# Lab Report
# Removing ε-Transitions from a Non-Deterministic Finite Automaton

**Course:** Compiler Design
**Programming Language:** C
**Group Members:**
Benhabrou Zakaria
Habchi Samir
Boussouf Zineddine

# 1. Introduction

This lab aims to implement a C program that removes epsilon (ε) transitions from a Non-Deterministic Finite Automaton (NFA). ε-transitions allow state changes without consuming input symbols and are commonly produced when building automata from regular expressions. Eliminating ε-transitions simplifies the automaton and is a necessary step before determinization.

# 2. Theoretical Background

An ε-NFA is an NFA that allows transitions labeled with ε. The ε-closure of a state q is defined as the set of all states reachable from q using only ε-transitions, including q itself. Computing ε-closures is the foundation of removing ε-transitions while preserving the language recognized by the automaton.

# 3. Algorithm Description

The elimination of ε-transitions is performed using a structured algorithm divided into five tasks.

**Task 1: Reading the NFA**

The automaton is read from user input, including the set of states, input symbols, symbol transitions, ε-transitions, the start state, and the set of final states.

**Task 2: Computing ε-closures**

For each state, the ε-closure is computed iteratively by exploring ε-transitions until no new reachable states are found.

**Task 3: Computing New Transitions**

For each state q and input symbol a, the new transition $\delta'(q, a)$ is computed as: $\delta'(q, a) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q), a))$. This ensures that ε-transitions are completely removed while preserving behavior.

**Task 4: Computing New Final States**

A state is considered final in the new automaton if its ε-closure contains at least one original final state.

**Task 5: Displaying the Resulting NFA**

The resulting ε-free NFA is displayed by listing the states, start state, final states, and all symbol-based transitions.

## 4. Implementation Details

The program is implemented in C using static arrays for simplicity and clarity. The design follows a modular approach with one function per task. Dynamic memory allocation is avoided to reduce complexity and improve reliability.

## 5. Illustrative Example

Consider an ε-NFA with states {0, 1}, alphabet {a, b}, start state 0, and final state {1}. State 0 has an ε-transition to state 1. The ε-closure of state 0 is {0, 1}, while the ε-closure of state 1 is {1}. After eliminating ε-transitions, both states become final states and the resulting automaton recognizes the same language.

## 6. Conclusion

In this lab, ε-transitions were successfully eliminated from a Non-Deterministic Finite Automaton. By computing ε-closures and reconstructing transitions, an equivalent ε-free NFA was obtained. This transformation preserves the recognized language and prepares the automaton for determinization into a DFA.

## Appendix: Source Code

The complete C source code implementing the ε-transition elimination algorithm is provided separately in the appendix section as required.