National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS 354: Compiler Construction

Class: BSCS-5A

Lab [01]: DFA Simulation

Date: 6th September, 2018

Time: [09:00am – 12:00pm]

Instructor: Dr. Rabia Irfan

Lab Engineer: Mr. Azaz Farooq

Lab [01] : DFA Simulation

Introduction

A deterministic finite automaton (DFA)—also known as deterministic finite state machine—is a finite state machine that accepts/rejects finite strings of symbols and only produces a unique computation (or run) of the automaton for each input string. 'Deterministic' refers to the uniqueness of the computation.

Objectives

Successful understanding/implementation DFA in C/C++/Java

Tools/Software Requirement

gcc, g++, GNU Make or Visual Studio C++ or Choice of your own programming language

Description

A DFA is defined as an abstract mathematical concept, but due to the deterministic nature of a DFA, it is implementable in hardware and software for solving various problems. For example, a DFA can model software that decides whether or not online user-input such as email addresses are valid. DFAs recognize exactly the set of regular languages which are, among other things, useful for doing lexical analysis and pattern matching.

Lab Tasks

1. Consider the following Regular Expression:
   $$a(bb)*bc$$
   a. Draw a DFA for the above RE.
   b. Determine the language accepted by this automaton
   c. Implement this DFA in C/C++/Java.
      i. Your implementation should validate the input string for alphabet i.e. $\sum$ = {a, b, c}, before using it in the DFA.
      ii. Your first implementation of DFA should use **goto** statements only.
      iii. Your second implementation of DFA should use **switch** statement instead of the goto's.
   d. Test your implementation using the following inputs:

abc, abbc, abcd, abbbc, abbbbc

2. Consider the following language:

   L(M) = {w | w € {a, b}* and contains even number of a's and b's}

   a. Draw a DFA for the above language
      i. Your implementation should validate the input string for alphabet i.e. ∑ = **{a, b},** before using it in the DFA.
      ii. Your first implementation of DFA should use **goto** statements only.
      iii. Your second implementation of DFA should use **switch** statement instead of the goto's.
   b. Test your implementation using the following inputs:
      aa, abba, abab, aabbaabb, empty string

Deliverables

You are required to upload your task (Sources & Word/PDF document) using the link created on LMS followed by a viva.