Unit-3

→ Type-2 or Context free Grammer (CFG) –

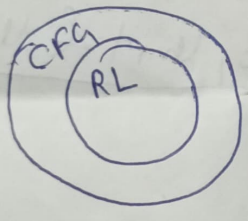
$A \to \alpha$

where $A \in V$

$\alpha \in (V \cup T)^*$

It is called context-free because whenver you see A, you did not worry about context it is appearing on, you can directly replace it with any of right.

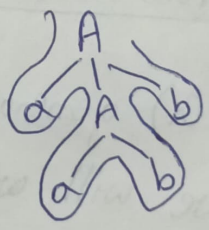
→ All Regular Grammer are CFG but not vice-versa.



→ $A \to aAb \mid ab$

$A \Rightarrow aAb$

$aabb$



→ Derivation Tree

→ Class of language → CFL generated by CFG

M/C that accept CFL → PDA (push down Automata)

CFG –

$w \in L(G)$ Does word belong to language?

- we use CYK algorithm but to use CYK algorithm, we first need to eliminate ϵ, unit & useless symbol.
- Time = $O(|w|)^3$

→ Elimination of ϵ, unit & useless –

1) ϵ- Production –

- If $L(G)$ generate ϵ, we cannot eliminate all, we will have atleast one production $S \to \epsilon$.
- If $L(G)$ does not contain ϵ, we can eliminate all ϵ-productions.

Method -

1) A ⇒ ∈
   A ⇒ ( ) ⇒ ( ) ⇒ ∈

Nullable variable → If a variable can directly generate ∈ or after few derivation can generate ∈.

2) Go to RHS of every production, write it with {A} and without {A}, whereever nullable variable present

3) Eliminate A → ∈

Note - Whenever start symbol produces ∈, it means language contain ∈ so we cannot eliminate all ∈-productions.

Ex1

S → aSb | aAb
A → ∈

Nullable variable = {A}

S → aSb | ~~aAb~~ | ab
~~A → ∈~~

S → aSb | ab

Q

$S \to AB$

$A \to aAA \mid \epsilon$

$B \to bBB \mid \epsilon$

Nullable = $\{A, B, S\}$

$S \to AB \mid B \mid A \mid \epsilon$

$A \to aAA \mid aA \mid a \mid \not\epsilon$

$B \to bBB \mid bB \mid b \mid \not\epsilon$

$\Downarrow$

$S \to AB \mid B \mid A \mid \epsilon$

$A \to aAA \mid aA \mid a$

$B \to bBB \mid bB \mid b$

Q

$S \to AbaC$

$A \to BC$

$B \to b \mid \epsilon$

$C \to D \mid \epsilon$

$D \to d$

1. $\{B, C, A\}$

2. $S \to AbaC \mid baC \mid Aba \mid ba$

   $A \to BC \mid B \mid C$

   $B \to b \mid \not\epsilon$

   $C \to D \mid \not\epsilon$

   $D \to d$

Ans

2) Unit - production -

$A \to B$
$B \to C$
$C \to d$

A
|
B
|
C
|
d

(neither length nor terminal is increasing)

Q $S \to Aa \mid B$
$B \to A \mid bb$
$A \to a \mid bc \mid B$

$S \to B \to A \to a$, bc, bb

$B \to A \to a$, bc

$A \to B \to bb$

$S \to Aa \mid bb \mid a \mid bc$
$B \to bb \mid a \mid bc$
$A \to a \mid bc \mid bb$

Q $S \to AB$
$A \to a$
$B \to C \mid b \to b \mid a$
$C \to D \to a$
$D \to E \to a$
$E \to a$

$B \to C \to D \to E \to a$

⇓

$S \to AB$
$A \to a$
$B \to a \mid b$
$C \to a$
$D \to a$
$E \to a$

3) Eliminate useless production

- not reachable from start symbol.

- $A \overset{*}{\Rightarrow} w$

↓

A is useful if derive some terminal or part of terminal and reachable from start symbol.

Q $S \to AB \mid a$

$A \to BC \mid b$

$B \to aB \mid C$

$C \to aC \mid B$

useful → {a, b, S, A}

① terminal always useful.

② if RHS is terminal, add LHS as useful variable.

③ if RHS are mode of useful only, then add LHS also,

④ Delete all productions having non-useful on LHS as well as RHS.

~~$S \to AB \mid a$~~

~~$A \to BC \mid b$~~

~~$B \to aB \mid C$~~

~~$C \to aC \mid B$~~

$S \to a$

$A \to b$

⑤ Reachability from S to A

so ~~$A \to b$~~

$\boxed{S \to a}$ final

Q

$S \to AB \mid AC$

$A \to aAb \mid bAa \mid a$

$B \to bbA \mid aaB \mid AB$

$C \to abCA \mid aDb$

$D \to bD \mid aC$

useful $= \{a, b, A, B, S\}$

$S \to AB \mid \cancel{AC}$

$A \to aAb \mid bAa \mid a$

$B \to bbA \mid aaB \mid AB$

~~$C \to abCA \mid aDb$~~

~~$D \to bD \mid aC$~~

Q

$S \to ABC \mid BaB$

$A \to aA \mid BaC \mid aaa$

$B \to bBb \mid a$

$C \to CA \mid AC$

$\{a, b, B, A, S\}$

$S \to \cancel{ABC} \mid BaB$

$A \to aA \mid \cancel{BaC} \mid aaa$

$B \to bBb \mid a$

~~$C \to CA \mid AC$~~

$\Rightarrow$

$S \to BaB$

~~$A \to aA \mid aaa$~~

$B \to bBb \mid a$

$\Downarrow$

$S \to BaB$

$B \to bBb \mid a$

Ambiguity –

G (V, T, P, S)

E → E+E | E*E | id

V = {E}

T = {+, *, id}

id + id * id

LMD (Leftmost Derivation) –

E → E+E
→ id+E
→ id+E*E
→ id+id*E
→ id+id*id

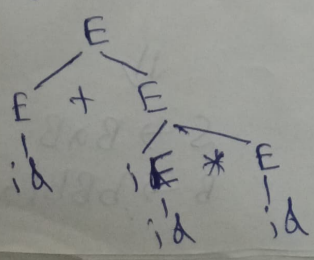
E → E*E
→ E+E*E
→ id+E*E
→ id+id*E
→ id+id*id

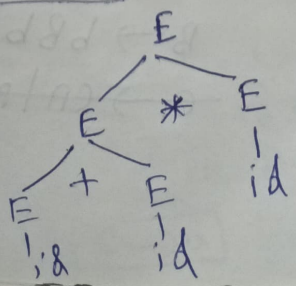
RMD (Rightmost Derivation) –

E ⇒ E+E
⇒ E+E*E
⇒ E+E*id
⇒ E+id*id
⇒ id+id*id

E ⇒ E*E
⇒ E*id
⇒ E+E*id
⇒ E+id*id
⇒ id+id*id

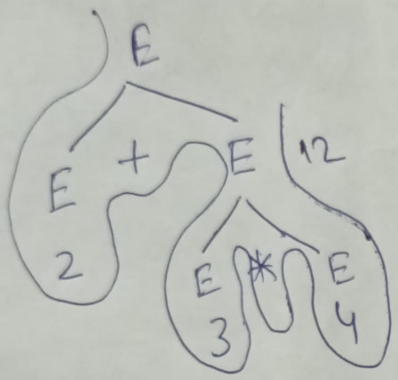
Using parse tree / derivation tree –



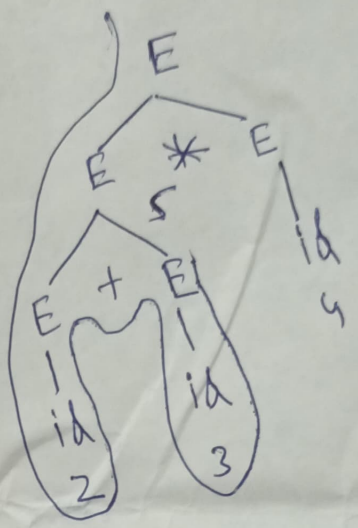


for given grammer, if we get more than one LMD or RMD or parse tree, then the given grammer is ambigous

Ex- 2+3*4



2+3*4=14



2+3*4=20

So, parser gets confused, it doesn't know which one to use.

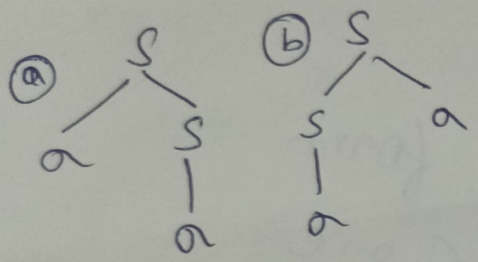
→ Problems in Ambigous Grammer -
- Rule of associativity fails.
- Rule of precedence fails.

→ How to find grammer is ambigous ?
- no algorithm exist so hit & trial.
- it is an undecidable problem.

Q Identify grammer is ambigous or not?

① $S \to aS \mid Sa \mid a$

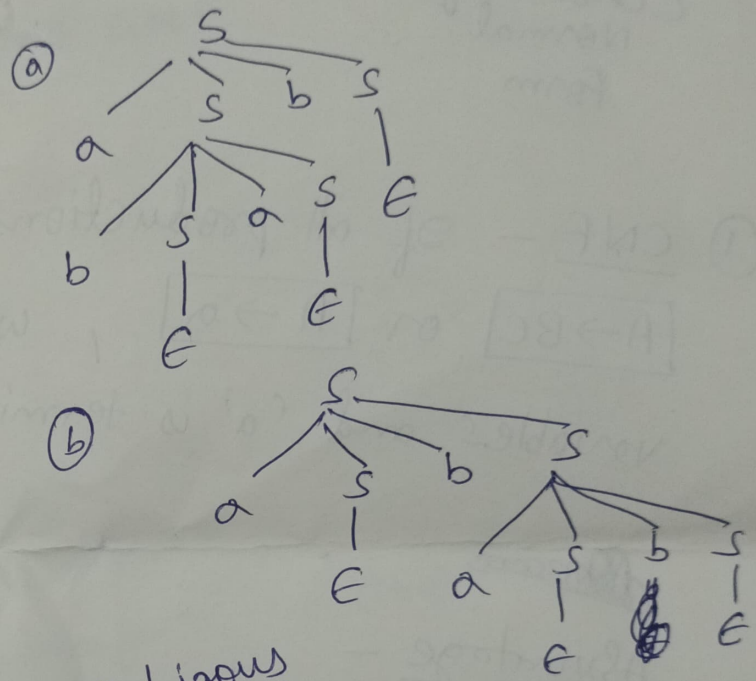
$w = aa$



Ambigous

② $S \to aSbS \mid bSaS \mid \epsilon$

abab



Ambigous

Grammers

Q $a^n b^n \mid n \geq 1$

$S \rightarrow aSb \mid ab$

Q $ww^R \cup waw^R \cup wbw^R$

↓ even length palindrome

↓ odd length palindrome

$w = abab \quad w^R = baba$

$S \rightarrow aSa \mid bSb \mid a \mid b \mid \in$

$a \mid b$ → odd length palindrome

$\in$ → even length palindrome

Q $a^n b^n c^m \mid n, m \geq 1$

$S \rightarrow AB$

$A \rightarrow aAb \mid ab$

$B \rightarrow cB \mid c$

Q $a^n c^m b^n \mid n, m \geq 1$

$S \to aSb \mid aAb$
$A \to cA \mid c$

Q $a^n b^m c^n d^n \mid n, m \geq 1$

$S \to AB$
$A \to aAb \mid ab$
$B \to cBd \mid cd$

Q $a^n b^{2n} \mid n \geq 1$

$S \to aSbb \mid abb$

Q $a^n b^m c^m d^n$

$S \to aSd \mid aAd$
$A \to bAc \mid bc$

Q $a^{m+n} b^m c^n$

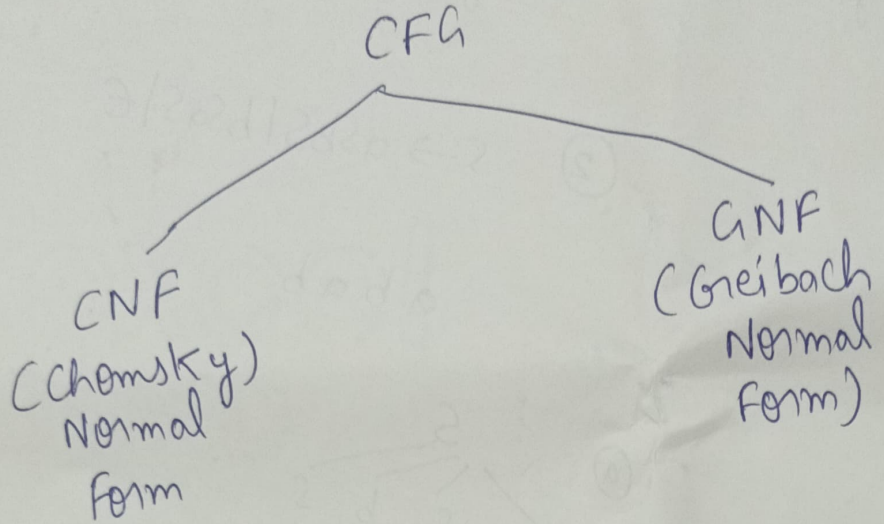
$a^n a^m b^m c^n$

$S \to aSb \mid aAb$
$A \to aAc \mid ac$

Q $a^n b^m c^{n+m}$

$a^n b^m c^m c^n$

$S \to aSc \mid aAc$
$A \to bAc \mid bc$



① CNF – If all productions are of form $A \to BC$ or $A \to a$, where A, B, C are variables and 'a' is terminal.

Advantage –

→ String length will increase by 1 in every step or no of terminals increase by by 1

→ Easy to apply CYK membership algorithm.

→ Ex- $S \to AB$
$A \to a$
$B \to b$
$A \to AB$

abb

$S \to AB$
$\to ABB$
$\to aBB$
$\to abB$
$\to abb$

$2(3)-1 = 5$

ab

$S \to AB$
$\to aB$
$\to ab$

$2(2)-1 = 3$

So, # steps required to derive a string of length |w|, is $(2|w|-1)$.

(2) GNF (Greibach Normal form) –

If all the productions are of form $A \to a\alpha$ where $\alpha \in V^*$, then it is called GNF.

Ex – $A \to aBCDE$
$A \to a$

Adv –

(1) # step required to generate a string of length |w| is |w| because in every step one terminal, w terminal w step.

(2) GNF is useful to convert CFG to PDA.

Closure Properties of CFL -

1) Union - closed

L_1 | L_2

CFG$_1$ | CFG$_2$

$S_1 \rightarrow \ldots$ | $S_2 \rightarrow \ldots$

$S \rightarrow S_1 | S_2$

(Resultant Grammer)

2) $L_1 \cdot L_2$ (Concatenate) - closed

$S \rightarrow S_1 . S_2$

$S_1 \rightarrow \ldots$

$S_2 \rightarrow \ldots$

3) Kleene Closure →

L_1 — CFG$_1$ S_1

$L_1^* \rightarrow S \rightarrow S_1 S | \epsilon$

④ Intersection –

$L_1 \cap L_2$ (may or may not be CFL)

$L_1 = \{a^n b^n c^m \mid n, m \geq 0\}$

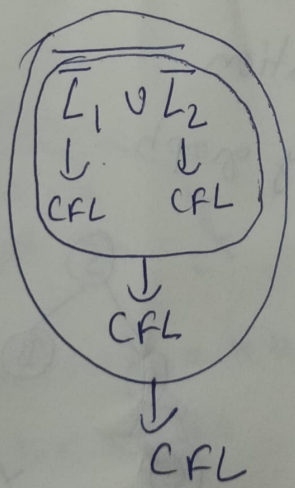
$L_2 = \{a^m b^n c^n \mid n, m \geq 0\}$

$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ CFG X

⑤ Complement –

Assume complement is closed,

$L_1 \cap L_2 =$



but we proved $L_1 \cap L_2$ is not CFL

so one assumption is wrong.

Decidable Problem on CFL's –

① Membership – Decidable

$w \in L$?

CYK algo

② Emptiness – decidable

is $L = \phi$?

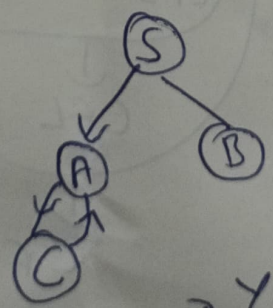
CFG → simplification (ϵ eliminate, useless symbol) → If start symbol is useless, then nothing can be generated so L is ϕ else not.

③ finiteness –

1) CFG → Simplification

2) Draw Dependency graph

$S \to AB$
$A \to aC \mid a$
$C \to aA \mid b$
$B \to a$



If DG has cycle → Y (infinite)
N (finite)

Pumping Lemma for CFL's –

for all CFL, L $\exists$ an integer $n \geq 1$ such that $\forall z \in L$ and $|z| \geq n$ and

(i) $z = uvwxy$

(ii) $|vwx| \leq |z|$

(iii) $|vx| \neq 0$ or $|vx| \geq 1$

that $uv^i w x^i y \in L \quad \forall i \geq 0$

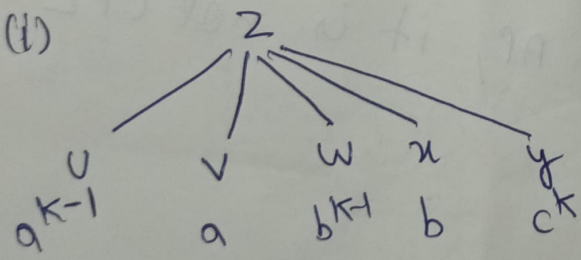
Q $L = \{a^m b^m c^m \mid m > 0\}$

Assume L to be CFL

$z = a^k b^k c^k$ and there exist n

$|z| = 3k \geq n$

(i)



(ii) $|vwx| = k+1 \leq 3k$

(iii) $|vx| = 2 \neq 0$

$a^{k-1} a^i b^{k-1} b^i c^k \in L \quad \forall i \geq 0$

$i = 2$, $a^{k+1} b^{k+1} c^k \notin L$

Hence our assumption is wrong

therefore not CFL.

Special Case -

If $\Sigma = \{a\}$ and L is defined over Σ then L is CFL iff lengths of strings in L are in AP.

Ex- $L = \{a^{3n} \mid n \geq 0\}$

$L = \{a^0, a^3, a^6, a^9, \ldots\}$

Since AP, it is CFL

Ex- $L = \{a^{3m!} \mid n \geq 0\}$

$L = \{a^0, a^3, a^6, a^{18}, a^{72}, \ldots\}$

not on AP, it is not CFL

Closure Property Table (to remember)

	RL	CFL
Union	✓	✓
Intersection	✓	×
Set difference	✓	×
Complement	✓	×
Concatenation	✓	✓
Kleene Closure	✓	✓
Reversal	✓	✓
Intersection with RL	✓	✓
Union with RL	✓	✓
Subset	×	×

Note-

→ No languages are closed under subset & infinite union.