

INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

PRACTICA 5

Autor:
Gomez Hernandez Alan Javier

27 diciembre, 2021—

EXPRESIÓN REGULAR

RESUMEN

Realizar un programa que genere 10 cadenas a partir de la siguiente expresión regular. $(0 + 10) * (e + 0 + 1)$

- El programa debe ser automático.
- Generar 10 cadenas de manera aleatoria, accediendo a cada parte de la ecuación de manera aleatoria.
- En el caso de la cerradura de Kleene, ésta podrá llegar como máximo hasta el valor 1,000.
- Las cadenas generadas deben de pertenecer al lenguaje y se deberán almacenar en un archivo de texto.
- En otro archivo imprimir la selección de las operaciones de la expresión.
- Si se continua ejecutando el programa, las siguientes cadenas deberán anexarse al mismo archivo.

INTRODUCCION

Con el propósito de simplificar la descripción de los lenguajes regulares se definen las llamadas expresiones regulares. La siguiente es la definición recursiva de las expresiones regulares sobre un alfabeto

Σ

dado. Dibujar un autómata puede ser una tarea engorrosa y utilizar mucho espacio. Una manera más práctica de representar y hacer alusión a un lenguaje regular es utilizando una expresión regular. Una expresión regular sintetiza las reglas que definen la creación de cada una de las cadenas de un lenguaje regular. Se puede demostrar que todo autómata finito como los descritos arriba puede ser expresado como una expresión regular y viceversa. Consiste en una cadena con elementos del alfabeto y otros más que indican la forma en que ellos pueden ser plasmados en la cadena final. Se pueden utilizar los siguientes símbolos.

- ***a***.. Una expresión regular. La más simple consta de un símbolo del alfabeto.
- ***ab***. Sí ***a*** y ***b*** son expresiones regulares, ***ab*** es su concatenación.
- **+**. Un operador lógico OR.

- $*$ (superíndice). La cerradura de Kleen. Equivale a repetir un número arbitrario de veces la expresión regular afectada.

Las expresiones regulares se pueden agrupar con paréntesis de forma que los operadores afecten a su interior como una sola unidad y no a símbolos individuales, como cabría esperar. Para el autómata finito presentado en esta entrada, una expresión regular que lo representa podría ser esta: $(0 + 10)^* (e + 0 + 1)$ y finalmente para esta expresión regular iremos creando cadenas aleatorias.

DESARROLLO

0.1 MAIN-EXPRESION-REGULAR (CODIGO)

```
import itertools
from sys import exit
import random

separador = ''*50
x = []
y = []

while True:
    lista = ["0", "10"]
    listavac = ['e']
    listaux = []
    print('\n\n%sMenu%s' % (separador, separador))
    print("""
        1.- Generar 10 cadenas
        2.- Salir
    """)
    opc = int(input("Selecciona una opcion valida: "))

    if (opc == 1):
        num = 10
    elif (opc == 2):
        print("""Salimos.....
        """)
        exit()
    else:
        print('Opción no válida')
        continue

    for repet in range(num):
        permutations = list(itertools.product(lista, repeat=repet + 1))
        listavac.append(permutations)
    for i in range(num + 1):
        listaux.extend(listavac[i])
    else:
        j = (len(listaux) - 1)
        u = 0
        z = []
        while u <= j:
            z.append(''.join([str(elem) for elem in listaux[u]]))
            u += 1
        x.append(num)
```

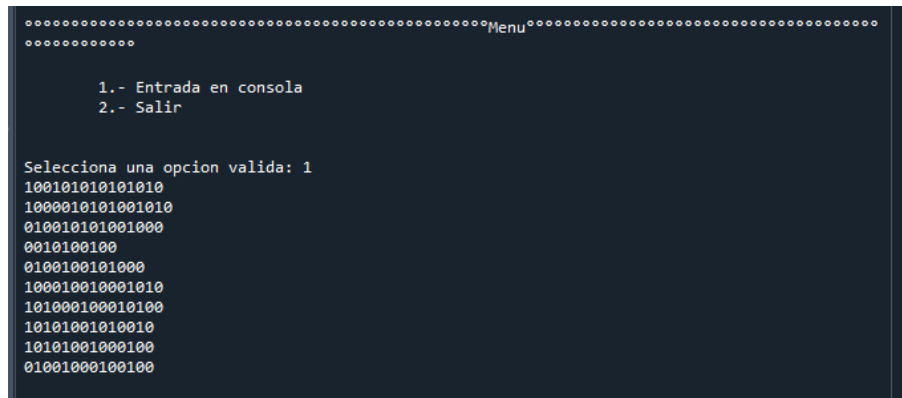
```

y.append((num - 1) * pow(2, num) + 1)

for i in z:
    numUnion = random.choice(range(1,4))
    if(numUnion == 1):
        i = i + '1'
    elif(numUnion == 2):
        i = i + '0'
    else:
        i = i + ' '
aux = 1
f = open("Cadenas.txt", "a+")
f2 = open("Operaciones.txt", "a+")
while (aux<=10):
    numaux = random.choice(range(len(z)))
    print(z[numaux])
    f.write(f'{z[numaux]} ')
    if(z[numaux][-1] == ' '):
        desc = 'e'
    else:
        desc = z[numaux][-1]
    f2.write(f'(L^{numaux})*({desc})\n')
    aux+=1
f.close()
f2.close()

```

0.2 FUNCIONAMIENTO Y PRUEBAS



```

.....Menu.....
.....

1.- Entrada en consola
2.- Salir

Selecciona una opcion valida: 1
1001010101010
1000010101001010
010010101001000
0010100100
0100100101000
100010010001010
101000100010100
10101001010010
10101001000100
01001000100100

```

Figure 1: Se generan 10 cadenas segun la expresion y la guardamos en el bloc de notas



Figure 2: Historia del automata y cadenas generadas y guardadas

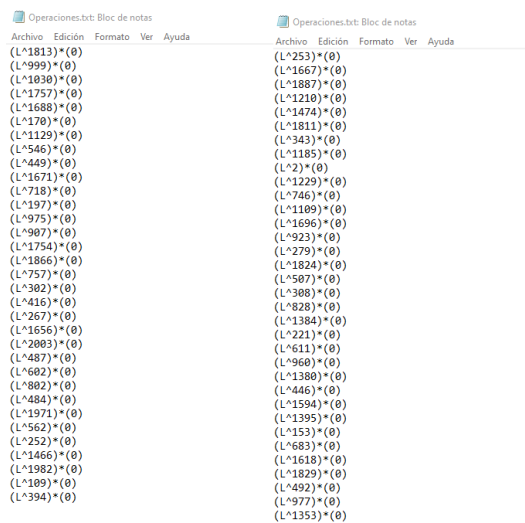


Figure 3: En este bloc iremos guardando como se van armando las cadenas aleatorios segun la expresion regular

1 CONCLUSION

En esta práctica se realizó una expresión regular para crear cadenas de tal forma que se aplique la cerradura de Kleen, por otro lado en esta practica no hubo tanto problema para las funciones e librerías que ocupamos para la elaboración, y bueno en el proceso se le da la opción de salir o continuar, si continua se generan nuevas cadenas pero el registro de como se fueron creando según la expresión lo manda al bloc de notas, con esto lo único que podríamos mejorar o bien tratar, diferentes expresiones.

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].
- [3] M. (2018). Teoria computacional. Didacticae, 1(4), 10–55. <http://ciencias.bogota.unal.edu.co/fileadmin/>[Consultado: 2021-12-20].