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

DOCUMENTACIÓN
PROYECTO AUTOMATAS NO DETERMINISTAS

Ángel Sebastián Altan 1031222
Juan Diego Gutierrez 1155222

Guatemala, 10 de mayo de 2024

INDICE

Introducción

Un autómata finito no determinista (AFN) es una de las estructuras más fundamentales en la teoría de la computación y el diseño de lenguajes de programación, proporcionando un marco formal para el análisis y diseño de algoritmos y sistemas. Los AFN son una extensión de los autómatas finitos deterministas (AFD), permitiendo múltiples posibilidades de transiciones para un mismo símbolo de entrada desde un estado dado, e incluso aceptan la existencia de transiciones sin consumo de símbolos, conocidas como transiciones ε. Esta característica de no determinismo confiere a los AFN una gran potencia y flexibilidad, siendo capaces de representar cualquier lenguaje regular, al igual que los AFD.

La estructura básica de un AFN se compone de un conjunto finito de estados, un conjunto de símbolos de entrada (alfabeto), una función de transición que asigna a cada par estado-símbolo un conjunto de posibles estados siguientes, un estado inicial y un conjunto de estados de aceptación. A diferencia de los AFD, donde la función de transición devuelve un único estado siguiente, en un AFN esta función puede devolver un conjunto de estados, o incluso un conjunto vacío, indicando que no hay transición posible para ese símbolo en ese estado. La capacidad de un AFN para moverse a varios estados a la vez desde un único estado y símbolo de entrada, o no moverse en absoluto, introduce una complejidad en la simulación de estos autómatas, ya que se debe mantener un seguimiento de todos los posibles estados activos en cada paso de la computación. Sin embargo, esta aparente complejidad no aumenta la potencia de los autómatas finitos, pues todo AFN puede ser convertido en un AFD equivalente mediante el algoritmo de construcción de subconjuntos, aunque esto puede resultar en un crecimiento exponencial del número de estados.

Diseño y análisis

Librerias:

1. **System**: Proporciona clases fundamentales y bases para las aplicaciones de .NET Framework.
2. **System.Collections.Generic:**
    - **List<T>:** Utilizada para trabajar con listas de objetos.
3. **System.Linq:**
    - Métodos de extensión para colecciones, como Any(), Last(), Select(), y Where(), que permiten realizar consultas convenientes y manipulaciones de datos.
4. **System.IO:**
    - **StreamReader:** Utilizado para leer archivos de texto.
5. **System.Text.RegularExpressions:**
    - **Regex.IsMatch():** Verifica si una cadena específica coincide con un patrón de expresión regular.
6. **System.Windows.Forms:**
    - Componentes de interfaz de usuario como Form, Button, TextBox, y Label.
7. **System.Text:**
    - **StringBuilder:** Utilizado para la construcción eficiente de cadenas a través de la manipulación de múltiples cadenas.

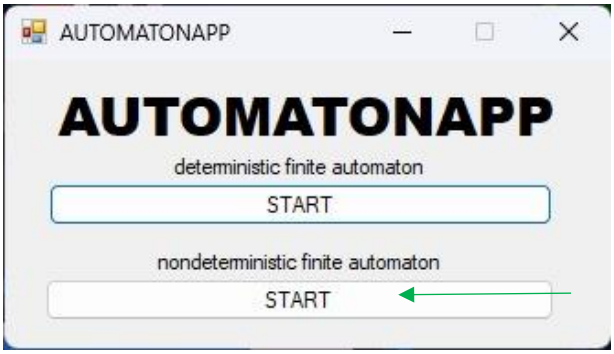
Métodos:

1. **ndfa_Load:** Método vacío asociado al evento de carga de la forma, no realiza ninguna acción explícita.
2. **restart:** Reinicia la configuración del autómata, limpiando listas y restableciendo valores predeterminados de la interfaz.
3. **get_alphabet**: Obtiene el alfabeto a partir de las transiciones excluyendo las transiciones ε.
4. **number_comprobation**: Verifica si una línea del archivo contiene sólo números.
5. **number_comprobation_final_states**: Procesa la línea que contiene los estados finales y los asigna a la lista de estados finales.
6. **states_comprobation:** Comprueba si un estado específico está presente en alguna de las transiciones.
7. **bt_open_file_Click**: Manejador del evento de clic para abrir un archivo, leer y procesar su contenido para establecer las configuraciones del autómata.
8. **FormatPaths:** Formatea las rutas de los estados por los que pasa el autómata para su visualización.
9. **bt_sent_string_Click:** Manejador del evento de clic para validar una cadena contra el autómata, determinando si es aceptada o no y mostrando los caminos procesados.
10. **strings_validation:** Valida si una cadena es aceptada por el autómata, procesando transiciones y verificando estados finales.
11. **PerformEpsilonTransitions:** Realiza transiciones ε en los caminos actualmente activos y añade los nuevos caminos resultantes.

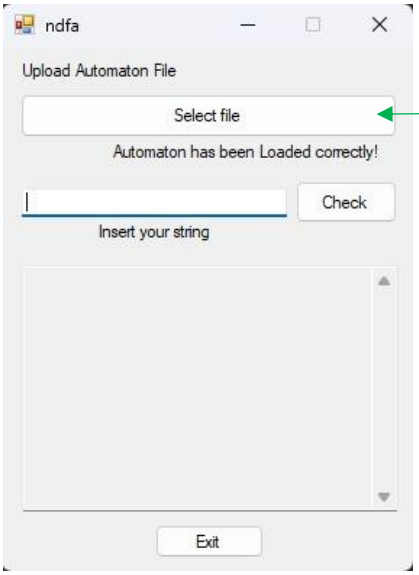
Manual de usuario

Paso 1:
Ejecuta el programa.

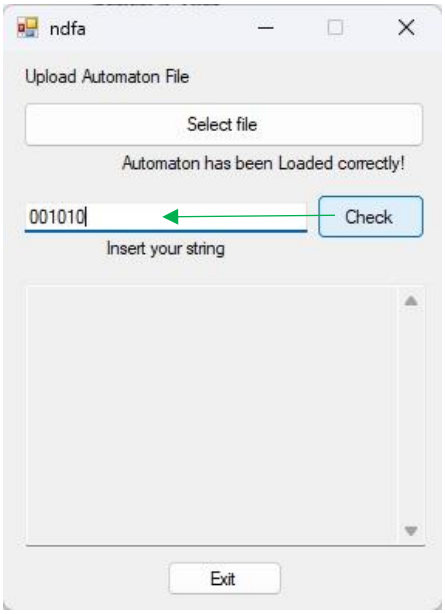
Paso 2:
Presiona el botón correspondiente a “Nondeterministic finite automaton” y se le redirigirá a un nuevo menú.



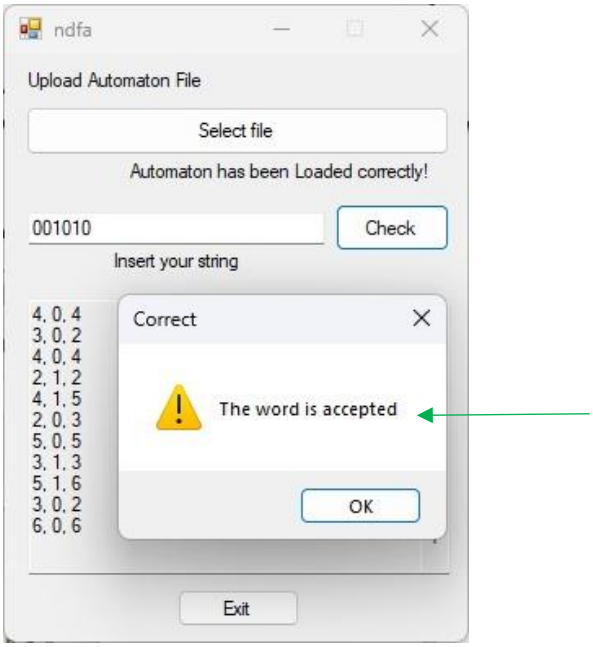
Paso 3:
Presiona el botón llamado “Select File” e ingrese el archivo “.txt” con el formato correspondiente.
(Si tiene alguna duda con el formato correspondiente véase en el apartado de anexos).



Paso 4:
Una vez cargado el archivo, diríjase al recuadro que se encuentra debajo del mensaje “Insert your string” e ingrese la cadena que desea verificar.



Paso 5:
Una vez ingresada la cadena, presione el botón “Check” y el resultado se desplegará en pantalla.



Anexos

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