Instituto Politécnico Nacional

Escuela Superior de Cómputo

Ingeniería en Sistemas Computacionales

Máquina de Turing

Nombre: Espinosa de los Monteros Martínez Eric Omar

Materia: Teoria de la computación

Profesor: Juarez Martínez Genaro

Grupo: 4CM5

Índice

1. Objetivo

El objetivo de esta práctica es comprender mejor el funcionamiento de una máquina de Turing y mejorar las habilidades de programación.

2. Desarrollo

La implementación que se presenta a continuación es una aplicación de una Máquina de Turing en Python, utilizando la biblioteca Tkinter para la interfaz gráfica. La Máquina de Turing diseñada tiene como objetivo validar si una cadena pertenece al lenguaje

```
{0^n 1^n | n >= 1}
```

. Este lenguaje consiste en cadenas que contienen un número igual de ceros seguido por un número igual de unos.

La práctica se estructura alrededor de diversas funciones que definen la lógica de la máquina de Turing, su animación en la interfaz gráfica y la validación de cadenas proporcionadas manualmente o generadas aleatoriamente. La implementación ofrece una representación visual de la ejecución de la máquina de Turing, con detalles como la cinta, el estado presente y las transiciones entre estados.

A través de la interfaz gráfica, se brinda la posibilidad de ingresar cadenas manualmente o generar cadenas aleatorias para su validación. La máquina de Turing sigue un conjunto predeterminado de funciones de transición que determinan su comportamiento.

La práctica también incluye un archivo de salida ('salida.txt') que registra las instantáneas de la ejecución de la máquina de Turing, proporcionando una visión detallada de cada paso. Este archivo se actualiza a medida que se ejecuta la máquina de Turing. El programa se realizó según los criterios de Classroom.

2.1. Explicación de las funciones

Función 'agregar_caracter_posicion(cadena_modificar, caracter_a_agregar, posicion_cadena)": Agrega un caracter en una posición específica dentro de una cadena, tambien agrega el carcater -"que separa cada paso.

Función "siguiente_paso(estado_presente, cinta, posicion_cabeza)": Realiza un paso en la máquina de Turing según las funciones de transición definidas. Actualiza el estado presente, la cinta y la posición de la cabeza.

Función "validar_cadena(cinta, ventana_anterior)": Muestra una interfaz gráfica para la animación de la ejecución de la máquina de Turing. Utiliza la función siguiente_paso para ejecutar

cada paso y actualiza la interfaz gráfica correspondientemente. También se encarga de guardar las instantáneas en un archivo de texto.

Función "mostrar_ventana_ingresar_cadena()": Muestra una ventana para ingresar una cadena manualmente.

Función "generar_cadena_binaria_aleatoria()": Genera una cadena binaria aleatoria de longitud variable.

Diccionario "funciones_transicion": Define las funciones de transición para la máquina de Turing.

2.2. Código

```
import tkinter as tk
import os
import random
import time
from PIL import Image, ImageTk

def agregar_caracter_posicion(cadena_modificar, caracter_a_agregar,
    posicion_cadena):
    # Agregar el caracter en la posicion indicada
    cadena_modificar = cadena_modificar[:posicion_cadena] + "{" +
       caracter_a_agregar  + "}" + cadena_modificar[posicion_cadena
       :] + "␣+␣"
    return cadena_modificar

def siguiente_paso(estado_presente, cinta, posicion_cabeza):
    validacion_cadena = ''
    if (estado_presente, cinta[posicion_cabeza]) not in
       funciones_transicion:  # No hay una transicion para el
       estado presente y el simbolo leido en la cinta
        validacion_cadena = 'False'
    else:

        cinta_sin_espacios = cinta.strip()
        with open('salida.txt', 'a') as archivo:
```

```
            archivo.write(agregar_caracter_posicion(
                cinta_sin_espacios, estado_presente, posicion_cabeza
                ))
        informacion_transicion = funciones_transicion[(
            estado_presente, cinta[posicion_cabeza])]
        estado_presente = informacion_transicion[0] # Actualiza el
            estado presente
        cinta_lista = list(cinta) # Convierto str a un objeto
            mutable list
        cinta_lista[posicion_cabeza] = informacion_transicion[1] #
            Escribe el carcater en la posicion de la cabeza
        cinta = ''.join(cinta_lista) # convierto la lista a str
            nuevamente
        movimiento_cinta = informacion_transicion[2]

        if movimiento_cinta == 'r': # Se mueve la cabeza
            posicion_cabeza += 1
        else:
            posicion_cabeza -= 1
    return estado_presente, cinta, posicion_cabeza,
        validacion_cadena


def validar_cadena(cinta, ventana_anterior):
    cinta_sin_espacios = cinta
    ventana_anterior.withdraw() # Oculta la ventana anterior
    ventana_animacion = tk.Toplevel()  # Crea una nueva ventana
    ventana_animacion.title("Animacion")

    estado_presente = 'q0' # Se establece el estado presente como
        el estado incial 'q0'
    posicion_cabeza = 0

    if len(cinta_sin_espacios) > 16:
```

```
45          etiqueta_mensaje = tk.Label(ventana_animacion, text="No se
               puede animar, porque la cadena es mayor a 16 caracteres\
               n La salida del programa se encuentra en el archivo '
               salida_programa'", font=("Arial", 16))
46          etiqueta_mensaje.grid(row=0, column=0, columnspan=20)
47      else:
48          # Acompleta la cinta con espacios en blanco hasta que la
               cinta tenga 1006 caracteres en total
49          cinta = cinta.ljust(1006)
50
51          etiqueta_cinta = tk.Label(ventana_animacion, text="Cinta",
               font=("Arial", 16))
52          etiqueta_cinta.grid(row=1, column=7, columnspan=2)
53
54          # Crear canvas para la cinta
55          cinta1 = tk.Canvas(ventana_animacion, width=30, height=30,
               borderwidth=2, relief="solid")
56          cinta1.grid(row=2, column=0)
57          cinta2 = tk.Canvas(ventana_animacion, width=30, height=30,
               borderwidth=2, relief="solid")
58          cinta2.grid(row=2, column=1)
59          cinta3 = tk.Canvas(ventana_animacion, width=30, height=30,
               borderwidth=2, relief="solid")
60          cinta3.grid(row=2, column=2)
61          cinta4 = tk.Canvas(ventana_animacion, width=30, height=30,
               borderwidth=2, relief="solid")
62          cinta4.grid(row=2, column=3)
63          cinta5 = tk.Canvas(ventana_animacion, width=30, height=30,
               borderwidth=2, relief="solid")
64          cinta5.grid(row=2, column=4)
65          cinta6 = tk.Canvas(ventana_animacion, width=30, height=30,
               borderwidth=2, relief="solid")
66          cinta6.grid(row=2, column=5)
67          cinta7 = tk.Canvas(ventana_animacion, width=30, height=30,
               borderwidth=2, relief="solid")
```

```
68          cinta7.grid(row=2, column=6)
69          cinta8 = tk.Canvas(ventana_animacion, width=30, height=30,
               borderwidth=2, relief="solid")
70          cinta8.grid(row=2, column=7)
71          cinta9 = tk.Canvas(ventana_animacion, width=30, height=30,
               borderwidth=2, relief="solid")
72          cinta9.grid(row=2, column=8)
73          cinta10 = tk.Canvas(ventana_animacion, width=30, height=30,
                borderwidth=2, relief="solid")
74          cinta10.grid(row=2, column=9)
75          cinta11 = tk.Canvas(ventana_animacion, width=30, height=30,
                borderwidth=2, relief="solid")
76          cinta11.grid(row=2, column=10)
77          cinta12 = tk.Canvas(ventana_animacion, width=30, height=30,
                borderwidth=2, relief="solid")
78          cinta12.grid(row=2, column=11)
79          cinta13 = tk.Canvas(ventana_animacion, width=30, height=30,
                borderwidth=2, relief="solid")
80          cinta13.grid(row=2, column=12)
81          cinta14 = tk.Canvas(ventana_animacion, width=30, height=30,
                borderwidth=2, relief="solid")
82          cinta14.grid(row=2, column=13)
83          cinta15 = tk.Canvas(ventana_animacion, width=30, height=30,
                borderwidth=2, relief="solid")
84          cinta15.grid(row=2, column=14)
85          cinta16 = tk.Canvas(ventana_animacion, width=30, height=30,
                borderwidth=2, relief="solid")
86          cinta16.grid(row=2, column=15)
87
88          # Agregar el texto de la cinta al canvas
89          texto_cinta1 = cinta1.create_text(19, 19, text=cinta[0],
               font=("Arial", 12), fill="black")
90          texto_cinta2 = cinta2.create_text(19, 19, text=cinta[1],
               font=("Arial", 12), fill="black")
```

```
91          texto_cinta3 = cinta3.create_text(19, 19, text=cinta[2],
               font=("Arial", 12), fill="black")
92          texto_cinta4 = cinta4.create_text(19, 19, text=cinta[3],
               font=("Arial", 12), fill="black")
93          texto_cinta5 = cinta5.create_text(19, 19, text=cinta[4],
               font=("Arial", 12), fill="black")
94          texto_cinta6 = cinta6.create_text(19, 19, text=cinta[5],
               font=("Arial", 12), fill="black")
95          texto_cinta7 = cinta7.create_text(19, 19, text=cinta[6],
               font=("Arial", 12), fill="black")
96          texto_cinta8 = cinta8.create_text(19, 19, text=cinta[7],
               font=("Arial", 12), fill="black")
97          texto_cinta9 = cinta9.create_text(19, 19, text=cinta[8],
               font=("Arial", 12), fill="black")
98          texto_cinta10 = cinta10.create_text(19, 19, text=cinta[9],
               font=("Arial", 12), fill="black")
99          texto_cinta11 = cinta11.create_text(19, 19, text=cinta[10],
                font=("Arial", 12), fill="black")
100         texto_cinta12 = cinta12.create_text(19, 19, text=cinta[11],
                font=("Arial", 12), fill="black")
101         texto_cinta13 = cinta13.create_text(19, 19, text=cinta[12],
                font=("Arial", 12), fill="black")
102         texto_cinta14 = cinta14.create_text(19, 19, text=cinta[13],
                font=("Arial", 12), fill="black")
103         texto_cinta15 = cinta15.create_text(19, 19, text=cinta[14],
                font=("Arial", 12), fill="black")
104         texto_cinta16 = cinta16.create_text(19, 19, text=cinta[15],
                font=("Arial", 12), fill="black")
105
106         # Cargar la imagen de la flecha hacia arriba
107         imagen_flecha = Image.open("flecha_arriba.png")
108         imagen_flecha = imagen_flecha.resize((30, 30))
109         imagen_flecha = ImageTk.PhotoImage(imagen_flecha)
110
111         eflecha = tk.Label(ventana_animacion, image=imagen_flecha)
```

```
112         eflecha.image = imagen_flecha  # Evitar que la imagen sea
                eliminada por la recoleccion de basura
113         eflecha.grid(row=3, column=0)
114 
115         etiqueta_estado_presente = tk.Label(ventana_animacion, text
                =f"Estado␣presente:␣{estado_presente}", font=("Arial",
                16))
116         etiqueta_estado_presente.grid(row=4, column=5, columnspan
                =6)
117 
118         etiqueta_estado_final = tk.Label(ventana_animacion, text="
                Estado␣final:␣q4", font=("Arial", 16))
119         etiqueta_estado_final.grid(row=5, column=5, columnspan=6)
120 
121         string_funciones_transicion = str(funciones_transicion)
122         aux_recortar = string_funciones_transicion[:31] + '\n'
123         funciones_transicion_lineas = aux_recortar
124         # Siguiente transicion
125         aux_recortar = string_funciones_transicion[31:]
126         aux_recortar = aux_recortar[:31] + '\n'
127         funciones_transicion_lineas = funciones_transicion_lineas +
                aux_recortar
128         # Siguiente transicion
129         aux_recortar = string_funciones_transicion[62:]
130         aux_recortar = aux_recortar[:31] + '\n'
131         funciones_transicion_lineas = funciones_transicion_lineas +
                aux_recortar
132         # Siguiente transicion
133         aux_recortar = string_funciones_transicion[93:]
134         aux_recortar = aux_recortar[:31] + '\n'
135         funciones_transicion_lineas = funciones_transicion_lineas +
                aux_recortar
136         # Siguiente transicion
137         aux_recortar = string_funciones_transicion[124:]
138         aux_recortar = aux_recortar[:31] + '\n'
```

```
139         funciones_transicion_lineas = funciones_transicion_lineas +
                aux_recortar
140         # Siguiente transicion
141         aux_recortar = string_funciones_transicion[155:]
142         aux_recortar = aux_recortar[:31] + '\n'
143         funciones_transicion_lineas = funciones_transicion_lineas +
                aux_recortar
144         # Siguiente transicion
145         aux_recortar = string_funciones_transicion[186:]
146         aux_recortar = aux_recortar[:31] + '\n'
147         funciones_transicion_lineas = funciones_transicion_lineas +
                aux_recortar
148         # Siguiente transicion
149         aux_recortar = string_funciones_transicion[217:]
150         aux_recortar = aux_recortar[:31] + '\n'
151         funciones_transicion_lineas = funciones_transicion_lineas +
                aux_recortar
152         # Siguiente transicion
153         aux_recortar = string_funciones_transicion[248:]
154         aux_recortar = aux_recortar[:31] + '\n'
155         funciones_transicion_lineas = funciones_transicion_lineas +
                aux_recortar
156         # Siguiente transicion
157         aux_recortar = string_funciones_transicion[279:]
158         aux_recortar = aux_recortar[:31] + '\n'
159         funciones_transicion_lineas = funciones_transicion_lineas +
                aux_recortar
160
161         etiqueta_funciones_transicion = tk.Label(ventana_animacion,
                text=f"Funciones␣transicion:␣\n{
              funciones_transicion_lineas}", font=("Arial", 16))
162         etiqueta_funciones_transicion.grid(row=6, column=5,
              columnspan=6)
163
```

```
    while estado_presente != 'q4' and cinta[posicion_cabeza] != '␣':
        estado_presente, cinta, posicion_cabeza, validacion_cadena = siguiente_paso(estado_presente, cinta, posicion_cabeza) # Ejecuto un paso en la maquina de Turing
        if validacion_cadena == 'False':
            break
        if len(cinta_sin_espacios) > 16:
            etiqueta_mensaje = tk.Label(ventana_animacion, text="No␣se␣puede␣animar,␣porque␣la␣cadena␣es␣mayor␣a␣16␣caracteres\n␣La␣salida␣del␣programa␣se␣encuentra␣en␣el␣archivo␣'salida_programa'", font=("Arial", 16))
            etiqueta_mensaje.grid(row=0, column=0, columnspan=20)
        else:
            # Actualizar la animacion
            ventana_animacion.update()
            if len(cinta_sin_espacios) < 16:
                time.sleep(velocidad_animacion)
            cinta1.itemconfig(texto_cinta1, text=cinta[0])
            cinta2.itemconfig(texto_cinta1, text=cinta[1])
            cinta3.itemconfig(texto_cinta1, text=cinta[2])
            cinta4.itemconfig(texto_cinta1, text=cinta[3])
            cinta5.itemconfig(texto_cinta1, text=cinta[4])
            cinta6.itemconfig(texto_cinta1, text=cinta[5])
            cinta7.itemconfig(texto_cinta1, text=cinta[6])
            cinta8.itemconfig(texto_cinta1, text=cinta[7])
            cinta9.itemconfig(texto_cinta1, text=cinta[8])
            cinta10.itemconfig(texto_cinta1, text=cinta[9])
            cinta11.itemconfig(texto_cinta1, text=cinta[10])
            cinta12.itemconfig(texto_cinta1, text=cinta[11])
            cinta13.itemconfig(texto_cinta1, text=cinta[12])
            cinta14.itemconfig(texto_cinta1, text=cinta[13])
            cinta15.itemconfig(texto_cinta1, text=cinta[14])
            cinta16.itemconfig(texto_cinta1, text=cinta[15])
```

```
192             etiqueta_estado_presente.config(text=f'Estado␣presente:
                    ␣{estado_presente}')
193             eflecha.grid(row=3, column=posicion_cabeza)
194
195     estado_presente, cinta, posicion_cabeza, validacion_cadena =
            siguiente_paso(estado_presente, cinta, posicion_cabeza) #
            Ejecuto un paso en la maquina de Turing
196     cinta_aux_sin_espacios = cinta.strip()
197     cinta_aux_sin_espacios = agregar_caracter_posicion(
            cinta_aux_sin_espacios, estado_presente, posicion_cabeza)
198     cinta_aux_sin_espacios = cinta_aux_sin_espacios[:len(
            cinta_aux_sin_espacios) - 3]
199
200     if estado_presente == 'q4' and cinta[posicion_cabeza] == '␣':
201         validacion_cadena = 'True'
202     if validacion_cadena == 'False':
203         if len(cinta_sin_espacios) > 16:
204             etiqueta_mensaje = tk.Label(ventana_animacion, text="No
                    ␣se␣puede␣animar,␣porque␣la␣cadena␣es␣mayor␣a␣16␣
                    caracteres\n␣La␣salida␣del␣programa␣se␣encuentra␣en␣
                    el␣archivo␣'salida_programa'", font=("Arial", 16))
205             etiqueta_mensaje.grid(row=0, column=0, columnspan=20)
206         else:
207             etiqueta_aceptacion = tk.Label(ventana_animacion, text=
                    "La␣cadena␣no␣pertenece␣al␣lenguaje␣{0^n1^n␣|␣n>=␣1}
                    ", font=("Arial", 16), foreground="red")
208             etiqueta_aceptacion.grid(row=7, column=1, columnspan
                    =13)
209     elif validacion_cadena == 'True':
210         if len(cinta_sin_espacios) > 16:
211             etiqueta_mensaje = tk.Label(ventana_animacion, text="No
                    ␣se␣puede␣animar,␣porque␣la␣cadena␣es␣mayor␣a␣16␣
                    caracteres\n␣La␣salida␣del␣programa␣se␣encuentra␣en␣
                    el␣archivo␣'salida_programa'", font=("Arial", 16))
212             etiqueta_mensaje.grid(row=0, column=0, columnspan=20)
```

```
213             else:
214                 etiqueta_aceptacion = tk.Label(ventana_animacion, text=
                        "La cadena pertenece al lenguaje {0^n1^n | n>= 1}",
                        font=("Arial", 16), foreground="green")
215                 etiqueta_aceptacion.grid(row=7, column=1, columnspan
                        =13)
216
217         with open('salida.txt', 'a') as archivo:
218             archivo.write(cinta_aux_sin_espacios)
219             if validacion_cadena == 'True':
220                 archivo.write(f'\n\nLa cadena es reconocida por la
                        maquina de Turing')
221             if validacion_cadena == 'False':
222                 archivo.write(f'\n\nLa cadena no es reconocida por la
                        maquina de Turing')
223
224         if len(cinta_sin_espacios) > 16:
225             etiqueta_mensaje = tk.Label(ventana_animacion, text="No se
                    puede animar, porque la cadena es mayor a 16 caracteres\
                    n La salida del programa se encuentra en el archivo '
                    salida_programa'", font=("Arial", 16))
226             etiqueta_mensaje.grid(row=0, column=0, columnspan=20)
227         else:
228             # Actualizar la animacion
229             ventana_animacion.update()
230             if len(cinta_sin_espacios) < 16:
231                 time.sleep(velocidad_animacion)
232             cinta1.itemconfig(texto_cinta1, text=cinta[0])
233             cinta2.itemconfig(texto_cinta1, text=cinta[1])
234             cinta3.itemconfig(texto_cinta1, text=cinta[2])
235             cinta4.itemconfig(texto_cinta1, text=cinta[3])
236             cinta5.itemconfig(texto_cinta1, text=cinta[4])
237             cinta6.itemconfig(texto_cinta1, text=cinta[5])
238             cinta7.itemconfig(texto_cinta1, text=cinta[6])
239             cinta8.itemconfig(texto_cinta1, text=cinta[7])
```

```
240             cinta9.itemconfig(texto_cinta1, text=cinta[8])
241             cinta10.itemconfig(texto_cinta1, text=cinta[9])
242             cinta11.itemconfig(texto_cinta1, text=cinta[10])
243             cinta12.itemconfig(texto_cinta1, text=cinta[11])
244             cinta13.itemconfig(texto_cinta1, text=cinta[12])
245             cinta14.itemconfig(texto_cinta1, text=cinta[13])
246             cinta15.itemconfig(texto_cinta1, text=cinta[14])
247             cinta16.itemconfig(texto_cinta1, text=cinta[15])
248             etiqueta_estado_presente.config(text=f'Estado presente: {
                    estado_presente}')
249             eflecha.grid(row=3, column=posicion_cabeza)
250
251         ventana_animacion.protocol("WM_DELETE_WINDOW", lambda: [ventana
                .destroy()])
252
253     def mostrar_ventana_ingresar_cadena():
254         ventana.withdraw()  # Oculta la ventana principal
255         ventana_cadena = tk.Toplevel()  # Crea una nueva ventana
256         ventana_cadena.title("Ingresar cadena")
257
258         etiqueta_cadena = tk.Label(ventana_cadena, text="Ingrese la
                cadena", font=("Arial", 16))
259         etiqueta_cadena.pack(padx=20, pady=20)
260
261         cinta = tk.Entry(ventana_cadena, font=("Arial", 12))
262         cinta.pack(pady=10)
263
264         boton_validar = tk.Button(ventana_cadena, text="Validar", width
                =20, height=2,
265                                   bg="#CAFFBF", font=("Arial", 12),
266                                   command=lambda: validar_cadena(cinta.
                                        get(), ventana_cadena))
267         boton_validar.pack(pady=10)
268
```

```
269     ventana_cadena.protocol("WM_DELETE_WINDOW", lambda: [ventana.
            destroy()])
270 
271 def generar_cadena_binaria_aleatoria():
272     longitud = random.randint(1, 1000)
273     cadena_binaria = ''.join(random.choice('01') for _ in range(
            longitud))
274     return cadena_binaria
275 
276 with open('salida.txt', 'w') as archivo:
277             archivo.write('Instantaneas\n')
278 
279 # Crear la ventana principal
280 ventana = tk.Tk()
281 ventana.title("Automata␣Pila")
282 
283 # Contenedor principal
284 contenedor_principal = tk.Frame(ventana)
285 contenedor_principal.grid(column=0, row=0, padx=(50,50), pady
    =(10,10))
286 
287 # Etiqueta de inicio
288 etiqueta = tk.Label(contenedor_principal, text="Validar␣si␣la␣
    maquina␣de␣Turing␣reconoce␣el␣lenguaje␣{0^n␣1^n␣|␣n␣>=␣1}", font
    =("Arial", 16))
289 etiqueta.grid(row=1, column=0, columnspan=2)
290 
291 boton_ingresar_cadena = tk.Button(contenedor_principal, text="
    Ingresar␣cadena", width=20, height=5,\
292     bg="#CAFFBF", font=("Arial", 12), command=
            mostrar_ventana_ingresar_cadena)
293 boton_ingresar_cadena.grid(row=3, column=0)
294 
295 
296 
```

```
297 boton_cadena_aleatoria = tk.Button(contenedor_principal, text="
    Cadena␣aleatoria", width=20, height=5,\
298     bg="#CAFFBF", font=("Arial", 12), command=lambda:
        validar_cadena(generar_cadena_binaria_aleatoria(), ventana))
299 boton_cadena_aleatoria.grid(row=3, column=1)
300
301 velocidad_animacion = 1
302
303 # Estructura de funciones_transicion
304 # ('estado_presente', 'simbolo_leido'): ('estado_siguiente', '
    simbolo_a_escribir', 'direcicion_mover_cabeza')
305 #


    r=rigth, l=left
306 funciones_transicion = {
307     ('q0', '0'): ('q1', 'x', 'r'),
308     ('q0', 'y'): ('q3', 'y', 'r'),
309     ('q1', '0'): ('q1', '0', 'r'),
310     ('q1', 'y'): ('q1', 'y', 'r'),
311     ('q1', '1'): ('q2', 'y', 'l'),
312     ('q2', '0'): ('q2', '0', 'l'),
313     ('q2', 'y'): ('q2', 'y', 'l'),
314     ('q2', 'x'): ('q0', 'x', 'r'),
315     ('q3', 'y'): ('q3', 'y', 'r'),
316     ('q3', '␣'): ('q4', '␣', 'r')
317 }
318
319 # Iniciar el bucle de eventos
320 ventana.mainloop()
```

2.3. Funcionamiento del programa

El programa comienza con una ventana donde hay dos botones, uno que nos permite ingresar una cadena manualmente o generar una cadena binaria aleatoria.

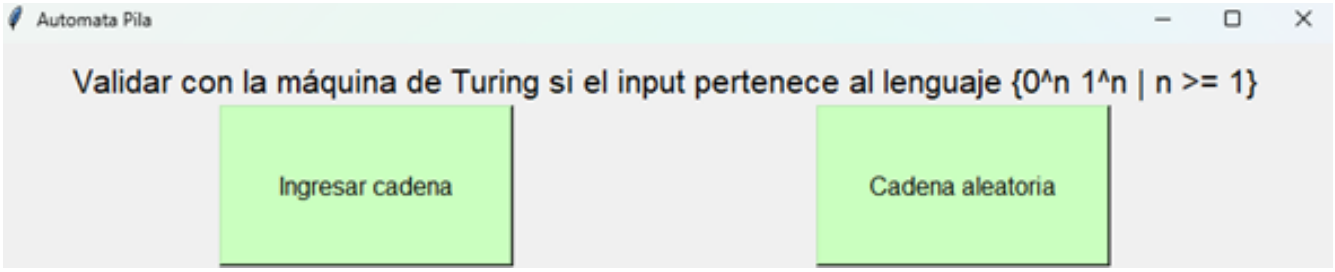

Figura 1: Ventana de inicio

Después si se selecciona el botón 'Ingresar cadena' se muestra una nueva ventana donde se ingresará una cadena como input para después al presionar el botón validar y ejecutar la animación de la máquina de Turing solo si la cadena es menor a 16 caracteres, en otro caso, la animación no se ejecutará y solamente se guardaran las instantáneas en el archivo de texto "salida.txt".

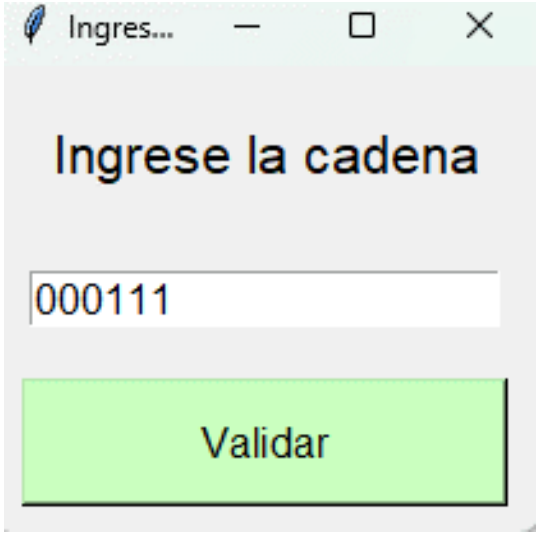

Figura 2: Ventana ingresar cadena

La animación mostrará la cinta donde la cabeza leerá el input e ira escribiendo sobre la ella, además se verá el estado presente y las funciones de transición. Finalmente si el input fue aceptado se mostrará un mensaje en verde que lo indicará y en caso contrario se mostrará un mensaje en rojo.

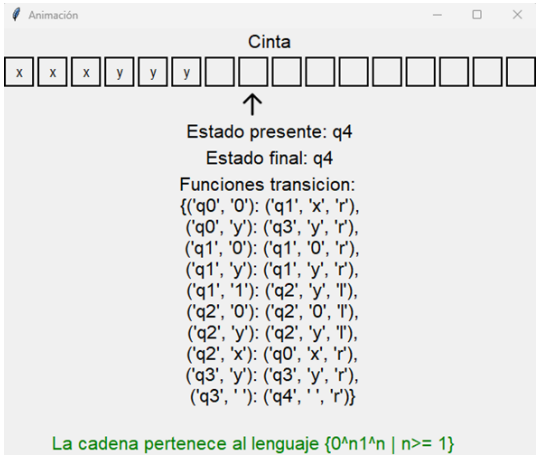

Figura 3: Ventana de animacón cuando es aceptado el input

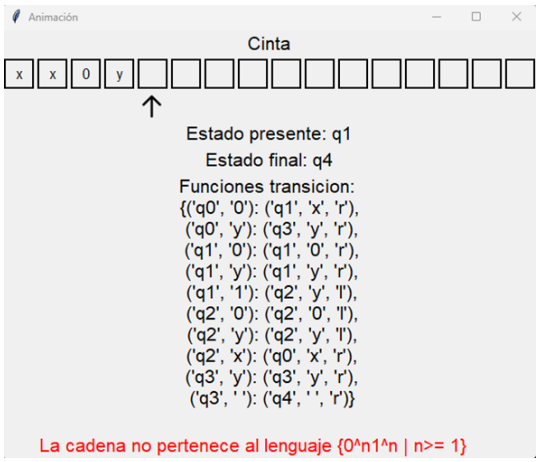

Figura 4: Ventana de animacón cuando no es aceptado el input

Finalmente, las instantáneas serán guardadas en el archivo 'salida.txt'.

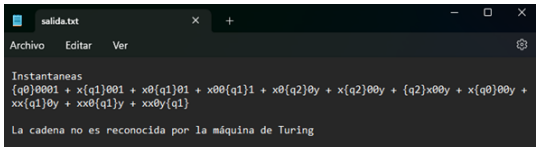

Figura 5: Ventana de animacón cuando no es aceptado el input

3. Conclusiones

La máquina de Turing, concebida por Alan Turing en 1936, representa un hito crucial en la teoría de la computación. Al proporcionar un modelo abstracto de una máquina capaz de simular cualquier algoritmo, la máquina de Turing se convierte en la base teórica para entender la computabilidad y la complejidad algorítmica. Su universalidad y simplicidad conceptual han permitido establecer límites fundamentales sobre lo que es computable y han influido profundamente en el desarrollo de la informática, brindando una herramienta esencial para analizar y comprender la naturaleza y las limitaciones de los procesos computacionales.