TOC Assignment ~ 3

Date.....................

Ques1 CFG Construction and Derivation Tree Analysis
Construct a context-free grammar G for the language $L = \{a^n b^m \mid n \geq 1, m \geq 0, \text{ and } n \text{ is even}\}$.

a) Define V, Σ, P, and S

- $\Sigma = \{a, b\}$
- $V = \{S, A\}$
- Start symbol : S
- Productions (P) :

$S \rightarrow aaS \mid aaA$
$A \rightarrow bA \mid \epsilon$

S generates an even number of a's (at least 2), and A generates any number of b's.

b) Show the leftmost and rightmost derivations for the string "aaaabb".

String : aaaabb
Leftmost Derivation :

$S \rightarrow aaS$
$\rightarrow aa\ aaA$
$\rightarrow aa\ aabA$
$\rightarrow aa\ aa\ b\ bA$
$\rightarrow aa\ aa\ b\ b$

Date.............

(c) Derivation Tree for the same string.

```
        S
      / | \
    aa  aa  A
            |
            b
            |
            b
```

Leaves: a a a a b b → "aaaabb"

(d) ~~Regularity~~ & ~~Context - Freeness~~ Prove that the language is not Regular but is context - free.

Regular expression for L:

$(aa)^+ \; b^*$

Since this RE exists, L is a Regular Language. And every regular language is also Context - Free

Ques2 Ambiguity Analysis of a Given Grammar

- Given the grammar : $S \to aSbS \mid \varepsilon$

a) Generate two different parse trees for the string "aabb".

→ Derive aabb

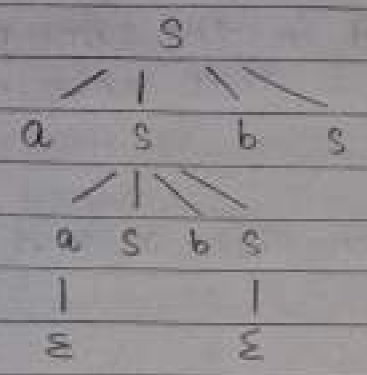
$S \to aSbS$

Spiral

→ a (a S b S) b S

→ a a ε b ε b ε = aabb

→ Parse tree



Leaves (left-to-right): a a b b → aabb

b) Prove that the grammar is ambiguous using formal definitions.

A grammar is ambiguous if there exists at least one string in the language that has two distinct parse trees (equivalently two different leftmost or rightmost derivations).

Claim: The grammar S → aSbS | ε is unambiguous.

Proof:

- Any nonempty string generated by this grammar must begin with a (because the only production introducing terminals start with a) and therefore must be

Date.....................

produced by one application of $S \rightarrow aS_1bS_2$.

- In any derivation of a nonempty string w, the first a in w must match some b later in w. Let that matching b be at position k. The grammar forces that matching b to be the b produced in the same production aS_1bS_2 that produced the first a. That uniquely splits w into three parts.
  - a (the first terminal)
  - substring generated by S_1 (between this a and it's matching b),
  - b (the matching b),
  - substring generated by S_2 (the remainder)

- The position k (first a's matching b) is uniquely determined by the usual balance argument (count a minus b scanning from left) — it is the smallest index where balance returns to the level before the first a. Hence the split into parts for S_1 and S_2 is unique.

- By induction on string length, the derivations (and therefore parse tree) of each of the two substrings are unique.

- Therefore the whole parse tree is unique.

So no string has two different parse trees is unambiguous.

(c) Construct a non-ambiguous grammar that generates the same language.

Date.....................

The given grammar itself is already non-ambiguous. You can present an equivalent (and maybe clearer) unambiguous grammar:

$S \rightarrow TS \mid \epsilon$
$T \rightarrow aSb$

T generates one matched pair a.....b where the .... is a balanced substring (generated by S), and then $S \rightarrow TS$ allows concatenation of such matched blocks. This grammar is the same as the original $S \rightarrow aSbS \mid \epsilon$ but written to emphasize the unique decomposition $S = (aSb)S$ — hence unambiguous.

Ques 3 PDA construction for Balanced Expressions
Construct a PDA that accepts the language $L = \{a^n b^n \mid n \geq 0\}$.

a) Define the PDA as a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$.

let $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ where:
- $Q = \{q_0, q_1, q_f\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{Z_0, X\}$ — Z_0 is initial stack symbol, X is marker for each a
- q_0 is start state
- Z_0 is initial stack symbol
- $F = \{q_f\}$ (accept by final state)

Date.....................

In q_0 push one X for each a. On seeing first b go to q_1 and pop one X per b. If input finished and stack has only Z_0, go to q_f.

b) Provide the complete transition table.

1) $\delta(q0, a, Z0) = \{(q0, XZ0)\}$
2) $\delta(q0, a, X) = \{(q0, XX)\}$
3) $\delta(q0, b, X) = \{(q1, \epsilon)\}$
4) $\delta(q1, b, X) = \{(q1, \epsilon)\}$
5) $\delta(q1, \epsilon, Z0) = \{(q_f, Z0)\}$
6) $\delta(q0, \epsilon, Z0) = \{(q_f, Z0)\}$

c) Show step-by step instantaneous descriptions (IDs) for the input "aaabbb".

Start: $(q_0, aaabbb, Z_0)$

1. $(q_0, aaabbb, Z_0)$
2. $\xRightarrow{a} (q_0, aabbb, XZ_0)$ // used $\delta(q0, a, Z0)$
3. $\xRightarrow{a} (q_0, abbb, XXZ_0)$ // $\delta(q0, a, X)$
4. $\xRightarrow{a} (q_0, bbb, XXXZ_0)$ // $\delta(q0, a, X)$

On reading first b, move to q1 and pop one X:

5. $\xRightarrow{b} (q_1, bb, XXZ_0)$ // $\delta(q0, b, X) \rightarrow (q1, \epsilon)$

Consume remaining remaining two b's in q1, popping Xs:

Date.....................

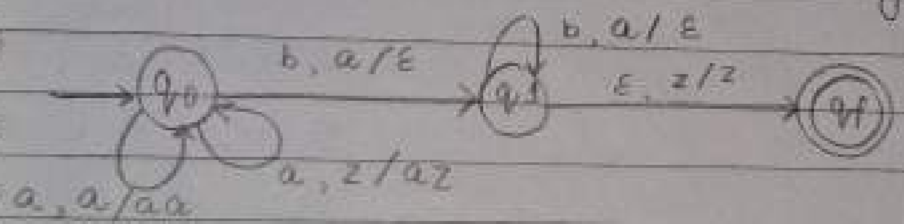
6. $\xrightarrow{b}$ (q_1, b, XZ_0) // $\delta(q_1, b, X)$
7. $\xrightarrow{b}$ (q_1, ε, Z_0) // $\delta(q_1, b, X)$

Now empty and stack has Z0 — take ε-move to final :

8. $\xrightarrow{\varepsilon}$ (q_f, ε, Z_0) // $\delta(q_1, \varepsilon, Z0)$

Since in final state q_f with empty input, string accepted.

d) Draw the state transition diagram.



Ques 4 Grammar Transformation and Normal forms

Given the CFG : $S \to aA \mid bB$; $A \to aA \mid \varepsilon$; $B \to bB / \varepsilon$

a) Eliminate ε-productions and unit productions

Step 1 – find nullable non terminals :

A is nullable (since $A \to \varepsilon$).

B is nullable (since $B \to \varepsilon$). S is not nullable.

Step 2 – add productions obtained by omitting nullable occurrences :

Spiral

Date.....................

- From $S \to aA$: because A nullable, add $S \to a$.
- From $S \to bB$: because B ~~a~~ nullable, add $S \to b$.
- From $A \to aA$: because A nullable, add $A \to a$.
- From $B \to bB$: because B nullable, add $B \to b$.

Step 3 — remove original ε-productions : remove $A \to \varepsilon$ and $B \to \varepsilon$.
No unit productions (of the form $X \to Y$) are present now.

Resulting grammar (no ε, no unit):

$S \to aA \mid a \mid bB \mid b$
$A \to aA \mid a$
$B \to bB \mid b$

b) Convert the resulting grammar into ~~Chom~~ Chomsky Normal Form (CNF).

CNF requires productions of the form $X \to YZ$ (two nonterminals) or $X \to a$ (single terminal). Also introduce helper nonterminals for terminals when they appear in longer right-hand sides.

Introduce terminal nonterminals :

$X\_a \to a$
$X\_b \to b$

Date......................

Replace terminals in mixed productions (length 2 where one symbol terminal):

From $S \rightarrow aA$ replace a by X_a : $S \rightarrow$ X_a A
From $S \rightarrow bB$ replace b by X_b : $S \rightarrow$ X_b B
From $A \rightarrow aA$ $\rightarrow$ $A \rightarrow$ X_a A
From $B \rightarrow bB$ $\rightarrow$ $B \rightarrow$ X_b B

Keep terminal-only productions:
$S \rightarrow a$ and $S \rightarrow b$ and $A \rightarrow a$ and $B \rightarrow b$ are of allowed form

$\Rightarrow$ Final CNF grammar:

$S \rightarrow$ X_a A
$S \rightarrow$ X_b B
$S \rightarrow a$
$S \rightarrow b$

$A \rightarrow$ X_a A
$A \rightarrow a$

$B \rightarrow$ X_b B
$B \rightarrow b$

X_a $\rightarrow a$
X_b $\rightarrow b$

c) Show derivation of the string "aab" in both the original and CNF forms.

Date.....................

Given grammar:

$S \rightarrow aA \mid bB$

$A \rightarrow aA \mid \epsilon$

$B \rightarrow bB \mid \epsilon$

This grammar generates strings of the form a^+ or b^+ only (like "a", "aa", "bbb" etc). Hence "aab" cannot be derived because it mixes both a's and b's — no rule allows switching from A(a's) to B(b's).

In both original & CNF, the same limitation exists, so "aab" is not derivable.

d) Discuss how CNF simplifies PDA simulation & parsing.

- Uniform structure: Every production is either $A \rightarrow BC$ or $A \rightarrow a$, making parsing steps systematic.

- Simpler PDA construction: Easier to simulate grammar as stack operations follow fixed patterns.

- Efficient algorithms: CNF allows algorithms like CYK to check string membership efficiently ($O(n^3)$).

- Less ambiguity: Reduces complexity and non determinism during parsing.

Ques^n CFL Membership via Pumping Lemma

Prove using the Pumping Lemma for CFLs that the

Date..................

Language $L = \{a^n b^n c^n \mid n \geq 0\}$ is not context-free.

a) State the pumping lemma clearly.

There is a pumping length $p \geq 1$ such that for every string $s \in L$ with $|s| \geq p$, we can write

$$s = uvwxy$$

satisfying

1) $|vwx| \leq p$,
2) $|vx| > 0$ (i.e. at least one of v, x is non empty),
3) for all $i \geq 0$, the string $uv^i wx^i y \in L$.

We will show this property fails for L, so L is not context-free.

b) Choose a string $s \in L$ where $|s| \geq p$ (pumping length).

Let p be the pumping length. Choose

$$s = a^p b^p c^p \in L,$$

so $|s| = 3p \geq p$

Assume $s = uvwxy$ satisfies the pumping-lemma conditions: $|vwx| \leq p$ and $|vx| > 0$.

c) Show some pumped string leaves L

Let $s = a^p b^p c^p$. Since $|vwx| \leq p$, the substring vwx lies fully inside one block (a's / b's / c's) or at most spans two.

Date..................

- If vwx is within one block, pumping adds/removes symbols from only that block → unequal a, b, c counts → not in L!

- If vwx spans two blocks, pumping disturbs their balance while the third block stays unchanged → counts mismatch → not in L.

Hence, for some $i \neq 1$, $uv^i wx^i y \notin L$.

d) Conclude why L cannot be accepted by any PDA.

Because the pumping lemma for CFLs is a necessary property of all context-free languages, and L violates it, L is not context-free. By the equivalence of context-free languages and pushdown automata, no PDA can accept L.

Ques6 Suppose, $L(G) = \{a^m b^n \mid m > 0 \text{ and } n \geq 0\}$. Find out the grammar G which produces L(G).

Given: $L(G) = \{a^m b^n \mid m > 0, n \geq 0\}$

To find: Context-Free Grammar G that generates this language.

Grammar:

Let $G = (V, \Sigma, P, S)$ where

- $V = \{S, T\}$
- $\Sigma = \{a, b\}$
- S is the start symbol

Date.....................

- Productions P:

$S \rightarrow aS \mid aT$
$T \rightarrow bT \mid \varepsilon$

⇒ Explanation:

- S ensures at least one 'a' is produced (since every derivation starts with a).
- T generates zero or more b's

Thus, all strings have one or more a's followed by any number of b's.

⇒ Eg. derivation:

$S \Rightarrow aS \Rightarrow aaT \Rightarrow aa\ bT \Rightarrow aa\ bbT \Rightarrow aa\ bbb$

Generated string: $aabbb \in L(G)$

Hence, the grammar correctly generates
$L(G) = \{a^m b^n \mid m > 0, n \geq 0\}$

Ques 7 Is this grammar ambiguous? If so, prove it and construct a non-ambiguous grammar that derives the same language.
$S \rightarrow aS \mid aSbS \mid c$

Yes, it is ambiguous. We show two different parse trees for this string.

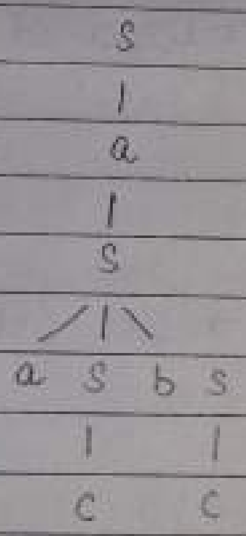
Date.....................

Two different parse trees for aacbc

Derivation A

1. S ⇒ aS
2. aS → a(aSbS)
3. a(aSbS) ⇒ a(acbS)
4. a(acbS) → a(acbc) = aacbc

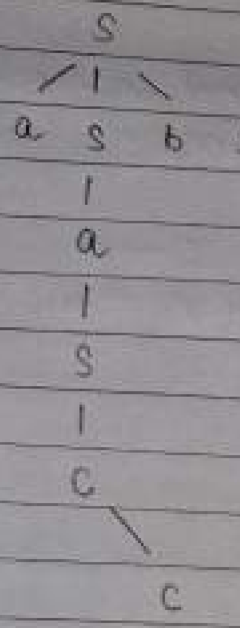
Parse tree A (shape):



Derivation B

1. S ⇒ aSbS
2. aSbS ⇒ a(aS)bS
3. a(aS)bS ⇒ a(ac) bS
4. a(ac) bS ⇒ a(ac)bc = aacbc

Parse tree B (different shape):



(rightmost $S \rightarrow c$)

→ The two trees are structurally different (in A the left child of the root is 'a' then a subtree aSbS; in B the root expands to aSbS immediately). Therefore the grammar is ambiguous (exists a string with two distinct parse trees).

→ A grammar is ambiguous iff some string has two different parse trees. We exhibited two distinct parse trees for aacbc. Hence the grammar is ambiguous.

Ques8 Give the CFG $G = (\{S, A, B\}, \{a\}, \{S \rightarrow A, A \rightarrow B, B \rightarrow a\}, S)$. Remove unit productions and rewrite the grammar.

Given CFG:

$G = (\{S, A, B\}, \{a\}, P, S)$ with

$P: S \rightarrow A, \; A \rightarrow B, \; B \rightarrow a$

Date.........

⇒ Remove unit productions :
We have unit chains $S \to A \to B \to a$ and $A \to B \to a$.
Replace them by direct terminal productions :

- From S follow chain to terminal a ⇒ add $S \to a$.
- From A follow chain to terminal a ⇒ add $A \to a$.
- Keep $B \to a$.

Remove unit rules $S \to A$ and $A \to B$

Resulting grammar (no unit productions):

$S \to a$
$A \to a$
$B \to a$

Note: All non-terminals still generate a. The language is {a}.

Ques: Give the CFG $G = (\{S, A, B\}, \{a, b, c\}, \{S \to A, A \to aB, B \to c\}, S)$. Remove useless productions and write the updated grammar.

Given :
$G = (\{S, A, B\}, \{a, b, c\}, P, S)$ with
$P : S \to A,\ A \to aB,\ B \to c$.

Step 1 – Find non terminals that generate terminals (useful for generating):

Date.....................

- B → C ⇒ B generates.
- A → aB and B generates ⇒ A generates
- S → A and A generates ⇒ S generates.

So all S, A, B are generating

Step 2 – Find reachable non terminals from start S:

- S is start (reachable).
- From S → A ⇒ A reachable
- From A → aB ⇒ B reachable

So, all S, A, B are reachable.

⇒ There are no useless productions (no unreachable or non-generating non-terminals).

* terminal grammar (after removing useless productions):

S → A
A → aB
B → c

Language produced : Strings of the form ac (specifically only "ac").

Ques 10 Convert the given CFG to CNF. Consider the given grammar G1:
S → a | aA | B

Date......................

$A \to aBB \mid \varepsilon$
$B \to Aa \mid b$

Step 1 – Remove ε- production :
$A \to \varepsilon$ is nullable ⇒ update others.

$S \to a \mid aA \mid B$
$A \to aBB$
$B \to Aa \mid a \mid b$

Step 2 – Remove unit productions :
$S \to B$ ⇒ replace with B's RHS.

$S \to a \mid b \mid aA \mid Aa$
$A \to aBB$
$B \to Aa \mid a \mid b$

Step 3 – Convert to CNF form :
Introduce $X\_a \to a$, $Y \to BB$.
Replace terminals in long RHS and make binary :

$S \to a \mid b \mid X\_a A \mid A X\_a$
$A \to X\_a Y$
$Y \to B B$
$B \to A X\_a \mid a \mid b$
$X\_a \to a$

Final CNF : All rules are either $A \to BC$ or $A \to a$.