Tecnológico de Monterrey

Implementación de métodos computacionales (Gpo 603)

Tarea 4.2. Diseño de MT para resolver un problema específico

Equipo 3

Diego Alejandro Espejo López A01638341

Julieta Carolina Arteaga Legorreta A0637444

Marcela Beatriz De La Rosa Barrios A01637239

Pablo Heredia Sahagún A01637103

Adela Alejandra Solorio Alcázar A01637205

30/04/2024

Tarea 4.2. Diseño de MT para resolver un problema específico

7.1 C = “The set of strings with an equal number of 0’s and 1’s”

Breve descripción del funcionamiento de la máquina que diseñaron.

Esta máquina de estados cuenta con 6 estados totales, de los cuales 1 es de aceptación, y 1 de basura. Esta implementación acepta todas las cadenas que cuenten con la misma cantidad de 0s que de 1s, independientemente del número con el que se inicia, por lo que acepta cadenas del estilo 0^n1^n y 1^n0^n.

Esta máquina de Turing cuenta con una cinta, la cual se utiliza para irla llenando de X cada vez que lee un carácter diferente, eventualmente se irá moviendo a la derecha hasta llegar al límite final de la cinta, cuando esto suceda, se regresa al límite inicial de la cinta y (por así decirlo) se reinicia el proceso, si llega el caso en el que regresa al límite final de la cinta, desde el límite inicial, y solo lee X, esto significa que la cadena tiene la misma cantidad de 0s y 1s, por lo que se acepta.

Si esto no sucede, el proceso se reinicia hasta que se cumpla esta condición, si llega a suceder que se llegue al límite final de la cinta antes de llegar al estado q3, en este caso automáticamente no se acepta la cadena.

Descripción formal de la máquina

M = (Q, ∑, Γ,δ, q0, B, F)

Q = {q0,q1,q2,q3,q4,ε}

∑ = {0, 1}

Γ = {0, 1, X, Δ}

δ = función de transición →

q0 = q0 (estado inicial)

B = “ ”

F = q4 (estado final)

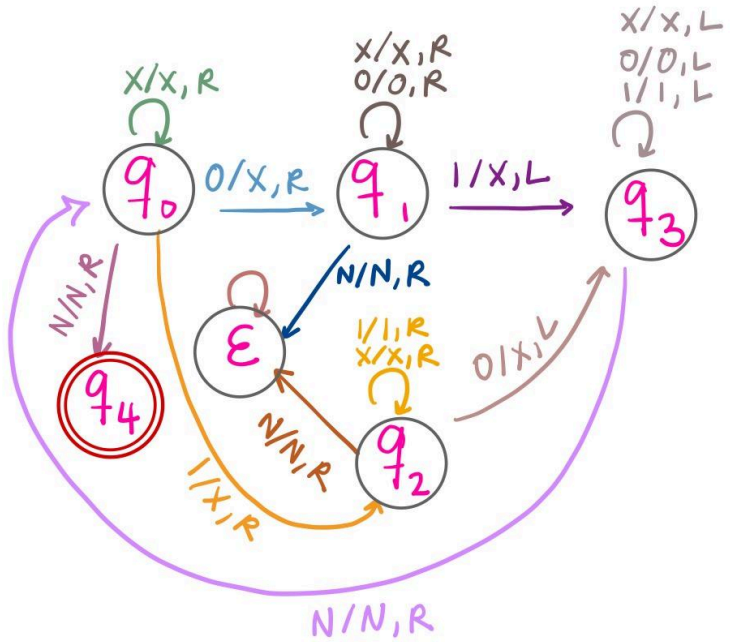
```
transition_function = {
    ("init", "x"): {"write": "x", "next_state": "init", "move": 1},
    ("init", "0"): {"write": "x", "next_state": "q1", "move": 1},
    ("init", "1"): {"write": "x", "next_state": "q2", "move": 1},
    ("init", " "): {"write": " ", "next_state": "final", "move": 1},

    ("q1", "x"): {"write": "x", "next_state": "q1", "move": 1},
    ("q1", "0"): {"write": "0", "next_state": "q1", "move": 1},
    ("q1", "1"): {"write": "x", "next_state": "q3", "move": -1},
    ("q1", " "): {"write": " ", "next_state": "E", "move": 1},

    ("q2", "x"): {"write": "x", "next_state": "q2", "move": 1},
```

```
    ("q2", "1"): {"write": "1", "next_state": "q2", "move": 1},
    ("q2", "0"): {"write": "x", "next_state": "q3", "move": -1},
    ("q2", " "): {"write": " ", "next_state": "E", "move": 1},
    ("q3", "0"): {"write": "0", "next_state": "q3", "move": -1},
    ("q3", "1"): {"write": "1", "next_state": "q3", "move": -1},
    ("q3", "x"): {"write": "x", "next_state": "q3", "move": -1},
    ("q3", " "): {"write": " ", "next_state": "init", "move": 1}
```

Diagrama de estados y transiciones.



Dos o tres ejemplos de funcionamiento.

Extra: Lo que está en itálicas es en donde se encuentra el puntero de la cinta.

1)

[N *0* 1 0 1 N]

[N X *0* 1 N]

[N *X* X 0 1 N]

[*N* X X 0 1 N]

[N X X X *1* N]

[N X X *X* X N]

[*N* X X X X N]

[N X X X X *N*] -> **Aceptado**

2)

[N 1 1 1 1 0 N]
[N X *I* 1 1 0 N]
[N X 1 1 *I* X N]
[*N* X 1 1 1 X N]
[N X *I* 1 1 X N] -> q1
[N X X *I* 1 X N]
[N X X 1 1 X *N*] -> q2
[N X X 1 1 X N] -> **No Aceptado**

3) [0 0 1]

```
Cinta actual:   001  Estado actual:  init Valor actual:  1
Cinta actual:   x01  Estado actual:  q1 Valor actual:  2
Cinta actual:   x01  Estado actual:  q1 Valor actual:  3
Cinta actual:   x0x  Estado actual:  q3 Valor actual:  2
Cinta actual:   x0x  Estado actual:  q3 Valor actual:  1
Cinta actual:   x0x  Estado actual:  q3 Valor actual:  0
Cinta actual:   x0x  Estado actual:  init Valor actual:  1
Cinta actual:   x0x  Estado actual:  init Valor actual:  2
Cinta actual:   xxx  Estado actual:  q1 Valor actual:  3
Cinta actual:   xxx  Estado actual:  q1 Valor actual:  4
Cinta actual:   xxx  Estado actual:  E Índice actual:  5
El string tiene distinta cantidad de 0s y 1s.
 xxx
```

-> **No Aceptado**

4) [1 0 1 0]

```
Cinta actual:   1010  Estado actual:  init Valor actual:  1
Cinta actual:   x010  Estado actual:  q2 Valor actual:  2
Cinta actual:   xx10  Estado actual:  q3 Valor actual:  1
Cinta actual:   xx10  Estado actual:  q3 Valor actual:  0
Cinta actual:   xx10  Estado actual:  init Valor actual:  1
Cinta actual:   xx10  Estado actual:  init Valor actual:  2
Cinta actual:   xx10  Estado actual:  init Valor actual:  3
Cinta actual:   xxx0  Estado actual:  q2 Valor actual:  4
Cinta actual:   xxxx  Estado actual:  q3 Valor actual:  3
Cinta actual:   xxxx  Estado actual:  q3 Valor actual:  2
Cinta actual:   xxxx  Estado actual:  q3 Valor actual:  1
Cinta actual:   xxxx  Estado actual:  q3 Valor actual:  0
Cinta actual:   xxxx  Estado actual:  init Valor actual:  1
Cinta actual:   xxxx  Estado actual:  init Valor actual:  2
Cinta actual:   xxxx  Estado actual:  init Valor actual:  3
Cinta actual:   xxxx  Estado actual:  init Valor actual:  4
Cinta actual:   xxxx  Estado actual:  init Valor actual:  5
Cinta actual:   xxxx  Estado actual:  final Índice actual:  6
El string tiene la misma cantidad de 0s y 1s.
 xxxx
```

-> **Aceptado**

Código con comentarios

```
class Tape(object):
    blank_symbol = " " # Símbolo en blanco (marca el inicio y final de la cinta)

    def __init__(self, tape_string=""):
        self.tape = list(tape_string) # Convierte la cadena en una lista de caracteres

    def __str__(self):
        s = ""
        for i in range(len(self.tape)):  # Itera por la cinta completa
            s += self.tape[i]  # Añade cada elemento a la cadena
        return s

    def __getitem__(self, index):
        if 0 <= index < len(self.tape):  # Checa que el índice sea válido dentro de la longitud de
la cinta
            return self.tape[index] # Regresa el valor en el índice
        return self.blank_symbol # Regresa el símbolo en blanco si el índice no es válido

    def __setitem__(self, pos, char):
        if 0 <= pos < len(self.tape):  # Checa que el índice sea válido dentro de la longitud de la
cinta
            self.tape[pos] = char # Cambia el valor en el índice
        else:
            # Caso en el que el índice está fuera de la longitud de la cinta
            self.tape.extend([self.blank_symbol] * (pos - len(self.tape) + 1)) # Extiende la cinta
con el símbolo en blanco
            self.tape[pos] = char  # Cambia el valor en la nueva posición

class TuringMachine(object):
    # Inicializa la máquina de Turing con la cinta, el estado actual, la función de transición y los
estados finales
    def __init__(self, tape="", current_state='init', transition_function=None, final_states=None):
        self.tape = Tape(tape)
        self.current_state = current_state
        self.transition_function = transition_function if transition_function else {}
        self.final_states = final_states if final_states else set()
        self.head = 1

    # Realiza un paso en la máquina de Turing
    def step(self):
        # Imprime la cinta, el estado actual y la cabeza
        print(f"Cinta actual: ", self.tape, "Estado actual: ", self.current_state, "Valor actual: ",
self.head)

        char = self.tape[self.head] # Obtiene el valor en la cabeza de la cinta
        action = self.transition_function.get((self.current_state, char)) # Obtiene la acción en la
función de transición
        # Si hay una acción en la función de transición, se realiza
        if action:
            self.tape[self.head] = action['write']
            self.current_state = action['next_state']
            self.head += action['move']

    # Ejecuta la máquina de Turing y valida el resultado
    def execute(self):
        while self.current_state not in self.final_states and self.head < len(str(self.tape)):
            self.step()
            if self.current_state == "E":
                break
            elif self.tape[self.head] != "0" and self.tape[self.head] != "1" and
self.tape[self.head] != " " and self.tape[self.head] != "x":
```

```
                self.current_state = "E"
                break

        # Imprime la cinta, el estado actual y la cabeza final
        print(f"Cinta actual: ", self.tape, "Estado actual: ", self.current_state, "Índice actual:
", self.head)

        if self.current_state == "E":
            print("El string tiene distinta cantidad de 0s y 1s.")

        elif self.current_state in self.final_states and self.head == len(str(self.tape)):
            print("El string tiene la misma cantidad de 0s y 1s.")

        elif self.current_state not in self.final_states and self.head >= len(str(self.tape)):
            print("El string tiene distinta cantidad de 0s y 1s.")

        else:
            print("El string tiene distinta cantidad de 0s y 1s.")
    # Regresa la cinta como una cadena
    def get_tape(self):
        return str(self.tape)
initial_state = {"init"}
final_states = {"final"}
transition_function = {
    ("init", "x"): {"write": "x", "next_state": "init", "move": 1},
    ("init", "0"): {"write": "x", "next_state": "q1", "move": 1},
    ("init", "1"): {"write": "x", "next_state": "q2", "move": 1},
    ("init", " "): {"write": " ", "next_state": "final", "move": 1},

    ("q1", "x"): {"write": "x", "next_state": "q1", "move": 1},
    ("q1", "0"): {"write": "0", "next_state": "q1", "move": 1},
    ("q1", "1"): {"write": "x", "next_state": "q3", "move": -1},
    ("q1", " "): {"write": " ", "next_state": "E", "move": 1},

    ("q2", "x"): {"write": "x", "next_state": "q2", "move": 1},
    ("q2", "1"): {"write": "1", "next_state": "q2", "move": 1},
    ("q2", "0"): {"write": "x", "next_state": "q3", "move": -1},
    ("q2", " "): {"write": " ", "next_state": "E", "move": 1},

    ("q3", "0"): {"write": "0", "next_state": "q3", "move": -1},
    ("q3", "1"): {"write": "1", "next_state": "q3", "move": -1},
    ("q3", "x"): {"write": "x", "next_state": "q3", "move": -1},
    ("q3", " "): {"write": " ", "next_state": "init", "move": 1}
}

t = TuringMachine(" 001100110 ",
                  current_state="init",
                  final_states=final_states,
                  transition_function=transition_function)

t.execute()
print(t.get_tape())
```