CSE396 Problem Set 3 Answer Key Spring 2017

(1) Convert the following two NFAs N_1 and N_2 into equivalent DFAs. Both have $\Sigma = \{a, b\}$. For each one, say how many more (or less?) states and instructions the DFA has, whether it is possible for the NFA to "die," and whether it is possible for a string to activate all 3 states of N_1 at the same time. (18 + 18 + 9 = 45 pts.)

N_1 has states $\{1, 2, 3\}$, instructions $\delta_1 = \{(1, a, 2), (1, a, 3), (1, b, 2), (2, a, 3), (3, b, 1), (3, b, 2)\}$, start state 1, and $F_1 = \{2\}$.
N_2 has $\delta_2 = \{(1, \epsilon, 2), (1, a, 3), (2, a, 2), (2, b, 4), (3, b, 2), (3, b, 4), (4, a, 4), (4, b, 1)\}$, $Q_2 = \{1, 2, 3, 4\}$, $s_2 = 1$, $F_2 = \{\,2\,\}$.

Answer: There are no ϵ-arcs, so my $\underline{\delta}$ is the same as the text's treatment of δ as a function:

$$\begin{bmatrix} \underline{\delta}(1,a) = \{2,3\} & \underline{\delta}(1,b) = \{2\} \\ \underline{\delta}(2,a) = \{3\} & \underline{\delta}(2,b) = \emptyset \\ \underline{\delta}(3,a) = \emptyset & \underline{\delta}(3,b) = \{1,2\} \end{bmatrix}.$$

Start is just $S = \{1\}$. Expanding by breadth-first search (BFS), we get:

$$\begin{aligned}
\Delta(\{1\}, a) &= \underline{\delta}(1,a) = \{2,3\} \quad \text{(new state)} \\
\Delta(\{1\}, b) &= \underline{\delta}(1,b) = \{2\} \quad \text{(new state)} \\
\Delta(\{2,3\}, a) &= \underline{\delta}(2,a) \cup \underline{\delta}(3,a) = \{3\} \cup \emptyset = \quad \text{(new)} \\
\Delta(\{2,3\}, b) &= \emptyset \cup \{1,2\} = \{1,2\} \quad \text{(new)} \\
\Delta(\{2\}, a) &= \underline{\delta}(2,a) = \{3\} \quad \text{(old)} \\
\Delta(\{2\}, b) &= \underline{\delta}(2,b) = \emptyset \quad \text{(new but dead)} \\
\Delta(\{3\}, a) &= \underline{\delta}(3,a) = \emptyset \quad \text{(old and dead)} \\
\Delta(\{3\}, b) &= \underline{\delta}(3,b) = \{1,2\} \quad \text{(old)} \\
\Delta(\{1,2\}, a) &= \underline{\delta}(1,a) \cup \underline{\delta}(2,a) = \{2,3\} \quad \text{(old)} \\
\Delta(\{1,2\}, b) &= \{2\} \cup \emptyset = \{2\} \quad \text{(old)} \\
\Delta(\emptyset, a) &= \Delta(\emptyset, b) = \emptyset
\end{aligned}$$

All new states have been expanded, so the DFA is closed. It has 6 states and 12 arcs, double what the NFA had in each. The state $\{1, 2, 3\}$ was never reached, and this means there is *no* string that activates all 3 states of N at once. (Indeed, we can't even get 1 and 3 on simultaneously. Looking at N_1 tells why: you can only get to 1 on a b and to 3 on an a.) But $\emptyset$ is reachable, most immediately by the straing bb, which causes (all processing in) the NFA to die. The set F' of final states in the DFA comprises all sets that include the accepting state 2 of the NFA, so $F' = \{\{2\}, \{1, 2\}, \{2, 3\}\}$.

For N_2, the ϵ-arc $(1, \epsilon, 2)$ (which is the only one) carries the meaning "whenever 1 then also 2" and makes the start state S_2 of the DFA M_2 equal $\{1, 2\}$, **not just {1}**. Since 2 is

accepting while 1 is not listed as such (though it could be), you need to include 2 to reflect that ϵ is in $L(N_2)$ so you need the start state of the DFA M_2 to be accepting.

The ϵ-arc also makes my $\underline{\delta}$ different from the text's δ as a function, so we chug it out:

$$\begin{bmatrix} \underline{\delta}(1,a) = \{3\} & \underline{\delta}(1,b) = \emptyset \\ \underline{\delta}(2,a) = \{2\} & \underline{\delta}(2,b) = \{4\} \\ \underline{\delta}(3,a) = \emptyset & \underline{\delta}(3,b) = \{2,4\} \\ \underline{\delta}(4,a) = \{4\} & \underline{\delta}(4,b) = \{1,2\} \end{bmatrix}.$$

Most important here is that on $\underline{\delta}(4,b)$ we included state 2 not just 1 by the rule, "whenever 1, then also 2." Now the rest is automatic by BFS:

$$\begin{aligned}
\Delta(S,a) &= \underline{\delta}(1,a) \cup \underline{\delta}(2,a) = \{2,3\} \\
\Delta(S,a) &= \underline{\delta}(1,b) \cup \underline{\delta}(2,b) = \emptyset \cup \{4\} = \{4\} \\
\Delta(\{2,3\},a) &= \{2\} \cup \emptyset = \{2\} \quad \text{(new again)} \\
\Delta(\{2,3\},b) &= \{4\} \cup \{2,4\} = \{2,4\} \quad \text{(and again)} \\
\Delta(\{4\},a) &= \{4\} \quad \text{(self-loop in DFA too here)} \\
\Delta(\{4\},a) &= \{1,2\} \\
\Delta(\{2\},a) &= \{2\} \quad \text{(and self-loop here)} \\
\Delta(\{2\},b) &= \{4\} \\
\Delta(\{2,4\},a) &= \{2.4\} \quad \text{(different way to get a self-loop)} \\
\Delta(\{2,4\},b) &= \{1,2,4\} \quad \text{(this counts as a new state)} \\
\Delta(\{1,2,4\},a) &= \{3\} \cup \{2\} \cup \{4\} = \{2,3,4\} \quad \text{(ditto!)} \\
\Delta(\{1,2,4\},b) &= \emptyset \cup \{4\} \cup \{1,2\} = \{1,2,4\} \quad \text{(not new)} \\
\Delta(\{2,3,4\},a) &= \{2\} \cup \emptyset \cup \{4\} = \{2,4\} \\
\Delta(\{2,3,4\},b) &= \{4\} \cup \{2,4\} \cup \{1,2\} = \{1,2,4\} \quad \text{(closes BFS)}
\end{aligned}$$

Finally, $F' = \{\text{anything with } 2\} = \{\{1,2\},\{2,3\},\{2,4\},\{1,2,4\},\{2,3,4\}\}$ = all the states except $\{4\}$. We went from 4 to only 6 states, tons better than the max of 16 (which was already a max of only 12 thanks to the ϵ-arc ruling out the 4 combinations that have 1 but not 2), and from 7 to 12 instructions. We found no dead state in the DFA M_2, and this means N_2 never dies.

Nor did we find the "full state" $\{1,2,3,4\}$ but here's why the problem set avoided asking about it for N_2: the DFA *does* have a "nirvana condition." Once you enter the state $\{2,4\}$ (say by the string ab) you never leave a cluster of three states all of which have 2 so they all accept. This means too that the DFA is not minimal. You could replace the cluster by a single "nirvana state" and here's the kicker: you might as well label it $\{1,2,3,4\}$. So even though there is in fact no way to light up all 4 of the NFA's states at the same time, the accept-reject behavior makes things *tantamount* to saying so, in a "virtual" sense. So even though a "yes" answer your be technically false, it would be "morally true" so it wouldn't be marked wrong—given proper essay-style explanation.

(2) Find regular expressions for the following languages. In all cases you must give some reasoning to explain how you got the expression—and it is your job to explain your interpretation of any element of the description that is possibly ambiguous. It is AOK but not necessary to sketch a DFA as part of this reasoning, nor demanded that you use the DFA-to-regexp conversion. (9+9 = 18 pts., for 63 total on the set).

(a) The language of strings over $\{a, b, \#\}$ that have a single # in them, and that either have at least two a's *immediately before* the #, or at least two b's *somewhere after* the #. (The two b's need not be consecutive like the two a's.)

(b) The language of strings over $\{a, b\}$ that do not begin with the substring bba.

Answer: (a) The "or" is the outermost operator and says to build a regexp r of the form $r = (r_1) \cup (r_2)$. Make r_1 handle the "at least two a's *immediately before* the #" part: $r_1 = (a \cup b)^* aa\#(a \cup b)^*$. Note how the final $(a \cup b)^*$ says that in this case we "don't care" what happens after the # (so long as we don't get another # which would be bad). Then you just have to reason out $r_2 = (a \cup b)^* \# a^* b a^* b (a \cup b)^*$. Also good is $r_2 = (a \cup b)^* \# (a \cup b)^* b (a \cup b)^* b (a \cup b)^*$ but the first one is more economical and fixates on the *first* two b's on the right-hand side. Putting it all together:

$$r = (a \cup b)^* aa\#(a \cup b)^* \quad \cup \quad (a \cup b)^* \# a^* b a^* b (a \cup b)^*.$$

Note that "and" would have been easier: $(a \cup b)^* aa\# a^* b a^* b (a \cup b)^*$. If you interpreted the prose as "xor" then you would still have the same structure $r = (r_1) \cup (r_2)$ but you'd have to negate the condition on the other side of the # rather than use an easy "don't care." Happily in this case the negations are not too painful: $\sim r_1 = (a \cup b)^* (b \cup ba) \cup \epsilon \cup a$ and $\sim r_2 = a^* (b \cup \epsilon) a^*$ and you get:

$$(a \cup b)^* aa\# a^* (b \cup \epsilon) a^* \quad \cup \quad ((a \cup b)^* (b \cup ba) \cup \epsilon \cup a) \# a^* b a^* b (a \cup b)^*.$$

(b) A string x avoids beginning with bba if x:

- begins with a,
- begins with ba,
- begins with bbb, OR:
- IS one of the strings ϵ, b, or bb.

The last clause is prone to being forgotten. One way to avoid forgetting was to build the easy "goal-oriented" DFA M for the language of strings that *do* begin with bba and then *complement* it to get a DFA M' with five states, call them s, p, q, d, f. Here the dead state d of M' was the "nirvana" state of M while f was the dead state of M. Since all states of M' except d are accepting, we get that the language L of M' is:

$$\begin{aligned} L &= L_{s,s} \cup L_{s,p} \cup L_{s,q} \cup L_{s,f} \\ &= \epsilon \cup b \cup bb \cup (a \cup ba \cup bbb)(a \cup b)^*. \end{aligned}$$

The second line is the required regular expression; it can be "unfactored" into six terms or factored even a bit more, but this is clear.