Práctica 6. "Pila"

Carmona Serrano Ian Carlo
Ingeniería en Inteligencia Artificial 5BM1
Teoría de la Computación , ESCOM- IPN

6 de Enero 2023

1 Introduction

Este practica presenta la implementación de un autómata de pila para reconocer cadenas con una cantidad igual de ceros seguida de una cantidad correspondiente de unos. El programa permite ingresar la cadena manual o automáticamente, generando cadenas aleatorias de hasta 100,000 caracteres. Evalúa el autómata, registrando y mostrando descripciones instantáneas en pantalla y en un archivo. Además, ofrece una animación gráfica del autómata para cadenas de hasta 10 caracteres.

Marco Teórico

Autómata de Pila

Un autómata de pila es una máquina que extiende la capacidad de un autómata finito al incluir una pila. Puede leer y escribir información en una pila, permitiendo reconocer lenguajes más complejos que los aceptados por autómatas finitos.

Lenguaje Libre de Contexto (LLC)

Los lenguajes libres de contexto son aquellos que pueden ser generados por una gramática libre de contexto. Estos lenguajes son reconocidos por autómatas de pila y pueden representar estructuras anidadas y recursivas.

Lenguaje $\{0^n\ 1^n$ — n 1$\}$

Este lenguaje consiste en cadenas con una cantidad igual de ceros seguida por una cantidad correspondiente de unos. Es un ejemplo clásico de un lenguaje no regular que puede ser reconocido por un autómata de pila.

Descripciones Instantáneas (IDs)

Son representaciones de la configuración de un autómata de pila en un punto específico de la ejecución, mostrando el estado, la cadena restante por leer y el contenido de la pila.

Desarrollo

El autómata de pila es una extensión de un autómata finito que incluye una pila como un elemento adicional de memoria. La pila permite al autómata procesar lenguajes que no son regulares, como el lenguaje $\{0^n\ 1^n$ — n $1\}$.

Funcionamiento del Autómata

Para comprender el funcionamiento del autómata de pila en el reconocimiento del lenguaje $\{0^n\ 1^n$ — n $1\}$, se describe su ejecución:

1. El autómata inicia en un estado inicial con la pila conteniendo el símbolo especial de fondo de pila $Z0$.
2. Lee la entrada de la cadena.
3. Por cada cero leído, se agrega un símbolo X a la pila.
4. Por cada uno leído, se elimina un símbolo X de la pila.
5. Al final de la cadena, se verifica que la pila esté vacía para determinar si la cadena es aceptada o no.

Implementación

La implementación se realizó en Python utilizando clases para el autómata de pila y funciones para evaluar la cadena, generar descripciones instantáneas y, en el caso de cadenas cortas, animar el funcionamiento del autómata.

Pantallas del Programa en Ejecución

```
Elija el modo:
1. Automatico
2. Manual
Inserte su opcion deseada: 2
Inserte su cadena: 00001111
```

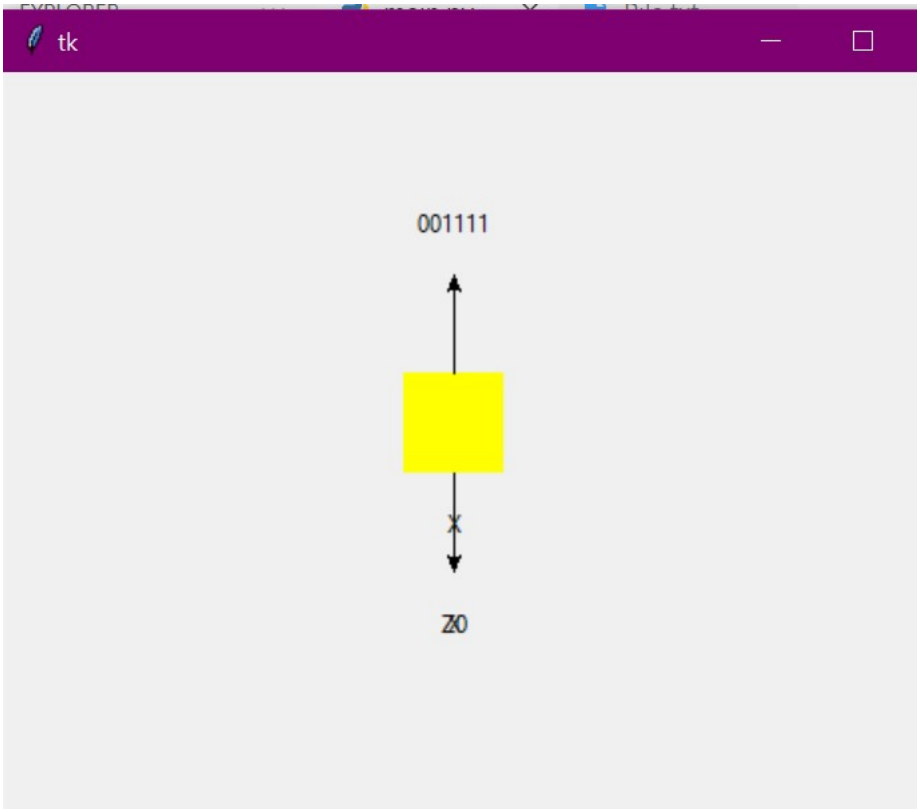

Figure 1: Animación

```
Elija el modo:
1. Automatico
2. Manual
Inserte su opcion deseada: 2
Inserte su cadena: 00001111
Su entrada fue:  00001111
Tope:  Z0
Cadena aceptada
```

```
00001111
d(q, 0, Z0) = [(q, XZ0)]
0001111
d(q, 0, X) = [(q, XXZ0)]
001111
d(q, 0, X) = [(q, XXXZ0)]
01111
d(q, 0, X) = [(q, XXXXZ0)]
1111
d(q, 1, XXXX) = [(p, XXX)]
d(p, 1, XXXX) = [(p, XXX)]
111
d(p, 1, XXX) = [(p, XX)]
11
d(p, 1, XX) = [(p, X)]
1
d(p, 1, X) = [(p, e)]
d(p, e, Z0) = [(f, Z0)]
```

Figure 2: txt

Código

```
1  import random
2  import time
3  from tkinter import Canvas, Tk
4
5  class Pila:
6      def __init__(self):
7          self.items = []
8
9      def esta_vacia(self):
10         return self.items == ['Z0']
11
12     def apilar(self, elemento):
13         self.items = self.items + [elemento]
14
15     def desapilar(self):
16
```

```
        self.items = self.items[:-1]
        return self.items

    def ver_tope(self):
        return self.items[len(self.items)-1]

    def tamano(self):
        return len(self.items)


def esta_valida(pila: Pila, cadena):

    x = "X"
    bandera = True

    for i in range(len(cadena)):
        bit = cadena[i]

        if bit == "0":
            if i == 0:
                f.write(f"(q, 0, Z0) = [(q, {x}Z0)]")
            else:
                x += 'X'
                f.write(f"(q, 0, X) = [(q, {x}Z0)]")

            pila.apilar('X')
        else:
            if bandera == True:
                f.write(f"(q, 1, {x}) = [(p, {x[:-1]})]\n")
                bandera = False

            if x[:-1] != '':
                f.write(f"(p, 1, {x}) = [(p, {x[:-1]})]")
            else:
                f.write(f"(p, 1, {x}) = [(p, e)]")

            x = x[:-1]

            if pila.esta_vacia():
                return False

            pila.desapilar()

        f.write("\n")


    if pila.ver_tope() != 'Z0':
        return False

    return True
```

```
67
68
69  def automataPila_Grafica(pila, entrada):
70      i = 0
71      aux = entrada
72
73      canv.create_rectangle(200, 150, 250, 200, width=0, fill=
        'yellow')
74      canv.create_line(225, 150, 225, 100, arrow="last")
75      canv.create_line(225, 200, 225, 250, arrow="last")
76      canv.create_text(225, 175, text=entrada, tags="entrada1"
        )
77      canv.create_text(225, 275, text="Z0", tags="pila{i}")
78      canv.pack()
79
80      x = 225
81      y = 225
82      x2 = 225
83      y2 = 75
84
85      canv.update()
86      time.sleep(2)
87
88      equis = "X"
89      bandera = True
90      for j in range(len(entrada)):
91          bit = entrada[j]
92          f.write(aux + "\n")
93          canv.delete("entrada1")
94
95          if bit == '0':
96              if j == 0:
97                  f.write(f"d(q, 0, Z0) = [(q, {equis}Z0)]")
98              else:
99                  equis += 'X'
100                 f.write(f"d(q, 0, X) = [(q, {equis}Z0)]")
101             canv.create_text(x, y, text="X", tags=f"pila{i}"
        )
102             pila.apilar('X')
103             i += 1
104             y += 50
105
106         else:
107             if bandera == True:
108                 f.write(f"d(q, 1, {equis}) = [(p, {equis
        [:-1]})]\n")
109                 bandera = False
110
111             if equis[:-1] != '':
```

```
                f.write(f"d(p, 1, {equis}) = [(p, {equis[:-1]})]")
            else:
                f.write(f"d(p, 1, {equis}) = [(p, e)]")

            equis = equis[:-1]

            y -= 50
            if (pila.esta_vacia()):
                return False
            i -= 1
            canv.delete(f"pila{i}")
            pila.desapilar()

        canv.delete("entrada")
        aux = entrada[j + 1:]
        canv.create_text(x2, y2, text=str(aux), tags="entrada")
        canv.update()

        time.sleep(1)
        f.write("\n")

    canv.update()
    if pila.ver_tope() != 'Z0':
        return False

    f.write("d(p, e, Z0) = [(f, Z0)]")
    return True


print("Elija el modo:")
print("1. Automatico")
print("2. Manual")
opc = input("Inserte su opcion deseada: ")
entrada = ""

if opc == '1':
    cantidadCeros = random.randint(1, 500)
    cantidadUnos = random.randint(1, 500)
    print("Cantidad ceros: ", cantidadCeros)
    print("Cantidad unos: ", cantidadUnos)

    for i in range(cantidadCeros):
        entrada += '0'
    for i in range(cantidadUnos):
        entrada += '1'
else:
    entrada = input("Inserte su cadena: ")

```

```
print("Su entrada fue: ", entrada)

pila = Pila()
pila.apilar('Z0')

f = open("Pila.txt", 'w')

esValida = False
if len(entrada) <= 10:
    ventana = Tk()
    canv = Canvas(ventana, width=500, height=500)
    ventana.geometry("500x500")
    esValida = automataPila_Grafica(pila, entrada)
    canv.update()
    canv.place(x=0, y=0)
    ventana.mainloop()
else:
    esValida = esta_valida(pila, entrada)

if esValida:
    print("Tope: ", pila.ver_tope())
    print("Cadena aceptada")
else:
    print("Tope: ", pila.ver_tope())
    print("No es valida")
```

Listing 1: Practica 6

2 Conclusión

La implementación y visualización del autómata de pila para el lenguaje $0^n1^n|n \geq 1$ ha demostrado la utilidad de las pilas en la validación de estructuras de cadenas. Esta práctica ha facilitado la comprensión práctica de los conceptos teóricos de los autómatas de pila y su aplicación en la verificación de lenguajes formales.

3 Bibliografía

Ullman, J.D. (2009-10). "CS154: Introduction to Automata and Complexity Theory". Sitio web: http://infolab.stanford.edu/ ullman/ialc/spr10/spr10.htmlLECTURE