CS/ECE 374 A (Spring 2022)
Homework 3 Solutions

Problem 3.1: For each of the following languages in parts (a), (b), and (c), describe an NFA that accepts the language, using as few states as you can. Provide a short explanation of your solution. Below, $\#_0(x)$ and $\#_1(x)$ denote the number of 0's and the number of 1's in x respectively.

(a) (30 pts) all strings $x \in \{0,1\}^*$ such that (x ends with 10101 or 11011) and ($\#_0(x)$ is divisible by 3 or $\#_1(x)$ is divisible by 3).

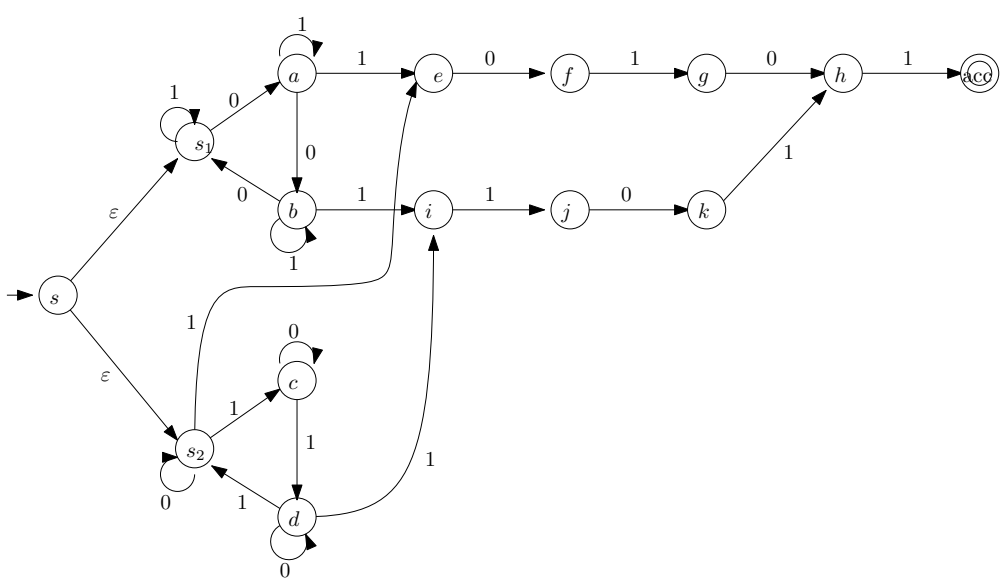
(b) (30 pts) the language defined by the regular expression $\big(((01)^*0+2)(100)^*1\big)^* \cdot (1^*+0^*2^*)$ over the alphabet $\{0,1,2\}$.

(c) (10 pts) all strings in $\{0,1\}^*$ that contains the pattern 0?1?0, where "?" denotes "don't care" (i.e., a single symbol that is either 0 or 1); in other words, the language defined by the regular expression $(0+1)^* \cdot 0(0+1)1(0+1)0 \cdot (0+1)^*$.

(d) (30 pts) Convert your NFA from part (c) to a DFA by using the subset construction (i.e., power set construction). [*Note*: don't include unreachable states; also, several accepting states can be collapsed into one in this DFA.]

Solution:

(a)



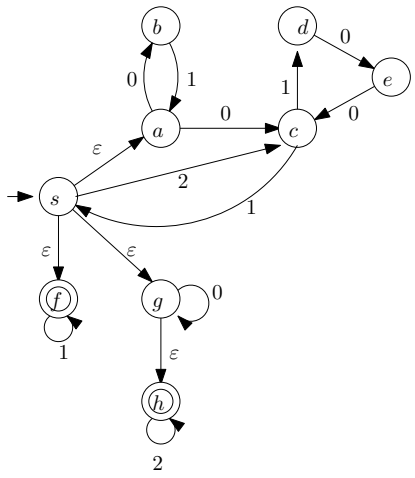
Explanation:

- The part from s_1 accepts all strings that end with 10101 and have number of 0's divisible by 3 (if we exit the cycle s_1ab via transition from a to e), or end with 11011 and have number of 0's divisible by 3 (if we exit the cycle s_1ab via the transition from b to i).

- Similarly, the part from s_2 accepts all strings that end with 10101 and have number of 1's divisible by 3 (if we exit the cycle s_2cd via the transition from s_2 to e), or end with 11011 and have number of 1's divisible by 3 (if we exit the cycle s_2cd via the transition from d to i).
- We take the union by having ε-transitions from s to s_1 and to s_2.

[Note: we will not deduct points if you use a small number of extra states, e.g., if you didn't reuse states in the two parts from s_1 and from s_2. But we will deduct points if you use a product construction, which would increase the number of states drastically.]

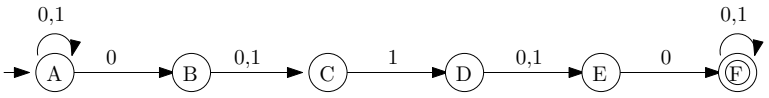
(b)



Explanation: We follow the recursive algorithm from class, taking a few obvious shortcuts and eliminating some of the unnecessary ε-transitions. For example, the states a, b correspond to $(01)^*$ and states c, d, e correspond to $(100)^*$, and so the states s, a, b, c, d, e correspond to $\big(((01)^*0+2)(100)^*1\big)^*$. To get to f, we append 1^*. To get to h, we append 0^*2^*.

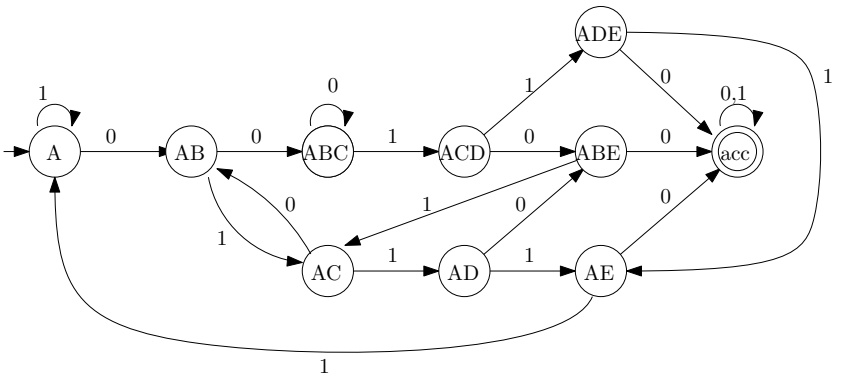
[Note: we will not deduct points for keeping some of the extra ε-transitions—sometimes, it's safer to include them.]

(c)



Explanation: A is the start state, B means that the string ends with 0, C means the string ends with 0?, D means the string ends with 0?1, E means the string ends with 0?1?, and F means the string contains 0?1?0.

(d)



[Note: in the above, we have collapsed all states whose subsets contain F into a single state "acc", because once we have reached such a state, we will accept regardless of the remainder of the input string, for this language. We will not deduct points if you don't collapse them, though the number of states would increase.]

Problem 3.2: Given a language L over the alphabet Σ, define

$$\text{MOVE-BACK}_8(L) = \{xayz : \ xyaz \in L, \ x, y, z \in \Sigma^*, \ a \in \Sigma, \ |y| \leq 8\}.$$

Prove that if L is regular, then $\text{MOVE-BACK}_8(L)$ is regular.

(For example, if $010010100\mathbf{1}0011 \in L$, then $01\mathbf{1}001010010011 \in \text{MOVE-BACK}_8(L)$.)

[*Hint*: given an NFA (or DFA) for L, construct an NFA for $\text{MOVE-BACK}_8(L)$. Give a formal description of your construction. Provide an explanation of how your NFA works, including the meaning of each state. A formal proof of correctness of your NFA is not required.]

Solution: Let L be a regular language over $\Sigma = \{0, 1\}$. By Kleene's theorem, L is accepted by some DFA $M = (Q, \Sigma, s, \delta, A)$. We construct an NFA $M' = (Q', \Sigma, s', \delta', A')$ accepting $\text{MOVE-BACK}_8(L)$ (which would imply that $\text{MOVE-BACK}_8(L)$ is regular by Kleene's theorem). The construction is as follows:

$$\begin{aligned}
Q' &= \{(q, \text{BEFORE}) : q \in Q\} \ \cup \\
&\quad \{(q, a, i, \text{MIDDLE}) : q \in Q, \ a \in \Sigma, \ i \in \{0, 1, \ldots, 8\}\} \ \cup \\
&\quad \{(q, \text{AFTER}) : q \in Q\} \\
s' &= (s, \text{BEFORE}) \\
A' &= \{(q, \text{AFTER}) : q \in A\}
\end{aligned}$$

$$\begin{aligned}
\delta'((q, \text{BEFORE}), a) &= \{(\delta(q, a), \text{BEFORE}), \\
&\quad (q, a, 0, \text{MIDDLE}), \\
&\quad (\delta(q, a), \text{AFTER})\} && \forall q \in Q, \ a \in \Sigma \\
\delta'((q, a, i, \text{MIDDLE}), b) &= \{(\delta(q, b), a, i + 1, \text{MIDDLE}), \\
&\quad (\delta(\delta(q, b), a), \text{AFTER})\} && \forall q \in Q, \ a, b \in \Sigma, \ i \in \{0, \ldots, 7\} \\
\delta'((q, a, 8, \text{MIDDLE}), b) &= \{(\delta(\delta(q, b), a), \text{AFTER})\} && \forall q \in Q, \ a, b, \in \Sigma \\
\delta'((q, \text{AFTER}), a) &= (\delta(q, a), \text{AFTER}) && \forall q \in Q, \ a \in \Sigma
\end{aligned}$$

(The number of states is clearly finite: $|Q'| = 2|Q| + 9|Q||\Sigma|$.)

Explanation: The idea is to divide the process into three phases: BEFORE (reading the prefix x), MIDDLE (reading the substring ay), and AFTER (reading the suffix z). We use nondeterminism to guess when to switch from the BEFORE phase to the MIDDLE phase, and when to switch from the MIDDLE phase to the AFTER phase. At the same time, we simulate M on the string $xyaz$.

Meaning of states in M':

- M' may be in state (q, BEFORE) after reading input w iff M may be in state q after reading input w.
- M' may be in state (q, a, i, MIDDLE) after reading input w iff M may be in state q after reading input xy for some strings x, y with $w = xay$ with $|y| = i$.
- M' may be in state (q, AFTER) after reading input w iff M may be in state q after reading input $xyaz$ for some strings x, y, z with $w = xayz$ with $|y| \leq 8$.

Remark: Alternatively, using ε-transitions, we may define δ' slightly more simply as follows:

$$\begin{array}{rcll}
\delta'((q, \text{BEFORE}), a) & = & \{(\delta(q,a), \text{BEFORE}), & \\
 & & (q, a, 0, \text{MIDDLE})\} & \forall q \in Q,\ a \in \Sigma \\
\delta'((q, a, i, \text{MIDDLE}), b) & = & \{(\delta(q,b), a, i+1, \text{MIDDLE})\} & \forall q \in Q,\ a, b \in \Sigma,\ i \in \{0, \ldots, 7\} \\
\delta'((q, a, i, \text{MIDDLE}), \varepsilon) & = & \{(\delta(q,a), \text{AFTER})\} & \forall q \in Q,\ a, b \in \Sigma,\ i \in \{0, \ldots, 8\} \\
\delta'((q, \text{AFTER}), a) & = & (\delta(q,a), \text{AFTER}) & \forall q \in Q,\ a \in \Sigma
\end{array}$$

[Note: There are also more complicated solutions that "remember" the last 8 characters read, but this would increase the number of states quite a bit.]