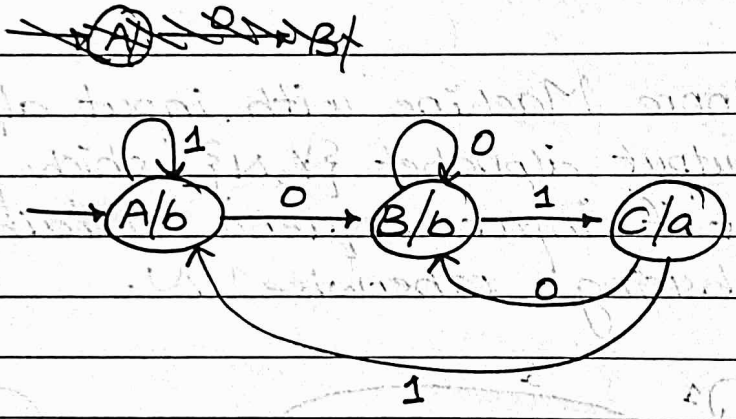
Conversion of Moore Machine to Mealy Machine

Ex1 Construct a Moore Machine that prints 'a' whenever the subsequence '01' is encountered in any input binary string, and then convert it to its equivalent Mealy Machine.

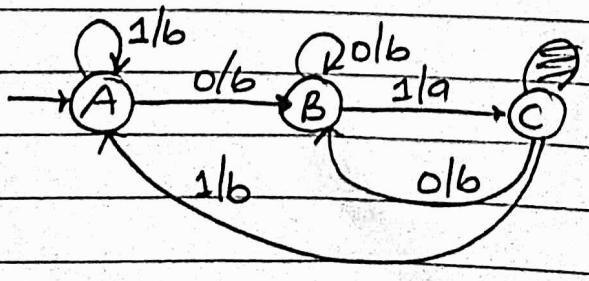
$\Sigma = \{0, 1\}$ $\quad \Delta = \{a, b\}$



Moore Machine

State	0	1	Output
→ A	B	A	b
B	B	C	b
C	B	A	a

Mealy Machine



State	0	1
→ A	B, b	A, b
B	B, b	C, a
C	B, b	A, b

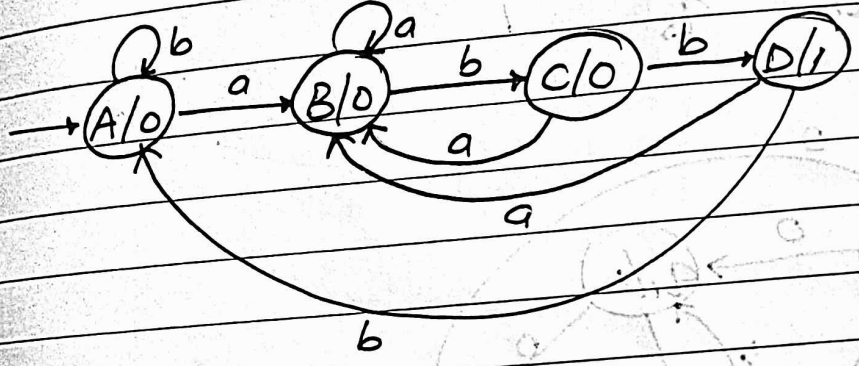
classfellow

Date: / / 2022

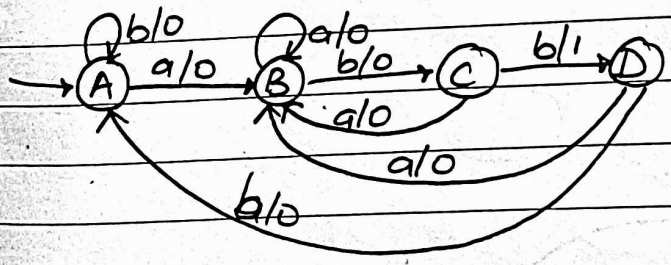
Page No.

Ex2 The given Moore Machine counts the occurences of the sequence 'abb' in any input binary strings over {a,b}. Convert it to its equivalent mealy machine.

$\Sigma = \{a, b\}$ $\Delta = \{0, 1\}$



Mealy Machine



State	a	b
→ A	B, 0	A, 0
B	B, 0	C, 0
C	B, 0	D, 1
D	B, 0	A, 0

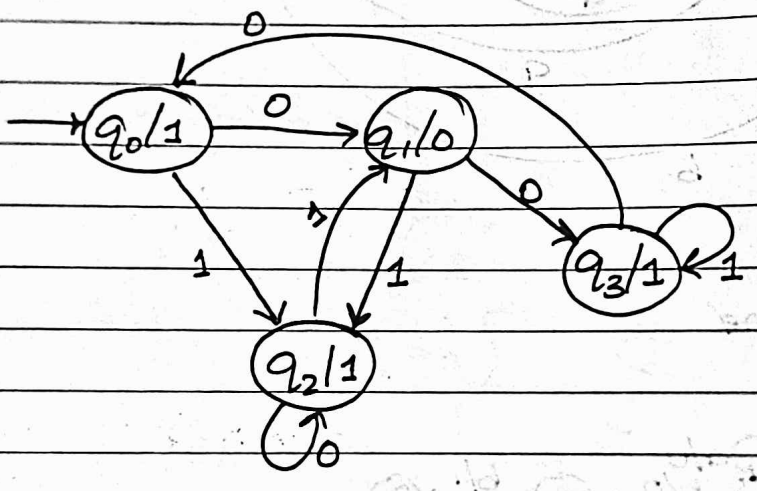
Date : / / 2022

Page No.

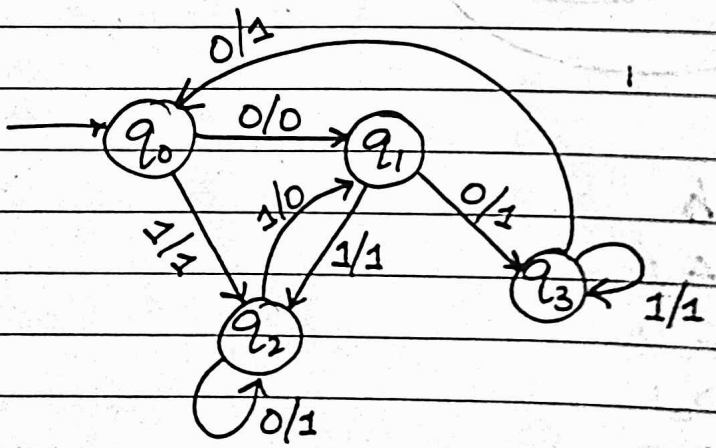
Ex3 Convert the given Moore Machine to its equivalent Mealy Machine

Moore

State	0	1	Output
→ q_0	q_1	q_2	1
q_1	q_3	q_2	0
q_2	q_2	q_1	1
q_3	q_0	q_3	1



Mealy



State	0	1
→ q_0	q_1, 0	q_2, 1
q_1	q_3, 1	q_2, 1
q_2	q_2, 1	q_1, 0
q_3	q_0, 1	q_3, 1

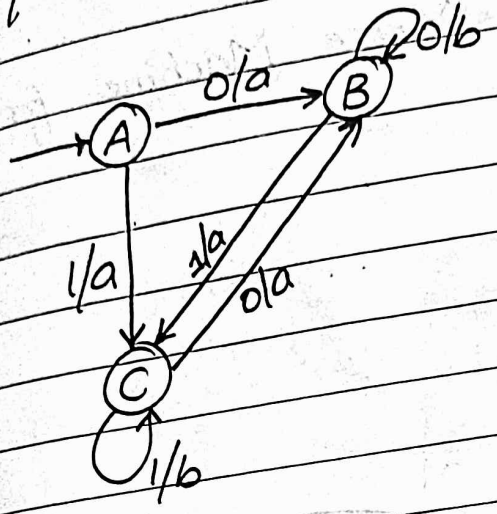
C P P L
Date : / / 2022
Page No.

classfollow

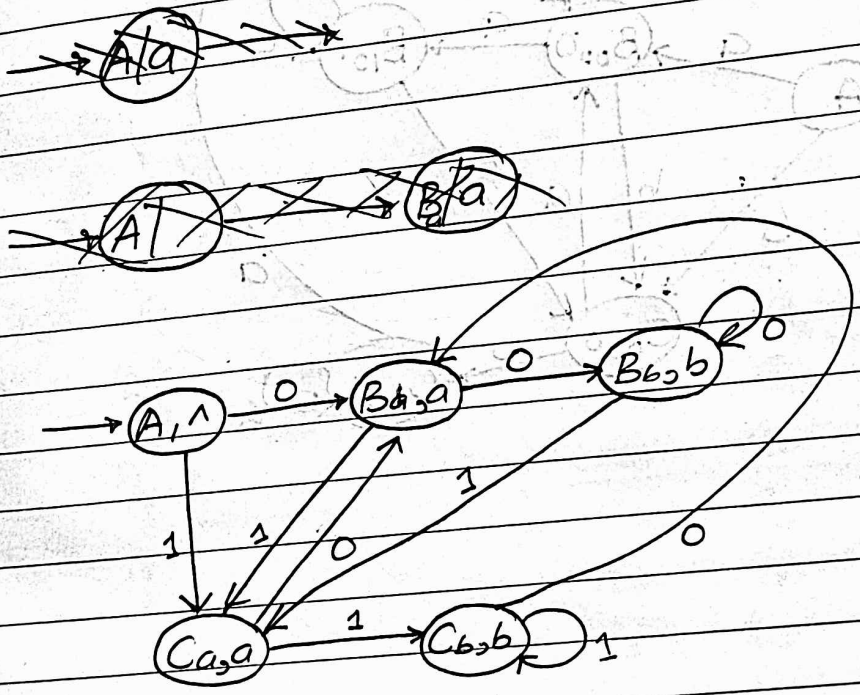
Conversion of Mealy Machine to Moore Machine

Ex1 Convert the following Mealy Machine to its equivalent Moore Machine

Mealy

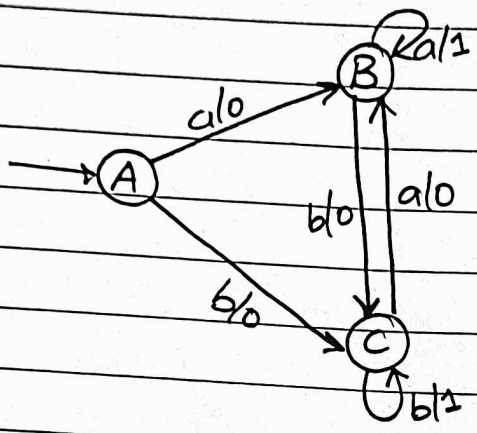


Moore

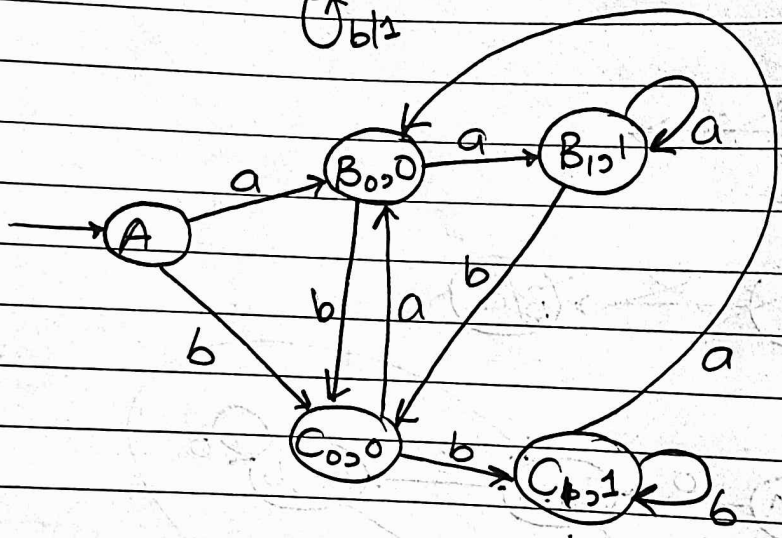


Ex2 Given below is a Mealy Machine that prints '1' whenever the sequence 'aa' or 'bb' is encountered in any input binary string from Σ^* where $\Sigma = \{a, b\}$
Design the equivalent Moore Machine for it.

Mealy

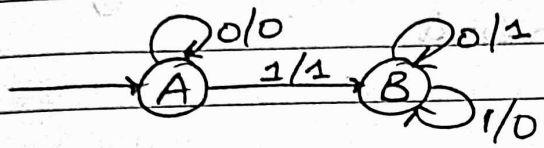


Moore

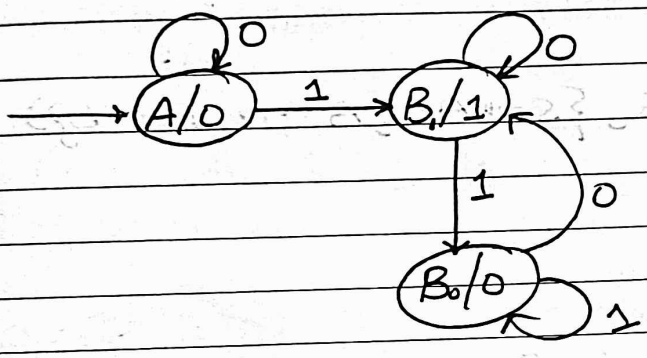


Ex 3 Convert the given Mealy Machine that gives 2's complement of any binary input to its equivalent MOORE Machine.

Mealy



Moore



Convert Mealy Machine to Moore Machine using Transition Table

Mealy

State	a	b
→ q_0	q_3, 0	q_1, 1
q_1	q_0, 1	q_3, 0
q_2	q_2, 1	q_2, 0
q_3	q_3, 0	q_0, 1

Moore

State	a	b	Output
→ q_0	q_3	q_{11}	1
q_{10}	q_0	q_3	0
q_{11}	q_0	q_3	1
q_{20}	q_{21}	q_{20}	0
q_{21}	q_{21}	q_{20}	1
q_3	q_{10}	q_0	0

Grammar

A grammar 'G' can be formally described using 4 tuples as $G = (V, T, S, P)$ where

V = Set of variables or non-terminal symbols

T = Set of terminal symbols

S = Start symbol

P = Production Rule

Example:

$G = (\{S, A, B\}, \{a, b\}, S, \{S \to AB, A \to a, B \to b\})$

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

$T = \{a, b\}$

$S = S$

$P = S \to AB,\ A \to a,\ B \to b$

$S \to AB$

$S \to aB$

$S \to ab$

Regular Grammar

Right Linear Grammar

A grammar is right linear if production is of the form

$A \to xB$

$A \to x$

where $A, B \in V$ and $x \in T$

Eg: $S \to abA | a$

Left Linear Grammar

A grammar is left linear if production is of the form

$A \to Bx$

$A \to x$

$A, B \in V$ and $x \in T$

Eg: $S \to Aab | a$

classfallow

Derivations from a Grammar

The set of all strings that can be derived from a Grammar is said to be the language generated from that Grammar.

Ex1 Consider the grammar

$G1 = (\{S, A\}, \{a, b\}, S, \{S \to aAb, aA \to aaAb, A \to \epsilon\})$

$S \to aAb$
$S \to aaAbb$
$S \to aaaAbbb$
$S \to aaabbb$

Ex2 $G2 = (\{S, A, B\}, \{a, b\}, S, \{S \to AB, A \to a, B \to b\})$

$S \to AB$
$S \to ab$

$L(G2) = \{ab\}$

Ex3 $G3 = (\{S, A, B\}, \{a, b\}, S, \{S \to AB, A \to aA|a, B \to bB|b\})$

$S \to AB$	$S \to AB$	$S \to AB$	$S \to AB$
$S \to aAbB$	$S \to ab$	$S \to aAb$	$S \to abB$
$S \to aabb$		$S \to aab$	$S \to abb$

~~$L(G3) = \{ab, aabb, aabb, abb\}$~~

$L(G3) = \{ab, a^2b^2, a^2b, ab^2, \ldots\}$

$L(G3) = \{a^m b^m \mid m \geq 0 \text{ and } n \geq 0\}$

Date : / / 2022

Page No.

classfellow

Context Free Language

A context free language is a language generated by context free grammar.

CFL → Accepted by Pushdown Automata

Context Free Grammar is defined by 4 tuples as $G=(V, \Sigma, S, P)$ where

V = Set of Non-Terminal Symbols

Σ = Set of Terminal Symbols

S = Start Symbol

P = Production Rule

CFG has production rule of the form

$$A \rightarrow \alpha$$

where $\alpha = \{V \cup \Sigma\}^*$ and $A \in V$

Example:

For generating a language that generates equal number of a's and b's in the form of $a^n b^n$, CFG will be:

$G = \{(S, A), (a, b), (S \rightarrow aAb, A \rightarrow aAb | \epsilon)\}$

$S \rightarrow aAb$

$S \rightarrow aaAbb$

$S \rightarrow aaaAbbb$

$S \rightarrow aaabbb$

$S \rightarrow a^3b^3$

$a^n b^n$.

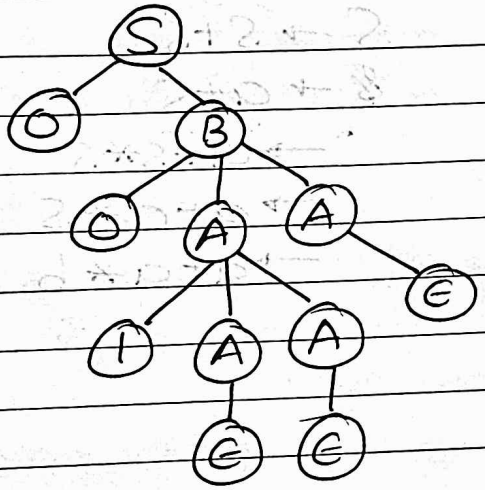
Derivation Tree

A derivation tree or parse tree is an ordered rooted tree that graphically represents the semantic information of strings derived from a ~~CFG~~ ~~CE~~ CFG.

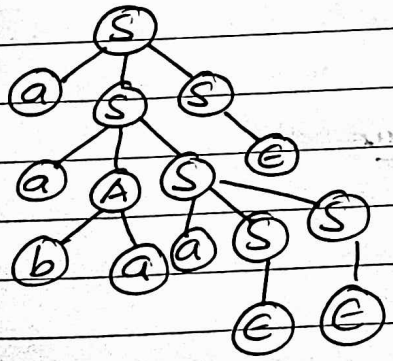
Example:

For the grammar $G=\{V, T, P, S\}$ where

$S \rightarrow 0B,\ A \rightarrow 1AA/\epsilon,\ B \rightarrow 0AA$



Left Derivation Tree

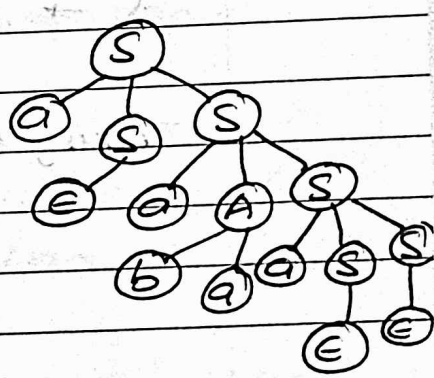


String aabaa

$S \rightarrow aAS/aSS/\epsilon$

$A \rightarrow SbA/ba$

Right Derivation Tree



Date : / / 2022

Page No.

Ambiguous Grammar

A Grammar is said to be Ambiguous if there exists two or more derivation tree for a string w (that means two or more left derivation tree)

Example: $G = (\{S\}, \{a+b, +, *\}, P, S\}$ where P consists of $S \to S+S \mid S*S \mid a \mid b$

String a+a*b can be generated as:-

$S \to S+S$

$\to a+S$

$\to a+S*S$

$\to a+a*S$

$\to a+a*b$

$S \to S*S$

$S \to S+S*S$

$S \to a+S*S$

$S \to a+a*S$

$S \to a+a*b$

Thus, Grammar is Ambiguous

classfellow

Reduction of CFG

In CFG, sometimes all the production rules and symbols are not needed for the derivation of strings. Besides this, there may also be some NULL Productions and UNIT Productions. Elimination of these productions and symbols is called Simplification of CFG.

Simplification consist of following steps:-

1. Reduction of CFG
2. Removal of Unit Production
3. Removal of Null Production

Reduction of CFG

CFGare reduced in two phases

Phase 1: Derivation of an equivalent grammar G', from the CFG, G, such that each variable derives some terminal string

Derivation Procedure:

1: Include all Symbols W_1 that derives some terminal and initialize $i=1$

2: Include symbols W_{i+1} that derives W_i

3: Increment i and repeat Step 2, until $W_{i+1} = W_i$

4: Include all production rules that have W_i in it.

Phase 2: Derivation of an equivalent grammar G'' from the CFG, G' such that each symbol appears in a sentential form

Derivation Procedure:-

1: Include Start Symbol in Y_1 and initialize $i=1$

2: Include all symbols Y_{i+1} that can be derived from Y_i and include all production rules that have been applied

3: Increment and repeat Step 2 until $Y_{i+1} = Y_i$

Ques Find a reduced grammar equivalent to grammar G having production rules

P: $S \to AC/B$, $A \to a$, $C \to dBC$, $E \to aA/e$

Phase1

$T = \{a, c, e\}$

$W_1 = \{A, C, E\}$

$W_2 = \{A, C, E, S\}$

$W_3 = \{A, C, E, S\}$

$G' = \{(A, C, E, S), \{a, c, e\}, P, \{S\}\}$

P: $S \to AC$, $A \to a$, $C \to c$, $E \to aA/e$

Phase2

$Y_1 = \{S\}$

$Y_2 = \{S, A, C\}$

$Y_3 = \{S, A, C, a, c\}$

$Y_4 = \{S, A, C, a, c\}$

$G'' = \{(S, A, C), \{a, c\}, P, \{S\}\}$

P: $S \to AC$, $A \to a$, $C \to c$

classfellow

Date: / / 2022

Page No.

Removal of Unit Productions

Any production rule of the form $A \to B$ where $A, B \in$ Non-Terminals is called Unit Production.

Ques Remove unit productions from the Grammar whose production rule is given by

$P: S \to XY,\ X \to a,\ Y \to Z|b,\ Z \to M,\ M \to N,\ N \to a$

Since $N \to a \Rightarrow M \to a$

$P: S \to XY,\ X \to a,\ Y \to Z|b,\ Z \to M,\ M \to a,\ N \to a$

Since $M \to a \Rightarrow Z \to a$

$P: S \to XY,\ X \to a,\ Y \to Z|b,\ Z \to a,\ M \to a,\ N \to a$

Since $Z \to a \Rightarrow Y \to Z$

$P: S \to XY,\ X \to a,\ Y \to a|b,\ Z \to a,\ M \to a,\ N \to a$

Remove unreachable symbols

$P: S \to XY,\ X \to a,\ Y \to a|b$

Removal of Null Production

In a CFG, a Non-Terminal Symbol 'A' is a nullable variable if there is a production $A \to \epsilon$ or there is a derivation that starts at 'A' and leads to ϵ

Ques Remove Null Productions from the following grammar:-

$S \to ABAC,\ A \to aA|\epsilon,\ B \to bB|\epsilon,\ C \to c$

1. To eliminate $A \to \epsilon$

$S \to ABAC$

$S \to ABC|BAC|BC$

$A \to aA$

$A \to a$

$S \to ABAC|ABC|BAC|BC,\ A \to aA|a,$
$B \to bB|\epsilon,\ C \to c$

2. To eliminate $B \to \epsilon$

$S \to AAC|AC|C,\ B \to b$

New production:

$S \to ABAC|ABC|BAC|BC|AAC|AC|C$
$A \to aA|a,\ B \to bB|b,\ C \to c$

Date: / / 2022

Page No.

Chomsky Normal Form

In CNF we have restriction on the length of RHS, elements in RHS should either be two variables or a Terminal.

A CFG is in CNF if productions are in the following forms:

$A \rightarrow a$

$A \rightarrow BC$

where A, B, C are non-terminals and a is terminal.

=> Conversion CFG to CNF

1. If Start symbol S occurs on the right side, create a new Start symbol S' and a new production $S' \rightarrow S$
2. Remove Null Productions
3. Remove Unit Productions
4. Replace each production $A \rightarrow B_1 \dots B_n$ where $n > 2$ with $A \rightarrow B_1 C$ where $C \rightarrow B_2 \dots B_n$ Repeat this step for all productions having two or more symbols on right side.
5. If right side of any production is of the form $A \rightarrow aB$ where 'a' is a terminal and A and B are non-terminals, then the Production is replaced by $A \rightarrow XB$ and $X \rightarrow a$. Repeat this step for every Production which is of the form $A \rightarrow aB$.

Date : / / 2022

Page No.

Ques Convert CFG to CNF

P: $S \to ASA|aB$

$A \to B|S$

$B \to b|\epsilon$

Sol 1. $S' \to S$

$S \to ASA|aB$

$A \to B|S$

$B \to b|\epsilon$

2. $B \to \epsilon$ | Removing $B \to \epsilon$

$S' \to S$

$S \to ASA|aB|a$

$A \to B|S|\epsilon$

$B \to b$

Removing $A \to \epsilon$

$S' \to S$

$S \to ASA|aB|a|AS|SA|S$

$A \to B|S$

$B \to b$

3. Remove unit production

After removing $S' \to S$

$S' \to ASA|aB|a|AS|SA|S$

After removing $S \to S$

$S' \to S$

$S \to ASA|aB|a|AS|SA$

$A \to B|S$

$B \to b$

Date: 1 1 2022
Page No.

classfellow

After removing $S' \to S$

$S' \to ASA|aB|a|AS|SA$

$S \to ASA|aB|a|AS|SA$

$A \to B|S$

$B \to b$

After removing $A \to B$

$S' \to ASA|aB|a|AS|SA$

$S \to ASA|aB|a|AS|SA$

$A \to b|S$

$B \to b$

After removing $A \to S$

$S' \to ASA|aB|a|AS|SA$

$S \to ASA|aB|a|AS|SA$

$A \to b|ASA|aB|a|AS|SA$

$B \to b$

4. $S' \to AX|aB|a|AS|SA$

$S \to AX|aB|a|AS|SA$

$A \to b|AX|aB|a|AS|SA$

$B \to b$

$X \to SA$

$S' \to AX|YB|a|AS|SA$

$S \to AX|YB|a|AS|SA$

$A \to b|AX|YB|a|AS|SA$

$B \to b$

$X \to SA$

$Y \to a$

CNF Form

classfellow

Greibach Normal Form

A CFG is in GNF if productions are of the form:

$A \rightarrow b$

$A \rightarrow bC_1C_2 \ldots C_n$

where $A, C_1, C_2 \ldots C_n$ are Non-Terminals and b is a Terminal.

Steps to convert to GNF

1. Remove null production and unit production if any
2. Convert to CNF
3. Change names of Non-Terminal Symbols into some A_j in ascending order of j.

~~Ques~~ ~~Convert to GNF~~ Example:

$S \rightarrow CA|BB$

$B \rightarrow b|SB$

$C \rightarrow b$

$A \rightarrow a$

Replace: S with A_1
C with A_2
A with A_3
B with A_4

$\Rightarrow$ $A_1 \rightarrow A_2A_3|A_4A_4$

$A_4 \rightarrow b|A_1A_4$

$A_2 \rightarrow b$

$A_3 \rightarrow a$

4. Alter the rules so that Non-Terminals are in ascending order, such that,
If production is of the form $A_i \rightarrow A_j x$, then $i < j$ and should never be $i \geq j$

Example: $A_4 \rightarrow b|A_1A_4$

$A_4 \rightarrow b|A_2A_3A_4|A_4A_4A_4$

$A_4 \rightarrow b|bA_3|A_4|A_4A_4A_4$

↓

Left Recursion

5. Remove Left Recursion

Ques Convert CFG to GNF

$A_1 \rightarrow A_2A_3 \mid A_4A_4$

$A_4 \rightarrow b \mid A_1A_4$ ⟶ $A_4 \rightarrow b \mid bA_3A_4 \mid A_4A_4A_4$ ← left recursion

$A_2 \rightarrow b$

$A_3 \rightarrow a$

Formula for left recursion

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

$A \rightarrow \beta Z$

$Z \rightarrow \alpha Z \mid \epsilon$

To remove left recursion,

$A_4 \rightarrow \underbrace{b}_{\beta} \mid \underbrace{bA_3A_4}_{\beta} \mid A_4\underbrace{A_4A_4}_{\alpha}$

$A_4 \rightarrow b \mid bA_3A_4 \mid bZ \mid bA_3A_4Z$

$Z \rightarrow A_4A_4Z \mid A_4A_4$

Now grammar is:

$A_1 \rightarrow A_2A_3 \mid A_4A_4$

$A_4 \rightarrow b \mid bA_3A_4 \mid bZ \mid bA_3A_4Z$

$Z \rightarrow A_4A_4Z \mid A_4A_4$

$A_2 \rightarrow b$

$A_3 \rightarrow a$

classfellow

$A_1 \rightarrow bA_3 \mid bA_4A_1 \mid bA_3A_4A_4 \mid bZA_4 \mid bA_3A_4ZA_4$

$A_4 \rightarrow b \mid bA_3A_4 \mid bZ \mid bA_3A_4Z$

$Z \rightarrow bA_4 \mid bA_3A_4A_4 \mid bZA_4 \mid bA_3A_4ZA_4 \mid$
$bA_4Z \mid bA_3A_4A_4Z \mid bZA_4Z \mid bA_3A_4ZA_4Z$

$A_2 \rightarrow b$

$A_3 \rightarrow a$

Required GNF Form