4 (100 PTS.) Regular Expressions.

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. (20 PTS.) All strings that do not contain the substring 011.

> **Solution:** The idea is to take a word in the language, and treat a run of 0s as a single meta character (denoted by $0^+ = 00^*$). An appearance of the substring 0^+1, must either be the end of the string, or must be followed by 1. As such, our language must contain all the strings in
>
> $$1^*\left(0^+1\right)^*0^*.$$
>
> 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 10 (the break is in between the two characters). So $w = s_1 s_2 \dots s_k$. The string s_1 can be just a run of 1s. Otherwise, if it contains 0, then it must be of the form 0^+1. Similarly, the last string s_k can be a run of 0s. All middle strings, s_i starts with a run of 0s. The strings s_i then can have a single 1, and then it must terminate. Thus, $s_i \in 0^+1$, as claimed.
>
> Another way to look at this is that any 0 can be followed by at most a single 1. Hence, you can start with any number of 1s. Once you encounter a 0, it can be followed by as many 0s and a single 1 or it can just be followed by 0s.

4.B. (20 PTS.) All strings that do not contain the subsequence 011.

> **Solution:** A string in this language can start with any run of 1s. As soon as, it encounters the first 0, it can only have a single 1 after it. Thus,
>
> $$1^*0^*(\epsilon + 1)0^*$$

4.C. (20 PTS.) All strings that start in 00 and contain 001 as a substring.

> **Solution:** A word in the language starts with a run of two or more 0s and then it has a 1, and then we do not care bout the rest. So:
>
> $$000^*1(0+1)^*$$
>
> Note that the regular expression $00(0+1)^*001(0+1)^*$ is incorrect as it excludes the strings $001(0+1)^*$.

4.D. (20 PTS.) All string that contain either the substring 10 or the substring 01, but not both.

> **Solution:** If a string contains 10 as substring but not 01, then it cannot have any 0 before 1 and it cannot contain any 1 after 0. Hence, it can only have a run of one or more 1s followed by a run of one or more 0 i.e. it is 1^+0^+. Similaryly, strings that contain 01 as substring but not 10 are 0^+1^+. Thus:
>
> $$1^+0^+ + 0^+1^+$$

4.E. (20 PTS.) All strings in which every nonempty maximal substring of consecutive 0s is of even length. For instance 01100 is not in the language while 10000111001 is.

> **Solution:** The trick is to think about any two consecutive 0 as being one block of characters. Then, we either have blocks of 1 or blocks of 00. We can repeat these blocks as many times as we want. Thus:
>
> $$(1 + (00)^*)^*$$

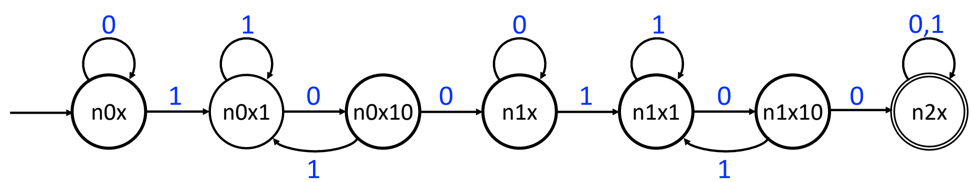
5 (100 PTS.) DFA

For each of the below languages L, describe a DFA that accepts 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.

5.A. (50 PTS.) Let L be the set of all strings in $\{0, 1\}^*$ that contain at least two occurences the substrings 100.

Solution:



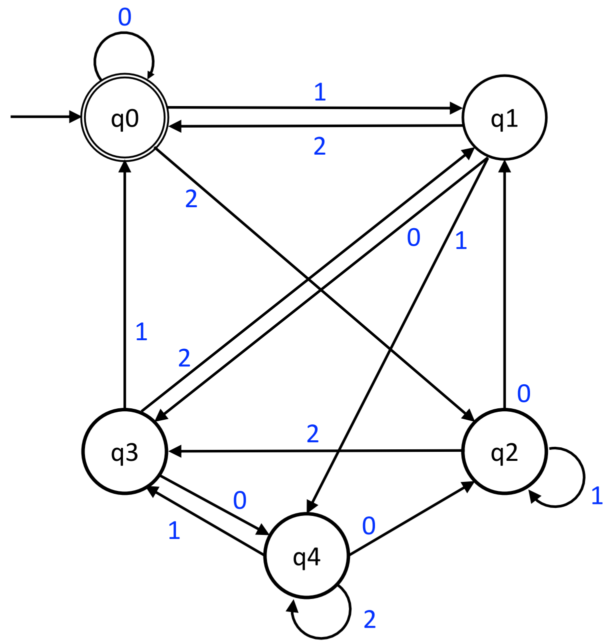
The state $n\langle i\rangle x\langle s\rangle$ corresponds to a state where i copies of 100 had been seen, and the string s was seen since then. Fortunately, we need to remember only if nothing, 1 or 10 had seen since the last pattern. Thus, the number of states is quite small. $n2x$ is the accept state and $n0x$ is the start state. The transition function is specified by the above figure.

5.B. (50 PTS.) Let L be the set of all strings in $\{0, 1, 2\}^*$ that represent ternary numbers divisible by 5 (i.e., numbers in base 3). For example, 120 would be in the language since $120_3 = 1 \cdot 3^2 + 2 \cdot 3 = 15$, while 200 would not. (Hint: It might be easier to describe this DFA than to draw it.)

Solution:

We will define the states as the number modulo 5. There are 5 such state; $Q = \{q_0, q_1, q_2, q_3, q_4\}$. Let k be the ternary number represented by the string, then the state is q_i where $i = k \mod 5$. The introduction of a new symbol a would shift the ternary number to the left. Hence, the new number becomes $3k + a$ and the new state is q_j where $j = (3k + a) \mod 5 = (3(k \mod 5) + a) \mod 5 = (3i + a) \mod 5$. Hence, the transition function $\delta(q_i, a) = q_j$ where $j = (3i + a) \mod 5$. Both the start and accept starts are q_0.

In case someone decides to draw,



6 (100 PTS.) More DFAs

(This exercise is about writing things formally – it is not difficult once you have cut through the formalism.)

6.A. (30 PTS.) Let $M = (Q, \Sigma, \delta, s, A)$ be a DFA. A state $q \in Q$ is ***bad***, if for all strings $w \in \Sigma^*$ we have that $\delta^*(q, w) \notin A$. Let $B(M) \subseteq Q$ be the set of bad states of M. Consider the DFA $M' = (Q, \Sigma, \delta, s, B(M))$. What is the language $L(M')$? Prove formally your answer!

Solution:

The set $L(M')$ is the language of all words that are not prefix of any word that is in $L(M)$. Namely, it is the complement language to $PREFIX(L)$.

Lemma 2.1. *If $x \in L(M')$ if and only if for all strings $y \in \Sigma^*$, we have that $xy \notin L(M)$.*

Proof: $\implies$: If $x \in L(M')$ then $\delta(s, x) \in B(M)$. In particular, for any string y, we have, by definition, that $\delta(q, xy) \notin A$. This implies that $xy \notin L(M)$.

$\impliedby$: Let x be a string such that for all $y \in \Sigma^*$, we have that $xy \notin L(M)$. Namely, we have that $\delta(q, xy) \notin A$, which in turn implies that $\delta(q, xy) = \delta\big(\delta(q, x), y\big) \notin A$, for all y. Bu definition, $\delta(q, x) \in B(M)$, which implies that $x \in L(M')$, as claimed. ■

6.B. (20 PTS.) Prove that if $x \in L(M')$ and $y \in \Sigma^*$, then $xy \in L(M')$.

Solution:

Proof: For any string $z \in \Sigma^*$, consider the string xyz. Since $x \in L(M')$, and by Lemma 2.1, we have that $xyz = (xy)z = x(yz) \notin L(M')$. Now, again by Lemma 2.1 ($\impliedby$) applied to xy, we have that $xy \in L(M')$, as claimed. ■

6.C. (50 PTS.) Let L_1 and L_2 be two regular languages over Σ accepted by the 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.

Describe a DFA $M = (Q, \Sigma, \delta, s, A)$ in terms of M_1 and M_2 that accepts

$$L = \{w \mid w \in L_2 \text{ and no prefix of } w \text{ is in } L_1\}$$

Formally specify the components Q, δ, s, and A for M in terms of components of M_1 and M_2.

Solution:

Lets try again. Let $\widehat{L}$ be the language of all words w such that no prefix of w is in L_1.

Let M_1 be the DFA for L_1. We add a final accept state, q_f. Any transition in M_1 into an accepting state is now translated into a transition into q_f, and furthermore, for any input, the new DFA stays at q_f. Let M_1' be the resulting automata.

Formally, $M_1' = (Q_1', \Sigma, \delta_1', s_1', A_1')$, where $Q_1' = Q_1 \cup \{q_f\}$, and

$$\delta_1'(q, a) = \begin{cases} q_f & q = q_f \\ q_f & q \in A_1 \text{ or } \delta_1(q, a) \in A_1 \\ \delta_1(q, a) & \text{otherwise.} \end{cases}$$

Also, we set $A_1' = A_1 \cup \{q_f\}$ and $s_1' = s_1$.

Clearly, we have

$$L(M_1') = \{w \in \Sigma^* \mid \exists x, y \in \Sigma^* \text{ s.t. } x \in L_1\}$$

The language $\widehat{L}$ ("no prefix of w is in L_1") is the complement of the language $L(M_1')$. In particular, let M_1'' be the machine resulting from inverting the accept states of M_1'. Formally, $M_1'' = (Q_1', \Sigma, \delta_1', s_1'', A_1'')$, where $s_1'' = s_1'$ and $A_1'' = Q_1' \setminus A_1'$. Clearly, $L(M_1'') = \widehat{L}$. Now, the desired language is $L(M_2) \cap L(M_1'')$. As such, all we need to do now, is an explicit product construction of M_2 and M_1''. And this is easy using what we had seen in class:

$$\begin{aligned} &Q = Q_1'' \times Q_2 \\ &\Sigma = \Sigma \\ &\delta = \delta_1'' \times \delta_2 \equiv \delta(q_1, q_2) = \Big(\delta_1''(q_1), \delta_2(q_2)\Big) \\ &s = (s_1'', s_2) \\ &A = A_1'' \times A_2. \end{aligned}$$

The DFA M is then the tuple

$$M = \Big(Q, \Sigma, \delta, s, A\Big)$$