Unit-II

Content free languages

Content free languages: Content free grammars - Parsing and ambiguity - content free grammars & Prog lang.

Simplification & Normal forms: Methods for transforming grammars - chomsky & Greibach Normal forms - membership algo: for CFGs

Push down Automata: Non-deterministic PDA - PDA and CFL - Deterministic PDA & deterministic CFL - Grammars for deterministic CFL.

Content Free Grammars (CFGs)

Regular lang - applicable to simple pattern
↓
left side single variable
& right side special form.

beyond complicated pattern are there.
↓
expressed by CFL.
↓
left side single variable
right side any no: of terminals variables.

Definition:

$G = (V, T, P, S)$

$$A \to x$$

$A \in V$, $x \in (V \cup T)^*$

Lang $L = L(G)$ Every regular grammar is CF.

	CFG	RG.
1.	Powerful than RE, & more expressive power	less powerful than CFG.
2.	Generate pattern of strings	Generate lexical construct like keywords, constants etc.
3.	Specify the recursive structure of the prog. lang.	can't specify recursive structure
4.	Specify regular & irregular lang.	defines regular lang.

Eg. Write CFG for $L(G) = \{ww^R : w \in \{a,b\}^*\}$

$S \to aSa$
$S \to bSb$
$S \to \lambda$

$w = abba$

$S \to aSa$
$\to abSba$
$\to abba$ ($S \to \lambda$)

Relation between regular & CFG:

$S \to abB$
$A \to aaBb$
$B \to bbAa$
$A \to \lambda$

} linear

$S \to aSa$
$S \to bSb$
$S \to \lambda$

} linear

Both examples are not only CFG, but linear also.

* Regular & linear grammars are CFG, but a CFG is not necessarily linear.

1. Write CFG for $L = \{a^n b^m : n \neq m\}$

(i) $n > m$ eg aab

$S \to AS_1$

$S_1 \to aS_1b \mid \lambda$

$A \to aA \mid a$

(ii) $n < m$ eg. abb

$S \to S_1B$

$S_1 \to aS_1b \mid \lambda$

$B \to bB \mid b$

$S \to AS_1 \mid S_1B$

$S_1 \to aS_1b \mid \lambda$ — equal no of a's & b's

$A \to aA \mid a$ $n > m$

$B \to bB \mid b$ $n < m$

eg. $w = aaabb$

$S \to AS_1$

$\to aAbS_1$

$\to aaAbbS_1$

$\to aaabb$ ✗

$S \to AS_1$

$\to aAS_1$

$\to aaS_1$

$\to aaaS_1b$

$\to aaaabb$ ✗

$S \to AS_1$

$\to aS_1$

$\to aaS_1b$

$\to aaaS_1bb$

$\to aaabb$ ✓

2. Find CFG for the following lang.

$L = \{a^n b^n, n \text{ is odd}\}$

$n = 1, 3, 5, 7 \ldots$

$w = a, aabb, aaabbb \ldots$

$S \to aAb$

$A \to aaAbb \mid \lambda$

$w = aaabbb$

$S \to aAb$

$\to aaaAbbb$

$\to aaabbb$

3. Find CFG for the following lang.

$L = \{a^n b^m : n \leq m+3\}$

$S \to aSb \mid B \mid A \mid \lambda$

$A \to aB \mid aaB \mid aaaB$

$B \to b \mid bB \mid \lambda$

$n=0 \quad m=0$
$S \to \lambda \quad 0<3$

$n=1 \quad m=0$
$S \to A \quad 1<3$

$n=1 \quad m=1$
$S \to B \quad 1<4$

$n=3, \quad m=2 \qquad 3<5$

$w = aaabb$

$S \to aSb$
$\to aAb$
$\to aaaBb$
$\to aaabBb$
$\to aaabb$ ($B \to \lambda$)

1. Find CFG for the following lang. $n \geq 0, m \geq 0, k \geq 0$

$L = \{a^n b^m c^k : k = n+2m\}$

$n=0, m=0, k=0$
$n=1, m=0, k=1$
$n=1, m=1, k=3$
$n=2, m=2, k=6$
$n=0, m=1 \quad k=2$

$S \to aSc \mid B \mid \lambda$

$B \to bBcc \mid \lambda$

$w = aabbccccccc$

$S \to aSc$
$\to aaScc$
$\to aabBccccc$
$\to aabbBccccccc$ ($B \to \lambda$)
$\to aabbccccccc$

$L = \{a^n b^m c^k : k = n+m\}$

$S \to aSc \mid bSc \mid \lambda$

$w = abcc$

$S \to aSc$
$\to abSc$ ($S \to \lambda$)

5. Find a CFG for the following lang with $n \geq 0, m \geq 0$

$$L = \{ w \in \{a, b\}^* : n_a(w) \neq n_b(w) \}$$

(i) $n_a > n_b$

$S \to S_1$

$S_1 \to S_1 S_1 \mid a S_1 b \mid b S_1 a \mid A$

$A \to aA \mid a \mid \lambda$

(ii) $n_b > n_a$

$S \to S_2$

$S_2 \to S_2 S_2 \mid a S_2 b \mid b S_2 a \mid B$

$B \to bB \mid b \mid \lambda$

$S \to S_1 \mid S_2 \mid \lambda$

$S_1 \to S_1 S_1 \mid a S_1 b \mid b S_1 a \mid A$

$A \to aA \mid a \mid \lambda$

$S_2 \to S_2 S_2 \mid a S_2 b \mid b S_2 a \mid B$

$B \to bB \mid b \mid \lambda$

$w = aab$

$S \to S_1$
$\to a S_1 b$
$\to a A b$
$\to a a A b \to aab$
λ

6. what lang. does the grammar

$S \to aSB \mid bSA$

$S \to \lambda$

$A \to a$

$B \to b$ generate?

$L = \{ w : w = aUb$ or $w = bUa$ with $w \in L, \lambda \in L \}$

7. Consider the grammar with productions.

$S \to aaB$

$A \to bBb \mid \lambda$

$B \to Aa$

Show that the string aabbabba is not in the lang generated by this grammar.

$S \to aaB$

$\to aaAa$

$\to aabBba$

$\to aabbAabba$

$\to aabbabba$

$S \to aaB$

$aaAa$

$aabBba$

$aabbBbAabba$

$aabbAabba$

λ

aabbabba is not in the grammar.

$S \to aaB$

$\to aaAa$

$\to aabBba$

$\to aabAaba$

$\to aabbBbaba$

$\to aabbAbaba$

$\to aabbbabaa$

aabbabaa.

aabbababa

Parsing & Ambiguity

Parsing - finding a sequence of productions by which $w \in L(G)$.

Parsing & membership.

$w \in L(G)$ - systematically construct all possible derivation.

→ consider $S \to x$ finding all x derived from S.

↓

none of it not matched to w

go to next round.

↓

apply all applicable productions to the left most variable of every x.

↓

some of it leads to w. → it takes a sentential form.

↓

finite length LMD or RMD.

↓

Top down parsing { This is called as exhaustive parsing / Brute force parsing.

↓

not used effectively

if it contains λ productions unable to find where derivation stop &

without λ - difficult to complete.

Eg

$S \to SS$

$S \to aSb$

$S \to bSa$

$S \to \lambda$

unable to predict when the derivation stopped.

$\Rightarrow$

$S \to SS$

$S \to aSb$

$S \to bSa$

$S \to ab \mid ba$

it takes $|w|$ length derivation. within $|w|$ able to predict whether $w \in L$ or $w \notin L$.

LMD & RMD

Left most derivation & right most derivation

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

$S \to AB \mid \lambda$

$A \to aaA$

$A \to \lambda$

$B \to Bb$

$B \to \lambda$

LMD

$S \Rightarrow AB$

$\underset{lm}{\Rightarrow} aaAB$

$\underset{lm}{\Rightarrow} aa\lambda B$

$\underset{lm}{\Rightarrow} aaBb$

$\underset{lm}{\Rightarrow} aab$

RMD

$S \overset{rm}{\Rightarrow} AB$

$\overset{rm}{\Rightarrow} Abb$

$\overset{rm}{\Rightarrow} Ab$

$\overset{rm}{\Rightarrow} aaAb$

$\overset{rm}{\Rightarrow} aab$

* LMD & RMD
  - Yield same sentence by using same productions.
  - difference is the order in which the productions are applied.
  - remove the irrelevant factors (ie) variables are replaced in a specific order.

Definition:

LMD – Each step the left most variable in the sentential form is replaced.

RMD – Each step the rightmost variable in the sentential form is replaced.

Derivation tree:

* It is an ordered tree.
* nodes are labeled with the left side of the production & children of the node represent the right side of the productions.

eg. $S \to aAB$

$A \to bBb$

$B \to a \mid \lambda$

LMD

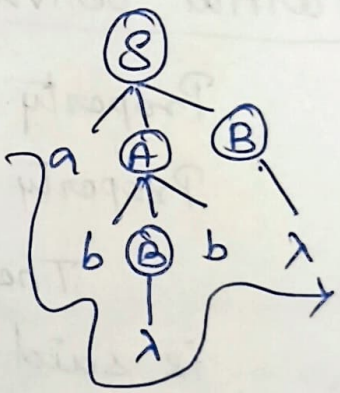
$S \to aAB$

$\to abBbB$

$\to ab\lambda bB$

$\to abbB$

$\to abb$



RMD

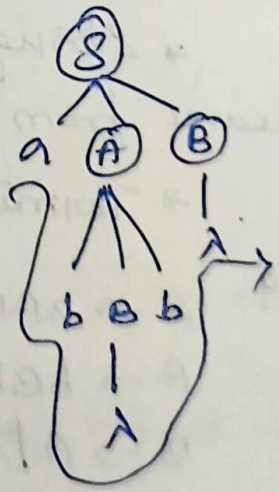
$S \to aAB$

$\to aA\lambda$

$\to abBb$

$\to ab\lambda b$

$\to abb$



Definition : Derivation Tree : CFG

Properties of derivation tree.

1. Root labeled with start symbol S.
2. Every leaf label from $T \cup \{\lambda\}$
3. Every internal nodes label from V
4. $A \rightarrow a_1, a_2 \cdots a_n$

productions from left to right.

5. Leaf Labeled with λ – no other children.

Partial Derivation Tree:

Property 1, 3, 4 & 5 hold.

Property 2 is changed as

The leaf contains $V \cup T \cup \{\lambda\}$:

is said to be a partial derivation tree.

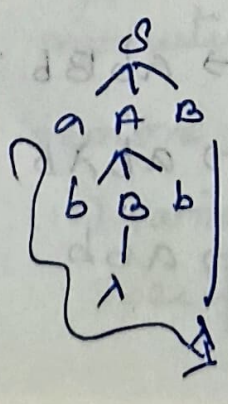
Yield of the tree:

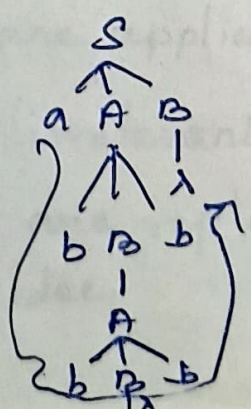
* String of symbols obtained by reading the leaves from left to right.
* Terminals read in the order of DFS order.

eg. $S \rightarrow aAB$

$A \rightarrow bBb$

$B \rightarrow A/\lambda$





yield
$w = abb$.

$w = abbbb$

Theorem:

Let $G=(V,T,P,S)$ be a CFG. Then for every $w \in L(G)$, there exist a derivation tree of G whose yield is w. The yield of any derivation tree is in $L(G)$. Also if t_G is any partial derivation tree for G whose tree root is labeled S, then the yield of t_G is a sentential form of G.

Proof:

* Assume that for every sentential form derivable in 'n' steps, there is a corresponding partial derivation tree.
* Now any w derivable in n+1 steps

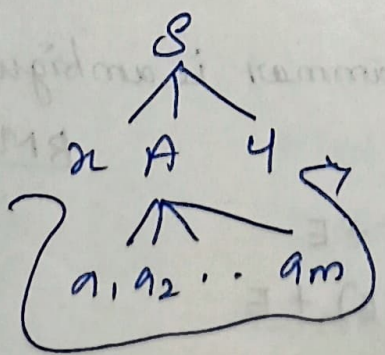
$S \Rightarrow xAy$ $\quad x, y \in (V \cup T)^*$, $A \in V$

in 'n' steps

$xAy \Rightarrow xa_1 a_2 \ldots a_m$

Yield by partial derivation tree with production $(A \to a_1 a_2 \ldots a_m)$

* Similarly every partial tree derivation represent some sentential form.



yield $w = xa_1 a_2 \ldots a_m y$ this result is true for all sentential form.

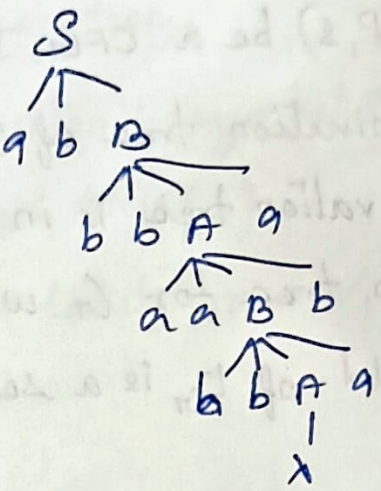
Give a derivation tree for $w = abbbaabbaba$

$S \to abB$

$A \to aaBb$

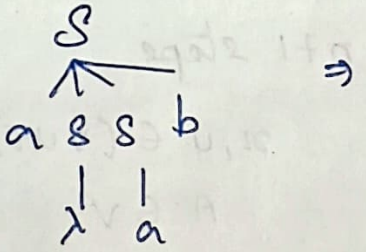
$B \to bbAa$

$A \to \lambda$



Consider the derivation tree

find a grammar & find a two sentences of $L(G)$.



$\Rightarrow$ $S \to aSSb$

$S \to a \mid \lambda$

$w = aab, ab, aaab \ldots$

Ambiguous:

A grammar that produces more than one parse tree for any input sentence is said to be an ambiguous grammar.

Eg. Prove the given grammar is ambiguous. $(id * id) + id$

$E \to E*E$

$E \to E+E$

$E \to (E)$

$E \to id$

LMD1

$E \to E+E$

$\Rightarrow (E) + E$

$\Rightarrow (E*E) + E$

$\Rightarrow (id * E) + E$

$\Rightarrow (id * id) + E$

$\Rightarrow (id * id) + id$

RMD1

$E \to E + E$

$\to (E) + E$

$\to (E*E) + E$

$\to (E*E) + id$

$\to (E * id) + id$

$\to (id * id) + id$

id + id * id

LMD1

$E \rightarrow E + E$

$\rightarrow E + E * E$

$\rightarrow id + E * E$

$\rightarrow id + id * E$

$\rightarrow id + id * id$

LMD2

$E \rightarrow E * E$

$\rightarrow E + E * E$

$\rightarrow id + E * E$

$\rightarrow id + id * E$

$\rightarrow id + id * id$

Example 2:

$S \rightarrow PC \mid AQ$

$P \rightarrow aPb \mid \lambda$

$C \rightarrow cC \mid \lambda$

$Q \rightarrow bQc \mid \lambda$

$A \rightarrow aA \mid \lambda$

is ambiguous or not.

w = abc.

$S \rightarrow PC$

$\rightarrow aPbc$

$\rightarrow abcC$

$\rightarrow abc$

$S \rightarrow AQ$

$\rightarrow aAQ$

$\rightarrow abQc$

$\rightarrow abc$

∴ it is ambiguous grammar.

Theorem:

Suppose that $G = (V, T, P, S)$ is a CFG that does not have any rules of the form $A \to \lambda$ or $A \to B$. where $A, B \in V$. By exhaustive search method tell $w \in \Sigma^*$ either producing a w or no parsing is possible.

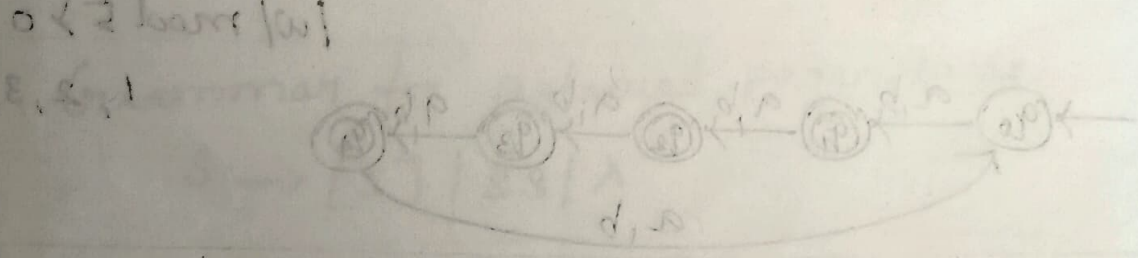
Proof:

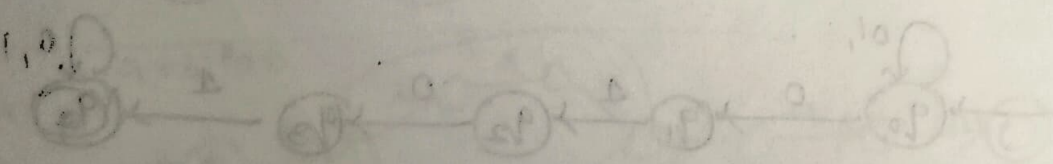
* For each sentential form, consider both its length & the number of terminal symbols.
* Each step in the derivation increase at least the length or terminals symbol.
* Neither a length of a sentential form nor the no: of terminals symbols can exceed $|w|$, a derivation can't involve more than $2|w|$ rounds, at which time either have a successful parsing or w can't be generated by the grammar.
* In a exhaustive search, the LMD have
  no more that $|p|$ sentential form after 1st round
  " " " $|p|^2$ " " " " 2nd "
  and so on.
  :
  till $2|w|$ rounds.

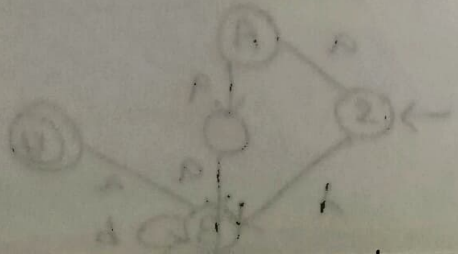
∴ The total no: of sentential forms cannot exceed.

$$M = |P| + |P|^2 + \cdots + |P|^{2|w|} = O(p^{2|w|+1})$$

The exhaustive search may grow exponentially with the length of the string.







Example:

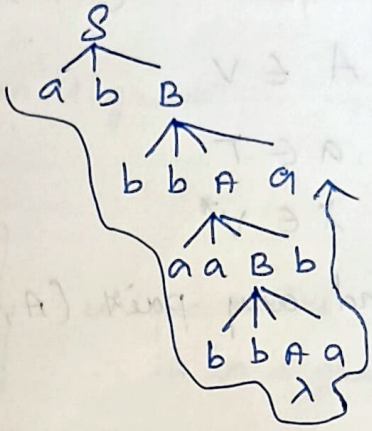
Give a derivation tree for $w = abbbaabbaba$.

$S \to abB$

$A \to aaBb$

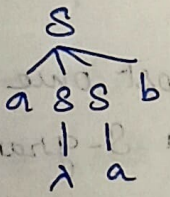
$B \to bbAa$

$A \to \lambda$



---

2. Consider the derivation tree



find a grammar & two sentences of a L(G).

$S \to aSSb$

$S \to a \mid \lambda$

$w = aab, ab, aaab \ldots$

---

3. check the grammar is ambiguous or not?

$S \to PC \mid AQ$

$P \to aPb \mid \lambda$

$C \to cC \mid \lambda$

$Q \to bQc \mid \lambda$

$A \to aA \mid \lambda$

$S \to PC$
$\Rightarrow aPbC$
$\Rightarrow abC$
$\Rightarrow abcC$
$\Rightarrow abc$

$S \to AQ$
$\Rightarrow aAQ$
$\Rightarrow aQ$
$\Rightarrow abQc$
$\Rightarrow abc.$

The given grammar is ambiguous.

Simple Grammar. (S-Grammar)

$A \to aX$ $\quad (A, a)$

where $A \in V$

$a \in T$

$X \in V^*$

and any pair (A, a) occurs at most once in P.

Example.

$S \to aS \mid bSS \mid c$

$S \to aS \quad (S, a)$
$S \to bSS \quad (S, b)$
$S \to c \quad (S, c)$

} occurs at most once ∴ it's a S-grammar.

2. $S \to aS \mid bSS \mid aSS \mid c$

$S \to aS \quad (S, a)$
$S \to bSS \quad (S, b)$
$S \to aSS \quad (S, a)$
$S \to c \quad (S, c)$

— occurs twice ∴ it's not a S-grammar.

* If G is S-grammar then any string $w \in L(G)$ can take $|w|$ derivations only.

$S \Rightarrow A_1 A_2 \cdots A_m$ $\quad A_1 \to a_1 \to$ at most one choice

$a_1 A_2 \cdots A_m$ $\quad A_2 \to a_2 \to$ "

$a_1 a_2 \cdots A_m$

$\vdots$

$a_n a_2 \cdots a_m$ $\Rightarrow$ takes $|w|$ steps to complete the process in S-grammar

Example:

Show that

$S \to aS \mid bS \mid cA$

$A \to aA \mid bS$ is an s-grammar.

Solution.

(S, a) (A, a)
(S, b) (A, b) } - unique pairs
(S, c)

∴ it is an s-grammar.

---

2. Find an s-grammar for

$L = \{a^n b^n : n \geq 2\}$

$S \to aS_1$ (S, a)

$S_1 \to aAB$ (S_1, a)

$A \to aAB \mid b$ (A, a) (A, b)

$B \to b$ (B, b)

n = 2
w = aabb

n = 3 w = aaabbb

$S \to aS_1$
$\to aaAB$
$\to aaaABB$
$\to aaabBB$
$\to aaabbB$
$\to aaabbb$

n = aabb

---

3. Find s-grammar for $L = \{a^n b^{n+1} : n \geq 2\}$

$w = aa(ab)^* bbb$

n = 2
aa(ab)bbb
aaa(ab)bbb

w = aabbb

$S \to aA_1$ (S, a)

$A_1 \to aA_2$ (A_1, a)

$A_2 \to aA_2B_1 \mid bB_2$ (A_2, a) (A_2, b)

$B_1 \to b$ (B_1, b)

$B_2 \to bB_2 \mid$ (B_2, b)

$B_3 \to b$ (B_3, b)

$S \Rightarrow aA_1$
$\Rightarrow aaA_2$
$\Rightarrow aabB_2$
$\Rightarrow aabbB_3$
$\Rightarrow aabbb$.

s-grammar
$L = \{a^n b^{n+1} \mid n \geq 1\}$

$S \to aXb$
$X \to aXb$
$X \to b$

4. Show that every s-grammar is unambiguous?

For every $w \in L(G)$ with G an s-grammar it is obvious that there is a unique choice for the leftmost derivation. So every s-grammar is unambiguous.

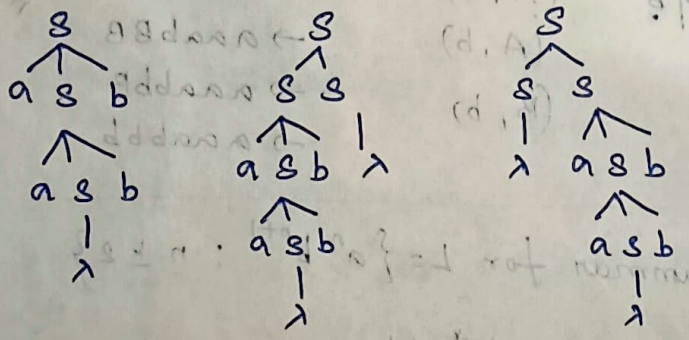
Ambiguity in grammar & Languages.

Definition:

A CFG G is said to be ambiguous if there exists some $w \in L(G)$ has at least two distinct (LMD or RMD) derivation trees.

eg.

$S \to aSb \mid SS \mid \lambda$



* In NLP - ambiguity is a common factor but in prog: lang: only one interpretation of each str: is required.

So removed ambiguity when it possible.

↳ write an equivalent unambiguous grammar.

↓

write prog: rules based on precedence rules.

Eg.

Generate the string a+b*c

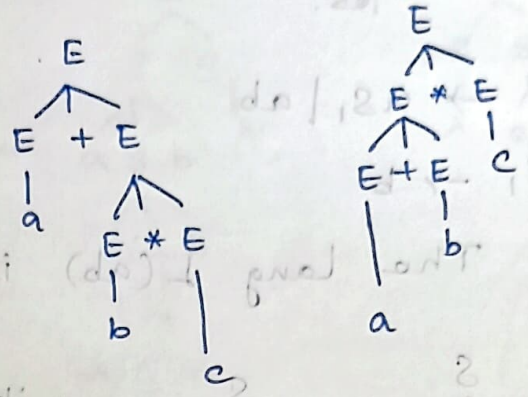
E → I

E → E+E

E → E*E

E → (E)

I → a|b|c



Removing ambiguity. – Introducing of new variables.

E → T

T → F

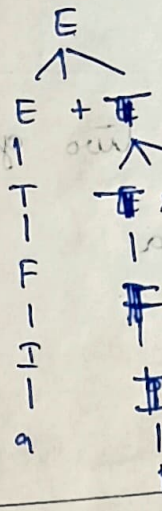
F → I

E → E+T

T → T*F

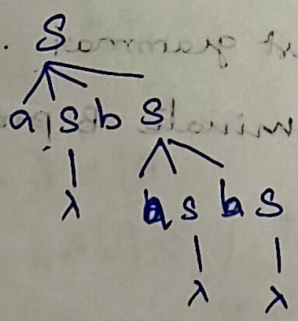
F → (E)

I → a|b|c



Show that the following grammar is ambiguous?

S → aSbS | bSaS | λ



S → aSbS

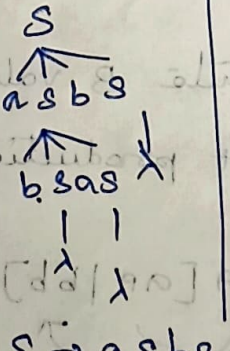
⇒ abS

⇒ abaSbS

⇒ ababS

⇒ ababλ

⇒ abab



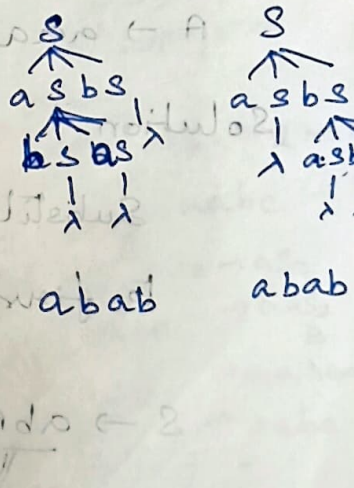
S ⇒ aSbS

⇒ abSaSbS

⇒ abaSbS

⇒ ababS

⇒ abab



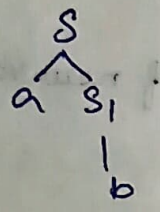
Is it possible for a regular grammar to be ambiguous?

Yes.

$S \to aS_1 \mid ab$

$S_1 \to b$

The lang $L(ab)$ is finite. So is regular



⇒ it's two distinct derivation

∴ it's ambiguous.

Show that the two grammars are equivalent.

$S \to abAB \mid ba$

$S \to aaa$

$B \to aA \mid bb$

and

$S \to abAaA \mid abAbb \mid ba$

$A \to aaa$

Solution.

Substitute B value in first grammar.

to first production & eliminate B production

$S \to abA[aA \mid bb] \mid ba.$

$S \to abAaA \mid abAbb \mid ba$

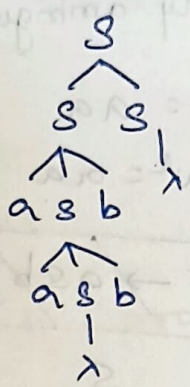
$S \to aaa$

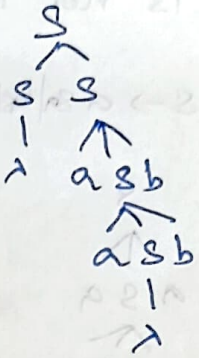
⇒ Now both grammars are equivalent.

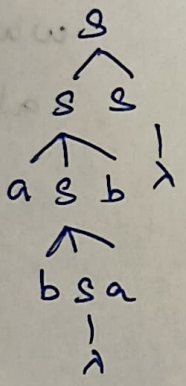
Show that the grammar is ambiguous.

$S \to SS \mid \lambda$

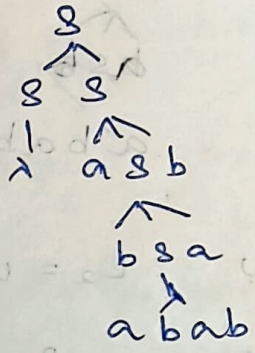
$S \to asb \mid bsa$







$\Rightarrow abab$



Inherantly Ambiguous.

→ Every Lang: that generates L is ambiguous Then the Lang: is called as inherantly ambiguous.

$L \Rightarrow a^n b^n c^m$ | $a^n b^m c^m$

↓ inherantly ambiguous

Eg. $L = \{a^n b^n c^m\} \cup \{a^n b^m c^m\}$

$L = L_1 \cup L_2$.

L_1

$S_1 \to S_1 c \mid A$

$A \to aAb \mid \lambda$

L_2

$S_2 \to aS_2 \mid B$

$B \to bBc \mid \lambda$

$S \to S_1 \mid S_2$

$S_1 \to S_1 c \mid A$

$A \to aAb \mid \lambda$

$S_2 \to aS_2 \mid B$

$B \to bBc \mid \lambda$

$L_1 \Rightarrow n = 1 \quad m = 2$

$abcc$

$S_1 \to S_1 c$

$\to S_1 cc$

$\to Acc$

$\to abcc$

$L = a^n b^n c^n$

$n = 1 \quad abc$

$S_1 \to S_1 c$

$\to Ac$

$\to aAbc$

$\to abc$

$L_2 = n = 2 \; m = 1$

$aabc$

$S \to S_2$

$\to aB$

$\to abBc$

$S_2 \to aS_2$

$\to aaS_2 B$

$\to aabBc$

$\Rightarrow aabc$

$abc \Rightarrow$ ambiguous but called as inherantly ambiguous

$S_2 \to aS_2$

$\to aB$

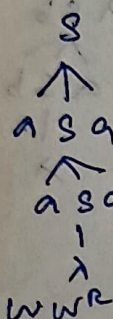
$\to abBc$

$\to abc$

Show that the Lang: $L = \{ w w^R : w \in \{a,b\}^* \}$ is not inherently ambiguous (Prove that the lang: is unambiguous)

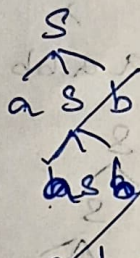
$S \to asa | bsb | \lambda$ $\quad w = aa$ $\quad w = bb$

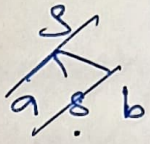
$w^R = aa$ $\quad w^R = bb$.

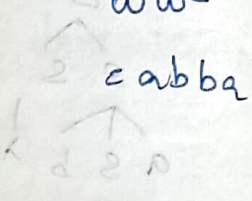


~~$S \to asb | bsa | \lambda$~~ $\quad w = ab$

$w^R = ba$







$L_1 = w$ $\quad L_2 = w^R$.

$L = w w^R$.

Show that the Lang: $L = \{ w w^R : w \in \{a,b\}^* \}$ is not inherently ambiguous

(or) Prove that the Lang: is unambiguous.

Solution

$S \to asa | bsb | \lambda$

$S \to asa$

$\to aasaa$

$\to \lambda$

$\Rightarrow aa$

$w \quad w^R$.

$S \to bsb$

$\to bbsbb$

$\Rightarrow bbbb$

$w \quad w^R$.

$S \to asa$

$\to abbba$

$\to abba$

$w \quad w^R$.

$S \to bsb$

$\to basab$

$\to baabab$

only one string derivation is possible.

Hence, the grammar is unambiguous.

Proof.

Basis: $|w| = \lambda \Rightarrow 0$

Induction: $|w| = 2 \cdot k$ produce an unambiguous grammar

$k=1$

$2 \cdot 1 \Rightarrow 2$

aa aa
w w^R.

Take $|w|$ = even → produce a unique LMD

That is G is unambiguous grammar & L is Language.

Context free Grammars & Programming languages.

* Formal Langs: used for compiler & interpreter construction.
* Regular & CFL — helps to write efficient interpreters.
* Regular — recognize simple patterns
  CFL — "compler"
* Ambiguous — leads to different results
  ↓ when a prog: run by different m/c or compilers.
  ↓ this issue avoided by unambiguous grammar.

whilestat → while (exp) (st)
↓ variable ↓ terminal ↓ variables