INSTITUTO POLITECNICO NACIONAL

Escuela Superior de Cómputo

PRACTICA 6

Autor:
Gomez Hernandez Alan Javier

26 diciembre, 2021—

AUTÓMATA DE PILA

RESUMEN

Programar un autómata de pila que sirva para reconocer el lenguaje libre de contexto $0^n1^n|n >= 1$.

- **La cadena puede ser ingresada por el usuario o automáticamente. Si es aleatoriamente, la cadena no podrá ser mayor a 100,000 caracteres.**
- **Mandar a un archivo y en pantalla la evaluación del autómata a través de descripciones instantáneas (IDs).**
- **Animar el autómata de pila, solo si la cadena es menor igual a 10 caracteres.**

INTRODUCCION

A través de Microsoft Store utiliza el intérprete básico de Python3, pero maneja la configuración de su PATH para el usuario actual (evitando la necesidad de acceso de administrador), además de proporcionar actualizaciones automáticas. Es posible ejecutar Python de diferentes maneras, por lo que cada usuario deberá realizarlo según las necesidades que posea. Sin embargo, es importante destacar que la mayoría de los sistemas de Python ya vienen instalados, por lo que no requiere de acción alguna. Es posible ejecutar Python desde la terminal o línea de comando IDE o bien emplear opciones en la nube que incluya: Cuadernos Jupyter Google Colab Los autómatas de pila, en forma similar a como se usan los autómatas finitos, también se pueden utilizar para aceptar cadenas de un lenguaje definido sobre un alfabeto A. Los autómatas de pila pueden aceptar lenguajes que no pueden aceptar los autómatas finitos. Un autómata de pila cuenta con una cinta de entrada y un mecanismo de control que puede encontrarse en uno de entre un número finito de estados. Uno de estos estados se designa como estado inicial, y además algunos estados se llaman de aceptación o finales. A diferencia de los autómatas finitos, los autómatas de pila cuentan con una memoria auxiliar llamada pila. Los símbolos (llamados símbolos de pila) pueden ser insertados o extraídos de la pila, de acuerdo con el manejo last-in-first-out (LIFO). Las transiciones entre los estados que ejecutan los autómatas de pila dependen de los símbolos de entrada y de los símbolos de la pila. El autómata acepta una cadena x si la secuencia de transiciones, comenzando en estado inicial y con pila vacía, conduce a un estado final, después de leer toda la cadena x.

En este programa se desarrollo un autómata de pila (vease figura) que aceptara el lenguaje de $0^n1^n|n >= 1$. El programa tiene un modo manual y autom´atico en ambos modos se evalúa una cadena de ceros y unos de longitud

$1 \leq n \leq 1000$ y se muestra si la cadena es valida o no junto con los pasos para llegar a este resultado que se mostraran en archivo y consola. Además, se puede observar la animación de este proceso como el de la figura si así se desea, esta acción remplaza a la historia del autómata en consola pero no en archivo.

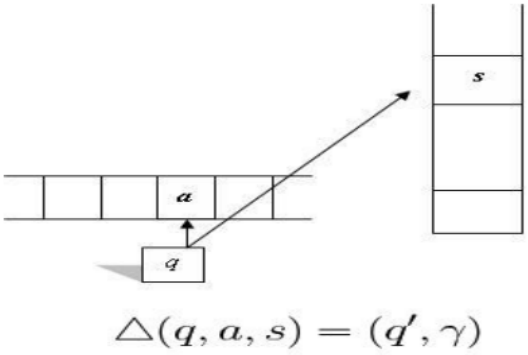

Figure 1: Representación de un autómata de Pila.

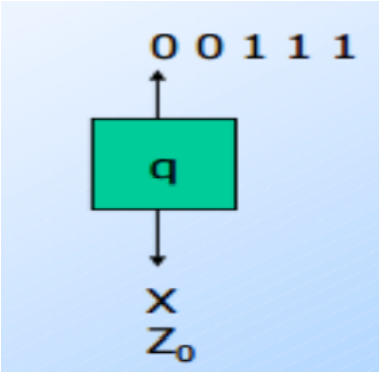

Figure 2: Representación de un autómata de Pila.

DESARROLLO

0.1 PILA (CODIGO)

```
# -*- coding: utf-8 -*-
from __future__ import print_function
import time

class Pila(object):
    def __init__(self):
        self.altura = -1
        self.elementos = []

    def vacio(self):
        if self.altura == -1:
            return True
        else:
            return False

    def sacar(self):
        if self.vacio():
            return 'e'
        else:
            valor = self.elementos[self.altura]
            self.altura -= 1
            return valor

    def meter(self, elemento):
        self.altura += 1
        self.elementos[self.altura:] = [elemento]

    def mostrar(self):
        i = self.altura
        cadena = ''
        while(i>-1):
            cadena += self.elementos[i]
            i -= 1
        return cadena

def automata(cadena, ver_animacion):
    pila = Pila()
    archivo = open('historia-pila.txt', 'w')
    pila.meter('Zo')
    estado = 'q'
    cadena_aux = cadena
    cadena = cadena + ' '
```

```
archivo.write('La cadena es: ' + cadena_aux + '\n')
for simbolo in cadena:
    if cadena_aux == '':
        cadena_aux = 'e'
    if ver_animacion:
        time.sleep(1)
        pintar(estado, cadena_aux, pila)
    else:
        print('(%s, %s, %s)' %(estado, cadena_aux, pila.mostrar()), end='')
    archivo.write('(%s, %s, %s)' %(estado, cadena_aux, pila.mostrar()))
    if estado == 'q':
        if simbolo == '0':
            pila.meter('X')
        elif simbolo == '1':
            if pila.sacar() == 'Zo':
                pila.meter('Zo')
                break
            estado = 'p'
        else:
            estado='q'
            break
    elif estado == 'p':
        if simbolo == '1':
            if pila.sacar() == 'Zo':
                estado = 'f'
                pila.meter('Zo')
                break
        elif simbolo == '0':
            pila.meter('X')
            cadena_aux = cadena_aux[1:]
            break
        elif simbolo == ' ':
            estado = 'f'
        else:
            break
    cadena_aux = cadena_aux[1:]
    print('|-', end='')
    archivo.write('|-')

if cadena_aux == '':
    cadena_aux = 'e'
if (pila.mostrar() == 'Zo') and cadena_aux == 'e' and estado=='f':
    if ver_animacion:
        time.sleep(1)
        pintar(estado, cadena_aux, pila)
    else:
```

```
            print('(%s, %s, %s)' %(estado, cadena_aux, pila.mostrar()))
            print('\n')
        archivo.write('(%s, %s, %s)' %(estado, cadena_aux, pila.mostrar()))
        print('\nCadena valida')
        archivo.write('\nCadena valida')
    else:
        print('\nCadena invalida')
        archivo.write('\nCadena invalida')
    archivo.close()

def pintar(estado, cadena_aux, stack):
    pila = 'Zo'
    if stack.mostrar() != '':
        pila = stack.mostrar()

    print('\n')

    print('     %s' %cadena_aux)

    print('|    %s    |' %estado)

    print('     %s' %pila)

        print('\n')
    print('\n')
```

0.2 MAIN-PILA (CODIGO)

```
# -*- coding: utf-8 -*-
from __future__ import print_function
from Pila import automata
import random

separador = '='*50
def iniciar():
    continuar = True
    while continuar:
        opcion = imprimir_menu()
        if opcion == 1:
            entrada_consola()
        elif opcion == 2:
            ejecutar_random()
        else:
```

```
            break # Sal del programa
        print('=' * 100)
        opcion = input("Reintentar [s/n]: ")
        if opcion.lower() != 's':
            continuar = False

    print('Saliendo del programa...')

def imprimir_menu():
    print('\n\n%sMenu%s' % (separador, separador))
    print("""
        1.- Entrada en consola (Manual)
        2.- Aleatorio (Automatico)
        3.- Salir
    """)
    try:
        opcion = int(input("Selecciona una opcion valida: "))
        return opcion
    except Exception as e:
        print('Error ', e)
        return 0

def entrada_consola():
    texto = input("Escribe el numero binario: ")
    animacion = ver_animacion()
    automata(texto, animacion)

def ver_animacion():
    opcion = input("Ver animacion [s/n]: ")
    if opcion == 's':
        return True
    else:
        return False

def ejecutar_random():
    i = 0
    longitud_random = random.randint(1, 1000)
    numero_binario = ''
    while i < longitud_random:
        numero_binario += random.choice(['0', '1'])
        i += 1

    print("El numero aleatorio es: ", numero_binario)
    animacion = ver_animacion()
    automata(numero_binario, animacion)
```

```
iniciar()
```

0.3 FUNCIONAMIENTO Y PRUEBAS

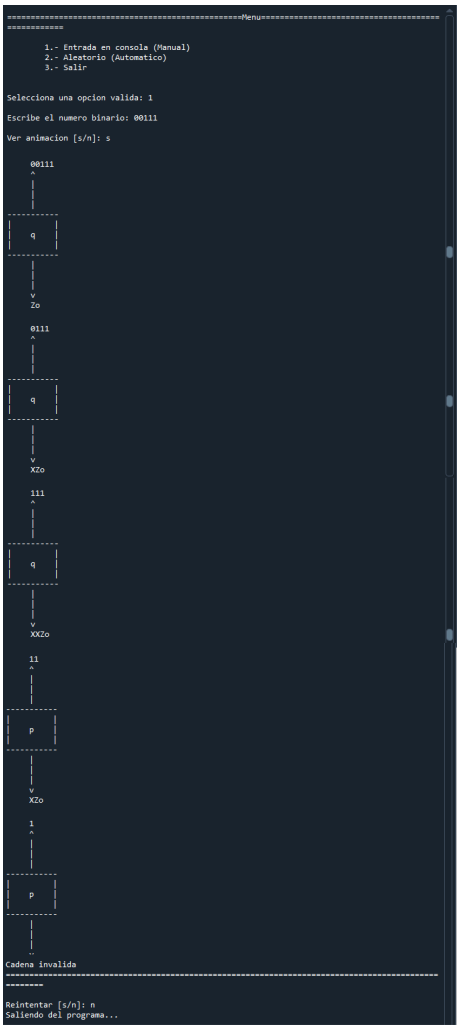

Figure 3: Menú Manual, Historia del Autómata de Pila en animación

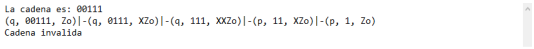

Figure 4: Historia del automata

Figure 5: Menu autómatico, Historia del Autómata de Pila en animación

1 CONCLUSION

Para esta práctica de autómatas de pila, viendo de forma similar los autómatas finitos, como función a las cadenas dadas pudimos ver de que forma con una memoria auxiliar llamada pila, las transiciones entre estados se ejecutan de tal forma que los caracteres dados se evalúan con las descripciones Ids, por lo que funciona correctamente inclusive en la animación. Finalmente para esta práctica hubo algunas complicaciones para la animación y obtener el valor de la pila correspondiente pero con las funciones se logro arreglar y para lo que podría mejorar de esta seria tal ves probar dentro del lenguaje otro tipo de caracteres y ver como aplican.

References

[1] J. E. Hopcroft, R. Motwani, and J. D. Ullman, Introduccion a La Teoria De Automatas, Lenguajes Y Computacion. Addison-Wesley, 2007.

[2] J. D. Ullman, "Finite Automata." http://infolab.stanford.edu/ ullman/ialc/spr10/slides/fa1.pdf, 2010. [Consultado: 2021-12-20].