CASTRO CRUCES JORGE E.	INSTITUTO POLITÉCNICO NACIONAL	ESCUELA SUPERIOR DE CÓMPUTO
SEGUNDO PARCIAL	PRACTICA 2 – GENERADOR DE AF- Ɛ	16/04/2021
3CV18	COMPILADORES	Hernández Olvera Luis E

SECCIÓN TEÓRICA O INTRODUCCIÓN

Autómatas finitos:

1) Los autómatas finitos son reconocedores; sólo dicen "sí" o "no" en relación con cada posible cadena de entrada.
2) Los autómatas finitos pueden ser de dos tipos:
   a) Los autómatas finitos no deterministas (AFN) no tienen restricciones en cuanto a las etiquetas de sus líneas. Un símbolo puede etiquetar a varias líneas que surgen del mismo estado, y, la cadena vacía, es una posible etiqueta.
   b) Los autómatas finitos deterministas (AFD) tienen, para cada estado, y para cada símbolo de su alfabeto de entrada, exactamente una línea con ese símbolo que sale de ese estado.

Tanto los autómatas finitos deterministas como los no deterministas son capaces de reconocer los mismos lenguajes. De hecho, estos lenguajes son exactamente los mismos lenguajes, conocidos como lenguajes regulares, que pueden describir las expresiones regulares.[1]

Autómatas finitos no deterministas:

Un autómata finito no determinista (AFN) consiste en:

1) Un conjunto finito de estados S.
2) Un conjunto de símbolos de entrada Σ, el alfabeto de entrada. Suponemos que, que representa a la cadena vacía, nunca será miembro de Σ.
3) Una función de transición que proporciona, para cada estado y para cada símbolo en Σ ∪ {}, un conjunto de estados siguientes.
4) Un estado s0 de S, que se distingue como el estado inicial.
5) Un conjunto de estados F, un subconjunto de S, que se distinguen como los estados aceptantes (o estados finales).[1]

AFD y lenguajes:

Para trabajar con los AFD es necesario usar ciertas definiciones y notaciones. Si M es un AFD, entonces el lenguaje aceptado por M es

$$L(M) = \{w \in \Sigma^* | w \; es \; aceptada \; por M\}$$

Por tanto, L (M) es el conjunto de cadenas que hacen que M pase de su estado inicial a un estado de aceptación. Por ejemplo, el lenguaje aceptado por el AFD (*), presentado en la última sección, es

$$L(M) = \{w \in \{a, b\}^* | w \; no \; contiene \; tres \; bes \; consecutivas\}$$

Merece la pena hacer hincapié en que L (M) está formado por todas las cadenas aceptadas por M, y no que es un conjunto de cadenas que son todas aceptadas por M. [2]

Python tiene muchos tipos de estructura de datos, una de ellas es la lista, es por ello por lo que en este tutorial aprenderemos a usarlas y veremos sus métodos más utilizados. Si no conoces qué es Python, te lo explico a continuación.

"Python es un lenguaje de programación interpretado de alto nivel y orientado a objetos, con el cual podemos crear todo tipo de aplicaciones."

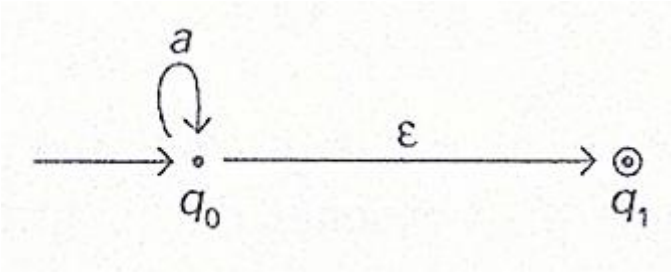
En algunos lenguajes de programación se las conocen como arreglos o matrices; y se caracterizan porque los elementos están entre corchetes y separados por una coma.

lista = [1, 2, 3, 4]

Ɛ-TRANSICIONES:

Podemos ampliar la definición de autómata finito no determinista para incluir transiciones de un estado a otro que no dependan de ninguna entrada. Tales transiciones se llaman 8-transiciones porque al realizarse no consumen ningún símbolo de la entrada.

El autómata puede cambiar su estado de q0 a q1 sin consumir nada en la entrada. Obsérvese que q1 es el único estado de aceptación de este AFN. Si w es cualquier cadena de 0 o más *aes*, este autómata cicla sobre q0 hasta que consume las *aes*. Una vez que la cadena se vacía, se desplaza a q1 y lo acepta.



Autómata con transición en Ɛ*[3]*

¿Qué es una lista?

Una lista es una estructura de datos y un tipo de dato en Python con características especiales. Lo especial de las listas en Python es que nos permiten almacenar cualquier tipo de valor como enteros, cadenas y hasta otras funciones; por ejemplo:

lista = [1, 2.5, 'DevCode', [5,6] ,4] *[4]*

PROBLEMAS RELACIONADOS AL TEMA A LOS CUALES TE ENFRENTASTE AL PROGRAMAR LA PRÁCTICA

Pues, comenzando desde lo más básico, que vengo comentando desde la práctica pasada, que es:

- Programación Orientada a Objetos
- Manejo de listas en lenguaje Python
- Comprensión de la Construcción de Thompson
- Aplicar la Construcción de Thompson en la práctica anterior
- Idear la forma más fácil de imprimir el camino que recorre la cadena ingresada por el usuario

Pero sobre todo, la parte más complicada con la que me topé fue la corrección del Autómata y poder indicar el estado exacto en el que produce el error.

Dicen que: Lo que no te mata solo te fortalece; Y creo firmemente que este tipo de retos forjan en el ingeniero el carácter necesario para poder afrontar los retos laborales que nos esperan en el futuro próximo.

CAPTURAS DE PANTALLA

Ejecución del programa leyendo el AF 1 con las cadenas siguientes:

ba

```
C:\Users\georg\Desktop\ESCOM\7mo. Semestre\Compis\Practicas\Practica2>python AF.py
Ingresa el nombre del archivo a LeerAF: Thompson1.txt
Ingrese la cadena que desea validar: ba

___________Cadena valida___________
Recorrido:
0(ϵ) -> 8(b) -> 9(ϵ) -> 10(a) -> 11
```

ba1a

```
C:\Users\georg\Desktop\ESCOM\7mo. Semestre\Compis\Practicas\Practica2>python AF.py
Ingresa el nombre del archivo a LeerAF: Thompson1.txt
Ingrese la cadena que desea validar: ba1a

___________Cadena valida___________
Recorrido:
0(ϵ) -> 1(ϵ) -> 2(ϵ) -> 3(b) -> 4(ϵ) -> 5(a) -> Simbolo erroneo (1)...ignorado -> 6(ϵ) -> 7(ϵ) -> 10(a) -> 11
```

abbzbaa

```
C:\Users\georg\Desktop\ESCOM\7mo. Semestre\Compis\Practicas\Practica2>python AF.py
Ingresa el nombre del archivo a LeerAF: Thompson1.txt
Ingrese la cadena que desea validar: abbzbaa

___________Cadena valida___________
Recorrido:
0(ϵ) -> 1(ϵ) -> 2(ϵ) -> 5(a) -> 6(ϵ) -> 2(ϵ) -> 3(b) -> 4(ϵ) -> 3(b) -> Simbolo erroneo (z)...ignorado -> 4(ϵ) -> 3(b) -> 4(ϵ) -> 5(a) -> 6(ϵ) -> 7(ϵ) -> 10(a) -> 11
```

ab@abaa

```
C:\Users\georg\Desktop\ESCOM\7mo. Semestre\Compis\Practicas\Practica2>python AF.py
Ingresa el nombre del archivo a LeerAF: Thompson1.txt
Ingrese la cadena que desea validar: ab@abaa

___________Cadena valida___________
Recorrido:
0(ϵ) -> 1(ϵ) -> 2(ϵ) -> 5(a) -> 6(ϵ) -> 2(ϵ) -> 3(b) -> Simbolo erroneo (@)...ignorado -> 4(ϵ) -> 5(a) -> 6(ϵ) -> 2(ϵ) -> 3(b) -> 4(ϵ) -> 5(a) -> 6(ϵ) -> 7(ϵ) -> 10(a)
-> 11
```

Comentario: El programa fue capaz de indicar y corregir el símbolo erróneo sin problema

Ejecución del programa leyendo el AF 2 con las cadenas siguientes:

aaaa

```
C:\Users\georg\Desktop\ESCOM\7mo. Semestre\Compis\Practicas\Practica2>python AF.py
Ingresa el nombre del archivo a LeerAF: Thompson2.txt
Ingrese la cadena que desea validar: aaaa

___________Cadena valida___________
Recorrido:
0(ϵ) -> 8(ϵ) -> 9(a) -> 10(ϵ) -> 9(a) -> 10(ϵ) -> 9(a) -> 10(ϵ) -> 9(a) -> 10(ϵ) -> 11(ϵ) -> 12
```

ab@ab

```
C:\Users\georg\Desktop\ESCOM\7mo. Semestre\Compis\Practicas\Practica2>python AF.py
Ingresa el nombre del archivo a LeerAF: Thompson2.txt
Ingrese la cadena que desea validar: ab@ab
Error: La cadena no es valida...
```

cbqbb

```
C:\Users\georg\Desktop\ESCOM\7mo. Semestre\Compis\Practicas\Practica2>python AF.py
Ingresa el nombre del archivo a LeerAF: Thompson2.txt
Ingrese la cadena que desea validar: cbqbb

___________Cadena valida___________
Recorrido:
0(ϵ) -> 1(c) -> 2(ϵ) -> 5(b) -> Simbolo erroneo (q)...ignorado -> 6(ϵ) -> 5(b) -> 6(ϵ) -> 5(b) -> 6(ϵ) -> 7(ϵ) -> 12
```

cb_a

```
C:\Users\georg\Desktop\ESCOM\7mo. Semestre\Compis\Practicas\Practica2>python AF.py
Ingresa el nombre del archivo a LeerAF: Thompson2.txt
Ingrese la cadena que desea validar: cb_a
Error: La cadena no es valida...
```

Comentario: El programa fue capaz de indicar y corregir el símbolo erróneo sin problema. Únicamente, en los casos que la cadena no fue valida, es por que de plano la cadena no logra llegar al Estado Final.

CONCLUSIÓN

¿Qué aprendí de esta práctica?

Claramente, aprendí muchas cosas, entre ellas está:

- Implementar el Método de Thompson en lenguaje Python
- Implementar transiciones en Ɛ de un AF en lenguaje Python
- Manejo de AF mediante listas en lenguaje Python
- Mostrar el camino que recorre una cadena dentro del AF para ser validada
- En caso de no ser válida la cadena:
    - Mostrar en que estado se generó el error
    - Corregirlo de forma que se pueda validar mediante el AF

¿En qué me ayudó esta práctica?

Principalmente, me ayudó a practicar y adquirir mayor conocimiento de la Programación Orientada a Objetos en lenguaje Python.

En un segundo plano, me apoyó bastante para adquirir una mejor comprensión de la diferencia entre un Autómata Finito Determinista y un Autómata Finito No Determinista, así como, las transiciones por Ɛ; Y las ventajas de aplicar el Método de Thompson.

REFERENCIAS

[1] Aho, A., 2008. Compiladores. 2nd ed. México: Pearson Educación de México, SA de CV, p.147.
[2] Kelley, D., Joyanes Aguilar, L. and Diéz Platas, L., 1995. Teoría de autómatas y lenguajes formales. 1st ed. Madrid: Prentice Hall, p.59.
[3] Kelley, D., Joyanes Aguilar, L. and Diéz Platas, L., 1995. Teoría de autómatas y lenguajes formales. 1st ed. Madrid: Prentice Hall, p.70.
[4] Mariños Urquiaga, J., 2021. Listas en Python. [online] DevCode Tutoriales. Available at: https://devcode.la/tutoriales/listas-python/ [Accessed 23 March 2021].