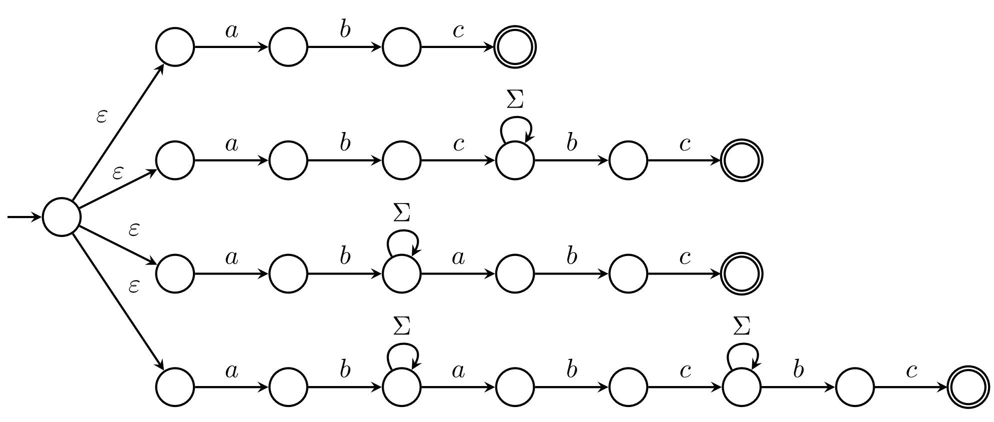
CS 181 Spring 2020 Homework Week 3 Solutions

1. For readability, we break down the solution into reusable pieces. Let $S = a \cup b \cup c$. Then the final answer is:
$$L_1 = S^*\big(a(SSS)^*a \cup b(SSS)^*b \cup c(SSS)^*c\big)S^*.$$
The $(SSS)^*$ part ensures that the identical symbols are separated by a string of length $0 \mod 3$, and the S^* on each end represent the idea that the special substring can be contained in any string. The three cases of a, b, and c are handled separately with "$\cup$", since the language only requires at least one to occur in a string.

2. (a) Notice that $\varepsilon \in a^*$ and $\varepsilon \in (aa)^*$, hence, the GNFA recognizes $ababba$ following a computational path
$$q_1 \xrightarrow{ab} q_2 \xrightarrow{\varepsilon} q_1 \xrightarrow{ab} q_1 \xrightarrow{\varepsilon} q_2 \xrightarrow{ba} q_a.$$
Now consider the string $aabbba$. Since the first two symbols are aa, there is only one possible option for the first transition: $q_0 \xrightarrow{a} q_2$. Observe that the edge $q_1 \xrightarrow{b^*} q_a$ can be used only as an ε-transition because the string ends with a. Finally, we notice that the other transitions (between q_1 and q_2, loops, and $q_2 \to q_a$) cannot generate three consecutive b's, so the GNFA does not generate $aabbba$.

   (b) There are many valid strings, including aab, $abab$, and $aaaba$, and $aaaab$. Two possible ways the GNFA can accept $aaaab$ are below:
$$q_0 \xrightarrow{a} q_2 \xrightarrow{aa} q_2 \xrightarrow{ab} q_a \qquad \text{and} \qquad q_0 \xrightarrow{a} q_2 \xrightarrow{\varepsilon} q_1 \xrightarrow{aa} q_2 \xrightarrow{ab} q_a.$$

3. Notice that a substring abc can overlap with the first two symbols ab, with the last two symbols bc, with both or with neither. Thus, L_3 is exactly the language of words that can be represented by at least one of the following forms: abc, $abc\Sigma^*bc$, $ab\Sigma^*abc$ or $ab\Sigma^*abc\Sigma^*bc$ where Σ^* denotes arbitrary (possibly empty) substrings. Hence, we can combine four standard NFAs for pattern matching using nondeterminism. The following NFA recognizes L_3.



4. L_4 is the language of strings that start with 0 (accepted by the bottom branch) or do not contain 000 as a substring (accepted by the top branch). Notice that there are strings which have different accepting paths, namely, all strings that start with 0 and do not contain 000.

5. L_5 is generated by the following CFG: $S \to \varepsilon \mid 00S1$.

6. Suppose L_6 were a FSL, then the pumping lemma would apply. Let p be the number in the pumping lemma, and choose $s = 1^{3p}0^p$. Since $s \in L$, and $|s| \geq p$, we know there exist substrings x, y, z such that s can be written $s = xyz$, where: $|xy| \leq p$, $|y| \geq 1$; and for all $i \geq 0$, $xy^iz \in L$. Then since xy is a prefix of s and $|xy| \leq p < 3p$, x and y must be entirely contained in the block of symbols 1^{3p}, and so xy must consist entirely of 1's. Since, $|x| \geq 0$, for some $n \geq 0$, $x = 1^n$. Similarly, since $|y| \geq 1$, for some $m \geq 1$, $y = 1^m$. So
$$s = \underbrace{1^n}_{x}\underbrace{1^m}_{y}\underbrace{1^{3p-n-m}0^p}_{z}.$$
We can confirm this by checking the arithmetic for the number of 1's in $s = xyz$, namely: $n + m + (3p - n - m) = 3p$. Then by the lemma, we can choose $s' = xy^2z$, which should also be in L_6. Then
$$s' = xy^2z = s = \underbrace{1^n}_{x}\underbrace{1^{2m}}_{y^2}\underbrace{1^{3p-n-m}0^p}_{z},$$
so combining the 1's shows that s' is of the form $1^{3p+m}0^p$. Since $|y| = m \geq 1$, we know that $p+m > p$; and therefore s' is not of the form $1^{3p}0^p$. To put it another way: $m > 0$, and in s' we added m 1's to s without adding any 0's. Therefore the number of 1's in s' cannot still be equal to 3 times the number of 0's. Thus, $s' \notin L_6$, which violates the Pumping Lemma showing that L_6 does not obey the Pumping Lemma. Therefore, L_6 is not a FSL by contradiction. *qed*

   Note: We could do this with a choice for s of something more like "$1^p0^{p/3}$", but then we would have to worry about the fact that the pumping lemma does not necessarily guarantee that p is divisible by 3. A solution which assumes p is divisible by 3 without any justification would be incorrect. So you would either have to try to justify that assumption or make the definition of the string, s, a little more complicated to guarantee that the "exponent" on the 0 is an integer. People often try to make s as short as possible — probably because they think it is "more mathematical" or shows more mathematical sophistication. But there is nothing in mathematics that says a shorter choice for s is any better than a longer one, and the latter approach just makes the proof harder to read. Generally, it is better to choose an s that makes the proof simple and clear, rather than trying to make s as short as possible. Here's a follow-up question for those of you who are interested: given the definition of L_6, is it even true that p has to be divisible by 3? If not, does it have to be divisible by some other number? (OK, I know. That's two follow-up questions.)