Universidad Internacional de Valencia (VIU)

13GIIN – Teoria de automatas y lenguajes formales

Primer Portafolio – ACT1

Alumno: Gagliardo Miguel Angel

Archivos incluidos

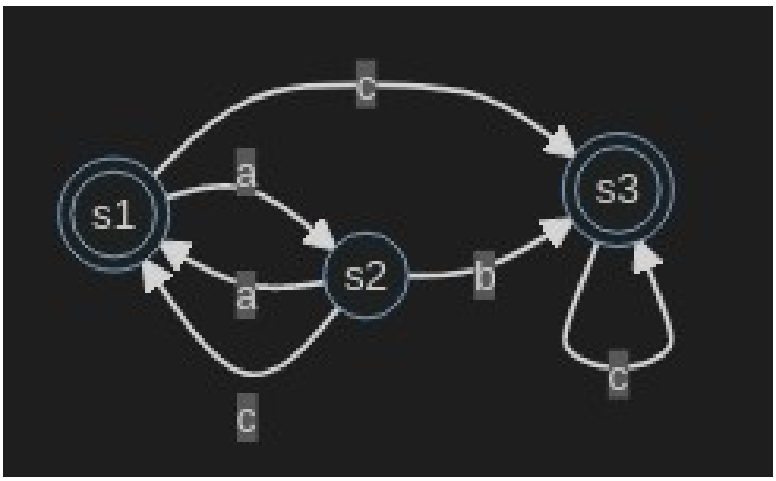
Para el trabajo se incluyen 3 archivos en un .zip

- afn.txt
- afn-e.txt
- afn-e.py

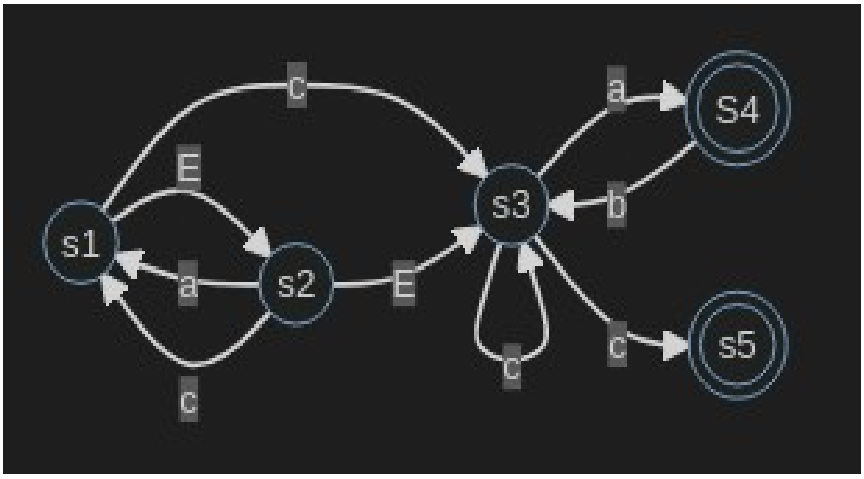
Tanto el primer como el segundo archivos txt contienen automatas, el primero un AFN sencillo, y el segundo un AFN-ε.

Dado que el caracter Epsilon no es parte del estandar ANSI sencilla, se utiliza la “E” mayuscula para simular el mismo en el alfabeto.

Para el automata AFN sencillo, se ha utilizado siguiendo el siguiente grafico:



Mientras que para el **AFN-ε** se ha utilizado un automata similar al siguiente.



Los ejemplos que se ven a continuacion, estaran relacionados con estas dos graficas.

Como ejecutar

Para ejecutar el programa, es requisito necesario tener **python 3.10** instalado, dado que hace uso del tipado opcional del lenguaje.

El programa pide que se ingrese un archivo, puede usted bien usar **afn.txt** o **afn-e.txt,** y luego una cadena que usted desee probar (puede ser valida o invalida).

Por ejemplo para el AFN-ε:

```
01:23 $ python3 afn-e.py
Ingrese un archivo a leer: afn-e.txt
Ingrese la cadena a ser reconocida por el automata: cc
La palabra `cc` ha sido reconocida por el automata AFN-ε.
```

Y para el AFN:

```
01:25 $ python3 afn-e.py
Ingrese un archivo a leer: afn.txt
Ingrese la cadena a ser reconocida por el automata: aaabc
La palabra `aaabc` ha sido reconocida por el automata AFN.
```

El programa internamente se divide en 2 clases (una clase padre y una que hereda de ella) y una funcion externa que lee el archivo ingresado. En la funcion main() decide en base al alfabeto del archivo, que tipo de objeto AFN instanciara (si el que soporta transiciones epsilon o no) y luego ejecuta el metodo **procesar.**

Por ultimo, ambos automatas reconocen cadenas **total y parcialmente**.

```
02:39 $ python3 afn-e.py
Ingrese un archivo a leer: afn.txt
Ingrese la cadena a ser reconocida por el automata: aacb
La palabra `aacb` no ha sido reconocida en su totalidad, pero `aac` ha sido reconocida parcialmente por el automata AFN.
```

Comparacion con AFD

Al comparar con el programa hecho previamente en clase para el AFD, podemos ver unas claras diferencias:

1. Los AFN dado que tienen mas de un estado final, necesitan recorrer todas las opciones de la cadena al mismo tiempo para verificar si por alguna de las bifurcaciones se llega a uno de los estados aceptados.

2. Para el AFN-ε, se ha debido modificar el metodo **_disparable** para que, antes de recorrer las transiciones en busqueda del simbolo, se realiza la **_cerradura_epsilon,** quedando asi solamente como output los estados alcanzables desde el simbolo indicado.

3. La compleijdad de los programas claramente es mayor, dado que se debe verificar no solo varios caminos al mismo tiempo, si no varias opciones de estado final, loops, etc.

4. Por ultimo, el archivo del programa esta comentado en las lineas y metodos mas reelevantes utilizando el estandar PEP8 de python para facilitar la comprension al leerlo.