Aplicación de un AFD

Para la resolución de problemas en la vida real

Cuatrimestre	7°
Grupo	B
Apellidos	Flores Nataren
Nombre(s)	Kevin Daniel
Matrícula	203418
Corte	1
Actividad	1 – recuperación
Repositorio Github	https://github.com/203418/lenguajes-y-automatas.git

Introducción:

Un autómata finito determinista (AFD) se puede definir como un autómata reconocedor, el cual se limita a aceptar o no una determinada cadena recibida en la entrada, por lo tanto, se puede decir que la salida de los mismos solo tendrá dos valores posibles: aceptar o no la cadena de entrada.

Al igual que en las máquinas secuenciales, estos autómatas transitarán entre un conjunto finito de estados posibles, a medida que reciban sucesivamente los caracteres de entrada, en un instante determinado de tiempo el autómata solo podrá estar en uno y solo uno de los estados posibles.

Los autómatas finitos deterministas quedarán formalmente definido mediante una quíntupla como sigue:

$$AFD = (\sum , Q, q_0, F, f)$$

Dónde:

$\sum$	Alfabeto de símbolos de entrada.
Q	Conjunto finito de estados
q_0	$q_0 \in Q$ – estado inicial previsto
F	$F \subseteq Q$ - es el conjunto de estado finales de aceptación.
f	Función de transición de estados definida como $f: Q \times \sum \longrightarrow Q$

Este autómata recibirá una cadena en la cinta de entrada e ira procesando de uno a la vez los símbolos de entrada. Comenzará su funcionamiento posicionado en el estado inicial, y desde este estado comenzará su ejecución.

En cada intervalo de tiempo discreto realizará dos acciones las cuales serán consideradas como acciones indivisibles en un determinado intervalo de tiempo.

Las acciones que realiza son:

- Leer el próximo símbolo, desde la cinta de entrada.
- Transitar, cambiar de estado

El autómata finito determinista realizará transiciones de estados a través de la función f solo cuando reciba un símbolo de entrada. Esto puede generalizarse a una palabra completa, o cuando reciba la palabra vacía, en este caso se denominará una función de transición f´ como la función

Problemática

Una estructura switch permite considerar decisiones para más de dos posibilidades en el cual la variable se evalúa y compara sucesivamente con todas las constantes (numéricas) que aparecen junto a la palabra reservada case. Si alguno de ellos es igual al valor de la variable, se ejecuta la sentencia dentro de ese case. Si no aparece la palabra reservada break, continua con la comparación con el resto de las opciones. Si aparece break, se termina la ejecución de la estructura switch. La opción default, es opcional e indica la sentencia que se ejecuta en caso de que el valor de la variable no se corresponda con ninguna de las constantes expresadas.

Los compiladores hoy en día son capaces de verificar si esta estructura está bien definida o no en cuestión de la sintaxis proporcionada para lo cual se basan en autómatas y otras herramientas para validar detalles del mismo.

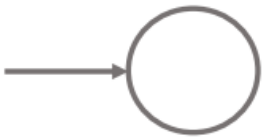
Se desea realizar un autómata capaz de validar la sentencia switch con base un archivo ".txt" proporcionado, dónde se evaluará:

- El número de casos existentes uno por uno.
- El contenido de cada case de manera independiente.
- La asignación de variables de tipo entero y flotante
- El nombramiento de cada variable siguiendo la estructura lower camel case.

El autómata debe ser capaz de verificar una estructura switch ingresada cumpla con las reglas básicas de un switch en el lenguaje JAVA como, por ejemplo:

```
switch (a) {
    case 0:
    case 1:
        int variable2=2;
        int variable3=2;
    break;
    default:
        int vari=132;
        float vares=12.232;
    break;
}
```

Y la simbología a aplicar es la siguiente:



Estado Inicial

Estado Estándar

Estado Final

Modelado

A continuación, se presenta el modelado utilizado para darle solución a la problemática expuesta anteriormente, dónde se puede observar que nuestro autómata recibe como entrada nuestro switch con su palabra reservada, paréntesis contenedores de la variable clave, corchete de apertura, contenido (varios case y un default) hasta llegar al corchete de cierre.

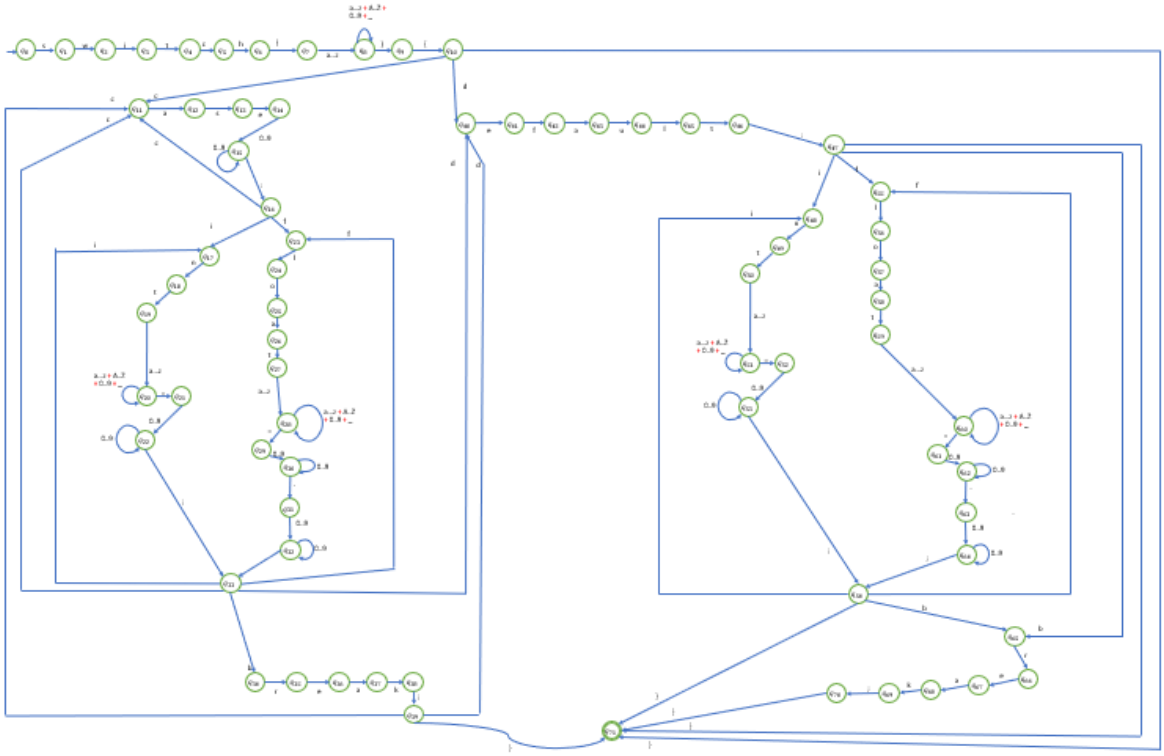


Ilustración 1 Modelado AFD

*Si no se alcanza a apreciar de manera correcta, puedes ver el autómata en la siguiente liga:

https://drive.google.com/file/d/1lD4Pvn3NN7_J3ALZJGUoXJX7_7K1W7GT/view?usp=sharing

De todos modos, se incluirá en los archivos del proyecto.

Descripción del proyecto:

Σ: {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9','_',':',';','(',')','{','}','=','.'}

Q:{'Q0','Q1','Q2','Q3','Q4','Q5','Q6','Q7','Q8','Q9','Q10','Q11','Q12','Q13','Q14','Q15','Q16','Q17','Q18','Q19','Q20','Q21','Q22','Q23','Q24','Q25','Q26','Q27','Q28','Q29','Q30','Q31','Q32','Q33','Q34','Q35','Q36','Q37','Q38','Q39','Q40','Q41','Q42','Q43','Q44','Q45','Q46','Q47','Q48','Q49','Q50','Q51'

,'Q52','Q53','Q54','Q55','Q56','Q57','Q58','Q59','Q60','Q61','Q62','Q63','Q64','Q65','Q66','Q67','Q6
8','Q69','Q70','Q71'}

q0: {Q0}

F:{Q71}

δ: { ('Q0','s'):'Q1',('Q1','w'):'Q2',('Q2','i'):'Q3',('Q3','t'):'Q4',('Q4','c'):'Q5',('Q5','h'):'Q6',('Q6','('):'Q7',
('Q7','a'):'Q8',('Q7','b'):'Q8',('Q7','c'):'Q8',('Q7','d'):'Q8',('Q7','e'):'Q8',('Q7','f'):'Q8',('Q7','g'):'Q8',
('Q7','h'):'Q8',('Q7','i'):'Q8',('Q7','j'):'Q8',('Q7','k'):'Q8',('Q7','l'):'Q8',('Q7','m'):'Q8',('Q7','n'):'Q8',
('Q7','o'):'Q8',('Q7','p'):'Q8',('Q7','q'):'Q8',('Q7','r'):'Q8',('Q7','s'):'Q8',('Q7','t'):'Q8',('Q7','u'):'Q8',
('Q7','v'):'Q8',('Q7','w'):'Q8',('Q7','x'):'Q8',('Q7','y'):'Q8',('Q7','z'):'Q8',('Q8','a'):'Q8',('Q8','b'):'Q8',
('Q8','c'):'Q8',('Q8','d'):'Q8',('Q8','e'):'Q8',('Q8','f'):'Q8',('Q8','g'):'Q8',('Q8','h'):'Q8',('Q8','i'):'Q8',
('Q8','j'):'Q8',('Q8','k'):'Q8',('Q8','l'):'Q8',('Q8','m'):'Q8',('Q8','n'):'Q8',('Q8','o'):'Q8',('Q8','p'):'Q8',
('Q8','q'):'Q8',('Q8','r'):'Q8',('Q8','s'):'Q8',('Q8','t'):'Q8',('Q8','u'):'Q8',('Q8','v'):'Q8',('Q8','w'):'Q8',
('Q8','x'):'Q8',('Q8','y'):'Q8',('Q8','z'):'Q8',('Q8','A'):'Q8',('Q8','B'):'Q8',('Q8','C'):'Q8',('Q8','D'):'Q8',
('Q8','E'):'Q8',('Q8','F'):'Q8',('Q8','G'):'Q8',('Q8','H'):'Q8',('Q8','I'):'Q8',('Q8','J'):'Q8',('Q8','K'):'Q8',
('Q8','L'):'Q8',('Q8','M'):'Q8',('Q8','N'):'Q8',('Q8','O'):'Q8',('Q8','P'):'Q8',('Q8','Q'):'Q8', ('Q8','R'):'Q8',
('Q8','S'):'Q8',('Q8','T'):'Q8',('Q8','U'):'Q8',('Q8','V'):'Q8',('Q8','W'):'Q8',('Q8','X'):'Q8',('Q8','Y'):'Q8',
('Q8','Z'):'Q8',('Q8','0'):'Q8',('Q8','1'):'Q8',('Q8','2'):'Q8',('Q8','3'):'Q8',('Q8','4'):'Q8',('Q8','5'):'Q8',
('Q8','6'):'Q8',('Q8','7'):'Q8',('Q8','8'):'Q8',('Q8','9'):'Q8',('Q8','_'):'Q8',('Q8',')'):'Q9',('Q9','{'):'Q10',
('Q10','c'):'Q11',('Q10','d'):'Q40',('Q10','}'):'Q71',('Q11','a'):'Q12',('Q12','s'):'Q13',('Q13','e'):'Q14',
('Q14','0'):'Q15',('Q14','1'):'Q15',('Q14','2'):'Q15',('Q14','3'):'Q15',('Q14','4'):'Q15',('Q14','5'):'Q15',
('Q14','6'):'Q15',('Q14','7'):'Q15',('Q14','8'):'Q15',('Q14','9'):'Q15',('Q15','0'):'Q15',('Q15','1'):'Q15',
('Q15','2'):'Q15',('Q15','3'):'Q15',('Q15','4'):'Q15',('Q15','5'):'Q15',('Q15','6'):'Q15',('Q15','7'):'Q15',
('Q15','8'):'Q15',('Q15','9'):'Q15',('Q15',':'):'Q16',('Q16','c'):'Q11',('Q16','i'):'Q17',('Q16','f'):'Q23',
('Q17','n'):'Q18',('Q18','t'):'Q19',('Q19','a'):'Q20',('Q19','b'):'Q20',('Q19','c'):'Q20',('Q19','d'):'Q20',
('Q19','e'):'Q20',('Q19','f'):'Q20',('Q19','g'):'Q20',('Q19','h'):'Q20',('Q19','i'):'Q20',('Q19','j'):'Q20',
('Q19','k'):'Q20',('Q19','l'):'Q20',('Q19','m'):'Q20',('Q19','n'):'Q20',('Q19','o'):'Q20',('Q19','p'):'Q20',
('Q19','q'):'Q20',('Q19','r'):'Q20',('Q19','s'):'Q20',('Q19','t'):'Q20',('Q19','u'):'Q20',('Q19','v'):'Q20',
('Q19','w'):'Q20',('Q19','x'):'Q20',('Q19','y'):'Q20',('Q19','z'):'Q20',('Q20','a'):'Q20',('Q20','b'):'Q20',
('Q20','c'):'Q20',('Q20','d'):'Q20',('Q20','e'):'Q20',('Q20','f'):'Q20',('Q20','g'):'Q20',('Q20','h'):'Q20',
('Q20','i'):'Q20',('Q20','j'):'Q20',('Q20','k'):'Q20',('Q20','l'):'Q20',('Q20','m'):'Q20',('Q20','n'):'Q20',
('Q20','o'):'Q20',('Q20','p'):'Q20',('Q20','q'):'Q20',('Q20','r'):'Q20',('Q20','s'):'Q20',('Q20','t'):'Q20',
('Q20','u'):'Q20',('Q20','v'):'Q20',('Q20','w'):'Q20',('Q20','x'):'Q20',('Q20','y'):'Q20',('Q20','z'):'Q20',
('Q20','A'):'Q20',('Q20','B'):'Q20',('Q20','C'):'Q20',('Q20','D'):'Q20',('Q20','E'):'Q20',('Q20','F'):'Q20',
('Q20','G'):'Q20',('Q20','H'):'Q20',('Q20','I'):'Q20',('Q20','J'):'Q20',('Q20','K'):'Q20',('Q20','L'):'Q20',
('Q20','M'):'Q20',('Q20','N'):'Q20',('Q20','O'):'Q20',('Q20','P'):'Q20',('Q20','Q'):'Q20',('Q20','R'):'Q20'
('Q20','S'):'Q20',('Q20','T'):'Q20',('Q20','U'):'Q20',('Q20','V'):'Q20',('Q20','W'):'Q20',('Q20','X'):'Q20',
('Q20','Y'):'Q20',('Q20','Z'):'Q20',('Q20','0'):'Q20',('Q20','1'):'Q20',('Q20','2'):'Q20',('Q20','3'):'Q20',
('Q20','4'):'Q20',('Q20','5'):'Q20',('Q20','6'):'Q20',('Q20','7'):'Q20',('Q20','8'):'Q20',('Q20','9'):'Q20',
('Q20','_'):'Q20',('Q20','='):'Q21',('Q21','0'):'Q22',('Q21','1'):'Q22',('Q21','2'):'Q22',('Q21','3'):'Q22',
('Q21','4'):'Q22',('Q21','5'):'Q22',('Q21','6'):'Q22',('Q21','7'):'Q22',('Q21','8'):'Q22',('Q21','9'):'Q22',
('Q22','0'):'Q22',('Q22','1'):'Q22',('Q22','2'):'Q22',('Q22','3'):'Q22',('Q22','4'):'Q22',('Q22','5'):'Q22',
('Q22','6'):'Q22',('Q22','7'):'Q22',('Q22','8'):'Q22',('Q22','9'):'Q22',('Q22',';'):'Q33',('Q23','l'):'Q24',
('Q24','o'):'Q25',('Q25','a'):'Q26',('Q26','t'):'Q27',('Q27','a'):'Q28',('Q27','b'):'Q28',('Q27','c'):'Q28',

('Q27','d'):'Q28',('Q27','e'):'Q28',('Q27','f'):'Q28',('Q27','g'):'Q28',('Q27','h'):'Q28',('Q27','i'):'Q28',
('Q27','j'):'Q28',('Q27','k'):'Q28',('Q27','l'):'Q28',('Q27','m'):'Q28',('Q27','n'):'Q28',('Q27','o'):'Q28',
('Q27','p'):'Q28',('Q27','q'):'Q28',('Q27','r'):'Q28',('Q27','s'):'Q28',('Q27','t'):'Q28',('Q27','u'):'Q28',
('Q27','v'):'Q28',('Q27','w'):'Q28',('Q27','x'):'Q28',('Q27','y'):'Q28',('Q27','z'):'Q28',('Q28','a'):'Q28',
('Q28','b'):'Q28',('Q28','c'):'Q28',('Q28','d'):'Q28',('Q28','e'):'Q28',('Q28','f'):'Q28',('Q28','g'):'Q28'
('Q28','h'):'Q28',('Q28','i'):'Q28',('Q28','j'):'Q28',('Q28','k'):'Q28',('Q28','l'):'Q28',('Q28','m'):'Q28',
('Q28','n'):'Q28',('Q28','o'):'Q28',('Q28','p'):'Q28',('Q28','q'):'Q28',('Q28','r'):'Q28',('Q28','s'):'Q28',
('Q28','t'):'Q28',('Q28','u'):'Q28',('Q28','v'):'Q28',('Q28','w'):'Q28',('Q28','x'):'Q28',('Q28','y'):'Q28',
('Q28','z'):'Q28',('Q28','A'):'Q28',('Q28','B'):'Q28',('Q28','C'):'Q28',('Q28','D'):'Q28',('Q28','E'):'Q28',
('Q28','F'):'Q28',('Q28','G'):'Q28',('Q28','H'):'Q28',('Q28','I'):'Q28',('Q28','J'):'Q28',('Q28','K'):'Q28',
('Q28','L'):'Q28',('Q28','M'):'Q28',('Q28','N'):'Q28',('Q28','O'):'Q28',('Q28','P'):'Q28',('Q28','Q'):'Q28'
('Q28','R'):'Q28',('Q28','S'):'Q28',('Q28','T'):'Q28',('Q28','U'):'Q28',('Q28','V'):'Q28',('Q28','W'):'Q28',
('Q28','X'):'Q28',('Q28','Y'):'Q28', ('Q28','Z'):'Q28',('Q28','0'):'Q28',('Q28','1'):'Q28',('Q28','2'):'Q28',
('Q28','3'):'Q28',('Q28','4'):'Q28',('Q28','5'):'Q28',('Q28','6'):'Q28',('Q28','7'):'Q28',('Q28','8'):'Q28',
('Q28','9'):'Q28',('Q28','_'):'Q28',('Q28','='):'Q29',('Q29','0'):'Q30',('Q29','1'):'Q30',('Q29','2'):'Q30',
('Q29','3'):'Q30',('Q29','4'):'Q30',('Q29','5'):'Q30',('Q29','6'):'Q30',('Q29','7'):'Q30',('Q29','8'):'Q30',
('Q29','9'):'Q30',('Q30','0'):'Q30',('Q30','1'):'Q30',('Q30','2'):'Q30',('Q30','3'):'Q30',('Q30','4'):'Q30',
('Q30','5'):'Q30',('Q30','6'):'Q30',('Q30','7'):'Q30',('Q30','8'):'Q30',('Q30','9'):'Q30',('Q30','.'):'Q31',
('Q31','0'):'Q32',('Q31','1'):'Q32',('Q31','2'):'Q32',('Q31','3'):'Q32',('Q31','4'):'Q32',('Q31','5'):'Q32',
('Q31','6'):'Q32',('Q31','7'):'Q32',('Q31','8'):'Q32',('Q31','9'):'Q32',('Q32','0'):'Q32',('Q32','1'):'Q32',
('Q32','2'):'Q32',('Q32','3'):'Q32',('Q32','4'):'Q32',('Q32','5'):'Q32',('Q32','6'):'Q32',('Q32','7'):'Q32',
('Q32','8'):'Q32',('Q32','9'):'Q32',('Q32',';'):'Q33',('Q33','c'):'Q11',('Q33','i'):'Q17',('Q33','f'):'Q23',
('Q33','d'):'Q40',('Q33','b'):'Q34',('Q34','r'):'Q35',('Q35','e'):'Q36',('Q36','a'):'Q37',('Q37','k'):'Q38',
('Q38',';'):'Q39',('Q39','d'):'Q40',('Q39','c'):'Q11',('Q39','}'):'Q71',('Q40','e'):'Q41',('Q41','f'):'Q42',
('Q42','a'):'Q43',('Q43','u'):'Q44',('Q44','l'):'Q45',('Q45','t'):'Q46',('Q46',':'):'Q47',('Q47','i'):'Q48',
('Q47','f'):'Q55',('Q47','b'):'Q65',('Q47','}'):'Q71',('Q48','n'):'Q49',('Q49','t'):'Q50',('Q50','a'):'Q51',
('Q50','b'):'Q51',('Q50','c'):'Q51',('Q50','d'):'Q51',('Q50','e'):'Q51',('Q50','f'):'Q51',('Q50','g'):'Q51',
('Q50','h'):'Q51',('Q50','i'):'Q51',('Q50','j'):'Q51',('Q50','k'):'Q51',('Q50','l'):'Q51',('Q50','m'):'Q51',
('Q50','n'):'Q51',('Q50','o'):'Q51',('Q50','p'):'Q51',('Q50','q'):'Q51',('Q50','r'):'Q51',('Q50','s'):'Q51',
('Q50','t'):'Q51',('Q50','u'):'Q51',('Q50','v'):'Q51',('Q50','w'):'Q51',('Q50','x'):'Q51',('Q50','y'):'Q51',
('Q50','z'):'Q51',('Q51','a'):'Q51',('Q51','b'):'Q51',('Q51','c'):'Q51',('Q51','d'):'Q51',('Q51','e'):'Q51',
('Q51','f'):'Q51',('Q51','g'):'Q51',('Q51','h'):'Q51',('Q51','i'):'Q51',('Q51','j'):'Q51',('Q51','k'):'Q51',
('Q51','l'):'Q51',('Q51','m'):'Q51',('Q51','n'):'Q51',('Q51','o'):'Q51',('Q51','p'):'Q51',('Q51','q'):'Q51',
('Q51','r'):'Q51',('Q51','s'):'Q51',('Q51','t'):'Q51',('Q51','u'):'Q51',('Q51','v'):'Q51',('Q51','w'):'Q51',
('Q51','x'):'Q51',('Q51','y'):'Q51',('Q51','z'):'Q51',('Q51','A'):'Q51',('Q51','B'):'Q51',('Q51','C'):'Q51',
('Q51','D'):'Q51',('Q51','E'):'Q51',('Q51','F'):'Q51',('Q51','G'):'Q51',('Q51','H'):'Q51',('Q51','I'):'Q51',
('Q51','J'):'Q51',('Q51','K'):'Q51',('Q51','L'):'Q51',('Q51','M'):'Q51',('Q51','N'):'Q51',('Q51','O'):'Q51',
('Q51','P'):'Q51',('Q51','Q'):'Q51',('Q51','R'):'Q51',('Q51','S'):'Q51',('Q51','T'):'Q51', ('Q51','U'):'Q51',
('Q51','V'):'Q51',('Q51','W'):'Q51',('Q51','X'):'Q51',('Q51','Y'):'Q51',('Q51','Z'):'Q51',('Q51','0'):'Q51',
('Q51','1'):'Q51',('Q51','2'):'Q51',('Q51','3'):'Q51',('Q51','4'):'Q51',('Q51','5'):'Q51',('Q51','6'):'Q51',
('Q51','7'):'Q51',('Q51','8'):'Q51',('Q51','9'):'Q51',('Q51','_'):'Q51',('Q51','='):'Q52',('Q52','0'):'Q53',
('Q52','1'):'Q53',('Q52','2'):'Q53',('Q52','3'):'Q53',('Q52','4'):'Q53',('Q52','5'):'Q53',('Q52','6'):'Q53',
('Q52','7'):'Q53',('Q52','8'):'Q53',('Q52','9'):'Q53',('Q53','0'):'Q53',('Q53','1'):'Q53',('Q53','2'):'Q53',
('Q53','3'):'Q53',('Q53','4'):'Q53',('Q53','5'):'Q53',('Q53','6'):'Q53',('Q53','7'):'Q53',('Q53','8'):'Q53',

('Q53','9'):'Q53',('Q53',';'):'Q54',('Q54','i'):'Q48',('Q54','f'):'Q55',('Q54','b'):'Q65',('Q54','}'):'Q71',
('Q55','l'):'Q56',('Q56','o'):'Q57',('Q57','a'):'Q58',('Q58','t'):'Q59',('Q59','a'):'Q60',('Q59','b'):'Q60',
('Q59','c'):'Q60',('Q59','d'):'Q60',('Q59','e'):'Q60',('Q59','f'):'Q60',('Q59','g'):'Q60',('Q59','h'):'Q60',
('Q59','i'):'Q60',('Q59','j'):'Q60',('Q59','k'):'Q60',('Q59','l'):'Q60',('Q59','m'):'Q60',('Q59','n'):'Q60',
('Q59','o'):'Q60',('Q59','p'):'Q60',('Q59','q'):'Q60',('Q59','r'):'Q60',('Q59','s'):'Q60',('Q59','t'):'Q60',
('Q59','u'):'Q60',('Q59','v'):'Q60',('Q59','w'):'Q60',('Q59','x'):'Q60',('Q59','y'):'Q60',('Q59','z'):'Q60',
('Q60','a'):'Q60',('Q60','b'):'Q60',('Q60','c'):'Q60',('Q60','d'):'Q60',('Q60','e'):'Q60',('Q60','f'):'Q60',
('Q60','g'):'Q60',('Q60','h'):'Q60',('Q60','i'):'Q60',('Q60','j'):'Q60',('Q60','k'):'Q60',('Q60','l'):'Q60',
('Q60','m'):'Q60',('Q60','n'):'Q60',('Q60','o'):'Q60',('Q60','p'):'Q60',('Q60','q'):'Q60',('Q60','r'):'Q60',
('Q60','s'):'Q60',('Q60','t'):'Q60',('Q60','u'):'Q60',('Q60','v'):'Q60',('Q60','w'):'Q60',('Q60','x'):'Q60',
('Q60','y'):'Q60',('Q60','z'):'Q60',('Q60','A'):'Q60',('Q60','B'):'Q60',('Q60','C'):'Q60',('Q60','D'):'Q60',
('Q60','E'):'Q60',('Q60','F'):'Q60',('Q60','G'):'Q60',('Q60','H'):'Q60',('Q60','I'):'Q60',('Q60','J'):'Q60',
('Q60','K'):'Q60',('Q60','L'):'Q60',('Q60','M'):'Q60',('Q60','N'):'Q60',('Q60','O'):'Q60',('Q60','P'):'Q60',
('Q60','Q'):'Q60',('Q60','R'):'Q60',('Q60','S'):'Q60',('Q60','T'):'Q60',('Q60','U'):'Q60',('Q60','V'):'Q60',
('Q60','W'):'Q60',('Q60','X'):'Q60',('Q60','Y'):'Q60',('Q60','Z'):'Q60',('Q60','0'):'Q60',('Q60','1'):'Q60',
('Q60','2'):'Q60',('Q60','3'):'Q60',('Q60','4'):'Q60',('Q60','5'):'Q60',('Q60','6'):'Q60',('Q60','7'):'Q60',
('Q60','8'):'Q60',('Q60','9'):'Q60',('Q60','_'):'Q60',('Q60','='):'Q61',('Q61','0'):'Q62',('Q61','1'):'Q62',
('Q61','2'):'Q62',('Q61','3'):'Q62',('Q61','4'):'Q62',('Q61','5'):'Q62',('Q61','6'):'Q62',('Q61','7'):'Q62',
('Q61','8'):'Q62',('Q61','9'):'Q62',('Q62','0'):'Q62',('Q62','1'):'Q62',('Q62','2'):'Q62',('Q62','3'):'Q62',
('Q62','4'):'Q62',('Q62','5'):'Q62',('Q62','6'):'Q62',('Q62','7'):'Q62',('Q62','8'):'Q62',('Q62','9'):'Q62',
('Q62','.'):'Q63',('Q63','0'):'Q64',('Q63','1'):'Q64',('Q63','2'):'Q64',('Q63','3'):'Q64',('Q63','4'):'Q64',
('Q63','5'):'Q64',('Q63','6'):'Q64',('Q63','7'):'Q64',('Q63','8'):'Q64',('Q63','9'):'Q64',('Q64','0'):'Q64',
('Q64','1'):'Q64',('Q64','2'):'Q64',('Q64','3'):'Q64',('Q64','4'):'Q64',('Q64','5'):'Q64',('Q64','6'):'Q64',
('Q64','7'):'Q64',('Q64','8'):'Q64',('Q64','9'):'Q64',('Q64',';'):'Q54',('Q65','r'):'Q66',('Q66','e'):'Q67',
('Q67','a'):'Q68',('Q68','k'):'Q69',('Q69',';'):'Q70',('Q70','}'):'Q71'}

Implementación

Para el desarrollo de este proyecto, se utilizó Python como lenguaje de programación en su versión 3.10.7 debido a los requerimientos del aplicativo, así como la facilidad de uso en sus métodos y propiedades. En un principio se desarrolló el aplicativo para mostrarse en consola para que, posteriormente fuera migrado a una aplicación gráfica hecha en QT Designer en su versión 5.12.5 (framework de Python para ambiente gráfico) con el fin de ser más amigable para la vista y porque así fue requerido.

El programa funciona con dos entradas diferentes en formato “.txt”, uno de ellos llamado “autómata.txt” que contiene la estructura del AFD (todos los estados posibles, alfabeto, estado inicial y estado final), este archivo es colocado de manera estática en la programación para especificar que solo ese autómata es aceptado (con la posibilidad de modificarse para aceptar cualquier autómata si se requiere en algún futuro) y leído por un método que lo convierte en un diccionario y así, poder ser usado de manera más sencilla. El otro archivo es el switch a testear, el cual puede ser nombrado de cualquier manera (con la extensión ”.txt”) el cual es seleccionado mediante un botón; este contiene la estructura

switch a evaluar con el autómata previamente leído de forma estática y así saber si es o no, una estructura switch válida.

Respecto al algoritmo utilizado para evaluar las cadenas, se crearon diversas funciones que cumplen con una determinada tarea, como lo es el recorrido del AFD, encontrar si un carácter de la cadena forma parte de nuestro alfabeto o si cumple con la transición hacia un estado determinado y finalmente valida si las cadenas cumplen con esas condiciones.

Al leer el archivo que contiene la estructura switch, este quita los espacios y deja todos los caracteres en una sola línea para que se lleve a cabo todo el proceso anterior, si la cadena es aceptada por el programa (si cumple con el autómata), cumple con la estructura del switch y se muestra en la línea de resultados del widget de lista en QT Designer.

```
import ast
from PyQt5 import QtWidgets, uic
import sys

class Main:
    D = ""
    testing_switch = ""
    results_list = []

    def read_file(self, filename):
        try:
            with open(filename, 'r', encoding='utf-8') as file:
                if "automata" in filename:
                    content = file.read()
                    AFD = ast.literal_eval(content)
                    self.D = AFD
                    return True
                content = file.read()
                cadena = ""
                switch = []
                for c in content:
                    if c != '\n' and c != ' ':
                        cadena = cadena + c
                switch.append(cadena)
                return switch
        except Exception as e:
            print(e)
            print(f"El archivo '{filename}' no existe.")

    def open_txt_test(self, interfaz):
      filename = QtWidgets.QFileDialog.getOpenFileName(None, "Abrir archivo",
      self.testing_switchs = self.read_file(filename[0])
      self.evaluate(interfaz)
```

Ilustración 2 Funciones de lectura

```
def step_afd(self,D,q,a):
  try:
     assert(a in D["Sigma"])
     assert(q in D["Q"])
     return D["Delta"][(q,a)]
  except:
     return False

def run_afd(self,D,w):
  curstate = D["q0"]
  if w == "":
     return curstate
  else:
     return self.run_afd_h(D,w[1:], self.step_afd(D,curstate,w[0]))

def run_afd_h(self,D,w,q):
  if w == "":
     return q
  else:
     return self.run_afd_h(D,w[1:], self.step_afd(D,q,w[0]))

def accepts_afd(self,D,w):
  return self.run_afd(D,w) in D["F"]

def evaluate(self,interfaz):
  for character in self.testing_switchs:
     if self.accepts_afd(self.D, character):
        self.results_list.append(f"La sentencia: '{character}' es aceptada
     else:
        self.results_list.append(f"La sentencia: ' {character} es rechazad
  interfaz.list_result.addItems(self.results_list)
```

Ilustración 3 Algoritmo de recorrido de AFD y búsqueda de MAC

```
if __name__ == '__main__' :
   app = QtWidgets.QApplication(sys.argv)

   main = Main()

   main.read_file('automata.txt')

   interfaz = uic.loadUi("interfaz.ui")
   interfaz.show()


   interfaz.select_test.clicked.connect(lambda: main.open_txt_test(interfaz))

   sys.exit(app.exec_())
```

Ilustración 4 Llamada a las funciones anteriores

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Dialog</class>
 <widget class="QDialog" name="Dialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>524</width>
    <height>381</height>
   </rect>
  </property>
  <property name="baseSize">
   <size>
    <width>50</width>
    <height>50</height>
   </size>
  </property>
  <property name="windowTitle">
   <string>AFD</string>
  </property>
  <property name="sizeGripEnabled">
   <bool>false</bool>
  </property>
  <widget class="QLabel" name="label">
   <property name="geometry">
    <rect>
     <x>120</x>
     <y>20</y>
     <width>300</width>
     <height>41</height>
    </rect>
   </property>
   <property name="font">
    <font>
     <family>Arial Rounded MT Bold</family>
     <pointsize>14</pointsize>
    </font>
   </property>
   <property name="text">
    <string>Autómata Finito Determinista</string>
   </property>
   <property name="alignment">
    <set>Qt::AlignCenter</set>
   </property>
  </widget>
```

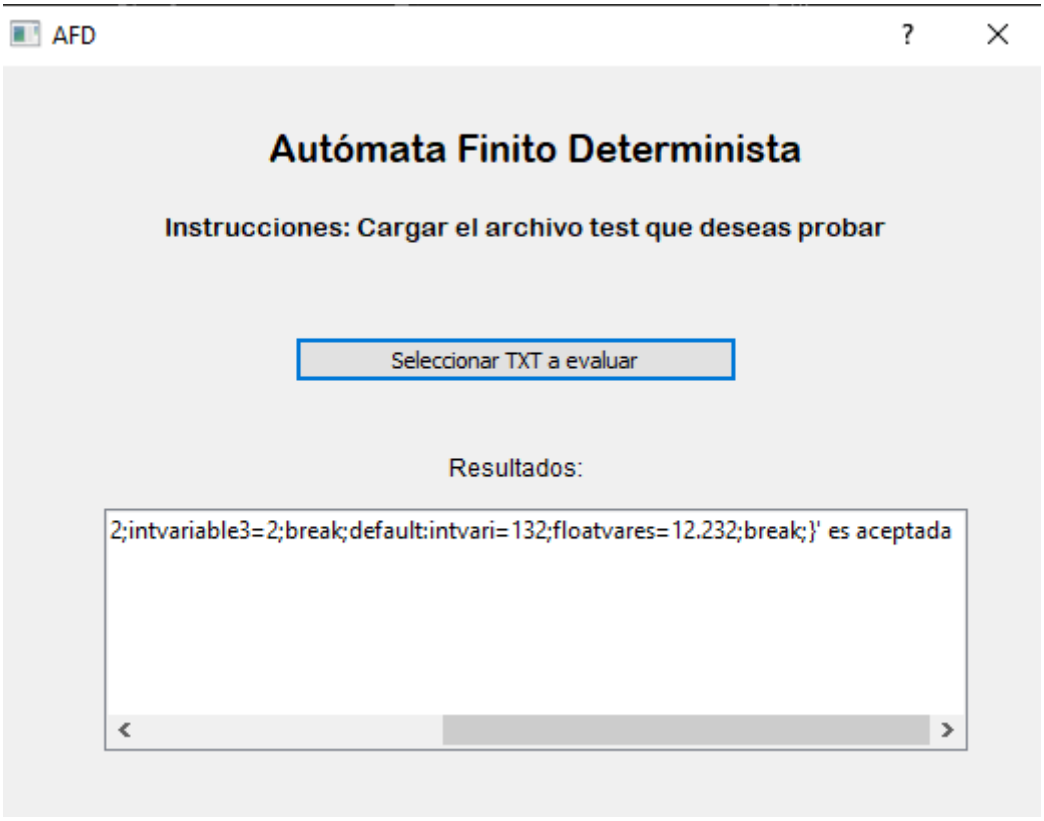
Ilustración 5 Diseño de la interfaz

Resultados

En un principio era el usuario quien ingresaba las estructuras switch a probar, después de que se implementara la función para leer el archivo .txt, el programa lee las estructuras desde esos archivos y las valida según el autómata leído del archivo "automata.txt". Las primeras pruebas, que se realizaron mediante consola, arrojaron los siguientes resultados:

```
kevin flores@DESKTOP-C2PU0AR MINGW64 /d/Claro Drive/Universidad Politécnica de Chiapas/7° Cuatrimestre/Lenguajes y autómatas/Actividades/Autómata/Archivos
$ python main.py
La sentencia: 'switch(a){case0:case1:intvariable2=2;intvariable3=2;break;default:intvari=132;floatvares=12.232;break;}' es aceptada
```

En consola obteniamos el estado de validación, ya sea RECHAZADO o ACEPTADO, Posteriormente se implementó la interfaz dónde ya podías seleccionar el archivo a testear y los resultados se muestran de la siguiente manera:



Discusión

Durante todo el proceso de desarrollo del programa y su ejecución se observó el comportamiento del autómata que se modeló, que consistía en verificar y validar que una cadena ingresada sea correcta según la estructura de un switch en JAVA 11. Al principio se hicieron se tuvo una confusión debido a todas las validaciones a implementar pero, una vez solventada esa parte, el proceso fue de yendo de manera más rápida.

Debido a que se basó en el libro que referenciamos y al uso del lenguaje Python, fue bastante sencillo el desarrollo de nuestro autómata. También a los conceptos ya definidos y el apoyo de los distintos, así como la designación del autómata como uno finito determinista, fue posible crear el grafo que sostendría el funcionamiento del autómata para este caso, lo cual hizo más simple el desarrollo del código.

Gracias a que ya se conocía la herramienta para diseñar interfaces gráficas en Python, omitimos la investigación sobre el uso de estás. En un principio se deseaba utilizar Tkinter para el desarrollo de la interfaz, pero a causa de complicaciones al implementarlo con lo que ya se tenía provisto, se decidió usar QT Designer, siendo esta una herramienta más sencilla de usar y de agregar a lo ya funcional.

Conclusión

Como conclusión de esta actividad se obtiene una definición más clara de lo que es un autómata y su uso; así como el funcionamiento de una estructura switch más a fondo. Gracias a la información que se recaudó, fue posible implementar un autómata y así, desarrollar un sistema que reconozca la estructura de un switch.

Aunado a lo anterior, se reforzó el conocimiento sobre el lenguaje de Python y las herramientas, así como la experiencia adquirida en el uso e implementación de autómatas en código, el uso e importancia de las tablas de transición al crear y recorrer un autómata y la implementación de una interfaz gráfica con QT Designer.

Referencias

Gopalakrishnan, G. (2019). *Automata and computability: programmer's perspective.* Brooklyn: Boca Raton, Fl: Crc Press, Taylor & Francis Group.

J. E. Hopcroft, R. M. (2007). *Introducción a la teoría de autómatas, lenguajes y computación (3a. ed.).* Madrid: Pearson Educación.

Ramos, L. P. (22 de enero de 2021). *Real Python*. Obtenido de https://realpython.com/qt-designer-python/#installing-and-running-qt-designer