INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

U.A: Teoría Computacional

“Reporte práctica 03”

Grupo: 2CV13

Profesor: De la O Torres Saul

Alumno: Sánchez Becerra Ernesto Daniel

Introducción:

¿Qué es un Autómata Finito Determinista?

Un AFD tiene un conjunto finito de estados y un conjunto finito de símbolos de entrada. El término "determinista" hace referencia al hecho de que para cada entrada sólo existe uno y sólo un estado al que el autómata puede hacer la transición a partir de su estado actual. Un estado se diseña para que sea el estado inicial, y cero o más estados para que sean estados de aceptación. Una función de transición determina cómo cambia el estado cada vez que se procesa un símbolo de entrada.

Un autómata finito determinista consta de:

1. Un conjunto finito de estados, a menudo designado como Q.
2. Un conjunto finito de símbolos de entrada, a menudo designado como Σ
3. Una función de transición que toma como argumentos un estado y un símbolo de entrada y devuelve un estado. la función de transición se designa habitualmente como δ. En nuestra representación gráfica informal del autómata, δ se ha representa mediante arcos entre los estados y las etiquetas sobre los arcos. Si q es un estado y a es un símbolo de entrada, entonces δ(q,a) es el estado p tal que existe un arco etiquetado a que va desde q hasta p.
4. Un estado inicial, uno de los estados de Q.
5. Un conjunto de estados finales o de aceptación F. El conjunto F es un subconjunto de Q.

¿Qué es un Autómata Finito No Determinista?

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 ∈ Q, tal que para un símbolo a ∈ Σ del alfabeto, existe más de una transición δ(q,a) posible. Lo que se descubre tras que alguien haga mal muchos autómatas finitos y lo arreglan con un nombre bonito. 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 $\delta(q,a)=q_1$ y $\delta(q,a)=q_2$, siendo $q_1 \neq q_2$;
- Que existan transiciones del tipo δ(q, ε), 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.

Objetivo:

El alumno realizará un programa que pueda realizar un programa que transforme un AFN a un AFD.

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 hacer la transformación de un Autómata Finito No Determinista (AFN) a un Autómata Finito Determinista (AFD), para ello primero se construyó la estructura de el AFN donde establecimos los estados que tienen participación en la estructura del autómata y después realizar la transformación pertinente.

Entonces, primeramente, tenemos en nuestro código la importación de una librería de Python que permite agilizar la construcción de matrices de tamaño “mxn” que deberá estar establecido dentro de bucles anidados, pero la novedad con esta librería es que no necesita de hacer la construcción convencional pues facilita esto en gran medida.

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 aceptación va a tener el recorrido de nuestro autómata. Como es un programa que debe tener un modelo a seguir, hice uso de la siguiente tabla para la construcción del AFN:

	0	1
->p	p	{p,q}
q	r	-
r	s	-
s	t	-
*t	-	-

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 que 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 cuales 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 establezca.

Después, se pedirá que se especifique a partir del estado actual a que 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.

Finalmente, mediante el uso de un ciclo for se van a almacenar los datos que se recolectaron en la definición de el AFN en una variable llamada “var” donde la vamos a utilizar para acomodar los valores para la construcción de el AFD. Entonces, después de hacer la copia de datos en “var”, se hará uso de un ciclo while para ir sacando los valores de “var” e irlos acomodando a manera de tabla cada uno de los estados y transiciones, pero ahora también se le pide al usuario que se especifique cuál es el estado de aceptación en el AFD y finalmente vamos a mostrar a manera de tabla el AFD y mostramos cuál o cuáles son los estados de aceptación de nuestro nuevo autómata.

Nota: El programa vendrá anexo al final del documento y de igual forma se encontrará disponible de manera adjunta en la carpeta donde se encuentra este reporte.

Conclusión:

Con la elaboración de esta práctica se comprendió mejor el tema de la elaboración de un autómata a partir de otro o bien a realizar la conversión entre autómatas.

También se pusieron en práctica conocimientos antes adquiridos que engloban el funcionamiento de los autómatas por ejemplo el uso de estados de aceptación, la definición de un lenguaje con el que trabaja el autómata, un estado inicial y final, las cadenas que se pueden aceptar para que se tena transición en el autómata, entre otras cosas.

Entonces, podríamos decir que se siguió la forma convencional de obtener y realizar un autómata a partir de uno ya creado siguiendo en gran medida las reglas por las que se rige cada tipo de autómata pues es importante tener en cuenta que no son del mismo tipo.

Nota 2: Antes de correr el programa se debe de hacer la instalación de la librería “pandas” porque en caso contrario se mostrará un error al compilar el programa. Para ello se debe de poner el comando “pip install pandas” en el cmd para que se instale la librería y el programa funcione sin ningún problema.
Ya que se instaló la librería, simplemente se debe de correr el programa utilizando el comando “Python “programa04.py” y el programa empezaría a correr con normalidad.

Código del programa:

```python
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("\nImprimir el AFN tabulado:")
nfa_table = pd.DataFrame(nfa)
print(nfa_table.transpose())

print("Ingrese el estado de aceptacion del AFD: ")
nfa_final_state = [x for x in input().split()]

new_state_list = []
dfa = {}
keys_list = list(list(nfa.keys())[0])
path_list = list(nfa[keys_list[0]].keys())

#########################################
dfa[keys_list[0]] = {}
for y in range(t):
    var = "".join(nfa[keys_list[0]][path_list[y]])
    dfa[keys_list[0]][path_list[y]] = var
    if var not in keys_list:
        new_state_list.append(var)
        keys_list.append(var)

#########################################
while len(new_state_list) != 0:
    dfa[new_state_list[0]] = {}
    for _ in range(len(new_state_list[0])):
        for i in range(len(path_list)):
            temp = []
            for j in range(len(new_state_list[0])):
                temp += nfa[new_state_list[0][j]][path_list[i]]
            s = ""
            s = s.join(temp)
            if s not in keys_list:
                new_state_list.append(s)
                keys_list.append(s)
            dfa[new_state_list[0]][path_list[i]] = s

    new_state_list.remove(new_state_list[0])

#################################################
print("\nImprimir AFD tabulado")
dfa_table = pd.DataFrame(dfa)
print(dfa_table.transpose())

dfa_state_list = list(dfa.keys())
dfa_final_states = []
for x in dfa_state_list:
    for i in x:
        if i in nfa_final_state:
            dfa_final_states.append(x)
            break
print("\nEstados de aceptacion del AFD nuevo: ", dfa_final_states)
```