# Module 3. Context free Grammar

- CFG is a formal grammar which is used to generate all possible patterns of strings in a given formal language.

$G = \{V, T, P, S\}$.

It consist of set of production rule

$S \rightarrow Ab$

S: Non terminal; Ab: combination of variables & Terminal

V is the final set of non-terminal symbol. It is denoted by capital letter.

~~eg:~~

eg :- $A = \{s, a, b, P, S\}$.

s → starting symbol

a, b → terminals represented by small characte

P → variable along with S.

Uses: for defining programming languages.

① Construct the CFG for language having any numbe of a's over the set $\Sigma = \{a\}$.

→ As we know the regular expression for the above lang is

$R.E = a^*$

Production rule for R.E is

$S \rightarrow aS$ : rule 1

$S \rightarrow \varepsilon$ rule 2

Here, $S \rightarrow aS$

$S = \varepsilon$

$S \rightarrow aS$

aaS rule 1

aaaS rule 1

aaaaS rule 2

→

aaaaas - rule 1
aaaaaae - rule 2
aaaaaa.

Q. Construct CFG for language $L = \{wcc^R\}$ where $w \in (a,b)^*$ }

→ The string that can be generated for the give language is {aacaa, bcb, abcab, bacab, abbcbba }

- The grammer could be
  S → a Sa rule 1
  S → bSb rule 2
  S → c rule 3

Q. S→aSa, S→bSb, S→c, t2

* abbcbba

| | |
|---|---|
| S → aSa | rule 1 |
| abSba | rule 2 |
| abbSbba | 2 |
| abbcbba | 3 |

Q. Construct CFG for language $L = a^n b^{2n}$ where $n \geq 1$

→ L = {abb, aabbbb, aaabbbbbb ... }

The grammar could be:
S → asbb
S → abb

Now if we want to derive a string "aabbbb", we can start with S → asbb
S → aabbbb

* Derivation :- sequences of production rules.

Q. Derive the string "abbb" for leftmost derivation & rightmost derivation & is.
S → AB | G
A → aB
B → Sb

→ Sol":-

LD

S → AB
rule 2 aB B
3 aSb B
1 aεbB
3 abB
1 abSbb
abεbb
abb

RD

S → AB
A S b 3 rule
A ε b 1 rule
A b
aBb 2 rule
aSbb 3 rule
aεbb 1 rule
abb

Q. Derive the string "00101" for leftmost derivation and rightmost derivation using a CFG given by
S → A|B
A → 0A | ε
B → 0B | 1B | ε

→ LD

S
A B
0A B
00A B
00 B
001 0B
00101B

RD

S
A B
A 10B
A 101B
A 101
0A 101
00A101
00101

13/2/23

## ★ Derivation tree

→ Parse tree properties.

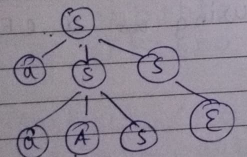
1. The root node is always a node indicating start symbol.
2. The derivation is read from left & to right
3. The leaf node is always terminal nodes.
4. The interior nodes are always the non-terminal nodes.

Root vertex : Must be labelled by S.

Left Most DT
obtained by applying production to the leftmost variable in each step.

Right Most DT.

eg:-



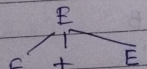
① Production rules:-

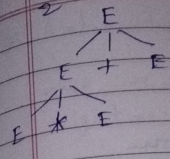
E = E + E
= E * E

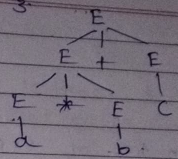
E = a | b | c

Input :- a * b + c

step 1:



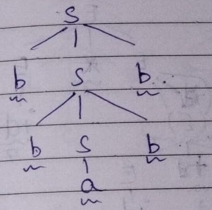




② Draw a derivation tree for string "bbabb" from the CFG given by

S → bSb | a | b

→



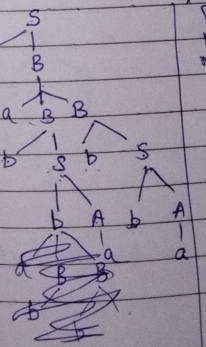
∴ bbabb.

③ aabbabba ...

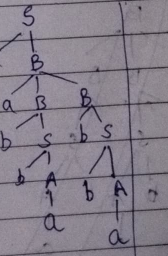
S → aB | bA
A → a | aS | bAA
B → b | bS | aBB

→

S
aB
aaBB
aabSB
aabbAbA
aabbabba



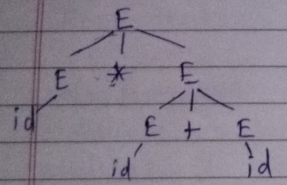
Done by myself



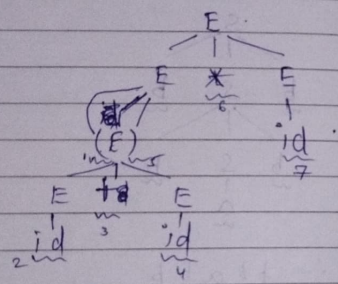
RMDT (HW)

④ E = E + E
E = E * E
E = (E)
E = id
obtain derivation tree of expression.
a. id * id + id
b. (id + id) * id

→ id * id + id



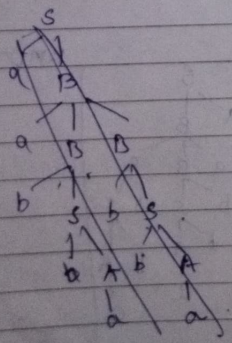

(id + id) * id




⑤ S → AB | ε
A → aB
B → Sb

aaabbbb

→



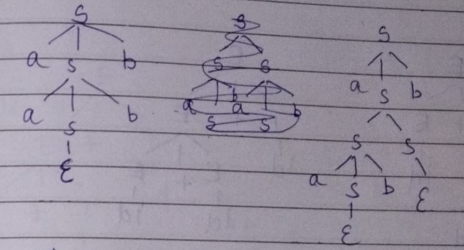

15/6/24

## * Ambiguity in Grammar.

more than one leftmost derivation or more than one rightmost derivation or more than one parse tree.

① ~~E = +~~ S = aSb | ss
~~E = E - E~~ S → ε
~~E = E * E~~ (aabb).

→



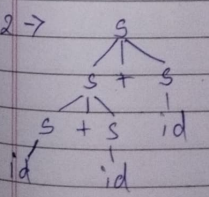

Since tree are 2 tree ∴ It is ambiguious.

② S → S + S , S → id , I/p = 'id + id + id'.

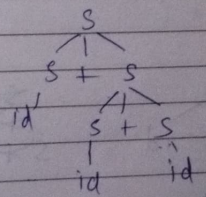
③ S → S + S , S → S * S , S → id , I/p = 'id + id * id'

④ E → E + E / id I/p = 'id + id + id'.

② → 

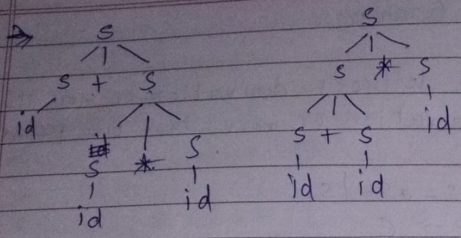



or



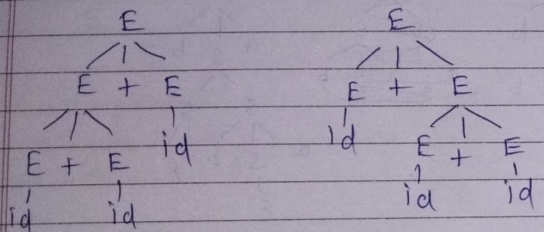

Ambiguous.

3. →




ambigous.

4. →



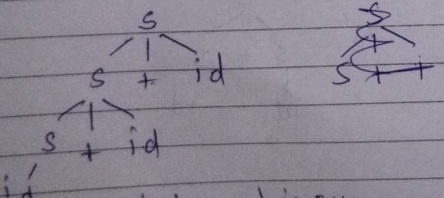

ambigous.

(5) $S \to S+id$, $S \to id$, I/p → 'id+id+id'.

(6) $S \to S+T$, $S \to T$, $T \to T*F$, $T \to F$, $F \to id$
I/p = 'id+id*id'

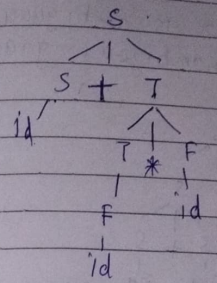
(7) $E \to E+T$, $E \to T$, $T \to id$, I/p = 'id+id+id'.

5. →



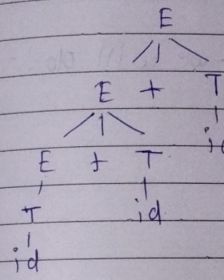

Not ambigous.

6. →




not

7. →

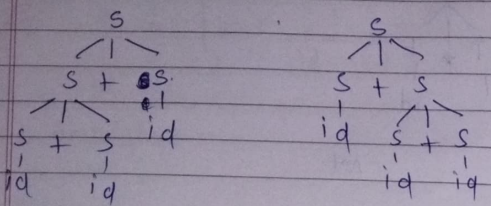



not

* Convert ambigous to non-ambigous.
1. eg :- $X \to Xa$
2. eg :- $X \to aX$

Q. Consider a Grammar $S \to S+S$, $S \to id$, I/p = 'id+id+id'. Determine whether grammar is ambiguous if it is ambigous construct unambigous grammar.

→

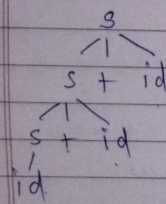



It is ambigous

For unambigous grammar we will do :-

$S \to S+id$

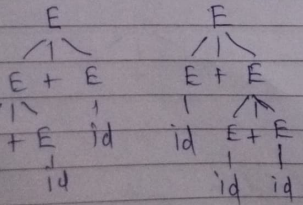
$S \to id$




Q. check if ambigous or not. If ambigous make it as unambigous.

$E \to E+E$
$E \to id$
} I/p = 'id+id+id'.

→




ambigous.

To make ambigous :-

$E \to E+T$

$E \to T$

$T \to id$

* Chomsky Hierarchy.

It represents the class of language that are accepted by the diff. machine. The category of language in CH is as given :-

Type 0 known as Unrestricted Grammar

Type 1 — " — Context Sensitive " "

Type 2 — " — Free " "

Type 3 Regular grammar

- Type 0 :- There is no restriction on grammar

For eg

$bAa \to aa$

$S \to s$

- Type 1 :- $S \to AT$
  $T \to xy$
  $A \to a$

- Type 2 :-
  eg :- $A \to a$

- Type s :- $A \to xy$

| Grammar Type | Grammar Accepted | Language Accepted | Automata |
|---|---|---|---|
| Type-0 | Unrestricted Grammar | Recursively Enumerable language | Turing Machine |
| Type-1 | context sensitive grammar | Context sensitive language | linear Bounded Automata |
| Type-2 | context free Grammar | context free language | Pushdown Automata |
| Type-3 | Regular Grammar | Regular Language | Finite state automata |

20/2/23

* Simplification of CFG

1. each variable (i.e non-terminal) & each terminal of G appears in the derivation of some word in L.
2. There should not be any production as $X \to Y$ where X and Y are non-terminal.
3. If $\varepsilon$ is not in lang.

RG (Regular Grammar)

- Removal of useless symbol
- Elimination of $\varepsilon$ Production
- Removal of unit production

① Removal of useless symbol.

If it does not appear on the right hand side of Prod. rule.

for eg:-

$T \to aaB \mid abA \mid aaT$

$A \to aA$

$B \to ab \mid b$

$C \to ad$ (useless symbol) (becoz it is not in right side of T)

aA is also useless becoz there is no way to terminate it. If it never terminates, then it can never produce a string. Hence this prodn. can never take part in any derivation we can not replace A with small letter.
∴ we need to replace all capital letters.

$T \to aaB \mid aaT$
$B \to ab \mid b$ ← final.

② Elimination of $\varepsilon$ Production

1. First find out all nullable non-terminal variable which derives $\varepsilon$.
2. for each prod. $A \to \alpha$, construct all production $A \to x$, where x is obtained from $\alpha$ by removing one or more non-terminal from step 1.

eg:- Remove the prod. from the foll. CFG by preserving the meaning of it.

$S \to XYX$
$X \to 0X \mid \varepsilon$
$Y \to 1Y \mid \varepsilon$

$S \to XYX$
$X \to 0X$
$X \to \varepsilon$ ✗
$Y \to 1Y$
$Y \to \varepsilon$ ✗

$S \to XYX$
$X \to 0X$
$Y \to 1Y$

③ Removing unit production :-

1. To remove $X \to Y$, add production $X \to \alpha$ to the grammar rule whenever $Y \to \alpha$ occurs ~~in grammar~~ in grammar.
2. Now delete $X \to Y$ from the grammar.
3. Repeat step 1 & step 3 until all unit production are remove

eg:- $S \to 0A \mid 1B \mid C$
$A \to 0S \mid 00$
$B \to 1 \mid A$
$C \to 01$

* we can write:

$S \to 0A$ $A \to 0S$ $B \to 1$ $C \to 01$
$S \to 1B$ $A \to 00$ $B \to A$
$S \to C$ ✗

New production rule

$S \to 0A \mid 2B \mid 01$
$A \to 0S \mid 00$
$B \to 0S \mid 00 \mid 11$
$C \to 01$

* Chomsky's Normal Form (CNF).

CFG is in CNF if all prod. rules satisfy one of the foll. condⁿ:

- start symbol generating ε.
  For eg, $A \to \varepsilon$
- A non-terminal generating two non-terminals.
  For eg, $S \to AB$
- A non-terminal generating a terminal
  For eg, $S \to a$.

① $G1 = \{S \to AB, S \to c, A \to a, B \to b\}$
G1 satisfy the rules specified for CNF, so the grammar G1 is in CNF

② $G2 = \{S \to aA, A \to a, B \to c\}$
CNF as $S \to aA$ contains terminal followed by non terminal.
∴ G2 Not in CNF.

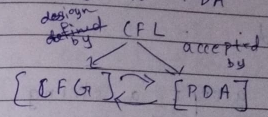
* Step's CFG to CNF.

1. $S1 \to S$ (eliminate start symbol from RHS). (S1 → new start symbol.)
2. Remove null, unit & useless prod.
3. $S \to aA$ (decomposed as :)
   $S \to RA$
   $R \to a$
4. $S \to ASB$ (can be decomposed as :)
   $S \to RS$ or $S \to RB$
   $R \to AB$ $R \to AS$

#5+24

* Equivalence : PDA & CFG

1) convert CFG to PDA



Q. Design a equivalent PDA for the given grammar.

$S \rightarrow 0S1 \mid 00 \mid 11$ (Variables → Terminals)

→ $\delta(q, \varepsilon, S) = (q, 0S1), (q, 00), (q, 11)$ — ① { Rule 1 $\delta(q, \varepsilon, A) = (q, B)$

$\delta(q, 0, 0) = (q, \varepsilon)$ — ② } Rule 2

$\delta(q, 1, 1) = (q, \varepsilon)$ — ③ { $\delta(q, a, a) = (q, \varepsilon)$

PDA is designed.

Now, we have to accept a string '0111'

1) $\delta(q, 0111, S)$ using 1.
2) $\delta(q, 0111, 0S1)$ Rule 2 (P, P)
3) $\delta(q, 111, S1)$ Rule 1
4) $\delta(q, 111, 111)$ Rule 3
5) $\delta(q, 11, 11)$ Rule 3
6) $\delta(q, 1, 1)$ Rule 3
7) $\delta(q, \varepsilon, \varepsilon)$ Accept

Q. Design a equivalent PDA for the given grammar.

$S \rightarrow 0BB$

$B \rightarrow 0S \mid 1S \mid 0$

string :- {010000}

→ $\delta(q, \varepsilon, S) = (q, 0BB)$ — ①

$\delta(q, \varepsilon, B) = (q, 0S), (q, 1S), (q, 00)$ — ② { Rule 1

$\delta(q, 0, 0) = (q, \varepsilon)$ — ③ } Rule 2

$\delta(q, 1, 1) = (q, \varepsilon)$ — ④ }

PDA is designed

Now, we have to accept a string '010000'

1) $\delta(q, 010000, S)$ using 1
$\delta(q, 010000, 0BB)$ using 3
$\delta(q, 10000, BB)$ using 2.
$\delta(q, 10000, 1SB)$ using 4
$\delta(q, 0000, SB)$ using 1
$\delta(q, 0000, 0BBB)$ using
$\delta(q, 000, BBB)$.
$\delta(q, 000, 0BBB)$ or $\delta(q, 000, 0SB)$
$\delta(q, 00, BB)$ $\delta(q, 00, SB)$
$\delta(q, 00, BB)$ $\delta(q, 00, 0BB)$
$\delta(q, 0, B)$ $\delta(q, 0, BB)$
$\delta(q, \varepsilon, 0)$ $\delta(q, 0, 0S1)$
$\delta(q, 0, 0)$
$\delta(q, \varepsilon, \varepsilon)$

Q $S = AA \mid a$ abbabb

$A = SA \mid b$

→ $\delta(q, \varepsilon, S) = (q, A, A), (q, a)$

2) Conversion PDA to CFG

$M = \{q, p\}, \{0, 1\}, \{x, z\}, \delta, q, z$ ← Initial stack symbol, initial state

Q  Σ  Γ

1) $\delta(q, 1, z) \Rightarrow (q, xz)$ Push
2) $\delta(q, 1, x) \Rightarrow (q, xx)$ push
3) $\delta(q, \epsilon, x) \Rightarrow (q, \epsilon)$ pop
4) $\delta(q, 0, x) \Rightarrow (p, x)$ Nop
5) $\delta(p, 1, x) \Rightarrow (p, \epsilon)$ Pop
6) $\delta(p, 0, z) \Rightarrow (q, z)$ NOP

→ $S \to [q\ z\ q]$, $S \to [q\ z\ p]$ } $\to [q_0, z_0, p]$

Step 1 $\delta(q, 1, z) \Rightarrow (q, xz)$ push.

$[q, z, q] = 1\,[q, x, q]\,[q, z, q]$

$[q, z, q] = 1\,[q, x, p]\,[p, z, q]$

$[q, z, p] = 1\,[q, x, q]\,[q, z, p]$

$[q, z, p] = [q, 4, p]\,[p\ z\ p]$

→ 1) write down the all combination of state * stack symbol.

$V = \{S, [q\,x\,q], [q\,x\,p], [p\,x\,q], [p\,x\,p], [p\,z\,q], [q\,z\,p], [q\,z\,p], [p\,z\,q]\}$

2) Write down the Grammer $\{q_0, z_0, p\}$.

$S \to [q\ z\ q]$
$S \to [q\ z\ p]$

3. solve for each transition

a) $\delta(q, 1, z) \Rightarrow (q, xz)$ push $\{z^2 = 4\}$

$[q, z, q] = 1\,[q, x, q]\,[q, z, q]$

$[q, z, q] = 1\,[q, x, p]\,[p, z, q]$

$[q, z, p] = 1\,[q, x, q]\,[q, z, p]$

$[q, z, p] = 1\,[q, x, p]\,[p, z, p]$

b) $\delta(q, 1, x) \Rightarrow (q, xx)$ push $(z^2 = 4)$

$[q, x, q] \to 1\,[q, x, q]\,[p, x, q]$

$[q, x, q] \to 1\,[q\ x, p]\,[p\ x, q]$

$[q$

c) $\delta(q, \epsilon, x) \Rightarrow (q, \epsilon)$ pop

$[q, x, q] \to \epsilon$

d) $\delta(q, 0, x) \Rightarrow (p, xz)$ NP

$[q, x, q] \to 0\,[p, x, q]$

$[q, x, p] \to 0\,[p, x, p]$

e.) $\delta(p, a, x) \rightarrow (p, \varepsilon)$ Pop

$[p, x, p] \rightarrow 1$

f) $\delta(p, 0, z) \Rightarrow (q, z)$ Nop

$[p, z, q] \Rightarrow 0\,[q, z, q]$

$[p, z, p] \Rightarrow 0\,[q, z, p]$

| Original Name | New name |
|---|---|
| $[q, z, q]$ | A |
| $[q, z, p]$ | B |
| $[p, z, q]$ | C |
| $[p, z, p]$ | D |
| $[q, x, q]$ | E |
| $[q, x, p]$ | F |
| $[p, x, q]$ | G |
| $[p, x, p]$ | H |

1) $A \rightarrow 1 \mid EA$
$A \rightarrow 1EC$
$B \rightarrow 1EB$
$B \rightarrow 1FD$

2) $E \rightarrow 1EE$
$E \rightarrow 1EG$
$F \rightarrow 1EF$
$F \rightarrow 1FH$

3) $E \rightarrow \varepsilon$

4) $E \rightarrow 0G$
$F \rightarrow 0H$

5) $H \rightarrow 1$

6) $C \rightarrow 0A$
$D \rightarrow 0B$

$S \rightarrow A \mid B$
$A \rightarrow 1EA \mid 1FC$
$B \rightarrow 1EB \mid 1FD$
$E \rightarrow 1EE \mid 1FG \mid 1E \mid 0G$
$F \rightarrow 1EF \mid 1FH \mid 0H$
$H \rightarrow 1$
$C \rightarrow 0A$
$D \rightarrow 0B$

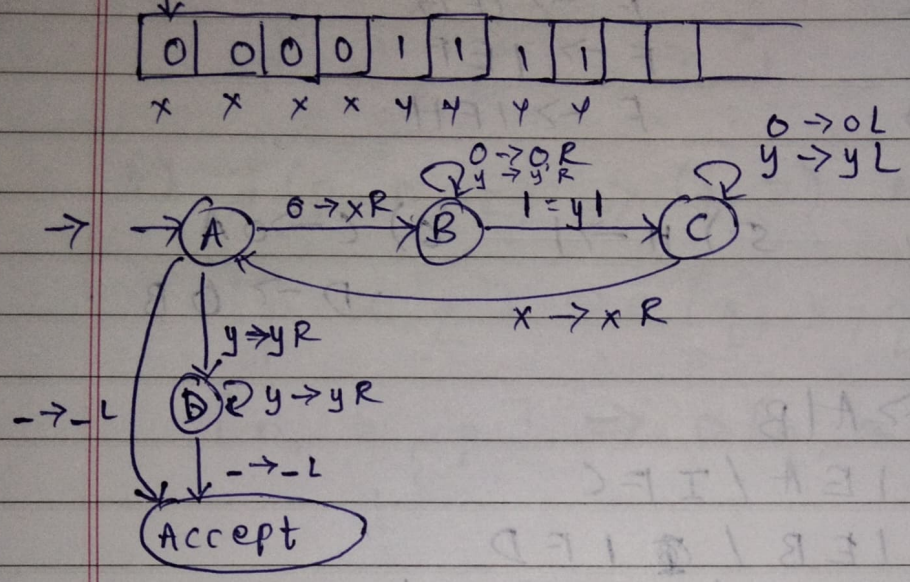
* PDA to CFG

1. $\delta(q_0, a, z_0) \Rightarrow (q_0, az_0)$
2. $\delta(q_0, a, a) \Rightarrow (q_0, aa)$
3. $(q_0, 1, a) \Rightarrow (q_1, a)$
4. $\delta(q_1, b, a) \Rightarrow (q_1, a)$
5. $\delta(q_1, a, a) \Rightarrow (q_1, \wedge)$
6. $\delta(q_1, \wedge, z_0) \Rightarrow (q_1, \wedge)$

13/3/24

# Module 5 and 6

Q. Design a turing Machine which represents $L = 0^N 1^N$




14/3/24 Q construct a DFA for the foll.

i. All strings that ~~accept~~ contains exactly 4 0's

ii All strings that does not contain '110'

→