Universidad Rafael Landívar

Facultad de Ingeniería

Laboratorio de Arquitectura del Computador

Catedrático: Ing. Juan Carlos Soto Santiago

DOCUMENTACION

PROYECTO #1

Godinez Gudiel, Javier Estuardo

Carné: 1179222

Cuevas Lau, Ubaldo Sebastian

Carne: 1034222

Ovalle Montenegro, Saul Alejandro

Carne: 1226122

Guatemala de la Asunción, 13 de Abril del 2024

Contenido

Objetivo

- El programa tiene como objetivo manejar autómatas finitos deterministas (AFD), permitiendo cargar la configuración del autómata desde un archivo, agregar transiciones, verificar cadenas y mostrar resultados de estas verificaciones.

Funcionalidades

- Cargar un Autómata: El programa debe ser capaz de leer desde un archivo los estados, el estado inicial, los estados finales y las transiciones de un autómata.

- Agregar Transiciones: Debe permitir añadir nuevas transiciones al autómata en tiempo de ejecución.

- Verificación de Cadenas: Debe verificar si una cadena dada es aceptada por el autómata, mostrando las transiciones utilizadas durante el proceso.

- Interfaz de Usuario: Una interfaz de consola para interactuar con el usuario, permitiendo realizar las operaciones anteriores y visualizar resultados.

Especificaciones

- **Entradas:**
  - Archivos de configuración del AFD en formatos TXT, JSON y CSV. Cada formato tendrá una estructura específica que deberá ser documentada para que los usuarios sepan cómo preparar sus archivos.
  - Cadenas ingresadas por el usuario para verificar contra el autómata.

- **Procesos:**
  - Lectura y análisis del archivo para construcción del autómata.
  - Adición de transiciones al autómata.
  - Evaluación de cadenas con registro de transiciones utilizadas.

- **Salidas:**
  - Mensajes de error o confirmación sobre la carga y análisis del archivo.
  - Resultado de la evaluación de cadenas (aceptada o no).
  - Listado de transiciones utilizadas durante la evaluación.

Diseño de la Solución

FiniteAutomata

- **Atributos:**
  - statesQuantity: Cantidad de estados del autómata.
  - initialState: Estado inicial.
  - finalStates: Lista de estados finales.
  - transitions: Diccionario para almacenar las transiciones.

- **Métodos:**
  - Constructor para inicialización.
  - AddTransition: Añade una transición al autómata.
  - AcceptsString: Verifica si una cadena es aceptada por el autómata.
  - PrintTransitions: Imprime todas las transiciones del autómata.
  - PrintUsedTransitions: Imprime las transiciones usadas para una cadena.

DocumentReader

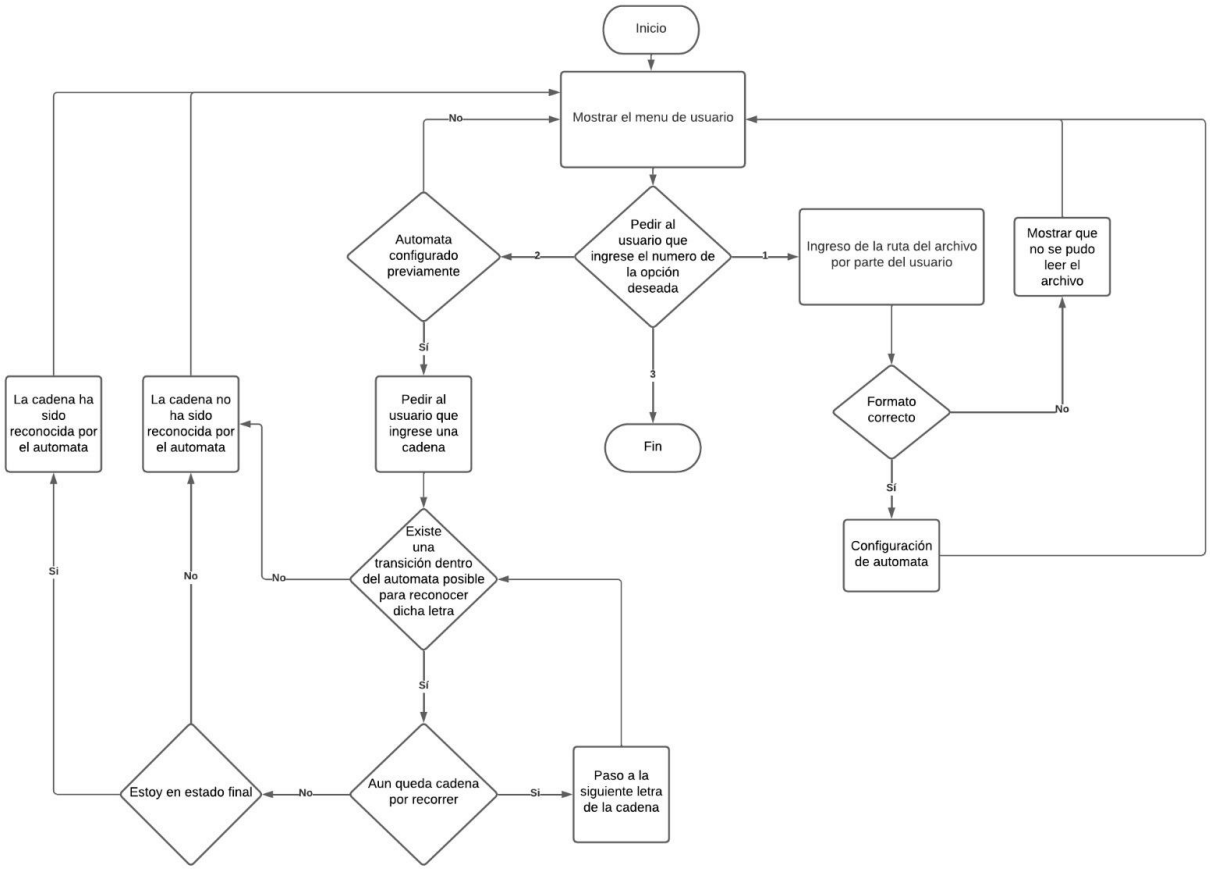
- **Atributos:**
  - automata: Instancia de FiniteAutomata.

- **Métodos:**
  - ReadAutomatonFile: Lee un archivo y configura el autómata basándose en su contenido.

Program

- **Métodos:**
  - Main: Bucle principal que maneja la interfaz de usuario y las interacciones.
  - Menu: Muestra opciones disponibles al usuario.

Diagrama de Flujo



Pseudocodigos

- **Main Program**

```
Método Main()
    automata = nulo
    documentReader = nuevo DocumentReader()

    Mientras verdadero
        Menu()
        Intentar
            imprimir("Ingrese la opción deseada:")
            decisión = leer línea y convertir a entero

            Si decisión = 1
                Limpiar consola
                imprimir("Ingrese ruta del archivo del AFD:")
                filePath = leer línea
                automata = documentReader.ReadAutomatonFile(filePath)

            Si decisión = 2
                Si automata no es nulo
                    Limpiar consola
                    imprimir("Ingrese una cadena para verificar:")
                    userInput = leer línea

                    Si userInput no está vacío
                        accepted = automata.AcceptsString(userInput, usadoTransiciones)
```

```
            automata.PrintUsedTransitions(usadoTransiciones)
            Si accepted
                imprimir en verde("El autómata ha aceptado la cadena.")
            Sino
                imprimir en rojo("El autómata no ha aceptado la cadena.")
        Sino
            imprimir("Ingrese algo válido")

    Atrapar
        imprimir("Formato inválido")
        Limpiar consola

    Si decisión es diferente de 1 o 2
        romper bucle
```

- **Clase DocumentReader**

```
Privado:
    automata: FiniteAutomata

Método ReadAutomatonFile(filePath)
    Intentar
        líneas = leer todas las líneas de archivo en filePath

        Si longitud de líneas < 3
            imprimir("Formato de archivo incorrecto")
            retorna nulo
```

```
        totalStates = convertir a entero líneas[0]
        initialState = convertir a entero líneas[1]
        finalStates = convertir a lista de enteros líneas[2] separadas por ','

        automata = nuevo FiniteAutomata(totalStates, initialState, finalStates)

        Para i = 3 hasta longitud de líneas
            valores = líneas[i] separadas por ','
            Si longitud de valores = 3
                transitionInitialState = convertir a entero valores[0]
                stringRead = valores[1] recortado
                transitionFinalState = convertir a entero valores[2]
                automata.AddTransition(transitionInitialState, stringRead, transitionFinalState)
            Sino
                imprimir("Formato de línea inválido")

        imprimir("Estados finales y total de transiciones")
        retorna automata
    Atrapar excepción
        imprimir("Error al leer archivo")
        retorna nulo
```

- **Clase AutomataFinito**

```
Clase FiniteAutomata
  Privado:
    estadosQuantity: entero
    initialState: entero
```

```
    finalStates: lista de enteros
    transitions: diccionario de ((entero, cadena), entero)

Constructor(statesQuantity, initialState, finalStates)
    this.statesQuantity = statesQuantity
    this.initialState = initialState
    this.finalStates = finalStates
    this.transitions = nuevo diccionario

Método AddTransition(initialState, stringRead, finalState)
    clave = (initialState, stringRead)
    transitions[clave] = finalState

Método AcceptsString(inputStr, usadoTransiciones)
    currentState = initialState
    usadoTransiciones = nueva lista de cadenas

    Para cada símbolo en inputStr
        stringRead = símbolo a cadena
        clave = (currentState, stringRead)

        Si transitions contiene clave
            nextState = transitions[clave]
            usadoTransiciones.agregar("currentState --(stringRead)--> nextState")
            currentState = nextState
        Sino
            retorna falso

    retorna finalStates contiene currentState
```

```
Método PrintTransitions()
    Para cada transición en transitions
        imprimir("transition.Key.Item1 --(transition.Key.Item2)--> transition.Value")

Método PrintUsedTransitions(usedTransitions)
    imprimir("Transiciones utilizadas:")
    Para cada transición en usedTransitions
        imprimir(transición)
```