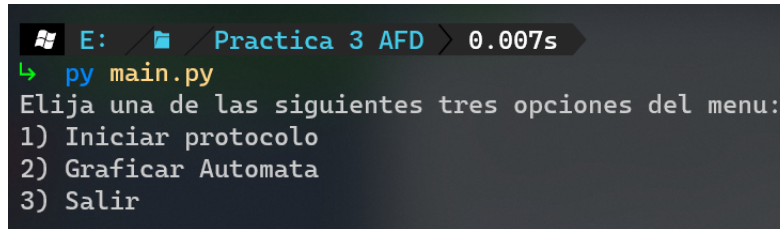
with open("approved_strings.txt") as file:
    numbers = file.readlines()
    for number in numbers:
        if not (number.count('1') % 2 and number.count('0') % 2):
            continue
        else:
            print("Test 1: Fallido")
            break

with open("rejected_strings.txt") as file:
    numbers = file.readlines()
    for number in numbers:
        if number.count('1') % 2 and number.count('0') % 2:
            continue
        else:
            print("Test 2: Fallido")

```

Para realizar el testeo, tal como se observa en la figura 2, se ejecuto el

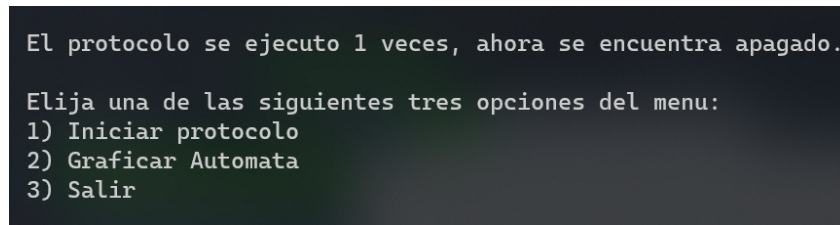
programa principal (main.py) para generar las 10^6 cadenas y posteriormente clasificarlas en los archivos. Solo hace falta seleccionar la opción 1 del menú del programa.



```
E: / Practica 3 AFD > 0.007s
py main.py
Elija una de las siguientes tres opciones del menu:
1) Iniciar protocolo
2) Graficar Automata
3) Salir
```

Figura 2. Ejecución del programa principal. Fuente: Elaboración propia.

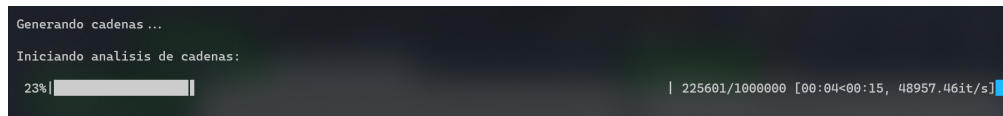
Como se puede observar en la figura 3, una vez ejecutado el AFD, muestra las veces que se ejecutó, esto debido a que el protocolo en este caso, elige de forma aleatoria cuando estar apagado o encendido.



```
El protocolo se ejecuto 1 veces, ahora se encuentra apagado.

Elija una de las siguientes tres opciones del menu:
1) Iniciar protocolo
2) Graficar Automata
3) Salir
```

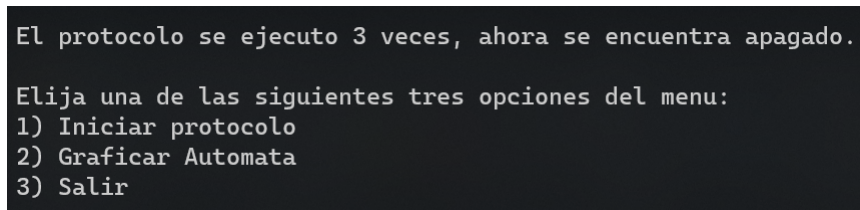
Figura 3. Se ejecuta una sola vez la paridad. Fuente: Elaboración propia.



```
Generando cadenas ...
Iniciando analisis de cadenas:
23% | 225601/1000000 [00:04<00:15, 48957.46it/s]
```

Figura 4. Ejecución del AFD de paridad. Fuente: Elaboración propia.

La figura 5 muestra como algunas veces el protocolo se ejecuta varias veces, y otras no se ejecuta tal como muestra la figura 6.



```
El protocolo se ejecuto 3 veces, ahora se encuentra apagado.

Elija una de las siguientes tres opciones del menu:
1) Iniciar protocolo
2) Graficar Automata
3) Salir
```

Figura 5. El protocolo se ejecuta 3 veces. Fuente: Elaboración propia.

```

El protocolo se encuentra apagado.

Elija una de las siguientes tres opciones del menu:
1) Iniciar protocolo
2) Graficar Automata
3) Salir

```

Figura 6. El protocolo no se ejecuta al primer intento. Fuente: Elaboración propia

A continuación, en la figura 7 se muestra la ejecución del test para evaluar los resultados del AFD, como resultado el programa no muestra ninguna excepción por lo que el autómata funciona correctamente.

```

E: \Practica 3 AFD > 0.009s
py test.py

E: \Practica 3 AFD > 1:27.493s

```

Figura 7. Resultados del test. Fuente: Elaboración propia.

5. Conclusión

Para la realización de esta práctica me resulto bastante interesante como funciona la complejidad de un autómata. Esto debido a que a pesar de lo complejo que pueden llegar a ser los autómatas al ver su diagrama, la complejidad siempre resulta lineal, a pesar de no ser lo más eficiente, resulta bastante simple de implementar sin complicarse con cuestiones de bucles con diferentes operaciones. Además en esta práctica se muestra un caso real en el que se puede aplicar un autómata; el algoritmo de paridad es altamente utilizado en la electrónica para la transmisión de señales, y ya que un autómata se puede implementar mediante compuertas, hace que los autómatas se conviertan en una herramienta muy útil.

Referencias

- [1] H. Wilber, “¿Qué es un autómata?” 2018. [Online]. Available: <https://medium.com/@maniakhitoccori/qu%C3%A9-es-un-aut%C3%B3mata-fbf309138755>
- [2] “Autómata finito determinista,” *Wikipedia*, 2019. [Online]. Available: https://es.wikipedia.org/wiki/Aut%C3%B3mata_finito_determinista