1. Find CFG for the following languages.

$$L = \{a^n b^m : n \neq 2m, n \geq 0, m \geq 0\}$$

Ans) To find CFG for above languages, we should think about two cases.

case 1: $n < 2m$

$S_1 \to aaS_1b \mid aA \mid A$

$A \to bA \mid b$

case 2: $n > 2m$

$S_2 \to aaS_2b \mid B$

$B \to aB \mid a$

Then, we will combine two cases production rule to make a CFG for L.

$S \to S_1 \mid S_2$

$S_1 \to aaS_1b \mid aA \mid A$

$A \to bA \mid b$

$S_2 \to aaS_2b \mid B$

$B \to aB \mid a$

2. Show that the grammar is unambiguous.

$$G_1 = (\{A, S\}, \{a, b\}, S, P_1)$$

$$S \to aAb \mid \lambda$$

$$A \to aAb \mid \lambda$$

$$L(G_1) = \{a^n b^n : n \geq 0\}$$

Ans To prove the grammar is unambiguous, we should show that all strings are parsed in one way.

To show this, we will introduce inductive step. no λ-rules except one λ in S.

i) $n = 0$.

If $n = 0$, then G_1 generates λ.

λ has only one parse tree here.

ii) $n \geq 1$.

In this case, $S \to aAb$ must be used initially because $S \to \lambda$ derives only λ and is not applicable for $n \geq 1$.

Then, the string aAb will be turned $a^n b^n$

This requires exactly $n-1$ more pairs of a and b to be produced from A.

And A will recursively expand until $A \to \lambda$ is used.

$$A \to aAb \to aaAbb \to \cdots\cdots \to a^{n-1}Ab^{n-1}.$$

At each step, there is no alternative but to use $A \to aAb$ until reaching $A \to \lambda$.

It means that there is no other production rule for S or A that could produce $a^n b^n$ in any other form or order.

This proves the grammar G_1 is unambiguous by showing each string in the $L(G_1)$ has exactly one parse tree corresponding to its derivation.

3. Convert the grammar into CNF.

$$S \to baAB,$$
$$A \to bAB \mid \lambda,$$
$$B \to BAa \mid A \mid \lambda$$

Ans

· Definition of CNF.

→ In the CFG, All production rules must be followed below form.

$$A \to BC \quad \text{or} \quad A \to a$$
$$(\because A, B, C \in V,\ a \in T)$$

To convert CFG into CNF, we should take 5 steps.

I) Add new start symbol S_0 (and add rule $S_0 \to S$)

II) Remove λ rules from the tail (before : $B \to xAy$ and $A \to \lambda$, after : $B \to xAy \mid xy$)

III) Remove unit productions $A \to B$ (by the head) ($A \to B$ and $B \to xCy$ become $A \to xCy$ and $B \to xCy$)

IIII) Shorten all rules to two : $A \to B_1B_2 \cdots B_k$ becomes $A \to B_1A_1$, $A_1 \to B_2A_2$, ... , $A_{k-2} \to B_{k-1}B_k$

IIIII) Replace ill-placed terminals "a" by T_a with $T_a \to a$.

Then, By taking step by step, we can get a CNF.

Step 1. Remove λ-rule.

$S \to baAB \mid baA \mid baB \mid ba$

$A \to bAB \mid bA \mid bB \mid b$

$B \to BAa \mid Ba \mid Aa \mid a \mid A$

Step 2. Remove unit productions.

we can find unit production `$B \to A$'.

So we can replace $A \to bAB \mid bA \mid bB \mid b$.

$B \to BAa \mid Ba \mid Aa \mid a \mid bAB \mid bA \mid bB \mid b$

Step3. Shorten all rules to two.

we can define new production rule for doing step3.

$X \to ba$

$Y \to AB$

$Z \to Aa$

Now rewrite the grammar using new three production rules.

$S \to XY \mid XZ \mid ba$

$A \to bY \mid bA \mid bB \mid b$

$B \to BZ \mid B\bar{a} \mid A\bar{a} \mid \bar{b}Y \mid \bar{b}A \mid \bar{b}B \mid \bar{a} \mid \bar{b}$

$X \to ba$

$Y \to AB$

$Z \to Aa$

Step 4. Replaced ill-placed terminals "a" by T_a with $T_a \to a$.

$T_a \to a$

$T_b \to b$

Then, we can get CNF For grammar.

$S \to XY \mid XZ \mid T_bT_a$

$A \to T_bY \mid T_bA \mid T_bB \mid b$

$B \to BZ \mid BT_a \mid AT_a \mid T_bY \mid T_bA \mid T_bB \mid a \mid b$

$X \to T_bT_a$

$Y \to AB$

$Z \to AT_a$

$T_a \to a$

$T_b \to b$

4. Using the CYK Algorithm to find parsing of the string aab using the given grammar.

$S \to AB,$

$A \to BB \mid a,$

$B \to AB \mid b.$

$\boxed{\text{Ans}}$ To use the CYK Algorithm to find parsing of the string, we need to define some definition.

1) Define substrings $W_{ij} = a_i \dots a_j$.

2) Define subsets $V_{ij} = \{A \in V : A \overset{*}{\Rightarrow} w_{ij}\}$.

And $V_{ij} = \bigcup_{k \in \{i, i+1, \dots, j-1\}} \{A : A \to BC, \text{ with } B \in V_{ik},\ C \in V_{k+1,j}\}$

Then, let's compute

$W_{11} = a \qquad W_{22} = a \qquad W_{33} = b$

$W_{12} = aa \qquad W_{23} = ab$

$W_{13} = aab$

$V_{11} = \{A\} \qquad V_{22} = \{A\} \qquad V_{33} = \{B\}$

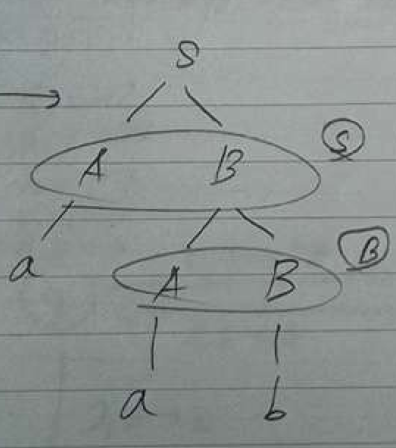
$V_{12} = \{A : A \to BC, \text{ with } B \in V_{11},\ C \in V_{22}\} = \text{empty} = \emptyset$

$V_{23} = \{A : A \to BC, \text{ with } B \in V_{22},\ C \in V_{33}\} = \{S, B\}$ ($\because$ set of the variable that has AB on right side.)

Then, Finally we can get V_{13}.

($V_{13} = \{S, B\} = \{A : A \to BC, \text{ with } B \in V_{11},\ C \in V_{22}\} \cup \{A : A \to BC, \text{ with } B \in V_{22},\ C \in V_{23}\}$
($\because k \in \{1, 2\}$))

Result $V_{13} = \{S, B\}$ $\longrightarrow$



5. Construct npda's that accept the following language on $\Sigma = \{a, b, c\}$.

$$L = \{w : n_a(w) + n_b(w) = n_c(w)\}$$

Ans npda is defined by

$$M = \{Q, \Sigma, \Gamma, \delta, q_0, Z, F\}$$

$Q = \{q_0, q_1, q_2, q_3\}$ → set of internal state

$\Sigma = \{a, b, c\}$ → Input alphabet

$\Gamma = \{X, Y, Z\}$ → stack alphabet // when push

$F = \{q_3\}$ → set of the accept state

and $\delta = Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to 2^{(Q \times \Gamma^*)}$ as follows:

$\delta(q_0, \lambda, \lambda) = \{(q_1, Z)\}$

$\delta(q_1, a, Z) = \{(q_1, XZ)\}$ // when a enters, push X in stack.

$\delta(q_1, a, X) = \{(q_1, XX)\}$

$\delta(q_1, b, Z) = \{(q_1, XZ)\}$ // when b enters, push X in stack

$\delta(q_1, b, X) = \{(q_1, XX)\}$

$\delta(q_1, c, X) = \{(q_1, \lambda)\}$ // when reading c, pop X

$\delta(q_1, c, Z) = \{(q_2, YZ)\}$ // when c comes first, without stack pushing, then push Y in stack and pop when reading a or b.

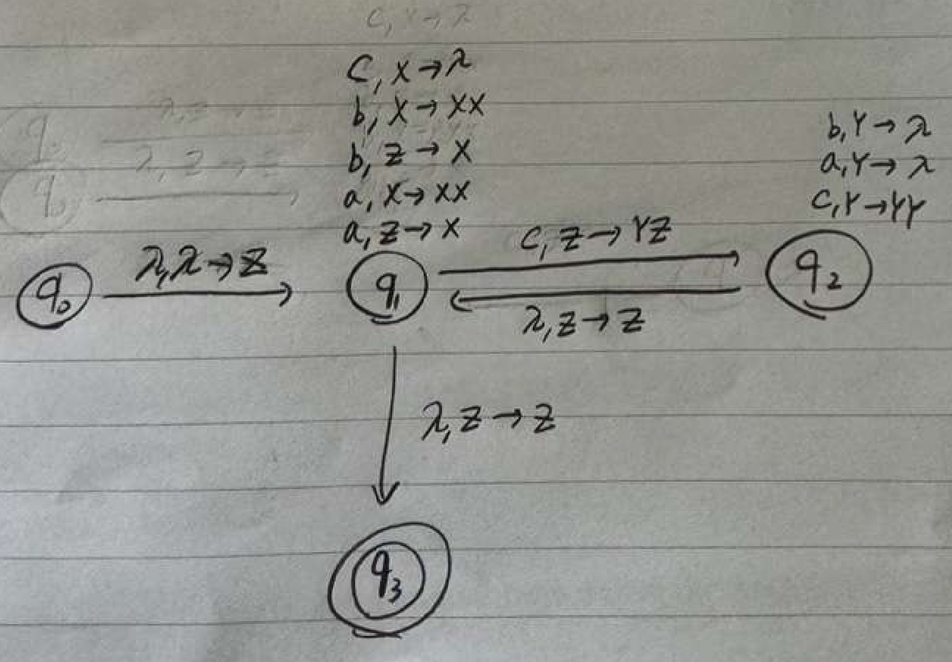
$\delta(q_2, c, Y) = \{(q_2, YY)\}$

$\delta(q_2, a, Y) = \{(q_2, \lambda)\}$

$\delta(q_2, b, Y) = \{(q_2, \lambda)\}$

$\delta(q_2, \lambda, Z) = \{(q_1, Z)\}$

$\delta(q_1, \lambda, Z) = \{(q_3, Z)\}$ // Final state



6. Find LL grammar for the following Languages, assuming $\Sigma = \{a, b, c\}$.

$$L = \{a^n b^n c^{n+m} : n \geq 1, m \geq 1\}.$$

[Ans] To find LL grammar for L,

we need to analyze L.

I) The string consists of sequences $\underbrace{aa \cdots a}_{n} \overbrace{b \cdots b}^{m} \cdots \underbrace{c \cdots c}_{n+m}$.

II) Because of $n \geq 1, m \geq 1$, there exists at least one a's and one b's in every valid string.

Then, we can define grammar (LL).

LL grammar must be recognized by LL parser.

$S \rightarrow aSc \mid aAc$

$A \rightarrow bAc \mid bc$