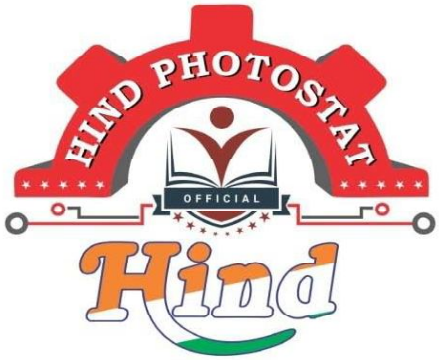


HindPhotostat

Hind Photostat & Book Store

Best Quality Classroom Topper Hand Written Notes to Crack GATE, IES, PSU's & Other Government Competitive/ Entrance Exams

MADE EASY

Computer Science Engineering / IT

Toppers Handwritten Notes

Compiler Design

By-Prasad sir

- Theory
- Explanation
- Derivation
- Example
- Shortcuts
- Previous Years Question With Solution

Visit us:-www.hindphotostat.com

Courier Facility All Over India
(DTDC & INDIA POST)

Mob-9311989030

HindPhotostat

GET IT ON Google Play

MADE EASY , IES MASTER , ACE ACADEMY , KREATRYX

ESE , GATE, PSU BEST QUALITY TOPPER HAND WRITTEN NOTES
MINIMUM PRICE AVAILABLE @ OUR WEBSITE

1. ELECTRONICS ENGINEERING
2. ELECTRICAL ENGINEERING
3. MECHANICAL ENGINEERING
4. CIVIL ENGINEERING
5. INSTRUMENTION ENGINEERING
6. COMPUTER SCIENCE

IES ,GATE , PSU TEST SERIES AVAILABLE @ OUR WEBSITE

- IES –PRELIMS & MAINS
- GATE

➢ NOTE;- ALL ENGINEERING BRANCHS

➢ ALL PSUs PREVIOUS YEAR QUESTION PAPER @ OUR WEBSITE

PUBLICATIONS BOOKS -

MADE EASY , IES MASTER ,ACE ACADEMY ,KREATRYX ,GATE ACADEMY, ARIHANT ,GK

RAKESH YADAV, KD CAMPUS , FOUNDATION , MC –GRAW HILL (TMH) ,PEARSON...OTHERS

HEAVY DISCOUNTS BOOKS AVAILABLE @ OUR WEBSITE

F230, Lado Sarai New Delhi-110030 Phone: 9311 989 030	Shop No: 46 100 Futa M.G. Rd Near Made Easy Ghitorni, New Delhi-30 Phone:9711475393	F518 Near Kali Maa Mandir Lado Sarai New Delhi-110030 Phone: 9560 163 471	Shop No.7/8 Saidulajab Market Neb Sarai More, Saket, New Delhi-30

Website: www.hindPhotostat.com

Contact Us: 9311 989 030

Courier Facility All Over India

(DTDC & INDIA POST)

COMPILER

Grammar $G = (V, T, P, S)$ — S → Start Symbol; V → Variable; P → Production; T → Terminals

Example →
$S \longrightarrow ABC$
$AB \longrightarrow CD$
$C \longrightarrow a$
$D \longrightarrow b$

Variables : {S, A, B, ...} — given by grammar (check), otherwise if not given, consider Capital letters as variables.

Types of Grammar (according to Noam Chomsky)

1. Type–0 (unrestricted Grammar) — By default, every grammar is Type–0.
2. Type–1 (Context Sensitive Grammar)
3. Type–2 (Context Free Grammar)
4. Type–3 (Regular Grammar)

Type–0 → $\alpha \longrightarrow \beta$ — unrestricted because no restrictions
where $\alpha, \beta \in (V+T)^*$

Type–1 → ① Type–0 ($A \to \epsilon$, production not allowed)
② $|\alpha| \leq |\beta|$

Type–2 → ① $A \longrightarrow \beta$ (single variable on left side) no restriction on right side.
② $A \in V$
$\beta \in (V+T)^*$

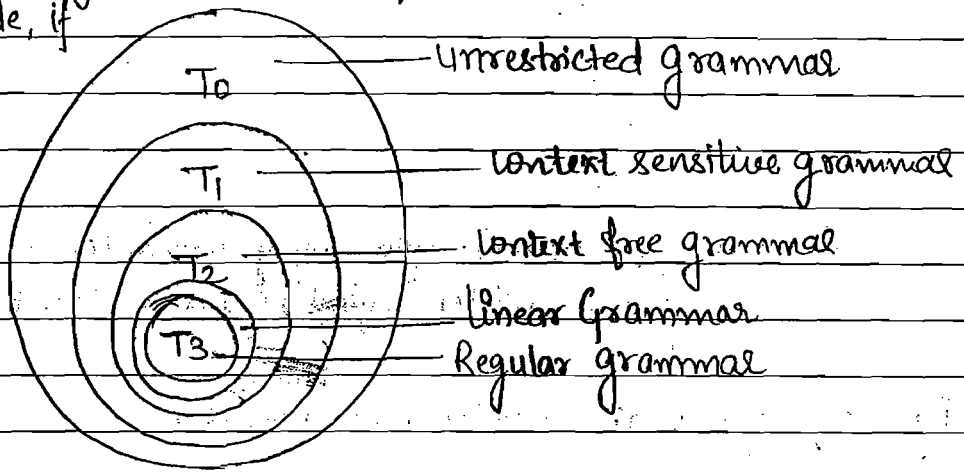
Type–3 → ① $A \longrightarrow \beta T^* / T^*$ (Left linear Grammar)
or
$A \longrightarrow T^* \beta / T^*$ (Right linear Grammar)

Page. No.
Date : / /

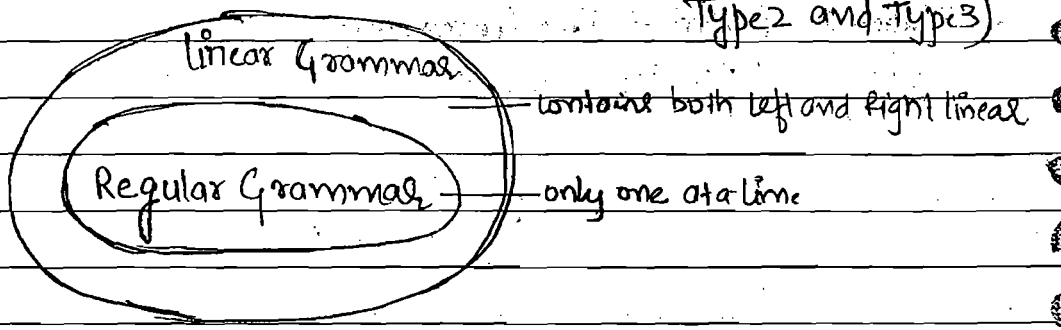
Ex → S → ABC
A → ab
B → CD

T_2 ✓, T_3 ✗ (Right side)
CFG

① First check left side, single variable, T_2 confirmed
② Then check Right side, if doesn't follow left and Right linear, then not T_3 (X)



$V \rightarrow T^*VT^* / T^*$ → Linear Grammar (In between Type2 and Type3)



Context Free Grammar (CFL)

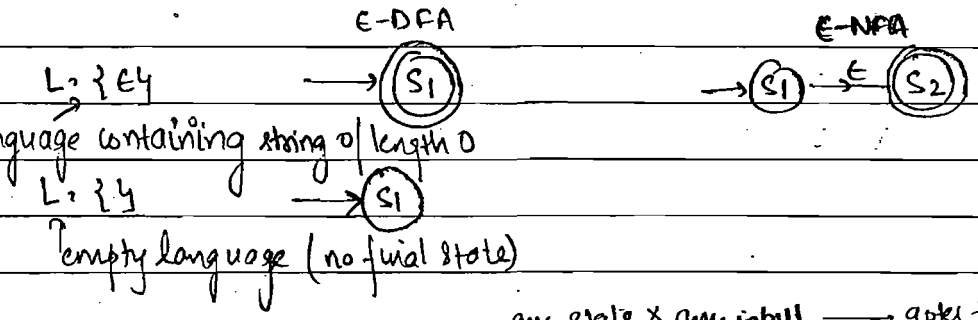
① Write a context free Grammar for a language
$L_2 = \{a^m b^n \mid m, n \geq 0\}$

$S \rightarrow AB$
$A \rightarrow aA \mid \epsilon$
$B \rightarrow bB \mid \epsilon$

Page. No.
Date : / /

ϵ ⟶ string of length 0 (empty string)

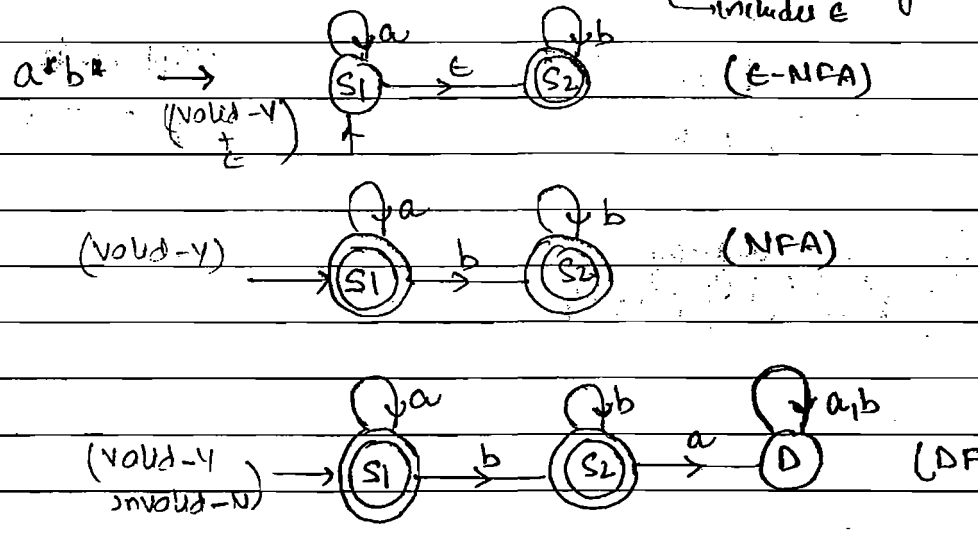
$L = \{\}$ ⟶ Empty language



Transition function ⟶ DFA: $Q \times \Sigma \longrightarrow Q$ (any state × any input ⟶ goes to one of state)

NFA: $Q \times \Sigma \longrightarrow 2^Q$

ϵ-NFA: $Q \times \Sigma \cup \{\epsilon\} \longrightarrow 2^Q$ can go to any number of states (includes ϵ)



Dead or Trap state → Permanent Non-final states.
Non-final states → Temporary Non-final states

→ Can DFA have more than one final state?

↳ DFA can have multiple final states and dfa doesn't accept the NULL move (ϵ ✗ dfa)

Page. No.
Date : / /

② Give context free Grammar for language

$L = \{ a^m b^n c^n \mid m, n \geq 1 \}$

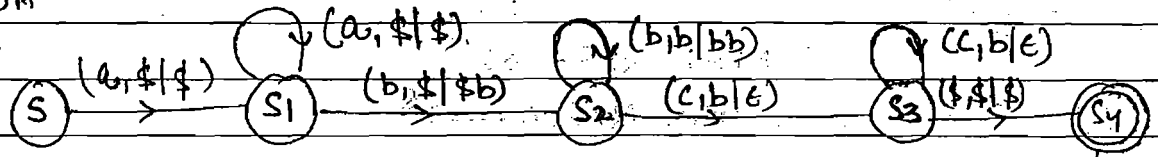
$S \longrightarrow AB$

$A \longrightarrow aA \mid a$

$B \longrightarrow bBC \mid bc$

PDA = Finite Automata + 1 stack

DPDA



acceptance by final state

PDA → acceptance by final state
→ acceptance by empty stack

③ Give CFG for the language

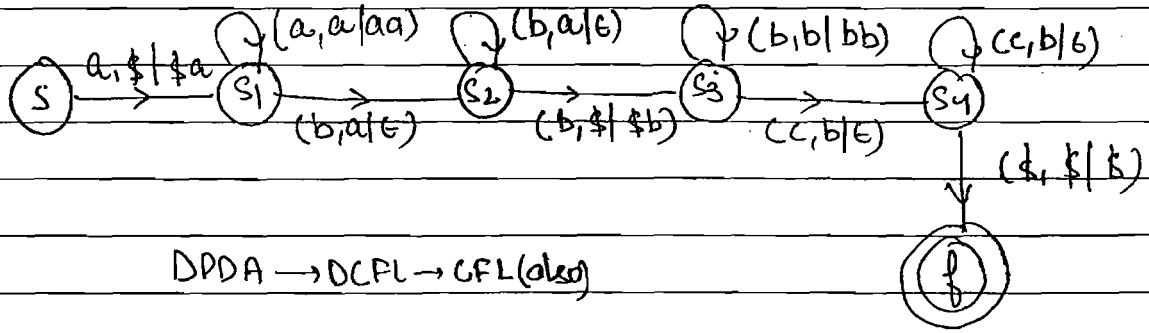
$L = \{ a^i b^{i+j} c^j \mid i, j \geq 1 \}$

$a^i b^i b^j c^j$

$S \longrightarrow AB$

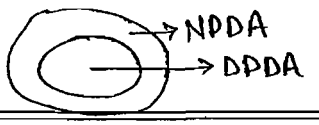
$A \longrightarrow aAb \mid ab$

$B \longrightarrow bBC \mid bc$



DPDA → DCFL → CFL (also)

→ NFA is equivalent to DFA
→ NPDA has more power than DPDA.



aabbcc

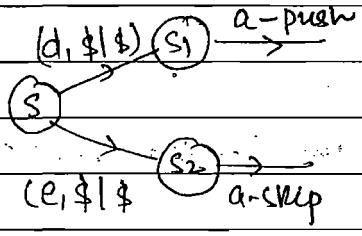
Page. No.
Date : / /

(3) $L_1 = \{ a^m b^m c^n \cup a^m b^n c^n \mid m, n \geq 1 \}$

$S \longrightarrow S_1 \mid S_2$
$S_1 \longrightarrow AB$
$A \longrightarrow aAb \mid ab$
$B \longrightarrow cB \mid c$
$S_2 \longrightarrow CD$
$C \longrightarrow aC \mid a$
$D \longrightarrow bDc \mid bc$

NPDA → CFL language

If $L_2 = \{ d a^m b^m c^n \cup e a^m b^n c^n \mid m, n \geq 1 \}$



→ no ambiguity
→ DPDA

(5) CFG for language, $L = \{ a^n b^n c^n \mid n \geq 1 \}$
not CFL → It is CSL

LBA = Finite Automata + 2 stacks

(1) → Union of two DCFL's need not be DCFL.

Ex → (4) above. $L_1 = \{ a^m b^m c^n \cup a^m b^n c^n \mid m, n \geq 1 \}$

DCFL CCFL

Union

→ CFL (here in this example)

But can be possible with some example or not.

Page. No.
Date : / /

⑤ Intersection of two DCFL's, need not be DCFL.

⑥ Intersection of two CFL's, need not be CFL.

⑦ CFL are not closed under complementation.

Ex → $a^n b^n c^n$ ← CSL — complement → Other than $a^n b^n c^n$ comb — CFL

(Compiler part)

⑥ Given CFG for language L = set of all Arithmetic expression over the α, β, id.

$E \rightarrow id \mid E+E \mid E*E \mid E-E \mid E/E \mid (E)$

$id \rightarrow +/-/*/1,$

$\alpha \rightarrow (id+id)\alpha$

$A\alpha \rightarrow id\alpha / id\beta A$

$\alpha \rightarrow +/-/*/$

⑦ Give CFG for language
L = set of all Boolean expressions, over the alphabet 0 and 1.

$B \rightarrow 0/1/B \text{ or } B/B \text{ and } B/\text{not } B/(B)$

⑧ Give CFG for language L = set of all Regular Expressions over the alphabet a,b ($\Sigma = \{a,b\}$).

$R \rightarrow \epsilon/a/b/R^*/R+R/R*R/(R)$

$(a+b)^*$

$A \rightarrow a/b/A^*/(a+A)$

$\epsilon \quad A+A)$

$a^* \; b^*$

$R*$

$(R+R)^*$

$(a+b)^*$

$a(a+b)^*$

$R\cdot(R+R)^*$

$a\cdot(a+b)^*$

⑨ Consider the following Grammar

$S \rightarrow as/sa/a$

i/p string : aaa

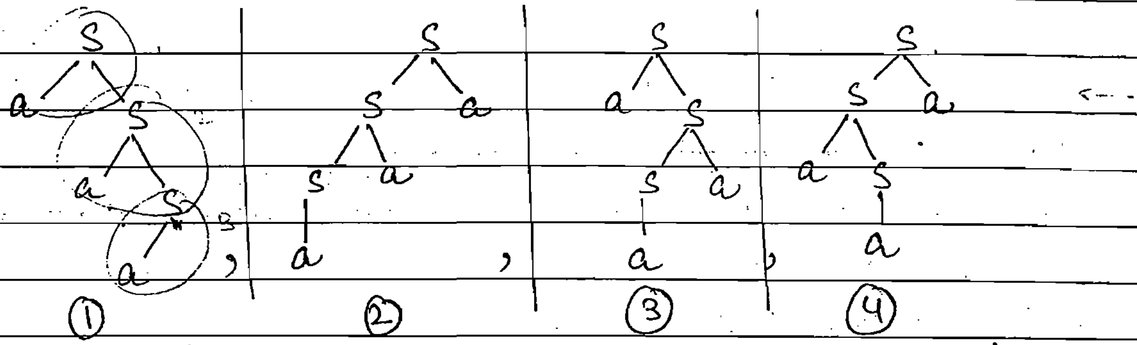
How many parse tree?

(Derivation tree or syntax tree)

Network Layer—
Transport Layer—
Page. No.
Date : / /

left most derivation → always explore left most variable first
Right most derivation → always explore Right most derivation first

string → aaa



→ To know, what string is generated → read all leaf nodes from left to right.

In above, left most derivation tree → 4
Right most derivation tree → 4

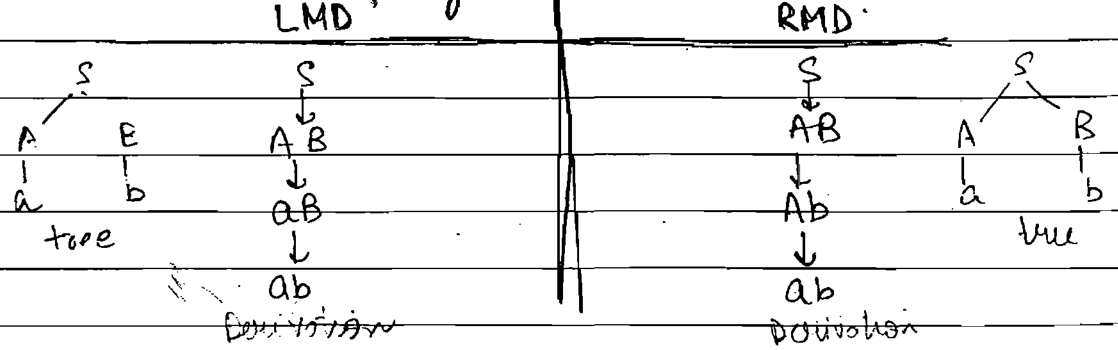
(even numbering too)

- In above example, only one variable, so LMDT and RMDT same
- If more than one variable, then check (can differ in numbering)

→ Always no. of left Most Derivation tree = no. of Right Most Derivation tree, only numbers changes (which may not even change sometimes)

No. of Left Most Derivation tree = No. of Right Most Derivation tree = No. of Parse tree

→ left most Derivation and Right most Derivation may differ, but their trees will always be same.



→ So, Derivation are differing, but both have same parse tree.

Page. No.
Date : / /

→ Above Grammar is Ambigious Grammar, because more than one Parse tree possible..

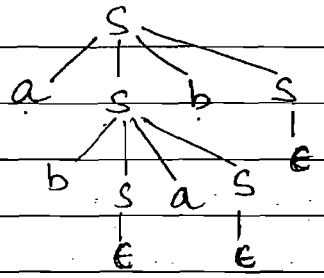
→ A Grammar G is said to be ambigious, if for atleast one string, more than one parse tree possible

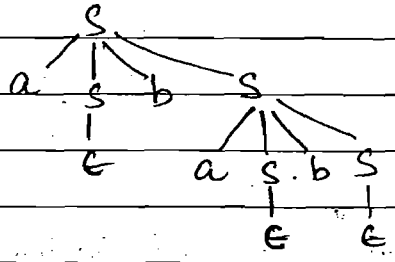
→ Checking given grammar is ~~ambigious grammar or not~~ undecidable, because ~~their is no algorithm~~ ambigious or not, is undecidable, beloz there is no algorithm to check it.

Q) Consider the following grammar

$S \rightarrow aSbS \mid bSaS \mid \epsilon$

i/p string → abab$



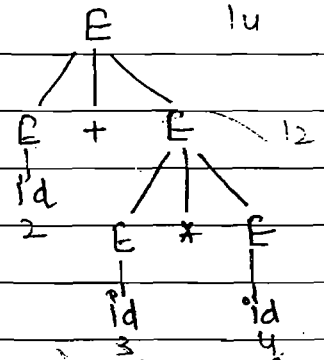


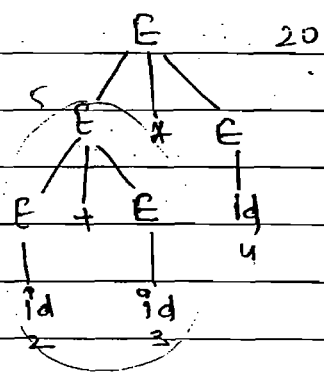
Above is Ambigious grammar.

Q) Consider the following grammar

$E \rightarrow id \mid E+E \mid E*E$

Input string: id+id*id
2+3*4





→ Ambigious Grammar.