Q7) Design DFA that accept binary string containing both '101' and '110' as substring.

---

1. Introduction

In Formal Language Theory, a Deterministic Finite Automaton (DFA) is a mathematical model used to recognize patterns in strings of symbols. This report discusses the construction of a DFA in JFLAP.

A DFA is a simple machine that reads input symbols one at a time and transitions between states based on the current state and the input symbol. It accepts or rejects the input string based on whether it ends in an "accept" state.

Key Uses:

- **Pattern recognition:** Finding specific patterns in text.
- **Lexical analysis:** Identifying tokens (keywords, identifiers) in code.
- **Protocol validation:** Checking if data conforms to a specific format.

---

2. Problem Statement

Design a single-tape deterministic Turing Machine to accept precisely those strings of the form:

L = {w | w contains both '101' and '110' as substrings}

Any string in LLL consists of:

1. **At least one occurrence of '101'** as a substring.
2. **At least one occurrence of '110'** as a substring.
3. These substrings can occur in **any order**, and they may **overlap**.

For example:

- Strings in LLL: "101110", "110101", "10100110", "1110110","1101".
- Strings not in LLL: "1010", "100110", "111".

This report provides a detailed explanation of the design, implementation, and validation of this DFA using the JFLAP tool. The machine should ensure that the input string contains both '101' and '110' as substrings, in any order or with possible overlap.

Any deviation—whether an incorrect count of characters, an incorrect order, or an absence of required blocks—should lead the machine to reject.

---

3. High-Level Design Approach

Understand the Requirements:

- The DFA must:
  - Detect the substring '101'.
  - Detect the substring '110'.
  - Accept only if **both substrings** are present in the input string.

State Design:

- **Initial State (q0)**:
  - Start of the DFA, no progress toward detecting '101' or '110'.
- **Tracking States (q1, q2, q3)**:
  - Track progress toward '101' and '110' simultaneously.
  - q1: '1' seen, starting detection of both substrings.
  - q2: '10' seen, progressing toward '101'.
  - q3: '11' seen, progressing toward '110'.
- **Detection States (q4, q5)**:
  - q4: '101' detected, now look for '110'.
  - q5: '110' detected, now look for '101'.
- **Intermediate State (q6)**:
  - Handle overlaps between '101' and '110' (e.g., '1101').
- **Final State (q9)**:
  - Accepting state, both substrings detected.

Transitions:

- **Simultaneous Detection**: Use transitions to track prefixes of '101' and '110' concurrently.
- **Overlap Handling**: For cases like '1101', ensure the DFA continues detecting the second substring without restarting.
- **Acceptance**: Transition to q9 only when both substrings are detected.

General Workflow:

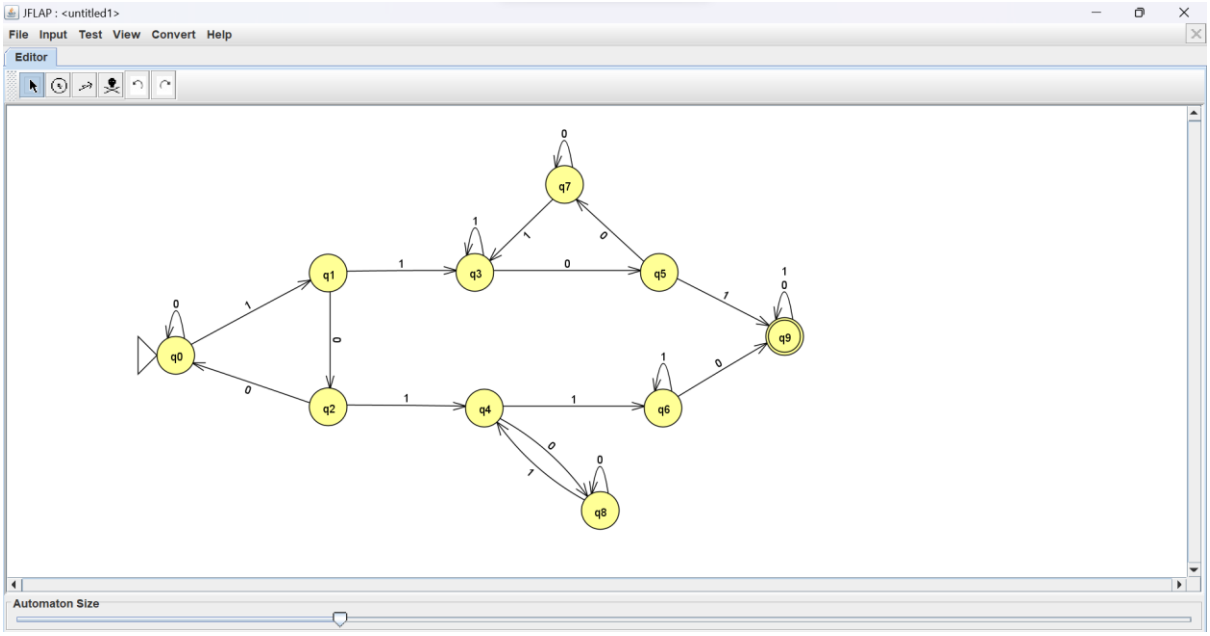
- Start at q0.
- Read the input string symbol by symbol.
- Transition between states based on whether input matches prefixes of '101' or '110'.
- Accept if both substrings are found (reach q9)
- Reject if the input ends without reaching q9.

Efficiency

- The DFA processes the input in **O (n)** time, where n is the length of the string.
- Tracks both substrings simultaneously, minimizing redundant computations.

4. Explanation of States and Transitions

Below is a summary of each state and its purpose. The figure (screenshot in JFLAP) illustrates how these states connect.



States:

- q0: Start state, no progress towards '101' or '110'.
- q1: '1' seen, possible start of '101' or '110'.
- q2: '10' seen, progressing towards '101'.
- q3: '11' seen, progressing towards '110'.
- q4: '101' detected, now searching for '110'.
- q5: '110' detected, now searching for '101'.
- q6: Intermediate state, handling overlaps.
- q9: Final state, both '101' and '110' detected.

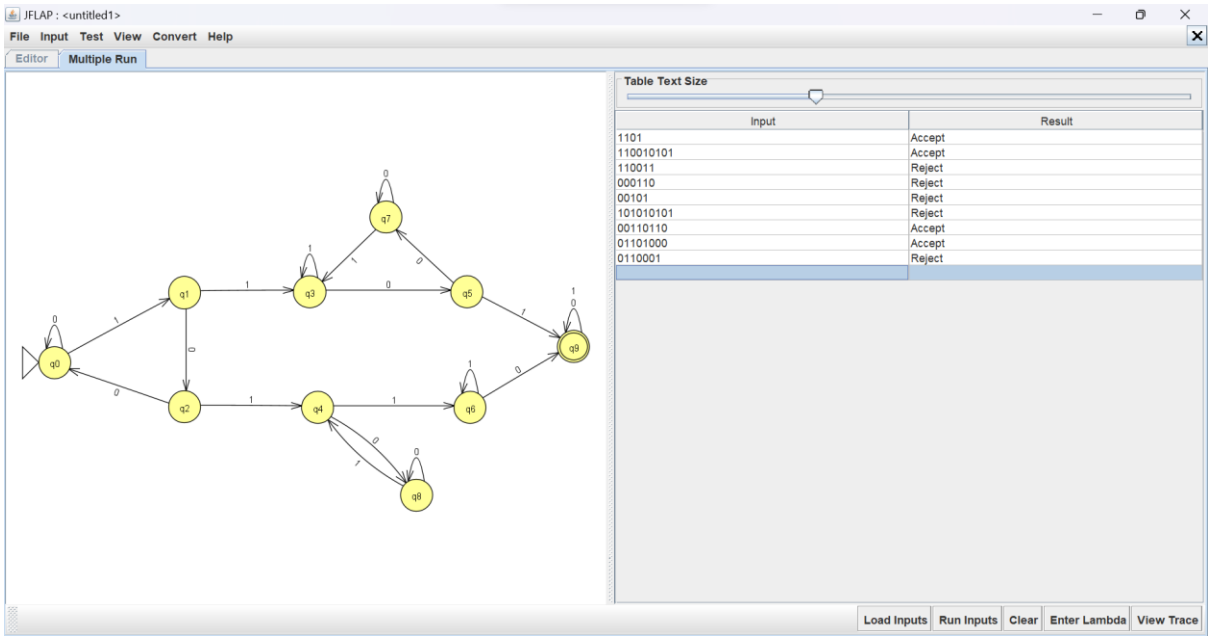
Transitions:

- Track prefixes of '101' and '110' simultaneously.
- Handle overlaps (e.g., '1101' contains both substrings).
- Accept only when both substrings are found (transition to q9).

How the DFA Works:

- The DFA starts at q0 and reads the input string one symbol at a time.
- As it reads the input, it transitions between states based on the prefixes of '101' and '110' it encounters.
- If both '101' and '110' are found, the DFA transitions to q9, which is the accepting state.
- If the input ends without reaching q9, the string is rejected.

5. Sample Traces



5.1. Valid Input Example

Consider the string: 1101

Trace Table for w=1101:

Step	Current State	Input Symbol	Next State	Remark
1	q0	1	q1	Transition on 1
2	q1	1	q3	Transition on 1
3	q3	0	q5	Transition on 0
4	q5	1	q9	Transition on 1

Conclusion: Since the DFA ends in a final state (q9), the string w=1101 is accepted.

5.2. Invalid Input Example

Consider the string: 00101

Trace Table for w=00101:

Step	Current State	Input Symbol	Next State	Remark
1	q0	0	q0	Transition on 0
2	q0	0	q0	Transition on 0
3	q0	1	q1	Transition on 1
4	q1	0	q2	Transition on 0
5	q2	1	q4	Transition on 1

Conclusion: Since the DFA ends in a non-final state (q4), the string w=00101 is **rejected**.

6. Final Representation

The final DFA can be described as: **M = (Q, Σ, δ, q0, F)**

Q = {q0, q1, q2, q3, q4, q5, q6, q7, q8, q9} **Σ** = {0, 1}

q0 is the start state q0 **q9** is the final state F.

δ – Transition Functions:

From q0:	**From q1:**	**From q2:**	**From q3:**
δ(q0,0)=q0	δ(q1,0)=q2	δ(q2,0)=q0	δ(q3,0)=q5
δ(q0,1)=q1	δ(q1,1)=q3	δ(q2,1)=q4	δ(q3,1)=q3
From q4:	**From q5:**	**From q6:**	**From q7:**
δ(q4,0)=q8	δ(q5,0)=q7	δ(q6,0)=q9	δ(q7,0)=q7
δ(q4,1)=q6	δ(q5,1)=q9	δ(q6,1)=q6	δ(q7,1)=q3
From q8:	**From q9:**		
δ(q8,0)=q8	δ(q9,0)=q9		
δ(q8,1)=q4	δ(q9,1)=q9		

Transition Table:

State	Input 0	Input 1
q0	q0	q1
q1	q2	q3
q2	q0	q4
q3	q5	q3
q4	q8	q6
q5	q7	q9
q6	q9	q6
q7	q7	q3
q8	q8	q4
q9	q9	q9

7. Conclusion

The constructed DFA, as implemented in JFLAP, successfully recognizes strings in the language **L = {w | w contains both '101' and '110' as substrings}**

The machine ensures:

- **At least one occurrence of '101'** as a substring.
- **At least one occurrence of '110'** as a substring.
- These substrings can occur in **any order**, and they may **overlap**.
- Immediate rejection of invalid patterns, such as unmatched symbols or incorrect order.

This implementation demonstrates the systematic use of marking, scanning, and state transitions to validate the given language. The behaviour of the machine can be verified through step-by-step execution in JFLAP using both valid and invalid test cases.