Lectures and Reading. After Friday’s lecture on regular expressions and their equivalent NFAs (with \(\epsilon\)-transitions), next week’s lectures will complete the cycle of showing those two formalisms recognize the same class of languages as DFAs. The class is standardly denoted by \(\text{REG}\) for the class of regular languages. Then lectures will move into the Myhill-Nerode technique for showing that certain other languages are non-regular and for showing that certain sets of strings must all be processed to distinguished states by a DFA. So read Debray’s notes up through the end of section 3 on page 11, in tandem with the slides notes called “Extra notes for Weeks 2–3” on the course webpage. Then also read the “Myhill-Nerode” handout for CSE396.

The first two problems are based on lectures through this week; the latter two next week (and would nominally be assigned on Wed. 9/12 in a weekly-HW scheme).

1. Let positive natural numbers be written in standard binary notation but with the leading ‘1’ removed: \(\epsilon\) stands for 1, 0 for 10 = 2, for 1 for 11 = 3, 00 for 100 = 4, and so on. This gives a 1-to-1 correspondence between \(\{0, 1\}^*\) and \(\mathbb{N}^+\). Under this correspondence, design a deterministic finite automaton \(M_6\) that recognizes the language \(L_6\) of strings denoting integers that are 1 more than a multiple of 6:

\[
L_6 = \{\epsilon, 11, 101, 0011, \ldots\} \quad \text{standing for} \quad \{1, 7, 13, 19, \ldots\}.
\]

You may design \(M_6\) directly or as a combination of smaller machines, but especially in the former case, you should provide “code comments” explaining the meaning and functions of the states and showing why \(L(M_6) = L_6\). (24 pts. total)

2. Design both nondeterministic finite automata \(N_a, N_b, N_c\) and regular expressions \(r_a, r_b, r_c\) that denote the following three languages described in prose. (It is OK for your NFAs to have \(\epsilon\)-arcs, and it is fine if one or more are DFAs since a DFA “Is-A” NFA.) All use the alphabet \(\Sigma = \{0, 1\}\). (3 \times (6 + 6) = 36 points total)

(a) \(L_a = \) the set of binary strings ending in 010 or 101.

(b) \(L_b = \) the set of binary strings that have both a 000 and a 111 in them somewhere.

(c) \(L_c = \) the set of binary strings having no occurrence of the substring 00 and at most one occurrence of 11.

(set continues overleaf)
(3) Convert the following NFA $N$ into an equivalent DFA $M$:

Here $N = (Q, \Sigma, \delta, s, F)$ with $Q = \{1, 2, 3, 4\}$, $\Sigma = \{a, b\}$, $s = 1$, $F = \{2\}$, and

$$\delta = \{(1, \epsilon, 2), (1, b, 4), (2, a, 2), (2, b, 3), (3, a, 4), (4, \epsilon, 1), (4, b, 3)\}.$$

Also answer:

(a) Is there a string $u$ that $N$ can process from its start state to each and every one of its four states? Viewing your DFA $M$, give a shortest $u$ if so.

(b) Is there a string $v$ that $N$ cannot process from start to any state? Again give a shortest such $v$.

(c) Is there a string $w$ such that no matter what string $y$ follows it, the string $wy$ is accepted? Again use your $M$.

(d) Stronger than (b) and counter to (c), is there a string $z$ that $N$ cannot process at all, not from any of its states $p$ to any state $q$? Again use your $M$ to explain your answer.

(The NFA drawing was produced via http://madebyevan.com/fsm/, Evan Wallace’s “Finite State Machine Designer”; note that the LaTeX source can be edited after pasting. This question has $18 + 3 \times 4 = 30$ pts. total.)

(4) Use the FA-to-regular-expression algorithm (with shortcuts allowed) to build a regular expression $r$ such that $L(r) = L(N) = L(M)$ in problem (3). (You may find it easier to work from $N$ not $M$. 24 pts., for 114 total on the combined set.)