Compiler Design and Principles

**Class/Home Assignment 2**

Mar 10, 2025. **Due: March 14 EoD, 2025**

(Individual Effort)

Q1: Construct a DFA equivalent to the following NFA, where the alphabet is {0,1}.

| State | 0 Transition | 1 Transition |
|---|---|---|
| q0 | {q0, q1} | {q0} |
| q1 | {q2} | Ø |
| *q2 | {q2} | {q1} |

Identify the final states of the resulting DFA. Minimize the DFA if possible.

Q2: Convert the following ε-NFA into an equivalent DFA.

| State | ε-Transition | 0 Transition | 1 Transition |
|---|---|---|---|
| q0 | {q1} | {q2} | Ø |
| q1 | {q3} | Ø | {q2} |
| q2 | Ø | {q2} | {q3} |
| *q3 | Ø | Ø | {q2} |

Compute the ε-closures before applying subset construction.

Draw the resulting DFA state transition table.

Q3: Construct an NFA for the following regular expression: **(0|1)∗00(0|1)∗**

Convert it into a DFA using subset construction. Draw the DFA transition table and minimize it.

Q4: Write a regular expression for the following language:

Strings over {a, b} where every string starts with 'a' and ends with 'b'.

Convert this regular expression into an NFA.

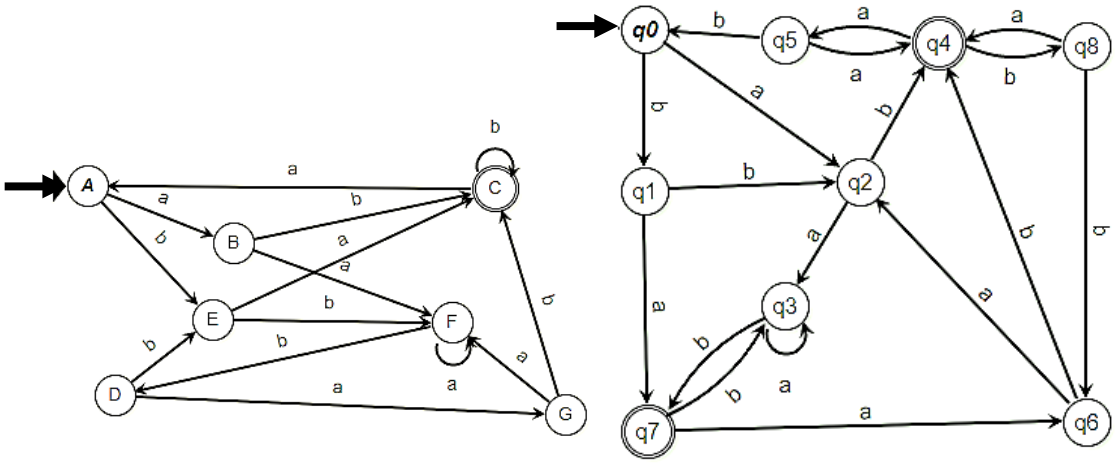
Q5: Write a regular expression that describes all binary strings that do not contain consecutive 1s.

Convert this regular expression into an ε-NFA. Convert the ε-NFA into a DFA.

Q6: Analyze the following RE: ^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$

Generate a DFA (NFA or ε-NFA converted to DFA).

Q7: Minimize the Following DFAs.




Q8: Write regular expressions and DFA for the following:

A valid variable/function name in C. Minimum 1 and maximum 31 characters, cannot start with a number, but may contain numbers, can contain and start from _ and may contain uppercase or lowercase alphabet.