Assignment 3

COL 352
Introduction to Automata & Theory of Computation

Problem 1

Give context-free grammars generating the following sets

(a) the set of all strings over alphabet $\{a, b, ., +, ^*, (,), \epsilon, \phi\}$ that are well-formed regular expression over alphabet $\{a, b\}$. Note that we must distinguish between ε as the empty string and ϵ as a symbol in the regular expression

(b) The set of all strings over alphabet $\{a, b\}$ not of the form ww for some string w

Solution:

(a) Let $\Sigma = \{a, b, ., +, ^*, (,), \epsilon, \phi\}$. Consider the following CFG G -

$$\begin{aligned} S &\rightarrow \phi \mid B \\ B &\rightarrow a \mid b \mid \epsilon \mid B + B \mid B.B \mid (B) \mid B^* \end{aligned}$$

To show that S represents the language L of all the well formed regular expressions over alphabet $\{a, b\}$.

I) Claim : $L(G) \subseteq L$ (any string generated by G is a well formed regular expression)

Proof : By induction on the number of production rules through which a string is generated.

Basis : For $n = 1$, ϕ can only be generated with one production rule.

Induction Hypothesis: Strings generated with $\leq n$ production rules are be well formed regular expressions.

Let us assume the Induction Hypothesis is true for k.

Induction Step : For $n = k + 1$, The last rule applied will be $S \rightarrow B$. The second last rule can be:

1. $B \rightarrow B + B$: From induction hypothesis (and the fact that the last rule for $n = k$ also is $S \rightarrow B$), each B will lead to a well formed regular expression, say r_1 and r_2. And, $r_1 + r_2$ is a regular expression. Similarly, other cases.
2. $B \rightarrow B.B$
3. $B \rightarrow B^*$
4. $B \rightarrow (B)$
5. Trivial Cases: $B \rightarrow a \mid b \mid \epsilon$

II) Claim : $L \subseteq L(G)$ (any well formed regular expression can be generated from the Grammar G)

Proof: By induction on length of regular expression r.

Basis : For $n = 0$, ϕ can be generated from G.

Induction Hypothesis: All regular expressions of length less than or equal to n can be generated by G.

Let us assume the induction hypothesis is true for $n = k$.

Induction Step : For $n = k + 1$, a regular expression r can be formed via

1. $r_1 + r_2$: From induction hypothesis, r_1 and r_2 can be generated by G through a series of steps, say S_1 and S_2. Then, using the rule $S \to B$ and $B \to B + B$ followed by S_1 and S_2, we can generate the regular expression. Similarly, for other cases.
2. $r_1.r_2$
3. (r_1)
4. r_1^*

From I and II, $L(G) = L$. Thus, G is the required grammar.

b) Let $\Sigma = \{a, b\}$, consider the following CFG G -

$$S \to AB \mid BA \mid A \mid B$$
$$A \to CAC \mid a$$
$$B \to CBC \mid b$$
$$C \to a \mid b$$

To prove that this grammar generates set of all strings not of the form ww

I) Claim : $L(G) \subseteq L$ (any string generated by G is not of the form ww)

Proof : Let the length of the string generated by A and B be $2m + 1$ and $2n + 1$ respectively.

The length of w to form ww would be $m + n + 1$. The middle a of the string generated from A is at a distance $m+1$ and the middle b of the string generated from B is at a distance of $2m+n+2(= 2m+1+b+1)$ from the beginning.

This means that the $(m+1)^{th}$ character of the first w, w_1, is a and the $(m+1)^{th}(= (2m+n+2)-(m+n+1))$ character of the second w, w_2, is b. Therefore, $w_1 \neq$w$_2$.

Hence this grammar generates all strings not of the form ww.

II) Claim : $L \subseteq L(G)$ (all strings not of the form ww can be generated by the grammar)

Proof: Consider a string x not of the form ww

Case 1 - $|x|$ is odd. Proof by induction on the length of x

Basis : $x = a$ or $x = b$ can be derived using the rules $S \to A \to a$ and $S \to B \to b$

Induction Hypothesis : All odd length strings x, such that $|x| \leq n$ can be derived from grammar, i.e.,

$S \to A \xrightarrow{*} x$ or $S \to B \xrightarrow{*} x$

Induction Step : Let x' be the next odd length string such that $|x'| = n + 2$. Then,

$$S \to A$$
$$S \to CAC$$
$$S \to CxC$$

or replace A by B. $x' = CxC$ such that $|x'| = |CxC| = n + 2$

Therefore all strings of odd length can be generated from the grammar

Case 2 - $|x|$ is even

Since x is not of type ww, there exists atleast one i such that $x_i \neq x_{i+|x|/2}$.

We can replace x_i and $x_{i+|x|/2}$ by A and B and the others by C. Then x can be viewed as:

$$(CC...C)_{i-1}A(CC...C)_{i-1}(CC...C)_{j-1}B(CC...C)_{j-1}$$

such that $(i - 1) + (j - 1) + 1 = |x|/2$. From induction hypothesis, this string can be generated by our grammar, and thus all even length strings can be generated.

From I and II, $L(G) = L$. Thus, G is the required grammar.

Problem 2

Show that the language $L = \{a^i b^j c^k \mid i < j < k\}$ is not context free.

Solution : We can prove this via Pumping Lemma. Let the pumping constant be n.

Consider the string $S = a^n b^{n+1} c^{n+2} \in L$. Let $S = uvwxy$ where $|vx| \geq 1$ and $|vwx| \leq n$.

The following cases arise:

1. vwx is in a^n: For $i = 2$, $S' = uv^i wx^i y$ has more(or equal) a's than b's $\implies S' \notin L$
2. vwx is in b^n: For $i = 0$, $S' = uv^i wx^i y$ has more(or equal) a's than b's $\implies S' \notin L$
3. vwx is in c^n: For $i = 0$, $S' = uv^i wx^i y$ has more(or equal) b's than c's $\implies S' \notin L$
4. vwx contains both a and b i.e. is across $a^n b^{n+1}$: Since x has at least one b, for $i = 2$, $S' = uv^i wx^i y$ has more(or equal) b's than c's $\implies S' \notin L$.
5. vwx contains both b and c i.e. is across $b^{n+1} c^{n+2}$: Since v has at least one b, for $i = 0$, $S' = uv^i wx^i y$ has more(or equal) a's than b's $\implies S' \notin L$.

So, by Pumping Lemma, the given language is not context free.

Problem 3

Show that the language $L = \{a^i b^j \mid i \neq j \text{ and } i \neq 2j\}$ is a CFL.

Solution : Define

$$L_1 = \{a^i b^j \mid i < j\}$$
$$L_2 = \{a^i b^j \mid j < i < 2j\}$$
$$L_3 = \{a^i b^j \mid i > 2j\}$$

Claim : $L = L_1 \cup L_2 \cup L_3$ is a CFL.

Proof : Since union of CFLs is a CFL, the problem reduces to providing a CFG for each of L_1, L_2 and L_3

I) CFG_1 for L_1

$$S \to AB \mid B$$
$$B \to bB \mid b$$
$$A \to aAb \mid ab$$

A produces strings with equal number of a's and b's. B produces strings containing only b's. When concatenated, S produces strings with a's followed by b's where number of b's is greater than a's.

Alternately, any string in L_1 can be split into a string containing equal number of a's and b's followed by only b's. The first string can be generated by A and the other by B. So, $L(CFG_1) = L_1$

II) CFG_2 for L_2

$$S \to aEb$$
$$E \to aEb \mid D$$
$$D \to aaDb \mid aab$$

D generates strings with a's followed by b's where number of a's is double than that of b's. Say, number of a's $= 2x$ and number of b's $= x$. $(x \geq 1)$

E concatenates a's in the front and an equal number of b's in the end. Let, y be number of a's (and b's) added through this production rule where $y \geq 0$.

S concatenates an a in the front and b at the end. So, the resulting string is a's followed by b's where number of a's i.a $n_a = 2x + y + 1$ and number of b's i.e. $n_b = x + y + 1$. Clearly, $n_a > n_b \because x \geq 1$ and $n_a < 2n_b \because y \geq 0$. So, $L(CFG_2) \subseteq L_2$.

Also, any string in L_2 of the form $a^i b^j$ can be split as $a^{2x} a^y b^y b^x$ where $x = i - j$ and $y = 2j - i$ which are valid because of the constraints in L_2. So, $L_2 \subseteq L(CFG_2)$.

Hence, $L(CFG_2) = L_2$

III) CFG_3 for L_3

$$S \to AX \mid A$$

$$B \to aA \mid a$$

$$X \to aaXb \mid aab$$

$L(CFG_3) = L_3$. (Analysis similar to CFG_1.)

From I, II and III, L_1, L_2 and L_3 are CFLs, so L is a CFL