ANDIATING EDUBE DAUDA
BHU/20/04/05/0052
COMPUTER SCIENCE
CMP 313

According to Chomsky hierarchy, grammar is divided into 4 types as follows:-

1. Type 0 grammar also known as unrestricted grammar
2. Type 1 grammar also refered to as context-sensitive grandmar
3. Type 2 grammar also refered to as a context-free-grammar
4. Type 3 grammar known to be Regular grammar.

(1.) Type 0: Unrestricted Grammar

It includes all formal grammar. Type 0 grammar languages are recognized by turing machine (TM). These languages are as well known as Recursively Enumerable Languages.

Grammar Production in the form of $\alpha \longrightarrow \beta$

where;

\alpha, is $(V+T)^* V (V+T)^*$

V: Variables

T: Terminals.

β is $(V+T)^*$.

grammars

In type 0, there must be at least one variable on the Left side of production.

Examples:-

① $Sab \rightarrow ba$

$A \rightarrow S$

Here, Variables are S & A while Terminals are a & b. There is no restriction on the grammar rules of Type 0 grammars.

Eg 2:- $bAa \rightarrow aa$

$S \rightarrow S$

where A, S are Variables and a, b and s are Terminals.

② Type 1: Context-Sensitive Grammar

Type 1 grammars generate context-sensitive languages. The languages generated by the grammar are recognized as the Linear Bound Automata (LBA).

In Type 1 :

- Firstly, all Type 1 grammars should be Type 0.
- Grammar production in the form of $\alpha \rightarrow \beta$

$|Alpha| <= |beta|$

Meaning, the count of symbol α is less than or equal to β.

$\beta \in (V + T)^+$

i.e β cannot be ε.

③

Example;-

$S \rightarrow AB$
$AB \rightarrow abc$
$B \rightarrow b$

③ Type 2; Context-free Grammar: Type-2 grammars generate context-free languages. The language generated by the grammar is recognized by a Pushdown automata.

For Type 2 grammar:
- It should be Type 1
- The left-hand side of production can have only one variable and there is no restriction on B. $|\alpha| = 1$.

Example;
① $S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow b$

② $A \rightarrow aBb$
$A \rightarrow b$
$B \rightarrow a$

The production rule is of this form; $A \rightarrow \alpha$ where A is any single non-terminal and is any combination of terminals and non-terminals.

④ Type 3: Regular Grammars:- These grammars generate regular languages. They are exactly all languages that can be accepted by a finite state automaton. Type 3 grammars is the most restricted form of grammar.

Type 3 should be in the given form only:-

$V \rightarrow VT/T$ (left-regular grammars)

or

$V \rightarrow TV/T$ (right-regular grammars)

Example; $S \rightarrow a$.) regular

The above is called strictly grammars

The second form of regular grammars is called extended regular grammars.

It has this form,:-

$V \rightarrow VT^*/T^*$ (extended left-regular grammars)

or

$V \rightarrow T^*V/T^*$ (extended right-regular grammars)

Example,:-

$S \rightarrow ab$

Name: Atiku Iorfa Raphael
Dept: Computer Science
Mat No: BTH/2010/105/0067
C.C: Comp 313: Automata Theory

Type 0: Include all formal grammars. Its grammar is Unrestricted. Type 0 generates recusively enumerable languages
Productions can be in form of $\alpha \longrightarrow \beta$
Where α - string of terminals & non-terminals with at least 1 non-terminal and α cannot be null
β - string of terminal and non-terminal
Example:
$S \longrightarrow ACaB$
$Bc \longrightarrow aCB$
$cB \longrightarrow DB$
$aD \longrightarrow Db$

Type 1: generates Context-Sensitive languages: productions must be in the form. $\alpha A \beta \longrightarrow \alpha V \beta$
where $A \in N$ (i.e A are non-terminal)
$\alpha \beta, V \in (T \cup N)$ (i.e α, β, V are strings of terminal and non-terminals)
Strings α and β may be empty, but V must be non-empty

Examples
$AB \longrightarrow Ab\,Bc$
$A \longrightarrow bcA$
$B \longrightarrow b$

Type 2: generate Context-free languages. The production must be in the form. $A \longrightarrow V$
Where A is non-terminal
and V is a string of terminals & non-terminals

Name: Abiso Iorfa Raphael
Dept: Computer Science
Mat No: Bku/2016/105/0067
C.C: Comp 313: Automata Theory

Type 0: include all formal grammars. Its grammar is unrestricted. Type-0 generates recursively enumerable languages
Productions can be in form of $\alpha \rightarrow \beta$
Where α - string of terminals & non-terminals with at least 1 non-terminal and α cannot be null
β - string of terminal and non-terminal
Example
$S \rightarrow ACaB$
$Bc \rightarrow aCB$
$cB \rightarrow DB$
$aD \rightarrow Db$

Type 1: generates Context-Sensitive Languages: production must be in the form. $\alpha A \beta \rightarrow \alpha V \beta$
where $A \in N$ (i.e A are non-terminal)
& $\alpha, \beta, V \in (T \cup N)$ (i.e α, β, V are strings of terminal and non-terminals)
Strings α and β may be empty, but V must be non-empty

Examples
$AB \rightarrow Ab\,Bc$
$A \rightarrow bcA$
$B \rightarrow b$

Type 2: generate Context-free Languages. The production must be in the form: $A \rightarrow V$
Where A is non-terminal
and V is a string of terminals & non-terminals

Example

$S \longrightarrow X_a$

$X \rightarrow a$

$X \rightarrow aX$

$X \longrightarrow abc$

$X \rightarrow \epsilon$

Type 3: generates regular (anagrammars). The production must be in the form.

$$X \longrightarrow a \text{ OR}$$

$$X \longrightarrow aY$$

∴ Type 3 grammar must have a single non-terminal on the left hand & right ~~hand~~ having a single terminal of a single terminal followed by a non-terminal.