Unit-3

Regular grammar & context free grammar.

Grammar:-

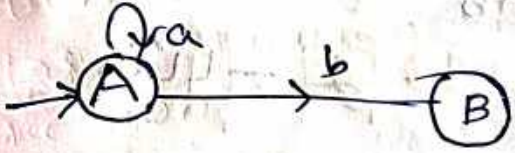
It is the set that contains 4 tuples like $G = (V, T, P, S)$

V = variables

T = Terminals

P = set of production rules

S = start Symbol with which Strings on grammar are divided



$V = \{A, B\}$

$T = \{a, b\}$

$S = \{A\}$

$P = A \rightarrow aA$

$A \rightarrow bB$

outgoing edges

$[A \rightarrow aA, A \rightarrow bB]$

⇓ Grammar

Left Linear Grammar (LLG):-

$A \rightarrow Aa$

$A \rightarrow a$

Variable is left side of the terminal.

Right Linear Grammar (RLG):-

Variable is Right side to the terminal

$A \rightarrow aA$

$A \rightarrow a$

conversion of

To overcome compilation errors we use Linear Grammars.

Rules:-

(i) Design Finite-Automata
(ii) Interchange the initial and final state
(iii) Reverse the directions
(iv)

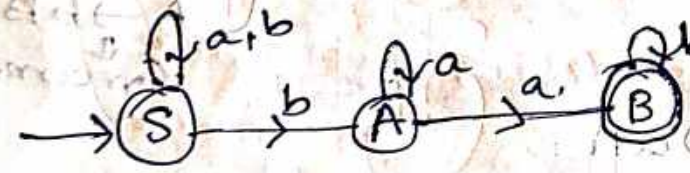
Ex:-1. convert following LLG to RLG.

$S \rightarrow Sa|Sb|Ab$

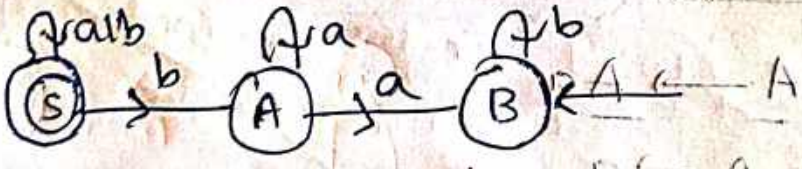
$A \rightarrow Aa|Ba$

$B \rightarrow Bb|\epsilon$

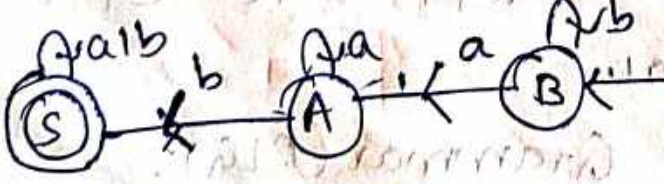
(i) Design a finite Automata



(ii) Interchange initial and final states.



(iii) Reverse the directions.



RLG

$S \to aS \mid bS$ — final state.

$B \to bB \mid aA$

$A \to aA \mid bS$

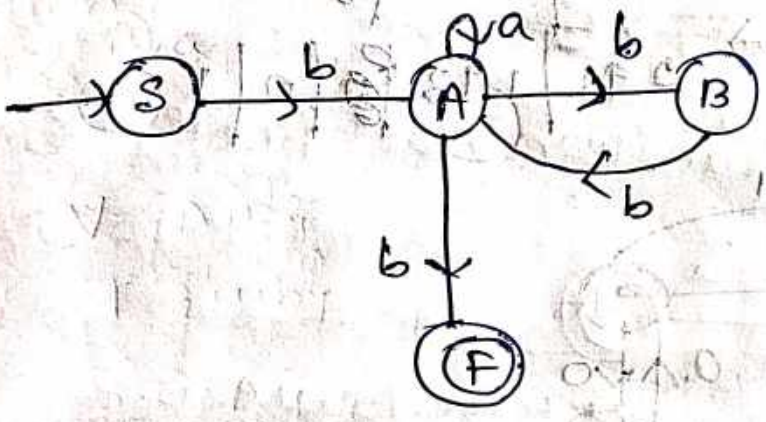
EX-2 (RLG — LLG)

$S \to bA$

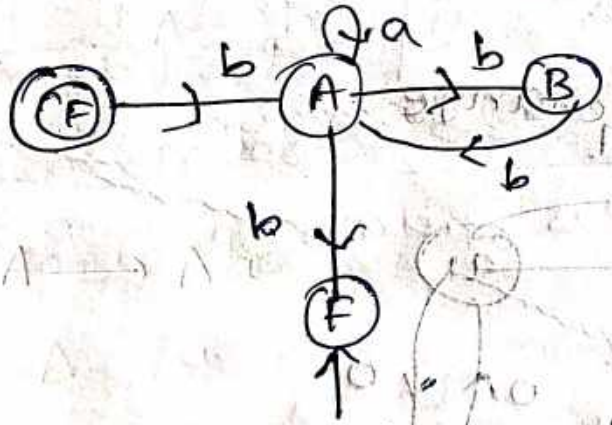
$A \to aA \mid bB \mid b$

$B \to bA$

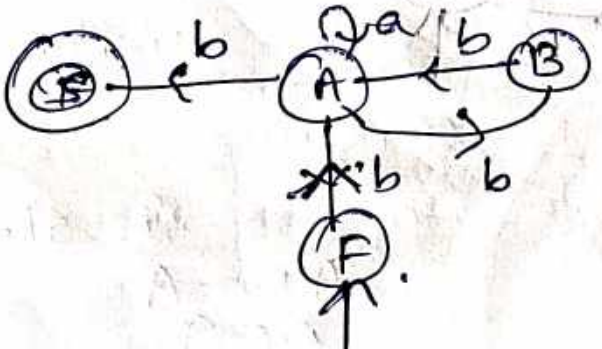
(i) Design Finite Automata



(ii) Interchang initial & final states



(iii) reverse directions



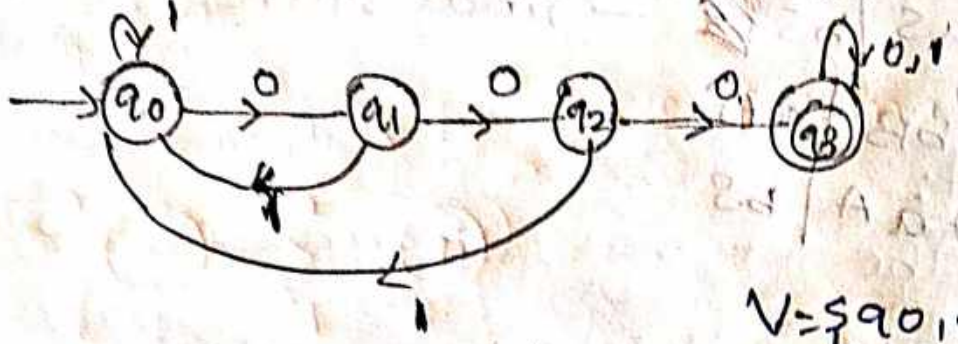
$S \to$

$A \to Aa \mid Bb \mid b$

$B \to Ab$

$P \to Ab$

(iii) Find the RLG for the DFA



$V = \{q_0, q_1, q_2, q_3\}$

$T = \{0, 1\}$

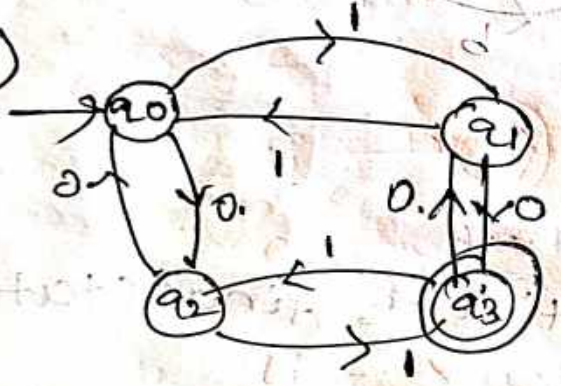
$q_0 \rightarrow 1\,q_0 \mid 0\,q_1$

$q_1 \rightarrow 0\,q_2 \mid 1\,q_0$

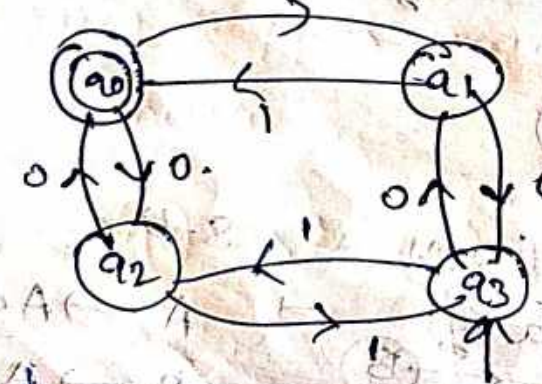
$q_2 \rightarrow 0\,q_3 \mid 1\,q_0 \mid 0$

$q_3 \rightarrow 0\,q_3 \mid 1\,q_3 \mid 0 \mid 1$

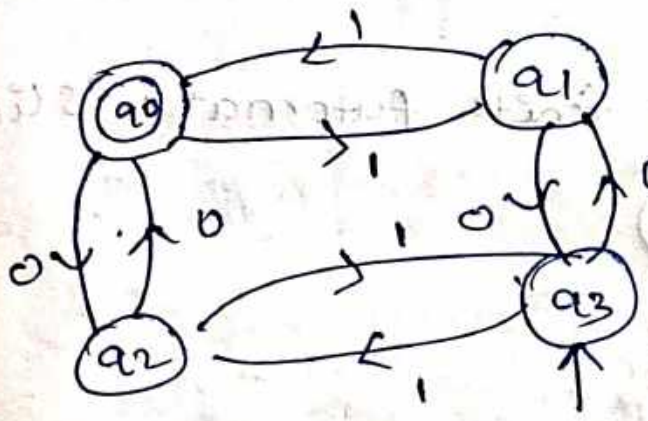
Q)



q_0 interchange



reverse direction



RLG

$q_3 \rightarrow 1q_2 | 0q_1$

$q_2 \rightarrow 1q_3 | 0q_0 | 0$

$q_1 \rightarrow 0q_3 | 1q_0 | 1$

$q_0 \rightarrow 1q_1 | 0q_2$

LLG

$q_3 \rightarrow q_1 \cdot 0 | q_2 \cdot 1$

$q_2 \rightarrow q_3 \cdot 1 | q_0 \cdot 0 | \not{\varepsilon} 1$

$q_1 \rightarrow q_3 \cdot 0 | q_0 \cdot 1 | \not{\varepsilon} 0$

$q_0 \rightarrow q_1 \cdot 1 | q_2 \cdot 0$

Conversion from finite Automata to Regular grammars:-

Let $M = \{Q, \Sigma, \delta, q_0, F\}$ be a finite Automata

It contains n no. of states like $Q = \{q_0, q_1, q_2, q_3 \cdots q_n\}$

$\Sigma = \{a_1, a_2, \cdots, a_n\}$

Therefore, the regular grammar $G = \{V, T, P, S\}$ is defined as

$V = \{q_0, q_1, q_2, q_3, \cdots q_n\}$

$T = \{a_1, a_2, a_3, \cdots a_n\}$

$S = q_0$

P = Transitions of Finite Automata.

Rules:-

(i) ~~Let~~ Gr

(i) If the transition of finite Automata is like

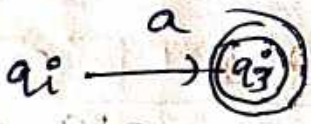
$q_i \longrightarrow a \cdot q_j$

(ii) If there is a state on F.A like

then the production rule is

$q_i \longrightarrow \epsilon$

(iii) If the transition of finite Automata is like



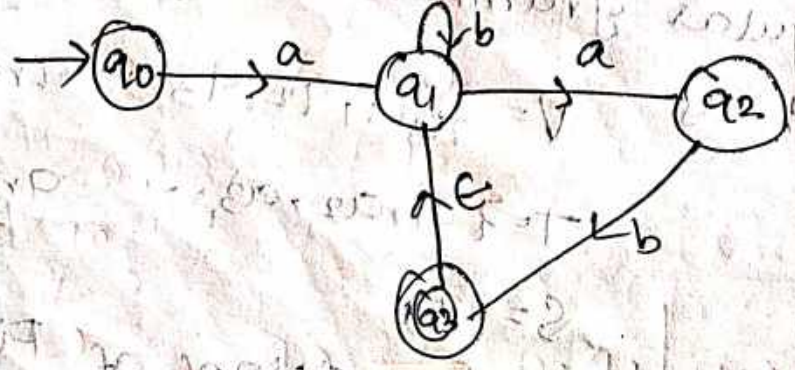
then production rule is.

$q_i \longrightarrow a \cdot q_j \mid a$ (or) $q_i \longrightarrow a \cdot q_j$

$q_i \longrightarrow a$

Example:-

Construct regular grammar from given finite Automata



$q_0 \longrightarrow a \cdot q_1$

$q_1 \longrightarrow \lambda$

Regular grammar:-

(i) $M = \{Q, \Sigma, \delta, q_0, F\}$ [TUPLES]

$Q = \{q_0, q_1, q_2, q_3\}$.

$\Sigma = \{a, b\}$

$q_0 = q_0$ (Initial state)

$F = q_3$ (Final state)

(ii) Transition table

	a	b	ϵ
q_0	q_1	ϕ	ϕ
q_1	q_2	q_1	ϕ
q_2	ϕ	q_3	ϕ
q_3	ϕ	ϕ	q_1

(iii) RG

$\{V, T, P, S\}$

$V = \{q_0, q_1, q_2, q_3\}$

$T = \{a, b\}$

$S = q_0$

P ⇒ $q_0 \longrightarrow a \cdot q_1$

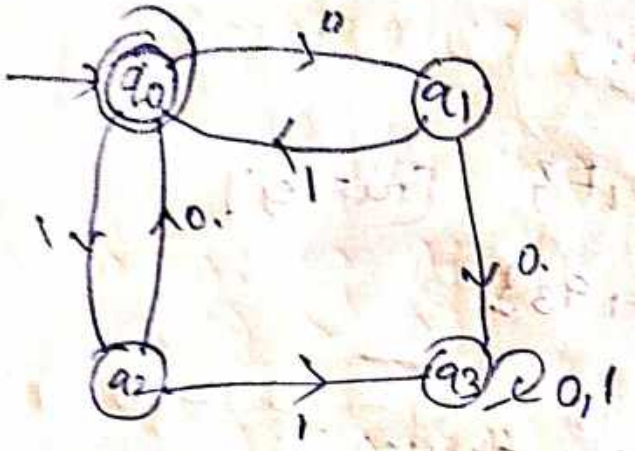
$q_1 \longrightarrow a \cdot q_2 \mid b \cdot q_1$

$q_2 \longrightarrow b \cdot q_3 \mid b$

$q_3 \longrightarrow \epsilon \cdot q_1$

[Write RLG]

Q) conversion of regular grammar from finite Automata.



$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$

$q_0 = q_0$

$F = q_0$

Transition table

	0	1
q_0	q_1	q_2
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_3	q_3

RG

$V = \{q_0, q_1, q_2, q_3\}$

$T = \{0, 1\}$

$S = \{q_0\}$

P ⇒ $q_0 \longrightarrow 0 \cdot q_1 \mid 1 \cdot q_2$

$q_1 \rightarrow 0.q_3 \mid 1.q_0 \mid 1$

$q_2 \rightarrow 1.q_3 \mid 0.q_0 \mid 0.$

$q_3 \rightarrow 0.q_3 \mid 1.q_3$

Conversion from Regular grammar to Finite Automata :-

Let $G = \{V, T, P, S\}$ be a regular grammar. we can construct DFA whoose :-

(i) States corresponding to variables V

(ii) Starting state corresponding to S.

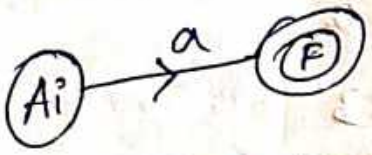
(iii) Transitions in M corresponding to production rules in P input symbols in M corresponding to Terminal symbol T.

(iv) If there is a production of the form $q_i \rightarrow a$ then the transition is terminated at new state called final state

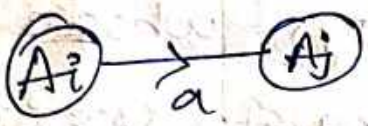
rules:-

(i) If the production rule is of the form $A \rightarrow \epsilon$ then A is final state represented as (A)

(ii) If the production rule is of the form $A_i \rightarrow a$ then there is a transition from A_i to final state labeled with a



(iii) If the production rule is off the form $A_i \longrightarrow a.A_j$ then there is a transition from A_i to A_j labeled with a represented as



(iv) If a production rule is off the form

$A_i \longrightarrow a_1 a_2 a_3 \cdots a_n A_j$ then there is a transition from A_i to A_j and add a intermediate states labelled by $a_1, a_2, a_3 \cdots a_n$ is represented as

No. of terminals $(a_1 a_2 \ldots a_n)^2$ No. of intermediate states.

Example:-

i) $S \rightarrow aA \mid B$

$A \rightarrow aaB$

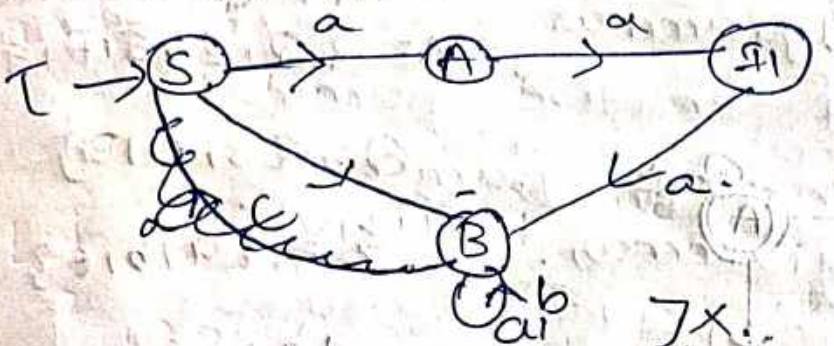
$B \rightarrow bB \mid a$

Note:- (Grammar should be always in RLG. If not we have to convert into

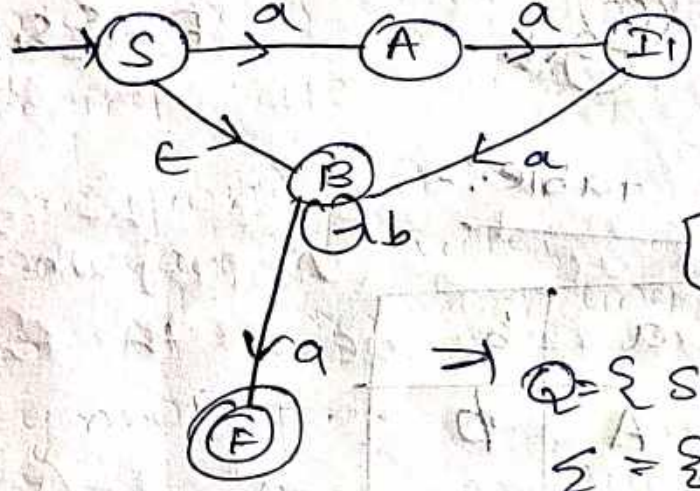
→ Given grammar already on RLG.

$V = \{S, A, B\}$

$T = \{a, b\}$



∃x..



[F = Assume Final State].

→ $Q = \{S, A, I, B, F\}$

$\Sigma = \{a, b, \epsilon\}$

$F = F$

$Q_0 = \{S\}$

(ii)

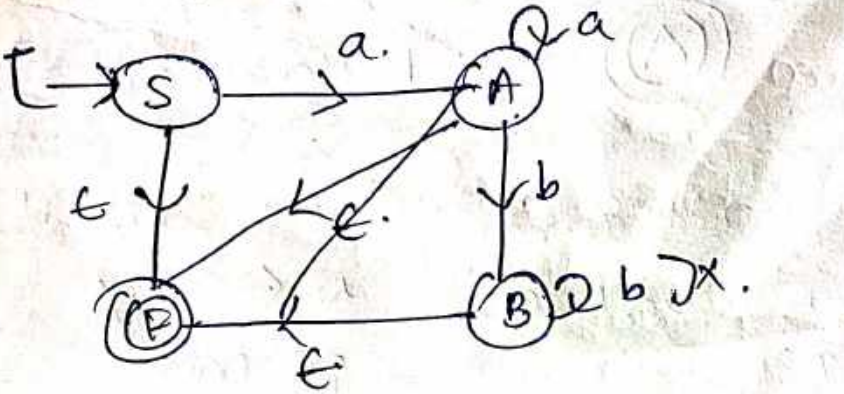
$S \rightarrow aA \mid \epsilon$

$A \rightarrow aA \mid bB \mid \epsilon$

$B \rightarrow bB \mid \epsilon$

$V = \{S, A, B\}$

$T = \{a, b, \epsilon\}$



∃x.

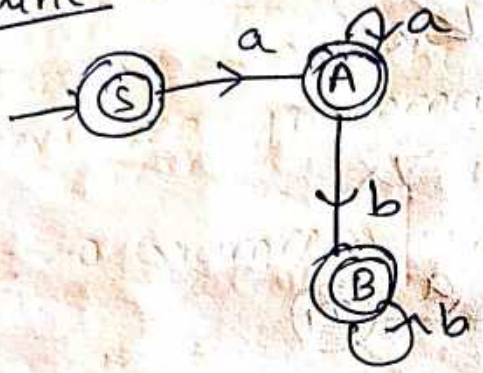
$G = \{V, T, P, S\}$

$V = \{S, A, B\}$

$T = \{a, b, \in\}$

$S = \{S\}$

Machine



Before machine

$M = \{Q, \Sigma, \delta, F, q_0\}$

$Q = \{S, A, B\}$

$\Sigma = \{a, b, \in\}$

$q_0 = \{S\}$, $F = \{\}$

After Machine:

$M = \{Q, \Sigma, \delta, F, q_0\}$

$Q = \{S, A, B\}$

$\Sigma = \{a, b, \in\}$

$q_0 = \{S\}$

$F = \{S, A, B\}$.

Transition table:-

δ	a	b
S	A	ϕ
A	A	B
B	ϕ	B

Types of Grammars:-

(i) Regular grammar (ii) Type-3 grammar.

(ii) context free grammar (ii) Type-2 grammar

1) Regular grammar (or) Type-3 grammar:-
It is defined as $G=\{V,T,P,S\}$ where P is set of production rules are of the form L.L.G (or) R.L.G.

2) Type-2 grammar (or) context-free grammars:-
context free grammars defined as (CFG):-
$\{V,T,P,S\}$ where P is set of productions are of the form

$$\alpha \rightarrow \beta$$
where $\alpha \in V$
$\beta \in (V \cup T)^*$

$A \rightarrow aAb$ } left side only single
$A \rightarrow ab$ } Rightside can be many Numbers

Note:-
All Regular grammars are context free grammars But all context free grammars are NOT Regular

$A \rightarrow Aa$ } — Both Regular & context free grammars.
$A \rightarrow a$ }

3) Type-1 grammar (or) context sensitive grammar (CSG):-
context sensitive grammar is defined as $G=\{V,T,P,S\}$ where P is set of production rules are of the form $\alpha \rightarrow \beta$ where $\alpha \in (V \cup T)^+$
$\beta \in (V \cup T)^+$ [length of $\alpha \leq$ length of β]

$\alpha \in (V \cup T)^+$

$\beta \in (V \cup T)^*$

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

NOTE:- Every Regular grammars & context free gram are context sensitive grammar. But context sensitive grammar need not context free grammar.

4) Type -0 Grammar | unrestricted grammar | REG (Recursive Enumerable grammar

Def:-

It is defined as $G = \{V, T, P, S\}$ where P is the production rules are of the form set of

$\alpha \rightarrow \beta$ where $\alpha \in (V \cup T)^*$, $\beta \in (V \cup T)^*$, $|\alpha| \geq |\beta|$

eg:- $bAa \rightarrow aA$

Context free grammar | Type-2 grammar:-

context free grammar defined as $\{V, T, P, S\}$ where P is a set of production rules on the form

$\alpha \rightarrow \beta$

where $\alpha \in V$

$\beta \in (V \cup T)^*$

$A \rightarrow aab$

$A \rightarrow ab$

closure properties of context free grammar:-

(i) context free languages are closed under union
(ii) CF languages are closed under concatenation
(iii) CF languages are closed under kleen-closure
(iv) CF languages are closed under reversal
(v) Homomorphism
(vi) inverse Homomorphism
(vii) closed under substitution
(viii) closed under infix | prefix
(ix) closed under cycle operation.

derivation:-

It is a process of generating string from a given grammar It can be represented graphically known as derivation. ~~tree~~ classified into two types:-

(i) left most derivation (LMD)
(ii) Right most derivation (RMD),

LMD:-

In this we can replace a left most variable to obtain the given input

RMD:-

In this we can replace a right most variable to obtain the given input

Derivation tree:-

Let $G=\{V, T, P, S\}$ be a context free grammar then there is a derivation tree for G if and only if

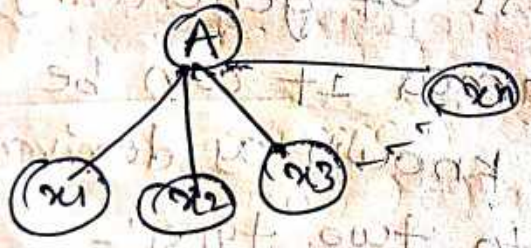
(i) root node of tree is labelled with start symbol S of G.

(ii) All leaf nodes of the tree are labelled with terminals (or) special symbols of G

(iii) the interior nodes are labelled by variables of G.

(iv) If any p in G are of the form

$A \rightarrow x_1 x_2 x_3 \cdots x_n$ then the derivation tree is



Example

find the RMD & LMD and parse tree for the input string

id + id * id from the following grammar

$E \rightarrow E+E$

$E \rightarrow E*E$

$E \rightarrow id$

LMD

$E \rightarrow E+E$

$\rightarrow id+E$

~~$\rightarrow id+id$~~

$\rightarrow id+E*E$

$\rightarrow id+id*E$

$\rightarrow id+id*id$

(NEXT E also id means given one is not generating [E=id]. [E=id]

Evaluation is from L to R so it is LMD

RMD

$E \rightarrow E+E$

$\rightarrow E+E*E$

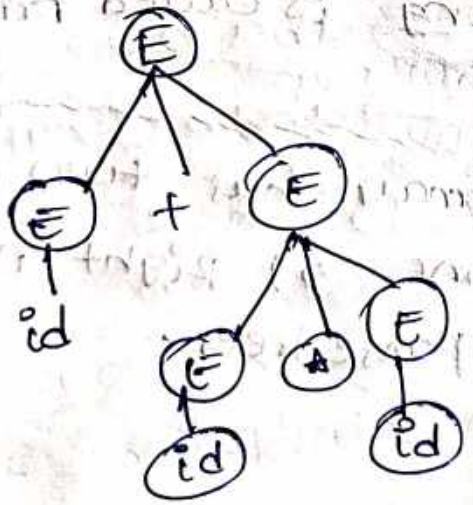
$\rightarrow E+E*id$

$\rightarrow E+id*id$

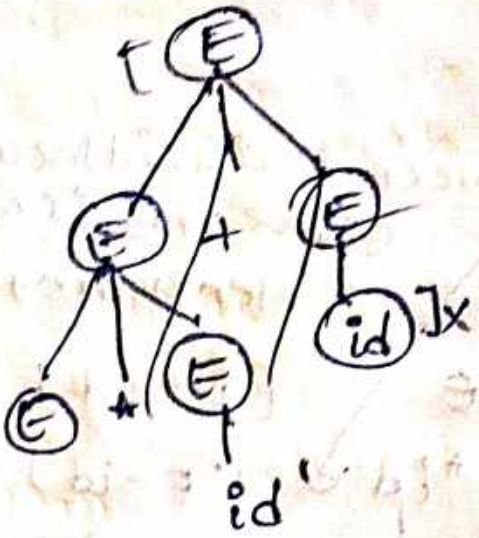
$\rightarrow id+id*id$

[R to L = RMD]

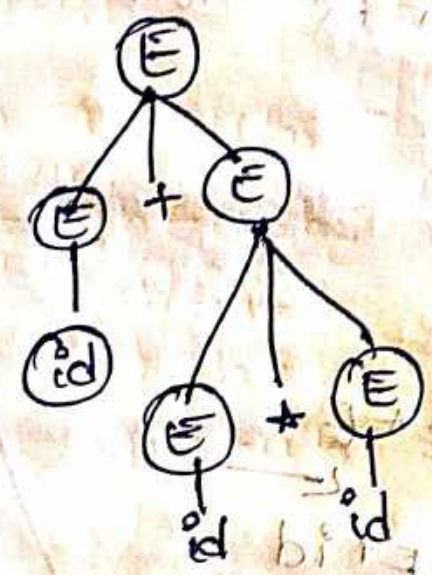
Parsetree



- LMD



RMD



RMD

Ambiguous grammars:-

A context free grammar where G={V,T,P,S} which generate two (or) more parse tree for a given input string is called Ambiguous grammars.

An Ambigious grammar has two (or) more left most derivations (or) Right most derivations or) parse trees.

Ex:- Given grammar is ambigious (or) not

Given string

id+id*id

$E \rightarrow E+E$

$E \rightarrow E*E$

$E \rightarrow id$

LMD

$E \rightarrow E+E$

$\rightarrow id+E$

$\rightarrow id+E*E$

$\rightarrow id+id*E$

$\rightarrow id+id*id$

RMD

$E \rightarrow E*E$

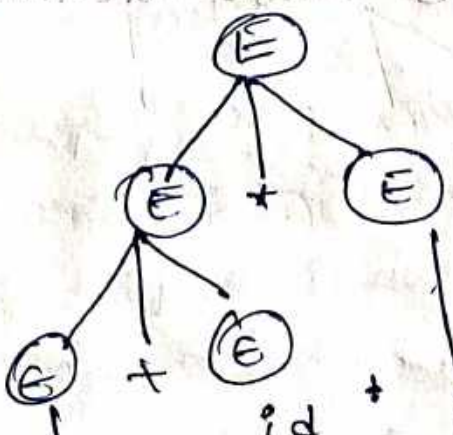
$E \rightarrow E+E*E$

$\rightarrow id+E*E$

$\rightarrow id+id*E$

$\rightarrow id+id*id$

Two LMD are Explained Now we have to design a parse tree



(id+(id*id)) → Two different parse tree

id+id so it is called ambiguous grammar

Ex:- Check whether given ambiguous (or) not.

string = abab

$S \rightarrow asbs$

$S \rightarrow bsbs$

$S \rightarrow \epsilon$

1-LMD

$S \rightarrow asbs$ [Evaluate Non terminal]

$S \rightarrow absasbs$

$\rightarrow ab\epsilon asbs$ [$ab\epsilon asbs$ ⇒ $abasbs$]

$\rightarrow ab\epsilon a\epsilon bs$

$\rightarrow ababs$

$\Rightarrow abab$

2nd-LMD

~~$S \rightarrow bsas$~~ $S \rightarrow asbs$

$S \rightarrow a\epsilon bs$

$\rightarrow abs$

$\rightarrow ababs$ [$S \rightarrow asbs$]

$\rightarrow abab$ [$S \rightarrow \epsilon$] [final]

2 LMD are exist so these are ambiguous grammar.

(2) $E \longrightarrow E+E$

$E \rightarrow E*E$

$E \rightarrow id$ ✗

simplification (or) minimization of context free grammars :-

We have seen various languages can Effectively be represented by context free grammar. All grammars are not always optimized that means a grammar may contain some extra (or) unneeessary symbol this will increase the length of the grammar so that the simplification of the grammar involves removing all these unnecessary symbols.

Simplification of grammar including the following conditions :-

---

(i) Elimination of useless symbols

(ii) Elimination of ϵ products

(iii) Elimination of unit products

of the form $A \longrightarrow B$

Elimination of useless Symbols :-

(i) A Variable is said to be useless if And only if it doesn't genelates a terminal string

ii) It doesn't used on the derivation of a string atleast once.

1) Ex:- Eliminate useless symbols in the following grammar

$S \rightarrow BC \mid AB \mid CA$

$A \rightarrow a$

$C \rightarrow aB \mid b$

A contains input $A \rightarrow a$

contains input $a, b \Rightarrow C \rightarrow aB \mid b$

B does not contain any terminal so remove all the B products

$S \rightarrow CA$

$A \rightarrow a$

$C \rightarrow b$

2)

$A \rightarrow aS \mid A \mid BC$

$A \rightarrow a$

$B \rightarrow aa$

$C \rightarrow aCb$

$S \rightarrow aS \mid A$ S=start state.

$A \rightarrow a$

$B \rightarrow aa$

Elimination of ∈ productions:-

A production is of the form $A \to \epsilon$ is called Null (or) ∈ production. If the grammar contains $A \to \epsilon$ then replace A with ∈ in the remaining productions remove $A \to \epsilon$ from the grammar.

Ex:- Eliminate null products in the following grammar

$A \to 0B1 \mid 1B1$

$B \to 0B \mid 1B \mid \epsilon$

$A \to 0B1 \mid 1B1$

$B \to 0B \mid 1B \mid \epsilon$

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

$B \to 0B \mid 0 \mid 1B \mid 1$

$0B1 = 0\epsilon1 = 01$
$1B1 = 1\epsilon1 = 11$
$0B = 0\epsilon = 0$
$1B = 1\epsilon = 1$

⇒ given gramma ∈ grammar

⇒ remove ∈

Ex:-

$S \to aA$

$A \to BB$

$B \to aBb \mid \epsilon$

$S \to aA \mid a$

$A \to BB \mid B$

$B \to aBb \mid ab$

[∵ $a\epsilon b = ab$]

$A \to \epsilon$ substitute in $S \to aA$

No useless symbols. [$B \to ab$, $A \to B$].

Ex:-

$S \to ABaC$

$A \to BC$

$B \to b \mid \epsilon$

$C \to D \mid \epsilon$

$D \to d$

$D \to d$

$C \to D$

$A \to B \mid C \mid BC$

$S \to ABac \mid Aac \mid Aa \mid BaC \mid ABa \mid a$

$B \to b$

No unused/useless Symbols

Elimination of Unit productions:-

(i) the unit products of the form $A \to B$ for each pair of non-terminals $A \& B$ such that there is a production $A \to B$ & A Non-unit production from B

$B \to s_1 \mid s_2 \mid \ldots \mid s_n$

$s_i \in (V+T)^*$ or string of terminals non-terminals.

then create new productions as

$A \to s_1 \mid s_2 \mid \ldots \mid s_n$ do the same for all the pairs of A and B simultaneously

remove all unit productions.

Ex:- Eliminate the unit productions in the grammar:- $S \to A|bb$

$A \to B|b$

$B \to S|a$

$S \to a|b|bb$

$A \to a|b|bb$

$B \to a|b|bb$

Ex:-

$S \to Aa|B$

$B \to A|bb$

$A \to a|bc|B$

$A \to a|bc|bb$

$B \to a|bc|bb$

~~$S \to a|a|abc|abb|a|bc|bb$~~

$Aa \to X$

$S \to Aa|a|bc|bb.$

Q) Elimination of use symbols of a string:

$S \rightarrow aAa$

$A \rightarrow bBB$

$B \rightarrow ab$

$C \rightarrow ab.$

C is useless because B contains same inputs NO need to use C

$S \rightarrow aAa$

$A \rightarrow bBB$

$B \rightarrow ab$

Q) Simplify the following grammar

$S \rightarrow ABA$

$A \rightarrow aA|\epsilon$

$B \rightarrow bB|\epsilon$

(i) Elimination of useless Symbols:- NO use less symbols

$S \rightarrow ABA$

$A \rightarrow aA|\epsilon$

$B \rightarrow bB|\epsilon$

(ii) Elimination of ϵ:-

$S \rightarrow ABA|BA|AB|AA|B|A$

$A \rightarrow aA|a$

$B \rightarrow bB|b$

(iii) Elimination of unit Symbols/productions

$B \rightarrow b$

$A \rightarrow aJX$

$S \rightarrow ABA \mid BA \mid AB \mid AA \mid B \mid A$

$\rightarrow ABA \mid BA \mid AB \mid AA \mid aA \mid a \mid bB \mid b$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

Q)

$S \rightarrow AB \mid aB$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

(i) NO use less symbols

(ii) NO ϵ elimination.

(iii) NO unit symbols

Q) $S \rightarrow AB \mid aB$

$A \rightarrow aab \mid \epsilon$

$B \rightarrow bbA$

(i) ~~OB use less symbols~~

$S \rightarrow \epsilon$

(i) NO useless symbols.

(ii) $A \rightarrow aab$

$B \rightarrow bbA \mid bb$

$S \rightarrow AB \mid B \mid aB$

} Elimination of ϵ.

(iii) $S \rightarrow AB \mid bbA \mid bb \mid aB$

$A \rightarrow aab$

$B \rightarrow bbA \mid bb$

} elimination of unit productions

2) $S \rightarrow aA \mid aBB$

$A \rightarrow a \mid AA \mid \epsilon$

$B \rightarrow bB \mid bbC$

$C \rightarrow B$

(i) B and C are useless

(ii) $S \rightarrow aA \mid a$

$A \rightarrow a \mid aA \mid aAA$

(ii) $S \rightarrow aA \mid a$

$A \rightarrow a \mid aA \mid aAA$

Normal forms:-

(i) The Grammar can be simplifying reducing the ϵ products & removing the useless and unit products

(ii) there is also aneed to have grammar on specific form

(iii) As you see on the context free grammar on the right hand side of the productions there are any no. of Non-terminals & terminals on any combination.

(iv) we need to normalize such a grammar to standardize the processing of strings that means we want the grammar in some specific format

(iv) there should be fixed no. of terminals & non-terminals in CFG with some criteria

(v) There are two important Normal forms

(i) Chomsky Normal Form (CNF)

(ii) Greibach Normal Form (GNF)

CNF:-

A context free grammar is in CNF if each of its productions has one of the two forms

(i) Non-terminal (N.T)

N.T ⟶ Strings of exactly two Non terminals

N.T → N.T.NT

(ii) Non terminal derive to one terminal

NT → T

Ex:- A ⟶ A

⇒ In a CNF the no. of symbols on the Right hand side of the production is strictly limited.

⇒ the Nature of symbols on the right hand side is restricted.

Example:-

Convert the following CFG into Chomsky Normal form (CNF) ?

$S \rightarrow bA \mid aB$

$A \rightarrow bAA \mid aS \mid a$

$B \rightarrow aBB \mid bS \mid b$

convert Non-terminals to terminals

$S \rightarrow C_b A \mid D_a B$

$C_b \rightarrow b$

$D_a \rightarrow a$

$A \rightarrow C_b AA \mid D_a S \mid a$

$A \rightarrow C_b E \mid D_a S \mid a$

$E \rightarrow AA$

$B \rightarrow D_a BB \mid C_b S \mid b$

F

$B \rightarrow D_a F \mid C_b S \mid b$

$F \rightarrow BB$

→ final gram CNF

Final grammar:-

$S \rightarrow C_b A \mid D_a B$

$A \rightarrow C_b E \mid D_a S \mid a$

$E \rightarrow AA$

$B \rightarrow D_a F \mid C_b S \mid b$

$F \rightarrow BB$

$C_b \rightarrow b$

$D_a \rightarrow a$

Final grammar

Q) CFG - CNP

$S \to AB \mid aB$

$A \to aab \mid \epsilon$

$B \to bbA$

$S \to AB \mid B \mid aB$

$A \to aab$

$B \to bbA \mid bb$

$\Rightarrow$ $S \to AB \mid aB \mid bbA \mid bb$

$A \to aab$

$B \to bbA \mid bb$

$A \to Ca$, $B \to Da$

$S \to Cada \mid Da \mid ada$

$A \to Ca$

$B \to Da$

$A \to aab$

$B \to bbCa \mid bb$

$S \to AB \mid CbCbA \mid CbCb \mid DaB$

$Cb \to b$

$Da \to a$

$S \to AB \mid EA \mid CbCb \mid DaB$

$E \to CbCb$

$Cb \rightarrow b$

$Da \rightarrow a$

$A \rightarrow \underbrace{DaDa}_{F}Cb$

$A \rightarrow FCb$

$F \rightarrow DaDa$

$B \rightarrow CbCb\ A \mid CbCb$

$B \rightarrow EA \mid CbCb$

Final grammar

~~$S \rightarrow AB \mid CbCbA \mid CbCb \mid DaB$~~

$Cb \rightarrow b$

$Da \rightarrow a$

$S \rightarrow AB \mid EA \mid CbCb \mid DaB$

$E \rightarrow CbCb$

$A \rightarrow FCb$

$F \rightarrow DaDa$

$B \rightarrow EA \mid CbCb$

Q) $S \rightarrow ABB \mid \epsilon$

$A \rightarrow aAS \mid a$

$B \rightarrow SbS \mid A \mid bb$

$S \rightarrow ASB \mid AB$

$A \rightarrow aAS \mid aA \mid a$

$B \rightarrow SbS \mid b \mid A \mid bb \mid bS \mid Sb$

$B \rightarrow SbS \mid b \mid aAS \mid aA \mid a \mid bb \mid bS \mid Sb$

$S \rightarrow ASB \mid AB$

~~$A \rightarrow Ca$~~ $C \rightarrow AS$

~~$S \rightarrow CaSB \mid AB$~~

$S \rightarrow CB \mid AB$

$C \rightarrow AS$

$A \rightarrow aAS \mid a$

~~$D \rightarrow a$~~ $Da \rightarrow a$

$A \rightarrow DaAS \mid a \mid aA$

$A \rightarrow DaAS \mid DaA \mid a$

$A \rightarrow DaC \mid DaA \mid a$

$Da \rightarrow a$

$B \rightarrow DC \mid DaA \mid EE \mid b \mid a \mid ES \mid SE \mid FS$

$E \rightarrow b$

$F \rightarrow SE$

Final grammar

$S \rightarrow CB \mid AB$

$A \rightarrow DA \mid DC \mid a$

$C \rightarrow AS$

$D \rightarrow a$

$B \rightarrow DC \mid DA \mid EE \mid ES \mid SE \mid FS \mid a \mid b$

$E \rightarrow b$

$F \rightarrow SE$

GNF (Griebach Normal Form):-

It is defined as $N \cdot T \rightarrow T \cdot$ [Any no. of non terminals]

$$N \cdot T \rightarrow T \cdot (NT)^*$$

$$NT \rightarrow T$$

For converting given grammar into GNF we use two lemma's:-

(i) Lemma 1 (Substitution rule):-

Let $G = \{V, T, P, S\}$ be a context free grammar. Let $A \rightarrow B\alpha$ be a production in B. if there is a production

$$B \rightarrow \beta_1 | \beta_2 | \beta_3 | \ldots | \beta_n$$

then equivalent grammar can be obtained by substituting B on A.

the resulting grammar is

$$A \rightarrow \beta_1\alpha | \beta_2\alpha | \beta_3\alpha | \ldots | \beta_n\alpha$$

(ii) Lemma 2 (Elimination of left recursion)

$A \rightarrow Aa \rightarrow$ same symbol left recursion grammar

Grammars of the form $A \rightarrow A\alpha/\beta$ is called LR grammar (left recursion grammar) To Eliminate left recursion rewrite the grammar as $A \rightarrow \beta A'$

$A' \rightarrow \alpha A' | \epsilon$ → minimizing we get A

$A \to \beta A' \mid \beta$

$A' \to \alpha A' \mid \alpha$

Q) convert the CFG to GNF

$S \to ABA$

$A \to aA \mid \epsilon$

$B \to bB \mid \epsilon$

$S \to AB \mid BA \mid B \mid ABA \mid AA \mid A$

$A \to aA \mid a$

$B \to bB \mid b$

$S \to AB \mid BA \mid bB \mid b \mid ABA \mid AA \mid aA \mid a$

$A \to aA \mid a$

$B \to bB \mid b$

this grammar solves by lemma1 because No right hand side No same starting Symbol

$S \to \underline{A}BA \mid \underline{B}A \mid \underline{A}B \mid \underline{A}A \mid aA \mid a \mid bB \mid b$

$A \to aA \mid a$

$B \to bB \mid b$

Substitute A on S

$S \to aABA \mid BA \mid bBA \mid bA \mid aAB \mid aB \mid aAA \mid aA \mid a \mid bB \mid b$

$A \to aA \mid a$

$B \to bB \mid b$

Q) $S \to CA$

$A \to a$

$C \to B \mid b$ useless

$S \to CA$

$A \to a$

$C \to b$

CNF :-

$S \to bA$

$A \to a$

$C \to b$

IMP

$S \to AA \mid a$

$A \to SS \mid b$

$S = A_1 \quad A = A_2$

$A_1 \to A_2 A_2 \mid a$ ①

$A_2 \to A_1 A_1 \mid b$ ②

① In ②

$A_2 \to A_2 A_2 A_1 \mid b$

$A \to A\alpha \mid \beta$

$A \to \beta A' \mid \beta$

$A \to \alpha A' \mid \alpha$

$A_2 \to bZ \mid b$

$Z \to$

Q) Imp

$S \rightarrow AA \mid a$

$A \rightarrow SS \mid b$

$S = A_1, A = A_2$

$A_1 \rightarrow A_2A_2 \mid a$ — ①

$A_2 \rightarrow A_1A_1 \mid b$ — ②

Substitute ① in ②

$A_2 \rightarrow A_2A_2A_1 \mid aA_1 \mid b$

$A \quad A \quad \alpha \quad \beta_1 \quad \beta_2$

$A \rightarrow A\alpha \mid \beta$

$A \rightarrow \beta A' \mid \beta$

$A' \rightarrow \alpha A' \mid \alpha$

$A_2 \rightarrow aA_1Z \mid bZ \mid aA_1 \mid b$ ~ GNF

$Z \rightarrow A_2A_1Z \mid A_2A_1$

Substitute A_2 in A_1

$A_1 \rightarrow A_2A_2 \mid a$

$A_1 \rightarrow aA_1ZA_2 \mid aA_2 \mid bZA_2 \mid aA_1A_2 \mid bA_2 \mid a$

~~$Z \rightarrow aA_1ZA_1Z \mid bZA_1Z$~~

$Z \rightarrow aA_1ZA_1Z \mid bZA_1Z \mid aA_1A_1Z \mid bA_1Z \mid aA_1A_1$

$\mid bZA_1 \mid aA_1A_1 \mid bA_1$

Q) $S \to XA|BB$

$B \to b|SB$

$X \to b$

$A \to a$

$S \to XA|BB$ } — ①

$B \to b|SB$ } — ②

$X \to b$ } — these Both are in GNF

$A \to a$ }

Substitute ① in ②

$B \to XAB|BBB|b$

$B \to \underbrace{BBB}_{\alpha A} | \underbrace{bAB}_{\beta_1 \beta_2} | b$

$A \to \beta A'|\beta$

$B \to bABA'|bAB|bA'|b$ — GNF

$A' \to \alpha A'|\alpha$

$A' \to BBA'|BB$

Substitute B in A'

$A' \to bABA'BA'|bABA'|bABBA'|bABB|bA'BA'|$
$bA'B|bBA'|bB$

Now substitute B in eq ①

$S \to bA|bABA'B|bABB|bA'B|bB$ — GNF

Final Grammar

$S \rightarrow bA \mid bABA'B \mid bABB \mid bA'B \mid bB$

$B \rightarrow bABA' \mid bAB \mid bA' \mid b$

$A' \rightarrow bABA'BA' \mid bABA' \mid bABBA' \mid bABB \mid bA'BA' \mid bA'B \mid bBA' \mid bB$

$X \rightarrow b$

$A \rightarrow a.$

Pumping Lemma for context free Grammars:

Pumping Lemma is used for proviDing the Given Language is Not a context free Language

Lemma:-

Let L be any context free language then there is a constant n which depends upon L. Such that there exist a string $z \in L$ and $|z| \geq n$ where $z = uvwxy$ such that

(i) $|vx| \geq 1$

(ii) $|vwx| \leq n$

(iii) for $i \geq 0$, $uv^iwx^iy \notin L$

Example

prove that $L = \{a^n b^n c^n \mid n \geq 0\}$ is Not a CFL

$L = \{0, abc, aabbcc, \ldots\ldots\}$

$z = a^n b^n c^n$

$z = a^n b^n c^n$

$\underbrace{aaa}_{u}\underbrace{bbb}_{v}\underbrace{ccc}_{w\,x\,y}$

$i=2$

$aaa^2 bbb^2 c^3$

$a^4 b^4 c^3$

⇒ This $a^4 b^4 c^3$ is Not on the given grammar so it is Not a context free language