Universidad Mariano Gálvez de Guatemala Facultad de Ingeniería en Sistemas

Centro Universitario de Boca del Monte – Villa Canales

Autómatas y Lenguajes Formales

Ing. Miguel L. Pichiya Catu VI Semestre

Eduardo Daniel Ovalle Cruz 76902022201

Guatemala,noviembre, 2,023

PROYECTO

SIMULADOR DE MAQUINA DE TURING

Descripción:

El proyecto consiste en la elaboración de un programa en lenguaje C ó Java, capaz de simular una máquina de Turing con una cabeza lectora/escritora y una cinta infinita.

Al inicio el programa preguntará por la ubicación de un archivo de texto, el cual contendrá la definición formal de una MT cualquiera. Una vez cargada esta definición, el programa debe permitir el ingreso de una serie de caracteres en pantalla a la cinta infinita (finita en este caso). Este ingreso será desde teclado y en modo interactivo. Se deberá validar que los caracteres ingresados a la cinta estén definidos tanto en el conjunto de símbolos de la cinta como de la máquina. El modo de ingreso terminará cuando el usuario presione la tecla ENTER.

Inmediatamente después del ingreso de la cadena, el programa empezará la simulación de la MT en pantalla, pudiendo quedar en dos estados posibles, ACEPTACIÓN ó NO ACEPTACIÓN. Una vez terminada la simulación el programa debe permitir el ingreso de una nueva cinta o de una nueva definición de máquina.

FORMATO DEL ARCHIVO DE ENTRADA

El archivo de entrada contendrá la definición de una MT siguiendo el siguiente formato:

Estados(estado1, estado2, estado3, … , estadon) Inicial(estado1)

Ha(estadoi) He (estadoi)

Alfabeto(simbolo1, simbolo2, simbolo3., …)

MT (estadoM, símbolo_leido) = (estadoK, símbolo_escrito, Movimiento)

Movimiento podrá tener únicamente los siguientes valores:

- D = Movimiento a la derecha
- I = Movimiento a la izquierda
- N = No se mueve

Presentación y Entrega

Deberá entregarse la siguiente documentación:

- Documentación estándar: Carátula, Introducción, Objetivos, Conclusiones, Bibliografía
- Diagramas de MT
- Código fuente del programa con su respectiva documentación interna. Entregar la carpeta total del proyecto generado por NetBeans, Eclipse, IntelliJ DEA, así como de las herramientas auxiliares utilizadas.
- Manual o guía para la compilación y ejecución del programa.
- Impresión de pantallas que muestren la salida/ejecución del programa.

Tabla de Calificación

No.	Aspecto a evaluar	Puntaje
1	Entrega	5
2	Documentación estándar	10
3	Diagramas de MT	10
4	Código fuente con documentación interna	20
5	Ejecución del programa	40
6	Impresión de pantallas de salida/ejecución del programa	15
	Total:	**100**
	Punteo Neto:	**15 pts EX. Final**

RECORDAR QUE ESTE PROYECTO TIENE UN PUNTEO DE 15 PUNTOS SOBRE LA NOTA DEL EXÁMEN FINAL.

Introducción

Una máquina de Turing es un dispositivo que manipula símbolos sobre una tira de cinta de acuerdo con una tabla de reglas. A pesar de su simplicidad, una máquina de Turing puede ser adaptada para simular la lógica de cualquier algoritmo de computador y es particularmente útil en la explicación de las funciones de una CPU dentro de un computador.

Originalmente fue definida por el matemático inglés Alan Turing como una «máquina automática» en 1936 en la revista Proceedings of the London Mathematical Societynota. La máquina de Turing no está diseñada como una tecnología de computación práctica, sino como un dispositivo hipotético que representa una máquina de computación. Las máquinas de Turing ayudan a los científicos a entender los límites del cálculo mecánico.

Turing dio una definición sucinta del experimento en su ensayo de 1948, «Máquinas inteligentes». Refiriéndose a su publicación de 1936, Turing escribió que la máquina de Turing, aquí llamada una máquina de computación lógica, consistía en:

una ilimitada capacidad de memoria obtenida en la forma de una cinta infinita marcada con cuadrados, en cada uno de los cuales podría imprimirse un símbolo. En cualquier momento hay un símbolo en la máquina; llamado el símbolo leído. La máquina puede alterar el símbolo leído y su comportamiento está en parte determinado por ese símbolo, pero los símbolos en otros lugares de la cinta no afectan el comportamiento de la máquina. Sin embargo, la cinta se puede mover hacia adelante y hacia atrás a través de la máquina, siendo esto una de las operaciones elementales de la máquina. Por lo tanto, cualquier símbolo en la cinta puede tener finalmente una oportunidad.

Objetivos

Diseño de máquinas de Turing con objetivos prefijados.

A continuación, vamos a diseñar máquina de Turing que realicen tareas concretas. En lo que sigue, salvo que se indique lo contrario, denotaremos una casilla vacía en la cinta de la maquina por el símbolo S0. Llamaremos estado de partida de una máquina de Turing al estado en el que se encuentra el dispositivo cuando comienza a actuar.

Intuitivamente, una descripción instantánea debe entenderse de la manera siguiente:

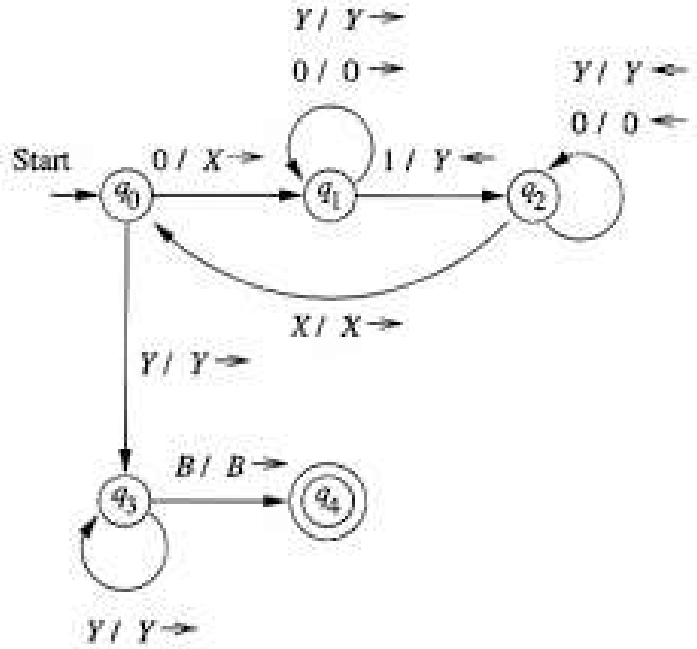
“Los símbolos de la cinta de las letras que aparecen en P sj y P´ (escritos éstos en celdas contiguas y casillas en blanco en el resto) y la cabeza lectora-inscriptora se encuentra el estado ei examinando la casilla que contine sj”.

Código Fuente

Enlace Repositorio GitHub:

https://github.com/educruz13/Proyecto-Final-Aut-matas.git

Diagrama



Código Fuente Textual Documentado (Internamente)

```
import re

class TuringMachine:
    def __init__(self, definition):
        self.states = set(definition.get("Estados", []))
        self.initial_state = definition.get("Inicial", None)
        self.accept_states = set(definition.get("Aceptación", []))
        self.reject_states = set(definition.get("Rechazo", []))
        self.alphabet = set(definition.get("Alfabeto", []))
        self.transitions = definition.get("Transiciones", {})

        if not self.initial_state:
            print("Error: Initial state not defined.")
            return

        self.current_state = self.initial_state
        self.tape = ['_']
        self.head_position = 0

    def load_tape(self, input_str):
        self.tape = ['_'] + list(input_str) + ['_']
        self.head_position = 1

    def run(self):
        while self.current_state not in self.accept_states and self.current_state not in self.reject_states:
            symbol_under_head = self.tape[self.head_position]

            if symbol_under_head not in self.alphabet:
                print(f"Error: Symbol '{symbol_under_head}' not defined in the alphabet.")
                break

            if (self.current_state, symbol_under_head) not in self.transitions:
                print(f"Error: No transition defined for the current state '{self.current_state}' and symbol '{symbol_under_head}'.")
                break

            next_state, write_symbol, move_direction = self.transitions[(self.current_state, symbol_under_head)]

            self.tape[self.head_position] = write_symbol

            if move_direction == "D":
                self.head_position += 1
            elif move_direction == "I":
                self.head_position -= 1
            elif move_direction != "N":
                print(f"Error: Invalid move direction '{move_direction}'.")
                break

            self.current_state = next_state

            # Agregar mensajes de depuración
            print(f"Current State: {self.current_state}, Symbol Under Head: {symbol_under_head}")
            print(f"Next State: {next_state}, Write Symbol: {write_symbol}, Move Direction: {move_direction}")
            print(f"Tape: {''.join(self.tape)}, Head Position: {self.head_position}")

            if self.head_position < 0:
                self.tape = ['_'] + self.tape
            elif self.head_position >= len(self.tape):
                self.tape.append('_')
```

```
        if self.current_state in self.accept_states:
            print("ACEPTACIÓN")
        else:
            print("NO ACEPTACIÓN")

        print("Cinta final:", ''.join(self.tape).rstrip('_'))

def parse_input_file(file_path):
    try:
        with open(file_path, 'r') as file:
            content = file.read()

        definition = {"Transiciones": {}}

        # Extraer estados
        match = re.search(r'Estados: (.+)', content)
        if match:
            definition["Estados"] = [state.strip() for state in match.group(1).split(',')]

        # Estado inicial
        match = re.search(r'Inicial: (.+)', content)
        if match:
            definition["Inicial"] = match.group(1).strip()

        # Estados de aceptación
        match = re.search(r'Aceptación: (.+)', content)
        if match:
            definition["Aceptación"] = [state.strip() for state in match.group(1).split(',')]

        # Estados de rechazo
        match = re.search(r'Rechazo: (.+)', content)
        if match:
            definition["Rechazo"] = [state.strip() for state in match.group(1).split(',')]

        # Alfabeto
        match = re.search(r'Alfabeto: (.+)', content)
        if match:
            definition["Alfabeto"] = [symbol.strip() for symbol in match.group(1).split(',')]

        # Transiciones
        transition_pattern = re.compile(r'(\S+),\s*(\S+)\s*->\s*(\S+),\s*(\S+),\s*(\S+)')
        transitions_text = re.search(r'Transiciones:(.+)', content, re.DOTALL)
        if transitions_text:
            transitions = transition_pattern.findall(transitions_text.group(1).strip())
            for transition in transitions:
                key = (transition[0], transition[1])
                value = (transition[2], transition[3], transition[4])
                definition["Transiciones"][key] = value

        return definition
    except Exception as e:
        print(f"Error al leer el archivo: {e}")
        return None
```

 # Ruta al archivo de definición de la MT

```
file_path = "regla.txt"

try:
    mt_definition = parse_input_file(file_path)
    if mt_definition is not None:
        tm = TuringMachine(mt_definition)
        if tm.initial_state:
            print("Estado inicial:", tm.initial_state)
            input_str = input("Ingrese una cadena para la cinta (presione ENTER para salir): ")
            tm.load_tape(input_str)
            tm.run()
except Exception as e:
    print(f"Error: {e}")
```

Compilación y Ejecución

```
PS C:\Users\Daniel\Desktop\Turing Maquina> & C:/Users/Daniel/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Daniel/Desktop/Turing Maquina/Turing Maquina/main.py"
PS C:\Users\Daniel\Desktop\Turing Maquina> & C:/Users/Daniel/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Daniel/Desktop/Turing Maquina/Turing Maquina/main.py"
Estado inicial: Q0
Ingrese una cadena para la cinta (presione ENTER para salir): 001
Current State: Q1, Symbol Under Head: 0
Next State: Q1, Write Symbol: x, Move Direction: D
Tape: _x01_, Head Position: 2
Current State: Q1, Symbol Under Head: 0
Next State: Q1, Write Symbol: 0, Move Direction: D
Tape: _x01_, Head Position: 3
Current State: Q2, Symbol Under Head: 1
Next State: Q2, Write Symbol: y, Move Direction: D
Tape: _x0y_, Head Position: 4
Current State: Q3, Symbol Under Head: _
Next State: Q3, Write Symbol: _, Move Direction: D
Tape: _x0y_, Head Position: 5
Current State: Q4, Symbol Under Head: _
Next State: Q4, Write Symbol: _, Move Direction: I
Tape: _x0y__, Head Position: 4
NO ACEPTACIÓN
Cinta final: _x0y
PS C:\Users\Daniel\Desktop\Turing Maquina>

110
PS C:\Users\Daniel\Desktop\Turing Maquina> & C:/Users/Daniel/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Daniel/Desktop/Turing Maquina/Turing Maquina/main.py"
Estado inicial: Q0
Ingrese una cadena para la cinta (presione ENTER para salir): 0110
Current State: Q1, Symbol Under Head: 0
Next State: Q1, Write Symbol: x, Move Direction: D
Tape: _x110_, Head Position: 2
Current State: Q2, Symbol Under Head: 1
Next State: Q2, Write Symbol: y, Move Direction: D
Tape: _xy10_, Head Position: 3
Current State: Q2, Symbol Under Head: 1
Next State: Q2, Write Symbol: 1, Move Direction: D
Tape: _xy10_, Head Position: 4
Current State: Q2, Symbol Under Head: 0
Next State: Q2, Write Symbol: 0, Move Direction: D
Tape: _xy10_, Head Position: 5
Current State: Q3, Symbol Under Head: _
Next State: Q3, Write Symbol: _, Move Direction: D
Tape: _xy10_, Head Position: 6
Current State: Q4, Symbol Under Head: _
Next State: Q4, Write Symbol: _, Move Direction: I
Tape: _xy10__, Head Position: 5
NO ACEPTACIÓN
Cinta final: _xy10
PS C:\Users\Daniel\Desktop\Turing Maquina>
```

Manual de Uso
Máquina de Turing

 # Máquina de Turing en Python
Este es un simple simulador de Máquina de Turing implementado en Python. La Máquina de Turing (MT) es un modelo teórico de cómputo que manipula símbolos en una cinta de acuerdo con un conjunto de reglas.

Requisitos
Asegúrate de tener Python instalado en tu máquina. Puedes descargarlo desde [python.org](https://www.python.org/).

Uso
1. Clona o descarga el código fuente.
2. Ejecuta el archivo proporcionado (`turing_machine.py`).
3. El programa solicitará la ruta del archivo de definición de la MT.

Archivo de Definición de la Máquina de Turing
El programa espera un archivo de definición siguiendo un formato específico. Aquí hay un ejemplo (`regla.txt`):

Estados: q0, q1, q2
Inicial: q0
Aceptación: q2
Rechazo: q1
Alfabeto: 0, 1, _
Transiciones:
q0, 0 -> q1, 1, D
q1, 1 -> q2, 1, N

- ***Estados**:* *Lista de estados separados por comas.*
- ***Inicial**:* Estado inicial.
- ***Aceptación**:* Estados de aceptación.
- ***Rechazo**:* Estados de rechazo.
- ***Alfabeto**:* Símbolos del alfabeto separados por comas.
- ***Transiciones**:* Reglas de transición en el formato `estado actual, símbolo leído -> próximo estado, símbolo escrito, dirección`.

Conclusiones

Una Máquina de Turing, o MT, se considerar una cinta infinita dividida en casillas, cada una de las cuales contiene un símbolo, y sobre la cual actúa un dispositivo que puede adoptar diversos estados, y que lee un símbolo de la casilla sobre la que está situado. En función de dicho símbolo y del estado actual, se pueden realizar tres acciones siguientes: pasa a un nuevo estado, imprime un símbolo en lugar del que acaba de leer y se desplaza a una posición hacia la izquierda, derecha, o se detiene.

La creación modular de una máquina de Turing permite desarrollar máquinas complejas a partir de bloques elementales, mediante diagramas de transiciones. La construcción de máquinas de Turing se lleva a cabo mediante dichos diagramas de transición, y sus combinaciones.

E grafías

- chromeextension://efaidnbmnnnibpcajpcglclefindmkaj/https://ocw.ehu.eus/pluginfile.php/45553/mod_page/content/1/tema2.pdf
- https://formatalent.com/que-es-una-maquina-de-turing-y-como-funciona/
- https://bootcampai.medium.com/m%C3%A1quinas-de-turing-c329ccc270f
- https://www.youtube.com/watch?v=iaXLDz_UeYY