INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

U.A: Teoría Computacional

“Reporte práctica 05”

Grupo: 2CV13

Profesor: De la O Torres Saul

Alumno: Sánchez Becerra Ernesto Daniel

Introducción:

¿Qué es un Autómata Finito No Determinista?

El AFND es un Finito que tiene transición, desde un estado puede tener múltiples destinos. Por eso se le llama no determinista.

Permite transiciones con cadenas vacías, no siempre permite el uso de backtracking y esté requiere menos espacio.

Un autómata finito no determinista (abreviado AFND) es un autómata finito que, a diferencia de los autómatas finitos deterministas (AFD), posee al menos un estado $q \in Q$, tal que para un símbolo $a \in \Sigma$ del alfabeto, existe más de una transición $\delta(q,a)$ posible.

Todo autómata finito no determinista puede ser convertido a autómata finito determinista. Pero no al contrario.
En un AFND puede darse cualquiera de estos dos casos:

- Que existan transiciones del tipo $\delta(q,a)=q1$ y $\delta(q,a)=q2$, siendo $q1 \neq q2$;
- Que existan transiciones del tipo $\delta(q, \varepsilon)$, siendo q un estado no-final, o bien un estado final pero con transiciones hacia otros estados.

Cuando se cumple el segundo caso, se dice que el autómata es un autómata finito no determinista con transiciones vacías o transiciones ε (abreviado AFND-ε). Estas transiciones permiten al autómata cambiar de estado sin procesar ningún símbolo de entrada. Considérese una modificación al modelo del autómata finito para permitirle ninguna, una o más transiciones de un estado sobre el mismo símbolo de entrada.

Entonces, para su funcionamiento La máquina comienza en el estado inicial especificado y lee una cadena de caracteres pertenecientes al alfabeto. El autómata utiliza la función de transición de estados T para determinar el siguiente estado, usando el estado actual y el símbolo que acaba de leer o la cadena vacía.

Sin embargo, "el estado siguiente de un AFND no solo depende del evento de entrada actual, sino que también en un número arbitrario de los eventos de entrada posterior. Hasta que se producen estos acontecimientos posteriores no es posible determinar en qué estado se encuentra la máquina".

Cuando el autómata ha terminado de leer, y se encuentra en un estado de aceptación, se dice que el AFND acepta la cadena, de lo contrario se dice que la cadena de caracteres es rechazada. Tanto para un AFND como para un autómata finito determinista (AFD) se puede aceptar el mismo lenguaje.

Por lo tanto, es posible convertir un AFND existente en un AFD para el desarrollo de una máquina tal vez más simple. Esto puede llevarse a cabo utilizando la construcción del conjunto potencia, que puede conducir a un aumento exponencial en el número de estados necesarios.

Objetivo:

Realizar un programa que sea capaz de construir un Autómata Finito No Determinista de acuerdo con la definición formal de este.

Desarrollo Experimental:

Para la elaboración de esta práctica se utilizó el entorno de desarrollo de programación “Visual Studio Code” pues este entorno acepta una gran cantidad de lenguajes de programación que para este caso específicamente se utilizó el lenguaje “Python”.

El programa consiste básicamente en elaborar un Autómata Finito No Determinista a partir de la teoría revisada en clase, entonces en este caso se ocupó una librería llamada “pandas” que entre otras cosas nos ayuda para realizar el manejo de valores o datos dentro de una matriz al momento de almacenar y mostrar estos en la salida de la consola.

Después, se le pide al usuario que proporcione los datos con los que se va a construir el autómata que son el número de estados que va a poseer y cuantos caminos o términos de transición va a tener el recorrido de nuestro autómata, cabe aclarar, que este programa permite numero de estados y caminos que el usuario quiera, pero se debe tener en cuenta que mientras más estados sean más transiciones va a haber y se volverá poco óptimo el programa.

Una vez proporcionados estos datos, se pedirá al usuario que ingrese el nombre de los estados que conforman al autómata ya sea poner algo como {q0, q1, q2.., qn} o {p0,p1,p2,pn} entre otras formas de nombrar a los estados.

Posteriormente, se pide al usuario que diga por cuál camino va a ir desde el último estado en el que se encuentra empezando por el estado con subíndice “0”. En el caso de este programa podemos establecer cuáles son los valores aceptados para realizar la transición de un estado a otro ya sea uno conformado por “a” y “b” o uno conformado por “0” y “1” o alguno otro que el usuario prefiera.

Finalmente, se pedirá que se especifique a partir del estado actual a qué otro estado vamos a pasar o bien podríamos definir que se avance al estado nulo con cualquiera de los dos valores de transición. Una vez que se establecieron estos valores se guardan los datos en la matriz y podremos mostrar la construcción del AFN en forma tabulada.

Nota: El código del programa vendrá anexo al final del documento, además de estar adjunto en la carpeta donde está el presente reporte. Hay que considerar que el programa al estar elaborado con Python se tendrá que instalar la librería pertinente para que no marque un error al ejecutarlo. Para ello se tendrá que aplicar el comando “pip install pandas” para que se descarguen los recursos de la librería y posteriormente para correr el programa bastará con poner el comando “Python programa05.py” y el programa empezará a correr.

Conclusión:

En conclusión, podemos decir que se comprendió de mejor forma la manera en que se define el AFND de manera teórica y la llevamos a la práctica donde lo más difícil tal vez fue entender la lógica que se necesitaba para obtener los datos y a partir de ellos empezar a trabajar armando la tabla de estados y transiciones para un mejor entendimiento de la información que se tiene.

Entonces, podemos decir que este tipo de autómata es una forma bastante óptima para el trato de información pues no requiere de un simple camino por el cual realizar una transición entre estados y esto es lo que en gran medida diferencia a ambos autómatas (AFD vs AFN) pues su estructura misma ayuda a diferenciarlos y ambos podrían abarcar un aplicación diferente cuando se esté trabajando con ellos.

Código de Programación:

```
import pandas as pd

nfa = {}
n = int (input("Ingrese el numero de estados: "))
t = int(input("Ingrese de caminos: "))

for i in range(n):
    state = input("Ingrese el nombre del estado: ")
    nfa[state] = {}
    for j in range(t):
        path = input("Introduce el camino a seguir: ")
        print("Ingrese el estado final desde el estado {} por el camino {}: " .format(state,path))
        reaching_state = [x for x in input().split()]
        nfa[state][path] = reaching_state

print("\nImpresion del AFN tabulado:")
nfa_table = pd.DataFrame(nfa)
print(nfa_table.transpose())
```