Theory of Computation

[CSE3702]

Submitted by

Godavarthi Sai Nikhil

210214

Department of Computer Science and Engineering

School of Engineering and Technology

BML Munjal University

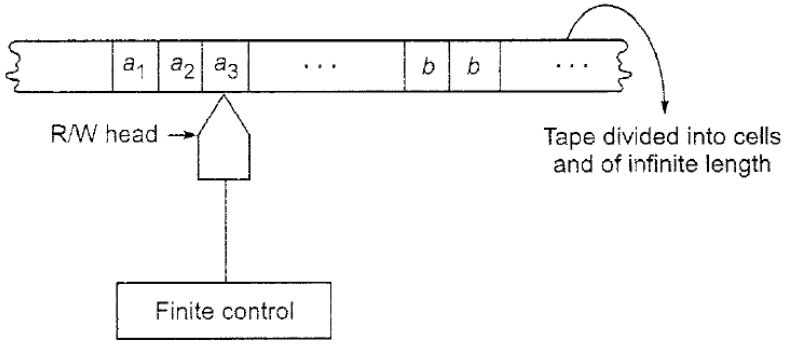
May 2024

1. Problem Presentation:

1.1 Turing Machine (TM):

A Turing Machine (TM) is a theoretical model of computation consisting of a tape divided into cells, a read/write head that can move left or right along the tape, and a finite set of states. The TM operates based on a transition function, which dictates how it changes states and modifies the symbols on the tape in response to its current state and the symbol being read from the tape.

The TM's operation involves moving between states according to the transition function, reading symbols from the tape, writing new symbols onto the tape, and moving the head left or right. It continues this process until it reaches a designated accept state, indicating that it has accepted the input string, or a reject state, indicating that it has rejected the input.



1.2 The components of TM:

A Turing Machine (TM) is composed of several essential components:

1. **Tape:** A potentially infinite tape divided into cells, each of which can hold a symbol from a finite alphabet.
2. **Head:** A read/write pointer that can move left or right along the tape.
3. **State:** A finite set of states in the TM.
4. **Transition Function:** A function that determines the next state and the symbol written on the tape based on the current state and the symbol read from the tape.

$$\mathbf{M} = (\boldsymbol{Q}, \boldsymbol{\varepsilon}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{q_0}, \boldsymbol{b}, \boldsymbol{F})$$

Where,

ε *is input symbols*

γ *is Finite non empty set of tape symbols*

b is a blank tape symbol

q_0 *is intial state*

δ *is the transition*

F is a set of finite states

1.3 Universal Turing Machine (UTM):

A Universal Turing Machine (UTM) is a computational device with the capability to execute any Turing Machine (TM) by utilizing the description of that TM and an input string.

The construction of a UTM involves the encoding of states, symbols, and transitions of the TM to be simulated, enabling the UTM to replicate its behavior on a provided input.

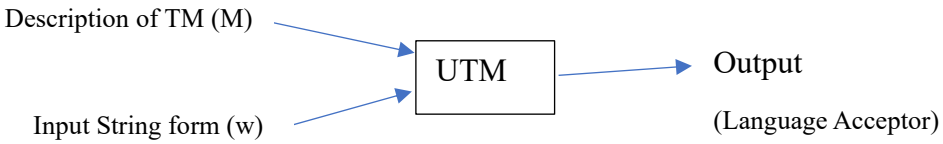
By analyzing the encoded information, the UTM can determine whether the simulated TM accepts or rejects the input string and then halts accordingly, thereby reflecting the behavior of the original TM.

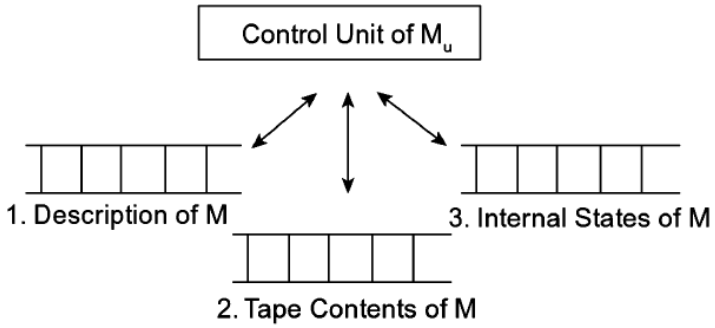
1.4 The components of UTM:

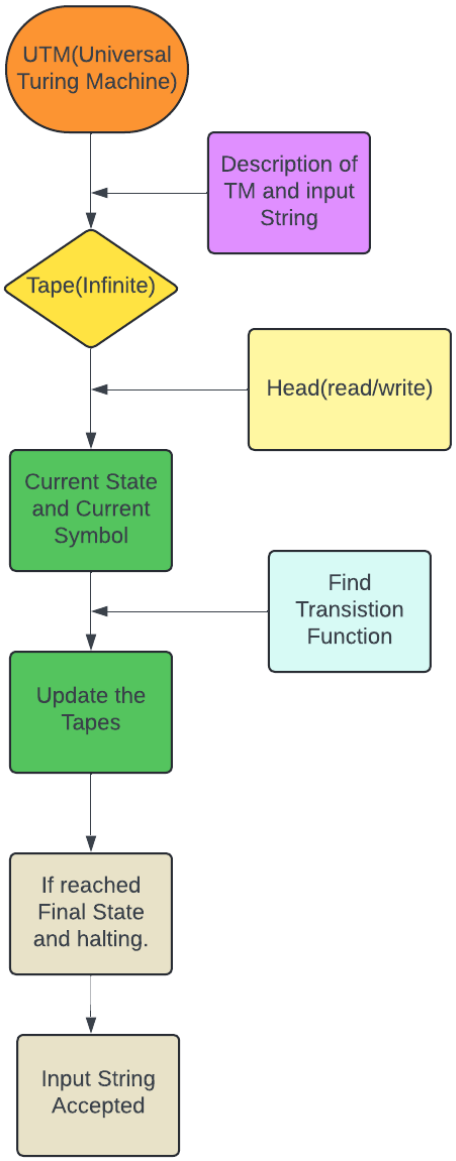
A universal TM, denoted as Mu, takes as input the description of any TM M and a string w, allowing it to simulate the computation of M for input w. The UTM encodes each transition in a binary format. Each transition is represented as a sequence of binary digits, with specific segments indicating the current state, input symbol, next state, new symbol, and direction to move the tape head. For example, a transition $\boldsymbol{\delta(q_i, a_j) = (q_k, a_i, D_m)}$ is encoded as **0^i10^j10^k10^l10^m**, where k and m represent the binary encoding of states and directions, respectively. The entire transition is then represented as a concatenated sequence of these encoded segments, separated by a delimiter, such as '1'. The binary code for the TM M with transitions t1, t2, ..., tn is represented as **111t111t211t311...11tn111**.
When verifying a string, the problem is represented as a tuple <M,w>, where M is the definition of TM and w is the input string.

The UTM's transition function is designed to simulate the behavior of the given TM. It considers the TM's state transitions and updates the UTM's state and tape accordingly.

1.5 Diagrams showing the working of the UTM and its representation:







The Universal Turing Machine (UTM) has three tapes:

Tape 1: This tape contains the encoded description of the Turing Machine M. The description includes details like states, transitions, and symbols used by M. It's represented in a binary format, where each transition is separated by a delimiter (usually '111').

Tape 2: Here lies the input string provided to the TM M. This tape holds the data that M will process according to its programmed rules.

Tape 3: This tape displays the internal states of TM M as it progresses through its computation. It provides a snapshot of M's current state during each step of its operation.

1.6 Implementation:

Let's take an example

States (Q): {q0, q1, q2}

Input Symbols (Σ): {0, 1}

Tape Symbols (Γ): {b, 0, 1}

Initial State – q0 and Final State – q1

Transition Function (δ):

- δ (q0, 0) = (q1, 1, R)
- δ (q0, 1) = (q0, 0, R)
- δ (q0, b) = (q2, b, R)

Input String: 101

Turing Machine (TM) Process:

1. **Input Initialization:**
   The input string is represented on a tape of length 8, initialized with blanks ('b').
   Tape: 1 0 1 b b b b b b
   The Head Position starts at index 0.
2. **Transition Evaluation:**
   Initially, the current symbol and state are 1 and q0. We search for a transition rule for the current state **q0** and symbol **1** in the transition table.
   If a matching transition is found:
   Transition: **δ (q0, 1) = (q0, 0, R)**
   This transition tells us to move to state q0, write symbol 0, and move the head to the Right Direction.
3. **State Update and Tape Modification:**
   Update the current state to q0.
   Write the new symbol 0 at the current head position (index 0).
   Move the head to the right, placing it at index 1.
4. **Updated Tape State:**
   The tape is now updated with the new symbol and head position.
   Tape: 0 0 1 b b b b b b
   The head position is now at index 1.
5. **Iteration and Final State Check:**
   Repeat steps 2-4 for subsequent symbols and transitions until the final state is reached.
   If the final state is reached, and the tape content matches the expected output, the input string is accepted by the TM.

Encoding Process:

We encode the TM description and input string and pass it to UTM as Input. we undertake an encoding process. This encoding involves representing the TM's states, tape symbols, directions, and transitions in a format that the UTM can interpret.

1. **Encoding States:**
   Each state of the TM is represented by a binary string, starting with '0' for the initial state and increasing in binary order for subsequent states (e.g., '00', '000', etc.).
2. **Encoding Tape Symbols:**
   Tape symbols are encoded using binary strings, with '0' representing the blank symbol 'b', '00' for '0', and '000' for '1'.
3. **Encoding Directions:**
   Directions 'L' (left) and 'R' (right) are encoded as '0' and '00'.
4. **Delimiter:**
   Delimiters are inserted between different encoded components (states, symbols, directions) to aid in parsing by the UTM.
5. **Encoding Input String:**
   The input string provided to the TM is encoded using the representations defined above.
   The input string is **'101'**, it would be encoded as **'00010010001'**.
6. **Encoding Transitions:**
   Transitions of the TM **(δ (q, a) = (p, b, D))** are encoded by concatenating the binary representations of their components, separated by delimiters.
   Here, 'q' represents the current state, 'a' the input symbol, 'p' the next state, 'b' the symbol to write, and 'D' the direction.
   For example, the transition **δ (q0, 1) = (q0, 0, R)** would be encoded as **'0100010100100'**.

Working of UTM:

UTM operates with three tapes: Tape 1 contains the encoded description of the TM, Tape 2 holds the encoded input string, and Tape 3 has the internal states of the TM.

1. **Initialization:** The UTM starts with the head at index 0 on Tape 1 and the initial state set to q0.
2. **Transition finding:** It searches Tape 1 for the encoded transition related to q0 and symbol 0.
3. **Symbol Update:** The UTM updates the current symbol and state based on the transition details.
4. **Head Movement:** Following the direction from the transition, the head moves left or right on Tape 2.
5. **Halt Condition:** The UTM continues processing until it reaches a halting state.
6. **Result:** If the final state matches any predefined final states, the UTM accepts the input string otherwise, it rejects it.

Both the TM and UTM accept the input string **101** in this example, which shows the UTM's ability to mimic the behavior of the TM.

1.7 Complexity and Applications:

- UTMs have the theoretical capability to simulate any modern computer. However, practical limitations such as finite tape size and time constraints often render them impractical for complex computations.
- The Halting Problem, an undecidable issue in computer science, is crucial in Turing machine theory. It highlights the limitations of computation and the solvability of algorithms.

2. Explanation of the Data Structures:

The data structures used were set, Hashmap, strings, and list

Set:

input states, symbols, and final states are implemented by using a set data structure. The'set' data structure ensures the ease of checking membership.

In the Turing Machine (TM) class:
self.states = possible_states

self.input_symbols = input_symbols

self.final_states = final_states

HashMap:

Hashmaps were employed to store transitions and their details in the examples selected for the assignment. Additionally, they were utilized to store encoded representations of states, tape symbols, and directions as key-value pairs.

in the assignment: self.transition_function.get((self.current_state, current_symbol)) retrieves the transition details based on the current state and symbol from the input string.

In the TransitionTable class:

transition_function = {(rule.current_state, rule.input_symbol): (rule.next_state, rule.input_symbol_value, rule.direction) for rule in transition_rules}

In the Encoding class:

self.encoded_states[state] = '0' * (index + 1)

self.encoded_tape_symbols[tape_symbol] = '0' + '0' * index

Lists:

It is a collection of items that are ordered and mutable. It allows storing multiple items in a single variable, accessed by indexing.

Lists are used to define the tape of the Turing machine.

Lists are a convenient and mutable way to store objects, allowing easy modification like adding, removing, and altering elements while preserving insertion order. They dynamically adjust in size to hold data, maintain duplicates, and offer straightforward manipulation.

Tape (Tape of the Turing machine is represented in the list of the symbols being read or written by the machine)

In the TuringMachine class:

self.tape_symbols = ['b'] + input_symbols

3. Results:

Example1:

For program the TM and UTM I have taken Example 9.5 from the KLP Mishra TextBook

Design a Turning machine to recognize all strings consisting of an even number of 1's.

In the program, the user need to enter a .txt file which contains the Transition table for the TM

Example1.txt is

q1,-,bRq2

q2,-,bRq1

Output from the Textbook:

$$M = (\{q_1, q_2\}, \{1, b\}, \{1, b\}, \delta, q, b, \{q_1\})$$

where δ is defined by Table 9.3.

TABLE 9.3 Transition Table for Example 9.5

Present state	1
$\rightarrow(q_1)$	bq_2R
q_2	bq_1R

Let us obtain the computation sequence of 11. Thus, $q_111 \vdash bq_21 \vdash bbq_1$. As q_1 is an accepting state, 11 is accepted. $q_1111 \vdash bq_211 \vdash bbq_11 \vdash bbbq_2$. M halts and as q_2 is not an accepting state, 111 is not accepted by M.

Output from the code:

```
Enter the name of the .txt file: example1.txt
Enter input symbols separated by spaces: 1
Possible states extracted from the table: ['q1', 'q2']
Enter the initial state: q1
Enter the final state: q1

States = ['q1', 'q2']
Input_symbols = ['1']
Tape_symbols = ['b', '1']
Transition_function = {
('q1', '1'): ('q2','b', 'R'),
('q2', '1'): ('q1','b', 'R'),
}
Initial_state = q1
Blank_symbol = b
Final_states = {'q1'}
```

```
Enter the input string: 1111
Original Tape: ['1', '1', '1', '1', 'b', 'b', 'b', 'b'] with tapehead at index 0 on state q1
['b', '1', '1', '1', 'b', 'b', 'b', 'b'] with tapehead at index 1 on state q2
['b', 'b', '1', '1', 'b', 'b', 'b', 'b'] with tapehead at index 2 on state q1
['b', 'b', 'b', '1', 'b', 'b', 'b', 'b'] with tapehead at index 3 on state q2
['b', 'b', 'b', 'b', 'b', 'b', 'b', 'b'] with tapehead at index 4 on state q1
Input string accepted by TM.
```

```
State Encoding: {'q1': '0', 'q2': '00'}
Tape Symbol Encoding: {'b': '0', '1': '00'}
Encoded transitions: 11101001001010011001001010100111
Input String:  1111
Encoded input string: 001001001001
```

```
Initial Configuration:
Tape 1 (Encoded Transitions): 11101001001010011001001010100111
Tape 2 (Encoded Input String): 001001001001
Tape 3 (Internal State): 0
Current symbol:  00
Transition Found for the current symbol:
Prefix of current state and current symbol:  110100 Suffix:  0010100 Transition:  11010010010100
Current Configuration:
Tape 1 (Encoded Transitions): 11101001001010011001001010100111
Tape 2 (Encoded Input String): 01001001001
Tape 3 (Internal State): 00
index:  2
Current symbol:  00
Transition Found for the current symbol:
Prefix of current state and current symbol:  1100100 Suffix:  010100 Transition:  11001001010100
Current Configuration:
Tape 1 (Encoded Transitions): 11101001001010011001001010100111
Tape 2 (Encoded Input String): 0101001001
Tape 3 (Internal State): 0
index:  4
Current symbol:  00
Transition Found for the current symbol:
Prefix of current state and current symbol:  110100 Suffix:  0010100 Transition:  11010010010100
Current Configuration:
Tape 1 (Encoded Transitions): 11101001001010011001001010100111
Tape 2 (Encoded Input String): 010101001
Tape 3 (Internal State): 00
index:  6
Current symbol:  00
Transition Found for the current symbol:
Prefix of current state and current symbol:  1100100 Suffix:  010100 Transition:  11001001010100
Current Configuration:
Tape 1 (Encoded Transitions): 11101001001010011001001010100111
Tape 2 (Encoded Input String): 01010101
Tape 3 (Internal State): 0
index:  8
Current symbol:
Reached final state. Halting and accepting.
Input string accepted by both Turing machine and UTM.
```

One example of the rejected string is “111”

```
Enter the input string: 111
Original Tape: ['1', '1', '1', 'b', 'b', 'b', 'b', 'b'] with tapehead at index 0 on state q1
['b', '1', '1', 'b', 'b', 'b', 'b', 'b'] with tapehead at index 1 on state q2
['b', 'b', '1', 'b', 'b', 'b', 'b', 'b'] with tapehead at index 2 on state q1
['b', 'b', 'b', 'b', 'b', 'b', 'b', 'b'] with tapehead at index 3 on state q2
Input string rejected by TM.
```

```
State Encoding: {'q1': '0', 'q2': '00'}
Tape Symbol Encoding: {'b': '0', '1': '00'}
Encoded transitions: 11101001001010011001001010100111
Input String:  111
Encoded input string: 001001001
```

```
Initial Configuration:
Tape 1 (Encoded Transitions): 11101001001010011001001010100111
Tape 2 (Encoded Input String): 001001001
Tape 3 (Internal State): 0
Current symbol:  00
Transition Found for the current symbol:
Prefix of current state and current symbol:  110100 Suffix:  0010100 Transition:  11010010010100
Current Configuration:
Tape 1 (Encoded Transitions): 11101001001010011001001010100111
Tape 2 (Encoded Input String): 01001001
Tape 3 (Internal State): 00
index:  2
Current symbol:  00
Transition Found for the current symbol:
Prefix of current state and current symbol:  1100100 Suffix:  010100 Transition:  11001001010100
Current Configuration:
Tape 1 (Encoded Transitions): 11101001001010011001001010100111
Tape 2 (Encoded Input String): 0101001
Tape 3 (Internal State): 0
index:  4
Current symbol:  00
Transition Found for the current symbol:
Prefix of current state and current symbol:  110100 Suffix:  0010100 Transition:  11010010010100
Current Configuration:
Tape 1 (Encoded Transitions): 11101001001010011001001010100111
Tape 2 (Encoded Input String): 010101
Tape 3 (Internal State): 00
index:  6
Current symbol:
Input string rejected by both Turing machine and UTM.
```

Example 2:

I have taken Example 9.2 from the KLP Mishra Text Book

TABLE 9.1 Transition Table of a Turing Machine

Present state	*Tape symbol*		
	b	0	1
$\rightarrow q_1$	$1Lq_2$	$0Rq_1$	
q_2	bRq_3	$0Lq_2$	$1Lq_2$
q_3		bRq_4	bRq_5
q_4	$0Rq_5$	$0Rq_4$	$1Rq_4$
(q_5)	$0Lq_2$		

Output from the Textbook:

For the input string $00b$, we get the following sequence:

$$q_100b \vdash 0q_10b \vdash 00q_1b \vdash 0q_201 \vdash q_2001$$
$$\vdash q_2b001 \vdash bq_3001 \vdash bbq_401 \vdash bb_0q41 \vdash bb_01q_4b$$
$$\vdash bb010q_5 \vdash bb01q_200 \vdash bb0q_2100 \vdash bbq_20100$$
$$\vdash bq_2b0100 \vdash bbq_30100 \vdash bbbq_4100 \vdash bbb_1q_400$$
$$\vdash bbb10q_40 \vdash bbb100q_4b \vdash bbb1000q_5b$$
$$\vdash bbb100q_200 \vdash bbb10q_2000 \vdash bbb1q_20000$$
$$\vdash bbbq_210000 \vdash bbq_2b10000 \vdash bbbq_310000 \vdash bbbbq_50000$$

Output from the code:

```
Enter the name of the .txt file: example.txt
Enter input symbols separated by spaces: 0 1
Possible states extracted from the table: ['q1', 'q2', 'q3', 'q4', 'q5']
Enter the initial state: q1
Enter the final state: q5

States = ['q1', 'q2', 'q3', 'q4', 'q5']
Input_symbols = ['0', '1']
Tape_symbols = ['b', '0', '1']
Transition_function = {
('q1', 'b'): ('q2','1', 'L'),
('q1', '0'): ('q1','0', 'R'),
('q2', 'b'): ('q3','b', 'R'),
('q2', '0'): ('q2','0', 'L'),
('q2', '1'): ('q2','1', 'L'),
('q3', '0'): ('q4','b', 'R'),
('q3', '1'): ('q5','b', 'R'),
('q4', 'b'): ('q5','0', 'R'),
('q4', '0'): ('q4','0', 'R'),
('q4', '1'): ('q4','1', 'R'),
('q5', 'b'): ('q2','0', 'L'),
}
Initial_state = q1
Blank_symbol = b
Final_states = {'q5'}
```

```
Enter the input string: 00b
Original Tape: ['0', '0', 'b', 'b', 'b', 'b', 'b', 'b'] with tapehead at index 0 on state q1
['0', '0', 'b', 'b', 'b', 'b', 'b', 'b'] with tapehead at index 1 on state q1
['0', '0', 'b', 'b', 'b', 'b', 'b', 'b'] with tapehead at index 2 on state q1
['0', '0', '1', 'b', 'b', 'b', 'b', 'b'] with tapehead at index 1 on state q2
['0', '0', '1', 'b', 'b', 'b', 'b', 'b'] with tapehead at index 0 on state q2
Tape head moved to index -1. Adding 'b' at the beginning and popping from the end.
['b', '0', '0', '1', 'b', 'b', 'b', 'b'] with tapehead at index 0 on state q2
['b', '0', '0', '1', 'b', 'b', 'b', 'b'] with tapehead at index 1 on state q3
['b', 'b', '0', '1', 'b', 'b', 'b', 'b'] with tapehead at index 2 on state q4
['b', 'b', '0', '1', 'b', 'b', 'b', 'b'] with tapehead at index 3 on state q4
['b', 'b', '0', '1', 'b', 'b', 'b', 'b'] with tapehead at index 4 on state q4
['b', 'b', '0', '1', '0', 'b', 'b', 'b'] with tapehead at index 5 on state q5
['b', 'b', '0', '1', '0', '0', 'b', 'b'] with tapehead at index 4 on state q2
['b', 'b', '0', '1', '0', '0', 'b', 'b'] with tapehead at index 3 on state q2
['b', 'b', '0', '1', '0', '0', 'b', 'b'] with tapehead at index 2 on state q2
['b', 'b', '0', '1', '0', '0', 'b', 'b'] with tapehead at index 1 on state q2
['b', 'b', '0', '1', '0', '0', 'b', 'b'] with tapehead at index 2 on state q3
['b', 'b', 'b', '1', '0', '0', 'b', 'b'] with tapehead at index 3 on state q4
['b', 'b', 'b', '1', '0', '0', 'b', 'b'] with tapehead at index 4 on state q4
['b', 'b', 'b', '1', '0', '0', 'b', 'b'] with tapehead at index 5 on state q4
['b', 'b', 'b', '1', '0', '0', 'b', 'b'] with tapehead at index 6 on state q4
['b', 'b', 'b', '1', '0', '0', '0', 'b'] with tapehead at index 7 on state q5
['b', 'b', 'b', '1', '0', '0', '0', '0'] with tapehead at index 6 on state q2
['b', 'b', 'b', '1', '0', '0', '0', '0'] with tapehead at index 5 on state q2
['b', 'b', 'b', '1', '0', '0', '0', '0'] with tapehead at index 4 on state q2
['b', 'b', 'b', '1', '0', '0', '0', '0'] with tapehead at index 3 on state q2
['b', 'b', 'b', '1', '0', '0', '0', '0'] with tapehead at index 2 on state q2
['b', 'b', 'b', '1', '0', '0', '0', '0'] with tapehead at index 3 on state q3
['b', 'b', 'b', 'b', '0', '0', '0', '0'] with tapehead at index 4 on state q5
Input string accepted by TM.
```

```
State Encoding: {'q1': '0', 'q2': '00', 'q3': '000', 'q4': '0000', 'q5': '00000'}
Tape Symbol Encoding: {'b': '0', '0': '00', '1': '000'}
Encoded transitions: 1110101001000101110100101001001100101000101001100100100100101100100010010
Input String:  00b
Encoded input string: 00100101
```

```
Initial Configuration:
Tape 1 (Encoded Transitions): 11101010010001011010010100100110010100010100110010010010011010001001000101100
Tape 2 (Encoded Input String): 00100101
Tape 3 (Internal State): 0
Current symbol:  00
Transition Found for the current symbol:
Prefix of current state and current symbol:  110100 Suffix:  0100100 Transition:  11010010100100
Current Configuration:
Tape 1 (Encoded Transitions): 11101010010001011010010100100110010100010100110010010010011010001001000101100
Tape 2 (Encoded Input String): 00100101
Tape 3 (Internal State): 0
index:  3
Current symbol:  00
Transition Found for the current symbol:
Prefix of current state and current symbol:  110100 Suffix:  0100100 Transition:  11010010100100
Current Configuration:
Tape 1 (Encoded Transitions): 11101010010001011010010100100110010100010100110010010010011010001001000101100
Tape 2 (Encoded Input String): 00100101
Tape 3 (Internal State): 0
index:  6
Current symbol:  0
Transition Found for the current symbol:
Prefix of current state and current symbol:  11010 Suffix:  00100010 Transition:  11010100100010
Current Configuration:
Tape 1 (Encoded Transitions): 11101010010001011010010100100110010100010100110010010010011010001001000101100
Tape 2 (Encoded Input String): 0010010001
Tape 3 (Internal State): 00
Input string accepted by both Turing machine and UTM.
```

One example of the rejected string is “11”

```
Enter the input string: 11
Original Tape: ['1', '1', 'b', 'b', 'b', 'b', 'b', 'b'] with tapehead at index 0 on state q1
Input string rejected by TM.
```

```
State Encoding: {'q1': '0', 'q2': '00', 'q3': '000', 'q4': '0000', 'q5': '00000'}
Tape Symbol Encoding: {'b': '0', '0': '00', '1': '000'}
Encoded transitions: 11101010010001011010010100100110010100010100110010010010011010001001000101100
Input String:  11
Encoded input string: 00010001
```

```
Initial Configuration:
Tape 1 (Encoded Transitions): 11101010010001011010010100100110010100010100110010010
Tape 2 (Encoded Input String): 00010001
Tape 3 (Internal State): 0
Current symbol:  000
No transition found for the current symbol and state. Halting and rejecting.
Input string rejected by both Turing machine and UTM.
```

Colab Link:

https://colab.research.google.com/drive/1jbKiyeR0cxqs0HjMOF_dl4UECSJIbg4O?usp=sharing

References

K. V. N Sunitha, N. K. (n.d.). *Formal Languages and Automata Theory.* Hyderabad : Pearson.

K.K.P Mishra, N. C. (2008). *Theory of Computer Science Automata, Languages, and Computation Third Edition.* New Delhi: Prentice Hall of India Private Limited.