Hindbookcenter

GET IT ON Google Play

ALL NOTES BOOKS AVAILABLEALL STUDY MATERIAL AVAILABLE

COURIERS SERVICE AVAILABLE

MADE EASY , IES MASTER , ACE ACADEMY , KREATRYX

ESE , GATE, PSUs 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

Shop No.7/8 Saidulajab Market Neb Sarai More, Saket, New Delhi-30	Shop No: 46 100 Futa M.G. Rd Near Made Easy Ghitorni, New Delhi-30	F518 Near Kali Maa Mandir Lado Sarai New Delhi-110030	Shop No.7/8 Saidulajab Market Neb Sarai More, Saket, New Delhi-30

Website: www.hindbookcenter.com

Contact Us: 9711475393

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, a} — 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

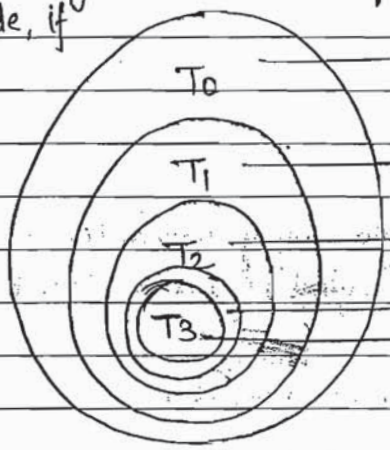
$A \longrightarrow T^* \beta / T^*$ (Right linear Grammar)

Page. No.
Date : / /

Ex → S → ABC
A → ab
B → CD

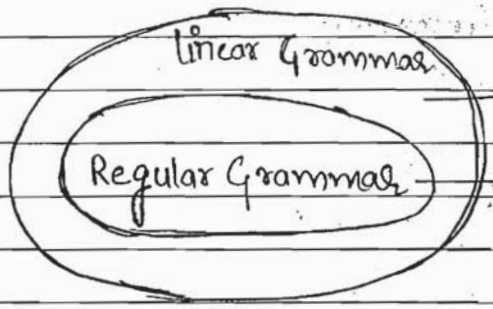
✓ T_2, ✗ T_3 (Right side)
CFG

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



Unrestricted grammar
context sensitive grammar
context free grammar
Linear Grammar
Regular grammar

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



contains both left and right linear
only one at a time

Context Free Grammar (CFL)

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

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

Page. No.
Date : / /

ϵ ⟶ string of length 0 (empty string)

$L = \{\}$ ⟶ Empty language

ϵ-DFA

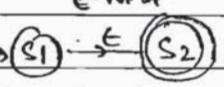
$L = \{\epsilon\}$ ⟶ 

language containing string of length 0

$L = \{\}$ ⟶ 

Empty language (no final state)

ϵ-NFA

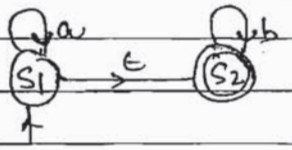


Transition function → DFA : $Q \times \Sigma \longrightarrow Q$ (any state × any input ⟶ goes to one of state)

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

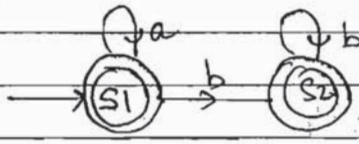
ϵ-NFA : $Q \times \Sigma \cup \{\epsilon\} \longrightarrow 2^Q$ (can go to any number of states; includes ϵ)

a^*b^* ⟶ (Valid-Y, ϵ)



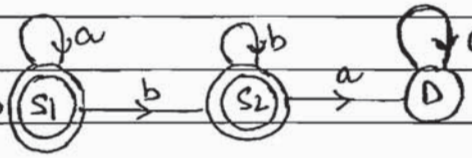
(ϵ-NFA)

(Valid-Y)



(NFA)

(Valid-Y, Invalid-N)



(DFA)

Dead or Trap state → Permanent Non-final states.
Non-final state ⟶ 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-X dfa)

Page. No.
Date : / /

② Give context free Grammar for Language

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

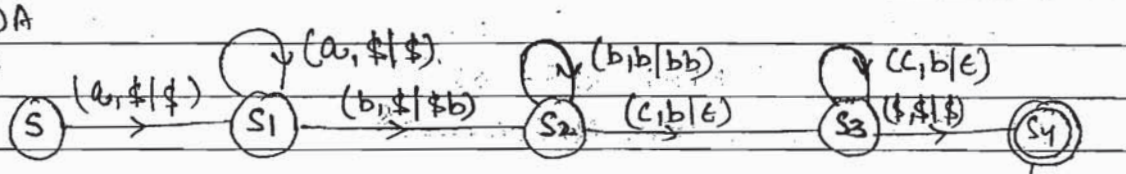
$S \longrightarrow AB$

$A \longrightarrow aA \mid a$

$B \longrightarrow bBC \mid bc$

PDA = Finite Automata + 1 Stack

DPDA



PDA → acceptance by final state
→ acceptance by empty stack

acceptance by final state

③ Give CFG for the language

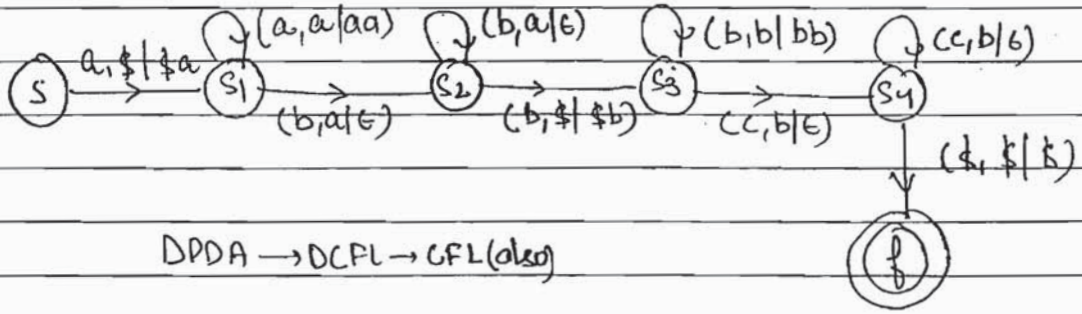
$L = \{ a^i b^{i+j} c^j \mid i, j \geq 1 \}$

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

$S \longrightarrow AB$

$A \longrightarrow aAb \mid ab$

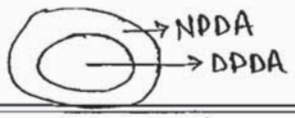
$B \longrightarrow bBC \mid bc$



DPDA → DCFL → CFL (also)

→ NFA is equivalent to DFA

→ NPDA has more power than DPDA.

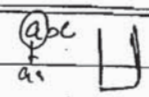


aabbcc

Page. No.

Date: / /

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



$S \longrightarrow S_1 / S_2$

$S_1 \longrightarrow AB$

$A \longrightarrow aAb / ab$

$B \longrightarrow cB / c$

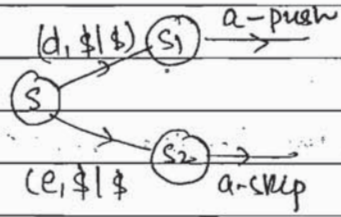
$S_2 \longrightarrow CD$

$C \longrightarrow aC / a$

$D \longrightarrow bDc / bc$

NPDA → CFL language

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



→ no ambiguity

→ DPDA

(5) CFL 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. $L = \{ a^m b^m c^n \cup a^m b^n c^n \mid m, n \geq 1 \}$

DCFL CCFL

Union

CFL (here in this example)

But can be possible with some example or not.

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

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

⊙ 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 expressions over the α, β, id.

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

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

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

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

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

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 ($\Sigma = \{a, b\}$).

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

$(a+b)^*$

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

ϵ

$a^* b^*$

$R*$

$(R+R)^*$

$(a+b)^*$

$a(a+b)^*$

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

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

⑨ Consider the following Grammar

$S \rightarrow as/sa/a$

i/p string : aaa

How many parse tree?
(Derivation tree or syntax tree)