



Teoria de
Computacion

ING. Cesar

Gerardo Gonzales

21741355

2 de Septiembre
del 2020

San Pedro Sula,Cortes

Contenido

Introduccion.....	3
Tecnologias	4
Lenguaje de programacion de alto nive Python	4
JSON	4
Networkx.....	4
Matplotlib. pyplot	5
Collections.....	5
Visual Studio Code	5
Justificacion.....	6
Archivo main.Py	7
Archivo Automata.Py	8
Archivo Evaluadores.py.....	10
LA FUNCION dfa_evaluate	10
Pruebas	11
Bibliografia	13

Introduccion

El proyecto a realizar consta de lo aprendido en las primeras 5 semanas de clases, entre ellos los temas de Leguajes, Alfabetos y aceptaciones de un lenguaje todo esto basado en la implemtacion de un automata para falicitir las validaciones de un legnguaje. Un automata es una Máquina automática programable capaz de realizar determinadas operaciones de manera autónoma y sustituir a los seres humanos en algunas tareas, en especial las pesadas, repetitivas o peligrosas; puede estar dotada de sensores, que le permiten adaptarse a nuevas situaciones. Todo esto se basa en la creacion y evaluacion de DFA, en la cual tambien incluimos los Nfa, Nfa epsilon y expresiones regulares en los que en base a lo aprendido en clase, obtenemos su equivalencia en DFA lo cual nos permite evaluar cadenas que podrian ser aceptadas y graficar dicho automatas.

La teoría de autómatas es una rama de la teoría de la computación que estudia las máquinas abstractas y los problemas que éstas son capaces de resolver. La teoría de autómatas está estrechamente relacionada con la teoría del lenguaje formal ya que los autómatas son clasificados a menudo por la clase de lenguajes formales que son capaces de reconocer. También son de gran utilidad en la teoría de la complejidad computacional.

Un autómata es un modelo matemático para una máquina de estado finito (FSM sus siglas en inglés). Una FSM es una máquina que, dada una entrada de símbolos, "salta" a través de una serie de estados de acuerdo a una función de transición (que puede ser expresada como una tabla). En la variedad común "Mealy" de FSMs, esta función de transición dice al autómata a qué estado cambiar dados unos determinados estado y símbolo

Tecnologías

Lenguaje de programación de alto nivel Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

El trabajo realizado en Python para este tipo de programas que tiene un nivel de dificultad alto es muy conveniente debido a que la documentación, tanto como las librerías existentes son de mucha ayuda ya que hacen que nuestros programas sean fáciles de realizar, al ser un lenguaje orientado también a lo funcional tiene acceso a funciones que permiten que el programa sea más rápido en tiempo de ejecución ayudando a resolver los autómatas de manera rápida y eficaz.

JSON

JSON es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera un formato independiente del lenguaje.

El uso de JSON nos permite definir los autómatas de forma ordenada, al momento de importarlos a nuestro código hace que la definición sea mucho más rápida y nos da una facilidad de acceso a la información general del autómata, tanto como a la administración de esa información.

Networkx

NetworkX es una biblioteca de Python para el estudio de gráficos y análisis de redes. NetworkX es un software libre publicado bajo la licencia BSD-new.

Debido al gran parecido que podemos encontrar entre un grafo y un autómata, decidí utilizar esta librería la cual en pocas líneas de código nos permite graficar nuestro autómata.

Matplotlib. pyplot

pyplot es una colección de funciones de estilo de comando que hacen que matplotlib funcione como MATLAB. Cada función de pyplot realiza algún cambio en una figura: por ejemplo, crea una figura, crea un área de trazado en una figura, traza algunas líneas en un área de trazado, decora la trama con etiquetas, etc. En matplotlib

Se necesita esta librería para poder combinarla con network para que lo que graficamos en nuestro automata lo podamos hacer visual.

Collections

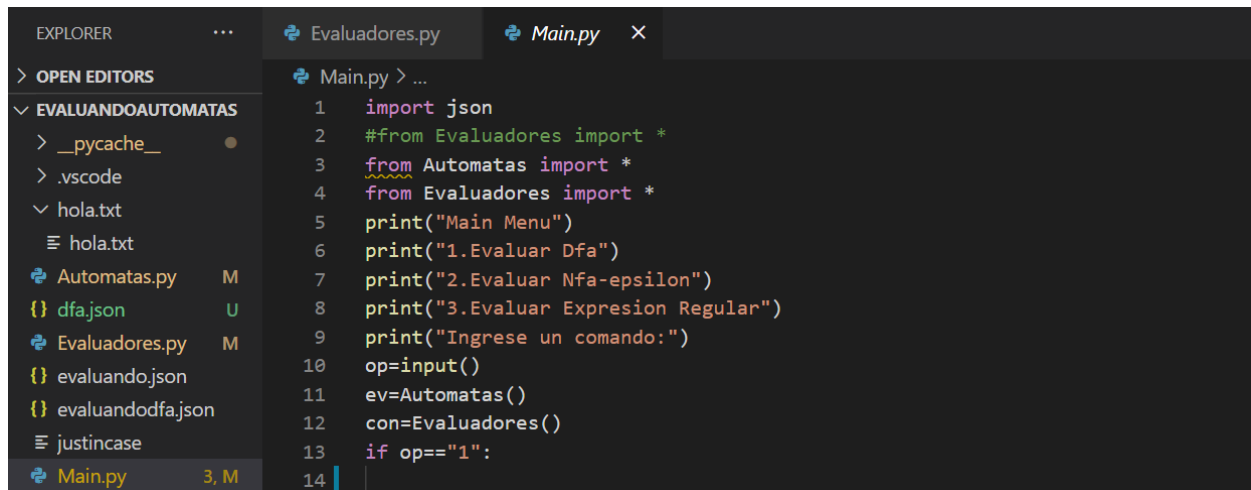
Librería la cual me permite acceder a OrderedDict la cual nos facilita el eliminar los duplicados de una lista o una cadena

Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

Justificacion

Mi codigo se estructura en un main y 2 clase Automatas en la cual hago las conversion y Evaluadores donde evaluo las equivalencias



```
1 import json
2 #from Evaluadores import *
3 from Automatas import *
4 from Evaluadores import *
5 print("Main Menu")
6 print("1.Evaluar Dfa")
7 print("2.Evaluar Nfa-epsilon")
8 print("3.Evaluar Expresion Regular")
9 print("Ingrese un comando:")
10 op=input()
11 ev=Automatas()
12 con=Evaluadores()
13 if op=="1":
14
```

El Primer Paso es definir la estructura del Json la cual esta definida de los diferentes elementos que tiene un automata con sus respectivos valores.



```
1
2 {"Automata": [
3   {
4     "alphabet": ["0", "1"],
5     "states": ["A", "B"],
6     "initial_state": "A",
7     "accepting_states": ["B"],
8     "transitions": [
9       ["A", "0", "A"],
10      ["A", "1", "A"],
11      ["A", "1", "B"]
12    ]
13   }
14 ]
15 }
16
```

Archivo main.Py

Contiene el menu principal de nuestra aplicación en ella tenemos la importacion de diferente archivos los cuales nos permitiran evaluar según la opcion que elijamos y todo esa a una variable que ingresamos por teclado la cual nos permite saber que opcion queremos que se ejecute y dentro de cada uno de los if según lo que hallamos seleccionado estan las diferentes llamadas a funciones que nos permitiran realizar ese proceso.

```

Main.py  X
Main.py > ...
1  import json
2  #from Evaluadores import *
3  from Automatas import *
4  from Evaluadores import *
5  print("Main Menu")
6  print("1.Evaluar Dfa")
7  print("2.Evaluar Nfa-epsilon")
8  print("3.Evaluar Expresion Regular")
9  print("Ingrese un comando:")
10 op=input()
11 ev=Automatas()
12 con=Evaluadores()
13 if op=="1":
14
15     with open('evaluandodfa.json') as contenido:
16         Automaton = json.load(contenido)
17         for automatonValues in Automaton['Automata']:
18             alphabet = automatonValues['alphabet']
19             states = automatonValues['states']
20             initial_states = automatonValues['initial_state']
21             accepting_states = automatonValues['accepting_states']
22             transitions = automatonValues['transitions']
23
24
25     print("Ingrese cadena a evaluar:")
26     src=input()
27
28     con.dfa_evaluate(alphabet, states, initial_states, accepting_states, transitions,src)
29
```

El main consta de la creacion de 2 variables globale las cuales nos permiten acceder a las Funciones dentro de las Clases Automatas y Evaluadores

Archivo Automata.Py

Dentro de este archivos tenemos la definicion e implemetacion de las funciones que trasforman de nfa epsilon a dfa equivalente y la evaluacion de las expresiones regulares todo esto perteneciente a la clase de Automatas.

```
Automatas.py X
Automatas.py > Automatas
1  import json
2  from collections import OrderedDict
3  from Evaluadores import *
4
5  class Automatas:
6
7  > def nfae_dfa(self,alphabet, states, initial_state, accepting_states, transitions):...
130
131
132 > def expresionregular(self,expresion):...
146
147
148
```

La transformacion de nfa epsilon a dfa equivalente consta de que tengo un arreglo vacio el cual es la cerradura de epsilo para empezar paso por cada un de los estados y junto a ellos por cada estado voy revisando todas las transiciones para obtener su cerradura lo cual en el arreglo termina siendo la concatenacion de todos los estados a los cuales puede llegar con epsilon. Luego hacemos la delta cerradura de epsilon la cual empezamos con otro arreglo pero para ellos esta vez necesitaremos las cerradura entonces en base a cada una de las posibilidades del alfabeto voy evaluando a donde podria llegar la cerradura de epsilon de cada uno de los estados concatenados, luego creamos otro arreglo que en base a los obtenido en el paso anterior obtenemos la cerradura de epsilon de cada uno de los que tenemos siempre tomando en cuenta lo que puede tener nuestro alfabeto. Por ultimo hacemos una tranformacion de nfa a dfa equivalente la cual es un poco cmplicada ya que hago 3 for anidado del mismo arreglo ya que en uno lleno en el otro reviso si lo que quiero aregar éxito y el otro lo uso para llenar las nuevas transiciones que saldrán y todo esto validando la informacion, por

ultimo ingresamos la cadena a evaluar y evaluamos el dfa equivalente.

```
Automatas.py X
Automatas.py > Automatas > nfae_dfa
1  import json
2  from collections import OrderedDict
3  from Evaluadores import *
4
5  class Automatas:
6
7      def nfae_dfa(self,alphabet, states, initial_state, accepting_states, transitions):
8
9          cerraduraEpsilon=[]
10
11         for est in states:
12             newCerr = ''
13             for trans in transitions:
14                 if trans[0]==est and trans[1]=='e':
15                     newCerr=newCerr+trans[2]
16             newCerr=newCerr+est
17             cerraduraEpsilon.append((est,newCerr))
18
19
20         #print(cerraduraEpsilon)
21         #print("-----Cerradura Epsilon-----")
22         for i in range(len(cerraduraEpsilon)):
23             newword = list(OrderedDict.fromkeys(cerraduraEpsilon[i][1]))
24             concatword=""
25             for c in newword:
26                 concatword= concatword+c
27             cerraduraEpsilon[i]=(cerraduraEpsilon[i][0],concatword)
28         deltaCerraduraEpsilon=[]
29
30         for alph in alphabet:
```

Archivo Evaluadores.py

En este archivos podemos encontrar diferentes funciones auxiliares que utilize para resolver los problemas que surgian poco a poco.

```
Evaluadores.py X
Evaluadores.py > Evaluadores > printAutomaton
1
2 import networkx as nx
3 import matplotlib.pyplot as plt
4
5
6
7 class Evaluadores():
8
9 > def dfa_evaluate(self,alphabet, states, initial_state, accepting_states, transitions, str_test):...
32
33 > def returnNextState(self,transitions,initial_state,source): ...
39
40 > def printAutomaton(self, estados, transiciones):...
49
```

LA FUNCION `dfa_evaluate` de la clase `Evaluadores` es la encargada de evaluar los automatas y por ello recibe toda la informacion que tiene un automata y dentro de un ciclo evalua la `src_test` que nosotros le mandemos para evaluar en el dfa y dentro de ella usamos la funcion `returnNextState` a cual le mandamos las transiciones y retorn hacia donde me movere en el automata por ultimo depues de evaluar y decir si la cadena pertenece o no logramos graficar de manera visual el automata gracias a las librerias `networkc` y `matplotlin.pyplot`.

Pd:la funcion `evaluate` es la que creamos en la clase

Pruebas

Evaluando Dfa Simple

Figure 1

```
{} evaluandofajson > ...
1  {"Automata": [
2    {
3      "alphabet": ["0", "1"],
4      "states": ["s0", "s1", "s2"],
5      "initial_state": "s0",
6      "accepting_states": ["s0"],
7      "transitions": [
8        ["s0", "0", "s0"],
9        ["s0", "1", "s1"],
10       ["s1", "0", "s2"],
11       ["s1", "1", "s1"],
12       ["s2", "1", "s1"],
13       ["s2", "0", "s1"]
14     ]
15   }
16 ]
17 }
18
19
20
21
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Ingrese cadena a evaluar:
100
s0 1 s1
s1 0 s2
s2 0 s1
No pertenece a L(M)

x=-1.195 y=-0.331

Figure 1

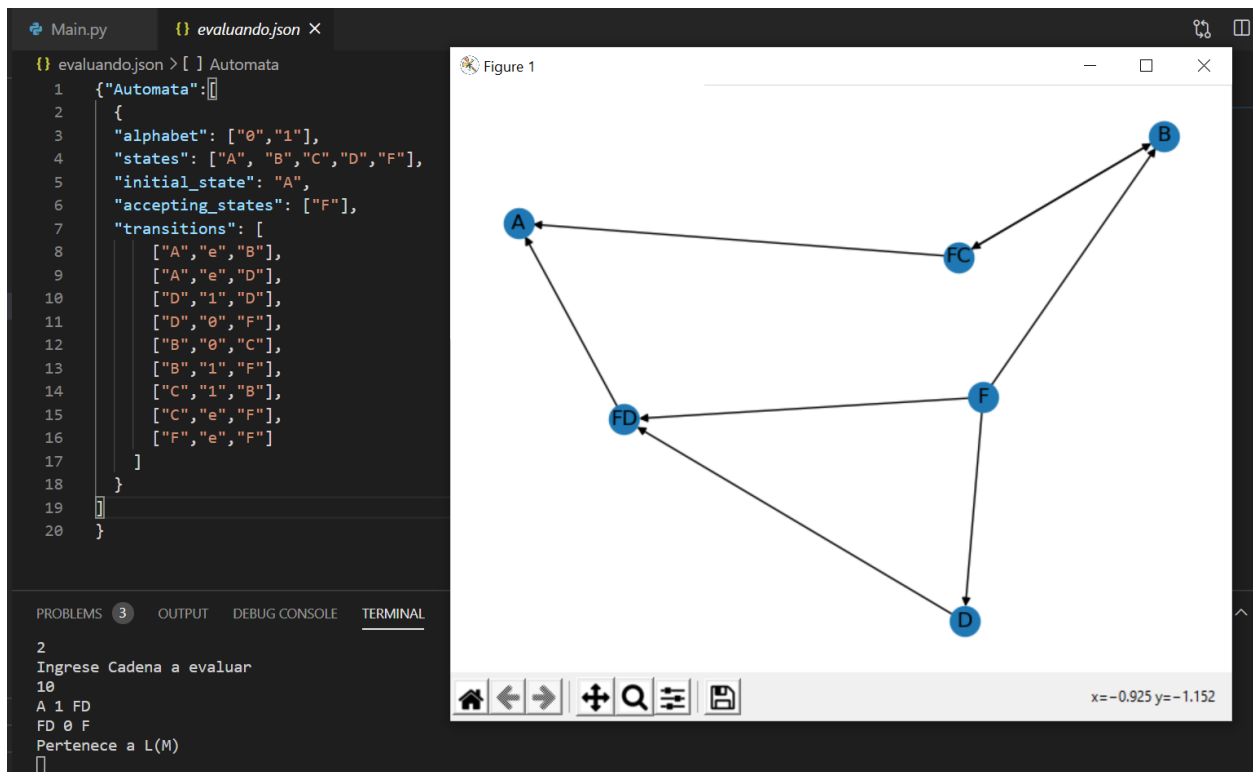
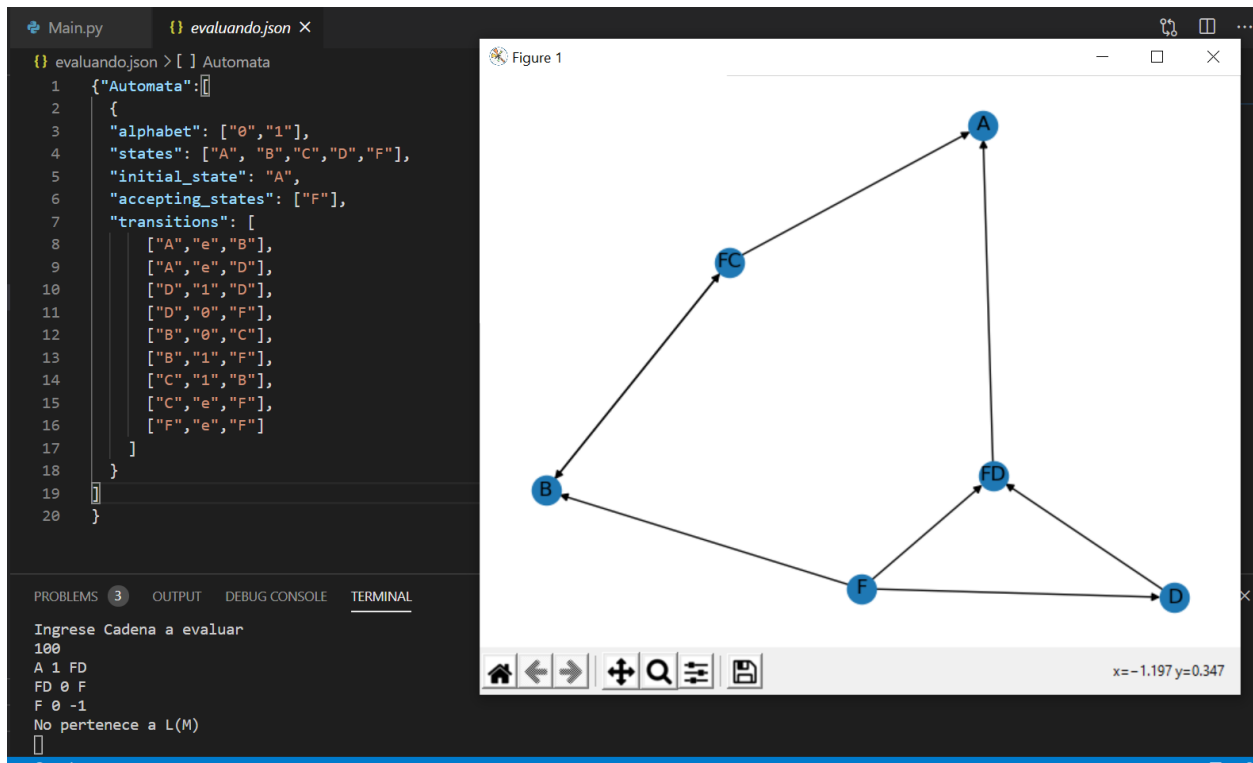
```
{} dfajson > ...
1  {"Automata": [
2    {
3      "alphabet": ["0", "1"],
4      "states": ["A", "B"],
5      "initial_state": "A",
6      "accepting_states": ["B"],
7      "transitions": [
8        ["A", "0", "A"],
9        ["A", "1", "A"],
10       ["A", "1", "B"]
11     ]
12   }
13 ]
14 }
15
16
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL

Ingrese cadena a evaluar:
000
s0 0 s0
s0 0 s0
s0 0 s0
Pertenece a L(M)

x=-0.2984 y=-0.562

Convirtiendo nfa epsilon y evaluando en dfa equivalente



Bibliografia

https://es.wikipedia.org/wiki/Teor%C3%ADa_de_aut%C3%B3matas

https://es.wikipedia.org/wiki/Aut%C3%B3mata_programable#:~:text=En%20electr%C3%B3nica%20un%20aut%C3%B3mata%20es,en%20ambiente%20industrial%2C%20procesos%20secuenciales.

https://es.wikipedia.org/wiki/Visual_Studio_Code

<https://matplotlib.org/3.1.1/tutorials/introductory/pyplot.html#:~:text=pyplot%20is%20a%20collection%20of,in%20matplotlib.>