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, viernes 11 de mayo de 2024.

Análisis de la solución

1. Requisitos funcionales: Leer archivos de extensión .txt 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 contra el AFN, asimismo, ser capaz de imprimir las mientras se realiza la validación.

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 con el AFN 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 la extensión txt 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 AFN
        - Actor: Usuario
        - Objetivo: Cargar la definición de un AFN 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, 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 AFN 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 AFN.
iv. El sistema imprime el recorrido que realiza el AFN durante la validación.
v. El sistema muestra el resultado del reconocimiento (Válido/No válido).

Diseño de la solución

1. Gestión de recursividad:

a. Función RecorrerAFN:

- Recorre el autómata de forma recursiva, verificando en cada paso si el estado actual es un estado final y actualiza el recorrido.

b. Función ImprimirPasoN:

- Imprime el paso actual del recorrido, mostrando el estado actual, el símbolo de entrada y el siguiente estado, utilizando recursión para continuar el recorrido del AFN.

2. Manejo de casos especiales:
   a. Función RecorrerAFN(sobrecargada):
   - Se utiliza en donde se necesitan condiciones adicionales. Por ejemplo, cuando se detecta una ruta fallida o se necesita reconsiderar el último estado fallido en el recorrido.

   b. Función otrasRutasAFN:
   - Maneja casos en los que una ruta falla y se necesitan explorar otras posibilidades. Utiliza los recorridos anteriores para determinar las posibles decisiones y continuar la validación.
   c. Función VerifEstFinalN:
   - Verifica si el estado actual es un estado final. Si lo es, imprime un mensaje indicando que la cadena es aceptable. Si no es un estado final, imprime un mensaje indicando que la cadena no es aceptable. Maneja los estados de Epsilon y las rutas fallidas.

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. Función ConvertirEpsilon:
   - Convierte los espacios en blanco en el símbolo de Epsilon (ε) para su procesamiento en el AFN.

c. Función SigEstadoN:
   - Devuelve el siguiente estado en el AFN dado el estado actual y el símbolo de entrada. Comprueba las transiciones disponibles en el AFN y devuelve el siguiente estado correspondiente.