UNIT-III

Subject-Theory of Computation

Prof. Shweta Tiwaskar

Shweta.tiwaskar@viit.ac.in

Department of Computer Engineering

BRACT'S, Vishwakarma Institute of Information Technology, Pune-48

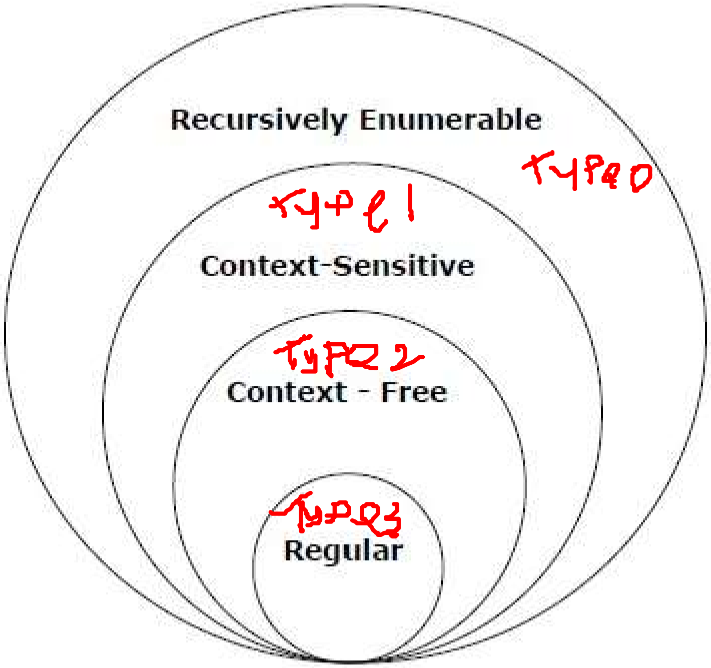
(An Autonomous Institute affiliated to Savitribai Phule Pune University)

(NBA and NAAC accredited, ISO 9001:2015 certified)

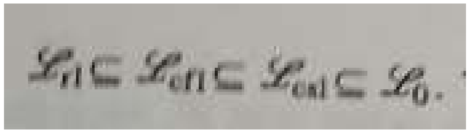
Chomsky Hierarchy of Grammar

Grammar Type	Grammar	Language accepted	Automaton
Type 0	Unrestricted grammar	Recursively enumerable language	Turing Machine (TM)
Type1	Context Sensitive Grammar(CSG)	Context Sensitive Language(CSL)	Linear Bounded Automata(LBA)
TYPE 2	Context Free Grammar(CFG)	Context Free Language(CFL)	Push down Automata(PDA)
TYPE 3	Regular Grammar	Regular Language	Finite Automata

Chomsky Hierarchy of languages



Chomsky Hierarchy of languages



$$\mathscr{L}_{rl} \subseteq \mathscr{L}_{cfl} \subseteq \mathscr{L}_{csl} \subseteq \mathscr{L}_0 .$$

1. Regular language is a subset of Context free language(CFL).
2. CFL is a subset of Context Sensitive Language(CSL)
3. CSL is a subset of type 0 language.

Type 0

- In Type-0 grammars include all formal grammars
- Type 0 grammar language are recognized by turing machine.
- These languages are also known as the Recursively Enumerable languages.

e.g $A ab \rightarrow b c$

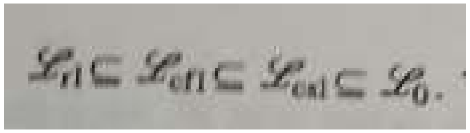
$\alpha \rightarrow \beta$

$|\alpha| > |\beta|$

Type 1: Context Sensitive Grammar)

- Type-1 grammars generate the context-sensitive languages. The language generated by the grammar are recognized by
  the Linear Bound Automata
  In Type 1
  I. First of all Type 1 grammar should be Type 0.
  II. Grammar Production in the form of

TYPE1 (CSG) Grammar

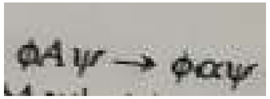

$\mathcal{L}_{rl} \subseteq \mathcal{L}_{cfl} \subseteq \mathcal{L}_{csl} \subseteq \mathcal{L}_0$.

✓ $\phi A \psi \rightarrow \phi \alpha \psi$

$=$

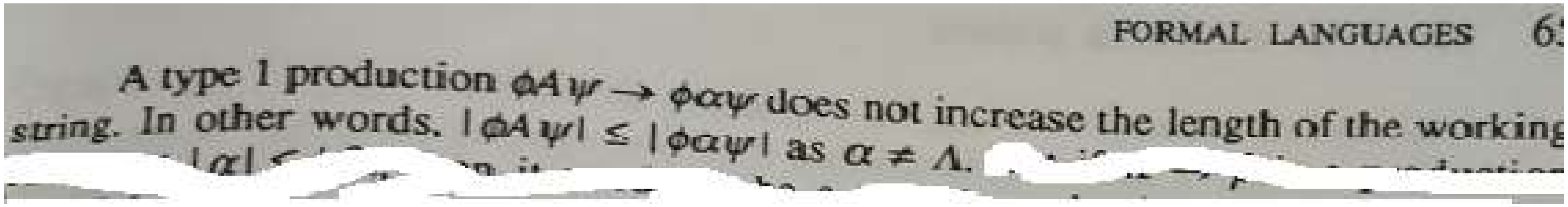
$abAB = ab\alpha B$

Type1 Grammar or Context Sensitive Grammar(CSG)

In CSG all productions are of the form  $\phi A\psi \rightarrow \phi\alpha\psi$

ϕ → lef context & , ψ → right context

ab B


FORMAL LANGUAGES 65

A type 1 production $\phi A\psi \rightarrow \phi\alpha\psi$ does not increase the length of the working string. In other words, $|\phi A\psi| \le |\phi\alpha\psi|$ as $\alpha \neq \Lambda$.

$\phi A \psi \rightarrow \phi \alpha \psi$

- IN CSG, we can Replace NON terminal 'A' by 'α' only when it appears between the left context ϕ and right context ψ.

wh

Where 'ϕ' & 'ψ' are context denoting string.

TYPE 2(CFG) Grammar

- In CFG, context denoting 'ϕ' & 'ψ'

are both required to be empty

$\underline{\phi A \psi} \rightarrow \underline{\phi \omega \psi}$

ϕ - Left context

ψ Right Context

are both required to be empty. Thus the possibility of repeating a non- terminal symbol is independent of the adjacent variable /symbol i.e it is independent of the context.

If G is CFG then L(G) is called CFL.

Type 2: Context Free Grammar:

- Type-2 grammars generate the context-free languages.
- In Conte
- The language generated by the grammar is recognized by a Pushdown automata
- Type-2 grammars generate the context-free languages.
  1. First of all it should be Type 1.
  2. Left hand side of production can have

- E.g For example,
  S –> AB
  A –> a
  B –> b

$\alpha \to \beta$

$|\alpha| \leq |\beta|$

α – L.H.S Prod^n

β – R.H.S Prod^n

Type 3: Regular Grammar:

Type-3 grammars generate regular languages. These languages are exactly all languages that can be accepted by a finite state automaton.

Type - 3 Grammar

- **Type-3 grammars** generate regular languages. Type-3 grammars must have a single non-terminal on the left-hand side and a right-hand side consisting of a single terminal or single terminal followed by a single non-terminal.
- The productions must be in the form **X → a or X → aY**

$A \rightarrow aB$ — right linear

$A \rightarrow Ba$ — left linear

- Regular grammar are of two types

1. left linear 2. right linear

- Left linear grammar: if the nonterminal appears to the left of terminal.

e.g. A->Ba

- Right linear grammar: If the nonterminal appears to the right of terminal

e.g. A->aB