UNIT-III: Grammar Formalism

Objective:

To understand regular grammars and context free grammars.

Syllabus:

Grammars, Chomsky hierarchy of languages, regular grammars - right linear and left linear grammars, equivalence between regular linear grammar and FA, Contextfree grammars, derivation trees, sentential forms, rightmost and leftmost derivation of strings, ambiguity in context free grammars, Minimization of context free grammars.

Learning Outcomes:

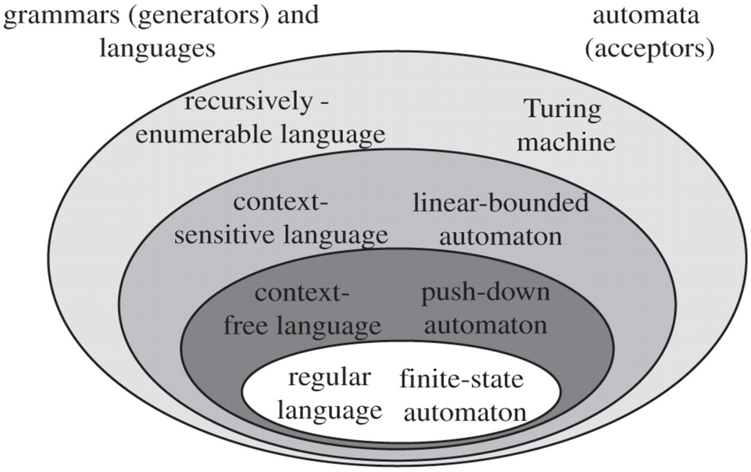
Students will be able to:

- understand Chomsky hierarchy of languages.
- understand and construct the regular grammar for the given regular language or regular expression.
- convert Regular Grammar into equivalent DFA and viceversa.
- construct Context free grammar for the given language.
- construct right most, left most derivation and derivation trees for the given string and grammar.
- Ambiguity and Minimization of Context free grammar

Learning Material

3.1 Chomsky hierarchy of languages:

The four classes of languages are often called the Chomsky hierarchy, after Noam Chomsky, who defined these classes as potential models of natural languages.



the traditional Chomsky hierarchy

Fig:4.1 **Chomsky hierarchy**

Chomsky classifies the grammar into four types:

Table:4.1Types of Grammar

Grammar	**Languages**	**Automaton**	**Production rules**
Type 0	Recursively enumerable/ Phrase Structured	Turing machines	$\alpha \rightarrow \beta$
Type 1	Context-sensitive	Linear-bound automata	$\alpha \rightarrow \beta$ $\|\alpha\| <= \|\beta\|$
Type 2	Context-free	Push-down automata	$A \rightarrow \alpha$
Type 3	Regular	Finite-state automata	A → w A → wB A → Bw

3.1.1Regular Grammar:

A right- or left-linear grammar is called a regular grammar.

Right-Linear Grammar:

If all productions of a grammar are of the form A → wB or A → w, where A and B are variables and w is a (possibly empty) string of terminals, then we say the grammar is right-linear.

Example:

Represent the language 0(10)* by the right-linear grammar.

The language generated by the given Regular Expression is

L = {0, 010, 01010, 0101010, …….}

Right-Linear Grammar:

S→0A

A→10A | ε

Left-Linear Grammar:

If all productions are of the form A → Bw or A → w, we call it left-linear.

Example:

Represent the language 0(10)* by the left-linear grammar.

The language generated by the given Regular Expression is

L = {0, 010, 01010, 0101010, …….}

Left-Linear Grammar:

S→S10 | 0

Equivalence of regular grammars and finite automata:

A language is regular if and only if it has a left-linear grammar and if and only if it has a right-linear grammar.

Construction of a Regular Grammar for a given DFA:

Let M = ({q0, q1... qn} **, ∑, δ, q_o, F)**. We construct G as G = ({A0, A1, ..., An},∑**, P, A0)**

where P is defined by the following rules:

(i) Ai → aAj is included in P if **δ**(qi, a) = qj ∉ F.

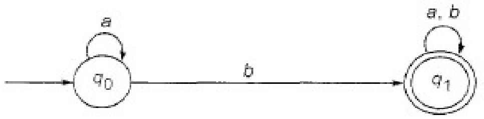
(ii) Ai → aAj and Ai → a are included in P if **δ**(qi, a) = qj ∈ F.

Note: We can construct only right linear grammar for the given DFA.

If we want to construct **left linear grammar** for the given DFA, reverse the edges of the given DFA and interchange initial and final states.

Example:

1. ***Construct regular grammar (right linear grammar) for the given DFA.***



Given M= ({q0,q1}, {a,b}, **δ, q_o, {q1})**

Construct G= ({A0,A1}, {a,b} ,P, A0) where *P* is given by

(i) Ai → aAj is included in P if δ(qi, a) = qj ∉ F.

δ(q0, a) = q0 ∉ F ⇒ A0→aA0

(ii) Ai → aAj and Ai → a are included in P if δ(qi, a) = qj ∈ F.

δ(q0, b) = q1∈ F ⇒ A0→bA1 and A0→b

δ(q1, a) = q1∈ F ⇒ A1→aA1 and A1→a

δ(q1, b) = q1∈ F ⇒ A1→bA1 and A1→b

∴ P is given by

A0→aA0, A0→bA1, A0→b

A1→aA1, A1→a, A1→bA1, A1→b

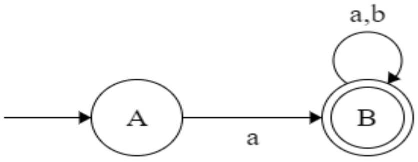
Steps to convert Finite Automata to Left Linear Grammar:

Step 1: Reverse all the edges of the given automata and interchange initial state and final states.

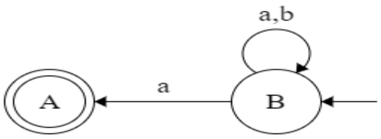
Step 2: Represent the productions using Left Linear Grammar.

Example:

2. ***Construct left linear grammar for the given DFA.***



Step 1: Reverse all the edges of the given automata and interchange initial state and final states.



Step 2: Represent the productions using Left Linear Grammar.

B→Ba B→Aa
B→Bb B→a

3.1.2 Construction of a DFA for a given Regular Grammar:

Let G = **({A0, A1, …., An},∑, P, A0).** We construct a DFA M whose

(i) states correspond to variables.
(ii) initial state corresponds to A0.
(iii) transitions in M correspond to productions in P. As the last production applied in any derivation is of the form Ai→ a, the corresponding transition terminates at a new state, and this is the unique final state.

We define M as **({q0, q1… qn, qf} , ∑, δ, q₀, {qf})** where **δ** is defined as follows:

(i) Each production Ai→aAj induces a transition from qi to qj with label a,

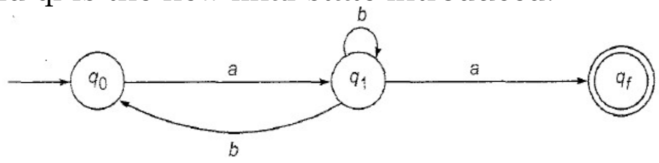
(ii) Each production Ak →a induces a transition from qk to qf with label a.

Example:

1. G= ({A0, A1}, {a,b} ,P, A0) where P consists of A0→aA1, A1→bA1, A1→a, A1→bA0. Construct a DFA M accepting L(G).

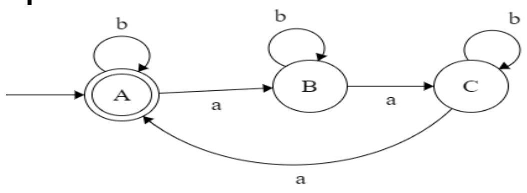
A0→aA1 induces a transition from q0 to q1 with label a.
A1→ bA1 induces a transition from q1 to q1 with label b.
A1→bA0 induces a transition from q1 to q0 with label b.
A1→a induces a transition from q1 to qf with label a.

M = **({q0, q1, qf} , ∑, δ, q₀, {qf}),** where q0 and qf correspond to A0 and A1 respectively and qf is the new final state introduced.



2. Construct Finite Automata for the grammar which consists of the productions

A→ aB | bA | b
B→ aC | bB
C→ aA | bC | a



<u>3.2 Context-Free Grammar</u>:

A context-free grammar (CFG or just grammar) is denoted G = (V, T, P, S), where

- V and T are finite sets of variables and terminals, respectively.
- P is a finite set of productions; each production is of the form A → α, where A is a variable and α is a string of symbols from $(V \cup T)^*$.
- S is a special variable called the start symbol.

The language generated by G [denoted L(G)] is {w | w is in T* and $S \overset{*}{\underset{G}{\Rightarrow}} w$}.
That is, a string is in L(G) if:
1) The string consists solely of terminals.
2) The string can be derived from S.
We call L a context-free language (CFL) if it is L(G) for some CFG G.

***Note:* C language is an example for Context Free Language.**

Examples:

1. Write CFG for the language L= $\{a^n b^n \mid n>=1\}$.

   L= {ab, aabb, aaabbb, aaaabbbb, aaaaabbbbb, …………}
   G = ({S}, {a, b}, P, S)
   P: S -> aSb | ab
   (Or)
   S -> aSB
   S -> aB
   B -> b

2. Write CFG for the language L= $\{a^n b^m \mid n , m >=1\}$.

   L= {a, b, ab, aab, abb,aabb, aaabbb, aaaabbbb, aaaaabbbbb,……………………..}
   G = ({S, A, B}, {a, b}, P, S)
   P: S -> AB

A -> aA | a
B -> bB | b

3. Write CFG for the language L={aa,ab,ba,bb}
G = ({S, A}, {a, b}, P, S)
P: S -> AA
A -> a | b

4. Write CFG for the language L= { a^n | n>=0}.
L= { ε, a, aa, aaa, aaaa, aaaaa, aaaaaa,………………………}
G = ({A}, {a}, P, A)
P: A -> aA | ε

5. Write CFG for the regular expression (a+b)*.
L= { ε, a, b, aa, ab , ba, bb, aaa,abb,aba,………………………}
G = ({S}, {a, b}, P, S)
P: S -> aS | bS | ε

6. Write CFG to generate all strings of {a, b} whose length is atleast 2.
L= { aa, ab , ba, bb, aaa,abb,aba,………………………}
G = ({S, A, B}, {a, b}, P, S)
P: S -> AAB
A -> a | b
B -> aB | bB | ε

7. Write CFG to generate all strings of {a, b} whose length is atmost 2.
L= { ε , a,b, aa, ab , ba, bb}
G = ({S, A}, {a, b}, P, S)
P: S -> AA
A -> a | b | ε

8. Write CFG to generate palindromes over {a, b}.
L= { ε , a,b, aa,bb,aba,bab,aaaa,abba,………………}
G = ({S}, {a, b}, P, S)
P: S -> aSa | bSb
S -> a | b | ε

9. Write CFG to generate equal number of a's and b's.
L= { ab, ba,aabb, abab, bbaa,baba,………………………}
G = (V, T, P, S), where V = {S, A, B}, T = {a, b},S and P .
P: S -> aB   A -> bAA
S -> bA   B ->b
A ->a   B ->bS

A ->aS   B -> aBB

<u>Sentential Form</u>:

A string of terminals and variables α is called a sentential form if $S \overset{*}{\Rightarrow} \alpha$.

<u>Derivation</u>:

Derivation is the process of applying productions repeatedly to expand non-terminals in terms of terminals or non-terminals, until there are no more non-terminals.

A derivation can be either Leftmost derivation or Right most derivation.

<u>Leftmost derivation</u>:

If at each step in a derivation a production is applied to the leftmost variable, then the derivation is said to be leftmost.

Example:

Consider the grammar G = ({S, A}, {a, b}, P, S), where P consists of

S →aAS | a
A→ SbA|SS|ba

The corresponding leftmost derivation is

S => aAS => aSbAS => aabAS => aabbaS => aabbaa.

<u>Rightmost derivation</u>:

A derivation in which the rightmost variable is replaced at each step is said to be rightmost.

Example:

Consider the grammar G = ({S, A}, {a, b}, P, S), where P consists of

S →aAS | a
A→ SbA|SS|ba

The corresponding rightmost derivation is

S => aAS => aAa => aSbAa => aSbbaa => aabbaa.

Note:“If w is in L(G) for CFG G, then w has at least one parse tree, and corresponding to a particular parse tree, w has a unique leftmost and a unique rightmost derivation.”

<u>3.3 Derivation Trees (or) Parse tree</u>:

The derivations in a CFG can be represented using trees. Such trees representing derivations are called derivation trees.

Let **G = (V, T, P, S)** be a CFG. A tree is a derivation (or parse) tree for G if:

1) Every vertex has a label, which is a symbol of V ∪ T ∪ {ε}.
2) The label of the root is S.
3) If a vertex is interior and has label A, then A must be in V.

4) If n has label A and vertices n1, n2, n3, ..., nk are the sons of vertex n, in order from the left, with labels X1, X2, ..., Xk, respectively, then A→X1X2 ....Xk must be a production in P.

5) If vertex n has label ε, then n is a leaf and is the only son of its father.

Example:

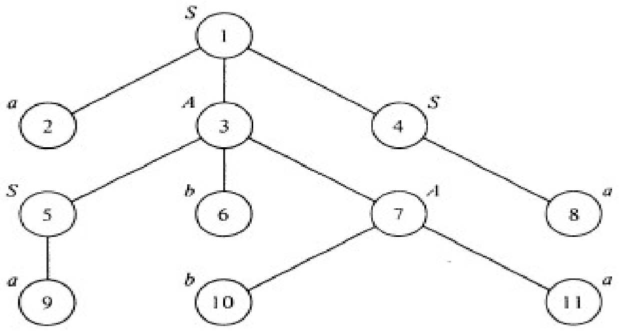
Consider the grammar G = ({S, A}, {a, b], P, S), where P consists of

S →aAS | a

A→ SbA|SS|ba

Construct a derivation tree for the string "aabbaa"

A derivation tree is a natural description of the derivation of a particular sentential form of the grammar G. If we read the labels of the leaves from left to right, we have a sentential form. We call this string the yield of the derivation tree.



S => aAS => aSbAS => aabAS => aabbaS => aabbaa.

Note: Some leaves could be labelled by ε.

3.4 Ambiguity in context free grammars:

A context-free grammar G is said to be ambiguous if it has two parse trees for some word.

(or)

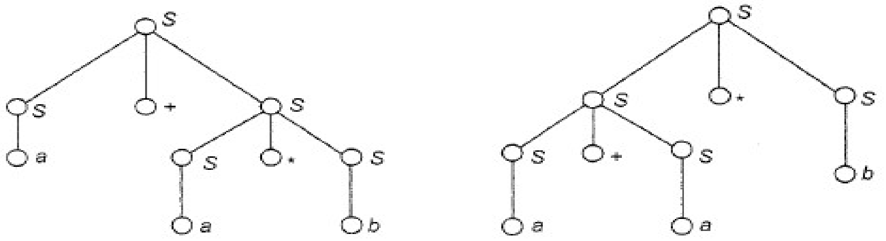
A word which has more than one leftmost derivation or more than one rightmost derivation is said to be ambiguous.

Note: A CFL for which every CFG is ambiguous is said to be an inherently ambiguous CFL.

Example:

G = *({S}, {a, b,* +, *}, *P. S),* where *P* consists of S→S+S | S*S | a | b

We have two derivation trees for *a* + *a* * *b*



Two derivation trees for a + a * *b*

3.5 Minimization of Context Free Grammars:

1) Elimination of useless symbols.

2) Elimination of ε –Productions.

3) Elimination of Unit Productions.

Elimination of Useless Symbols:

Let G=(V, T, P, S) be a grammar. A symbol X is useless if it is not involved in derivation.

(or)

A symbol X is useless if there is no way of getting a terminal string from it.

Example:
Consider the grammar

S→AB | a
A→ a

We find that no terminal string is derivable from B. We therefore eliminate B and the production S → AB.
Then the grammar is

S→a
A→a

We find that only S and a appear in sentential forms. Thus ({S}, {a}, {S → a}, S) is an
equivalent grammar with no useless symbols.

Elimination of ε –Productions:

A production of the form *A* → ε, where *A* is a variable, is called a *null production.*
If L = L(G) for some CFG G = (V, T, P, S), then L - { ε } is L(G') for a CFG G' with no useless symbols or ε -productions.

Example:

Consider the grammar

$A \rightarrow 0B1 \mid 1B1$

$B \rightarrow 0B \mid 1B \mid \varepsilon$

Remove ε-productions from the grammar.

B→ ε is the null production.

The new productions after elimination of ε are

$A \rightarrow 0B1 \mid 1B1 \mid 01 \mid 11$

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

<u>Elimination of Unit Productions</u>:

A production of the form A→B whose right-hand side consists of a single variable is called a unit production.

All other productions, including those of the form A →a and ε -productions, are nonunit productions.

Example:

Consider the grammar

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

$A \rightarrow 0S \mid 00$

$B \rightarrow 1 \mid A$

$C \rightarrow 01$

Remove unit production from the grammar.

S→C and B→A are the unit productions

The new productions after elimination of unit productions are

$S \rightarrow 0A \mid 1B \mid 01$

$A \rightarrow 0S \mid 00$

$B \rightarrow 1 \mid 0S \mid 00$

$C \rightarrow 01$

C is a useless symbol. So eliminate C production.

The final set of productions are

$S \rightarrow 0A \mid 1B \mid 01$

$A \rightarrow 0S \mid 00$

$B \rightarrow 1 \mid 0S \mid 00$