UMC-205 Assignment-4

Aditya Gupta
SR No: 22205

Question 1

We want to construct a construct free grammar for the language:

$$L = \{xy \in \{0,1\}^* \mid |x| = |y| \text{ and } y \text{ contains a } 1\}$$

Consider the grammar $G = (N, A, S, P)$ where $N = \{S, T, P, X\}$, $A = \{0, 1\}$ and $S = S$. The production rules P are:

$$\begin{aligned} S &\to XSX \mid T \\ T &\to 0P1 \mid 1P1 \\ P &\to XPX \mid \epsilon \\ X &\to 0 \mid 1 \end{aligned}$$

The symmetric nature of S, T and P ensures that the length of x and y are equal. The production rules of T ensure that y contains a 1. The production rules of X ensure that the string is made up of 0s and 1s. The string formed as a result will be of the form:

$$(0+1)^n \cdot (0+1) \cdot (0+1)^m \cdot (0+1)^m \cdot 1 \cdot (0+1)^n$$

where $n, m \geq 0$. This is the required language.

Question 2

The context free grammar G given to us is:

$$\begin{aligned} S &\to AAA \mid B \\ A &\to aA \mid B \\ B &\to \epsilon \end{aligned}$$

To convert this grammar to the Chomsky Normal Form, we first construct the language G' using the productions of G and the rules:

1. If $A \to \alpha B \beta$ and $B \to \epsilon$ then add the production $A \to \alpha\beta$
2. If $A \to B$ and $B \to \gamma$ then add the production $A \to \gamma$

Using this method, we get the language G' as:

$$
\begin{align}
&S \to AAA \text{ (from } G\text{)} \tag{1}\\
&S \to B \text{ (from } G\text{)} \tag{2}\\
&A \to aA \text{ (from } G\text{)} \tag{3}\\
&A \to B \text{ (from } G\text{)} \tag{4}\\
&B \to \epsilon \text{ (from } G\text{)} \tag{5}\\
&S \to \epsilon \text{ (from rule 1 on (2) and (5))} \tag{6}\\
&A \to \epsilon \text{ (from rule 1 on (4) and (5))} \tag{7}\\
&S \to AA \text{ (from rule 1 on (1) and (7))} \tag{8}\\
&A \to a \text{ (from rule 1 on (3) and (7))} \tag{9}\\
&S \to A \text{ (from rule 1 on (7) and (8))} \tag{10}\\
&S \to a \text{ (from rule 2 on (9) and (10))} \tag{11}
\end{align}
$$

By dropping the unit and ϵ productions from G', we get the language G" as:

$$
\begin{aligned}
&S \to AAA\\
&A \to aA\\
&S \to AA\\
&A \to a\\
&S \to a
\end{aligned}
$$

To convert G" to Chomsky Normal Form, we introduce two new non-terminal symbols X and Y. Using these, we write the productions of the Chomsky Normal Form as:

$$
\begin{aligned}
&S \to AX \mid AA \mid a\\
&A \to YA \mid a\\
&X \to AA\\
&Y \to a
\end{aligned}
$$

Note that the language generated by this grammar is the same as the language generated by the original grammar G, apart from the fact that this does not generate the empty string ϵ. To account for this, we could add the transition $S \to \epsilon$ to the Chomsky Normal Form grammar.

Question 3

We have the language $L = \{a^n b^{n^2} \mid n \geq 0\}$. To show that this does not satisfy the Pumping Lemma, we need to show that for every $k \geq 0$, there exists a word $z \in L$ such that $|z| \geq k$ and for all u, v, w, x, y such that $z = uvwxy, |vwx| \leq k, vx \neq \epsilon$, there exists an $i \geq 0$ such that $uv^iwx^iy \notin L$.

Let $k \geq 0$ be given. Consider the word $z = a^k b^{k^2}$. We can write $z = uvwxy$ in the following ways:

1. vwx has only a's: For $i = 0$, $uv^iwx^iy = a^{k-|vx|}b^{k^2} \notin L$ as $k - |vx| < k$.
2. vwx has only b's: For $i = 0$, $uv^iwx^iy = a^k b^{k^2-|vx|} \notin L$ as $k^2 - |vx| < k^2$.
3. vwx has both a's and b's: In this case, v must have atleast one a and x must have atleast one b. Consider $i = 0$. Then uv^iwx^iy will have atmost $k - 1$ a's. Now $|vwx| \leq k$. Thus vwx can have atmost $k - 1$ b's. Then uv^iwx^iy will have atleast $k^2 - (k - 1) = k^2 - k + 1$ b's. However $k^2 - k + 1 > k^2 - 2k + 1 = (k - 1)^2$ since $k \geq 2$ (atleast one a and atleast one b). Thus in uv^iwx^iy, $(max\{\#a\})^2 < min\{\#b\}$. Clearly, $uv^iwx^iy \notin L$.

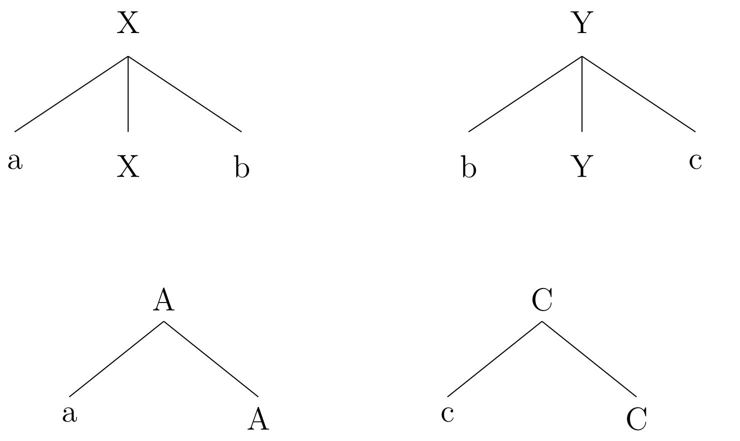
Thus we can see that the language L does not satisfy the Pumping Lemma and hence is not context free.

Question 4

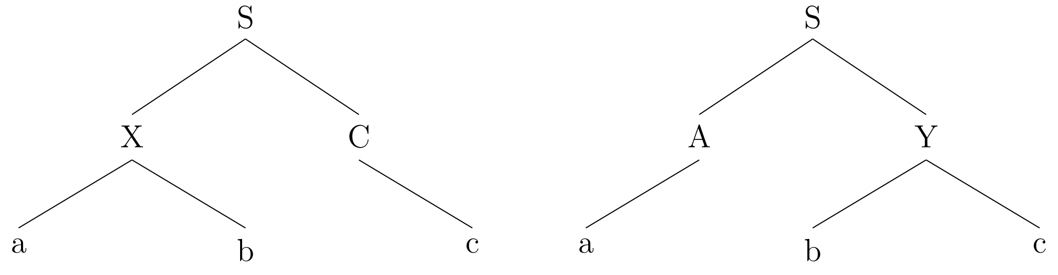
a. X produces strings of the form $a^n b^n$ where $n \geq 1$. C produces strings of the form c^m where $m \geq 1$. A produces strings of the form a^p where $p \geq 1$. Y produces strings of the form $b^q c^q$ where $q \geq 1$. From S, we get either XC or AY. Thus we get either $a^n b^n c^m$ or $a^p b^q c^q$. The language generated by this grammar can be written as:

$$L(G) = \{a^p b^q c^r \mid p = q \text{ or } q = r \text{ where } p, q, r \geq 1\}$$

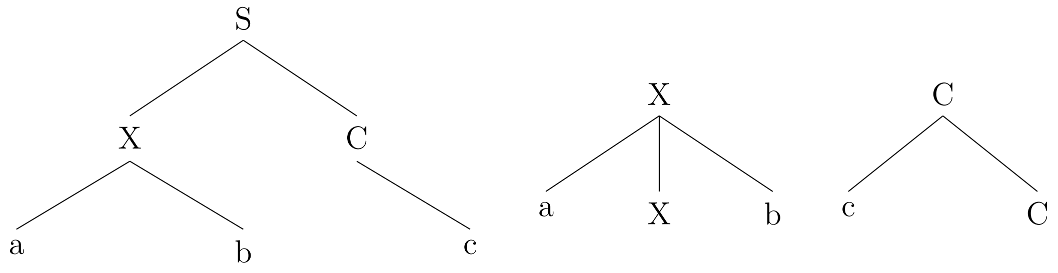
b. (i) The basic pumps for G are:



(ii) The $\leq$-minimal parse trees for G are:

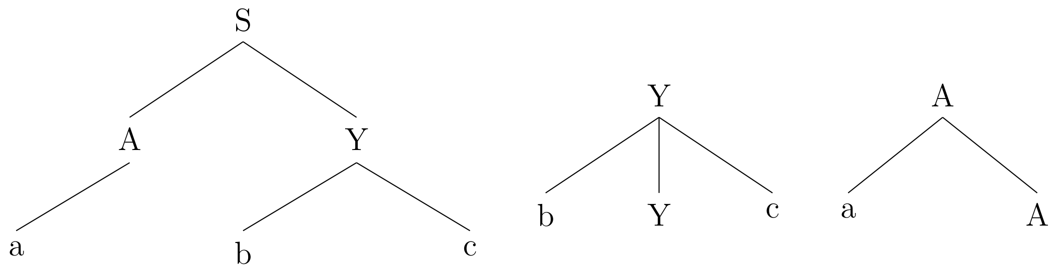


(iii) The semi-linear expression for $\psi(L(G))$ is given by:
(a) $a^p b^q c^r$ where $p = q$



Expression obtained $= (1,1,1) + \langle\langle(1,1,0) + (0,0,1)\rangle\rangle$

(b) $a^p b^q c^r$ where $q = r$



Expression obtained $= (1,1,1) + \langle\langle(1,0,0) + (0,1,1)\rangle\rangle$

Hence the net semi-linear expression for $L(G)$ is:

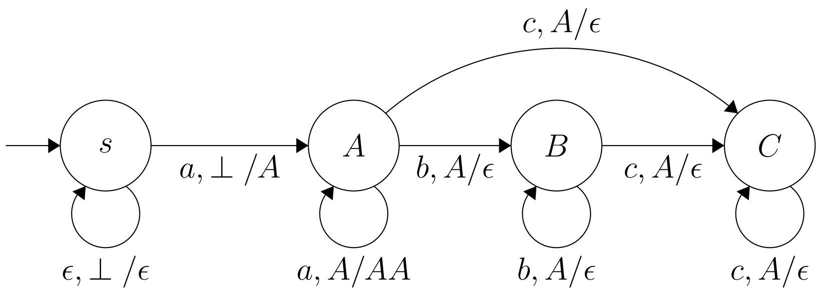
$\psi(L(G)) = \{(1,1,1) + \langle\langle(1,1,0) + (0,0,1)\rangle\rangle\} \cup \{(1,1,1) + \langle\langle(1,0,0) + (0,1,1)\rangle\rangle\}$

c. To obtain a letter equivalent regular expression, replace every 1 in the semi linear set by the corresponding letter. Thus the regular expression that is letter equivalent to $L(G)$ is:

$$abc \cdot (ab + c)^* + abc \cdot (bc + a)^* = abc \cdot ((ab + c)^* + (bc + a)^*)$$

Question 5

The transition diagram of a PDA for the language $L = \{a^l b^m c^n \mid l = m + n\}$ is:



This PDA has states $Q = \{s, A, B, C\}$, start state s, input alphabet $A = \{a, b, c, \epsilon\}$, stack alphabet $\Gamma = \{A, \perp\}$, bottom of stack symbol $\perp$ and transitions δ as shown in the diagram. The PDA accepts by empty stack.

When we read an a, we enter the state A. We keep pushing an A on the stack everytime we read an a. Thus the stack represents the total number of a's read. When we read b, we enter the state B. We keep popping an A from the stack everytime we read b. When we read c, we enter the state C. We keep popping an A from the stack everytime we read c. Since we could directly encounter c after an a, we also add a direct transition from state A to state C to account for this. This way the stack would be empty only if the total number of a's read is equal to the total number of b's and c's read. This is the required language.

Question 6

The class of context free languages is closed under the prefix operation.

Let L be a context free language. Let $G = (N, A, S, P)$ be a CFG generating L in the Chomsky Normal Form. We construct a new CFG $G' = (N', A, S, P')$ as follows:

a. For every non terminal X in N, add X and a new non terminal X' to N'.

b. Add every production in P to P'.

c. For every production $S \to AB$ in P, add the productions $S \to AB'$ and $S \to A'$ to P'.

d. For every production $X \to YZ$ in P, add the productions $X' \to YZ'$ and $X' \to Y'$ to P'.

e. For every non terminal X in N, add the production $X' \to \epsilon$ to P'.

Claim: L$(G') \equiv$ pref(L), where pref(L) is the set of all subsets of all strings in L

Proof: We show set containment in both directions.

(i) pref(L) $\subset$ L(G'):

Firstly, since G' contains all the productions from G, it can form all the strings that G can form. Now we need to show that it can also form the prefixes of these strings.

The grammar G is in the Chomsky Normal Form. This means that we can picture the steps in the construction of any string by a binary tree, whose terminal roots give us the string when read in inorder. Consider a string $w = a_1a_2 \dots a_k$ in the language L. If $|w| = 1$, then it is a direct transition from S to a terminal. Its only prefix is ϵ which is formed by the production $S \to X \to \epsilon$.

If $|w| \geq 2$, then we can divide the strings into two parts based on the initial transition $S \to XY$. We can say that the non terminal X derives the string $a_1a_2 \dots a_{k'}$ and the non terminal Y derives the string $a_{k'+1}a_{k'+2} \dots a_k$ for some $k' < k$. Now suppose we need to form a prefix $w' = a_1a_2 \dots a_j$ of w. We construct this by considering two cases:

a. **$j < k'$:** Start with the transition $S \to X'$ instead of $S \to XY$. This ensures that the length of the final string is less than k' if we follow the construction of w in G.

b. **$j \geq k'$:** Start with the transition $S \to XY'$ instead of $S \to XY$. This ensures that the length of the final string is atleast k' if we follow the construction of X as it was in w without any change, and then continue with the construction of Y to form the rest of the string.

Once we have done this, we can then inductively repeat the same process for the next level of the binary tree. Using this, we can limit the length of our word by stopping the construction at any point. This way, we can form any prefix of any string in L using the context free grammar G'. Thus pref(L) $\subset$ L(G').

(ii) L(G') $\subset$ pref(L):

To show this, we observe that the derivation of the string at any point consists of atmost one non terminal of the form X' and that it will always be at the rightmost position. We can show this by induction.

For the first transition from S, it is clear that this property holds. If we choose $S \to XY$, then we proceed with the construction as we would for a string in L. If we choose $S \to XY'$ or $S \to X'$, then we get one non terminal of the form X' at the rightmost position. Now assume this holds for a derivation of length n. Since we can only get a X' type non terminal from another X' type non terminal, we have to construct it at the rightmost position. Thus the claim holds by induction.

Now observe that the X' type non terminals can only terminate in ϵ. So, if we continue with the derivation for any string, at some point we send the X' to ϵ and stop the growth of the string. This cuts off some of the rightmost parts of the string, without affecting the leftmost parts. Thus the obtained string is a prefix of the original string. Therefore, L(G') $\subset$ pref(L).

Since the prefix language can be represented by the context free grammar G', we conclude that the class of context free languages is closed under the prefix operation.