Context Free Grammar (or) Type 2 Grammar

→ It is also known as type 2 grammar

→ Regular grammar is also known as type 3 grammar.

→ A context free grammar is denoted to

$$G = (V, T, P, S)$$

where V = finite set of variables or non-terminals

T = finite set of terminals

P = finite set of productions of the form $A \to (V \cup T)^*$ i.e $A \to \alpha$ $\alpha \in (V \cup T)^*$

S = Special variable called start symbol of the given grammer.

→ The language generated CFG is called context free language (CFL).

→ Consider a grammar $G = (V, T, P, S)$ where

$V = \{S\}$; $T = \{a, b\}$; $P = \{S \to asb, S \to ab\}$

Check what type of grammar is this & generate the language described by this grammar.

The given grammar is context free grammar (or) Type-2 Grammar. The language generated

by this grammar is

$L = \{ab, aabb, aaabbb \cdots\}$

$L = \{a^n b^n \mid n \geq 1\}$

2) Construct CFG to generate set of palindromes over i/p alphabet $\Sigma = \{a, b\}$ asa absba

$P = \{S \to asa,$
$S \to b\}$ (or)

$P = \{S \to asa, S \to bsb, S \to \epsilon\}$ - Even palindrome

$P = \{S \to asa, S \to bsb, S \to a|b\}$ - odd "

$P = \{S \to asa, S \to bsb, S \to \epsilon, S \to a|b\}$ - both odd & even palindromes

Note: The grammar generated for odd palindrome is $P = \{S \to asa, S \to bsb, S \to a|b\}$

Even - $P = \{S \to asa, S \to bsb, S \to \epsilon\}$

Construct CFG for the following language

$L = \{0^{2n} 1^m \mid n \geq 0, m \geq 0\}$

$L = \{\epsilon, 00, 0000 \cdots, 1, 11, 111, 1111 \cdots, 001, 00011 \cdots\}$

Non-terminal A generates required no. of 0's & Variable B generates required no. of 1's

so the grammar is $S \to AB$

$A \to 00A|\epsilon$

$B \to 1B|\epsilon$

Derivation tree: or parse tree:

A derivation tree is a "n" ordered tree in which nodes r labelled with d left side of production and in which the children of a node represents its corresponding right side.

Ex: Consider the grammar

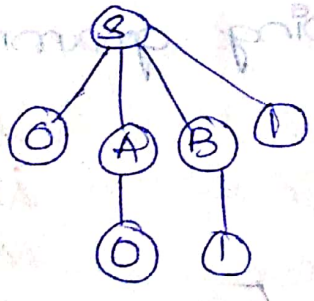
$S \rightarrow 0AB1$

$A \rightarrow 0$

$B \rightarrow 1$

→ Consider the grammar.

The derivation P for the string 0011



More Let $G = \{V, T, P, S\}$ be a context free Grammar, a tree is a derivation tree for the string generated by G if

(i) Every vertex has a label which is a symbol of V U T U {ε}

(ii) The label of the loop verlex is starting symbol of the given grammar (S).

"If a vertex is interior (other than root and leaf node) and has a label A then "A" must be any variable in V.

(iv) If a vertex has a label A belongs to V and its children are labelled (from left to Right) $a_1, a_2 \dots a_n$. then P must contain a production of Ps of the form

$A \to a_1 a_2 a_3 \dots a_n$.

(v) If vertex has labelled ε then that vertex is a leaf vertex and is the only child of its parent.

→ Consider the following grammar

$P = \{S \to aSA/b$

$A \to Ab/a\}$

A⇒a

⇒aSA

⇒aAbA

⇒aAbab

Design (i) Derivation tree

(ii) Left Most Derivation tree

(iii) Right " " "

for the string aababa

$S \to aSA/b$

$A \to Ab/a$

$S \Rightarrow aSA$

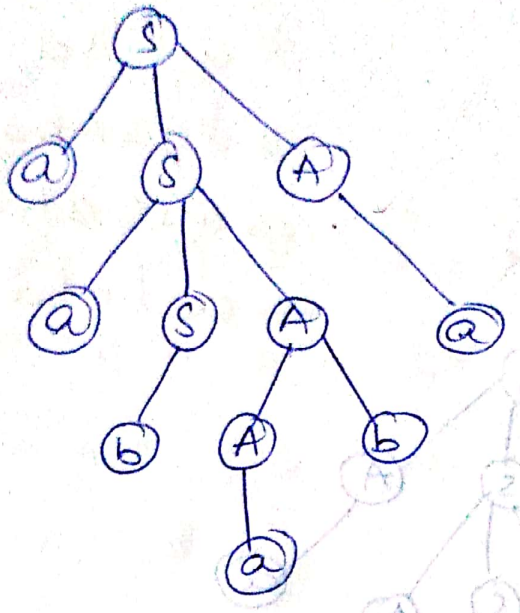
$\Rightarrow aaSAA$

$\Rightarrow aaSAbA$

$\Rightarrow aabAbA$

$\Rightarrow$ aabAba

$\Rightarrow$ aababa.



String aababa.

Left

$S \rightarrow asA/b$ $\Rightarrow S \Rightarrow asA$

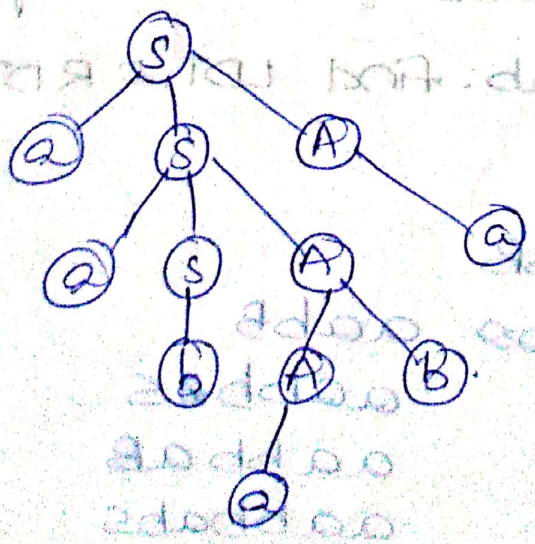
$S \rightarrow Ab/a$ $\Rightarrow aasAA$

$\Rightarrow aabAAb$

$\Rightarrow aabAbA$

$\Rightarrow aababA$

$\Rightarrow aababa$.



Right:

$S \to aSA|b$

$A \to Ab|a$.

$S \Rightarrow aSA$.

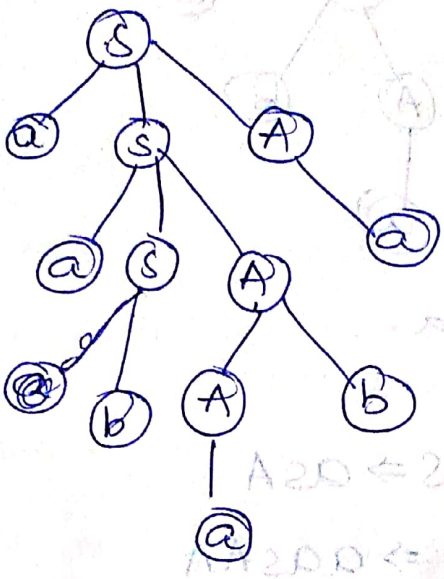
$\Rightarrow aSd$

$\Rightarrow aaSAa$

$\Rightarrow aaSAba$

$\Rightarrow aaSaba$

$\Rightarrow aababa$.



2) Let G be the grammar consisting of

$S \to aB|bA$

$A \to a|as|bAA$

$B \to b|bS|aBB$ for the string

aabbabab. find LDT & RDT

$S \to aB$

$aaBB$

~~$aaBbS$~~ $aabB$

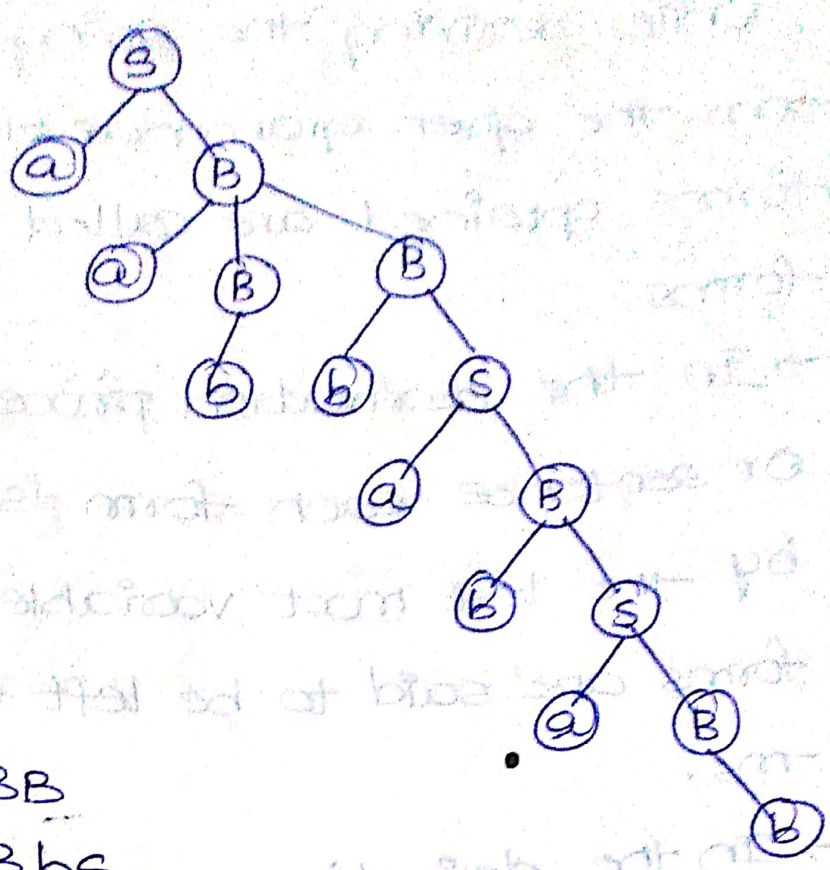
$aabbAS$

$aabbaB$

$aabbabS$

$aabbabaB$

⇒ aabbabab.



RDT

S → aB

a aBB

aaBbS

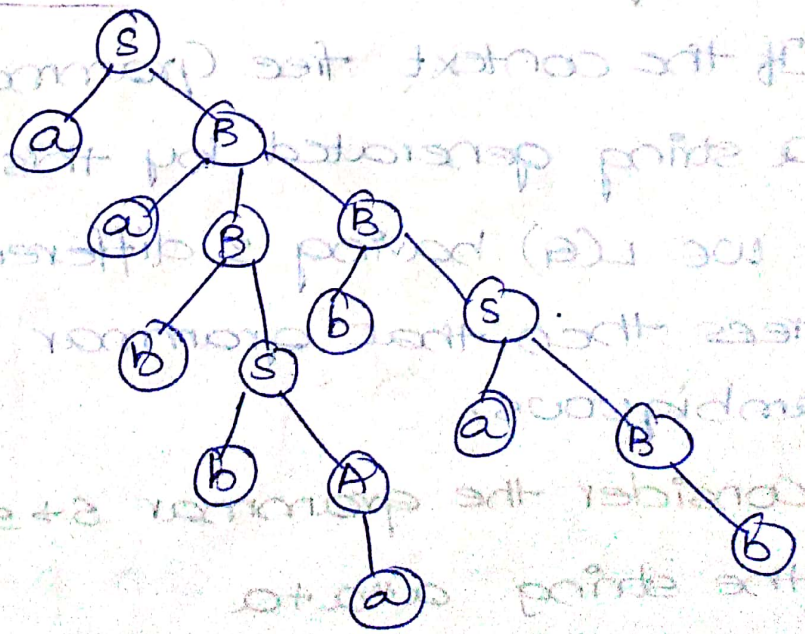
aaBbaB

aaBbab

aabSbab

aabbAbab

aabbabab



Sentential Forms:

While deriving the string or sentence from the given grammar the different forms obtained are called sentential forms.

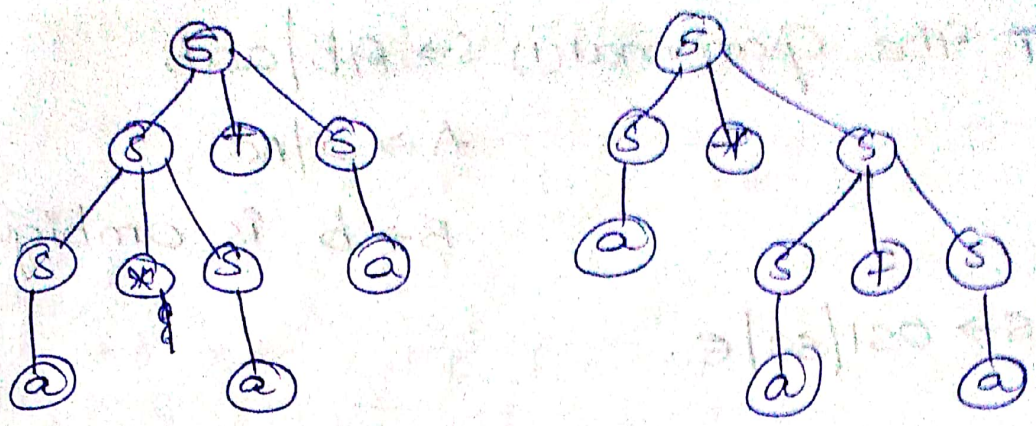
→ In the derivation process of a string or sentence each form is substituted by the left most variable then those forms are said to be left sentential forms.

→ In the derivation process of a string or sentence each form is substituted by the right most variable then those forms r said to be right ~~seqpen~~ sentential forms.

Ambigious Grammar:

If the context free Grammar G exists, a string generated by this grammar we L(G) having 2 different derivation trees then that grammar is said to be ambiguous.

Consider the grammar S→S+S/S*S/a for the string a*a+a



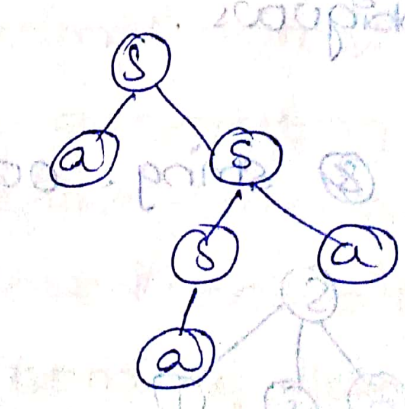
→ As two different derivation trees exists for the string a*a+a so the given grammar is said to be ambiguous.

check the following grammar is ambiguous or not.

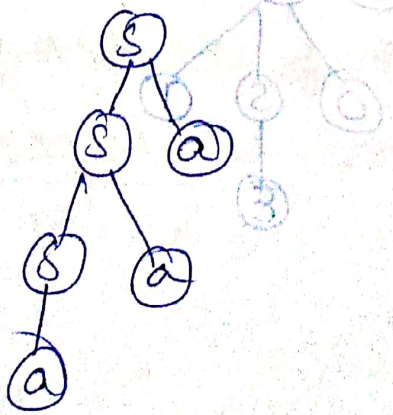
S → as/sa/a.

consider string as aaa.

S → as
asa
aaa



S → Sa
Saa
aaa



for the string aaa 2 different derivation trees exists.

P.T the Grammar(i) $S \to AB \mid aaB$

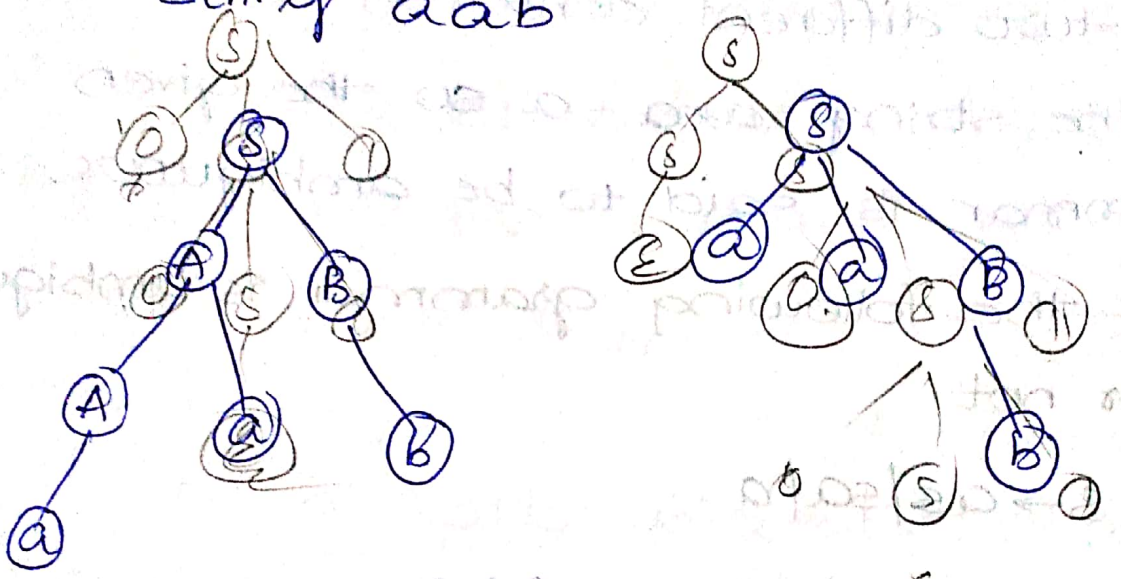
$A \to a \mid Aa$

$B \to b$ is ambiguous

(ii) $S \to 0S1 \mid SS \mid \epsilon$.

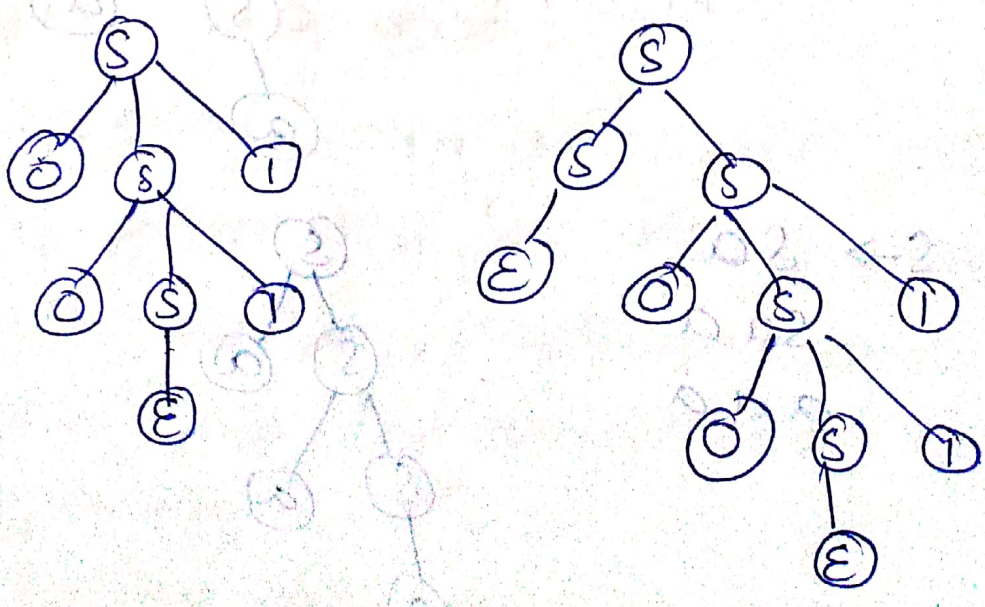
0011 0101

(i) String aab



Ambiguous.

(ii) String - 0011



Applications of Context Free Grammar.

→ In the implementation of YACC parser generator.

YACC (Yet Another compilu compiler)

→ In XML structured definitions

→ To define new programming language syntax rules.

Note: In a context Free grammar without "ε" generating & whose productions are of the form $A \to a\alpha$ where A is any variable 'a' is terminal.

$\alpha \in (V+T)^*$ or $\alpha \in (V \cup T)^*$ is said to be in GNF (Greibach normal form).

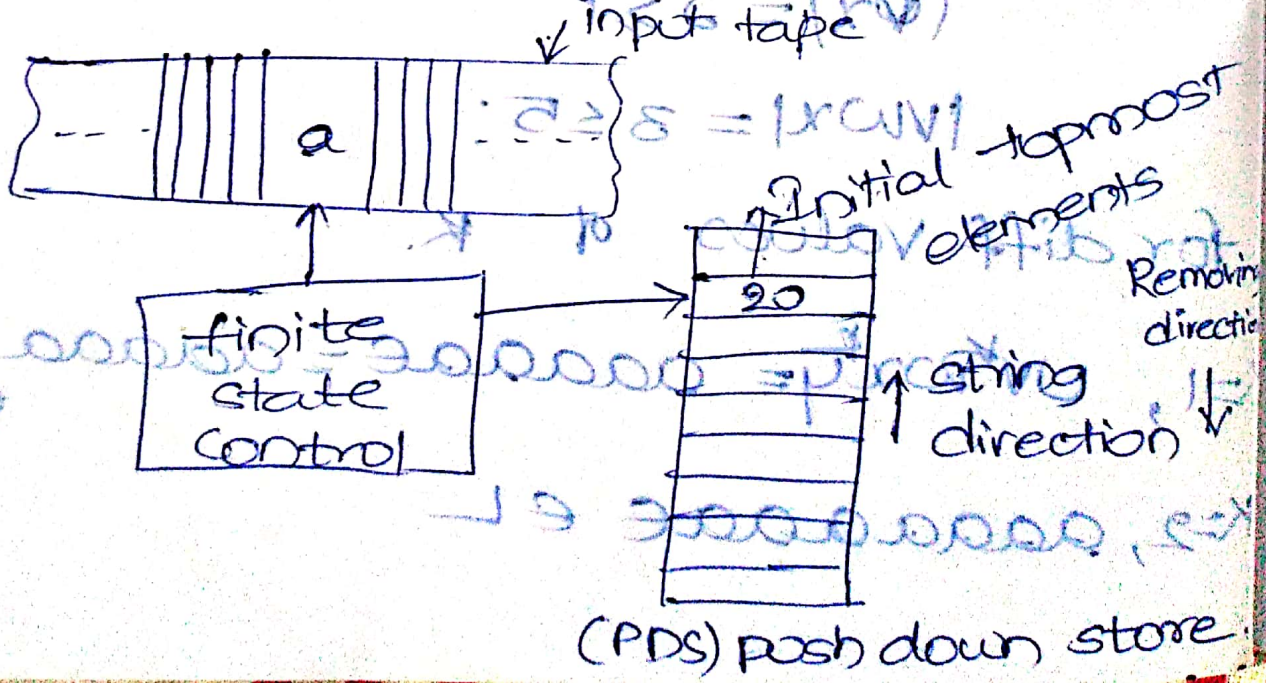
→ In CFG without "ε" generating & whose productions are of the form $A \to BC$ or $A \to a$ where $A, B, C \in V$ and a is terminal is said to be in Chomsky Normal Form.

→ Grammar is a generating device where as automata is a acceptor device.

Push Down Automata (PDA):-

→ It has a read only tape, input alphabet, finite state control, a set of final states & an initial state as in the case of finite automata. in addition to this it has a stack called push down store (PDS). It is a read write push down store. as we add the elements to PDS and remove elements from PDS.

→ Model of Push down Automata:-



A push down automata consists of 7 tuples

$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

where Q = a finite non empty set of states

Σ = a finite non empty set of i/p symbols

Γ = a finite non empty set of push down store (or) stack symbols.

q_0 = is initial state.

z_0 = a special push down store symbol called initial top most element of the push down store or stack.

F = set of final states.

δ = The transition of "δ" is defined as

$\delta : Q \times (\Sigma \cup \epsilon) \times \Gamma \rightarrow Q \times \Gamma^*$

Design PDA which accepts equal no. of a's and b's.

PDA is defined as

$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

$Q = \{q_0\}$, $\Sigma = \{a, b\}$, $\Gamma = \{R, B, G\}$

$q_0 = q_0$, $z_0 = R$, $F = \phi$.

δ is defined as

(i) If the machine is in initial state (q_0), initial top most element of the stack is R.

" for reading a,

→ push "B" into the stack, for reading "b", push "G" into the stack, without changing state.

$\delta(q_0, a, R) = (q_0, BR)$

$\delta(q_0, b, R) = (q_0, GR)$

→ If the topmost element of the stack is B, & reading a, then push "B" into the stack, if the topmost element of the stack is "G" Then

$\delta(q_0, a, B) =$ push B into stack

$\delta(q_0, b, G) =$ " G " "

→ If top of this stack is B & reading b, then pop the element from the stack,

→ if topmost element of stack is G & reading a, then pop the element from stack. without changing any state.

→ For removal of initial topmost element of the stack is

$\delta(q_0, \epsilon, R) = (q_0, \epsilon)$

→ Now, δ is defined as

$\delta(q_0, a, R) = (q_0, BR)$

$\delta(q_0, b, R) = (q_0, GR)$

$\delta(q_0, a, B) = (q_0, BB)$

$\delta(q_0, b, q) = (q_0, qq)$

$\delta(q_0, a, q) = (q_0, \epsilon)$

$\delta(q_0, b, B) = (q_0, \epsilon)$

$\delta(q_0, \epsilon, B) = (q_0, \epsilon)$

① for the string "ababab"

$\delta(q_0, ababab, B)$

$\Rightarrow (q_0, babab, BB)$

$\Rightarrow (q_0, abab, B)$

$\Rightarrow (q_0, bab, BB)$

$\Rightarrow (q_0, ab, B)$

$\Rightarrow (q_0, bB, B)$

$\Rightarrow (q_0, B) \Rightarrow (q_0, \epsilon)$

As the given string contains equal no. of a's & equal no. of b's stack became empty after processing the string. so, the given is accepted.

② for the string "aaabb".

$\delta(q_0, aaa, bb, B)$

$\Rightarrow (q_0, aabb, BB)$

$\Rightarrow (q_0, abb, BBB)$

$\Rightarrow (q_0, bb, BBBB)$

$\Rightarrow (q_0, b, BBB) \Rightarrow (q_0, \epsilon, BB)$

Since the stack is not empty after reading the string, the string is rejected.

3) Construct PDA for the language

$$L = \{wcw^R \mid w \in (0+1)^*\}$$

where w^R is reverse string of w.

(Next paper)

2) Given $L = \{a^n b^n \mid n \geq 1\}$

$\delta(q_0, a, R) = (q_0, BR)$

$\delta(q_0, a, B) = (q_0, BB)$

$\delta(q_0, b, B) = (q_1, \epsilon)$

$\delta(q_1, b, B) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, R) = (q_1, \epsilon)$.

String aabb.

$\delta(q_0, aabb, R) = (q_0, BR)$

$\delta(q_0, abb, R) = (q_0, BBR)$

$\delta(q_0, bb, R) = (q_1, \epsilon, R)$

$\delta(q_1, BR) \rightarrow (q_1, \epsilon)$

Since the stack is empty it is accepted.

String abab $\delta(q_0, abab, R) = (q_0, BR)$

$\rightarrow b \Rightarrow (q_1, \epsilon, R)$

a - the machine is halting.

3) Sol: The string before "c" is exactly reversed & is followed by "c".

→ To implement this language we are considering 2 states. q_0, q_1.

→ State q_0 is continued before reading c, once we read c, it will enter into state q_1.

→ On state q_0 reading symbol 0, B is pushed into the stack irrespective of the top of the stack.

→ Similarly, reading symbol "1", push G into the stack irrespective of the top of the stack

$\delta(q_0, 0, R) = (q_0, BR)$

$\delta(q_0, 0, B) = (q_0, BB)$

$\delta(q_0, 0, G) = (q_0, BG)$

$\delta(q_0, 1, R) = (q_0, G, R)$

$\delta(q_0, 1, B) = (q_0, GB)$

$\delta(q_0, 1, G) = (q_0, GG)$

} Instantaneous descriptors

→ If the machine or PDA is in q_0 state, on reading c, change the state without changing from q_0 to q_1 without

changing the contents of the stack, irrespective of the top the stack.

$\delta(q_0, C, R) = (q_1, R)$

$\delta(q_0, C, B) = (q_1, B)$

$\delta(q_0, C, G) = (q_1, G)$

} IDs

→ On state q_1, reading symbol 0, if the top of the stack is B pop element from the stack.

→ Illy, on state q_1, reading symbol 1, if the top of the stack is G, pop element from the stack.

$\delta(q_1, 0, B) = (q_1, \epsilon)$

$\delta(q_1, 1, G) = (q_1, \epsilon)$

} IDs

To remove the initial top most element of the stack & to be make a stack as empty

$\delta(q_1, \epsilon, R) = (q_1, \epsilon)$.

→ For the string 011C110.

$\delta(q_0, 011C110, R) = (q_0, BR)$

$\delta(q_0, 11C110, B) = (q_0, GBR)$

$\delta(q_0, 1C110, G) = (q_0, GGBR)$

$\delta(q_0, C110, G) = (q_1, GGGBR)$

$\delta(q_0, 110, GGBR) = (q_1, \epsilon, GBR)$

$\delta(q_1, 10, GBR) = (q_1, \epsilon, BR)$

$\delta(q_1, 0, BR) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, R) = (q_1, \epsilon)$

Stack is empty so the string is accepted.

→ Consider the string 001C001, it will be processed as follows.

$\delta(q_0, 001C001, R) \Rightarrow \delta(q_0, 01C001, BR)$

$\Rightarrow \delta(q_0, 1C001, BBR)$

$\Rightarrow \delta(q_0, C001, GBBR)$

$\Rightarrow \delta(q_1, 001, GBBR)$

Halting, after reading c only pop operation should be performed since after q is the top most element we can pop b so machine is halting. so the string is rejected.

- There is no move for $\delta(q_1, 0, G)$ so PDA will halt, but stack contains some elements when PDA is halted so, the string is rejected.

Note: The above PDA's deterministic PDA's because every ID because every ID is generating only one move in the design

NON-DETERMINISTIC PDA.

→ A push down automata is said to be non-deterministic, if any ID is generating more than one move in the design.

→ Design PDA for the Language

$$L=\{ww^R \mid w\in(0+1)^*\}$$

→ It is not possible to design deterministic PDA for this problem

→ We can construct non deterministic PDA as follows:

(i) If the first symbol is either 0 or 1 defenetly it belongs to 'w'. so perform the push operation (if the symbol belongs to w perform push operation, if the symbol belongs to w^R perform the pop operation.

$$\delta(q_0, 0, R) = (q_0, BR)$$

$\delta(q_0, 1, R) = (q_0, GR)$

→ If the symbol just now we read is B/0 (based on the top of the element in stack) & the current symbol it is reading is 1, definitly that 1 belongs to w. so perform the push operation.

→ Illy $\delta(q_0, 1, B) = (q_0, GBR)$

$\delta(q_0, 0, G) = (q_0, BGR)$

→ Illy the symbol just now we read is 1 then the current symbol reading is 0, definitly that 0 belongs to w.

→ If the symbol just now we read is zero & the current reading symbol is zero it is not possible to identify that the current symbol 0 belongs to w or w^R. so, define both operations push & pop.

→ Either one of these 2 operations are executed based on the input string.

→ Illy If the symbol just now we read is 1 & the current input symbol is 1 it is not possible to determine that the current symbol belongs to w or w^R.

So we define both push & pop operations.

$\delta(q_0, 0, B) = (q_0, BB)_P$

$= (q_1, \epsilon)_{POP}$

$\delta(q_0, 1, G) = (q_0, GG)$ (q_0)

$= (q_1, \epsilon)$

→ Change the state of the PDA from q_0 to q_1 when middle of the string is reached. i.e, when 1st pop operation is performed.

→ When PDA in state q_1 perform only pop operations already the middle of the string is reached.

$\delta(q_1, 0, B) = (q_1, \epsilon)$

$\delta(q_1, 1, G) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, R) = (q_1, \epsilon)$

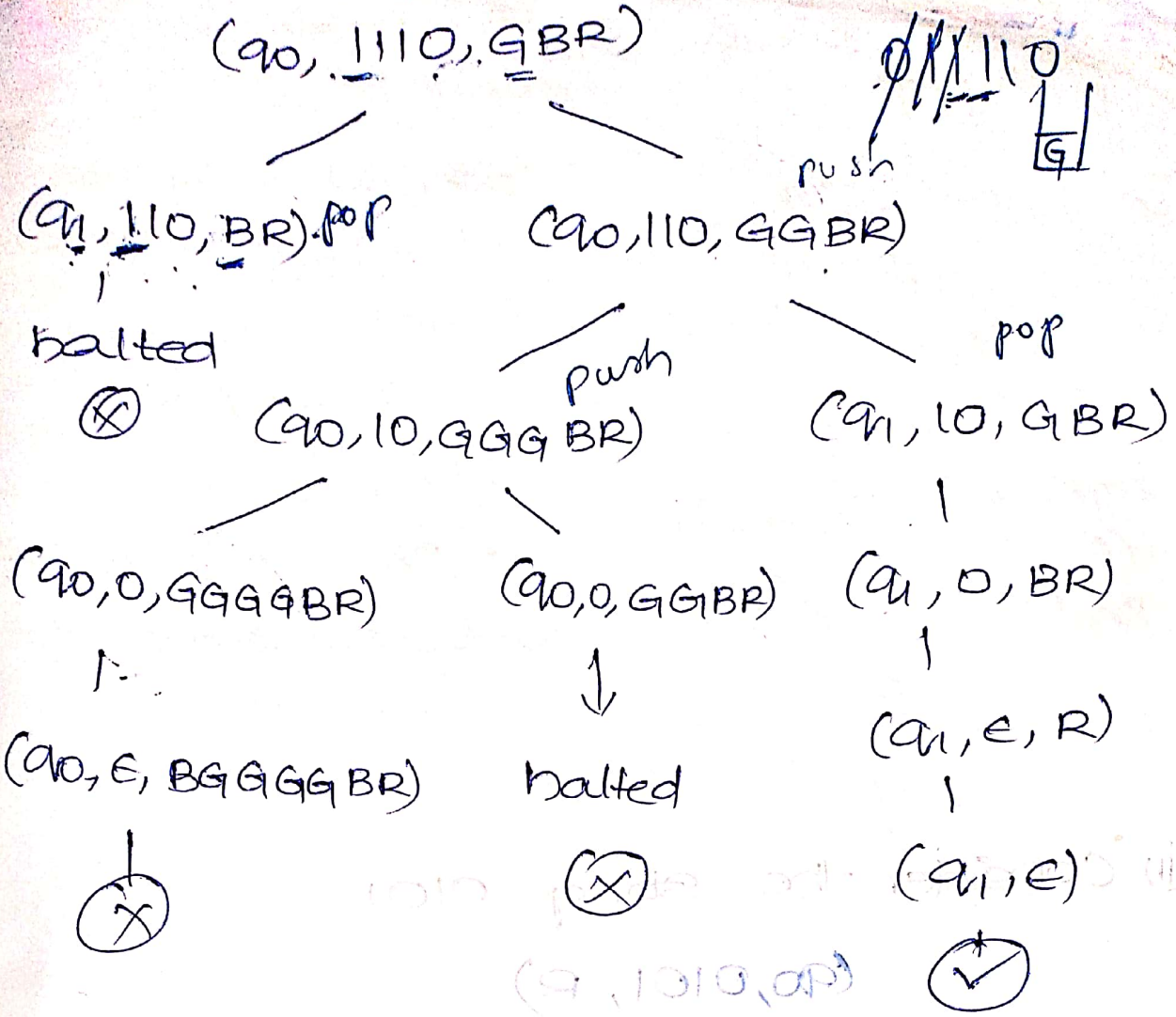
#3

Consider a string :- 01110

$(q_0, 01110, R)$

↓

$(q_0, 1110, BR)$



→ For non deterministic PDA we have to check for all the possible paths. if atleast one path is leading to the empty stack then that string is accepted, otherwise it is rejected.

→ As the given string 01110 is in the form of ww^R it is accepted. in the above processing, one path is leading to the empty stack.

(ii) Consider the string 0101

$(q_0, 0101, R)$

↓

$(q_0, 101, BR)$

↓

$(q_0, 01, GBR)$

↓

$(q_0, 1, BGBR)$

↓

$(q_0, \epsilon, GBGBR)$

↓

$(q_0, \epsilon,$ Halted but stack is not empty. So, the string is rejected. & the string 0101 is not in the form of ww^R.

1. construct context free grammar for a given PDA.

$M = (\{q_0, q_1\}, \{0,1\}, \{R, Z_0\}, \delta, q_0, Z_0, \phi)$

where δ is given as

$\delta(q_0, 0, Z_0) = (q_0, RZ_0)$

$\delta(q_0, 0, R) = (q_0, RR)$

$\delta(q_0, 1, R) = (q_1, \epsilon)$

$\delta(q_1, 1, R) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, R) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, Z_0) = (q_1, \epsilon)$

Sol: Context free Grammar $G = (V, T, P, S)$ can be constructed as follows.

$V = \{S, [q_0 R q_0], [q_0 R q_1], [q_1 R q_0], [q_1 R q_1], [q_0 Z_0 q_0], [q_0 Z_0 q_1] [q_1 Z_0 q_0] [q_1 Z_0 q_1]\}$

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

$S = S$

Productions P can be constructed as

$S \rightarrow [q_0, Z_0, q_0]$

$S \rightarrow [q_0, Z_0, q_1]$

for move $\delta(q_0, 0, Z_0)$ apply the 2nd rule

II Rule:-

$\delta(q, a, A) = (q_1, B_1 B_2 ---- B_m)$

$[q, A, q_{m+1}] \rightarrow a[q_1 B_1 q_2][q_2 B_2 q_3] ---$

$---- [q_m B_m q_{m+1}]$

for each $q_2, q_3 --- q_{m+1} \in Q$.

for move

i) $\overset{q}{\delta}(\overset{a}{q_0}, \overset{A}{0}, \overset{a_1 B_1 B_2}{z_0}) = (q_0, R z_0)$

$[q_0, z_0, q_0] \rightarrow 0[q_0 R q_0][q_0 z_0 q_0]$

$[q_0, z_0, q_0] \rightarrow 0[q_0 R q_1][q_1 z_0 q_0]$

$[q_0, z_0, q_1] \rightarrow 0[q_0 R q_0][q_0 z_0 q_1]$

$[q_0, z_0, q_1] \rightarrow 0[q_0 R q_1][q_1 z_0 q_1]$

ii) $\delta(q_0, 0, R) = (q_0, RR)$

$[q_0, R, q_0] = 0[q_0, R, q_0][q_0, \overset{R}{z_0}, q_0]$

$[q_0, R, q_0] = 0[q_0, R, q_1][q_1, \overset{R}{z_0}, q_0]$

$[q_0, R, q_1] = 0[q_0, R, q_0][q_0, \overset{R}{z_0}, q_1]$

$[q_0, R, q_1] = 0[q_0, R, q_1][q_1 \overset{R}{z_0} q_1]$

iii) $\delta(q_0, 1, R) = (q_1, \epsilon)$

$[q_0, R, q_1] \rightarrow \epsilon$

(iv) $\delta(q_1, 1, R) = (q_1, \epsilon)$

$[q_1, R, q_1] \rightarrow 1$

(v) $\delta(q_1, \epsilon, R) = (q_1, \epsilon)$

$[q_1, R, q_1] \rightarrow \epsilon$

(vi) $\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$

$[q_1, z_0, q_1] \rightarrow \epsilon$

2. Construct an equivalent CFG for the following PDA.

$M = (\{q_0, q_1\}, \{a, b\}, \{z, z_0\}, \delta, q_0, z_0, \phi)$

where δ is given as

$\delta(q_0, b, z_0) = (q_0, zz_0)$

$\delta(q_0, \epsilon, z_0) = (q_0, \epsilon)$

$\delta(q_0, b, z) = (q_0, zz)$

$\delta(q_0, a, z) = (q_1, z)$

$\delta(q_1, b, z) = (q_1, \epsilon)$

$\delta(q_1, a, z_0) = (q_0, z_0)$

We can construct an equivalent context free grammar for the given PDA as follows.

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

$V = \{S\} \cup \{[q_0 z q_0] [q_0 z q_1], [q_1 z q_0], [q_1 z q_1], [q_0 z_0 q_0], [q_0 z_0 q_1], [q_1 z_0 q_0], [q_1 z_0 q_1]\}$

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

$S = S$

P consists of $S \to [q_0 z_0 q_0]$

$S \to [q_0 z_0 q_1]$

Consider $\delta(q_0, b, z_0) = (q_0, z z_0)$

we know that

$\delta(q_1, a, A) = (q_1, B_1 B_2 \dots B_m)$

$[q_1, A, q_{m+1}] \to a [q_1 B_1 q_2] [q_2 B_2 q_3] \dots$

$\dots [q_m B_m q_{m+1}]$

if no elements then,

$[q A q_1] \to a$.

$[q_0 z_0 q_0] \to b [q_0 z q_0] [q_0 z_0 q_0]$

$\to b [q_0 z q_1] [q_1 z_0 q_0]$

$[q_0 z_0 q_1] \to b [q_0 z q_0] [q_0 z_0 q_1]$

$\to b [q_0 z q_1] [q_1 z_0 q_1]$

(ii) $\delta(q_0, \epsilon, z_0) = (q_0, \epsilon)$

$[q_0\ z_0\ q_0] \to \epsilon$

(iii) $\delta(q_0, b, z) = (q_0, zz)$

$[q_0 z q_0] \to b[q_0 z q_0][q_0 z q_0]$

$\hookrightarrow b[q_0 z q_1][q_1 z q_0]$

$[q_0 z q_1] \to b[q_0 z q_0][q_0 z q_1]$

$\hookrightarrow b[q_0 z q_1][q_1 z q_1]$

(iv) $\delta(q_0, a, z) = (q_1, z)$

$[q_0 z q_0] \to a[q_1 z q_0]$

$[q_0 z q_1] \to a[q_1 z q_1]$

(v) $\delta(q_1, b, z) = (q_1, \epsilon)$

$[q_1 z q_0] \to b$

(vi) $\delta(q_1, a, z_0) = (q_0, z_0)$

$[q_1 z_0 q_0] \to a[q_0 z_0 q_0]$

$[q_1 z_0 q_1] \to a[q_0 z_0 q_1]$