Universidad Rafael Landívar
Facultad de Ingeniería
Lenguajes Autómatas y Deterministas
Sección: 02
Ing. Carlos Soto

DOCUMENTACIÓN
PROYECTO AUTOMATAS DETERMINISTAS

Ángel Sebastián Altan 1031222
Juan Diego Gutierrez 1155222

Guatemala, 13 de abril de 2024

INDICE

INTRODUCCIÓN

Los autómatas finitos deterministas (AFD) son modelos matemáticos simples pero poderosos en teoría de la computación y ciencias de la computación. Estos autómatas se utilizan para representar sistemas que pueden estar en un número finito de estados y que, dado un símbolo de entrada, hacen una transición a otro estado determinísticamente. Cada estado tiene una acción asociada que indica qué hacer cuando se encuentra en ese estado y se recibe un símbolo específico.

En el presente manual se enseñará a utilizar la aplicación paso a paso, para el ingreso de autómatas y la validación de cadenas dentro del mismo.

DISEÑO Y ANÁLISIS

LIBRERIAS

1) **System**: Proporciona las clases fundamentales y los tipos de datos básicos de .NET.
2) **System.IO**: Contiene tipos que permiten leer y escribir en archivos y flujos de datos.
3) **System.Collections.Generic**: Ofrece colecciones genéricas como listas, diccionarios, conjuntos, etc., para almacenar y manipular datos.
4) **System.ComponentModel**: Proporciona clases para el manejo de componentes y eventos.
5) **System.Data**: Contiene clases para trabajar con bases de datos y conjuntos de datos.
6) **System.Drawing**: Permite trabajar con gráficos y dibujar en formularios y controles.
7) **System.Linq**: Proporciona funcionalidades para consultas y manipulación de datos en colecciones.
8) **System.Text**: Contiene clases para el manejo de codificación y decodificación de caracteres.
9) **System.Threading.Tasks**: Permite trabajar con tareas asincrónicas y paralelas.
10) **System.Windows.Forms**: Contiene clases y controles para el desarrollo de aplicaciones de Windows Forms, como formularios, botones, etiquetas, etc.
11) **System.Security.Cryptography.X509Certificates**: Proporciona clases para trabajar con certificados digitales y cifrado asimétrico.
12) **System.Text.RegularExpressions**: Permite trabajar con expresiones regulares para el manejo de patrones de texto.
13) **static System.Windows.Forms.LinkLabel**: Hace referencia al control LinkLabel dentro del namespace System.Windows.Forms para crear enlaces en formularios.
14) **static System.Windows.Forms.AxHost**: Proporciona funcionalidades para hospedar controles ActiveX en aplicaciones Windows Forms.

MÉTODOS

1. **public void restart()**: Este método restablece todas las variables en la memoria local a sus valores predeterminados.
2. **public bool number_comprobation(string line, bool transitionComprobation)**: Este método utiliza expresiones regulares para comprobar si una línea de texto es un número, dependiendo de si se está comprobando una transición específica o no.
3. **public void number_comprobation_final_states(string line)**: Este método procesa una línea de texto que contiene estados finales separados por comas y los agrega a la lista de estados finales.
4. **public bool transitions_validations(List<string> transitions)**: Este método realiza validaciones sobre el formato de las transiciones en el autómata, verificando que tengan la estructura adecuada y que los estados involucrados sean números.
5. **public bool states_comprobation(List<string> transitions, string state)**: Este método comprueba que un estado dado esté presente en las transiciones del autómata.
6. **public bool strings_validation(string initial_state, List<string> transitions, string word, List<string> final_states, List<string> alphabet)**: Este método valida una cadena de entrada con respecto al autómata, verificando que cumpla con las transiciones y llegue a un estado final válido.
7. **public List<string> get_alphabet(List<string> transitions)**: Este método obtiene el alfabeto del autómata a partir de las transiciones definidas.

MANUAL DE USUARIO

Paso 1:

Baja el repositorio de GitHub.

JuandiGuti/proyecto_automatas_1_aa1031222_jd1155222: Proyecto no. 1 de lenguajes formales y automatas, ADF y ADN (github.com)

Paso 2:

En dentro del repositorio en la siguiente ruta: automatonApp\automatonApp\publish, se encuentra el instalador llamado como “setup.exe”.

Paso 3:

Al inicial el programa “automatonapp.exe” se despliega el siguiente menú, ver imagen 1 en anexos. En este menú se observan 2 botones los cuales desencadenan otras funciones depende de lo que necesite el usuario.

UTILIZACIÓN DEL “DETERMINISTIC FINITE AUTOMATON”

Paso 1:

Presiona el botón llamado “Select File” e ingrese el archivo “.txt” con el formato correspondiente, ver imagen 2 en anexos.

Paso 2:

Una vez cargado el archivo, diríjase al recuadro que se encuentra debajo del mensaje “Insert your string” e ingrese la cadena que desea verificar, ver imagen 3 en anexos.

Paso 3:

Una vez ingresada la cadena, presione el botón “Check” y el resultado se desplegará en pantalla, ver imagen 4 en anexos.

ANEXOS

MENÚ DE INICIO

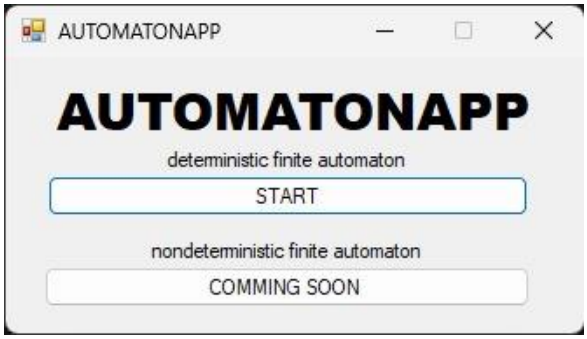

Imagen 1. Menú de inicio.

FORMATO DE ARCHIVO .TXT

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

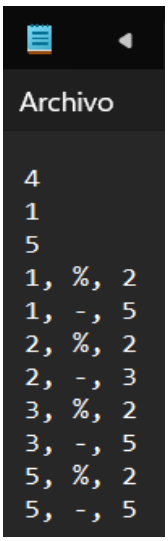

Imagen 2. Ejemplo de formato archivo txt.

INGRESO DE CADENA A VERIFICAR

Imagen 3. Ejemplo de ingreso de cadena a verificar.

VALIDACIÓN DE CADENA

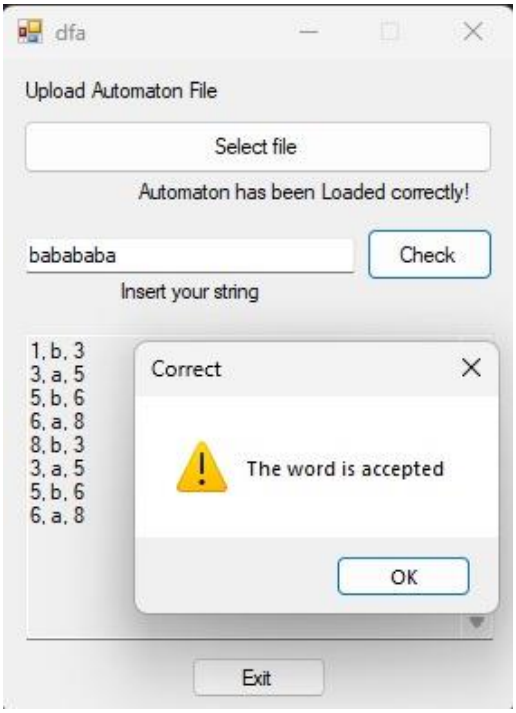

Imagen 4. Ejemplo validación de cadena.