CSE396, Spring 2019 **Problem Set 2** **Due Thu. 2/21, 11:59pm**

Reading: Next week will cover the proofs that every NFA has an equivalent DFA and that every finite automaton can be converted into an equivalent regular expression. The former is in section 1.2 of reading already assigned, but now read to the end of section 1.3 for the latter. A few notes:

- This will complete the cycle of equivalences started on Thu. 2/14 by converting regular expressions into equivalent NFAs.
- In my opinion, the Generalized NFA (GNFA) is not "real" the way the Tuesday 2/12 lecture tried to put across that NFAs are real. It is a bookkeeping device to execute the algorithm embodied in the proof.
- The course webpage has an optional handout that gives a "programmatic" version of the algorithm: `https://cse.buffalo.edu/~regan/cse396/CSE396.regexpalg` But lectures will use pictures like the text does.
- Lectures will however present some "economizers" to make the process of doing problems less painful and typo-prone. One is using all 2-state GNFas as the base case so you save a step at the end when the expressions are biggest. Another is never having to make a new start state. Another is that you only need to make a new final state when the NFA has two or more accepting states that are different from its start state. Understanding why these shortcuts are valid is IMHO a mark of really understanding the algorithm on the whole.

Homework—part online and all *individual work*—due **Thu.** 2/21, 11:59pm:

(1) Using *TopHat*, the "Worksheet" titled **Spr'19 HW2.1**. There are 10 questions, each worth 2 points, for 20 total.

(2) Let $D = (A \cap B) \setminus C$ where

$$\begin{aligned} A &= \{w \in \{a,b\}^* : \#a(w)\%2 == 0\} \\ B &= \{w \in \{a,b\}^* : \#b(w)\%2 == 1\} \\ C &= (a+b)^*ab(a+b)^*. \end{aligned}$$

(a) Build a DFA M with 5 states such that $L(M) = D$. As usual, the states should have some logic comments.

(b) Build an NFA N with 4 states such that $L(N) = D$.

(c) Give a regular expression r such that $L(r) = D$.

Part (b) may seem trivial and N might not "feel like" an NFA—but you can consider whether it helps you focus on what was needed in part (c). (12 + 6 + 9 = 27 pts.)

(3) Again with $\Sigma = \{a, b\}$, define L_1 to be the language of the regular expression $b(ab)^*(a + \epsilon)$ and L_2 to be the language of strings that start with b and have an even number of a's. Design DFAs M_1 and M_2 such that $L(M_1) = L_1$ and $L(M_2) = L_2$, with just a few appropriate strategy comments. Then expressly use the Cartesian Product construction to design a DFA M_3 such that $L(M_3) = L(M_1) \cap L(M_2)$. (You might fear that M_3 would have a lot of states, but since this is the $\cap$ case and both L_1 and L_2 have "dead conditions," you should note that you can economize quite a bit. 6+6+18 = 30 pts., for 77 on the set.)