# Assignment 5

Advanced Computing concepts for Engineering

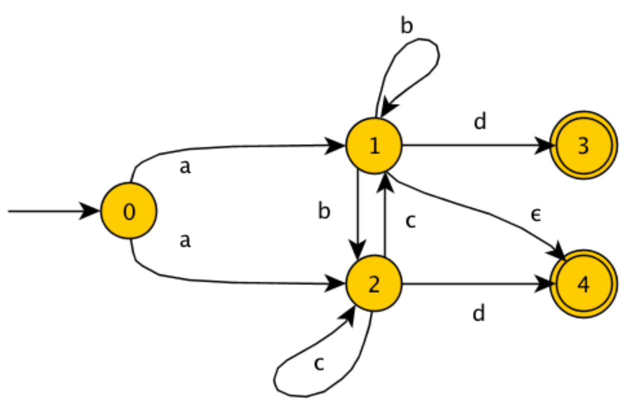
Due 2018 March 1

Note that the work that you turn in for this assignment must represent your individual effort. You are welcome to help your fellow students to understand the material of the course and the meaning of the assignment questions, however, the answer that you submit must be created by you alone.

**Q0.** Apply Thompson's algorithm to these regular expressions:
(a) ('1'; '0'|'0';'1')$^*$
(b) ('1'; '0')$^*$ | ('0';'1')$^*$

**Q1. Language**

Use both definitions of $L(A)$ to show that "abbcd" is in the language of this NDFR:




**Q2. Recognition**

Trace the behaviour of the recognition algorithm for the NDFR in Q1 for the strings "ac", "abad", "a" and "adb".

**Q3.** Show that any two states $q$ and $r$ such that $q \in \epsilon$-closure $(r)$ and $r \in \epsilon$-closure $(q)$ can be replaced by a single state without changing the language.

**Q4. C Comments again**

(a) Design an NDFR for C comments. (To simplify, use an alphabet of 3 symbols: star, slash, and other.) Did you find it easier or harder to design an NDFR or a regular expression?

(b) Try to make an NDFR with no $\epsilon$ transitions and such that, for each state/symbol pair $(q, a)$, there is exactly one $r$ such that $(q, a, r)$ is a transition.

(c) With the NDFR from (b), simulate the recognition algorithm for strings "/*o*/", "/**/*/", "/*oo". What do you note about the size of $R$?

**Extra challenges.**

(a) Design a recognition algorithm for regular expressions that does not use translation to another form. Hint: First define a function from regular expressions and symbols to regular expressions such that $f(x, a)$ is a regular expression that accepts $s$ iff $x$ accepts $[a]; s$.

(b) Implement Thompson's algorithm in the programming language of your choice.

(c) Implement the recognition algorithm in the programming language of your choice.

(d) Show how any NDFR such can be represented by a synchronous circuit that recognizes strings. The strings should be entered into the machine at a rate of one symbol per clock period. Hint: Use one flip-flop per state. It might be easier to start by assuming that there are no $\epsilon$ transitions, then figure out how to handle $\epsilon$ transitions.