4 (100 PTS.) Regularize this.

For each of the following languages over the alphabet $\{0, 1\}$, give a regular expression that describes that language, and briefly argue why your expression is correct.

4.A. All strings that contain the subsequence 101.

Solution:

$$(0+1)^*1(0+1)^*0(0+1)^*1(0+1)^*$$

4.B. All strings that do not contain the subsequence 111.

Solution: That implies that the strings in this language contains at most two 1s. As such, we have

$$0^* + 0^*10^* + 0^*10^*10^* = 0^*(\epsilon + 1 + 10^*1)0^*$$

4.C. All strings that start in 11 and contain 110 as a substring.

Solution: A word in the language starts with a run of two or more 1s and then it has a zero, and then we do not care bout the rest. So:

$$111^*0(0+1)^*$$

4.D. All strings that do not contain the substring 100.

Solution: The idea is to take a word in the language, and treat a run of 1s as a single meta character (denoted by $1^+ = 11^*$). An appearance of the substring 1^+0, must either be the end of the string, or must be followed by 1. As such, our language must contain all the strings in

$$0^*\left(1^+0\right)^*1^*.$$

We claim that is indeed the regular expression for the desired language. To this end, given a word w in the language, scan it and break it whenever encountering 01 (the break is in between the two characters). So $w = s_1 s_2 \ldots s_k$. The string s_1 can be just a run of 0s. Otherwise, if it contains 1, then it must be of the form 1^+0. Similarly, the last string s_k can be a run of 1s. All middle strings, s_i starts with a run of 1s. The strings s_i then can have a single 0, and then it must terminate. Thus, $s_i \in 1^+0$, as claimed.

4.E. All strings in which every nonempty maximal substring of consecutive 0s is of length 1. For instance 1001 is not in the language while 10111 is.

Solution:

Yuk. Let start with some strings that are in the language: 11^*0. In fact, the star closure of this language is also in the language: $(11^*0)^*$. We can of course suffix this with a run of ones: $(11^*0)^*1^*$. This indeed captures all the strings in the language that starts with 1, since one can break the string after each appearance of 0, which give the above expression.

So, we only have to handle, in addition, the strings in the language that starts with 0. Such a strings can be generated only by $0(11^*0)^*1^*$. As such, the overall regular expression is

$$(\epsilon + 0)(11^*0)^*1^*.$$

Let us prove that this is correct. Given a string w in the language, break it into strings $s_1, \ldots, s_k$, where we cut the string w after each appearance of 0 (i.e., $w = s_1 s_2 \cdots s_k$). If $s_1 = 0$, then we choose 0 in the left side, otherwise, we choose ε in the above regular expression. Similarly, if s_k contains no 0s, then we assign it to the suffix 1^*. All other strings, are assigned to the generating expression 11^*0. To see the correctness, observe that every s_i is a non-empty run of 1s followed by a zero.

5 (100 PTS.) Then, shalt thou find two runs of three.

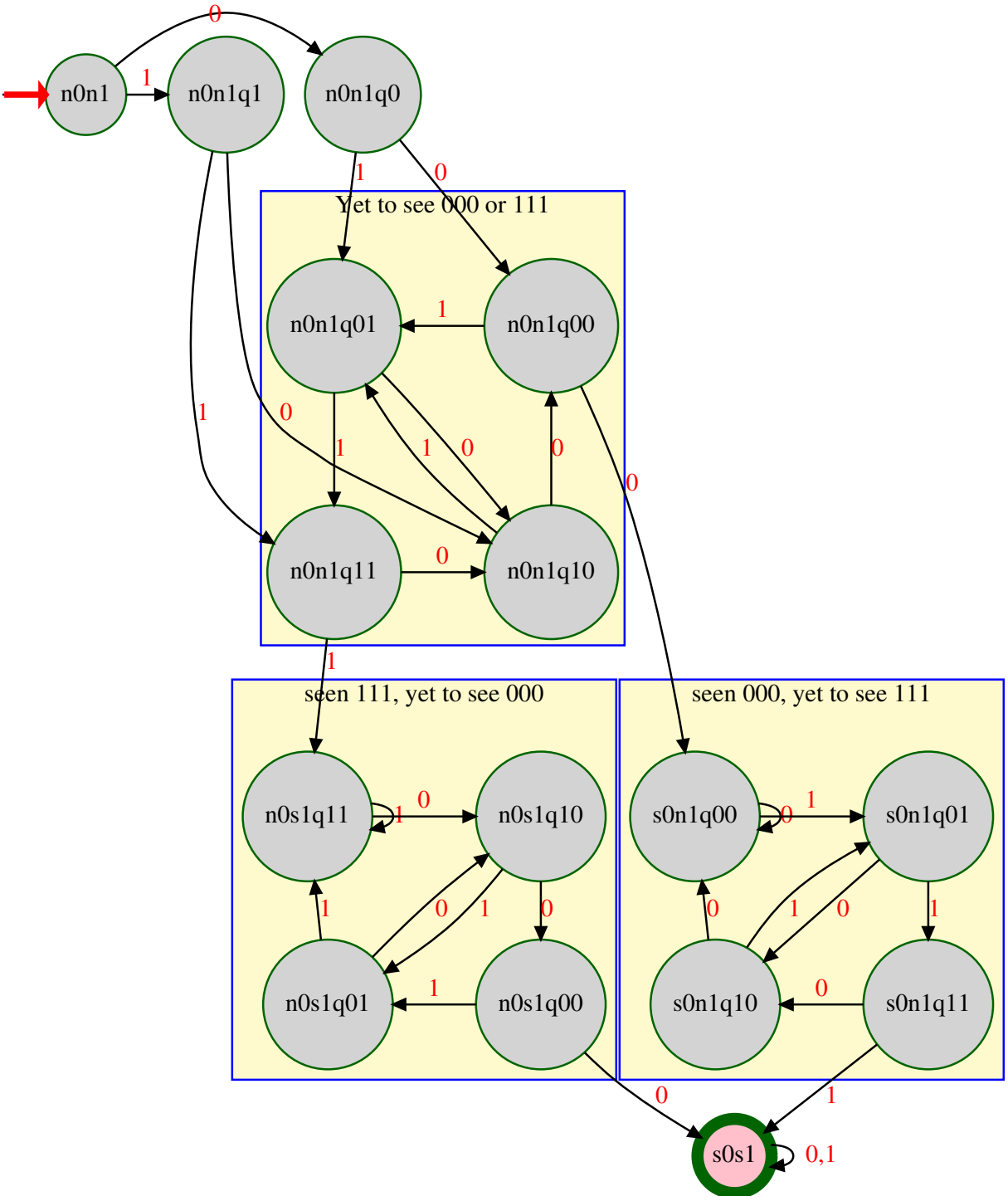
Let L be the set of all strings in $\{0,1\}^*$ that contain the substrings 000 and 111.

5.A. Describe a DFA that over the alphabet $\Sigma = \{0,1\}$ that accepts the language L. Argue that your machine accepts every string in L and nothing else, by explaining what each state in your DFA *means*.

You may either draw the DFA or describe it formally, but the states Q, the start state s, the accepting states A, and the transition function δ must be clearly specified.

Solution:

The easy way to think about this question is a product construction. We need to keep track of the last two characters we had seen, and whether we had seen 000 or 111 yet.



5.B. Give a regular expression for L, and briefly argue why the expression is correct.

Solution:

The two desired substrings are disjoint, and there are two possibilities in which order they appear. Between their appearances we need to pad with other characters. As such, we get the following:

$$(0+1)^*000(0+1)^*111(0+1)^* + (0+1)^*111(0+1)^*000(0+1)^*.$$

6 (100 PTS.) Construct this.

Let L_1 and L_2 be regular languages over Σ accepted by DFAs $M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, A_2)$, respectively.

6.A. Describe a DFA $M = (Q, \Sigma, \delta, s, A)$ in terms of M_1 and M_2 that accepts $L = L_1 \cup \overline{L_2} \cup \{\epsilon\}$. Formally specify the components Q, δ, s, and A for M in terms of the components of M_1 and M_2.

Solution:

Lets do the product construction first, and then think. States:

$$Q' = Q_1 \times Q_2 = \{(q_1, q_2) \mid q_1 \in Q_1 \text{ and } q_2 \in Q_2\} \qquad \text{Start state: } s' = (s_1, s_2).$$

Transition function

$$\delta'\big((q_1, q_2), c\big) = \big(\delta_1(q_1, c), \delta_2(q_2, c)\big).$$

The more interesting thing is the set of accepting states. We have

$$A' = \{(q_1, q_2) \in Q_1 \times Q_2 \mid q_1 \in A_1 \text{ or } q_2 \in Q_2 \setminus A_2\}$$

We are kind of done, but not! There is a subtle issue – what we built is an automata that accepts the language $L_! \cup \overline{L_2}$. Close, but no quite. We need to also accept the empty string.

To this end we add a special start state denoted by S. We modify the above construction as follows:

$$Q = Q' \cup \{S\}. \qquad s = S. \qquad A = A' \cup \{S\} \qquad \delta(t, c) = \begin{cases} \delta((s_1, s_2), c) & t = S \\ \delta'(t, c) & t \neq S. \end{cases}$$

The resulting DFA is $M = (Q, \Sigma, \delta, s, A)$.

6.B. Let $H_1 \subseteq Q_1$ be the set of states q such that there exists a string $w \in \Sigma^*$ where $\delta_1^*(q, w) \in A_1$. Consider the DFA $M' = (Q_1, \Sigma, \delta_1, s_1, H_1)$. What is the language $L(M')$? Formally prove your answer!

Solution:

The set H_1 is the set of all states such that from them you can reach an accepting state. Or put differently, there are all the states, such that if you are there, you can add some string and get into an accepting state. Namely, $L(M')$ is the language of all prefixes of L_1.

Claim 2.1. *$L(M')$ is the language of all prefixes of $L(M_1)$.*

Proof: We show inclusion in both directions:

- $L(M') \subseteq \text{prefix}(L(M_1))$: If $x \in L(M')$, then there $q_m = \delta_1^*(s_1, x) \in H_1$. By definition of H_1, there exists a string y, such that $q_e = \delta_1^*(q_m, y) \in A_1$. But this implies that

$$\delta_1^*(s_1, xy) = \delta_1^*\Big(\delta_1^*(s_1, x), y\Big) = \delta_1^*\Big(q_m, y\Big) = q_e \in A_1,$$

which implies that $xy \in L(M_1)$. That is x is a prefix of a word in $L(M_1)$.

- $\text{prefix}(L(M_1)) \subseteq L(M')$: Consider any string $w \in L(M_1)$, and any prefix x of it, and let y be the corresponding suffix (i.e., $w = xy$). Let $q' = \delta_1^*(s_1, x)$. Observe that $\delta_1(q', y) = \delta_1^*(s_1, xy) \in A_1$, which readily implies that $q' \in H_1$. This implies that $x \in L(M')$, since $\delta_1^*(s_1, x) \in H_1$, as desired.

We conclude that $\text{prefix}(L(M_1)) = L(M')$. ■

6.C. Suppose that for every $q \in A_2$ and $a \in \Sigma$, we have $\delta_2(q, a) = q$. Prove that $\epsilon \in L_2$ if and only if $L_2 = \Sigma^*$.

Solution:

Informal proof. the specified condition implies that once the automata arrives to an accepting state, it stays there for any input. As such, it never leaves an accept state. However, if $\epsilon \in L_2$, then the start state must be an accept state. As such, the automata is really just a start state that goes back into itself for any input. Since this start state is accepting, it follows that this automata accepts all languages. The other directions is of course trivial – if the language of the automata is all strings, then it, in particular, contains the empty string.

Formal proof.

Lemma 2.2. *Let $M_2 = (Q_2, \Sigma, \delta_2, s_2, A_2)$ be a DFA, such that, for all $q \in A_2$ and all $a \in \Sigma$, we have $\delta_2(q, a) = q$. We have that $\epsilon \in L_2$ if and only if $L_2 = \Sigma^*$.*

Proof: $L_2 = \Sigma^* \implies \epsilon \in L_2$: The set Σ^* contains ϵ, which implies the claim.

$\epsilon \in L_2 \implies L_2 = \Sigma^*$: By definition, $\delta^*(s_2, \epsilon) = s_2$. Since $\epsilon \in L_2$, it follows that $s_2 = \delta^*(s_2, \epsilon) \in A_2$. Now, consider any string $x = x_1 x_2 \ldots x_m \in \Sigma^*$ (here $x_i \in \Sigma$ for all i). Let

$$q_i = \delta^*(s_2, x_1 x_2 \cdots x_i), \qquad i = 0, \ldots, m.$$

We have that $q_0 = s_2$.

We claim, by induction, that $q_i = s_2$, for all i. We have that $q_0 = s_2$ since $\delta^*(s_2, \epsilon) = s_2$ by definition. Assume the claim holds for $i = 1, \ldots, k-1$, and observe that

$$q_k = \delta^*(s_2, x_1 x_2 \cdots x_k) = \delta\Big(\delta^*(s_2, x_1 x_2 \cdots x_{k-1}), x_k\Big) = \delta\Big(s_2, x_k\Big) = s_2,$$

since $s_2 \in A_2$, and by assumption. This implies that $x \in L(M_2)$, which implies that $\Sigma^* = L(M_2)$. (If one wants to be painfully formal, the proof showed that $\Sigma^* \subseteq L(M_2)$, and since $L(M_2) \subseteq \Sigma^*$. ■