Pablo de la Mora Vega

A01020365

Reporte

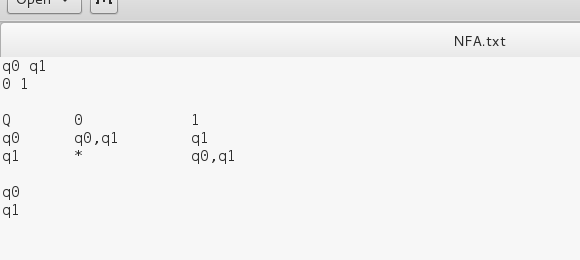
Proyecto 3

**Manual de Usuario:**

Para poder correr este programa lo que se necesita es tener el código del proyecto junto con 2 archivos de texto .txt, en la misma carpeta. También se necesita tener instalado graphviz que es una librería de Python en donde estoy imprimiendo el DFA que se crea a partir del NFA brindado por el usuario. Primero que nada se tiene que tener un archivo de texto llamado NFA.txt en donde se tiene que introducir los datos del NFA. La primera línea del archivo se refiere a los estados que contiene el NFA separados por espacios. La segunda línea se refiere a los símbolos aceptados por el alfabeto separado por espacios. Después de eso se tiene que dejar una línea en blanco y después se introduce la tabla de transiciones en donde la primera fila de la tabla es la letra que representa los estados, seguido de los símbolos aceptados en el alfabeto, después el primer elemento de cada renglón es el estado que se está representando seguido de los estados que visita ese primer estado. No importa la cantidad de espacios que se pongan, por lo que facilita la visualización de la tabla si se ordena bien como se podrá observar en el ejemplo. Para representar al vacío se pone un “ \* ”, para evitar confusiones.

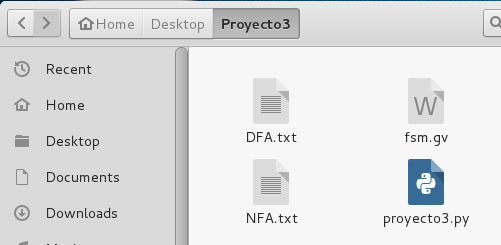
Los datos que se introducen en la tabla tienen que ir en orden alfabético, por ejemplo en la imagen del ejemplo abajo se ve cómo se introduce “q0, q1” y no “q1, q0”.

Ya que se introdujo la tabla se tiene que dejar otra línea en blanco abajo y en la línea que sigue se introduce el estado inicial. En la última línea lo que se introduce es el estado final. Un ejemplo de cómo se introducen los datos es así:



Como podemos ver en este ejemplo solo hay 2 estados que son q1 y q2, el lenguaje aceptado es 0,1. Después se deja el renglón necesario y se introduce la tabla, en donde se ve a donde llega cada estado si se le introduce un 0 y un 1. Después se vuelve a dejar el espacio necesario y se introduce el estado inicial. Y finalmente se introducen en la última línea los estados finales separados por espacios si es el caso que sea más de uno.

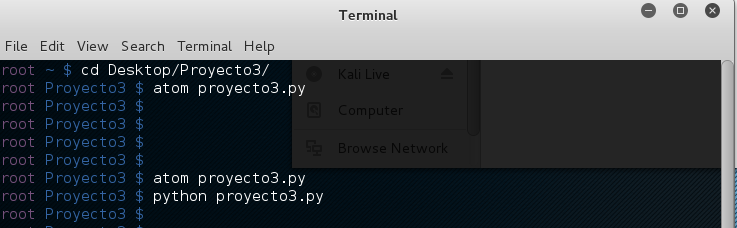
Ya que se tiene este archivo en la forma correcta se necesita tener otro archivo con formato .txt en la carpeta que se encuentre con el nombre de DFA. Ahí es donde el programa va a escribir los resultados ya que los datos sean procesados por lo que es importante tener ese archivo en la carpeta, de otra manera no va a funcionar.



En esta imagen se ve cómo se tiene los dos archivos de texto. De preferencia se prefiere tener el archivo DFA vacío ya que para Linux borra los datos que había y los sobrescribe con los nuevos datos con los que se corrió pero puede que para Windows no funcione de igual manera por lo que es preferible que se encuentre vacío para evitar problemas.

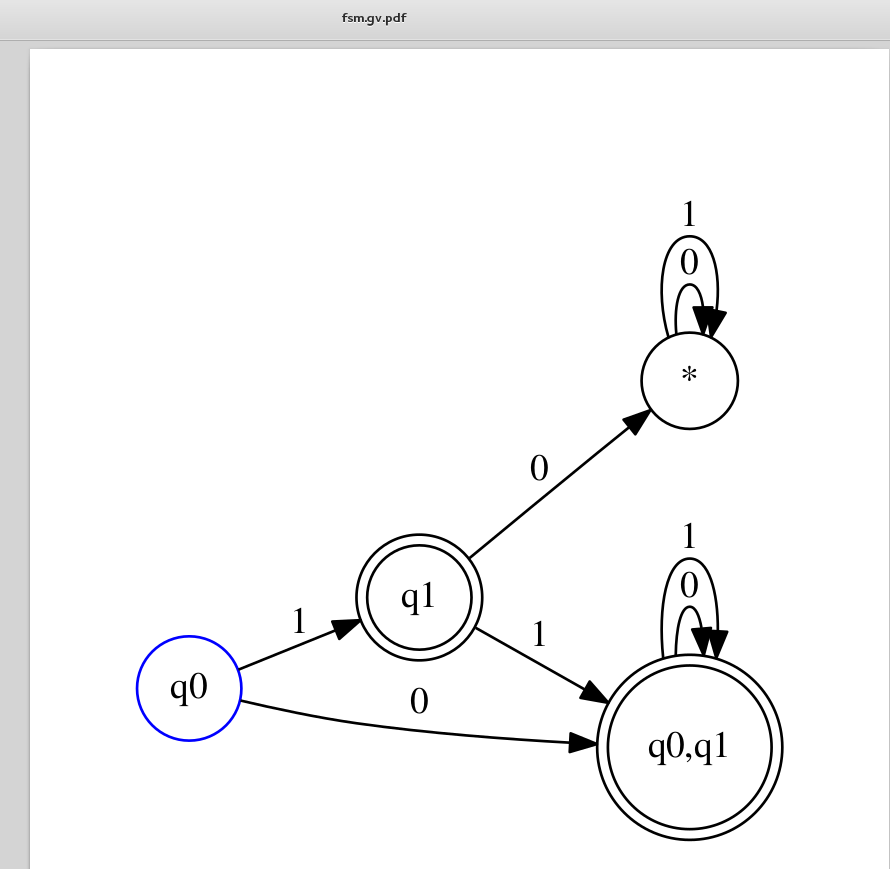
Ya que se contiene esto entonces si se puede ejecutar el programa. El programa desde Linux se ejecuta desde la línea de comandos. Para poder ejecutarlo hay que estar ubicado en la carpeta donde se contiene el código y escribir “python proyecto3.py”. Escribiendo esto sin las comillas ejecuta el programa. Una vez hecho esto no se va a mostrar nada en la línea de comandos ya que el programa no imprime nada ahí si no que lo hace en el archivo DFA.

Ejemplo:



Aquí únicamente importa la última línea que es la que ejecuta el programa cuando nos encontramos en la carpeta con todos los requerimientos ya dichos anteriormente.

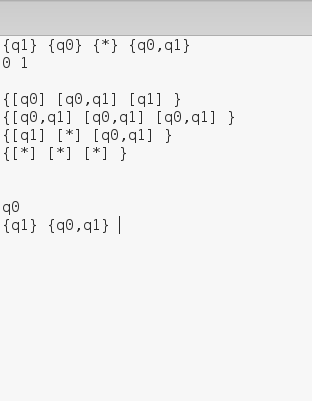
Ya que se ejecuta el programa también va a crear el DFA visualmente para que se pueda tener una mejor comprensión de la transformación. El archivo creado es un PDF y se muestra a continuación si se introducen los datos del ejemplo:



En la imagen se muestra con un círculo azul el estado inicial y con doble circulo los estados finales.

Ahora bien si se abre el archivo DFA se muestran todos los resultados en el mismo orden en el que se introdujeron. La primera línea muestra los estados del DFA, la segunda muestra los símbolos aceptados, después se muestra un espacio y se muestra la tabla de transiciones, omitiendo la primera línea que se introducía en la tabla de DFA que solo era para orientarse. Después de la tabla se tiene otra línea en blanco seguida del estado inicial y en la última línea se tienen los estados finales.

Ejemplo:



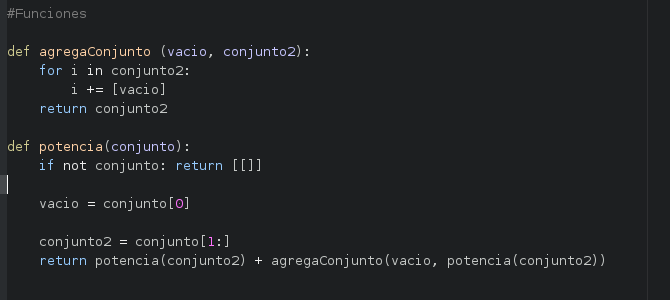
Esta tabla muestra el resultado final y si se compara con el grafo creado es lo mismo.

Un comentario importante es que el asterisco es tomado como el estado basura en todos los ejemplos, (si es que existe un estado basura).

**Descripción Técnica del Código:**

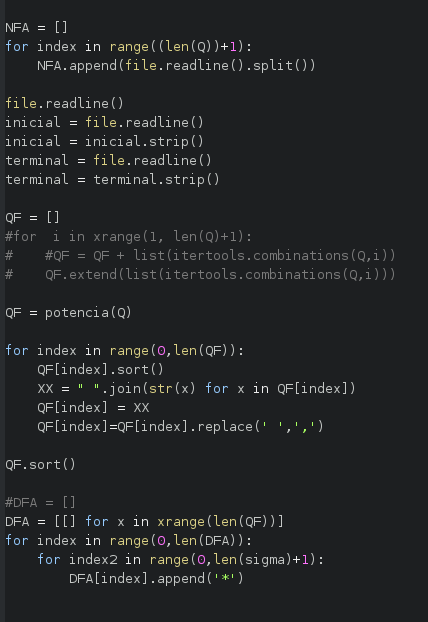
Este programa fue implementado en python. A continuación se van a explicar los procesos más importantes usados para que este programa funcionara.

Primero lo que se hace es leer todas las entradas en el archivo proporcionadas por el usuario y se almacenan en listas.



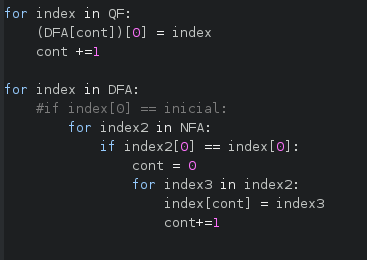
Estas funciones lo que hacen es que te muestran todos los conjuntos posibles que pueden ser creados a partir de los conjuntos Q proporcionados por el usuario. Se almacenan en una variable (se almacenan todas las posibles combinaciones pero no quiere decir que se vayan a usar todas).

La parte más importante del programa es como se trabaja con la tabla de transiciones para poder crear la nueva.



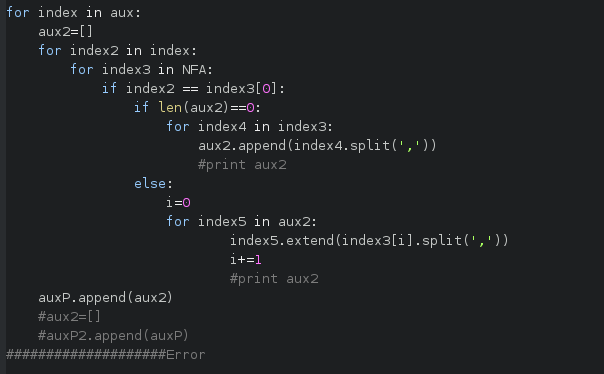
Primero se almacena la tabla proporcionada. Después de eso se tienen que hacer algunas modificaciones a las listas ya que los datos se almacenan en strings pero separados por comas por lo que se tienen que juntar en uno solo.

Ya que se tiene esto se crea la tabla DFA que al princio lo que se hace es crear una tabla con el número de columnas que la que se introdujo en el archivo NFA ya que el número de columnas no cambia pero con la cantidad de filas que creo la función para crear todas las posibilidades. Estas se rellenan con \* que significa vacío en el formato de este programa.



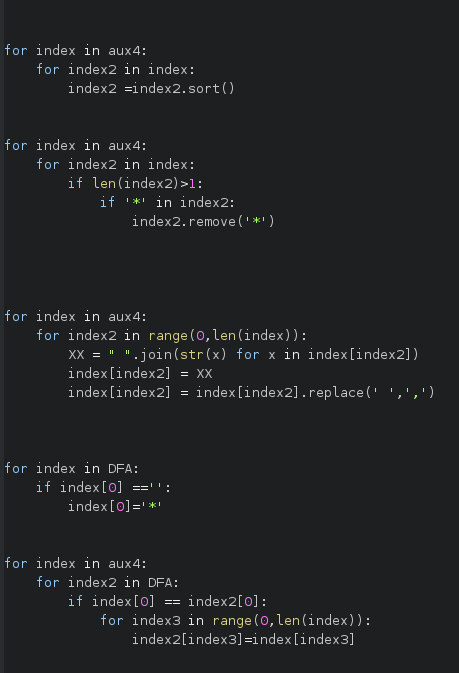
En esta imagen es donde las cosas se empiezan a poner interesantes ya que se empieza a ver como se llena la tabla de transiciones. Primero lo que se hace es buscar en la nueva tabla los elementos que ya se tienen los datos en la tabla NFA por lo que se buscan y se sustituyen esos datos debido a que son iguales.

Después de eso viene este proceso por el cual se terminan de llenar los datos que son nuevos:



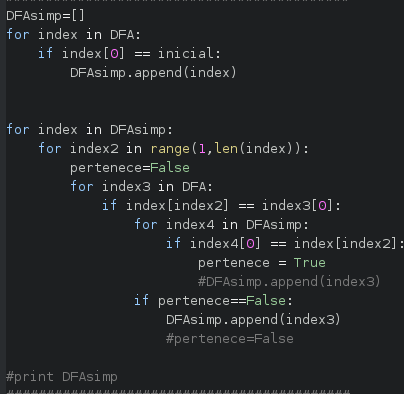
En este caso aux es el DFA guardado para auxiliar a armar la tabla. Lo que se hace es que se empieza a recorrer la tabla de NFA y busca los estados a los que llegan con los símbolos aceptados. Si se llega a más de un símbolo se separan estos símbolos y se identifican de forma individual a donde llegan. De esta manera te aseguras que veas a todos los estados a donde llega cada uno y los estados que visitan a más de un estado con un símbolo son los estados nuevos.

Todo lo que se hace a continuación tan solo son simplificaciones para que los estados no contengan \* si ya tienen un estado incluido.



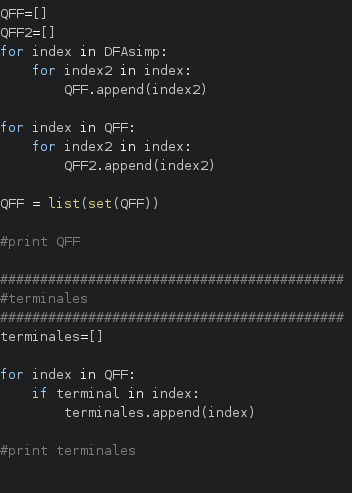
Ya que se tiene la tabla bien formada con todos los estados posibles lo que hice fue simplificar la tabla para que no contenga los estados a los cuales no se visita.

Esto fue posible gracias al siguiente código:



Lo que se hace aquí es comenzar por el nodo inicial y de ahí empezar a recorrer los nodos los cuales se visitan y agregarlos a una lista nueva llamada DFAsimp. Gracias a esto se eliminan muchos nodos innecesarios en la tabla.

Lo siguiente fue simplificar los estados de acuerdo a la nueva tabla que ya tenemos.



Y de la tabla tan solo checamos si contiene el substring de algún terminal en el estado lo que indica que también sería terminal.

Para finalizar solo se imprimen los estados en el archivo y se crea el grafo.



Se imprime de una manera sencilla. Primero crea los estados terminales con doble círculo y checa si el incial es terminal para ponerlo con doble círculo o no. Después se van creando con un for todos los demás nodos que no son terminales y se crean las relaciones.

Bibliografía:

Stackoverflow.com

<http://www.lawebdelprogramador.com>

<https://pypi.python.org/pypi/graphviz>

<http://graphviz.readthedocs.io/en/latest/manual.html>