18CSC301T – Formal Language and Automata

BINARY ENCODING OF TURING MACHINE

BY

RA2111026010127 Cheedella S V Abhinava Sai

RA2111026010117 Thangala Nithin Kumar

RA2111026010135 Harsha Shiva Shankar

Objective

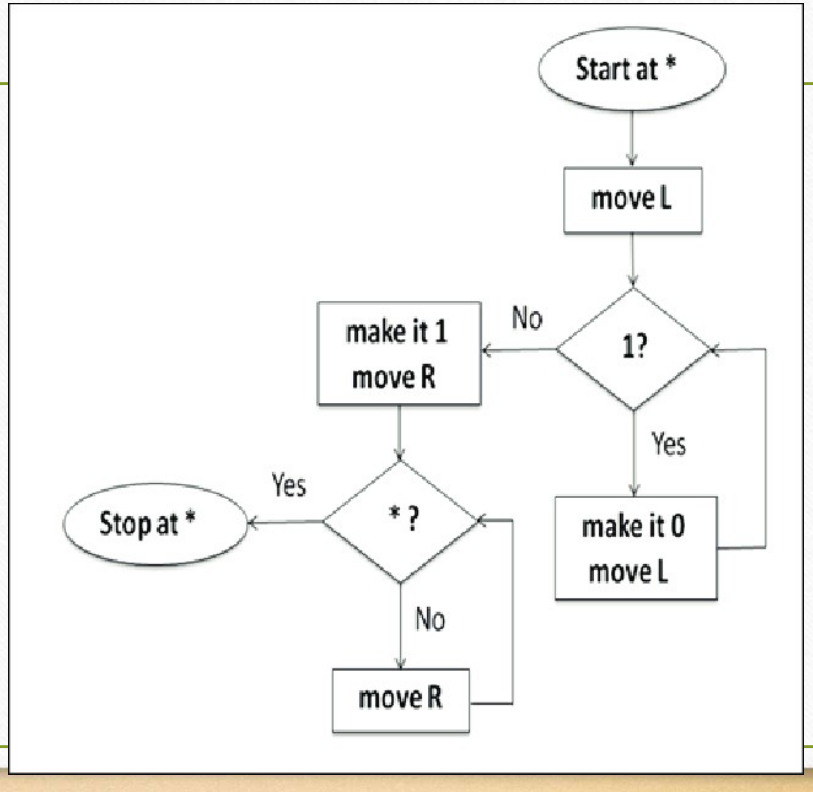
- The objective of using binary encoding in a Turing machine is primarily to represent and manipulate information or data in a more compact and standardized form.

Application

- Binary encoding in Turing machines has various applications across different fields and problem-solving domains:
- **Data Processing and Computation**
- **Algorithm Development and Analysis**
- **Data Compression and Transmission**
- **Computer Graphics and Image Processing**
- **Cryptography**

Flow Chart

- Diagrammatic representation of Binary Encoding of a Turing Machine



Procedure

- Encode the symbols that the Turing machine uses on its tape. For instance, if the machine has 'm' symbols, assign a unique binary code to each symbol
- Encode the transition rules for the Turing machine. These rules define what action the machine should take when in a particular state, reading a specific symbol. Each rule consists of: Current state, Symbol read, Action
- Represent the current state using its binary code
- Encode the symbol being read on the tape using its binary representation
- Define the action to take when in a certain state and reading a specific symbol
- Concatenate these binary representations for the current state, symbol read, action (write, move, next state) to form a single binary string that represents the transition rule
- This completes the binary encoding of a Turing machine

Example Problem

- Simple example of binary encoding of a Turing machine

Codes for Turing Machine

To represent a TM $M = (Q, \{0,1\}, \Gamma, \delta, q_1, B, F)$ as a binary string, first assign integers to states, tape symbols and directions L and R.

(i) Assume the states are $q_1, q_2, \ldots q_k$ for some k. Using the integers available in the suffix of each state, the string can be represented as $q_1 \rightarrow 0$, $q_2 \rightarrow 00$, $q_3 \rightarrow 000$ etc...

(ii) Assume the tape symbols 0, 1, B are represented as, $X_1, X_2, X_3 \ldots X_n$

Where $X_1 \rightarrow 0$
$X_2 \rightarrow 1$
$X_3 \rightarrow B$

(iii) Assume the directions are represented as D_1 and D_2, where.

$L \rightarrow D_1 \rightarrow 0$
$R \rightarrow D_2 \rightarrow 00$

After representing each state, symbol and direction using integers, we can encode the transition function.

$\delta(q_i, X_j) = (q_k, X_l, D_m)$ for some integers i, j, k, l, m.

The encoded string is given by $0^i 1 0^j 1 0^k 1 0^l 1 0^m$.

The code for entire TM consists of all strings, separated by pair of 1's.

$C_1 11 C_2 11 C_3 11 \cdots C_{n-1} 11 C_n$.

$C \rightarrow$ Transition Code.

Problem: obtain the code for $\langle M, 1011 \rangle$. Where $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_3\})$ has moves $\delta(q_1, 1) = (q_3, 0, R)$, $\delta(q_3, 0) = (q_1, 1, R)$, $\delta(q_3, 1) = (q_2, 0, R)$, $\delta(q_3, B) = (q_3, 1, L)$

Solution:

Assume the tape symbols be encoded as,

0 — x_1 = 0
1 — x_2 = 00
B — x_3 = 000

Directions can be encoded as, L — D_1 = 0, R — D_2 = 00

Transition are encoded as,

(i) $\delta(q_1, 1) = (q_3, 0, R)$ $C_1 \Rightarrow 0^1 1 0^2 1 0^3 1 0^1 1 0^2$

(ii) $\delta(q_3, 0) = (q_1, 1, R)$ $C_2 \Rightarrow 0^3 1 0^1 1 0^1 1 0^1 1 0^2$

(iii) $\delta(q_3, 1) = (q_2, 0, R)$ $C_3 \Rightarrow 0^3 1 0^2 1 0^2 1 0^1 1 0^2$

(iv) $\delta(q_3, B) = (q_3, 1, L)$ $C_4 \Rightarrow 0^3 1 0^3 1 0^3 1 0^2 1 0^1$

The complete code is given by,

$C_1 11 C_2 11 C_3 11 C_4$

$M =$ 0100100010100 11 000101010100 11 000100100101001 11 000100010010

$\therefore$ Code for $\langle M, 1011 \rangle$ is given by

$= \langle$ 0100100010100 11 000101010100 11 000100100101001 11 000100010010, 1011 $\rangle$.

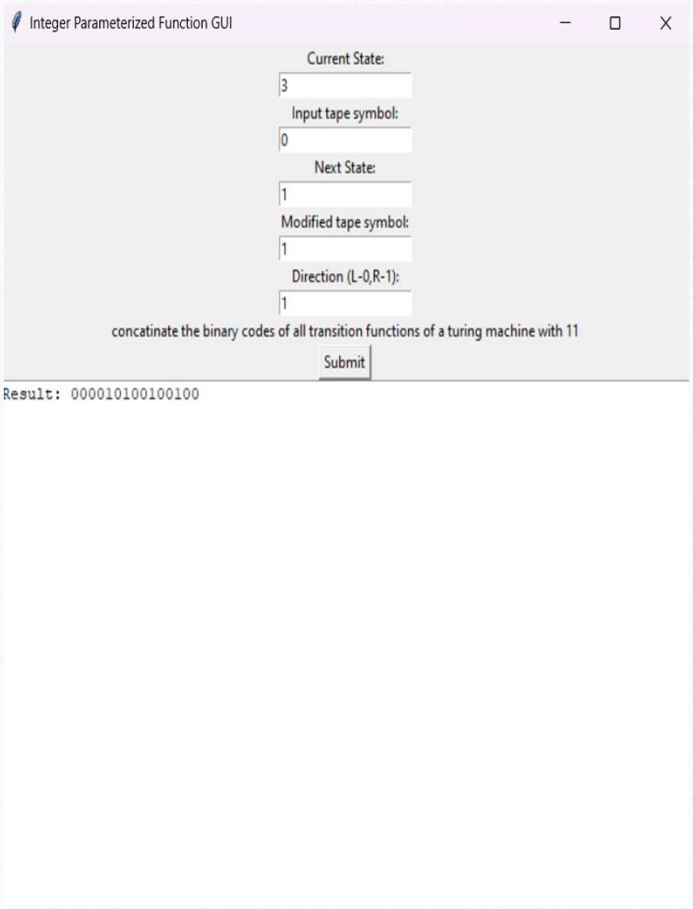
Implementation

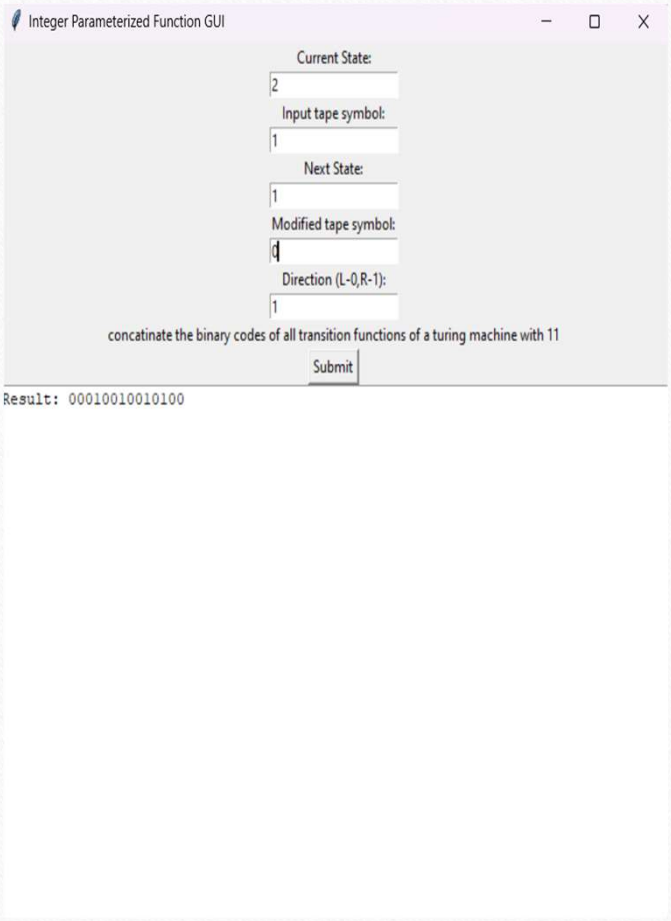
```
import tkinter as tk
def encoding(s1,t1,s2,t2,d):
    if s1==0:
        r1="0"
    elif s1==1:
        r1="00"
    elif s1==2:
        r1="000"
    elif s1==3:
        r1="0000"

    if t1==0:
        r2="0"
    elif t1==1:
        r2="00"
    elif t1==2:
        r2="000"


    if s2==0:
        r3="0"
    elif s2==1:
        r3="00"
    elif s2==2:
        r3="000"
    elif s2==3:
        r3="0000"

    if t2==0:
        r4="0"
    elif t2==1:
        r4="00"
    elif t2==2:
        r4="000"


    if d==0:
        r5="0"
    elif d==1:
        r5="00"
    strings = [r1,r2,r3,r4,r5]
    return "1".join(strings)
```

```
# Function that uses the input parameters
def process_input():
    try:
        param1 = int(input1.get())
        param2 = int(input2.get())
        param3 = int(input3.get())
        param4 = int(input4.get())
        param5 = int(input5.get())

        # Perform some operation using the parameters
        result = encoding(param1,param2,param3,param4,param5)
        result_text.delete(1.0, tk.END)  # Clear previous result
        result_text.insert(tk.END, f"Result: {result}\n")  # Display the result
    except ValueError:
        result_text.delete(1.0, tk.END)  # Clear previous result
        result_text.insert(tk.END, "Invalid input. Please enter integers.\n")

# Create the main application window
app = tk.Tk()
app.title("Integer Parameterized Function GUI")

# Create and place 5 input entry fields
input1_label = tk.Label(app, text="Current State:")
input1_label.pack()
input1 = tk.Entry(app)
input1.pack()

input2_label = tk.Label(app, text="Input tape symbol:")
input2_label.pack()
input2 = tk.Entry(app)
input2.pack()

input3_label = tk.Label(app, text="Next State:")
input3_label.pack()
input3 = tk.Entry(app)
input3.pack()

input4_label = tk.Label(app, text="Modified tape symbol:")
input4_label.pack()
input4 = tk.Entry(app)
input4.pack()
```

```
input5_label = tk.Label(app, text="Direction (L-0,R-1):")
input5_label.pack()
input5 = tk.Entry(app)
input5.pack()

# Create and place a submit button
submit_button = tk.Button(app, text="Submit", command=process_input)
submit_button.pack()

# Create a text box for displaying the result
result_text = tk.Text(app)
result_text.pack()

# Start the GUI application
app.mainloop()
```

Results


Integer Parameterized Function GUI
Current State:
3
Input tape symbol:
0
Next State:
1
Modified tape symbol:
1
Direction (L-0,R-1):
1
concatinate the binary codes of all transition functions of a turing machine with 11
Submit
Result: 000010100100100


Integer Parameterized Function GUI
Current State:
2
Input tape symbol:
1
Next State:
1
Modified tape symbol:
0
Direction (L-0,R-1):
1
concatinate the binary codes of all transition functions of a turing machine with 11
Submit
Result: 00010010010100

Conclusion

- In this project, we successfully implemented a binary encoding of a Turing machine transition function and provided a user-friendly GUI for users to interact with the encoded transition function. This project aimed to make the encoding and manipulation of Turing machine transition functions more accessible and intuitive, and we believe that it has achieved these goals.

Contribution

- PPT – Harsha Shiva Shankar
- Python Code for Binary Encoding – Cheedella S V Abhinava Sai
- GUI Integration – Thangala Nithin Kumar Reddy

- Github lilnk - Abhiat2004/Fla (github.com)