Facultad de

Ingeniería

Lenguajes Formales y Autómatas

Sección 02

Proyecto Autómatas Finitos

Segunda Entrega

Análisis y Diseño de la solución

Docente:

Ing. Juan Carlos Soto Santiago

Autores:

Diego Estuardo Azurdia Marín – 1010821

Camilo Javier Solís Mejía – 2311824

Katherine Andrea Mayen Rivera – 1129222

Ciudad de Guatemala, Guatemala, sábado 13 de abril de 2024.

Análisis de la solución

1. Requisitos funcionales: Leer archivos de extensión (.txt, .csv, .json) que cumplan con el formato
    - # de estados (n)
    - # estado inicial (1..n)
    - Conjunto de estados finales separados por comas

    Una línea por cada transición separando por comas: Estado Inicial (1..n), Cadena Leída, Estado

    Esto con la finalidad de crear un autómata que permita validar cadenas.

2. Requisitos no funcionales: Tener un menú en el que el usuario pueda elegir el tipo de autómata finito a validar y funciones que eviten que el usuario genere una sobrecarga.
3. Entradas y salidas: El programa debe leer archivos de extensión (.txt, .csv, .json) que cumplan con el formato
    - # de estados (n)
    - # estado inicial (1..n)
    - Conjunto de estados finales separados por comas

    Una línea por cada transición separando por comas: Estado Inicial (1..n), Cadena Leída, Estado

    Imprimir el autómata a partir de los datos leídos en las tres extensiones y con el formato correcto. Debe recibir cadenas dadas por el usuario, imprimir el recorrido que realiza el autómata con esas cadenas e imprimir si son aceptadas por el autómata o no.

4. Casos de uso:
    a. Cargar Definición del AFD
        - Actor: Usuario
        - Objetivo: Cargar la definición de un AFD desde un archivo.
        - Flujo:

        i. El usuario selecciona el tipo de AF a validar.
        ii. El usuario selecciona la opción de autómata finito determinista.
        iii. El sistema le solicita el path del archivo.
        iv. El usuario carga el path del archivo con extensión (.txt, .csv, .json) con el formato establecido.
        v. El sistema lee y valida el archivo, mostrando un mensaje de éxito o error.
        vi. El sistema imprime el autómata generado con la información del archivo.

    b. Validar Cadena de Entrada
        - Actor: Usuario

- Objetivo: Validar una cadena de entrada contra el AFD cargado.
- Flujo:

i. El sistema le solicita al usuario la cadena a validar.
ii. El usuario ingresa una cadena de texto para validar.
iii. El sistema realiza la validación contra el AFD.
iv. El sistema imprime el recorrido que realiza el AFD durante la validación.
v. El sistema muestra el resultado del reconocimiento (Válido/No válido).

Diseño de la solución

1. Arquitectura del sistema:
    a. Lectura del archivo:
        - La función LeerArchivos() permite al usuario ingresar la ruta de un archivo y procesarlo según su extensión (.txt, .csv, .json).
        - Las funciones LeerArchivoTxt(), LeerArchivoCsv(), y LeerArchivoJson() se encargan de leer y parsear los archivos de texto, CSV y JSON. y crear una instancia de AutomataEntity que representa el autómata definido en el archivo.
    b. Impresión de Detalles del Autómata:
        - La función ImprimirAutomata() muestra los detalles del autómata (estados, estado inicial, estados finales, transiciones) en la consola.
        - Validación de Cadenas en el Autómata:
        - La función ConsultarCadena() permite al usuario ingresar una cadena de caracteres para validarla contra el autómata.
        - La función RecorrerAF() realiza un recorrido recursivo del autómata con la cadena ingresada, siguiendo las transiciones correspondientes.
        - Las funciones SigEstado(), ImprimirPaso(), y VerifEstFinal() ayudan en el proceso de validación y recorrido del autómata.
    c. Interfaz de Usuario:
        - Las funciones MostrarMenu(), MostrarOpcion1(), MostrarOpcion2(), y MostrarRegresarMenu() manejan la interfaz de usuario mostrando un menú con diferentes opciones y navegando entre ellas.
    d. Funciones Auxiliares:
        - La función MostrarBienvenida() muestra un mensaje de bienvenida al inicio del programa.
        - La función MostrarEstructuraJson() muestra la estructura JSON esperada para los archivos .json.

2. Diseño de Clases y Componentes:

a. Entidades (AutomataEntity y TransicionEntity): Se utilizan para representar las estructuras de datos del autómata y sus transiciones.
   - AutomataEntity contiene propiedades como Estados, EstadoInicial, EstadosFinales, y Transiciones.
   - TransicionEntity contiene propiedades como EstadoOrigen, Simbolo, y EstadoDestino.
   - Esto ayuda a mantener una estructura clara y organizada del autómata y sus componentes. Permite utilizar un objeto en vez de variables globales para crear el autómata.

b. Funciones de Lectura (LeerArchivoTxt, LeerArchivoCsv, LeerArchivoJson):
   - Se encargan de la lectura y procesamiento de diferentes tipos de archivos para crear instancias de AutomataEntity.
   - Utilizan manejo de errores y excepciones para garantizar una lectura correcta y segura de los archivos.

c. Funciones de Validación y Recorrido (ConsultarCadena, RecorrerAF, SigEstado, ImprimirPaso, VerifEstFinal):
   - Implementan la lógica para validar cadenas en el autómata y realizar un recorrido basado en las transiciones.
   - Utilizan las entidades AutomataEntity y TransicionEntity para acceder y manipular la información del autómata.

3. Flujo de Datos:
   a. Entrada de Usuario:
      - El usuario proporciona la ruta del archivo y la cadena a validar a través de la consola.
      - Se validan y procesan estas entradas para garantizar su corrección y utilidad en las funciones correspondientes.
   b. Procesamiento y Validación:
      - Los archivos son leídos, parseados y convertidos en instancias de AutomataEntity.
      - Se utiliza un enfoque recursivo para validar y recorrer el autómata con la cadena proporcionada por el usuario, siguiendo las transiciones correspondientes.
   c. Salida y Visualización:
      - Los detalles del autómata y los resultados de la validación se muestran en la consola utilizando funciones de impresión y visualización.

4. Interfaces de Usuario:
   a. Menú Principal (MostrarMenu):
      - Presenta al usuario un menú con opciones para elegir entre diferentes tipos de autómatas y operaciones.

- Permite al usuario navegar entre las diferentes opciones y volver al menú principal o salir del programa.

b. Mensajes y Notificaciones:

- Se utilizan mensajes en la consola para guiar al usuario a través del proceso de ingreso de datos, visualización de resultados y navegación entre opciones.
- Se manejan diferentes escenarios y errores con mensajes descriptivos para informar al usuario sobre cualquier problema o resultado.