**Un letrero de color blanco

Descripción generada automáticamente con confianza mediaUniversidad Mariano Gálvez de Guatemala**

**Sede Boca del Monte**

**Ingeniería en Sistemas de Información**

**Sección “B”**

**Autómatas y Lenguajes Formales**

**Ing**. Ezequiel Urizar Araujo

**Manual de Técnico | Máquina de Turing**

Cesar Alberto Tecún Leiva 7690-22-11766

Jesús David Sarat Obispo 7690-22-21168

Jose Daniel Tobar Reyes 7690-21-13125

Javier Estefano Martínez Lutín 7690-22-19643

Pablo Antonio Ispache Arriaga 7690-17-940

Octubre - 2024

# Maquina de Turing

Una máquina de Turing es un modelo teórico de computación creado por el matemático británico Alan Turing en 1936- Este modelo describe una máquina que realiza operaciones de acuerdo con reglas predefinidas y es capaz de resolver cualquier problema computable, siempre que tenga tiempo y recursos suficientes. Es un concepto fundamental en el campo de la computación, porque estable los límites de lo que es posible calcular.

# Desarrollo y compatibilidad del programa

El programa fue desarrollado como una aplicación Java Maven utilizando al IDE Apache Netbeans ver. 21 , por lo cuál es compatible con dicha versión y con versiones posteriores , de igual forma se utilizo el JDK 21. Todo esto dentro de un ambiente de Windows 11 , sin embargo , cabe mencionar que existe retrocompatibilidad con Windows 10.

#Requisitos mínimos para ejecutar el programa

1. 800 MB de espacio libre en el disco duro

2. 512 MB de RAM

3. Procesador Intel Pentium III a 800 Mhz

4. Sistema operativo Windows 10 o superior

5. Apeche Net Beans 21 o superior

6. JDK 21 o superior

# Funcionalidad básica

El programa tiene como propósito simular a una máquina de Turing , es decir es posible ingresar un autómata en el programa y este evaluara el autómata y determinara que palabras o cadenas de caracteres son aceptados o no.

#Clase Maquina\_Turing

Esta es la base principal de la simulación de la máquina de Turing , en ella se incluyen las propiedad de cinta y cabezal , así como los estados y transiciones que la máquina puede realizar.

## Componentes de la clase Maquina\_Turing

\*Cinta\* una representación de la cinta como un array de String que contiene los símbolos de entrada y otros que se vayan escribiendo.

\*Cabezal\* representa la posición actual de la cinta.

\*Estado actual\* estado en el que se encuentra la máquina en un momento dado.

\*EstadoInicial y EstadoAceptado\* representa el estado inicial y el estado de aceptación respectivamente.

\*Transición\* Es un mapa de transiciones donde cada llave combina un estado y un símbolo leído. Esto permite que, dada una combinación , la máquina sepa cuál es la siguiente acción.

##Clase Transición

En esta clase se representa cada transición que realiza la máquina de Turing para que funciones correctamente. Una parte clave del funcionamiento de la máquina de es el direccionamiento , ya que es necesario establecer parámetros que determinen si el autómata debe dirigirse hacia la izquierda o derecha. En este caso , se optó por utilizar el valor 1 para identificar la derecha y -1 para la izquierda.

Transicion(String Estadosiguiente, char Simboloescrito, int Direccion\_movimiento) {

this.EstadoSiguiente = Estadosiguiente;

this.SimboloEscrito = Simboloescrito;

this.Direccion\_Movimiento = Direccion\_movimiento;

}

## Constructores

El constructor recibe los parámetros de: EstadoInicial y estadoAceptado , para dar inicio a la ejecución de la máquina ; acá también se define el tamaño de la cinta, la posición del cabezal.

public Maquina\_Turning(String Estadoinicial, String Estadoaceptado) {

this.Cinta = new String[100];

this.Cabezal = 50;

this.EstadoInicial = Estadoinicial;

this.EstadoActual = Estadoinicial;

this.EstadoAceptado = Estadoaceptado;

this.Transicion = new HashMap<>();

}

Luego de que el constructor recibe los parámetros , el método addTransition permite ingresar las transiciones a la máquina . Primero recibe un estado y un símbolo leído así como el estado siguiente, el símbolo a escribir y la dirección de movimiento.

Por otra parte los método set Input inicializa la cinta con un valor de entrada , los caracteres se van agregando en la cinta en base a la posición del cabezal. Por último el método \*run\* lee el símbolo actual en la posición del cabezal, verifica la validez de la transición , y si existe , escribe el símbolo, mueve el cabezal, y cambia el estado.

#Interfaz Gráfica

Para la interfaz gráfica del programa se utilizó la un Jrame y luego se procedió con la fase de diseño . Para el desarrollo de la interfaz gráfica de utilizar se utilizo un Jpanel , el cuál contiene todos los elementos visuales de la interfaz, así como campos para la cadena de entrada , está permite al usuario ingresar la cadena que la máquina de Turing procesará. Esta puede ser limpiada automáticamente en el primer clic. A continuación se describen los botones que conforman la interfaz gráfica.

#Librerias utilizadas para desarrollar la interfaz

1. Javax.swing
2. Java.awt
3. Java.awt.event
4. Javax.swing.event
5. Java.util
6. Java.io
7. Java.net
8. Javax.swing.table
9. Javax.swing.filechooser

##Botones de la interfaz

\*Validar\* Ejecuta la lógica de validación y puede inicializar la máquina y mostrar el estado inicial.

\*Siguiente\* permite avanzar un paso en el procesamiento de la cadena de entrada.

\*Importar\* permite al usuario seleccionar un archivo de texto para importar la configuración de la máquina de Turing.

\*Exportar\* permite al usuario guardar la configuración actual de la máquina en un archivo csv.

##Metodos importantes ejecutados por la interfaz en segundo plano

\*Validación de Cadenas\* al hacer click en un botón , se obtienen los estados de los campos de texto y se crea una instancia en la máquina de Turing. Se validad las transiciones en la tabla, asegurando que todos los campos estén completos. Luego de esto ya se ejecuta la máquina de Turing para determinar si la cadena es aceptada o rechazada , mostrando el resultado al usuario.

\*Agregar y eliminar transiciones\* los usuarios pueden agregar nuevas transiciones a la tabla. Se valida que todos los campos de la última fila estén completos antes de permitir la adición de una nueva fila vacía.

\*Exportación de Datos a un archivo CSV\* un \*JfileChooser\* permite al usuario seleccionar la ubicación y el nombre del archivo CSV donde se guardarán los datos. Se asegura que el archivo tenga la extensión .csv y maneja la sobreescritura si el archivo ya existe.

Los datos en la table se escriben en el archivo , excluyendo los encabezados, junto con dos campos de texto que contienen el estado inicial y final.

while ((line = reader.readLine()) != null) {

String[] data = line.split(","); // Dividir la línea en partes

if (data.length == 5) {

model.addRow(data); // Añadir la fila a la tabla

}

}

JOptionPane.showMessageDialog(this, "Datos exportados correctamente de: " + fileToOpen.getAbsolutePath());

} catch (IOException e) {

JOptionPane.showMessageDialog(this, "Tuvimos un error al cargar : " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

\*Lectura de Filas\* el bloque de código se encarga de leer las filas de un archivo CSV, dividiendo cada línea en partes usando la coma como delimitador.

\*Mensaje de éxito\* luego de concluida la lectura , se muestra un mensaje de éxito confirmando que los datos se han exportado correctamente.

#Futuras mejoras y actualizaciones

A partir de diferentes pruebas se encontró un bug , el cuál esta relacionado con el manejo de movimientos en un lenguaje causando un bucle infinito. Este error se presenta cuando al insertar los valores 1 y -1 en las columnas de movimiento , acompañadas de un espacio adicional.

En futuras versiones se espera corregir dicho bug , así como dar una mejor optimización al código y mejorar la calidad de la interfaz gráfica para que este sea más intuitiva y amigable visualmente para el usuario.