THEORY OF AUTOMATA

BSE 8A

SPRING 2025

ASSIGNMENT

Submitted by:

Somana Maqsood(01-131212-031)

Ayesha Malik(01-131212-008)

Zahrah Naveed(01-131212-038)

Submitted to: **Shahid Khan**

Department of Software Engineering

Bahria University H11/4 Campus

GROUP 8

REPORT

B PART

QUESTION 6

ASSIGNMENT 1

6 Language Closure on Operations [25 Points]

Consider following two languages over $\Sigma = \{a, b\}$

$$A = \{w \mid \text{length of w is multiple of 3}\}$$

$$B = \{w \mid \text{length of w is multiple of 2}\}$$

a) Draw 2x DFA that accept A and B [5 Points]

b) Construct DFA that Accepts $\overline{A}$ [5 Points]

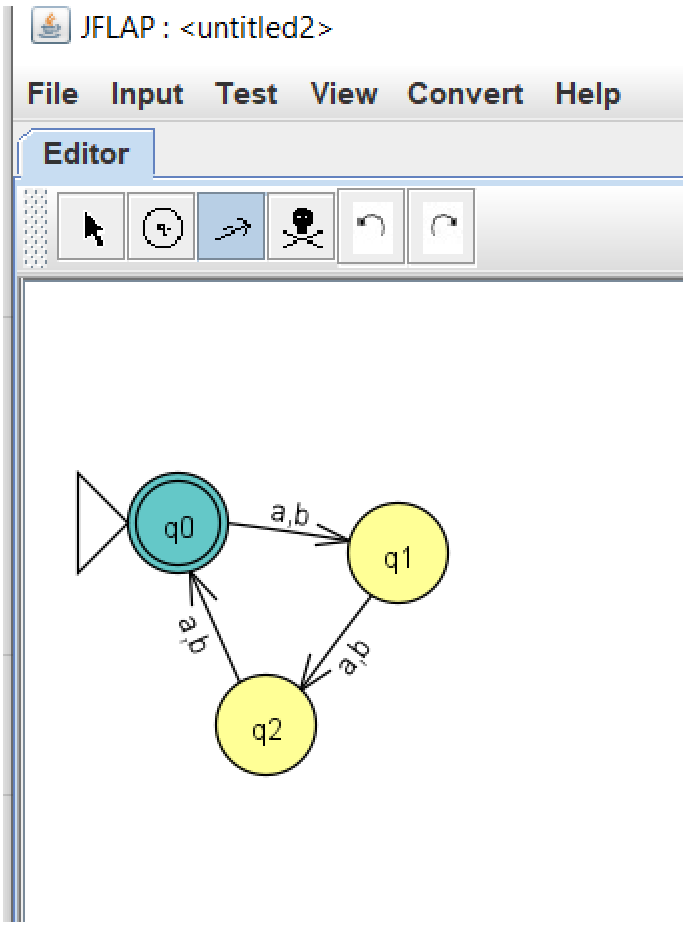
c) Construct DFA for $A \cup B$ [5 Points]

d) construct DFA for $A \cap B$ [5 Points]

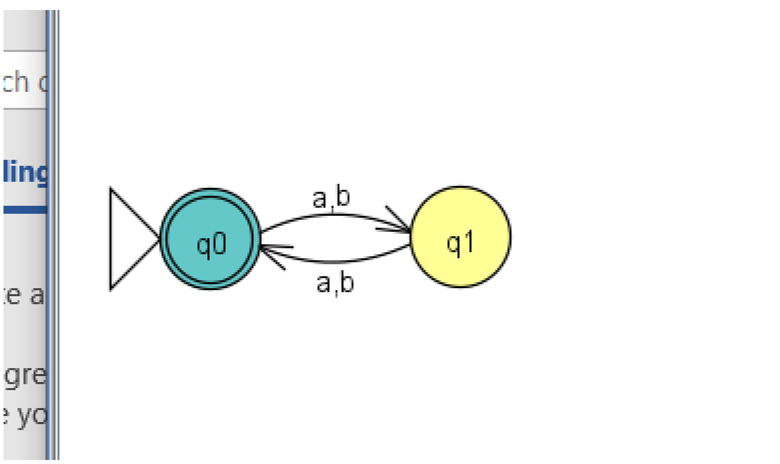
e) construct DFA for $A \setminus B$ [5 Points]

2

PART A(multiple of 3):

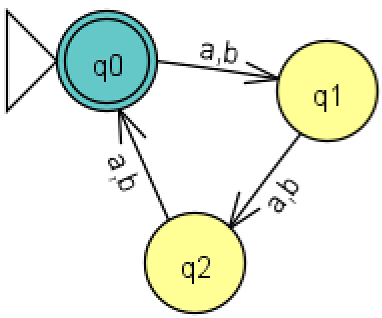


MULTIPLE OF 2



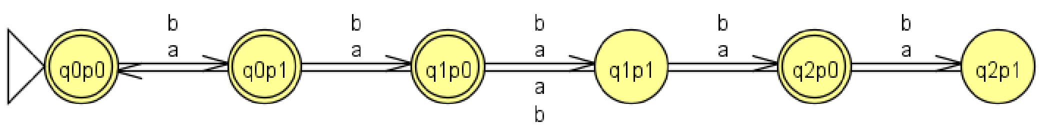
- **DFA for A:** 3 states (q0, q1, q2) in a cycle. Accepting state: q0.
- **DFA for B:** 2 states (p0, p1) toggling. Accepting state: p0.

PART B



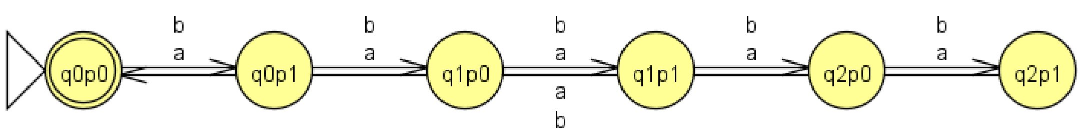
PART C

DFA for A ∪ B (Union)



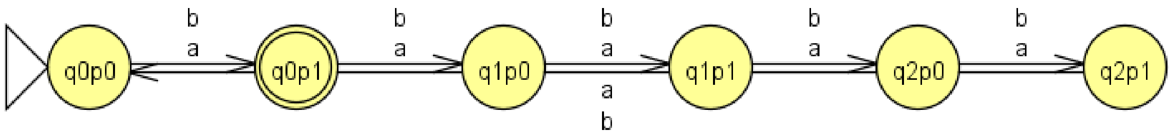
- Built using cross product of DFA-A and DFA-B.
- 6 states: (q0p0, q0p1, q1p0, q1p1, q2p0, q2p1)
- Accepting states: Any state where mod 3 = 0 or mod 2 = 0.
    - Accepting: q0p0, q0p1, q1p0, q2p0

DFA for Intersection



- Same 6-state product DFA.
- Accepting state: Only where mod 3 = 0 and mod 2 = 0.
    - Accepting: q0p0

DFA for Difference

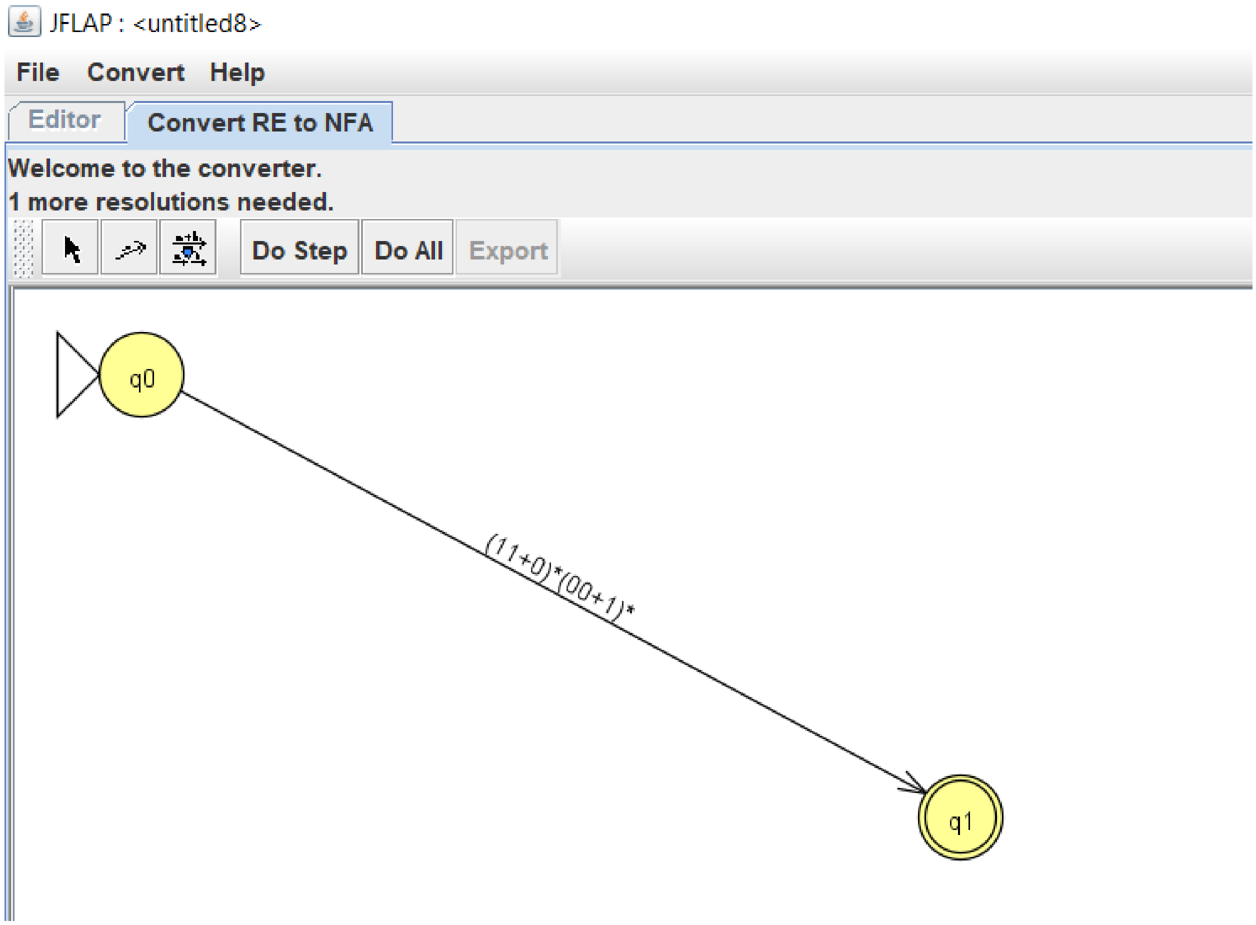


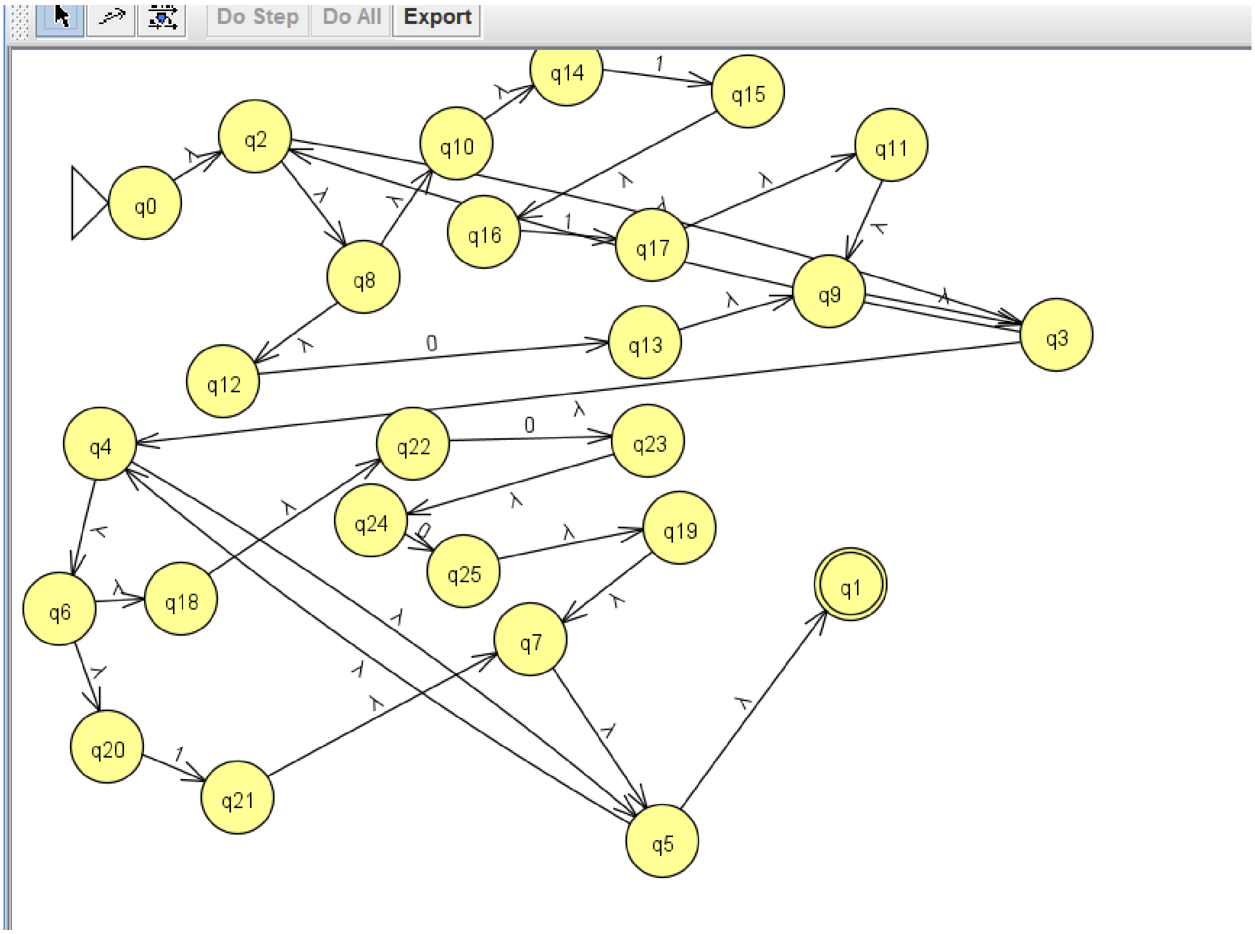
- Same product DFA.
- Accepting states: Where mod 3 = 0 and mod 2 ≠ 0.
  - Accepting: q0p1

PART D

ASSIGNMENT 2.

QUESTION 2
PART A


JFLAP : <untitled8>
File Convert Help
Editor
Convert RE to NFA
Welcome to the converter.
1 more resolutions needed.
Do Step
Do All
Export
q0
(11+0)*(00+1)*
q1

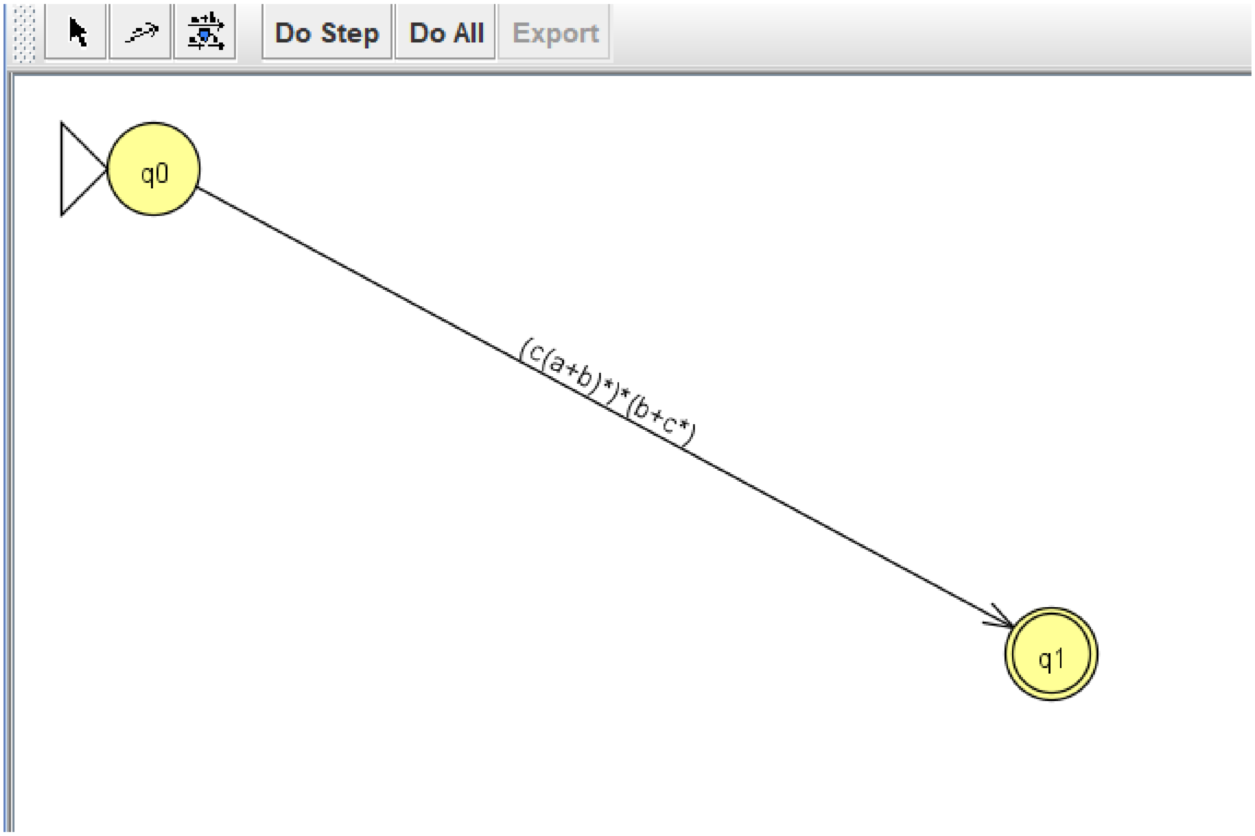


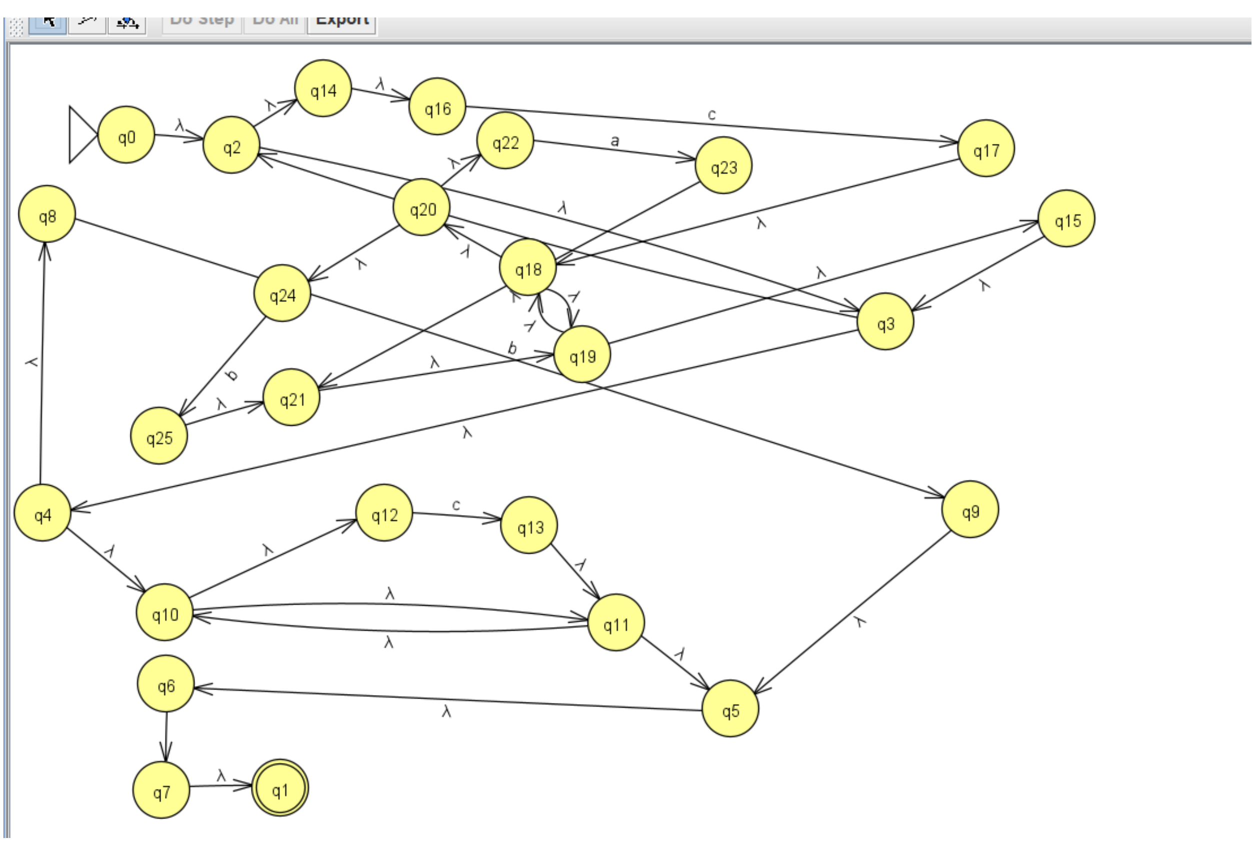
- The NFA begins by allowing zero or more repetitions of either the substring **"11"** or the single character **"0"**, modeled using branching ϵ-transitions for the choice (11 + 0)*.

- After completing this part, it moves to the second section which accepts zero or more repetitions of either **"00"** or the single character **"1"**, again using ϵ-transitions for choice and looping, representing (00 + 1)*.

- The combined automaton sequentially links these two parts, accepting strings formed by concatenating any number of (11 or 0) sequences followed by any number of (00 or 1) sequences.

PART B


Do Step
Do All
Export
q0
(c(a+b)*)*(b+c*)
q1


Do Step
Do All
Export
q0
q1
q2
q3
q4
q5
q6
q7
q8
q9
q10
q11
q12
q13
q14
q15
q16
q17
q18
q19
q20
q21
q22
q23
q24
q25
a
b
c

- This NFA allows zero or more repetitions of a pattern starting with **'c'** followed by zero or more **'a' or 'b'** characters, represented by $(c(a + b)^*)^*$.

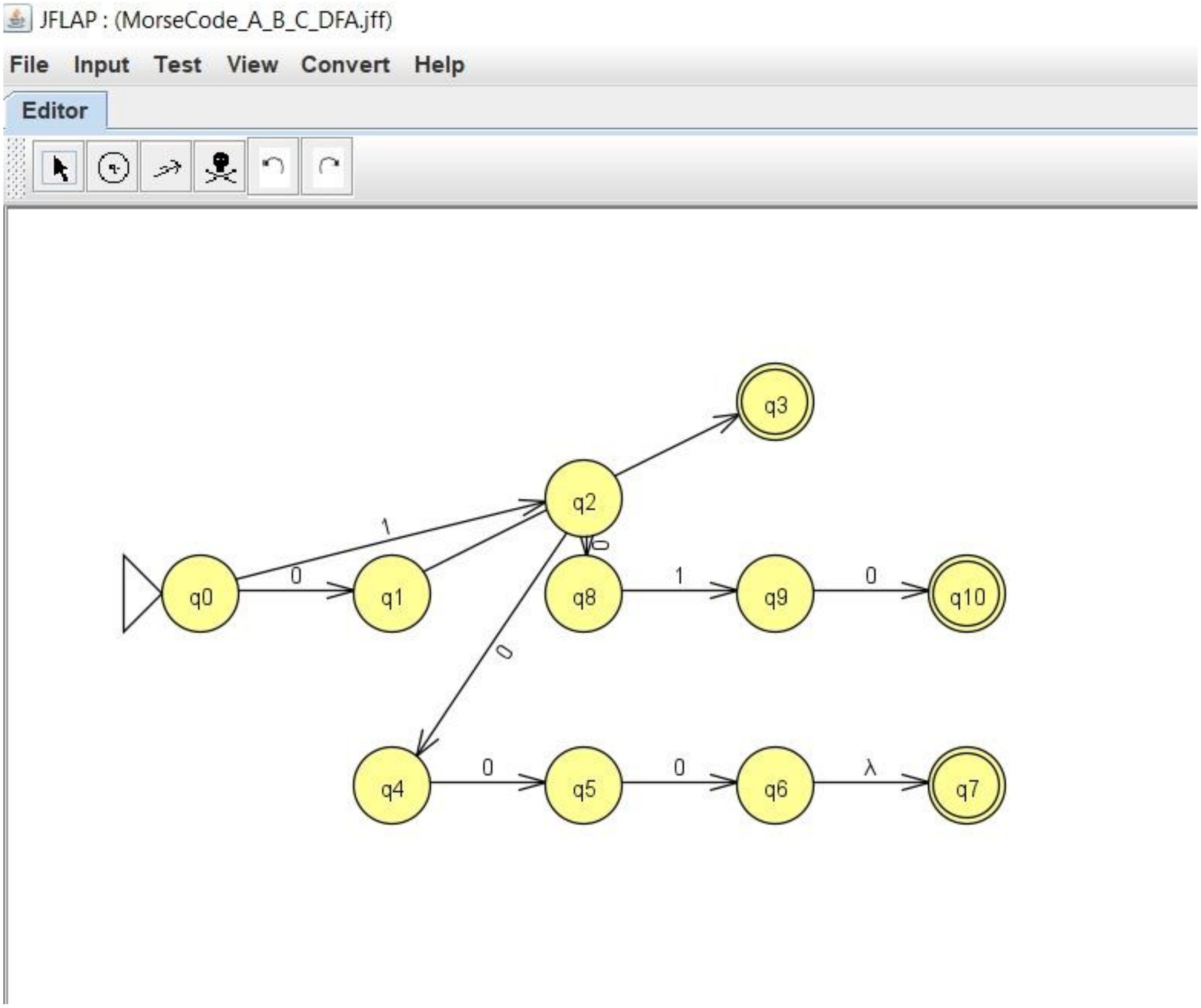
- ϵ-transitions enable looping back to accept multiple such sequences, allowing flexibility in the number of repetitions.

- After that, the automaton accepts either a single **'b'** or zero or more repetitions of **'c'**, $(b + c^*)$, using branching and looping transitions.

- Overall, the NFA captures strings formed by repeating the 'c' followed by 'a' or 'b' pattern any number of times, then ending with either a 'b' or a series of 'c's.

c) Melay/Moore Machine

Construct Morse-code related machine discussed in class. Assume we transmit only 3 letters A, B, C



Morse Code DFA Analysis for Letters A, B, and C

This DFA (Deterministic Finite Automaton) was constructed using JFLAP to recognize Morse code representations of the letters **A**, **B**, and **C**, where:

- 0 represents a **dot (·)**,
- 1 represents a **dash (−)**.

The Morse codes for the letters are:

- **A** = ·− → 01
- **B** = −··· → 1000
- **C** = −·−· → 1010

State Transitions and Structure

- The automaton begins at **q0**, the start state.
- Transitions follow the binary Morse encoding:
    - **A (01)** leads from q0 → q1 (0) → q2 (1) → q3 (final)
    - **B (1000)** leads from q0 → q2 (1) → q4 (0) → q5 (0) → q6 (0) → q7 (final)
    - **C (1010)** leads from q0 → q2 (1) → q8 (0) → q9 (1) → q10 (0) → final

Accept States

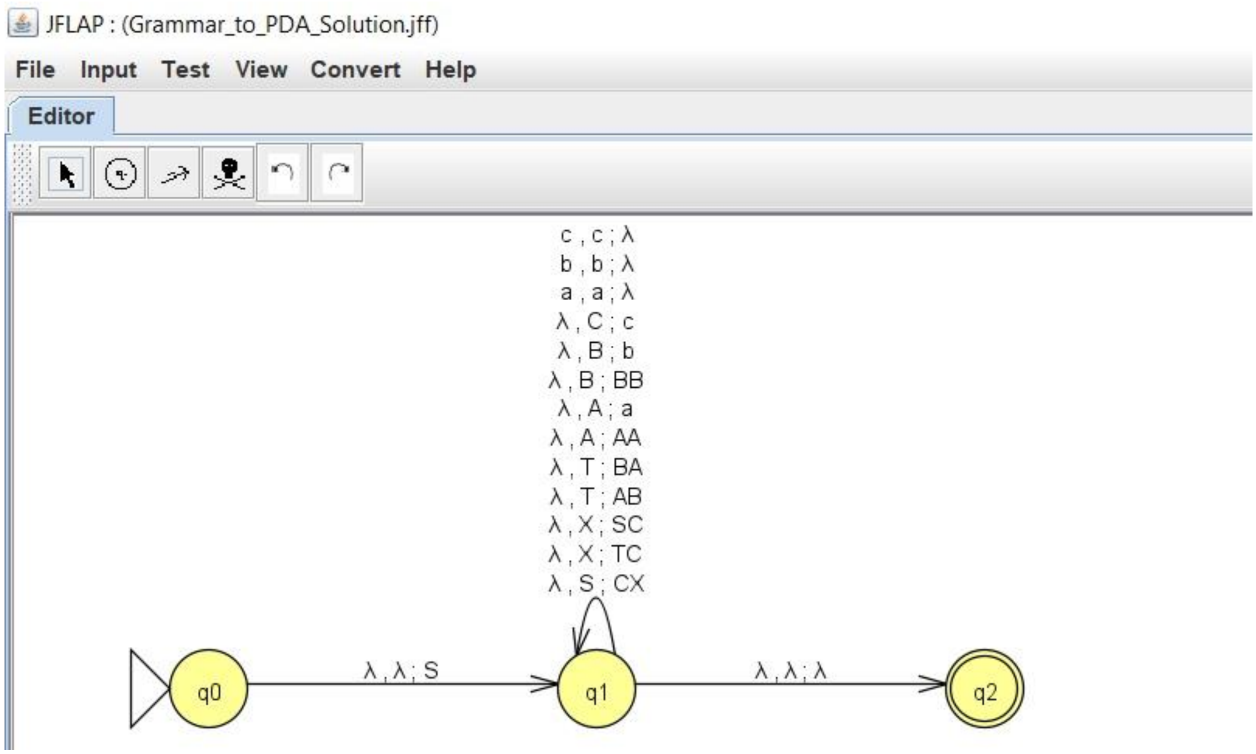
- **q3, q7, and q10** are accepting states, each corresponding to the recognition of one complete Morse code letter.

Epsilon Transitions

- The DFA uses a **λ (epsilon)** transition from **q6 to q7**, which is a non-standard DFA feature (technically making it an NFA), but JFLAP allows this for simplicity.

Mutual conversion of Grammar and PDA

Question 9



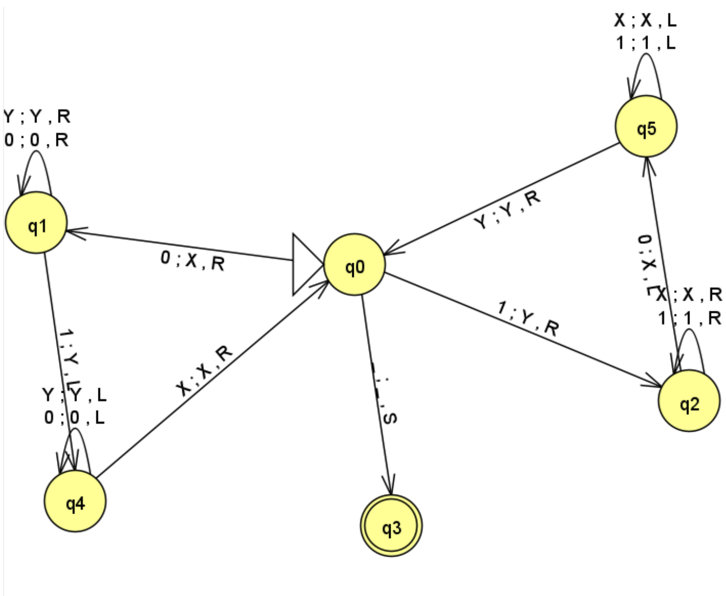
Analysis and Result of Grammar-to-PDA Conversion (Using JFLAP)

The PDA consists of three states: q0 (start state), q1 (processing state), and q2 (final/accepting state).

- The transition from q0 to q1 pushes the start symbol S onto the stack.
- In q1, a series of transitions simulate the leftmost derivation of the grammar using λ-transitions (i.e., without consuming input), replacing variables like S, A, B, C, etc., with their corresponding right-hand sides as per production rules.
- Transitions such as a, A ; λ or b, B ; λ represent terminal matching, where input symbols are consumed and the stack is popped accordingly.
- Once the input is completely read and the stack is empty, the PDA transitions to q2, indicating successful acceptance of the input string.

This PDA accepts strings generated by the original CFG, demonstrating the **equivalence between context-free grammars and pushdown automata**. The construction verifies that for any context-free language, a PDA can be designed to recognize it.

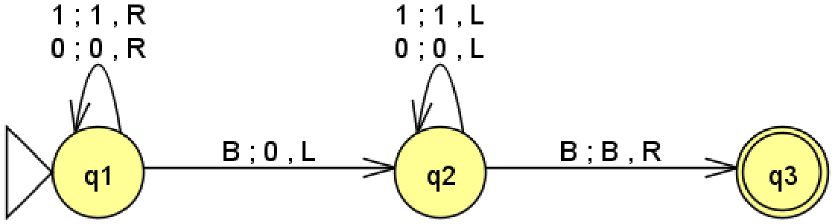
3. F(iv):



- It scans the input to find a 0 or 1.
- If it finds a 0, it replaces it with X and searches right for an unmatched 1, replacing it with Y.
- If it finds a 1 first, it marks it as Y and searches for a 0 to mark as X.
- After pairing a 0 and a 1, it returns to the beginning to repeat the process.
- If all 0s and 1s are matched (all marked as X and Y), the machine accepts.
- If it finds an unmatched 0 or 1, it halts and rejects.

This ensures the number of 0s equals the number of 1s.

g(i):



This Turing Machine has 3 states:

- q1: Start state
- q2: Moves left to reverse
- q3: Halting state (final)

Tape alphabet: 0, 1, B (blank)

Transitions:

- In q1, the machine moves **right** over all 0s and 1s.
- When it hits a blank (B), it switches to q2, writes 0, and moves **left**.
- In q2, it moves **left** over everything.
- When it hits the left blank (B), it switches to q3 and halts.

Input: 00111

Initial tape:
B 0 0 1 1 1 B

Steps:

1. Start at leftmost 0, in q1, move right.
2. Skip all input until hitting blank after last 1.
3. Write 0 over that blank and switch to q2, move left.
4. In q2, move left over all digits.
5. When it sees left blank, halt in q3.