CS 181 Spring 2020 Homework Week 4 Solutions

1. This is finite state, where one possible regular expression is

$$(b \cup c)^* \big(ab(b \cup c)^* ab(b \cup c)^*\big)^*.$$

Constructing this regular expression proceeds as follows. We want an even number of a's, so we want zero or more repetitions of a string that includes two a's; thus we will use the Kleene star operator on an expression that involves two a's. Moreover these a's must be followed by a b, so that Kleene star operator must be on an expression that involves two ab's. These ab's might be separated by an arbitrary string of b and c characters, which motivates the form $(ab(b \cup c)^* ab(b \cup c)^*)^*$. Finally, we want strings to have optional b and c characters in the beginning, so we introduce an arbitrary prefix $(b \cup c)^*$ and prepend it to the previous expression.

2. This language is CF but not FS.

Claim 1. $L_1 = \{w \in \Sigma^* \mid w = a^m b^n c^k,\ n = m + k\}$ *over the alphabet* $\Sigma = \{a, b, c\}$ *is not FS.*

Proof. Suppose L_1 is a FS language and hence has a corresponding pumping length p, such that strings in the language that are of length at least p satisfy the conditions of *the pumping lemma*. Consider string $s = a^p b^{2p} c^p \in L_1$. There exists some decomposition $s = xyz$ such that

$$\begin{aligned}
&|y| > 0,\\
&|xy| \leq p,\\
&xy^i z \in L_1 \quad \forall i = 0, 1, 2, \ldots.
\end{aligned}$$

Since $|xy| \leq p$, it must be the case that y consists of only a's. Since $|y| > 0$, $y = a^\beta$ for some $\beta \geq 1$. If we pump down (i.e. select $i = 0$), then we would be decreasing the count of a characters to $p - \beta$. It is critical that $\beta > 0$ so that we are actually decreasing the count by a nonzero amount. The pumping lemma guarantees this, since $\beta = |y| > 0$. By changing the count of a characters in s, it must be the case that the total count of b characters exceeds the total count of a and c characters. Symbolically,

$$xy^0 z = xz = a^{p-\beta} b^{2p} c^p \notin L_1 \qquad \text{because } (p - \beta) + p \neq 2p.$$

This contradicts the condition $xy^i z \in L_1\ \forall i \geq 0$, so our assumption is false. Hence, L_1 is not FS. □

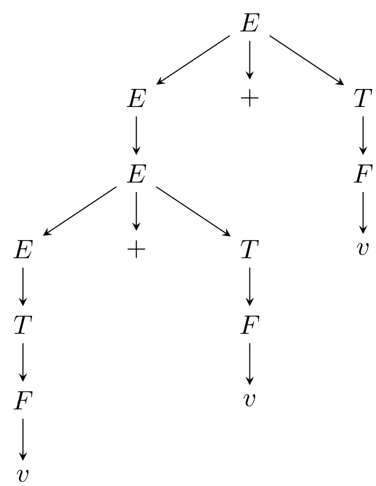
Claim 2. $L_1 = \{w \in \Sigma^* \mid w = a^m b^n c^k,\ n = m + k\}$ *over the alphabet* $\Sigma = \{a, b, c\}$ *is context-free.*

Proof. We show a context-free grammar for this language:

$$\begin{aligned}
S &\to AC\\
A &\to aAb \mid \varepsilon\\
C &\to bCc \mid \varepsilon.
\end{aligned}$$

This construction works because a b is added for every occurrence of a and c, so certainly any input generated should satisfy the condition that the number of a's and number of c's sum to the number of b's. In other words, A adds a's on the left and the same number of b's in the center while C adds c's on the right and the same number of b's in the middle. Conversely we must then argue that any string in the language is generated by the grammar. To generate $w = a^m b^n c^k$ where $n = m + k$, one first invokes $S \to AC$. For the A branch, use rule $A \to aAb$ m times followed by a $A \to \varepsilon$ so that the left hand side of the string is $a^m b^m$. Similarly expand the C branch using the rule $C \to bCc$ k times followed by a $C \to \varepsilon$ so that the right hand side of the string is $b^k c^k$. The final string is then $a^m b^{m+k} c^k = a^m b^n c^k$. Hence this grammar generates L_1. $\square$

3. (a) The parse tree for $v + v + v$:



(b) The right-most derivation for e (expanding the right-most variable):

$$\begin{aligned}
\underline{E} &\to E + \underline{T} \\
&\to E + \underline{F} \\
&\to \underline{E} + v \\
&\to E + \underline{T} + v \\
&\to E + \underline{F} + v \\
&\to \underline{E} + v + v \\
&\to \underline{T} + v + v \\
&\to \underline{F} + v + v \\
&\to v + v + v.
\end{aligned}$$

(c) The left-most derivation for e (expanding the left-most variable):

$$\begin{aligned}
\underline{E} &\to \underline{E} + T \\
&\to \underline{E} + T + T \\
&\to \underline{T} + T + T \\
&\to \underline{F} + T + T \\
&\to v + \underline{T} + T \\
&\to v + \underline{F} + T \\
&\to v + v + \underline{T} \\
&\to v + v + \underline{F} \\
&\to v + v + v.
\end{aligned}$$