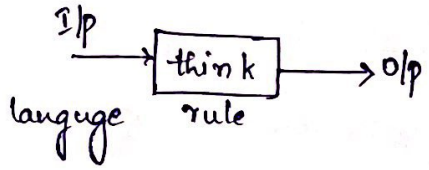
11/6/19

THEORY OF COMPUTATION

1) Language

2) Grammar



Phases of Compiler:-

(2)

* lexical Analysis
* Syntax Analysis
* Sematic Analysis
* intermediate Code generation
* Code Optimization
* Code generation

(3) Error Reporting & Recovery , (1) Symbol table Entry

lexical Analysis :-

Validity check is done by a language called regular language and the table finite grammar (model) by (finite automata)

Syntax Analysis :-

Validity is checked by - CFG (Context free language Grammar)

and grammar for it is Push down automatas.

It checks by sentence.

Sematic Analysis :-

Validity check is done by language called Context Sensitive language and grammar is Linear bounded automatas

Eg:- data type mismatch

12/06/2019

* Mathematical preliminary & Notations :-

1. Set :-

$S = \{a, b, c, d\}$ (or) $S = \{ i , i > 0 , \text{if } i \text{ is ev}$

$a \in S \rightarrow$ belong

$a \notin S \rightarrow$ doesn't belong

* Set operations :-

1. Union
2. Intersection
3. difference
4. Complementation
5. Size

* Proper Subset :- The set where $S_1 \subset S$ which is not equal to actual set.

* Power Set :- $S = \{a, b, c\}$ all possible combinations of subset are called PS.

$\Rightarrow P = \{ \{a\}, \{b\}, \{c\}, \{ab\}, \{bc\}, \{ca\}, \{abc\}, \phi \}$

* Principle of Inclusion and Exclusion :-

$$|A \cup B| = |A| + |B| - |A \cap B|$$

* Relative Complement - b/w 2 Sets
* Absolute Complement - Universal Set and Set

* Cartesian product :-

$S_1 = \{2, 4\}$

$S_2 = \{2, 3, 5, 6\}$

$S_1 \times S_2 = \{ (2,2), (2,3), (2,5), (2,6), (4,2), (4,3), (4,5), (4,6) \}$

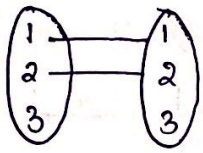
* Partition of a Set :-

A Set is divided into no. of subsets $\Rightarrow S_1 \cup S_2 \cup \dots \cup S_n = S$ and those subsets should be mutually exclusive and none of the S_i sho be empty.

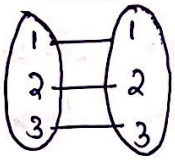
Functions and Relations :- $f : S_1 \rightarrow S_2$

(S_1 : Domain, S_2 : Range)

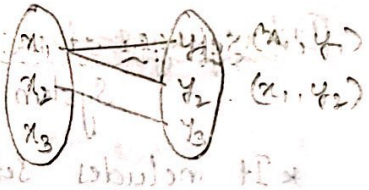
Partial



Total



Each x_i can occur atmost once as the 1st element of the pair.
If this is not satisfied then we will call it as Relations.
Relations are more general than functions.



Properties of Relation:-

1. Reflexive
2. Symmetric
3. Anti-Symmetric
4. Transitive.

① if $\langle a, a \rangle \in R \quad \forall a \in A$

$A = \{1, 2, 3\}$

$R = \{(1,1), (2,2), (3,3)\}$

② if $(a, b) \in R$ then $(b, a) \in R$

③ if $(a, b) \in R$ then $(b, a) \notin R$ then $a = b$

④ if $\langle a, b \rangle \in R$ then $(b, c) \in R \Rightarrow (a, c) \in R$.

Two types of Relation:-

1. Partial Order Relation (POSET)
2. Equivalence

① Reflexive, antisymmetric, transitive should be satisfied.

Eg:- The relation $\leq$ in N $A = \{2, 3, 4\}$

② Reflexive, Symmetric and transitive should be satisfied

Eg:- $x = \{1, 2, 3, 4, 5, 6, 7\}$,

Relation: $x - y$ divided by 3.

14/06/2019

Language and Grammar :-

* 3 basic concepts
  1. Languages
  2. Grammar
  3. Automata

① Languages :-

System suitable for expression of our own ideas facts or conce

* It includes set of Simple Symbols and Rules for their manipulation.

* Alphabets :- Finitie Non-empty set of Symbols are called alphabets. It represented by Σ

* String :- Sequance of Symbols from alphabets is called strings.
  Eg: $\Sigma = \{a, b\}$ then abab, aaabbb are string on Σ
  * Elements inside Σ are always lower case letters and name of String is always capital letter

* Concadination of 2 Strings : $w = a_1 a_2 \cdots a_n$ & $v = b_1 b_2 \cdots b_n$

$$wv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_n$$

$w^R = a_n \cdots a_2 a_1$ * length of String = No. of Symbols in String |w

* ϵ / λ = empty String $|\lambda| = 0$.
  i.e $\lambda w = w\lambda = w$.

* Prefix and Suffix of String :- $w = \{abbab\}$; then prefix = $\{\lambda, a, ab, abb, abba, abbab\}$.
  Suffix = $\{b, ba, bab..\}$

* if w is a string w^n stands for the string obtained by reapting w n times.

* $w^0 = \lambda$ → empty string.

* If Σ is an alphabet then Σ^* denotes the set of strings obtained by concadinating zero or more Symbols from Σ it includes empty string.

* Σ^+ it doesn't include λ.

$$\Sigma^+ = \Sigma^* - \{\lambda\}$$

* If Σ is an alphabet and finite then Σ^* and Σ^+ are infinite because there is no limit on the length of the string.

⊛ Language is defined Generally on a subset of Σ^* or set of strings on an alphabet Σ. A string in L will be called a Sentence of L.

Eg:- If $\Sigma = \{a, b\}$ then $\Sigma^* = \{\lambda, ab, aa, bb, aabb, aaa \cdots\}$

The Set $L = \{a, aa, aaa, aab\}$ is a language on Σ

(or) $L = \{a^n b^n ; n \geq 0\} \Rightarrow L = \{\lambda, ab, aabb, aaabbb \cdots\}$ → infinite.

* Complement of a language:- $\bar{L} = \Sigma^* - L$

* Reverse of a language :- $L^R = \{w^R ; w \in L\}$

* Concadination of two Languages :-

Suppose L_1, L_2 are 2 languages. If 1 concadinate both then $L_1 L_2 = \{xy ; x \in L_1, y \in L_2\}$.

* L^n → L concadinated with itself n times.

* L^0 → $\{\lambda\}$

⁂ Properties of language :-

1. Star closuer = $L^* = L^0 \cup L^1 \cup L^2 \cdots$
2. Additive closuer = $L^+ = L^1 \cup L^2 \cup L^3 \cdots$

⁂ Eg:- If $L = \{a^n b^n : n \geq 0\}$ then $L^2 = \{a^m b^n a^m b^n\}$

Let $L = \{\lambda, a, b\}$.

$L^* = \{aabb\}$, $L = \{ab\}$

$L^2 = \{abaabb, aabbab \cdots\}$.

$\therefore L^2 = \{a^n b^n a^m b^m\}$ Such that $n \geq 0, m \geq 0$. Not m & n are unrelated (≠)

② Grammar:-

Grammar For any language tell us that wheter it is formed correc or Not.

<Sentence> → <noun-phrase> <predicate>

<noun-phrase> → <Article> <noun>

<predicate> → <Verb>

18/06/2019

* Grammar is defined as quadruple. $G = (V, T, S, P)$

V = Set of finite & Variable. (Non-Terminals)

T = Finite set of Terminal Symbols.

S = Starting Variable

P = Finite set of productions.

* All terminal should be lower case and Non-Terminals can be capital letters.

Eg:- Consider the grammar $G = (\{S\}, \{a,b\}, \{S\}, P\}$. Suppose production given as $S \xrightarrow{①} aSb$, $S \xrightarrow{②} \lambda$

Sol:- ① $S \overset{①}{\Rightarrow} aSb$

$\overset{②}{\Rightarrow} a\lambda b$

ab

$aabb$

$aaabbb$

$a^n b^n$

$\Rightarrow (ab)^n$

2 Method :- $S \xrightarrow{①} aB \mid \overset{②}{\lambda}$

$B \xrightarrow{③} Sb$

$G = (\{S,B\}, \{a,b\}, S, P\}$

$S \overset{①}{\Rightarrow} aB$

$\overset{③}{\Rightarrow} aSb$

$\overset{④}{\Rightarrow} a\,aB\,b$

$\overset{③}{\Rightarrow} aaSbb$

$\overset{②}{\Rightarrow} aabb$

* If $w_1 \Rightarrow w_2 \Rightarrow w_3 \Rightarrow \cdots w_n$. Then we say that $w_1 \Rightarrow w_2$ (derives) and we can generally denoted as $w_1 \overset{*}{\Rightarrow} w_n$. By applying n number of prod.

* (*) indicates that the no. of steps can be taken to derive w_n from w

→ By applying production rule in diff order a given grammar can normally generate many strings.

→ Let $G = (V, T, S, P)$, then the set $L(G) = \{w \in T^* : S \overset{*}{\Rightarrow} w\}$ is the language generated by G.

* Problems :-

1. Find the Grammar that generates $L = \{a^n b^{n+1} : n \geq 0\}$. Find set of all productions.

Sol :- $G = (\{\ \}, \{a, b\}, S, P)$

b, abb, aabbb

$S \rightarrow aSbb$ } productions.
$S \rightarrow b$

① $\Rightarrow Sb = b$

② $\Rightarrow Sb = aSbb = abb$

2. Find grammar for $\Sigma = \{a, b\}$ that generates,

(a) All strings with exactly 1a → $S \Rightarrow Sb \mid bS \mid a$.

(b) All strings with atleast 1a

(c) All strings with no more than 3a → $Sa \mid aS \mid aSa \mid Sb \mid bS$

3. Let $\Sigma = \{a, b\}$ find the grammar that generates the language $L = \{a^n b^m ; n \geq 0, m > n\}$.

Ans :- b, abb - - - , ab,

$S = Sb \mid \lambda$ (① ②)

$S = aSb$ (③)

$S = b$ (④)

① $\Rightarrow Sb = b$ ✓

② $\Rightarrow asb \Rightarrow asbb \Rightarrow abb$ ✓

asb ③ asbb ④ abbb - -

19/06/19

4Q. What language does the grammar with these productions generate.

① $S \to Aa$ 3a Aaa Baa
② $A \to B$
③ $B \to Aa$.

$L = \phi$

5) Find Grammar for (i) $L = \{a^n b^{2n} ; n \geq 0\}$

(ii) $L = \{a^{n+2} b^n ; n \geq 1\}$

(iii) $L = \{a^n b^{n-3} ; n \geq 3\}$

6) Construct a grammar where $\Sigma = \{0, 1, c\}$ which accepts all strings of palindrome. use c as delimiter.

7) Find the grammar where $\Sigma = \{a\}$ given languages

(i) $L = \{w : |w| \bmod 3 = 0\}$

(ii) $L = \{w : |w| \bmod 3 > 0\}$

8) What is the language generated by grammar $S \to asa \mid bSb \mid a \mid b$

9) Consider the grammar $G = (\{A, S\}, \{a, b\}, S, P)$ with produ

$S \to aAb \mid \lambda$
$A \to aAb \mid \lambda$

Answers:-

4A: $S \to Aa$
$A \to B$
$B \to Aa$

This string never ends so, $L = \phi$

5A. (i) $L = \{a^n b^{2n} ; n \geq 0\}$

$L = \{\lambda, abb, aabbbb$ $L = a^n b^{2n}$

$S \to aSbb \mid \lambda$

$S \to Sbb$

~~$S \to$~~

$asbb$

$aasbbbb$

$aaasbbbb$

$aaasbbbbbb$ $aaabbbbbbb$

aasbbb aabbb

(i) ~~Ans~~:- $S \rightarrow asbb \mid \lambda$.

(ii)

21/06/19

③ ★ Automata :-

Its an abstract model of a digital computer which includes input file, control unit, output and storage.

* At any given time the control unit is in some internal state and i/p mechanism scanning a particular symbol on the i/p file. The internal state of control unit at the next time step is determined by transition function.

The transition fun produce the next state in terms of current state, current i/p symbol and info currently in temp storage.

2 Types of Automata.

1. Deterministic :- It is one in which each move is uniquely determined by current configuration.

2. Non-Deterministic : It may have several possible view so, we can only predict set of possible actions.

★ Models :-

Automata can be designed in 2 ways.

1. acceptor
2. transducer.

1. Acceptor : An automata whose o/p response is limited to a simple yes or No, is called as acceptor i.e. acceptor. either accepts the given string or rejects it.

2. Transducer : It is a more general automaton capable of producing strings of o/p symbols as o/p.

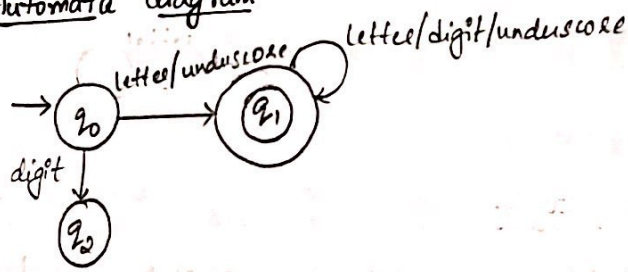
Eg :- 1. Acceptor :- <id> → <letter> <rest> | <underscore> <rest>

<rest> → <letter> <rest> | <digits> <rest> | <undersco> <rest>.

<letta> → a|b|---|z|A|B---Z

<digits> → 0|1|---|9 ; <underscore> → _

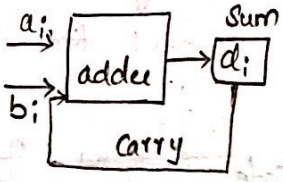
* Automata diagram

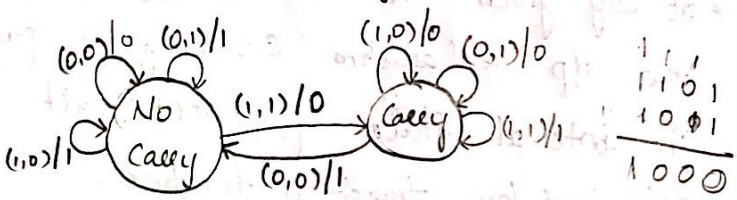


2. Transducer :- binary adder

	0	1
0	0	1
1	1	0

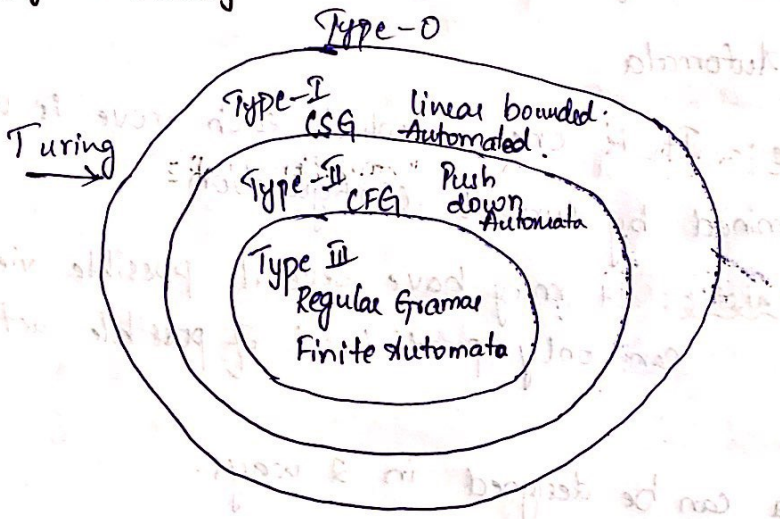
carry = 1





1 1 1
1 1 0 1
1 0 0 1
1 0 0 0

** Chomsky hierarchy :-



** Regular Grammar (Type-III) :-

$S \rightarrow Aa$

(or)

$S \rightarrow aA$

$S \rightarrow \epsilon | \lambda | a$

* 'S' should not appear on the ~~vertex~~ right side of the rule.

* Finite Automata :- (or) Finite Acceptors.

It doesn't include any temp storage. only Acceptor is only since temp storage is not available. it can process only limi input.

Two types of FA:-

1. Deterministic FA (DFA)
2. Non-Deterministic FA (NFA)

1. Deterministic Finite Automata:-

DFA is defined by quintuple (5) where

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

Q = Finite Set of internal States.

Σ = Finite Set of Symbols called input Alphabet.

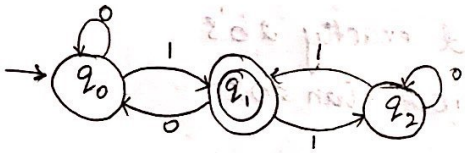
δ = Transition function of type total function.

$$\delta : Q \times \Sigma \rightarrow Q$$

q_0 = Initial State. $q_0 \in Q$

F = Set of Final States. $F \subseteq Q$

Eg:-



→ "Transition graph"

$Q = q_0, q_1, q_2$

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

$F = q_1$

$\delta(q_0, 0) = q_0$

$\delta(q_0, 1) = q_1$

$\delta(q_1, 1) = q_2$

$\delta(q_1, 0) = q_0$

$\delta(q_2, 0) = q_2$

$\delta(q_2, 1) = q_1$

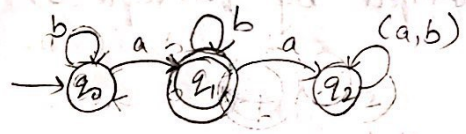
~~0~~, 100, ~~10~~, 111, 1101

This automata doesn't accept the string ends with '0'.

Problems:-

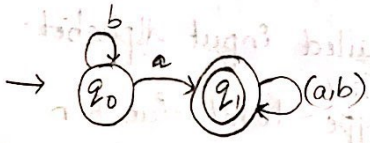
1. Construct DFA which accepts all strings with exactly 1a
2. All strings with atleast 1a.
3. All strings with no more than 3a's
4. All strings starting with 01

1A.



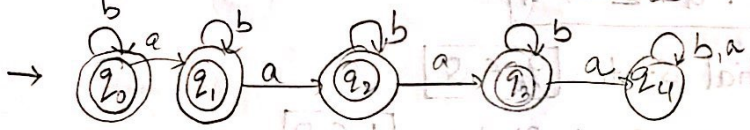
abbb...
abb
ab
abbb

2A.



ab
aab
aaab
...bbb

3A.



5Q. All strings with atleast 1a and exactly 2b's

6Q. All strings with 2a's and more than 2b's

22|06|19

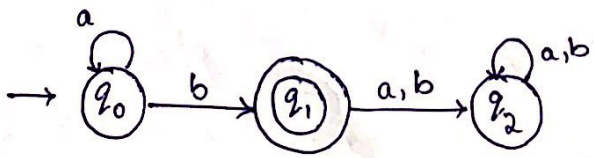
① Extended Transition Function:-

$$\delta^* : Q \times \Sigma^* \rightarrow Q.$$

$$\delta^*(q_0, ab) = q_2 \quad \delta^*(q_0, ab) = \delta(\delta^*(q_0, a)), b$$

$$\delta(q_1, b) = q_2.$$

② Transition table:-



	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_2	q_2

③ Languages and DFA's:-

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

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

$$\boxed{L(M) = \{ w \in \Sigma^* : \delta^*(q_0, w) \in F \}.}$$

* Questions

7Q. Construct DFA that accepts all strings which accepts 2a's & more than one b's.

8Q. All strings state containing 3 consecutive zero. followed by 2 consecutive one's.

9Q. All strings containing even no. of 0's and any num of 1's.

10Q. All strings containing 00 but not triple zero.

11Q. All strings with any no. of a's & b's.

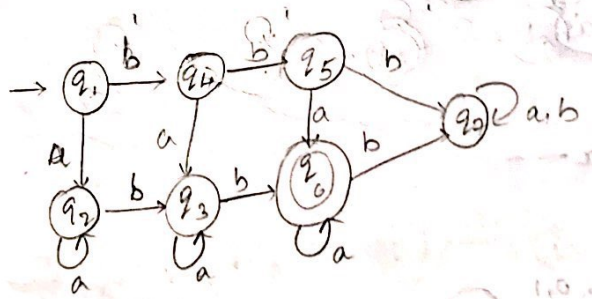
* 12Q. All strings in which the left most symbol differs from right most one. $\Sigma = \{0, 1\}$.

* 13Q. Left most 3 symbols and rightmost 3 symbols are identical

14Q. $L = \{ w : |w| \bmod 3 = 0 \}$

Ans:-

9A.



Non-Deterministic Finite Automata (NFA):-

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

δ = Transition function of type total fun

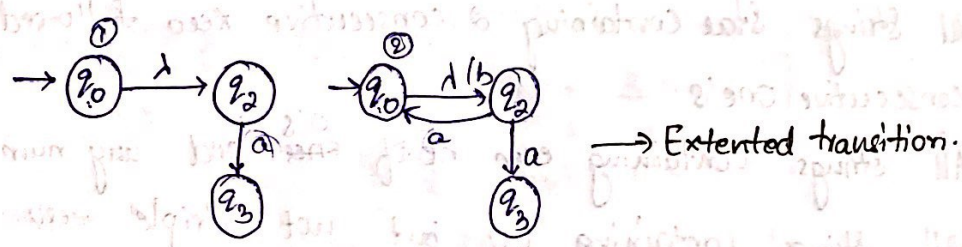
$$\boxed{\delta : Q \times (\Sigma \cup \{\lambda\}) \to 2^Q}$$

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

(2) $\delta(q_1, \lambda) = q_3$

(3) $\delta(q_1, a)$ may be empty.

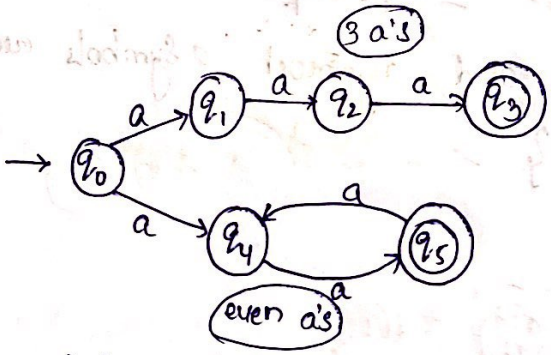
Eg:-



$(q_0, a) = \{q_3\}$ $(q_0, a) = \{q_0, q_2, q_3\}$

$(q_0, a) = \{\text{Null}\}$ for b.

1Eg:-



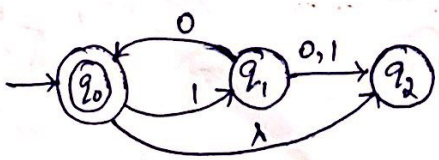
⇒ a a a

⇒ a a

⇒ a a a a ----

⇒ a a a a a a ---

2Eg:-



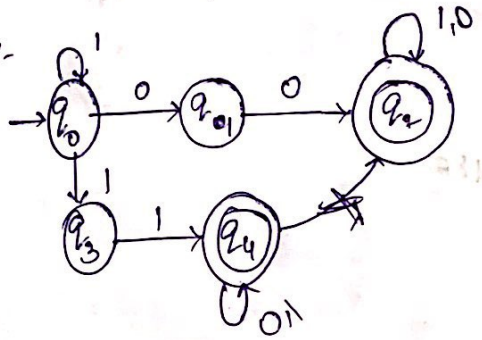
101010- - -

$L = \{(10)^n ; n \geq 0\}$.

*Problems

1Q. Design an NFA that accepts a binary string which have atleast 1pair 00 ,(or) 1 pair 11.

Ans :-



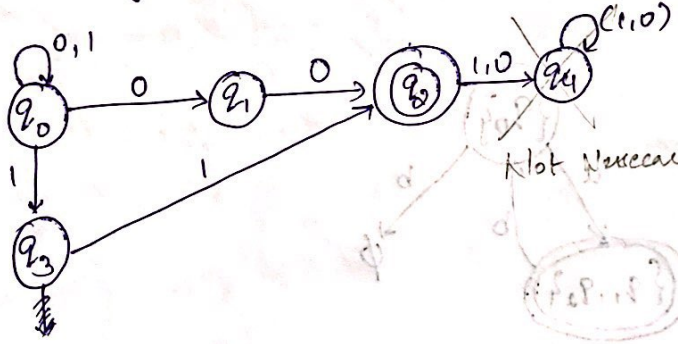
2Q. Design NFA, that ends with 00 or 11.

3Q. Construct NFA with 4 states where $L=\{a^n : n\geq 0\} \cup \{ba^n : n\geq 1\}$ (or)

4Q. $L = \{abaa^n : n\geq 0\} \cup \{aba^n : n\geq 0\}$

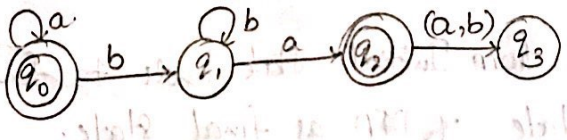
5Q. $L = \{ab, abc\}^*$

2Ans :-



3Ans :-

a^n	b^na
0	ba
a	bba
aa	bb
aaa...	...



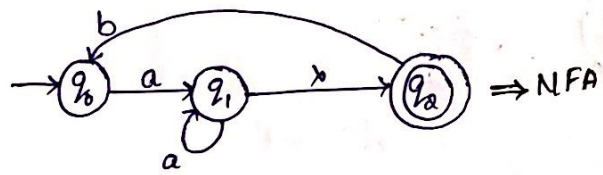
Q. Define a transducer to convert ~~binomial~~ binary to octal.

26/06/2019

* Conversion of NFA to DFA:-

* Every NFA has atleast one DFA.

eg:-



$\delta_N(q_0, a) = \{q_1, q_2\} \rightarrow ①$

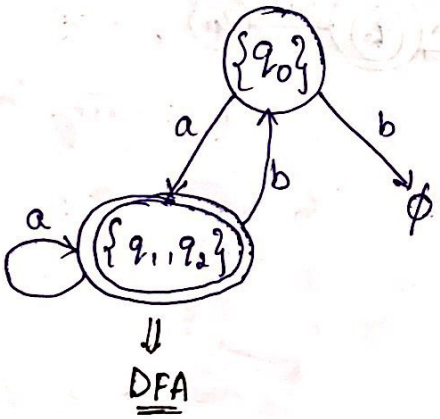
$\delta_N(q_0, b) = \{\phi\} \rightarrow ②$

$\delta_N(q_1, a) = \{q_1, q_2\} \rightarrow ③$

$\delta_N(q_1, b) = \{q_0\} \rightarrow ④$

$\delta_N(q_2, a) = \{\phi\} \rightarrow ⑤$

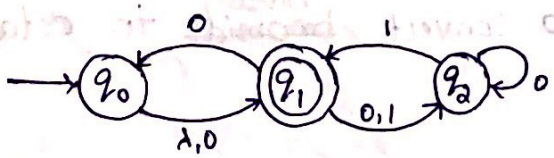
$\delta_N(q_2, b) = \{q_0\} \rightarrow ⑥$



* $\delta_N(\{q_1, q_2\}, a) = \delta_N(q_1, a) \cup \delta_N(q_2, a) = \{q_1, q_2\} \cup \phi = \{q_1, q_2\}$
* $\delta_N(\{q_1, q_2\}, b) = \delta_N(q_1, b) \cup \delta_N(q_2, b) = \{q_0\}$.

Note:- Any λ transition from Initial State leads to final State then mark initial state of DFA as final state.

②



$\delta_N(q_0, 0) = \{q_1, q_2\}$

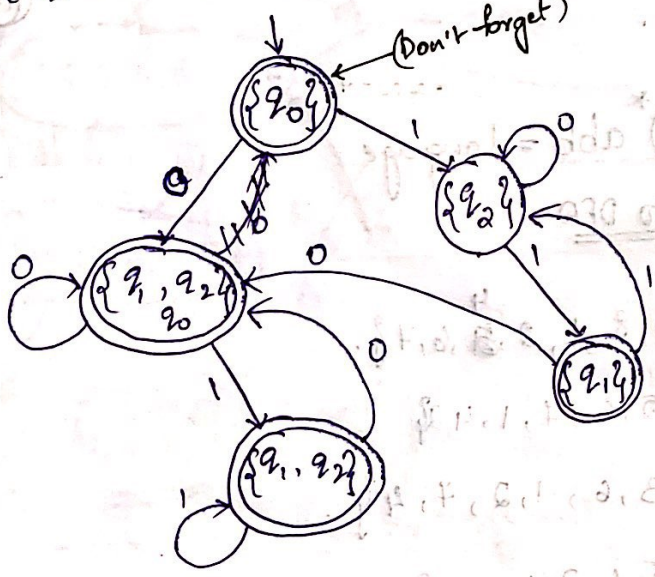
$\delta_N(q_0, 1) = \{\phi, q_2\}$

$\delta_N(q_1, 0) = \{q_0, q_1, q_2\}$

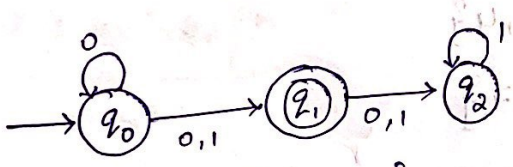
$\delta_N(q_1, 1) = \{q_2\}$

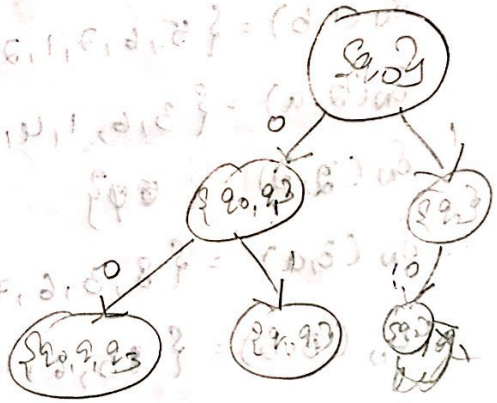
$\delta_N(q_2, 0) = \{q_2\}$

$\delta_N(q_2, 1) = \{q_1\}$



(3a)





$\delta_N(q_0, 0) = \{q_0, q_1\}$.

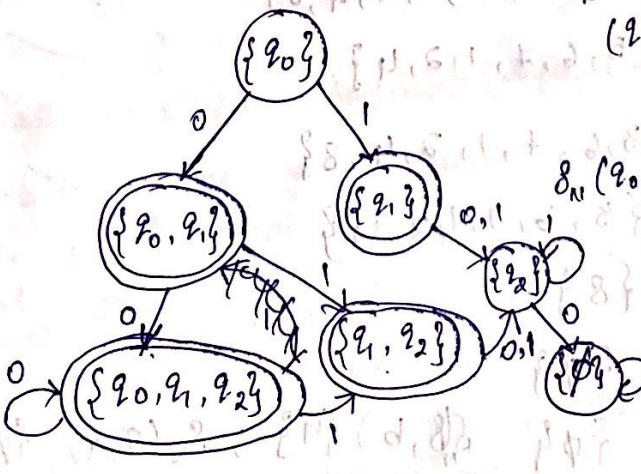
$\delta_N(q_0, 1) = \{q_1\}$

$\delta_N(q_1, 0) = \{q_2\}$

$\delta_N(q_2, 0) = \{\phi\}$

$\delta_N(q_1, 1) = \{q_2\}$

$\delta_N(q_2, 1) = \{q_2\}$.



$\delta_N(\{q_0, q_1\}, 0)$

$(q_0, 0) \cup (q_1, 0) = \{q_0, q_1\} \cup \{q_2\}$

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

$\delta_N(q_0, q_1, 1) = \{q_1\} \cup \{q_2\}$

$= \{q_1, q_2\}$.

q_1, q_2, q_2.

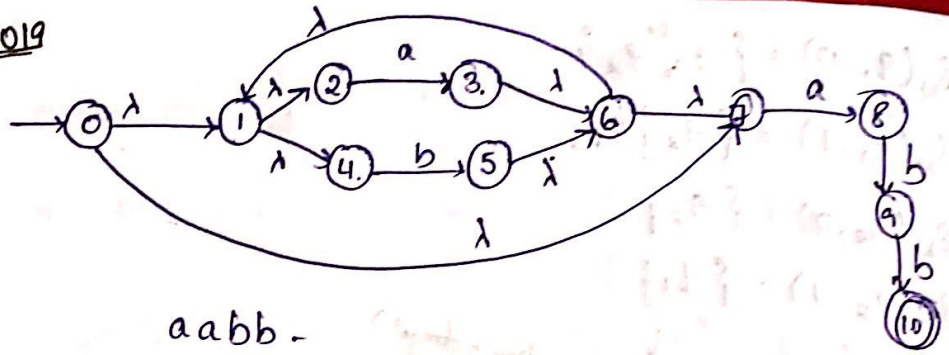
(q_2, ϕ)

q_0, q_1, q_2

q_1, q_2.

27/06/2019

(40)



aabb -

⇒ $(a|b)^* abb$ = Language

Converting NFA to DFA :-

$\delta_N(0,a) = \{3,8,1,2,4,6,7\}$.

$\delta_N(0,b) = \{5,6,7,1,4,2\}$

$\delta_N(1,a) = \{3,6,1,2,7,4\}$.

$\delta_N(1,b) = \{5,6,7,1,2,4\}$

$\delta_N(2,a) = \{3,6,1,4,7,2\}$

$\delta_N(2,b) = \{\phi\}$

$\delta_N(3,a) = \{8,3,6,7,1,4,2\}$

$\delta_N(3,b) = \{5,6,1,4,2,7\}$

$\delta_N(4,a) = \{\phi\}$

$\delta_N(4,b) = \{5,6,7,1,2,4\}$

$\delta_N(5,a) = \{3,6,7,1,2,4,8\}$

$\delta_N(5,b) = \{5,6,7,1,2,4\}$.

$\delta_N(6,a) = \{3,6,7,1,2,4,8\}$

$\delta_N(6,b) = \{5,6,7,1,2,4\}$

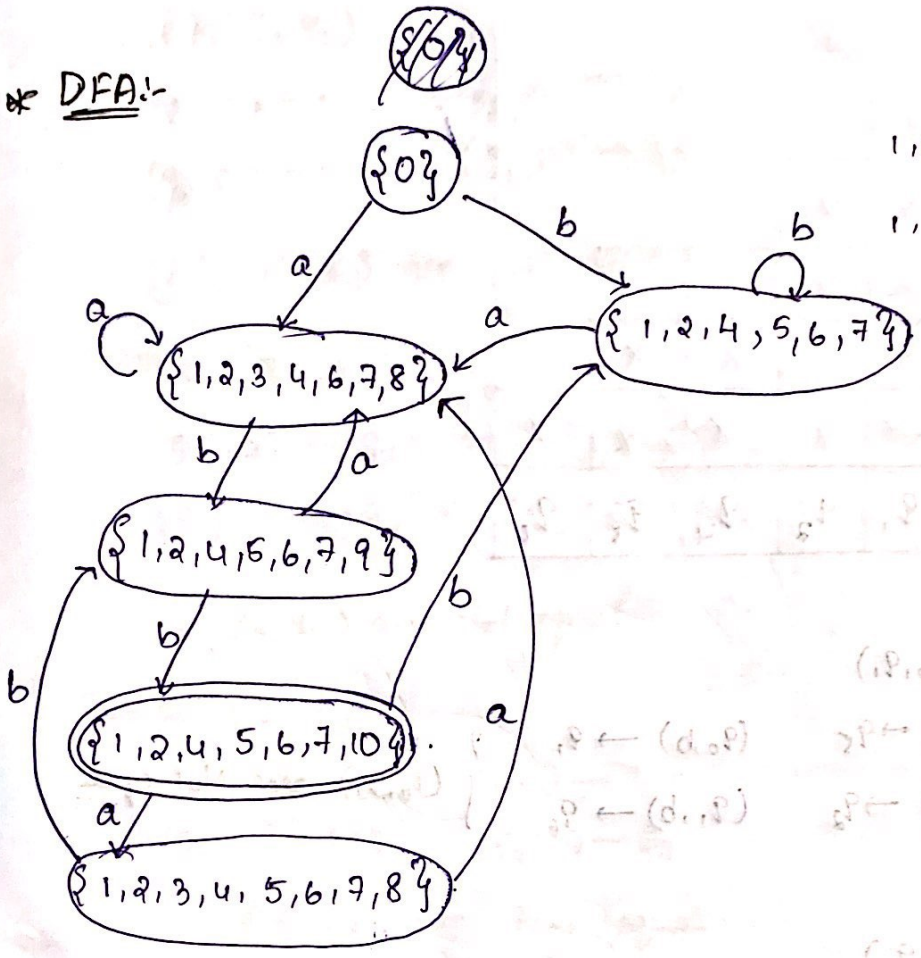
$\delta_N(7,a) = \{8\}$

$\delta_N(7,b) = \{\phi\}$

$\delta_N(8,a) = \{\phi\}$ $\delta_N(8,b) = \{9\}$; $\delta_N(9,a) = \{\phi\}$.

$\delta_N(9,b) = \{10\}$.

* DFA:-



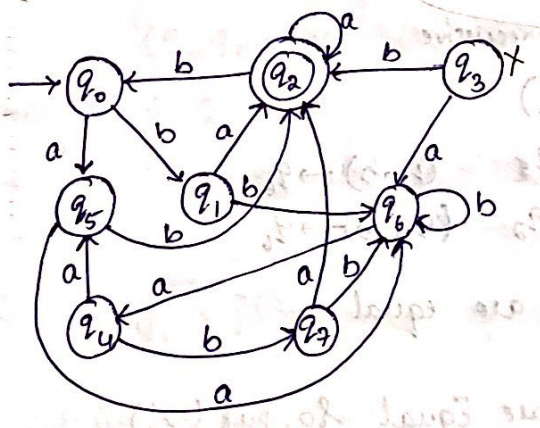
1,2,4,5,6,7,9,10

1,2,3,4,6,7,8
5

28/06/19

** Minimization of DFA:-

Eg:-



1. Remove all unreachable states/Node. ⇒ (q_3 Node).
2. Draw table x-axis, y-axis where x-axis = all but not last stage
   y-axis = all but not first.
3. All final state with non-final put (X).
4. $(a,w) \in F$ $(a,w) \in F$
   $(a,w) \in F$ $(a,w) \notin F$
   Indistinguishable equivalent — Distinguishable Not equivalent. } if equivalent do recursive or put X

q_1	X					
q_2	X	X				
q_3	✓	X	X			
q_5	X	X	X	X		
q_6	X	X	X	X	X	
q_7	X	✓	X	X	X	X
	q_0	q_1	q_2	q_4	q_5	q_6

Consider : (q_0, q_1)

$(q_0, a) \to q_5$ $\quad$ $(q_0, b) \to q_1$

$(q_1, a) \to q_2$ $\quad$ $(q_1, b) \to q_6$

} (q_0, q_1) are Not equal.

Consider :

(q_0, q_4)

$(q_0, a) \to q_5$ $\quad$ $(q_0, b) \to q_1$

$(q_4, a) \to q_5$ $\quad$ $(q_4, b) \to q_7$

recursive

Consider : (q_1, q_7)

$(q_1, a) \to q_2$ $\quad$ $(q_1, b) \to q_6$

$(q_7, a) \to q_2$ $\quad$ $(q_7, b) \to q_6$

$\Rightarrow (q_1, q_7)$ are equal. (✓)

$\Downarrow$

(q_0, q_4) are Equal so, put (✓)

Consider : (q_0, q_5)

$(q_0, a) \to q_5$ $\quad$ $(q_0, b) \to q_1$

$(q_5, a) \to q_6$ $\quad$ $(q_6, b) \to q_2$

$\therefore (q_0, q_5)$ are Not equal.

Consider: (q_0, q_6)

$(q_0, a) \rightarrow q_5$ $(q_0, b) \rightarrow q_1$

$(q_6, a) \rightarrow q_4$ $(q_6, b) \rightarrow q_6$

$\therefore (q_0, q_6)$ are Not equal.

consider: (q_0, q_7)

$(q_0, a) \rightarrow q_5$ $(q_0, b) \rightarrow q_1$

$(q_7, a) \rightarrow q_2$ $(q_7, b) \rightarrow q_6$

$\therefore (q_0, q_7)$ are Not equal.

Consider: (q_1, q_4)

$(q_1, a) \rightarrow q_2$ $(q_1, b) \rightarrow q_6$

$(q_4, a) \rightarrow q_5$ $(q_4, b) \rightarrow q_7$.

$\therefore (q_1, q_4)$ are Not equal.

Consider: (q_1, q_5)

$(q_1, a) \rightarrow q_2$ $(q_1, b) \rightarrow q_6$

$(q_5, a) \rightarrow q_6$ $(q_5, b) \rightarrow q_2$

$\therefore (q_1, q_5)$ are Not equal.

Consider: (q_1, q_6)

$(q_1, a) \rightarrow q_2$ $(q_1, b) \rightarrow q_6$

$(q_6, a) \rightarrow q_4$ $(q_6, b) \rightarrow q_6$. → equal.

$\therefore (q_1, q_6)$ are Not equal

Consider: (q_2, q_4)

$(q_2, a) \rightarrow q_2$ $(q_2, b) \rightarrow q_0$

$(q_4, a) \rightarrow q_5$ $(q_4, b) \rightarrow q_7$.

$\therefore (q_2, q_4)$ are Not equal.

consider: (q_1, q_7)

$(q_1, a) \rightarrow q_2$ $(q_1, b) \rightarrow q_6$

$(q_7, a) \rightarrow q_2$ $(q_7, b) \rightarrow q_6$

$\therefore (q_1, q_7)$ is Equal.

consider: (q_4, q_5)

$(q_4, a) \rightarrow q_5$ $(q_4, b) \rightarrow q_7$

$(q_5, a) \rightarrow q_4$ $(q_5, b) \rightarrow q_2$.

$\therefore (q_4, q_5)$ are Not equal.

consider: (q_4, q_6)

$(q_4, a) \rightarrow q_5$ $(q_4, b) \rightarrow q_7$

$(q_6, a) \rightarrow q_4$ $(q_6, b) \rightarrow q_6$

$\therefore (q_4, q_6)$ are Not equal.

consider: (q_4, q_7)

$(q_4, a) \rightarrow q_5$ $(q_4, b) \rightarrow q_7$

$(q_7, a) \rightarrow q_2$ $(q_7, b) \rightarrow q_6$.

$\therefore (q_4, q_7)$ are not equal.

consider: (q_5, q_6)

$(q_5, a) \rightarrow q_6$ $(q_5, b) \rightarrow q_2$

$(q_6, a) \rightarrow q_4$ $(q_6, b) \rightarrow q_6$

$\therefore (q_5, q_6)$ are not equal.

consider: (q_5, q_7)

$(q_5, a) \rightarrow q_6$ $(q_5, b) \rightarrow q_2$

$(q_7, a) \rightarrow q_2$ $(q_7, b) \rightarrow q_6$

$\therefore (q_5, q_7)$ are not equal.

consider: (q_6, q_7)

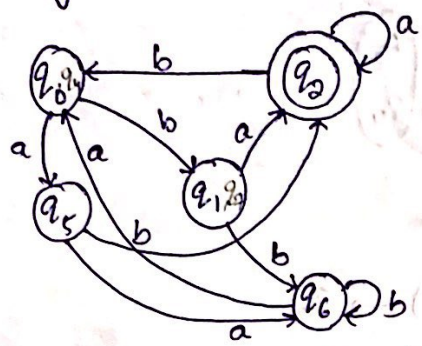
$(q_6, a) \rightarrow q_4$ $(q_6, b) \rightarrow q_6$

$(q_7, a) \rightarrow q_2$ $(q_7, b) \rightarrow q_6$.

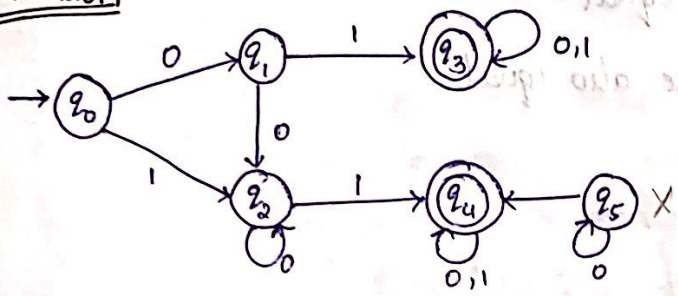
$(q_2, q_4) \rightarrow X$.

$\therefore (q_6, q_7)$ are Not equal.

(*) Redrawing :-



29/06/2019



q_1	X			
q_2	X	✓		
q_3	X	X	X	
q_4	X	X	X	✓
	q_0	q_1	q_2	q_3

(*) Note :- If both states are final we shouldn't mark as (X), we should workout and find.

Consider :- (q_0, q_1)

$(q_0, 0) \rightarrow q_1$ $\quad (q_0, 1) \rightarrow q_2$

$(q_1, 0) \rightarrow q_3$ $\quad (q_1, 1) \rightarrow q_3$

(q_1, q_2) $\quad \therefore$ So, (q_0, q_1) are Not equal.

~~$(q_1, 0) \rightarrow$~~ $\quad$ ~~$(q_2, 0) \rightarrow$~~

~~$(q_1, 1) \rightarrow$~~ $\quad$ ~~$(q_2, 1) \rightarrow$~~

Consider : (q_0, q_2)

$(q_0, 0) \rightarrow q_1$ $\quad (q_0, 1) \rightarrow q_2$

$(q_2, 0) \rightarrow q_2$ $\quad (q_2, 1) \rightarrow q_4$.

(q_0, q_2) are Not equal.

Consider : (q_1, q_2)

$(q_1, 0) \rightarrow q_2$ $\quad$ $(q_1, 1) \rightarrow q_3, q_4$

$(q_2, 0) \rightarrow q_2$. $\quad$ $(q_2, 1) \rightarrow q_3, q_4$

Consider (q_3, q_4)

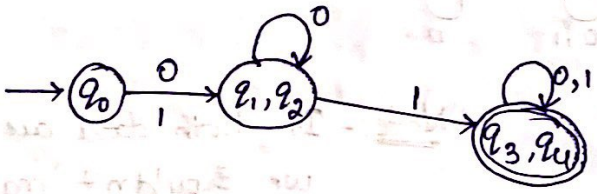
$(q_3, 0) \rightarrow q_3$ $\quad$ $(q_3, 1) \rightarrow q_3$

$(q_4, 0) \rightarrow q_4$ $\quad$ $(q_4, 1) \rightarrow q_4$

$\therefore (q_3, q_4)$ is equal.

$\Rightarrow$ Hence, (q_1, q_2) are also equal.

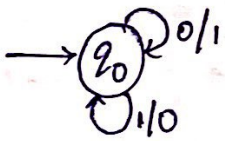
$\Rightarrow$ Drawing output:-



★ Transducer Problems:-

Q) Consider Finite state transducer to compute 1's complement.

Ans.

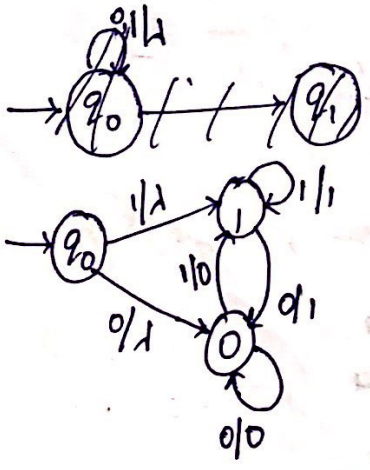


2Q. Design a unit delay transducer for input 0 & 1

Eg:. If I/p : 1011 ✓

O/p : λ101 ✓

Ans



③. Design a transducer for the input which reads the binary form which is divisible by 3. If the num is divisible by 3 then ans is 0. or else output is 1.

④ $L = \{ a^i b^j / i, j > 0 \}$. Design a transducer, if the strings are accepted the result is 1 else the result is zero.

⑤ Design a transducer to convert binary to octal ✓

* DFA:- (acceptor)

Q. A man travels with wolf, Goat and Cabbage and he wants to cross a river from east to west. A row boat is available only large enough for man along with any one of position. Wolf eats goat if left alone together. Goats eats cabbage if left alone together. How can man cross without loss.

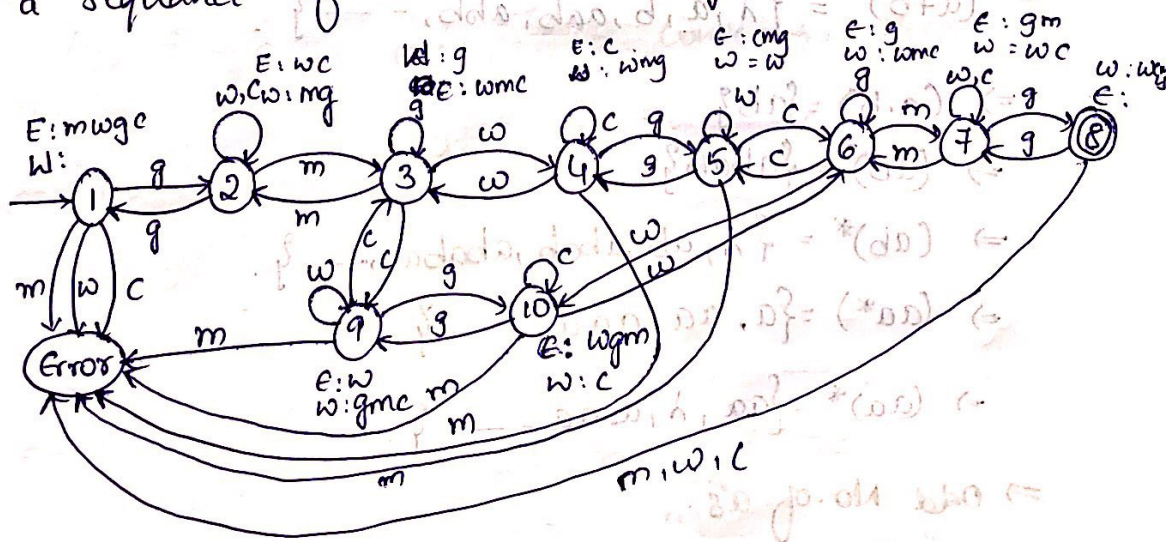
Hint:- 4 moves can be encoded as 4 symbols

1. Man crosses with wolf as (w)
2. " " " Goat as (G)
3. " " " Cabbage as (C)
4. " " " Nothing as (N).

$\Sigma = \{ w, G, C, N \}$.

Then a sequance of moves is a string such as gnwgcng

Sol:-



4/07/2019

* Regular Expression:-

Describing Regular language through a notation called regular expression.

→ The notation involves a combination of strings of symbols from some alphabets Sigma, paranthesis [(•)], +, ·, *

* There are Two types of Regular Expressions:-

1. Primitive
2. Composite.

1. Primitive:- ϕ, λ and $a \in \Sigma$

2. Composite:- $r_1, r_2, r_1+r_2, r_1 \cdot r_2, r_1^*(\cdot r_1)$

Important:-

* (, ∪, +) Same [All are same which gives same output].

(a, b)

(a ∪ b)

(a + b)

$(a+b)^2 \Rightarrow (a+b)(a+b)$

$\Rightarrow (aa, ab, ba, bb)$

$(a+b)^* = \{\lambda, a, b, aab, abb, ---\}$.

$\Rightarrow (a.b) = \{ab\}$

$\Rightarrow (ab)^2 = \{abab\}$

$\Rightarrow (ab)^* = \{\lambda, ab, abab, ababab, ---\}$.

$\Rightarrow (aa^*) = \{a, aa, aaa, ---\}$.

$\Rightarrow (aa)^* = \{aa, \lambda, aaaa ---\}$.

⇒ Odd No. of a's.

1Q. ~~Odd No. of a's~~ Construct regular Exp for language which accepts only a / b

Ans:-

$r = a+b$

2Q. Regular Expression for language ab alone

$r = a \cdot b$.

3Q. R.E for language which ends with one over the set $\{0,1\}$

Ans:- $r = (0+1)^* \cdot 1$

(or)

$r = (0^* \cdot 1^*)^* \cdot 1$

4Q. Given $r = (a+b)^* (a+bb)$

Ans ↓ all comb of a&b

$\Rightarrow L(r) = \{$ all Strings ~~with~~ ending with a or bb $\}$.

5Q. Given $r = (aa)^* (bb^*)^* . b$

Ans ↓ aa → all ↓ bb → all

$L(r)$ = Strings with even no. of a's and odd b's

$L(r) = \{ a^{2n} b^{2m+1} ; n \geq 0, m \geq 0 \}$.

6Q $\Sigma = \{0,1\}$ give a RE such that $L(r) = \{ w \in \Sigma^* \}$ where w has atleast one pair of consecutive zeros.

Ans:- $r = [(00)^+ + (0,1)^*]^+$

0100

0001 0100 01

7Q Find regular Exp for $L(r) = \{ a^n b^m, n \geq 3, m \text{ is even} \}$.

Ans:- $r =$

(ii) $L(r) = \{ a^n b^m, (m+n) \text{ is even} \}$

(iii) $L(r) = \{ a^n b^m, n \geq 4, m \leq 3 \}$.

(iv) $L(r) = \{ w \in \{0,1\}^* \text{ where } w \text{ has no pair of consecutive zeros} \}$

Q. Give $r \in$ for $L(r) = \{ w : |w| \bmod 3 = 0 \}$

5/07/19

Chomsky Hierarchy:-

Type 0, 1, 2, 3 [Grammar]

Type 0 (or) Recursively Renewable Grammar:-

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

Eg:- $S \rightarrow ACaB$

$Bc \rightarrow acB$

$CB \rightarrow DB$

$aD \rightarrow Db$

Type 1 Grammar:-

Contex Sensitive Grammar.

$\alpha A \beta \rightarrow \alpha \gamma \beta$

$A \rightarrow$ Non-teeminal

$\alpha, \beta, \gamma \in (T \cup V)^*$

$\gamma \rightarrow$ Can't be empty

$S \rightarrow \epsilon$ → Is possible only if S doesn't appear on the right hand side of any value.

Eg:- $AB \rightarrow AbBC$

$A \rightarrow bcA$

$B \rightarrow b$

3. Type 2 Grammar:-

Context free Grammar.

$A \rightarrow \gamma$

$A \rightarrow$ Non-terminals

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

4. Type-3 Grammar:-

Regular Grammar.

$S \rightarrow Aa$ (or) $S \rightarrow aA$.

$S \rightarrow \Sigma / \lambda$

$S \rightarrow a$.

* Right side either one / λ / combined with a should be there.

→ This is done by finite automata.

$S \rightarrow \Sigma / \lambda$

$S \rightarrow asb|bsa$ → middle linear → only for LG not for R.G

* Right side either one / λ / combined with 2 should be there.

→ This is done by Finite automata.

17/7/19

** Relation-ship b|w Regular Expression and Regular Language :-

$RL \leftrightarrow RE$

Def:- Language is Regular if it is accepted by 'DFA'.
Because of the equivalence of NFA's and DFA's a language is accepted by NFA is also called as Regular language.

Q. How to construct NFA from a Regular Expression?

Sol: *Rule :- (Thompson's Rule)

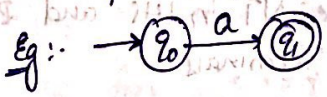
① NFA accepts ϕ (empty)

Eg: 

② NFA accepts $\{\lambda\}$

Eg: 

③ NFA accepts any alphabets ⇒ {a}

Eg:- 

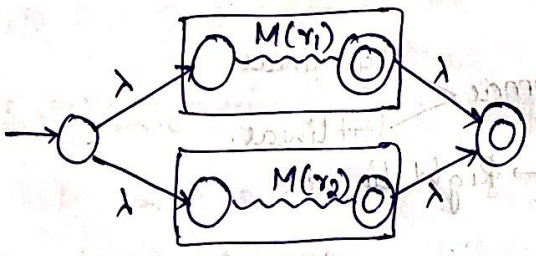
④ Schematic representation of an NFA accepting L(r) (Regular language)

Eg:- 

→ (Schematic Representation.)

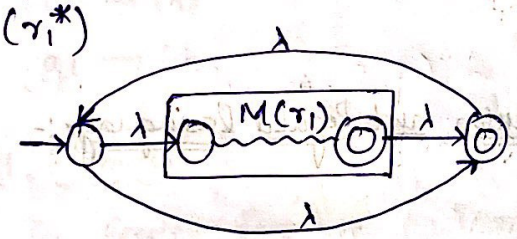
Ex:-

② ⇒ $L(r_1 + r_2)$



③ $L(r_1 r_2)$

④ $L(r_1^*)$

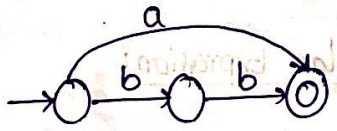


*Problems:-

1Q. Find an NFA that accepts L(r), where $r = (a+bb)^*(ba^*+\lambda)$

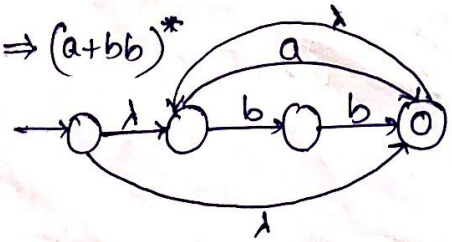
(r_1 = m, r_2 = n)

Sol:- $L(r_1)$ = abb, bba, λ, aabb, abbb - - -



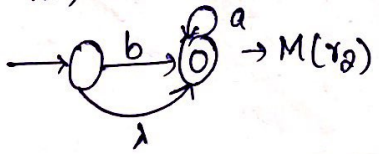
$= (a+bb) \rightarrow M(r_1)$

$\Rightarrow (a+bb)^*$



$\rightarrow M(r_1)^*$

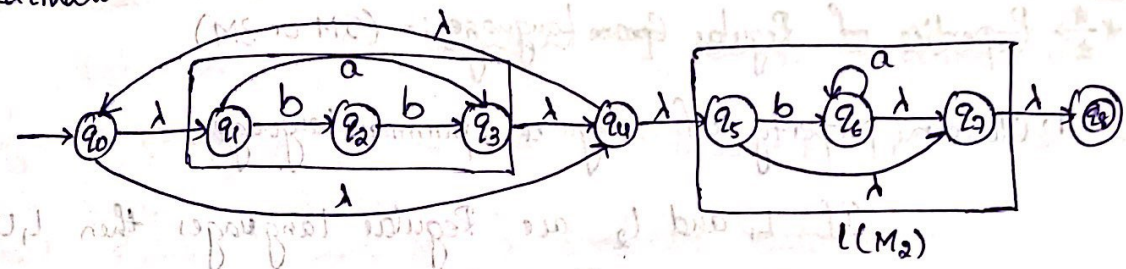
$\Rightarrow (ba^*+\lambda)$



$\rightarrow M(r_2)$

Concatinate $M(r_1)^* \cdot M(r_2)$.

$\Rightarrow$



2Q. Find an NFA for (i) $L(aa^* + aba^*b^*)$.

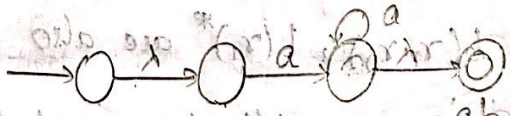
(ii) $L((a+b)^* b(a+bb)^*)$

(iii) $L(ab^*aa + bba^*ab)$

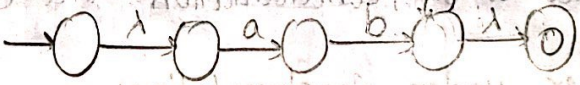
Sol: (i) $L(aa^* + aba^*b^*)$

$\{a, aa, aaa, \cdots, ab, aba, abab, abaab, ababb, abaabb \cdots$
$aab, aaaba\}$

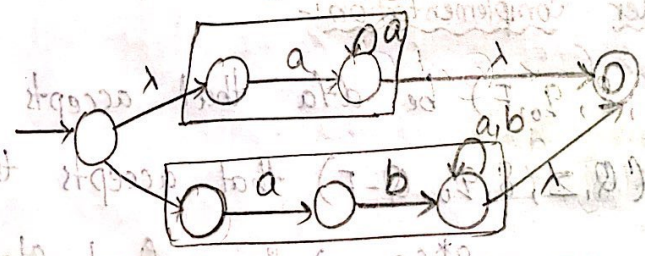
$L(r_1) = (aa^*) \Rightarrow$



$L(r_2) = aba^*b^* \Rightarrow$

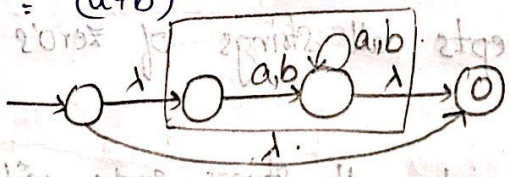


$\Rightarrow$

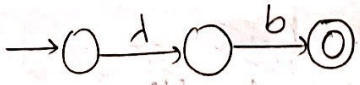


(ii) $L((a+b)^* \cdot b(a+bb)^*)$.

$L(r_1) = (a+b)^*$



$L(r_2) = b(a+bb)^*$



19/7/19

Properties of Regular Gram Language :- (5M or 3M)

(1) Clouser property of Regular Grammar/Language :-

If L_1 and L_2 are Regular languages then $L_1 \cup L_2$, $L_1 L_2$, $L_1 \cap L_2$, $\overline{L_1}$, L_2^* will also produce regular language ; it is clouser under the operations.

* Proof :-

If L_1 and L_2 are regular then their exists r_1 and r_2 such that $L_1 = L(r_1)$ and $L_2 = L(r_2)$

By def :- $r_1 + r_2$, $r_1 \cdot r_2$, r_1^* are also regular expression

$\Downarrow$

$\therefore L(r_1 + r_2)$, $L(r_1 \cdot r_2)$, $L(r_1)^*$ are also regular languages.

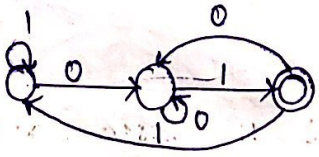
$\therefore$ Clouser under union, concadination and (*) clouser.

To show clouser under Complementation :-

Let $M = (Q, \Sigma, \delta, q_0, F)$ be dfa that accepts the lang then the dfa for $\overline{M} = (Q, \Sigma, \delta, q_0, Q-F)$ that accepts the langua where the transitions are $\delta^*(q_0, \omega)$ is a final state we $\delta^*(q_0, \omega) \in Q-F$ and $\omega \in \overline{L_1}$

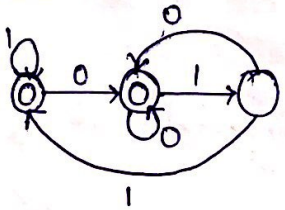
Eg :- Design a dfa that accepts all strings of zero's and one's doesn't end with 01.

Sol :- Construct dfa that accepts all strings ends with 01 and do complement of it.



→ ends with 01.

Complementation :-



→ doesn't end with 01.

Closure under Intersection:-

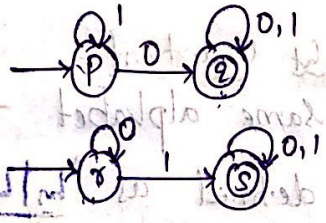
Let $L_1 = L(M_1)$ and $L_2 = L(M_2)$ where $M_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$ and $M_2 = (P, \Sigma, \delta_2, P_0, F_2)$ are dfa's [as Σ is same for M_1, M_2 we can do $\cap$]

Construct from M_1 and M_2 a combined automaton.

$\Rightarrow \hat{M} = (\hat{Q}, \Sigma, \hat{\delta}, (q_0, P_0), \hat{F})$, where $\hat{Q} = P \times Q \to (q_i, P_j)$

$\hat{F} = (q_i, P_j)$ where $q_i \in F_1$ & $P_j \in F_2$; $\hat{\delta} = ((q_i, P_j), a) = (q_k, P_l)$

Eg:- Consider dfa's

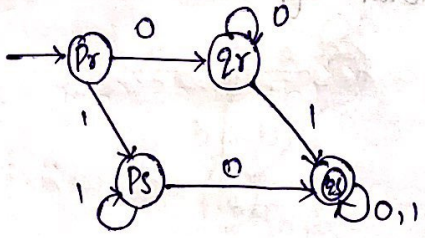


$\hat{Q} = \{Pr, qr, Ps, qs\}$; $\hat{\delta} = (Pr, 0) : qr \mid (qr, 0) : qr$

$(Pr, 1) : Ps \mid (qr, 1) : qs$

$\to (Ps, 0) : qs \mid (qs, 0) : qs$

$(Ps, 1) : Ps \mid (qs, 1) : qs.$



[only if (qs) are both final states mark it as final state]

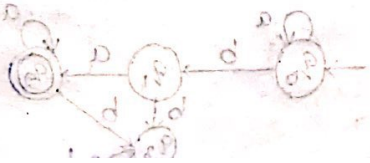
Other Way:- $\bar{L}_1, \bar{L}_2$

$\bar{L}_1 \cup \bar{L}_2$

$\Rightarrow \boxed{\overline{\bar{L}_1 \cup \bar{L}_2} = L_1 \cap L_2}$

Show that the family of RL is closed under difference :-

$\Rightarrow \boxed{L_1 - L_2 = L_1 \cap \bar{L}_2}$



23/7/19

* Regular Language closed under Reverse:-

In the transition graph make the initial vertex as final vertex and final as initial vertex and reverse the direction on all the edges. (L^R)

V.V.Imp

* Regular Language closed under Homomorphism:-

(It is easy it seems Learn it on your own.)

Imp

* Regular Language closed under Right Quotient:-

Let L_1 and L_2 be languages on same alphabet then the right quotient of L_1 with L_2 is defined as L_1/L_2

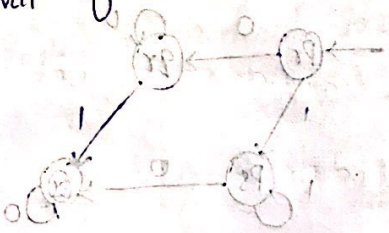
$$L_1/L_2 = \{ x : xy \in L_1 \text{ for some } y \in L_2 \}$$

→ To form the Right quotient of L_1 with L_2, take all strings in L_1 that have a suffix belonging to L_2.

→ Every such string after removal of this suffix is called as L_1/L_2

Eg:- let $L_1 = L(a^*baa^*)$

$L_2 = L(ab^*)$ find L_1/L_2

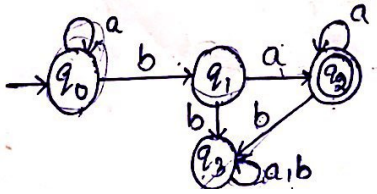


$L_1 = \{ ba, aba, abaa, aaba, aabaa \cdots baaa \cdots \}$

$L_2 = \{ a, ab, abbb, abb, abbb \cdots \}$

$L_1/L_2 = \{ b, ab, aba, \cdots \}$

Step1:- draw dfa for L_1



Step2:- Find from each state collect the languages and do intersection with L_2.

$L(M_0) \cap L_2 = \phi$ [ie there is no intersection with L_2].

$L(M_1) \cap L_2 = \{a\} \neq \phi$

$L(M_2) \cap L_2 = \{a\} \neq \phi$

$L(M_3) \cap L_3 = \phi$

$L(M_0) = \{aba, aaba, abaa, --\}$.

$L(M_1) = \{a, aa, aaa --\}$.

$L(M_2) = \{a, --\}$

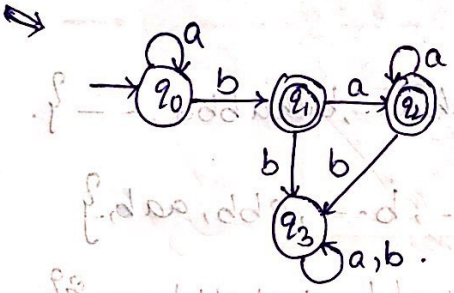
$L(M_3) = X$.

Step3:- If $L(M_i) \cap L_2 = \phi \Rightarrow$ then don't change anything.

$L(M_i) \cap L_2 \neq \phi \Rightarrow$ if it's not final state then make it as final state.

" $\Rightarrow$ if it's final leave (don't change).

$L(M_i) \cap L_2 = \phi \Rightarrow$ if M_i is final state then make it as non-final state



Q. Let $L_1 = L(a^*baa^*)$, $L_2 = L(aba^*)$ find $L_1 | L_2$.

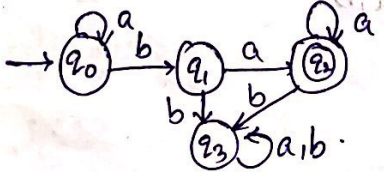
Sol:-

$L_1 = \{ba, aba, abaa, aabaa --\}$

$L_2 = \{ab, aba, abaa, --\}$.

$L_1 | L_2 = \{a, --\}$.

Step1: dfa for L_1



Step 2:- $L(M_0) = \{aba, aaba, abaa, --\}$.

$L(M_1) = \{a, aa, --\}$

$L(M_2) = \{a, aa, --\}$

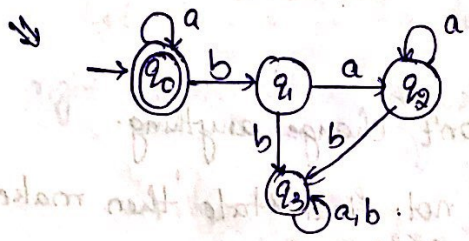
$L(M_3) = X$.

$L(M_0) \cap L_2 = \{aba, abaa, --\} \neq \phi$

$L(M_1) \cap L_2 = \{a, aa, --\} \neq \phi$

$L(M_2) \cap L_2 = \{a\} \neq \phi$

$L(M_3) \cap L_2 \; X$.



Q. $L_1 = babab^*a^*$ Find $L_1 | L_2$

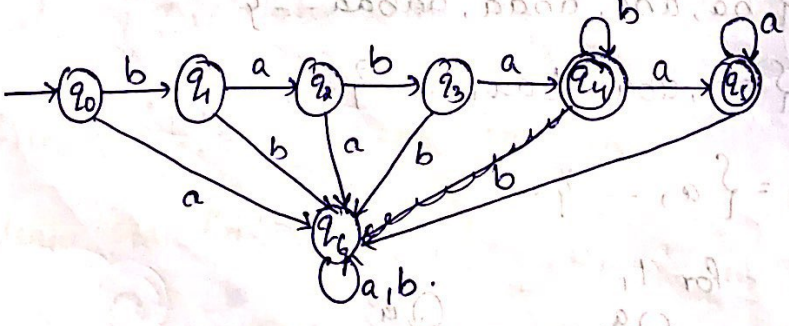
$L_2 = a^*b^*$

Sol:- $L_1 = \{baba, babab, babaa, bababba, --\}$.

$L_2 = \{\lambda, ab, a, b, a--, b--, abb, aab, --\}$

$L_1 | L_2 = \{bab, baba, bababb, bababbb-- \}$

① ⇒ dfa for L_1.



⑥ $L(M_0) = \{ababb^*, bbabb^*a^*\}$.

$L(M_1) = \{abab^*a^*\}$.

$L(M_2) = \{bab^*a^*\}$

$L(M_3) = \{ab^*a^*\}$

$L(M_4) = \{b^*, a^*, b^*a^*\}$

$L(M_5) = \{a^*\}$

$L(M_6)$ X.

$L(M_0) \cap L_2 = \{\phi\}$

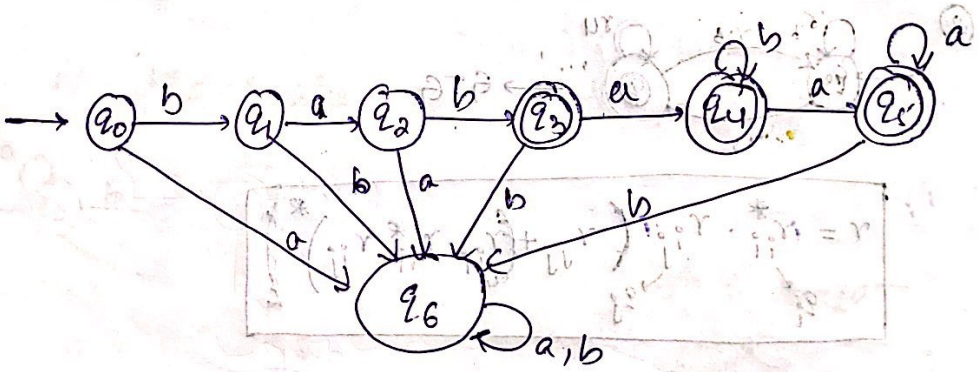
$L(M_1) \cap L_2 = \phi$

$L(M_2) \cap L_2 = \phi$

$L(M_3) \cap L_2 \neq \phi$

$L(M_4) \cap L_2 \neq \phi$

$L(M_5) \cap L_2 \neq \phi$



24/7/19 ★ Homomorphism:-

Suppose Σ & Γ are alphabets then a function $h: \Sigma \rightarrow \Gamma^*$ is called "Homomorphism".

→ Homomorphism is a substitution in which a single letter is replaced with a string.

→ If $w = a_1, a_2, \ldots, a_n$ then $h(w) = h(a_1)\, h(a_2) \ldots h(a_n)$

Eg: Let $\Sigma = \{a, b\}$ and $\Gamma = \{a, b, c\}$ and h is defined as $h(a) = ab$, $h(b) = bbc$, then find $r = (a + b^*)(aa)^*$

Sol:- $r = (ab + bbc^*)(abab)^*$

just Substitue h(a) in place of a and h(b) in place of b.

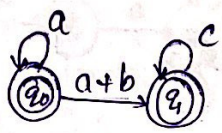
Regular Expression For Regular Languages :-

There are several ways to find the regular expression from regular languages.

1. GTG (Generalized Transition Graph)

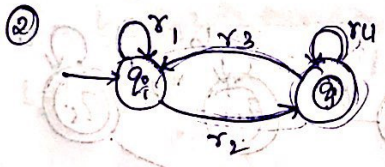
* **GTG:-** It is a transition graph whose edges are labeled with Regular Expressions.

Eg:- ①



$RE = a^*(a+b)c^* \quad a^* + [a^* + (a+b) + c^*]$

②



→ GTG

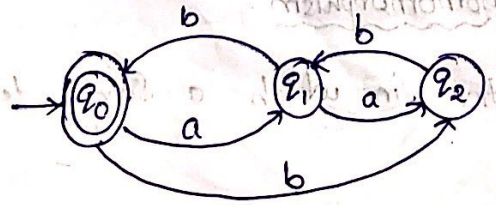
$$r = r_{ii}^* \cdot r_{ij} \left(r_{jj} + r_{ji} \cdot r_{ii}^* r_{ij} \right)^*$$

(q_i, q_j)

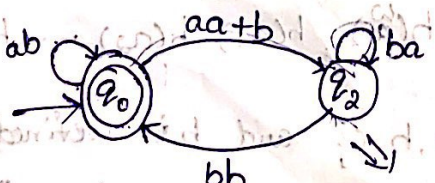
$$r = r_1^* r_2 (r_4 + r_3 r_1^* r_2)^*$$

Problems:-

1Q.



Sol:-

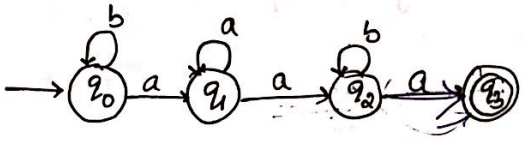


$r = (ab)^*(aa+b)$

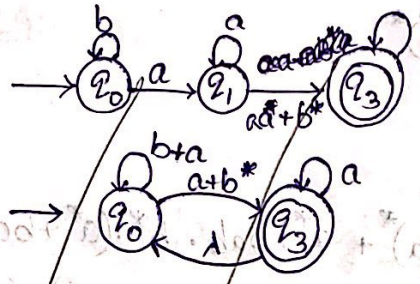
$q_0 \quad [(ab)^* + (aa+b)(ba)^* bb]^*$

26/7/19

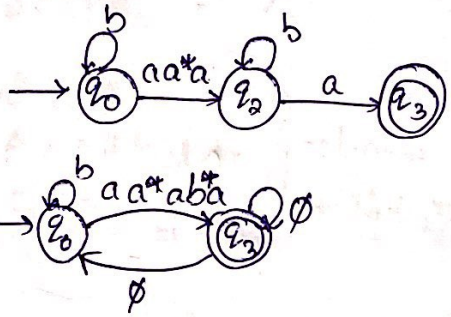
2Q Find RE for FA-



Ans:-

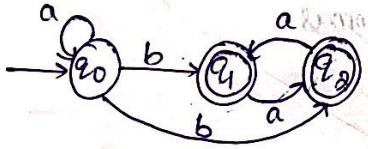


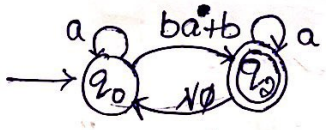
$\Rightarrow RE = (b+a)(a+b^*)[a+(b+a)^*(a+b^*)]^*$



$RE = b^*aa^*ab^*a$

3Q.

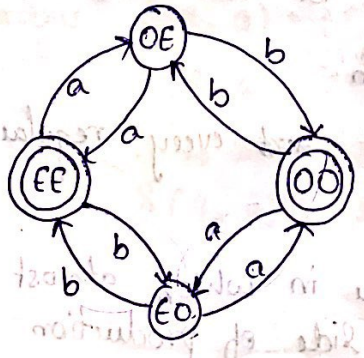




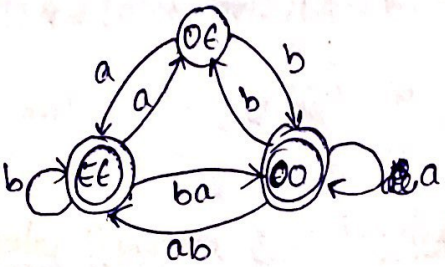
$\Rightarrow a^*(ba^*+b)[a+a^*ba^*b]^* \Rightarrow a^*(b+ba)(aa)^*$

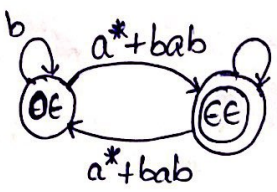
$\Rightarrow a^*ba^*.$

4Q.



Ans





$$R\epsilon = b^*.(a^*+bab)\left[b+(abba)^* + a^*+bab \cdot b^*.(a^*+bab)\right]^*$$

31/7/19

* Regular Grammar:-

↓ left

(T) (NT)

S → terminal Non-terminal

A → terminal

Eg:- S → a(A) → right side (NT)

A → λ | b | aS

* This is called Right Linear Grammar.

S → (A)a.

A → b | (S)a

* This is left linear Grammar.

* Combination of both is called "Linear Grammar".

Note:- The Regular Grammar can only be either left or right Linear Grammar but not both.

* Not Every Linear is regular but every regular is linear grammar.

Linear Grammar:- It is a grammar in which atmost one variable can occur on the right side of production without restriction on the position of the variable.

→ Right Linear Grammar :-

A grammar $G=(V,T,S,P)$ is said to be right linear if all the prod of the form $A \to xB$ where $A,B \in V$
$B \to x$

and $x \in T^*$

→ Left LG:-

$A \to Bx$
$B \to x$

→ Eg:- ① $S \to abS|a$ → Right LG

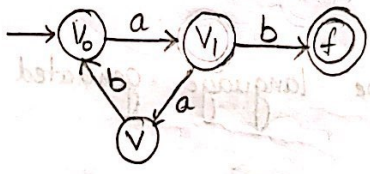
② $S \to S_1 ab$
$S_1 \to S_1 ab | S_2$ } → Left LG.
$S_2 \to a$

③ $S \to A$
$A \to aB | \lambda$ } → linear Grammar
$B \to Ab$ → Not regular Grammar.

Problem :-

① Construct a finite automata that accepts the language generated by the grammar $V_0 \to aV_1$
$V_1 \to abV_0 | b$

Sol:-



$\Rightarrow (aab)^*ab$.

② Construct a right linear grammar for the given language $L(aab^*a)$

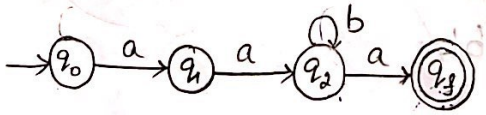
Transitions $\delta(q_0,a) = \{q_1\}$ and convert Right LG to Left LG?

$\delta(q_1,a) = \{q_2\}$

$\delta(q_2,b) = \{q_2\}$

$\delta(q_2,a) = \{q_f\}$

Sol:-



$\Rightarrow aab^*a$.

RLG :- $q_0 \to aq_1$ '

$q_1 \to aq_2$

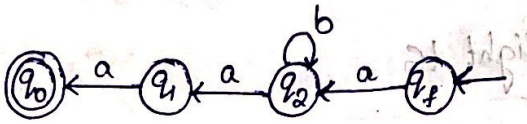
$q_2 \to bq_2$

$q_2 \to aq_f$

$q_f \to \lambda$

* Converting to left :-

① Find reverse to the transition graph.



②

$q_f \to aq_2$

$q_2 \to bq_2$

$q_2 \to aq_1$

$q_1 \to a$

③ Reverse the above grammar production.

$q_f \to q_2a$

$q_2 \to bq_2$

$q_2 \to q_1a$

$q_1 \to a$

} ④ This is the left linear Gramm

③ Construct a ~~dfa~~ automata that accepts the language generated by th grammar $S \to abA$

$A \to baB$

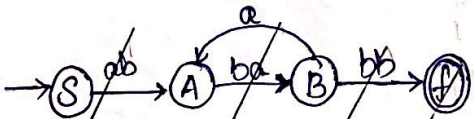
$B \to aA \mid bb$.

④ Convert the given RLG to LLG. (above grammar)

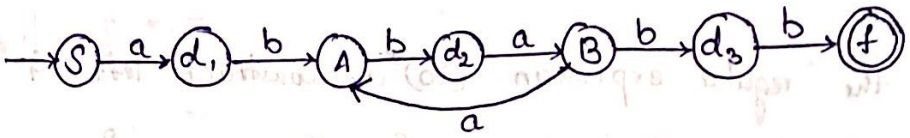
⑤ Find a regular grammar that generates the language $L(aa^*(ab$

Answers :-

3Ans :-



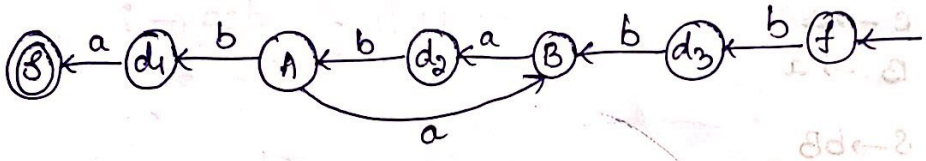
$\to ab.(aba+a)^*.bb.$



$RE = ab.(baa)^+.bb$

Ans:- $S \rightarrow abA$
$A \rightarrow baB$
$B \rightarrow aA \mid bb$

Reverse RLG

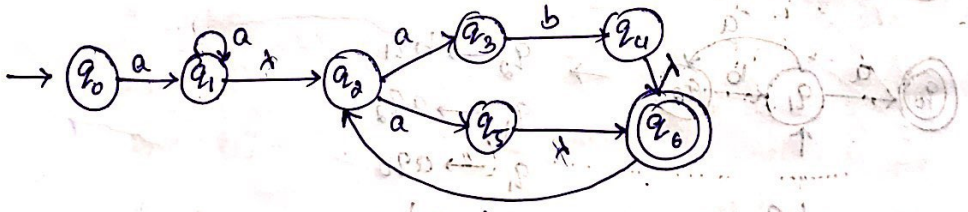


$S \rightarrow Bbb$
$B \rightarrow Aba \mid bb$

$f \rightarrow d_3 b$
$d_3 \rightarrow Bb$
$B \rightarrow d_2 a$
$d_2 \rightarrow Ab$
$A \rightarrow Ba \mid d_1 b$
$d_1 \rightarrow Sa$

$\Rightarrow S \rightarrow Bbb$
$B \rightarrow Aba$
$A \rightarrow Ba \mid Sab$
$S \rightarrow \lambda$

Ans:- $L(aa^*(ab+a)^*)$



$q_0 \rightarrow aq_1$
$q_1 \rightarrow aq_1$
$q_1 \rightarrow q_2$
$q_2 \rightarrow aq_3$
$q_3 \rightarrow bq_4$
$q_4 \rightarrow q_6$
$q_2 \rightarrow aq_5$
$q_5 \rightarrow q_6 \quad ; \quad q_6 \rightarrow q_2$

2/8/19

1. Consider the regular expression $(ab)^*a$. Construct left L.G.

2. Construct RLG for the language $\{a^n b^m / n \geq 2, m \geq 3\}$.

3. Construct LLG for the $L = \{aab(ab)^*\}$

4. Convert the given right linear to left linear

(1) $S \rightarrow bB$
$B \rightarrow bc / aB$
$c \rightarrow a$
$B \rightarrow a$

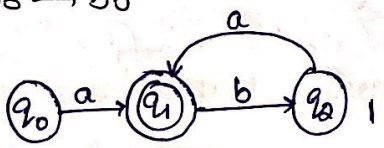
(2) $S \rightarrow bB$
$B \rightarrow bC / aB / b$
$C \rightarrow a$

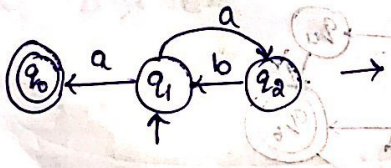
(3) $S \rightarrow 0A$
$A \rightarrow 10A / 1$

5. Construct DFA that accepts the language.

$S \rightarrow abA$
$A \rightarrow baB$
$B \rightarrow aA$
$B \rightarrow bb$

Ans:-





$q_2 \rightarrow bq_1$
$q_1 \rightarrow aq_2$
$q_1 \rightarrow aq_0$
$q_0 \rightarrow \lambda$

$q_2 \rightarrow bq_1$
$q_1 \rightarrow aq_2$
$q_1 \rightarrow aq_0$
$q_0 \rightarrow \lambda$
} Left linear

(or) $S \rightarrow a / abS$

(or) $S \rightarrow S_1 a$
$S_1 \rightarrow S_1 ab / \lambda$

2Ans:- $\{a^n b^m \quad n \geq 2, m \geq 3\}$

aabbb

$S \rightarrow aaA$

$A \rightarrow aA \mid bbbB$

$B \rightarrow bB \mid \lambda$

3Ans:- $\{aab(ab)^*\}$

$S \rightarrow aab \mid Sab$

$S \rightarrow aabA$

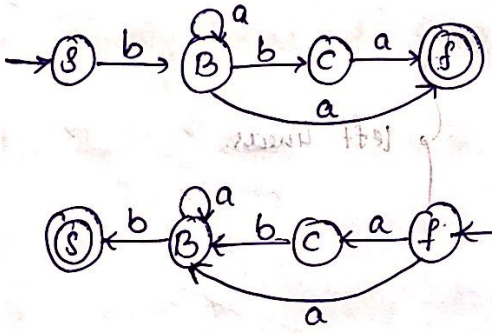
$A \rightarrow abA$

4Ans:- (i) $S \rightarrow bB$

$B \rightarrow bC \mid aB$

$C \rightarrow a$

$B \rightarrow a$



$f \rightarrow aC$

$f \rightarrow aB$

$C \rightarrow bB$

$B \rightarrow aB$

$B \rightarrow bS$

$f \rightarrow Ca$

$f \rightarrow Ba$

$C \rightarrow Bb$

$B \rightarrow Ba$

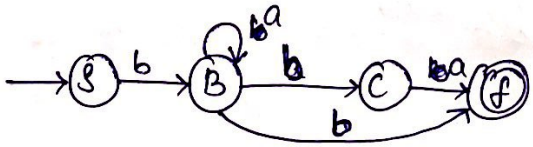
$B \rightarrow Sb$

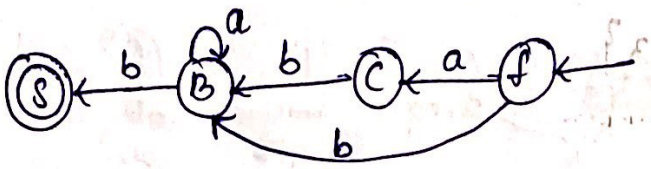
Left Linear.

(ii) $S \rightarrow bB$

$B \rightarrow bC \mid aB \mid b$

$C \rightarrow a$



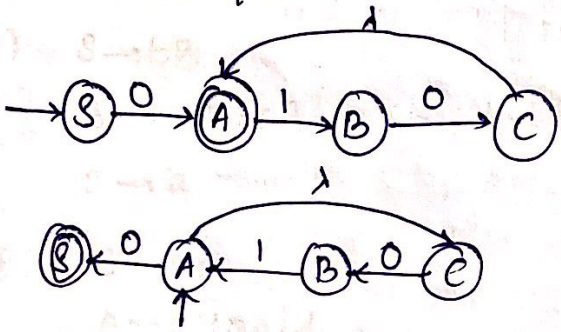


$f \to aC$
$f \to bB$
$C \to bB$
$B \to aB$
$B \to bS$

$f \to Ca$
$f \to Bb$
$C \to Bb$
$B \to Ba$
$B \to Sb$

} left linear

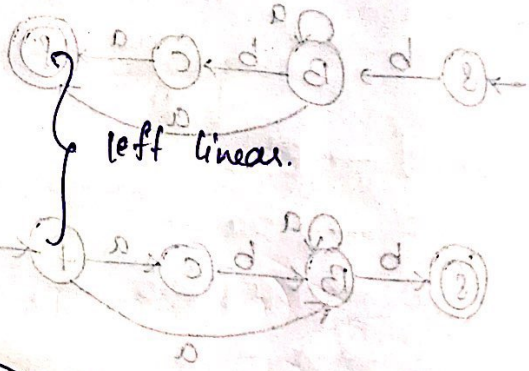
(3) $S \to 0A$
$A \to 10A \mid \lambda$



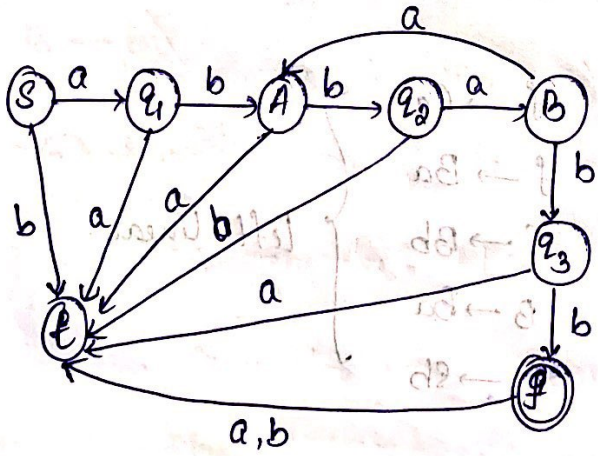
$A \to 0S$
$A \to C$
$C \to 0B$
$B \to 1A$

$A \to S0$
$A \to C$
$C \to B0$
$B \to A1$

} left linear.



5Ans :-



6/8/19

Identify Non-Regular Languages :-

(Can't draw FA, dfa)

Eg: $L = \{a^n b^n ; n \geq 0\}$

$\{\lambda, ab, aabb, ---\}$ → we can't draw the dfa bcoz it can't store the n value. → Non-regular

* $L = \{a^n b^m, \begin{matrix} n \geq 0 \\ m \geq 0 \end{matrix}\}$ → This can have a dfa as there is no relation b/w n & m so, it is regular grammar

Pumping Lemma :- (Game)

It uses pegion hole principle. based on that it will use a game called pumping lemma.

① Use pumping lemma to show that $L = \{a^n b^n, n \geq 0\}$ is Non-regular

Step 1 :- The opponent is asked to pick a m value. $\Rightarrow m = 3$.

Step 2 :- Now choose a string w from L in such a way its length is equal or greater than m. $|w| \geq m$. $= \{aabb\}$.

Step 3 :- choose the decomposition $w_i = xy^i z$

where $|xy| \leq m$ & $|y| \geq 1$

$\underset{x}{a} | \underset{y}{a} | \underset{z}{bb}$

Step 4 :- Try to pick the value of i in such a way that pumped string w_i is not in L.

$i = 0$; $w = abb$ (doesn't belong to given language)

∴ It is not a RG as there is atleast one i value w_i's not the given string in language.

Q. S.T $L = \{(ab)^n a^k : n > k, k \geq 0\}$.

$L = \{ab, ababa, --\}$.

Sol:- Step1:- $m = 2$.

Step2:- $w = \{abab\}$

Step3:- $w_i = xy^i z$

where $|xy| \leq m$ & $|y| \geq 1$

$\underline{a}\underline{b}\,|\,\underline{a}\underline{b}$ $\quad$ $a\,|\,b\,|\,ab$

$x\,|\,y\,|\,z$ $\quad$ $x\,|\,y\,|\,z$

Step4:- $i = 0 \Rightarrow aab \notin L$.

$\therefore$ proved.

2Q. $L = \{a^n b^k c^{n+k}, n \geq 0, k \geq 0\}$. P.T it is not regular using pumping

3Q. $L = \{a^n, n$ is a perfect square$\}$. " "

4Q. S.T $L = \{w \cdot w^R ; w \in \Sigma^*\}$. P.T " "

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

** Context free Grammar :- (We can't draw dfa).

A Grammar $G = (V, T, S, P)$ is said to be context fr

if all productions in P of the form $A \rightarrow x$; $A \in V$, $x \in (V$

*. The language CFG generates CFL.

Regular Grammar and LG are context free but a context fre grammar is not ~~necessarily~~ neccessarily linear or regular Gramm

1Q. Construct a CFG where the string starts and ends with sam symbol. The symbols are $\Sigma = \{a, b\}$

Sol:- $S \rightarrow aAa \mid bAb \mid \lambda$

$A \rightarrow aA \mid Aa \mid bA \mid Ab$

2Q. Find the context free grammar for (i) $L = \{a^n b^m ; n \leq m+3\}$

(ii) $L = \{a^n b^m ; 2n \leq m \leq 3n\}$

(iii) $L = \{a^n b^m c^k ; k = n+m, m, n \geq 0\}$.

7/8/19

Left Most and Right Most derivation :-

Eg: Grammar $G = \{(\{A, B, S\}), \{a, b\}, S, P\}$.

$S \rightarrow AB$

$A \rightarrow aaA$

$A \rightarrow \lambda$

$B \rightarrow Bb$

$B \rightarrow \lambda$

LM

$S \rightarrow AB$

$\rightarrow aaAB$

$\rightarrow aa\lambda B$

$\rightarrow Bb$

$\rightarrow \lambda b$

RM

$S \Rightarrow AB$

$\Rightarrow ABb$

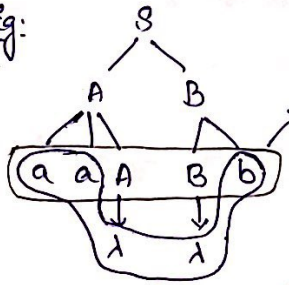
$\Rightarrow aaAb$

$\Rightarrow aa\lambda b$

Derivation Tree :-

The another way of showing derivations, independant of the order in which productions are used is called a derivation tree or parse tree.

Eg:



ten Sential form → Intermediate rep of the tree.

$\Rightarrow aab$

Yield of the tree:-

The string of symbols obtained by reading the leafs of the tree from left to right, vomiting λ's is said to be yield of the tree.

Ambiguous Grammar:-

For a string if you can draw 2 diff parse tree then such a grammar is called as ambiguous grammar.

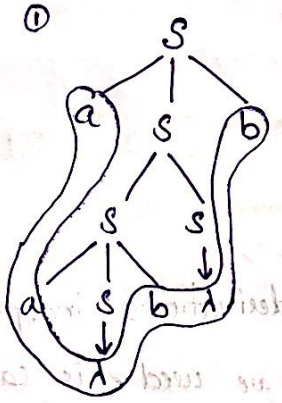
def 2 → For a string if more than one LM der or RM der are available then such a grammar is called 'AG'.

Eg: Check whether the given Grammar is ambiguous or not

Q. $S \rightarrow asb \mid ss \mid \lambda$

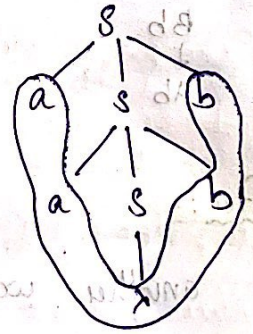
Sol: aabb.

①



aabb

②



aabb

∴ 2 diff parse tree.

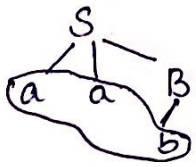
(2Q) $S \rightarrow AB \mid aaB$

$A \rightarrow a \mid Aa$

$B \rightarrow b$

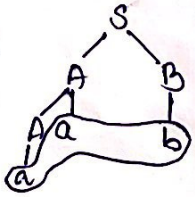
aab

①



aab

②



aab

* Simplification of CFG :-

CFG Simplification can be done through 2 normal forms

(1) Chomskiy normal form

(2) Greibach Normal form.

To apply the above 2 we have to first simplify CFG. by foll steps

① → Removing λ production

② → Removing useless production

③ → Removing unit production

If Those 3 are done through a concept called "Usefull Substitution Rule".

① Removing Useless Production :-

Eg: when even a production of grammar that can never take part in any derivation is called as useless production.

Eg: $S \rightarrow asb \mid \lambda \mid A$

$A \rightarrow aA$

$\Downarrow$

$S \rightarrow asb \mid \lambda$.

② $S \rightarrow abAB \mid ba$

$A \rightarrow aaa$

$B \rightarrow aA \mid bb$

$\Rightarrow S \rightarrow abAaA \mid abAbb \mid ba$

$A \rightarrow aaa$.

② Removing λ Productions :

① It can't be reached from start symbol

② It can't derive a terminal string.

Eg: ① $S \rightarrow aS \mid AB$

$A \rightarrow bA$

$B \rightarrow AA$

$L = \{\phi\}$.

② $S \rightarrow AaB \mid aaB$

$A \rightarrow \lambda$

$B \rightarrow bbA \mid \lambda$.

Sol: remove $A \rightarrow \lambda \rightarrow$ ①

↓

③ $S \rightarrow aB \mid aaB$

$B \rightarrow bb \mid \lambda$.

④ $S \rightarrow a \mid aa \mid aB \mid aaB$

$B \rightarrow bb$