Homework 2: DFAs and NFAs

CSE 30151 Spring 2016

Due 2016/01/26 at 11:59pm

Instructions

- You can prepare your solutions however you like (by hand, LaTeX, Jupyter/IPython notebook, etc.), but you must submit them as a single PDF file.
  - Scan written solutions in the library or using a smartphone (e.g., CamScanner).
  - Convert a Jupyter/IPython notebook using one of these commands:

    ```
    jupyter nbconvert netid-hw2.ipynb --to pdf
    ipython nbconvert netid-hw2.ipynb --to pdf
    ```

- Please give every PDF file a unique name.
  - If you're making a complete submission, name your PDF file `netid-hw2.pdf`, where `netid` is replaced with your NetID.
  - If you're submitting some problems now and want to submit other problems later, name your PDF file `netid-hw2-1234.pdf`, where `1234` is replaced with the problems you are submitting at this time.
  - If you use the same name twice, only the most recent version will be graded!
- Submit your PDF file in Sakai. Don't forget to click the Submit (or Resubmit) button!

Problems

Each problem is worth 7 points. The remaining 2 points are for legibility and clarity. (Hint: One way to ensure legibility is to use LaTeX!)

1. **Designing finite automata.** Write a finite automaton for base-10 natural numbers that are
   (a) divisible by 2
   (b) divisible by 4

(c) divisible by 3

Note: Leading zeros should not be allowed.
Alternative: Instead of the above, you can describe more generally how to construct, for any natural number k, a finite automaton for base-10 natural numbers that are divisible by k.

2. **Boolean operations.** Define $L_1 \uparrow L_2 = (L_1 \cap L_2)^C$ (that is, the NAND operation on languages).

   (a) Use a product construction, as in the proof of Theorem 1.25, to prove that regular languages are closed under the $\uparrow$ (NAND) operation.

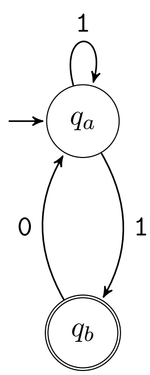
   (b) Recall from Logic Design that any Boolean function (that is, a function from k Boolean values to a Boolean value) can be expressed in terms of NAND gates. Briefly explain how to construct such an expression.

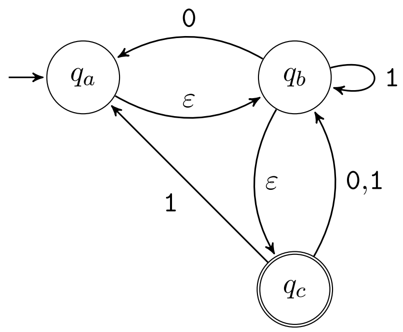
   (c) Conclude that regular languages are closed under any Boolean function. That is, if $L_1, \ldots, L_k$ are regular languages, and f is a function from k Boolean values to a Boolean value, then the language $f(L_1, \ldots, L_k)$ is regular:

   $$w \in f(L_1, \ldots, L_k) \Leftrightarrow f(w \in L_1, \ldots, w \in L_k).$$

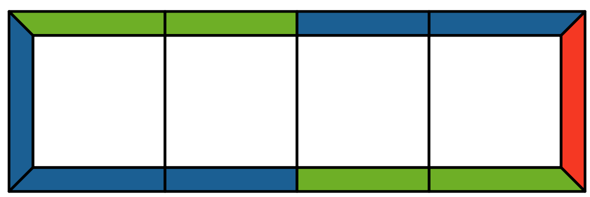
   (Here "$x \in X$" is being used as a Boolean expression, which is a little nonstandard for mathematical writing.)

3. **The subset construction.** Use the construction given by Theorem 1.39 (page 55) to convert the following two nondeterministic finite automata into equivalent deterministic finite automata.

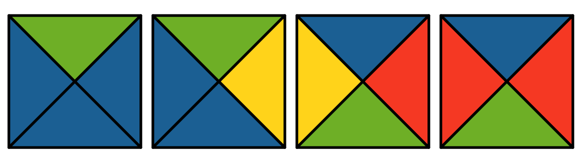
   (a) 

(b) 

4. **Solving puzzle #1** In class, we did three puzzles, the first of which is equivalent to finite automata. In general, a puzzle of this type has a frame like this (but possibly with more/fewer squares and different colors):



And a finite set of tiles like this (but possibly with more/fewer tiles and different colors):



The tiles must be arranged so that adjacent areas have matching colors. There is an unlimited number of copies of each tile.

(a) Show how every puzzle of this type can be converted into a finite automaton M and a string w such that M accepts w if and only if the puzzle has a solution.

(b) Apply your construction to the above instance.

(c) Briefly describe how this gives an $O(n)$ algorithm for solving puzzles of this type.