PART-A

1) Positive closure :- (+ closure)

Let $\Sigma = \{a\}$

Positive closure $L^+ = \sum_{i=1}^{n} L^i$

$L^+ = \{a, aa, aaa, \ldots\}$.

Positive closure is a set of one or more occurance of a string.

Kleen closure :- (* closure)

Let $\Sigma = \{a\}$

Kleen closure $L^* = \sum_{i=0}^{n} L^i$

$L^* = \{\epsilon, a, aa, aaa, \ldots\}$

Kleen closure is a set of zero or more occurance of a string.

2) Non-Deterministic Finite Automata with ϵ-transition :-

NFA with ϵ-transition is defined as $M = (Q, \Sigma, \delta, q_0, \{F\})$.

$Q \rightarrow$ finite set of states

$\Sigma \rightarrow$ set of input string

$q_0 \rightarrow$ initial state

$F \rightarrow$ set of final state.

$\delta \rightarrow$ is transition function mapping from $Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$.

(q, a), q is the set of all states of p, where a is ϵ or any symbol.

3) Regular Expression :-

* ϕ is a regular expression denote the language $\{\}$ (empty set)

* ϵ is a regular expression denote the language $\{\epsilon\}$

* For any a in Σ, a is a regular expression denote the language $\{a\}$.

* r and s are regular expression denote the language R and S, then $r+s$, rs and r^* are regular expressions denote the language $R \cup S$, RS and R^*.

4) For language accepting the string, it must start with 1 and ends with 0, The regular expression is

$r = 1(0+1)^* 0$.

5) $S \to aB \mid bA$

$A \to a \mid aS \mid bAA$

$B \to b \mid bS \mid aBB.$

$S \Rightarrow aB \quad \because B \to aBB$

$\Rightarrow aaBB \quad \because B \to aBB$

$\Rightarrow aaaBBB \quad \because B \to b$

$\Rightarrow aaabBB \quad \because B \to b$

$\Rightarrow aaabbB \quad \because B \to aBB$

$\Rightarrow aaabbaBB \quad \because B \to b$

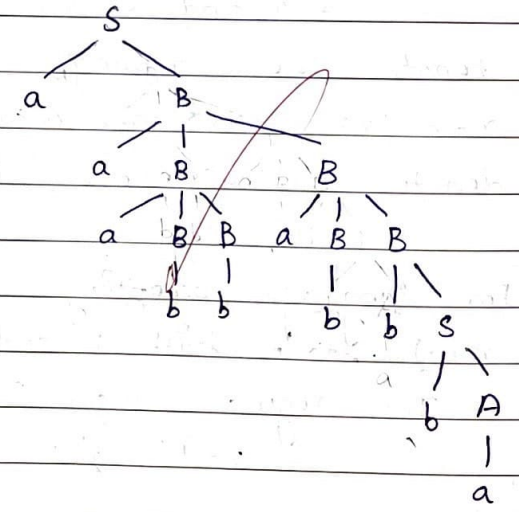
$\Rightarrow aaabbabB \quad \because B \to bS$

$\Rightarrow aaabbabbS \quad \because S \to bA$

$\Rightarrow aaabbabbbA \quad \because A \to a$

$\boxed{S \Rightarrow aaabbabbba}$

Parse tree :-



6) Context Free Grammar :-

Context Free Grammar G_1 is denoted as ~~$M = (Q, \Sigma, F, \delta, q_0, z_0, F)$~~.

$G_1 = (V, T, P, S)$.

where, V = set of non terminals & variable.

T = set of terminals

P = set of Production in the form $A \to \alpha$

where, $A \to$ variable

$\alpha \to$ string of terminals & non terminals

S = special symbol, called start symbol.

Eg :-

$G_1 = (\{E\}, \{+, *, (,), id\}, P, E)$.

where p consists of $E \to E + E$

$E \to E * E$

$E \to (E)$

$E \to id.$

7) Context Free Grammar G_1 which has more than one parse tree is said to be ambiguous.

The equivalent ambiguous is defined as the derivation contain one or more

left most derivation and rightmost derivation.

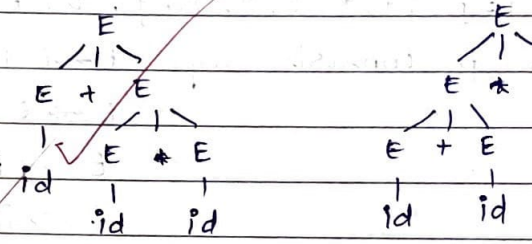
Eg:- $E \to E+E \mid E*E \mid (E) \mid id$.

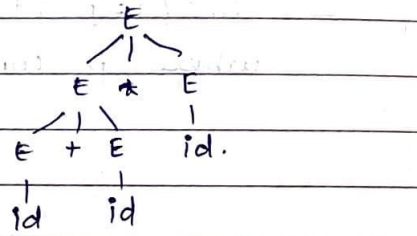
$E \Rightarrow E+E$
$\Rightarrow E+E*E$
$\Rightarrow id+E*E$
$\Rightarrow id+id*E$
$\Rightarrow id+id*id$.

$E \Rightarrow E*E$
$\Rightarrow E+E*E$
$\Rightarrow id+E*E$
$\Rightarrow id+id*E$
$\Rightarrow id+id*id$.

The string id+id*id is derived in 2 different derivation.

Parse tree:-





Given example obtain 2 different parse tree.

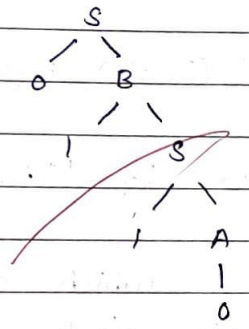
∴ The grammar is ambiguous.

8) $S \to 0B \mid 1A$
$A \to 0 \mid 0S \mid 1AA$
$B \to 1 \mid 1S \mid 0BB$.

Right Most Derivation:-

$S \Rightarrow 0B$ ∵ $B \to 1S$
$\Rightarrow 01S$ ∵ $S \to 1A$
$\Rightarrow 011A$ ∵ $A \to 0$
$S \Rightarrow 0110$

Parse tree:-



9) Two types of moves in PDA:-

* In first type of move, input string is needed. Depending on the input string, state of finite control, ~~stack~~ top of a stack symbol, which

base
~~it~~ to choice the ~~finite control~~ input string.

* Each choice has the ~~stack symbol~~ finite control which has to move next

* Input head is incremented.

$(q, a, z) = (P_1, \gamma_1), (P_2, \gamma_2), (P_3, \gamma_3) \ldots\ldots (P_m, \gamma_m)$

* In second type of move is called ϵ-move, in this input string is not required and input head need not to be increment.

$(q, \epsilon, z) = (P_1, \gamma_1), (P_2, \gamma_2), (P_3, \gamma_3), \ldots, (P_{m-1}, \gamma_m)$

10) Languages handled by PDA:-

1) Languages which are handled by PDA by Empty stack

$N(M) = \{ w \mid (q_0, w, z_0) \vdash^* (P, \epsilon, \epsilon) \}$

2) Languages which are handled by PDA by ~~final~~ final state

$L(M) = \{ w \mid (q_0, w, z_0) \vdash^* (P, \epsilon, \gamma), \text{ for } P \text{ in } F \}$

PART-C.

16) a) G_1 be a grammar

$S \to 0B \mid 1A$

$A \to 0 \mid 0S \mid 1AA$

$B \to 1 \mid 1S \mid 0BB$ string = 0011010 1

Left most derivation:-

$S \Rightarrow 0B$ $\because B \to 0BB$

$\Rightarrow 00BB$ $\because B \to 1$

$\Rightarrow 001BS$ $\because B \to 1S$

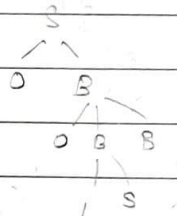
$\Rightarrow 0011S$ $\because S \to 0B$

$\Rightarrow 00110B$ $\because B \to 1S$

$\Rightarrow 001101S$ $\because S \to 0B$

$\Rightarrow 0011010B$ $B \to 1$

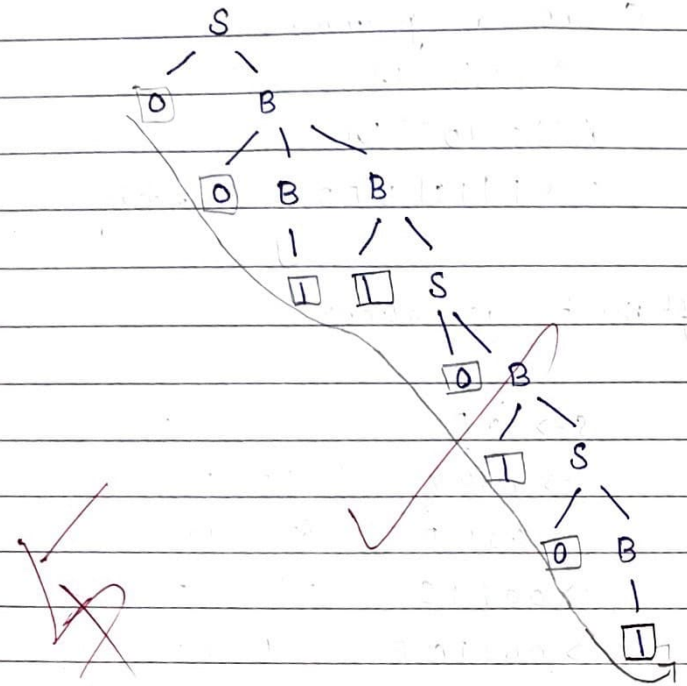
$\Rightarrow 00110101$



$S \Rightarrow 00110101$

By, left most derivation, the string 00110101 is obtained.

derivation tree :-



b) G is the grammar.

S → SbS | a.

① One derivation

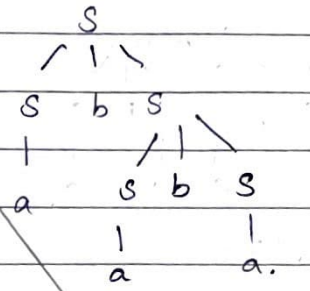
S ⇒ S b S
⇒ S b S b S
⇒ a b S b S
⇒ a b a b S
S ⇒ a b a b a

② Another derivation.

S ⇒ S b S
⇒ S b S b S
⇒ a b S b S
⇒ a b a b S
S ⇒ a b a b a

Derivation tree :-

①



② Another derivation :-

S ⇒ S b S
⇒ a b S
⇒ a b S b S
⇒ a b

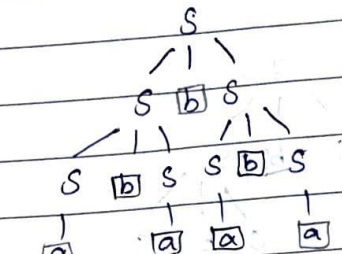
16) b) G is the grammar.

S → SbS | a

First derivation :-

S ⇒ S b S
⇒ S b S b S
⇒ a b S b S
⇒ a b a b S
⇒ a b a b S b S
⇒ ab ab a b S
S ⇒ ab ab ab a

Derivation tree:-



Second derivation:-

$S \Rightarrow S\ b\ S$

$\Rightarrow a\ b\ S$

$\Rightarrow a\ b\ S\ b\ S$

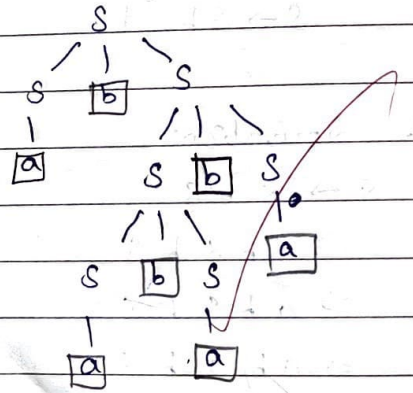
$\Rightarrow a\ b\ S\ b\ S\ b\ S$

$\Rightarrow a\ b\ a\ b\ S\ b\ S$

$\Rightarrow a\ b\ a\ b\ a\ b\ S$

$S \Rightarrow ab\ ab\ aba.$

Derivation tree:-



Grammar G1 has a string abababa which is derived from two different derivation.

Grammar G1 has 2 different parse tree.

∴ G1 is ambiguous.

Explain needed

PART-B.

14) a) Greibach Normal Form:-

$X_1 \rightarrow X_2 X_3$

$X_2 \rightarrow X_3 X_1 \mid b$

$X_3 \rightarrow X_1 X_2 \mid a.$

First we has to check the given form in in CNF, if not convert it into CNF

Then convert CNF into GNF.

For GNF, the left side is less than or equal to the right side.

$X_1 \rightarrow X_2 X_3$ (Acceptable)

$X_2 \rightarrow X_3 X_1$ (Acceptable)

Step 1 :-

$x_3 \rightarrow x_1 x_2$ (Not accepted)

So replace x_1.

$x_3 \rightarrow x_2 x_3 x_2$ (Not accepted).

Again replace it by x_2

$x_3 \rightarrow x_3 x_1 x_3 x_2 \mid b x_3 x_2$

Result :-

$x_1 \rightarrow x_2 x_3$

$x_2 \rightarrow x_3 x_1 \mid b$

$x_3 \rightarrow x_3 x_1 x_3 x_2 \mid b x_3 x_2 \mid a.$

Step 2 :-

Consider which has L.H.S = R.H.S (Same production)

$x_3 \rightarrow x_3 x_1 x_3 x_2 \mid b x_3 x_2 \mid a.$

$x_3 \rightarrow b x_3 x_2 \mid a.$

Now apply lemma 2, a new non-terminal B_3 is added.

$x_3 \rightarrow b x_3 x_2 B_3 \mid a B_3$

$B_3 \rightarrow x_1 x_3 x_2$

$B_3 \rightarrow x_1 x_3 x_2 B_3$

~~Step 3~~ :-

~~Replace x_3 in $x_2 \rightarrow x_3 x_1$ |~~

Resultant :-

$x_1 \rightarrow x_2 x_3$

$x_2 \rightarrow x_3 x_1 \mid b$

$x_3 \rightarrow b x_3 x_2 \mid a B_3 \mid b x_3 x_2 B_3 \mid a B_3$

$B_3 \rightarrow x_1 x_3 x_2 \mid x_1 x_3 x_2 B_3$

Step 3 :-

Replace x_3 in $x_2 \rightarrow x_3 x_1 \mid b.$

$x_2 \rightarrow b x_3 x_2 x_1 \mid a x_1 \mid b x_3 x_2 B_3 x_1 \mid a B_3 x_1 \mid b.$

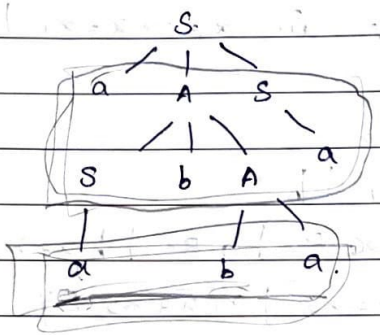
Replace x_2 in $x_1 \rightarrow x_2 x_3.$

$x_1 \rightarrow b x_3 x_2 x_1 x_3 \mid a x_1 x_3 \mid b x_3 x_2 B_3 x_1 x_3 \mid a B_3 x_1 x_3 \mid b x_3.$

Replace x_1 in $B_3 \rightarrow x_1 x_3 x_2$

$B_2 \rightarrow bx_3x_2x_1x_3x_3x_2 \mid ax_1x_3x_3x_2 \mid$
$bx_3x_2B_3x_1x_3x_3x_2 \mid aB_3x_1x_3x_3x_2 \mid$
$bx_3x_3x_2.$

13) b)



Derivation and Language :-

Let $A \rightarrow B$ is a production, where α and γ be any string of symbols.

$\alpha A \gamma \rightarrow \alpha B \gamma$.

In the production $A \rightarrow B$, the string $\alpha A \gamma$ is derived to obtain $\alpha B \gamma$.

Leftmost derivation :-

In the derivation, the production of the string is derived from the left most non-terminal is called leftmost derivation.

Rightmost derivation :-

In the derivation, the production of the string is derived from the right-most non-terminal is called right-most derivation.

Derivation tree :-

* Derivation tree is also known as Parse tree.
* The Derivation is displayed as a Derivation tree.
* In the Derivation tree, the Internal nodes are terminals.
* The leaf node are contain only non-terminals.
* The non-terminals in the leaf nodes combine to form a string.

Left most Derivation:

$S \Rightarrow a\,A\,S$
$\Rightarrow a\,S\,b\,A\,S$
$\Rightarrow a\,a\,b\,A\,S$
$\Rightarrow a\,a\,b\,b\,a\,S$
$\Rightarrow a\,a\,b\,b\,a\,a$

The left most Derivation for the given parse tree is a a b b a a.

Right most Derivation:

$S \Rightarrow a\,A\,S$
$\Rightarrow a\,A\,a$
$\Rightarrow a\,S\,b\,A\,a$
$\Rightarrow a\,S\,b\,b\,a\,a$
$S \Rightarrow a\,a\,b\,b\,a\,a$

The right most Derivation for the given parse tree is a a b b a a

11) a) Convert NFA to DFA:-

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

$\delta(q_0, 0) = \{q_0, q_1\}$

$\delta(q_0, 1) = \{q_1\}$

$\delta(q_1, 0) = \{\}$

$\delta(q_1, 1) = \{q_0, q_1\}$

$\delta =$	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_1\}$
$* q_1$	ϕ	$\{q_0, q_1\}$

Now construct DNF:-

$\delta =$	0	1
$[q_0]$	$[q_0, q_1]$	$[q_1]$
$[q_1]$	$[\phi]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$
$[\phi]$	$[\phi]$	$[\phi]$

Page No :

$\delta([q_0, q_1], 0) = ?$

$\delta(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$

$= \{q_0, q_1\} \cup \{\phi\}$

$= \{q_0, q_1\}$

$= [q_0, q_1]$

$\delta([q_0, q_1], 1) = ?$

$\delta(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$

$= \{q_1\} \cup \{q_0, q_1\}$

$= \{q_0, q_1\}$

$= [q_0, q_1]$

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

$Q = \{q_0, q_1\}$ $\quad q_0 = q_0$

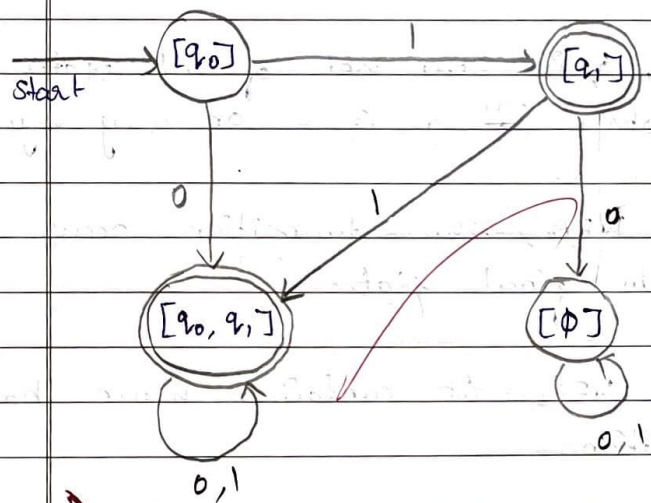
$\Sigma = \{0, 1\}$ $\quad F = \{q_0, (q_0, q_1)\}$

FRANCIS XAVIER ENGINEERING COLLEGE
AN AUTONOMOUS INSTITUTION
NBA Accredited & ISO Certified Institution | TIRUNELVELI - 627 003

Reg. No : 950120120065 Additional Sheet Page No : 2

δ =	0	1
$[q_0]$	$[q_0, q_1]$	$[q_1]$
$[q_1]$	$[\phi]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$
$[\phi]$	$[\phi]$	$[\phi]$

Transition Diagram:-



12) b) Finite Automata with ε-transition:-

~~Finite~~ Non-Deterministic Finite Automata ~~is defined~~ with ε-transition is defined as

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

Q => set of finite state
Σ => set of input string
q_0 => initial state
F => set of final state
δ => transition function mapping from $Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$

(q, a), q is the set of all states of p, where a is ε or any symbol.

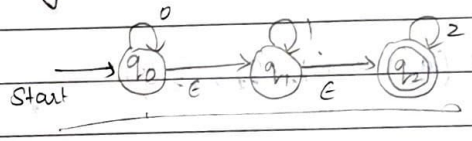
In NFA, the transition occur initial to final state.

In NFA, it containg more than 1 transition.

ε-closure (q_0) :-

ε-closure (q_0) is the set of vertices of P such that there is a path from q to p.

Eg:-



From the above transition, ε-closure of q_0 is $\{q_0, q_1, q_2\}$

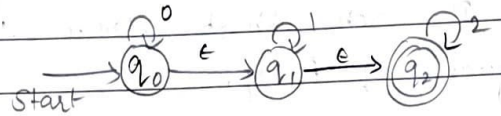
* In this q_0 is alone path for q_0, the ε move is from q_0 to q_0 Therefore ε-closure of q_0 is $\{q_0, q_1, q_2\}$

* ε-moves from q_0 to q_1. so q_1 is in ε-closure (q_0)

* ε-moves from q_0 to q_1 to q_2 so, q_2 is in ε-closure (q_0)

Example :-

Construct ~~NFA~~ NFA ~~with~~



$M = (\{q_0, q_1, q_2\}, \{0, 1, 2\}, \delta, q_0, \{q_2\})$.

ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$

ϵ-closure $(q_1) = \{q_1, q_2\}$

ϵ-closure $(q_2) = \{q_2\}$

$\delta(q_0, 0) = \epsilon$-closure $(\delta(\delta(q_0, \epsilon), 0))$

$= \epsilon$-closure $(\delta(\epsilon$-closure $(q_0), 0))$

$= \epsilon$-closure $(\delta\{q_0, q_1, q_2\}, 0)$

$= \epsilon$-closure $(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$

$= \epsilon$-closure $(\{q_0\} \cup \phi \cup \phi)$

$= \epsilon$-closure (q_0)

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

$\delta(q_0, 1) = \epsilon$-closure $(\delta(\delta(q_0, \epsilon), 1))$

$= \epsilon$-closure $(\delta(\epsilon$-closure $(q_0), 1))$

$= \epsilon$-closure $(\delta\{q_0, q_1, q_2\}, 1)$

$= \epsilon$-closure $(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1))$

$= \epsilon$-closure $(\phi \cup q_1 \cup \phi)$

$= \epsilon$-closure (q_1)

$= \{q_1, q_2\}$

$\delta(q_0, 2) = \epsilon$-closure $(\delta(\delta(q_0, \epsilon), 2))$

$= \epsilon$-closure $(\delta(\epsilon$-closure $(q_0), 2))$

$= \epsilon$-closure $(\delta\{q_0, q_1, q_2\}, 2)$

$= \epsilon$-closure $(\delta(q_0, 2) \cup \delta(q_0, 2) \cup \delta(q_0, 2))$

$= \epsilon$-closure $(\phi \cup \phi \cup q_2)$

$= \epsilon$-closure (q_2)

$= \{q_2\}$

$\delta(q_1, 0) = \epsilon$-closure $(\delta(\delta(q_1, \epsilon), 0))$

$= \epsilon$-closure $(\delta(\epsilon$-closure $(q_1), 0))$

$= \epsilon$-closure $(\delta\{q_1, q_2\}, 0)$

$= \epsilon$-closure $(\delta(q_1, 0) \cup \delta(q_2, 0))$

$= \epsilon$-closure $(\phi \cup \phi)$

$= \phi$

$\delta(q_1, 1)$ = ϵ-closure $[\delta(\delta(q_1, \epsilon), 1)]$
= ϵ-closure $(\delta(\epsilon\text{-closure}(q_1), 1))$
= ϵ-closure $(\delta\{q_1, q_2\}, 1)$
= ϵ-closure $(\delta(q_1, 1) \cup \delta(q_2, 1))$
= ϵ-closure $(q_1 \cup \phi)$
= ϵ-closure (q_1)
= $\{q_1, q_2\}$

$\delta(q_1, 2)$ = ϵ-closure $(\delta(\delta(q_1, \epsilon), 2))$
= ϵ-closure $(\delta(\epsilon\text{-closure}(q_1), 2)$
= ϵ-closure $(\delta\{q_1, q_2\}, 2)$
= ϵ-closure $(\delta(q_1, 2) \cup \delta(q_2, 2))$
= ϵ-closure $(\phi \cup q_2)$
= ϵ-closure (q_2)
= $\{q_2\}$

$\delta(q_2, 0)$ = ϵ-closure $(\delta(\delta(q_2, \epsilon), 0))$
= ϵ-closure $(\delta(\epsilon\text{-closure}(q_2), 0)$
= ϵ-closure $(\delta\{q_2\}, 0)$
= ϵ-closure $(\delta(q_2, 0))$
= ϵ-closure (ϕ)
= ϕ

$\delta(q_2, 1)$ = ϵ-closure $(\delta(\delta(q_2, \epsilon), 1))$
= ϵ-closure $(\delta(\epsilon\text{-closure}(q_2), 1))$
= ϵ-closure $(\delta\{q_2\}, 1)$
= ϵ-closure $(\delta(q_2, 1))$
= ϵ-closure (q_2)
= ϕ

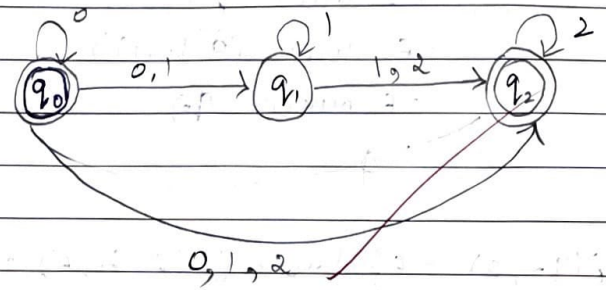
$\delta(q_2, 2)$ = ϵ-closure $(\delta(\delta(q_2, \epsilon), 2))$
= ϵ-closure $(\delta(\epsilon\text{-closure}(q_2), 2))$
= ϵ-closure $(\delta\{q_2\}, 2)$
= ϵ-closure $(\delta(q_2, 2))$
= ϵ-closure (q_2)
= $\{q_2\}$

δ =

	0	1	2
q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$

Page No :

Transition Diagram:-



FRANCIS XAVIER ENGINEERING COLLEGE
AN AUTONOMOUS INSTITUTION
NBA Accredited & ISO Certified Institution | TIRUNELVELI - 627 003

Reg. No :.....9507201065..... Additional Sheet Page No : 4

15) a) Push Down Automata :-

* It contain input tape, finite control and stack.
* Stack has the string of symbols in any alphabets.

Definition :-

Push Down Automata is defined as

$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$.

where Q => set of finite states
Σ => is a alphabet called Input alphabet
Γ => is a alphabet called stack alphabet
δ -> transition function mapping from $Q \times (\Sigma \cup \{\in\}) \times \Gamma \rightarrow$ finite subset of $Q \times \Gamma^*$
q_0 => Initial state
Z_0 => particular symbol, called start symbol.
F => Set of Final state.

ID :-

* ID contains state, input, stack.
* ID is defined as 3 tuples.

(q, w, U)

where q => state of finite control.
w => string of input state.
U => string of stack.

There are 2 types of moves in Push Down Automata :-

* In first type of move, input string is needed. Depending on the input string, state of finite control, top of a stack symbol, which has to choice input string.
* Each choice has a finite control which has to move next
* Input head is incremented

$(q, a, z) = (P_1, \gamma_1)\ (P_2, \gamma_2)\ (P_3, \gamma_3), \ldots (P_m, \gamma_m)$

* In second type of move is called ϵ-moves, in this type,

the input string is not needed and the input head need not to be increment.

$(q, \epsilon, z) = (P_1, \gamma_1)\ (P_2, \gamma_2), \ldots (P_m, \gamma_m)$

Language Accepted by Push Down Automata :-

There are 2 types of language accepted by Push Down Automata.

* By empty stack.

language accepted by PDA is to be the set of all string in which the sequence make a PDA into Empty its stack.

$$N(M) = \{w \mid (q_0, w, z_0) \vdash^* (P, \epsilon, \epsilon)\}$$

* By Final state.

language accepted by PDA is to be the set of all strings which the sequence make a PDA enter into final state.

$$L(M) = \{w \mid (q_0, w, z_0) \vdash^* (P, \epsilon, \gamma),\ \text{for } P \text{ in } F\}$$

Language accepted by Push Down Automata is context free ~~grammar~~ language.

Context free grammar:-

Context free grammar G is defined as (V, T, P, S)

where V => set of non-terminals or Variables

T => set of Terminals.

P => set of production in the form of $A \rightarrow \alpha$.

where A => is a variable.

α => strings of terminals and non-terminals.

S => special symbol called start symbol.

Eg:- $G_1 = (\{E\}, \{+, *, (,), id\}, P, E)$

where P consists of $E \rightarrow E+E$

$E \rightarrow E*E$

$E \rightarrow (E)$

$E \rightarrow id.$

Context Free Grammar consist of 4 types of grammer.

In that Type 3 grammer is accepted by push Down Automata.

The production of the form

$A \rightarrow \alpha.$

where A is a variable.

α is string of terminal and non-terminal.

Context free language is derived from context free grammar.