Instituto Politécnico Nacional

Escuela Superior de Computación

Teoría de la Computación

Práctica 9

Máquina de Turing

Profesor: Juarez Martinez Genaro

Alumno: Jimenez Luna Rodrigo Efren

Email: jimenez.luna.efren@gmail.com

Grupo: 4CM2

Índice

1. Introducción

La Máquina de Turing es un modelo teórico que simula la lógica de un computador y se utiliza para estudiar la capacidad de los sistemas de cómputo. Consiste en una cinta infinita dividida en casillas que pueden contener símbolos. La cabeza de lectura/escritura se mueve a lo largo de la cinta, leyendo y escribiendo símbolos según las reglas de transición definidas.

El lenguaje $0^n1^n|n \geq 1$ es un ejemplo clásico que no puede ser reconocido por una gramática libre de contexto, pero sí puede ser reconocido por una Máquina de Turing. Esto se debe a que el lenguaje requiere contar la cantidad de ceros y luego asegurarse de que la cantidad de unos sea igual a la cantidad de ceros. La capacidad de una Máquina de Turing para realizar cómputos y realizar seguimiento de información permite manejar este tipo de lenguajes más complejos.

En esta práctica, se implementará una Máquina de Turing específicamente diseñada para aceptar las cadenas del lenguaje $0^n1^n|n \geq 1$. A través de la definición de estados, símbolos y reglas de transición, se obtendrá la Máquina de Turing que reconociera y aceptara las cadenas que cumplen con esta estructura específica.

2. Marco Teórico

2.1. Máquina de Turing

La Máquina de Turing es un modelo teórico propuesto por Alan Turing en 1936, que ha demostrado ser un instrumento fundamental en el estudio de la computabilidad y la teoría de la computación. Esta máquina es capaz de simular cualquier algoritmo computacional, lo que la convierte en una herramienta poderosa para analizar la capacidad de los sistemas de cómputo [1].

Esta consta de una cinta infinita dividida en casillas, donde cada casilla puede contener un símbolo de un alfabeto dado. También posee una cabeza de lectura/escritura que se desplaza a lo largo de la cinta y puede leer y escribir símbolos en las casillas. La Máquina de Turing utiliza una serie de estados y reglas de transición para guiar su funcionamiento [2].

Las reglas de transición especifican cómo cambia de estado en respuesta al símbolo leído y la acción a realizar. Estas reglas pueden incluir la escritura de un nuevo símbolo en la casilla actual, mover la cabeza hacia la izquierda o hacia la derecha, y cambiar de estado. El comportamiento de la Máquina de Turing está determinado por su configuración inicial y las reglas de transición [2].

3. Desarrollo

3.1. Programa 9 Máquina de Turing

En esta práctica se plantea el siguiente problema: Implementar la máquina de Turing que reconozca el lenguaje $0^n1^n|n \geq 1$ donde el usuario mediante un menú indica si desea generar una la cadena de forma aleatoria o ingresarla manualmente. Estas cadenas no pueden superar los 1000 caracteres. En el caso que la cadena sea de longitud 10 o menor, el programa deberá animar el proceso que sigue la máquina de Turing así como en cualquier caso imprimir dentro de un archivo de texto las descripciones instantaneas de la Máquina.

3.2. Planteamiento y solución

Para la solución de este problema hace falta obtener las transiciones de la máquina para el lenguaje en específico. Transiciones que son proporcionadas en la asignación y se muestran en la figura 1.

	Symbol				
State	0	1	X	Y	B
q_0	(q_1, X, R)	–	–	(q_3, Y, R)	–
q_1	$(q_1, 0, R)$	(q_2, Y, L)	–	(q_1, Y, R)	–
q_2	$(q_2, 0, L)$	–	(q_0, X, R)	(q_2, Y, L)	–
q_3	–	–	–	(q_3, Y, R)	(q_4, B, R)
q_4	–	–	–	–	–

Figura 1. Tabla de transiciones de la máquina de Turing a implementar. Fuente: [3].

Para la animación será necesario el uso de la librería Tkinter de python que permite realizar interfaces de usuario

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)

```
import os
import random

from functions import *

option = 1
```

```
while option:
    # Menu
    ##########################################################################
    print("Elija una de las siguientes opciones:")
    print("1) Modo Manual")
    print("2) Modo Aleatorio")
    print("3) Salir")

    option = int(input())

    os.system("clear")

    if option in (1, 2, 3):
        if option == 1:
            print("Ingrese una cadena de 0's y 1's menor de 1,000 caracteres: ")
            str = input()
            # print("{}".format(str))

            if validateStr(str):
                print("La cadena contiene caracteres diferentes de 0 y 1!\n")

            if len(str) > 1000:
                print("La cadena contiene m s caracteres de los permitidos\n")
                continue
        elif option == 2:
            str = createStr(random.randint(1, 1000))
            print("Evaluando la cadena de tama o {} en maquina de turing...".fo
        else:
            print("Sesion terminada")
            break
    else:
        print("Opcion Invalida")
        continue;

    if len(str) <= 10:
        if animateTM(str):
            print("Presione <Enter> para continuar\n")
            input()
            os.system("clear")
        else:
            print("Presione <Enter> para continuar\n")
            input()
            os.system("clear")
    else:
        if TM(str):
```

```
            print("\nLa cadena es v lida!\n")
            print("Presione <Enter> para continuar\n")
            input()
            os.system("clear")
        else:
            print("\nLa cadena no se encuentra en el lenguaje {0^n 1^n | n >= 1}
            print("Presione <Enter> para continuar\n")
            input()
            os.system("clear")
```

3.3.2. Código de los métodos utilizados (functions.py)

```
import random
import tkinter as tk
import time

def validateStr(str):
    for c in str:
        if not c in ("1", "0"):
            return 1

    return 0

def createStr(size):
    aproved = random.getrandbits(1)
    string = ""

    if aproved:
        for i in range(size // 2):
            string += "0"
        for i in range(size - size // 2):
            string += "1"
    else:
        for i in range(size):
            string += str(random.getrandbits(1))

    return string

def TM(str):
    str += "  "
    output = open("output.txt", "w")
    state = "Q0"
    symbol = ""
    pointer = 0
```

```
output.write("Cadena: " + str + "\n\n")

while True:
    if state == "Q0":
        if str[pointer] == "0":
            state = "Q1"
            symbol = "X"
            str = str[0:pointer] + symbol + str[pointer + 1:]
            pointer += 1
            output.write("(" + state + ", " + symbol + ", R)    \n")
        elif str[pointer] == "1":
            output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
            output.close()
            return 0
        elif str[pointer] == "X":
            output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
            output.close()
            return 0
        elif str[pointer] == "Y":
            state = "Q3"
            symbol = "Y"
            str = str[0:pointer] + symbol + str[pointer + 1:]
            pointer += 1
            output.write("(" + state + ", " + symbol + ", R)    \n")
        elif str[pointer] == " ":
            output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
            output.close()
            return 0
    if state == "Q1":
        if str[pointer] == "0":
            state = "Q1"
            symbol = "0"
            str = str[0:pointer] + symbol + str[pointer + 1:]
            pointer += 1
            output.write("(" + state + ", " + symbol + ", R)    \n")
        elif str[pointer] == "1":
            state = "Q2"
            symbol = "Y"
            str = str[0:pointer] + symbol + str[pointer + 1:]
            pointer -= 1
            output.write("(" + state + ", " + symbol + ", L)    \n")
        elif str[pointer] == "X":
            output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
            output.close()
            return 0
        elif str[pointer] == "Y":
```

```
            state = "Q1"
            symbol = "Y"
            str = str[0:pointer] + symbol + str[pointer + 1:]
            pointer += 1
            output.write("(" + state + ", " + symbol + ", R)     \n")
        elif str[pointer] == " ":
            output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
            output.close()
            return 0
    if state == "Q2":
        if str[pointer] == "0":
            state = "Q2"
            symbol = "0"
            str = str[0:pointer] + symbol + str[pointer + 1:]
            pointer -= 1
            output.write("(" + state + ", " + symbol + ", L)     \n")
        elif str[pointer] == "1":
            output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
            output.close()
            return 0
        elif str[pointer] == "X":
            state = "Q0"
            symbol = "X"
            str = str[0:pointer] + symbol + str[pointer + 1:]
            pointer += 1
            output.write("(" + state + ", " + symbol + ", R)     \n")
        elif str[pointer] == "Y":
            state = "Q2"
            symbol = "Y"
            str = str[0:pointer] + symbol + str[pointer + 1:]
            pointer -= 1
            output.write("(" + state + ", " + symbol + ", L)     \n")
        elif str[pointer] == " ":
            output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
            output.close()
            return 0
    if state == "Q3":
        if str[pointer] == "0":
            output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
            output.close()
            return 0
        elif str[pointer] == "1":
            output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
            output.close()
            return 0
        elif str[pointer] == "X":
```

```
                output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
                output.close()
                return 0
            elif str[pointer] == "Y":
                state = "Q3"
                symbol = "Y"
                str = str[0:pointer] + symbol + str[pointer + 1:]
                pointer += 1
                output.write("(" + state + ", " + symbol + ", R)     \n")
            elif str[pointer] == " ":
                state = "Q4"
                symbol = "B"
                str = str[0:pointer] + symbol + str[pointer + 1:]
                pointer += 1
                output.write("(" + state + ", " + symbol + ", R)     \n")
        if state == "Q4":
            break

    output.close()

    return 1

def animateTM(str):
    output = open("output.txt", "w")
    state = "Q0"
    symbol = ""
    pointer = 0
    start = 10
    str += "  "

    output.write("Cadena: " + str + "\n\n")

    root = tk.Tk()
    root.title("Push Down Automata Animation")
    root.geometry("600x400")
    canvas = tk.Canvas(root, width=600, height=400)
    canvas.pack()

    canvas.create_rectangle(265, 50, 335, 120, fill="lightgreen")
    canvas.create_line(300, 120, 300, 170, arrow=tk.LAST, width=2)
    canvas.create_text(300, 85, text=state, font=("Arial", 14), anchor=tk.CENTER

    for i in range(22):
        canvas.create_rectangle(-15 + i * 30, 170, 15 + i * 30, 200, fill="light
        if i >= start - pointer and i < start + len(str) - 2 - pointer:
            canvas.create_text(i * 30, 185, text=str[i - start + pointer], font=
```

```
        else:
            canvas.create_text(i * 30, 185, text="B", font=("Arial", 14), anchor

    while True:
        root.update()
        time.sleep(.5)
        canvas.delete(tk.ALL)

        if state == "Q0":
            if str[pointer] == "0":
                state = "Q1"
                symbol = "X"
                str = str[0:pointer] + symbol + str[pointer + 1:]
                pointer += 1
                output.write("(" + state + ", " + symbol + ", R)     \n")
            elif str[pointer] == "1":
                output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
                output.close()
                break
            elif str[pointer] == "X":
                output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
                output.close()
                break
            elif str[pointer] == "Y":
                state = "Q3"
                symbol = "Y"
                str = str[0:pointer] + symbol + str[pointer + 1:]
                pointer += 1
                output.write("(" + state + ", " + symbol + ", R)     \n")
            elif str[pointer] == " ":
                output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
                output.close()
                break
        elif state == "Q1":
            if str[pointer] == "0":
                state = "Q1"
                symbol = "0"
                str = str[0:pointer] + symbol + str[pointer + 1:]
                pointer += 1
                output.write("(" + state + ", " + symbol + ", R)     \n")
            elif str[pointer] == "1":
                state = "Q2"
                symbol = "Y"
                str = str[0:pointer] + symbol + str[pointer + 1:]
                pointer -= 1
                output.write("(" + state + ", " + symbol + ", L)     \n")
```

```
    elif str[pointer] == "X":
        output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
        output.close()
        break
    elif str[pointer] == "Y":
        state = "Q1"
        symbol = "Y"
        str = str[0:pointer] + symbol + str[pointer + 1:]
        pointer += 1
        output.write("(" + state + ", " + symbol + ", R)    \n")
    elif str[pointer] == " ":
        output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
        output.close()
        break
elif state == "Q2":
    if str[pointer] == "0":
        state = "Q2"
        symbol = "0"
        str = str[0:pointer] + symbol + str[pointer + 1:]
        pointer -= 1
        output.write("(" + state + ", " + symbol + ", L)    \n")
    elif str[pointer] == "1":
        output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
        output.close()
        break
    elif str[pointer] == "X":
        state = "Q0"
        symbol = "X"
        str = str[0:pointer] + symbol + str[pointer + 1:]
        pointer += 1
        output.write("(" + state + ", " + symbol + ", R)    \n")
    elif str[pointer] == "Y":
        state = "Q2"
        symbol = "Y"
        str = str[0:pointer] + symbol + str[pointer + 1:]
        pointer -= 1
        output.write("(" + state + ", " + symbol + ", L)    \n")
    elif str[pointer] == " ":
        output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
        output.close()
        break
elif state == "Q3":
    if str[pointer] == "0":
        output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
        output.close()
        break
```

```
            elif str[pointer] == "1":
                output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
                output.close()
                break
            elif str[pointer] == "X":
                output.write("\nLa cadena no se encuentra en el lenguaje {0^n 1^
                output.close()
                break
            elif str[pointer] == "Y":
                state = "Q3"
                symbol = "Y"
                str = str[0:pointer] + symbol + str[pointer + 1:]
                pointer += 1
                output.write("(" + state + ", " + symbol + ", R)     \n")
            elif str[pointer] == " ":
                state = "Q4"
                symbol = "B"
                str = str[0:pointer] + symbol + str[pointer + 1:]
                pointer += 1
                output.write("(" + state + ", " + symbol + ", R)     \n")
        elif state == "Q4":
            break

        canvas.create_rectangle(265, 50, 335, 120, fill="lightgreen")
        canvas.create_line(300, 120, 300, 170, arrow=tk.LAST, width=2)
        canvas.create_text(300, 85, text=state, font=("Arial", 14), anchor=tk.CE

        for i in range(22):
            canvas.create_rectangle(-15 + i * 30, 170, 15 + i * 30, 200, fill="l
            if i >= start - pointer and i < start + len(str) - 2 - pointer:
                canvas.create_text(i * 30, 185, text=str[i - start + pointer], f
            else:
                canvas.create_text(i * 30, 185, text="B", font=("Arial", 14), an

    canvas.create_rectangle(265, 50, 335, 120, fill="lightgreen")
    canvas.create_line(300, 120, 300, 170, arrow=tk.LAST, width=2)
    canvas.create_text(300, 85, text=state, font=("Arial", 14), anchor=tk.CENTER

    for i in range(22):
        canvas.create_rectangle(-15 + i * 30, 170, 15 + i * 30, 200, fill="lightg
        if i >= start - pointer and i < start + len(str) - 2 - pointer:
            canvas.create_text(i * 30, 185, text=str[i - start + pointer], font=
        else:
            canvas.create_text(i * 30, 185, text="B", font=("Arial", 14), anchor

    if state == "Q4":
```

```
        canvas.create_text(300, 250, text="Cadena v lida!", font=("Arial", 14),
    else:
        canvas.create_text(300, 250, text="La cadena no pertenece al lenguaje!",

    root.mainloop()
    output.close()

    if state == "Q4":
        return 1
    else:
        return 0
```

4. Resultados

Para mostrar el funcionamiento del programa, se realizaron 3 pruebas. Para la primera de ellas, se ingreso la cadena 000111 de forma manual, cadena para la cual el programa indica la animación de la máquina tal como lo muestra la figura 2. Y tal como se esperaría el programa marca la cadena como válida además de imprimir todas las descripciones instantáneas en el archivo output.txt. El contenido de este archivo se muestra en la figura 3.

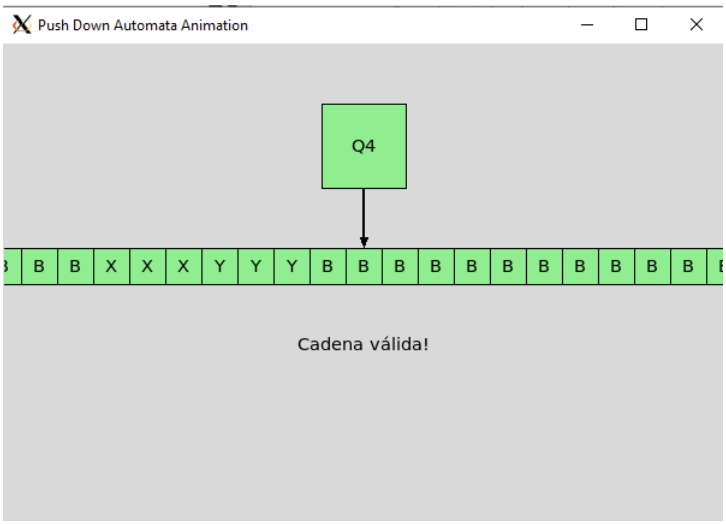

Figura 2. Resultado de ingresar la cadena 000111 en modo manual. Fuente: elaboración propia.

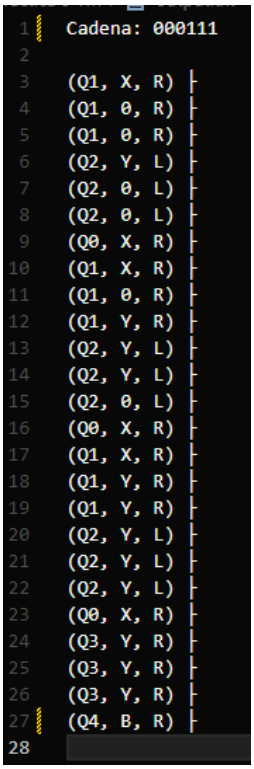

Figura 3. Contenido del archivo output.txt. Fuente: elaboración propia.

Para la siguiente prueba se ingreso la cadena 00011101 y tras la animación el programa muestra el resultado de la figura 4. Y como se esperaría, la cadena no pertenece al lenguaje.

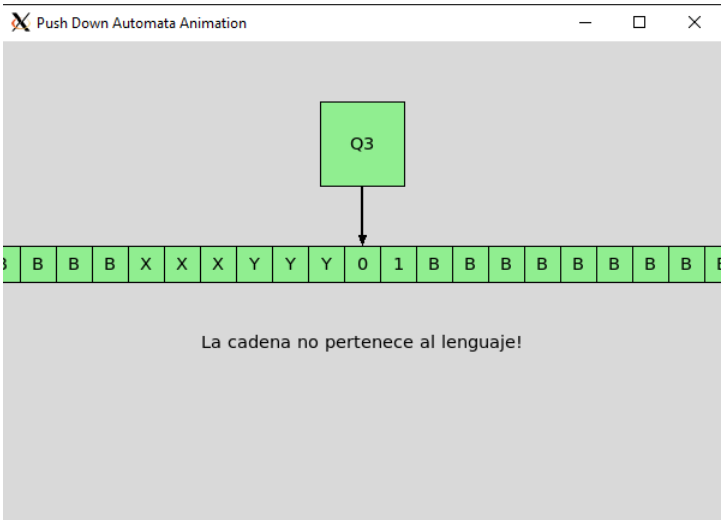

Figura 4. Resultado de la animación al ingresar la cadena 00011101. Fuente: Elaboración propia

Por último se realizó una tercera prueba mediante el modo aleatorio. En este modo, la cadena que es generada tiene una probabilidad de .25 % de ser válida. Para esta ejecución la cadena resultante fue de 96 caracteres resultando en una cadena que no pertenece al lenguaje, el resultado se muestra en la figura 5 y en la figura 6 se muestra el contenido del archivo output.txt.

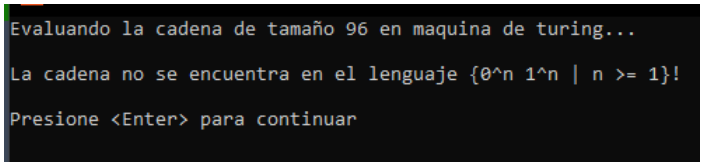

Figura 5. Resultado del modo aleatorio. Fuente: Elaboración propia

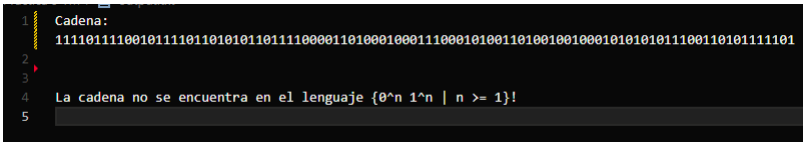

Figura 6. Contenido del archivo output.txt. Fuente: Elaboración propia

Obsérvese que en el archivo no se muestra ninguna descripción instantánea a diferencia de la figura 3, esto es debido a que la máquina de Turing se detiene instantáneamente al detectar un 1 sin haber leído ningún 0.

5. Conclusión

En esta práctica, se implementó una Máquina de Turing en Python para reconocer el lenguaje $0^n1^n | n \geq 1$, que consiste en cadenas que contienen una secuencia de ceros seguida de la misma cantidad de unos. Se observó que la máquina de Turing es un modelo teórico poderoso que permite simular cualquier algoritmo computacional.

Su capacidad para manipular una cinta infinita y realizar transiciones de estados en función de símbolos leídos ofrece una base sólida para el estudio de la computabilidad y la teoría de la computación. Adicionalmente se obesrvó que la implementación de una Máquina de Turing requiere un diseño cuidadoso de los estados, alfabeto y reglas de transición. Es fundamental comprender las características del lenguaje y las estructuras de control disponibles en Python para lograr una implementación precisa y eficiente.

Por último el desarrollo de esta práctica brindó una comprensión más profunda de los conceptos teóricos detrás de las Máquinas de Turing, como los estados, las transiciones y las configuraciones iniciales. Además, permitió adquirir experiencia práctica en la implementación de una Máquina de Turing en un lenguaje de programación concreto como Python.

Referencias

[1] M. Sipser, *Introduction to the Theory of Computation.* Cengage Learning, 2006.

[2] J. Hopcroft, *Introduction to Automata Theory, Languages and Computation.* Pearson Edutation, 2006.

[3] "Asignación máquina de turing," 2023. [Online]. Available: https://classroom.google.com/u/1/c/NTkzNjM5MzEyNjU2/a/NjEzMTE0NTU4OTI0/details