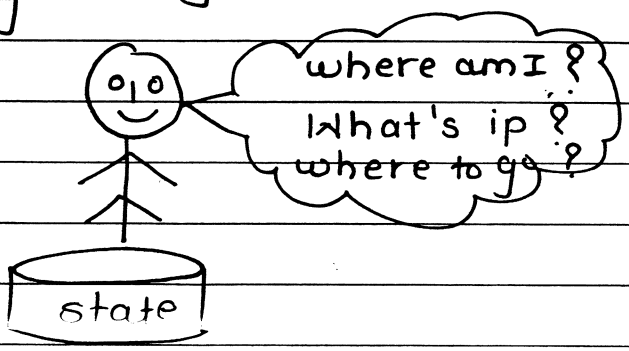
My Target is 8 7 +

I)



II) Method of induction

III) cut the left most symbol

IV) Divide and Conquer

ENIAC machine

(Electronic numerical Integator Computer)

developed in 1936 for destructive purpose

* 1500 sq. feet
* 18000 vaccum tubes
* 1500 relays
* 7000 resistors
* Decimal number system
* 200 megawatt
* 5000 Additions per second
* 6000 cables and switches

Von Neuman suggested Neuman Architectur

John Neuman's stored program concept

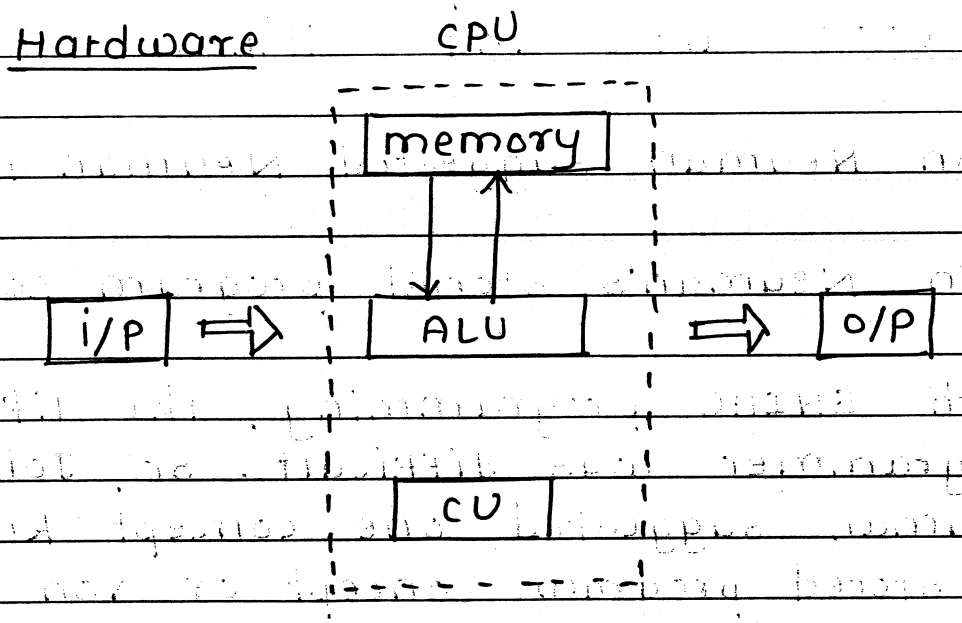
With ENIAC programming, the life of programmer was difficult. So, John Von Neuman suggested one concept known as stored program concept or Von Neuman architecture.

stored program concept (1946)

instruction data

stored program concept :

1. Programmer has to store his program into memory.
2. Let the processor
   1) Fetch it
      (Get program from somewhere)
   2) Decode it
      (Understanding meaning)
   3) Execute it
      (Arithmatic, logical, data)



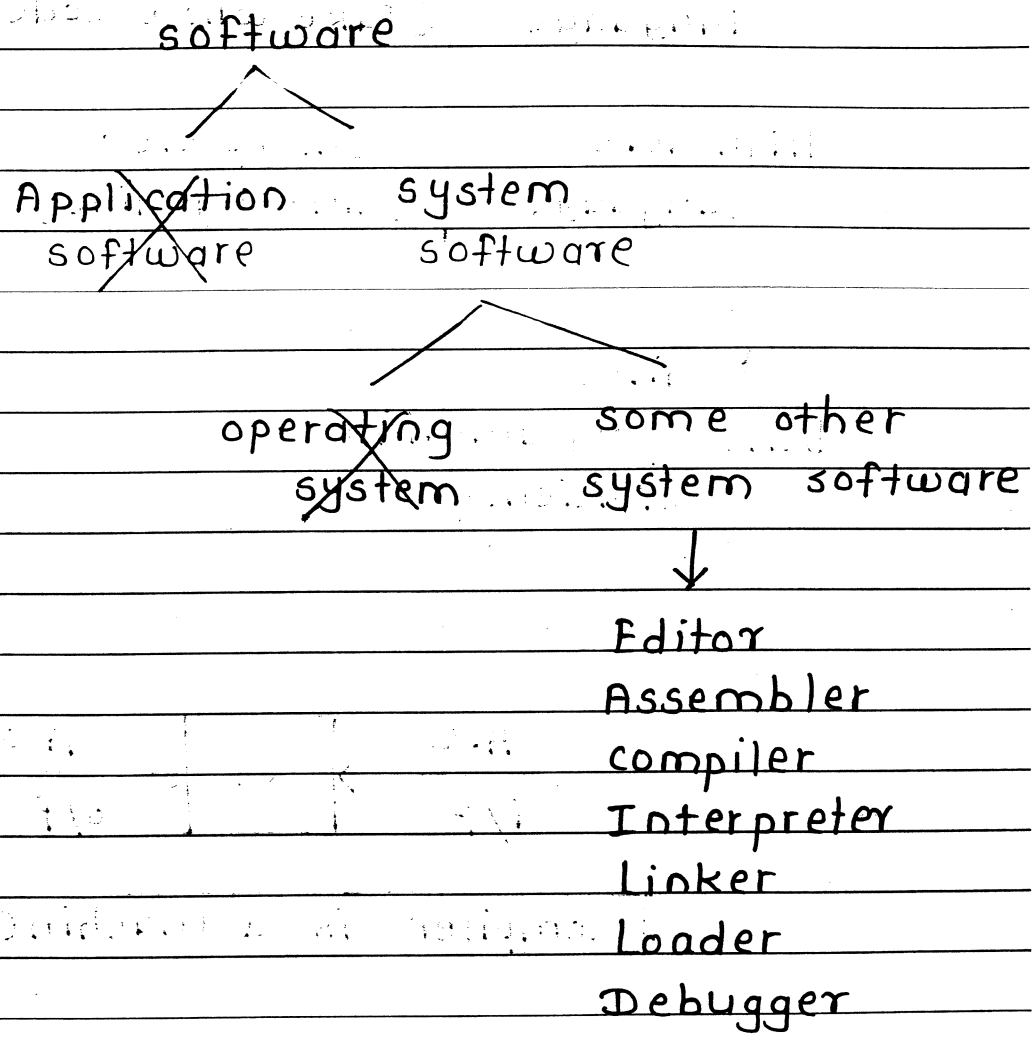
Software

application program

system software

System software

The software which makes system more convenient to be used by user and programmer

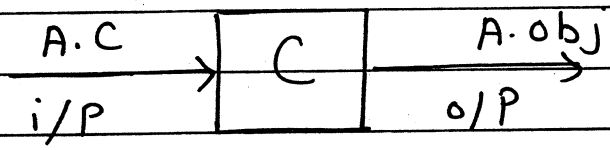


Gives better machine efficiency

low level language (Assembly)	Assembler	Binary language
middle level language	compiler (Take whole code)	machine level language
High level language	Interpreter (line by line)	

Gives better programming efficiency

compiler



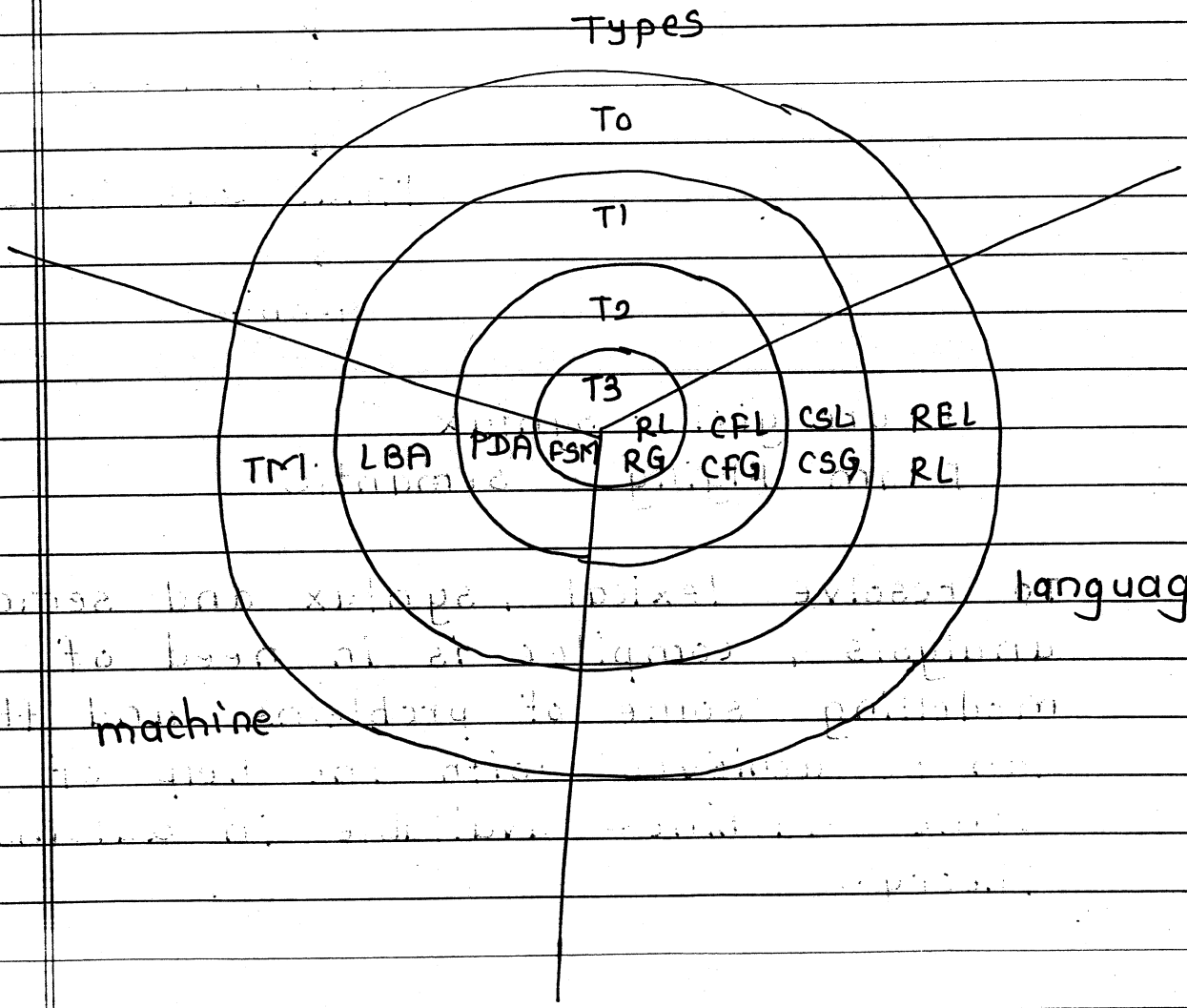
compiler is a machine

Symbol		A.C
↓		↓
Alphabet		Lexical Analysis
↓		↓
word	Error ⇒	Syntax Analysis
↓		↓
sentence	warning ⇒	Semantic Analysis
↓		↓
paragraph		Inter code generation
		↓
		optimization
		↓
		final code generation
		↓
		A.obj

I are girl - syntax
I am flying - semantic

To resolve lexical, syntax and semantic analysis, compiler is in need of modelling some of problems and that can be achieved with the help of some machines available in automata theory.

units :
1) Finite Automata / Finite state machine
2) Regular language / Expression
3) Context free Grammer
4) Push down Automata / Post machine
5) Turing machine
6) Tractable and Intractable.



* ToC is all about language and corresponding converter development.

Prerequisits for TOC :

Set Theory
Relations
Functions
languages
Proof technique

Set Theory

George Contar , Richard Dodeking

Set is a collection of well distinct elements.

1) Roster (list) — $A = \{2, 3, 5, 7, 11, 13, 17\}$

2) set builder — $A = \{x \mid x \text{ is prime } \& \ x < 18\}$

$11 \in A$

$9 \notin A$

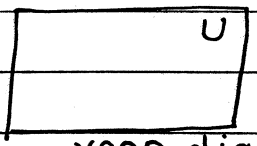
* Cardinality of the set
  Number of elements in set

  card (A) / n(A) / |A| = 8

* empty set / Null set (ϕ)

* Singleton set :
The set which contains single element.

* Universal set :
All sets are under study considered under a big umbrella known as universal set

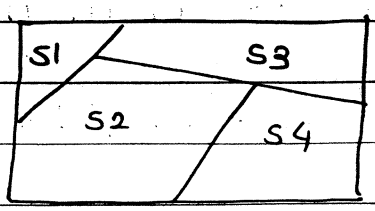


venn dia.

* Finite / Infinite set
The set in which counting of elements comes to an end, is known as finite set.

* Disjoint set
No common elements are present
Intersection is null set.

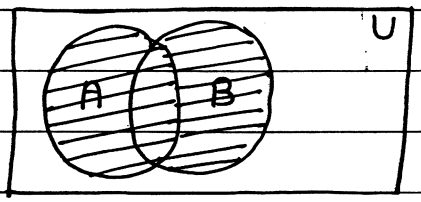
* partitions of set
Set can be divided into disjoint partitions



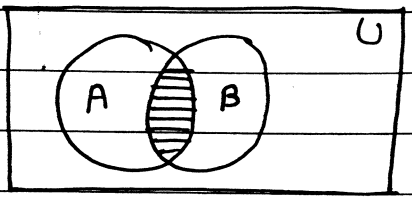
$S = S1 \cup S2 \cup S3 \cup S4$

Operations on Set :

Union



Intersection



identity

$A \cup \phi = A$ $\qquad$ $A \cap \phi = \phi$

$A \cup U = U$ $\qquad$ $A \cap U = A$

$A \cup A^{-1} = U$ $\qquad$ $A \cap A^{-1} = \phi$

Idempotent

$A \cup A = A$ $\qquad$ $A \cap A = A$

Commutation

$A \cup B = B \cup A$ $\qquad$ $A \cap B = B \cap A$

Association

$A \cup (B \cup C) = (A \cup B) \cup C$ $\qquad$ $A \cap (B \cap C) = (A \cap B) \cap C$

distribution

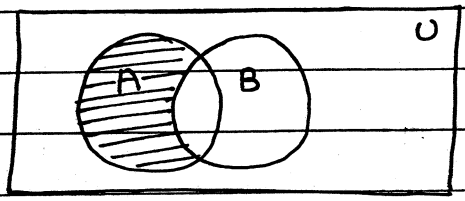
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ $\qquad$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap$

Demorgan's law

* Break the line, change the sign

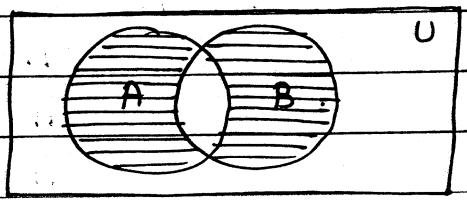
$\overline{A \cup B} = \overline{A} \cap \overline{B}$ $\qquad$ $\overline{A \cap B} = \overline{A} \cup \overline{B}$

set difference



A - B

Symmetric difference.



A - B

* order and inorder set

Sometimes ordering of elements is essential in set

e.g. enum data type in programming language

* Subset and proper subset

A = {1, 2}

B = {1, 2, 3, 4}

A $\subseteq$ B

subset.

A $\subset$ B → equality sign removed

proper subset

* Power set

Power set is a set of all possible subsets.

$A = \{1, 2\}$

$P(A) = \{\phi, \{1\}, \{2\}, \{1,2\}\}$

$$|P(A)| = 2^{|A|}$$

* Cartesian product

Let A and B are two sets, then cartesian product denoted by $A \times B$ consists of all ordered pair where first element of ordered pair belong to A and second element of ordered pair belongs to B.

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

$B = \{x, y, z\}$

$A \times B = \{(1,x)(1,y)(1,z)(2,x)(2,y)(2,z)$
$(3,x)(3,y)(3,z)\}$

always from A

$$|A \times B| = |A| * |B|$$

$|A \times B| \neq |B \times A|$ → as it is ordered pair

Relation

7 is less than 9

A is friend of B

Relation is a association of two distinct entities which some property that they possess.

Let A and B are two sets then relation R from set A to set B is a subset of cartesian product of A and B.

(domain A) $A = \{1, 2, 3\}$ $R \subseteq A \times B$

(Range B) $B = \{x, y, z\}$

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

always from A — always from B

$R = \{(a, b) \mid a \in A \text{ and } b \in B\}$

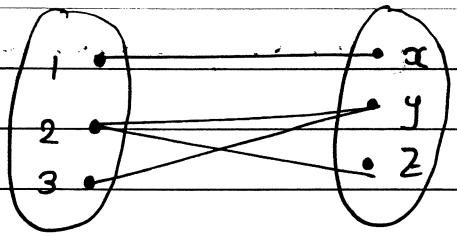
$a R b$

* Representation of relation

1. Tabular representation

		second set x	y	z
first set	1	1	0	0
	2	0	0	1
	3	0	0	1

2. Pictorial representation



* Universal and empty relation

Relation of set A with itself is known as universal relation

$$A \times A$$

and if relation is a null set, then it is known as empty relation.

* Identity relation / diagonal relation

If every element of a set is related t itself is known as identity relation

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

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

	1	2	3	
1	1	0	0	← diagon
2	0	1	0	matrix
3	0	0	1	

* Inverse of relation

It is represented as R^{-1} consists of all ordered pairs which when reversed will belong to R

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

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

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

$$R^{-1} = \{(b,a) \mid (a,b) \in R\}$$

* Composition of relation

Let A, B and C are sets

Relation R is on set A and B

$$R \subseteq A \times B$$

Relation S is on set B and C

$$S \subseteq B \times C$$

$$R = \{(a,b) \mid a \in A \text{ and } b \in B\}$$

$$S = \{(b,c) \mid b \in B \text{ and } c \in C\}$$

$$R \circ S = \{(a,c) \mid \text{there exists } b \text{ where } (a,b) \in R \text{ and } (b,c) \in S\}$$

Imp

* Reflexive Relation

A relation is said to be reflexive if for every $a \in A$ there exists $(a,a) \in R$ (ordered pair)

e.g. $A = \{1,2,3\}$

$R = \{(1,1)(2,3)(3,3)\}$ ✗

$R = \{(1,1)(2,2)(3,3)\}$ ✓

* Symmetric Relation

Let Relation R is on set A then R is said to be symmetric if for every $(a,b) \in$ there exists $(b,a) \in R$

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

$R = \{(1,1)(2,3)(3,2)\}$ ✓

* Transitive relation

Let R is a relation on set A then R is said to be transitive if for $(a,b) \in R$ and $(b,c) \in$ then there exists $(a,c) \in R$

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

$R = \{(1,1)(1,2)(2,1)(2,2)\}$ ✓

* Equivalence Relation

A relation is said to be equivalence relation if it is reflexive, symmetric, transitive

* closure of relation

It may happen that a given relation may not be reflexive, symmetric or transitive then to convert it into reflexive, symmetric and transitive, we need to take corresponding closure operation

PU imp

$S = \{a, b, c\}$

$R = \{(a,b)(b,c)(c,a)\}$

Reflexive closure

Reflexive closure $(R) = R \cup \Delta i$

$= \{(a,b)(b,c)(c,a)\} \cup \{(a,a)(b,b)(c,c)\}$

$= \{(a,b)(b,c)(c,a)(a,a)(b,b)(c,c)\}$

Transitive closure

If relation R is on set A with n elemen

Transitive closure $= R \cup R^2 \cup R^3 \cdots R^n$

$R^2 = R \circ R$

$R^3 = R^2 \circ R$

$R^4 = R^3 \circ R$

Transitive closure $(R) = R \cup R^2 \cup R^3$

$R^2 = R \circ R$

$= \{(a,b)(b,c)(c,a)\} \circ \{(a,b)(b,c)(c,a)\}$

$R^2 = \{(a,c)(b,a)(c,b)\}$

$R^3 = R^2 \circ R$

$= \{(a,c)(b,a)(c,b)\} \circ \{(a,b)(b,c)(c,a)\}$

$R^3 = \{(a,a)(b,b)(c,c)\}$

$R = \{(a,b)(b,c)(c,a)\} \cup \{(a,c)(b,a)(c,b)\} \cup \{(a,a)(b,b)(c,c)\}$

$R = \{(a,b)(b,c)(c,a)(a,c)(b,a)(c,b)(a,a)(b,b)\}$

Symmetric closure

Symmetric closure $= R \cup R^{-1}$

$= \{(a,b)(b,c)(c,a)\} \cup \{(b,a)(c,b)(a,c)\}$

$= \{(a,b)(b,c)(c,a)(b,a)(c,b)(a,c)\}$

Function

Let A and B are two sets then relation R which associates every element of set A to the unique element of set B is called as function f from set A into set B

$f : A \to B$

domain — codomain

$(x, y) \in f$

$x \in A$ $\quad$ $y \in B$

$(x, y) \in f$

$y = f(x)$

image — preimage

Every function is a relation but every relation may not be a function

closure - all possibilities

Language

Symbol

↓

Alphabet ⇒ Σ - finite set of input alphabets

↓

words

↓

sentence

↓

paragraph

KLEEN CLOSURE

$L = \Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots \Sigma^\infty$

Σ^0 : word with zero length

Σ^1 : word with length 1

$\Sigma = \{x\}$

$L = \Sigma^* = \{\varepsilon, x, xx, xxx, xxxx, \cdots\}$

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

$L = \Sigma^* = \{\varepsilon, 0, 1, 00, 11, 01, 10, \cdots\}$

ε = word with zero length

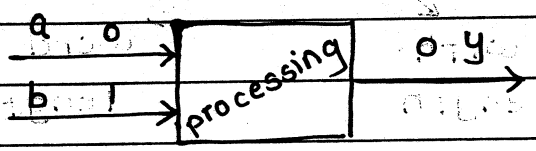
$\Sigma = \{a, b\}$ Find language till Σ^3

$L = \{ \varepsilon, a, b, aa, ab, ba, bb, aaa \ldots\ldots bbb \}$

* Finite Automata

Basic machine : Need of FA / FSM

It's a very primitive machine or output at any instance totally depends on input combinations only.
So, it is also known as combinational machine



a	b	y
0	0	1
0	1	0
1	0	1
1	1	0

This machine does not take care of previous state of machine to generate next state output.
Because of lake of memory, this machine is not able to hold present state of the system.

FA

Finite Automata

(2)

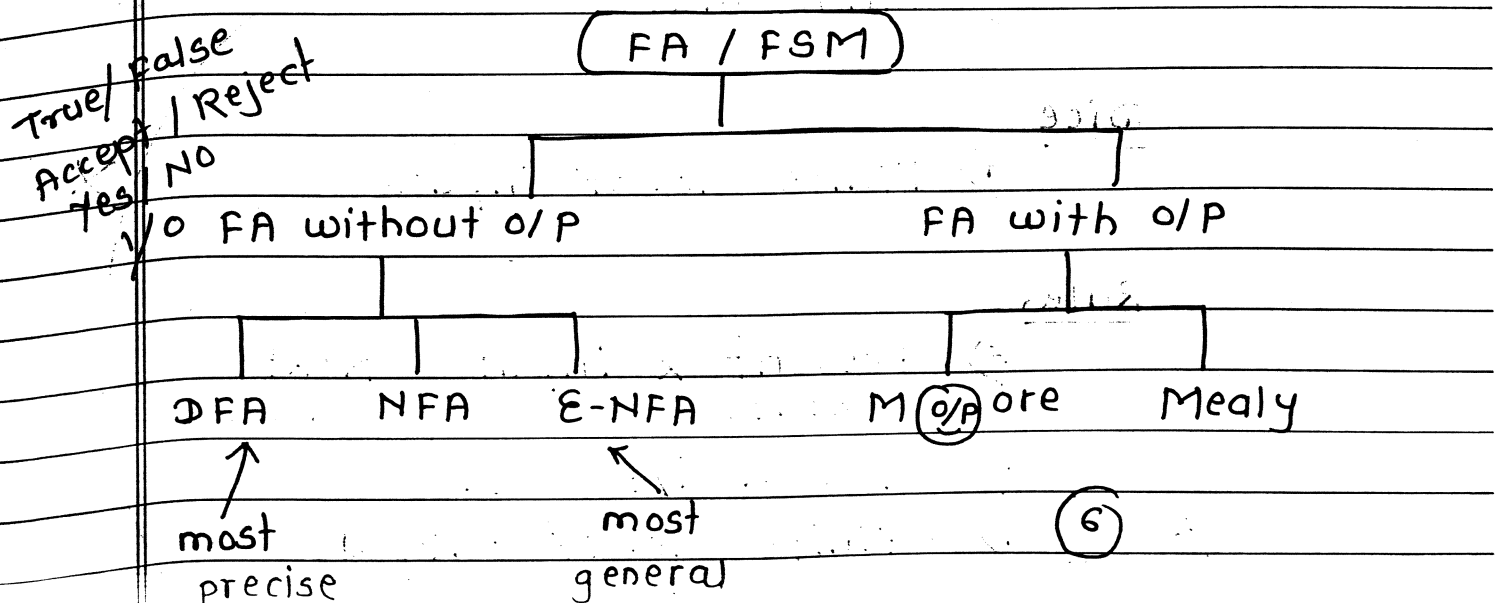
- no. of states
- no. of i/p alphabets

plural form of Greek word Automaton

output of machine is desicided by present state and input combinations automatically.

So, FSM has some limited memory to hold present state of the system.

Types of FSM



(5)

(6)

Even / odd (Divisibility by 2), Divisibility problems, multiple of problems

Design a DFA to accept language which consists of even decimal numbers

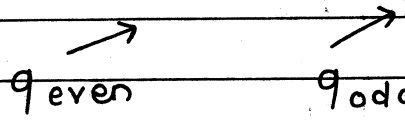
Req. Analysis

$\Sigma = \{0, 1, 2, 3 \ldots\ldots 9\}$

$L = \{0, 2, 4, 6, \ldots\ldots\}$

$2\overline{)12}$ $2\overline{)13}$ $2\overline{)14}$ $2\overline{)15}$

0 1 0 1



q_s - start state

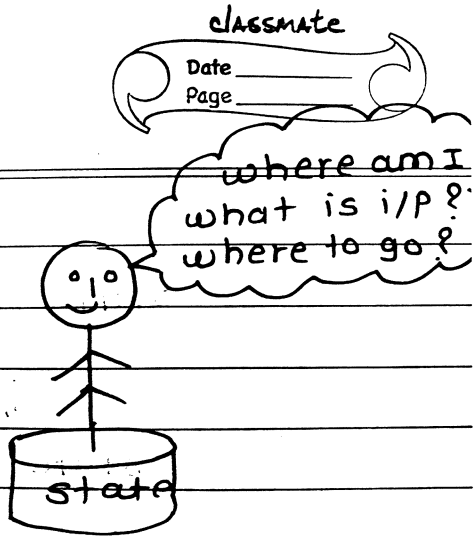
q_{even} = can see even states

q_{odd} = can see odd states

$Q = \{q_s, q_{even}, q_{odd}\}$

$q_0 = q_s$

$F = q_{even}$



δ : STF

P.S x i/P $\Rightarrow$ N.S

$Q_n \times \Sigma \Rightarrow Q_{n+1}$

state diagram state table

Every state of DFA will always have number of arrows coming out of it is equal to number of input alphabets.

$\Sigma = \{(0,2,4,6,8)\ (1,3,5,7,9)\}$

start state

2) 2 2) 5

0 1

qeven state

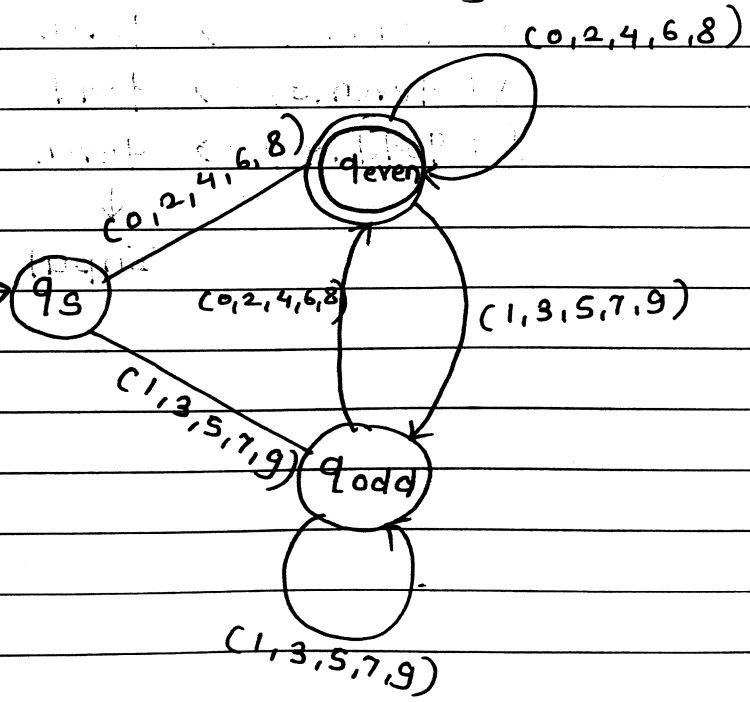
2) 24 2) 27

0 1

qodd state

2) 34 2) 35

0 1



1 2 3 4

1 3 1 1

state transition table

i/p / state	(0,2,4,6,8)	(1,3,5,7,9)
→ q_s	q_{even}	q_{odd}
q_{even}*	q_{even}	q_{odd}
q_{odd}	q_{even}	q_{odd}

Handrun :

1	2	3	4

$\delta(q_s, 1) \rightarrow q_{odd}$

$\delta(q_{odd}, 2) \rightarrow q_{even}$

$\delta(q_{even}, 3) \rightarrow q_{odd}$

$\delta(q_{odd}, 4) \rightarrow q_{even}$

↓

Input is Accepted

2) Design a DFA to accept all decimal numbers which are divisible by 3

$\Sigma = \{0, 1, 2, 3 \ldots\ldots 9\}$

$L = \{0, 3, 6, 9, 12, 15, \ldots\ldots\}$

$Q = ?$

$3\overline{)12}$ $3\overline{)13}$ $3\overline{)14}$ $3\overline{)15}$

0 1 2 0

$\rightarrow q_0$ $\rightarrow q_1$ $\rightarrow q_2$

q_0 = can see remainder 0
q_1 = can see remainder 1
q_2 = can see remainder 2
q_s = start state

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

$q_0 = q_s$ —— from 5 tuppel $(Q, \Sigma, \delta, q_0, F)$

$F = q_0$

$\Sigma = \{(0,3,6,9)\ (1,4,7)\ (2,5,8)\}$

divisible problems मध्ये final state q_0

start state

any no. from each group → 3)3 3)7)5

0 1 2

q_0 state → from each group

as remainder is 0 in q_0 3)06 3)04 3)08

0 1 2

q_1 state

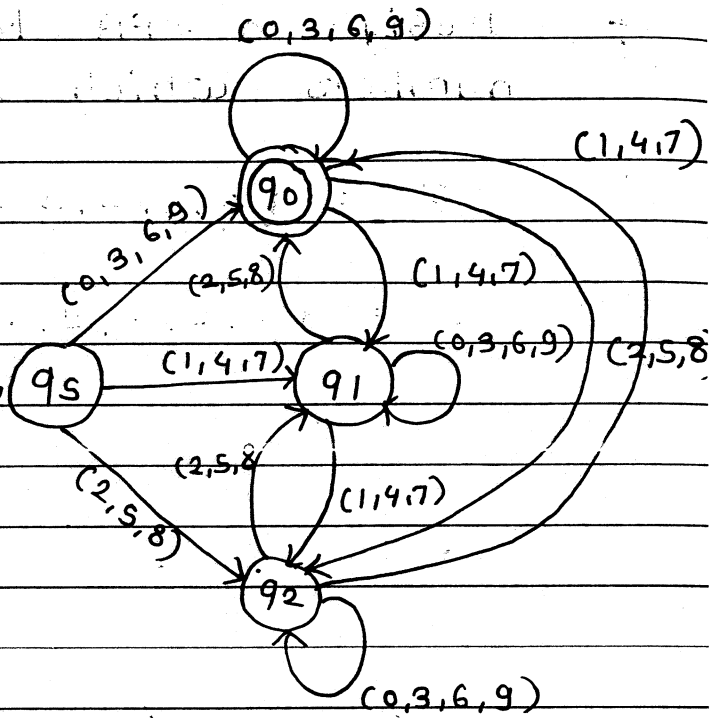
1 remainder 3)13 3)11 3)12

1 2 0 → in which state to go

q_2 state

2 remainder 3)20 3)21 3)22

2 0 1



i/p / State	(0,3,6,9)	(1,4,7)	(2,5,8)
→ q_s	q_0	q_1	q_2
q_0*	q_0	q_1	q_2
q_1	q_1	q_2	q_0
q_2	q_2	q_0	q_1

Handrun

1	2	3	4	2

$\delta(q_s, 1) \rightarrow q_1$

$\delta(q_1, 2) \rightarrow q_0$

$\delta(q_0, 3) \rightarrow q_0$

$\delta(q_0, 4) \rightarrow q_1$

$\delta(q_1, 2) \rightarrow q_0$

↑ input is accepted

Imp 3) Design a DFA to accept given binary number divisible by 3

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

$L = \{0, 11, 110, 1001, 1100, 1111, \ldots\}$

q_0 - can see remainder 0 → decimal → binary = 0

q_1 = —11— 1 = 1

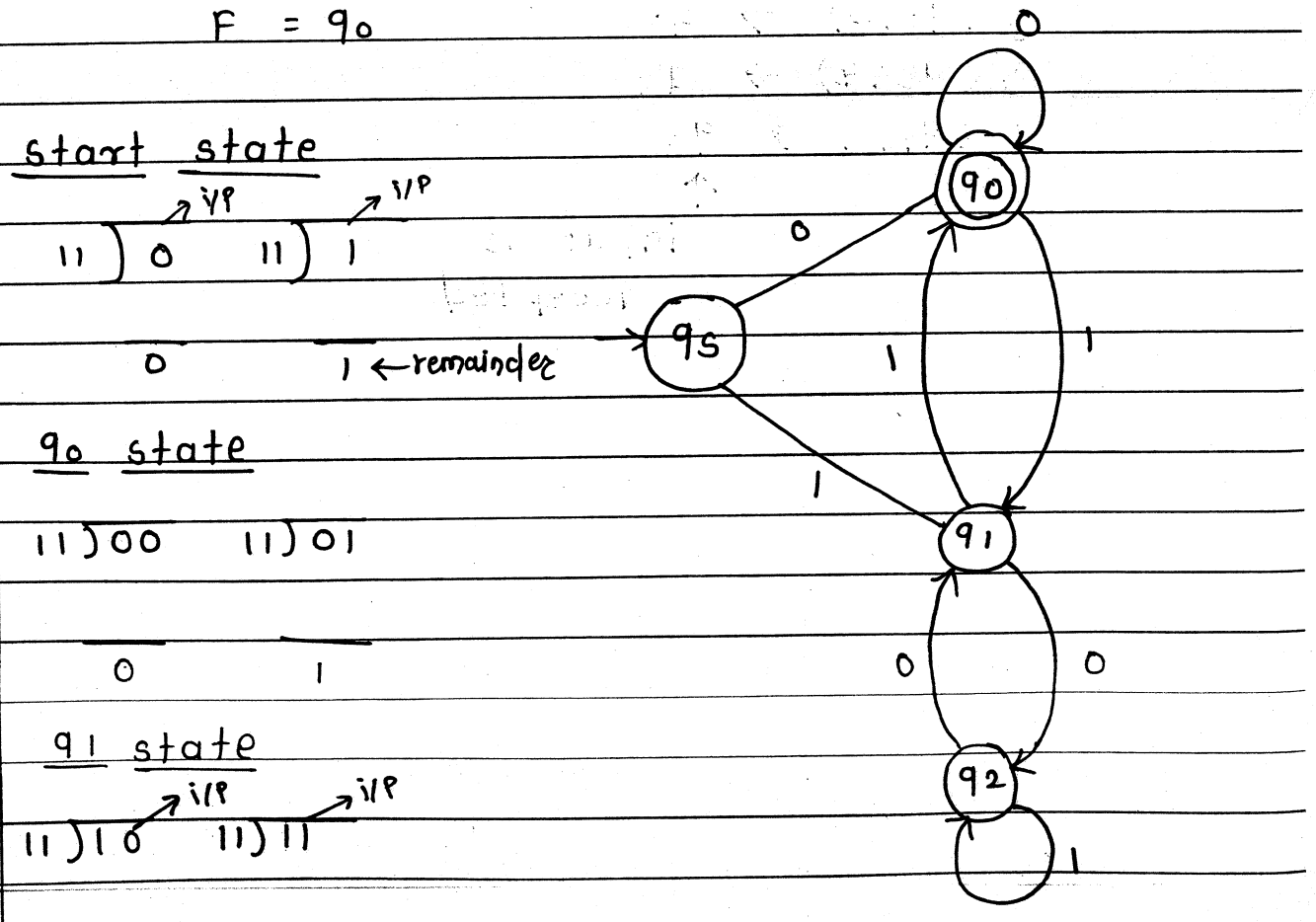
q_2 = —11— 2 = 10

q_s = start state

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

$q_0 = q_s$

$F = q_0$



92 state

11) 101 11) 100

10 001

i/p / curr	0	1
→ q_s	q_0	q_1
q_0*	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

Handrun

1	1	0	1	1

$\delta(q_s, 1) \rightarrow q_1$

$\delta(q_1, 1) \rightarrow q_0$

$\delta(q_0, 0) \rightarrow q_0$

$\delta(q_0, 1) \rightarrow q_1$

$\delta(q_1, 1) \rightarrow q_0$

↑ Input is accepted

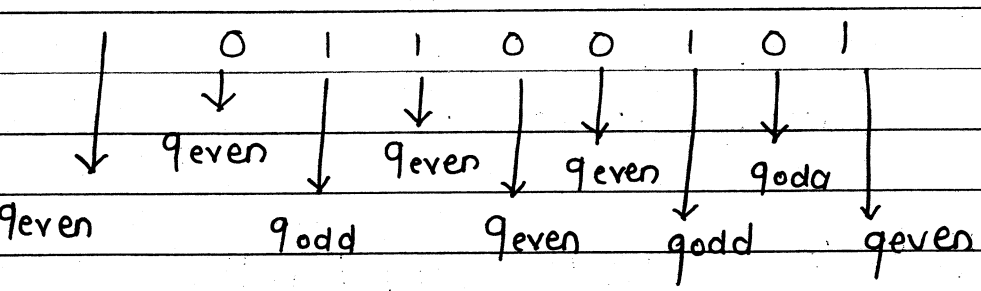
4) Design a DFA to accept binary number where each word consists of even no. of 1's into it.

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

word with zero length i.e. even no. of 1's

$L = \{\varepsilon, 0, 00, 11, 000, 111, 101, 110, \ldots\}$

Let us take an example



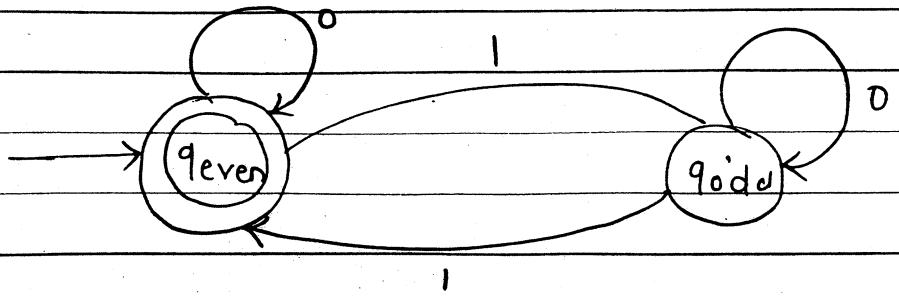
q_{even} = can see even number of 1's / start state

q_{odd} = can see odd no. of 1's

$Q = \{q_{even}, q_{odd}\}$

$q_0 = q_{even}$

$F = q_{even}$



I/P / states	0	1
q_{even}	q_{even}	q_{odd}
q_{odd}	q_{odd}	q_{even}

Handrun

1	0	1	1	0	1

$\delta(q_{even}, 1) = q_{odd}$

$\delta(q_{odd}, 0) = q_{odd}$

$\delta(q_{odd}, 1) = q_{even}$

$\delta(q_{even}, 1) = q_{odd}$

$\delta(q_{odd}, 0) = q_{odd}$

$\delta(q_{odd}, 1) = q_{even}$

↑ input is accepted

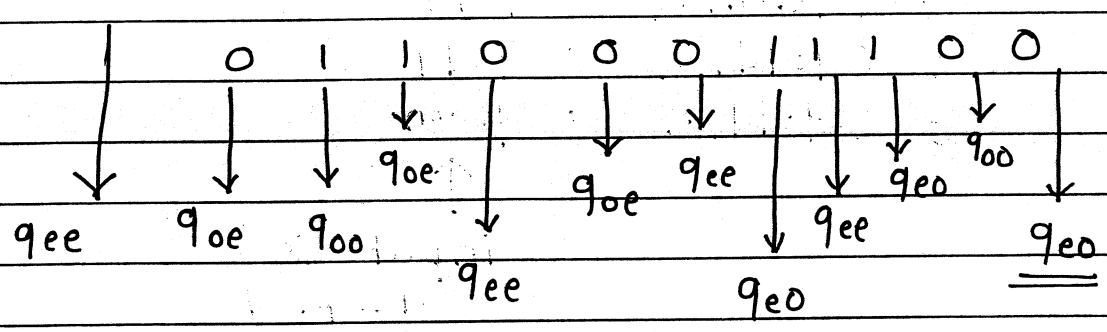
Vimp

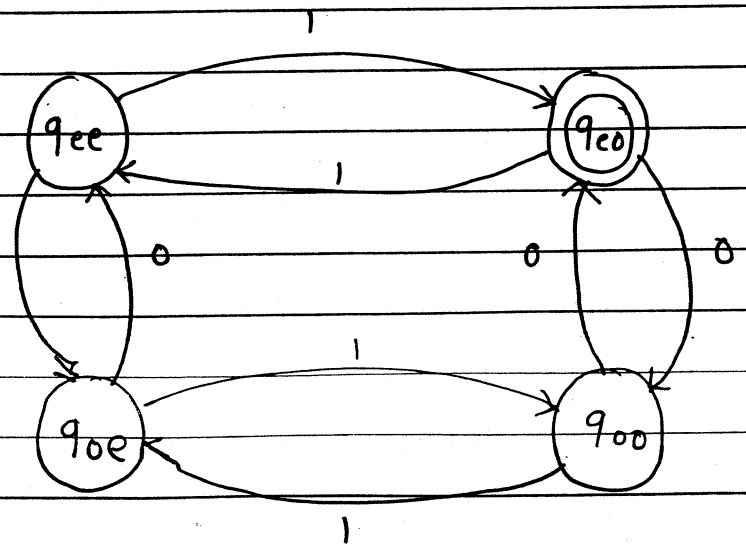
8m 5) Design a DFA to accept binary numbers where each word consists of even no. of zero's and odd no. of 1's

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

Even number of 0's & odd no. of 1's

no. of 0's	no of 1's	0's 1's
even	even	q_{ee}
odd	even	q_{oe}
even	odd	q_{eo}
odd	odd	q_{oo}





dead state always has self loop

classmate Date Page

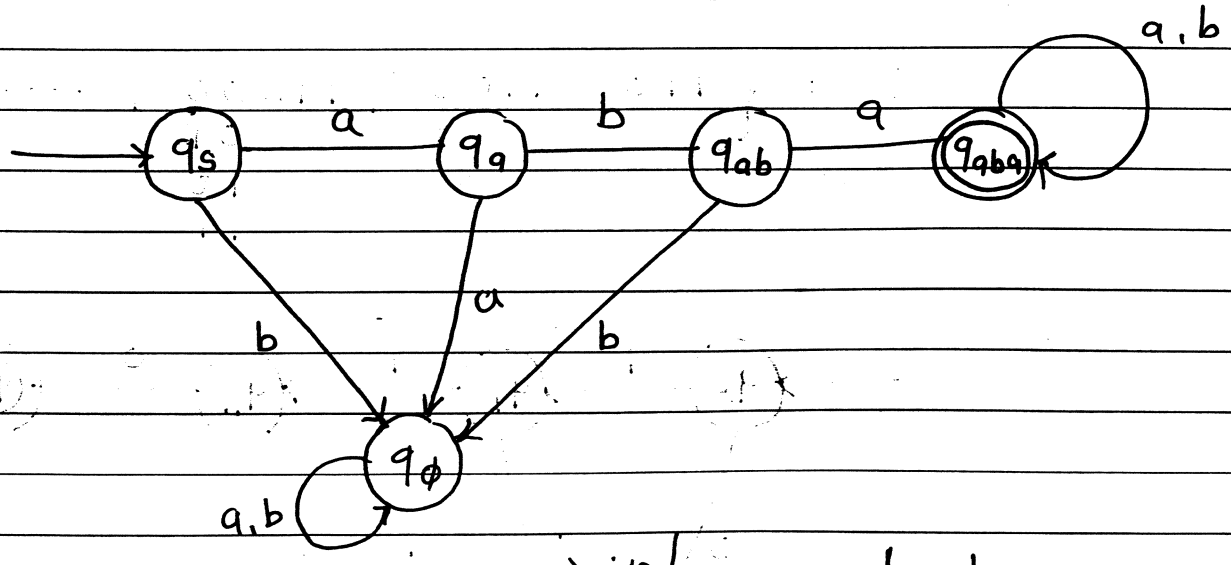
Starting with ~~slash~~ / substring of ~~sto~~ / ending with

6) Design a DFA to accept all strings starting with aba

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

$L = \{ aba, aba\,a, aba\,b, aba\,aa, aba\,ab, aba\,ba, aba\,bb, aba\,aaa, \ldots, aba\,bbb, \ldots \}$

for I st word of lang. to get base frame



q_s = start state

q_a = can see a

q_{ab} = can — " — ab

q_{aba} = — " — aba

i/p / state	a	b
q_s	q_a	q_ϕ
q_a	q_ϕ	q_{ab}
q_{ab}	q_{aba}	q_ϕ
q_{aba}	q_{ba}	q_{aba}
q_ϕ	q_ϕ	q_ϕ

No. of arrows = no. of input alphabet
i.e. Σ

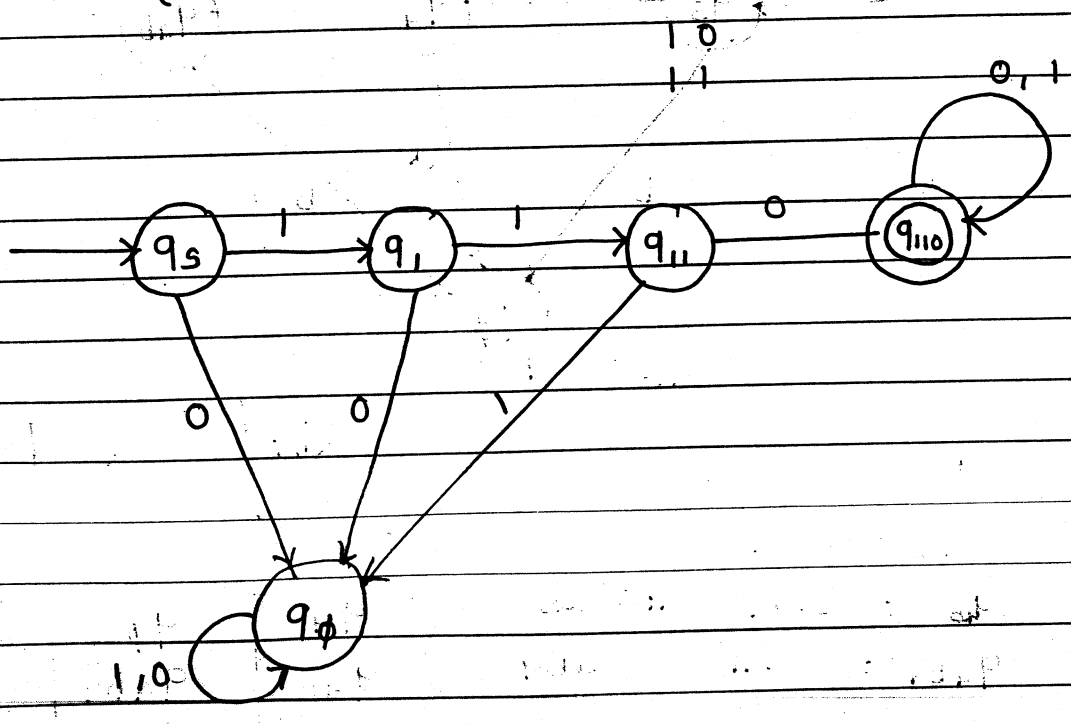
① As starting with is a very rigit constraint, if a word is started with any other DFA leads to dead state

② In starting with problems will always have self loop for final state for all input symbols.

7) Design a DFA to accept all string starting with 110 over an alphabet $\Sigma = \{0, 1\}$

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

$L = \{110, 1100, 11000, \ldots\}$



q_s = start state
q_ϕ = can see 01
q_{11} = can see 11
q_{110} = can see 110

i/p / state	0	1
q_s	q_ϕ	q_1
q_1	q_ϕ	q_{11}
q_{11}		
q_{110}		
q_ϕ		

8) Design a DFA to accept all strings consists of a substring aba over an alphabet $\Sigma = \{a, b\}$

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

$L = \{ aba, \underline{aba}\,a, \underline{aba}\,aa, \underline{aba}\,ab, a\,\underline{aba}, \underline{aba}\,ba, b\,\underline{aba}, \underline{aba}\,bb, aa\,\underline{aba}, \underline{aba}\,aba, ab\,\underline{aba}, ba\,\underline{aba}, bb\,\underline{aba}, a\,\underline{aba}\,a, a\,\underline{aba}\,b, b\,\underline{aba}\,a, b\,\underline{aba}\,b \}$

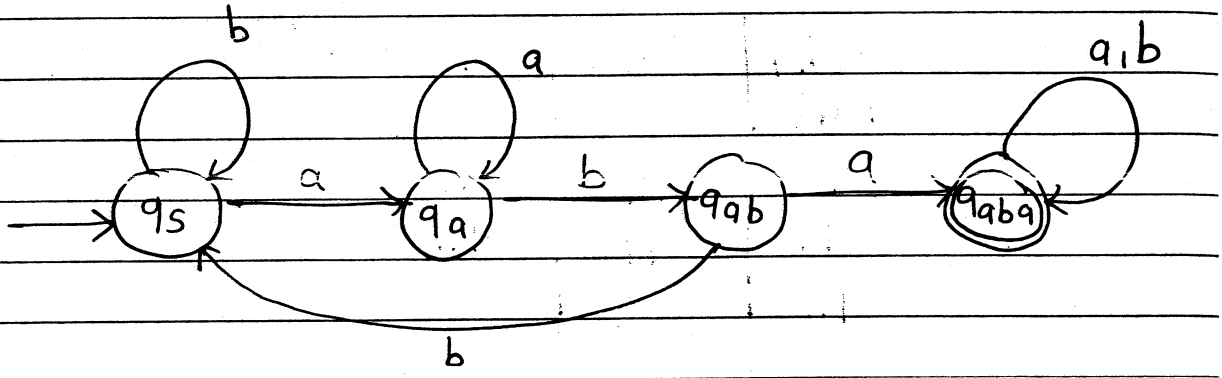
* substring problems will always have self loop at final state for all i/p symbols

q_s = start state / can see b

q_a = can see a

q_{ab} = can see ab

q_{aba} = can see aba



q_s : (a) b

q_a : ~~a~~ (ab)

q_{ab} : (aba) ~~a~~ ~~b~~ b

state transition table

i/p / States	a	b
q_s	q_a	q_s
q_a	q_a	q_{ab}
q_{ab}	q_{aba}	q_s
q_{aba}	q_{aba}	q_{aba}

Handrun

a	a	b	a	b

$\delta(q_s, a) = q_a$

$\delta(q_a, a) = q_a$

$\delta(q_a, b) = q_{ab}$

$\delta(q_{ab}, a) = q_{aba}$

$\delta(q_{aba}, b) = q_{aba}$ ← input is accepted

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

Design a DFA to accept all strings with ~~starting~~ substring 110

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

$L = \{ 110, 1100, \ldots \}$

1

0 110

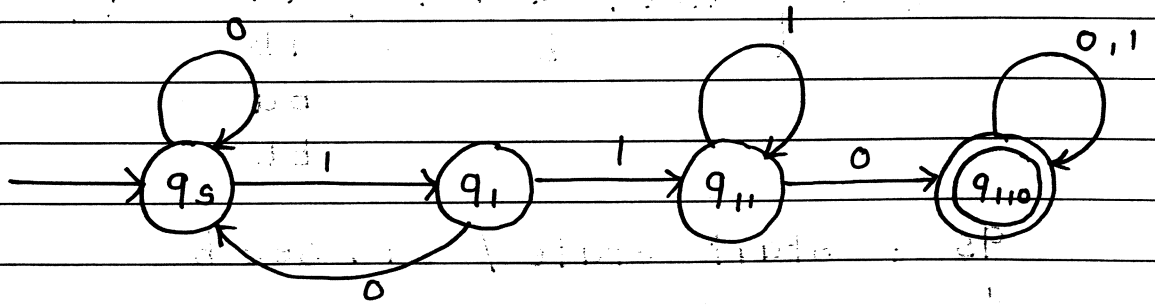
1

q_s = start state / can see 0

q_1 = can see 1

q_{11} = can see 11

q_{110} = can see 110



q_s	q_1		q_{11}	
0 (1)	1	1	11	11
	~~10~~	(11)	(110)	~~111~~

I/P / states	0	1
q_S	q_S	q_1
q_1	q_S	q_{11}
q_{11}	q_{110}	q_{11}
q_{110}	q_{110}	q_{110}

Handrun

1	1	1	1	0	0

$\delta(q_S, 1) = q_1$

$\delta(q_1, 1) = q_{11}$

$\delta(q_{11}, 1) = q_{11}$

$\delta(q_{11}, 1) = q_{11}$

$\delta(q_{11}, 0) = q_{110}$

$\delta(q_{110}, 0) = q_{110}$ ← input is accepted

10) Design a DFA to accept all strings ending with abaa, $\Sigma = \{a, b\}$

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

$L = \{ aba, \underline{a\ aba}, \underline{aa\ aba}, \ldots \}$

$a\ aba$: b

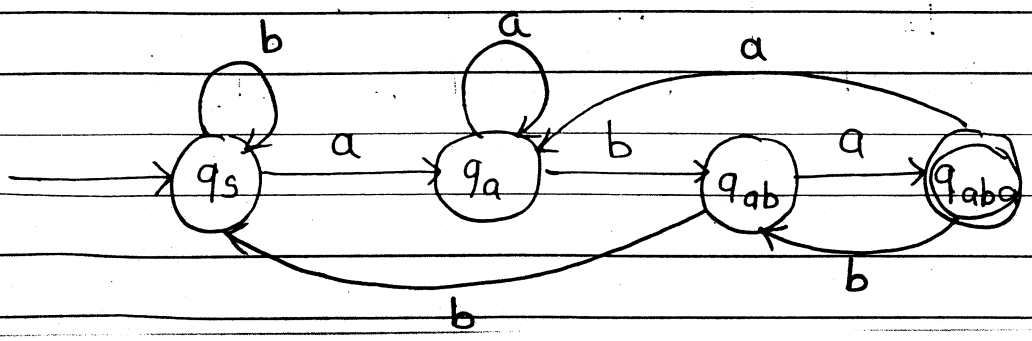
$aa\ aba$: ab, ba, bb

q_S - start state / can see b

q_a - can see a

q_{ab} = ab

q_{aba} = aba



q_s q_a q_{ab}

(a) b ~~aa~~ (ab) (aba) abb

q_{aba}

q_{aba} ~~aba~~ ~~aba~~

i/p / states	a	b
q_s	q_a	q_s
q_a	q_a	q_{ab}
q_{ab}	q_{aba}	q_s
q_{aba}	q_a	q_{ab}

Handrun

a	a	b	a	b	a

$\delta(q_s, a) = q_a$

$\delta(q_a, a) = q_a$

$\delta(q_a, b) = q_{ab}$

$\delta(q_{ab}, a) = q_{aba}$

$\delta(q_{aba}, b) = q_{ab}$

$\delta(q_{ab}, a) = q_{aba}$

classmate
Date
Page

8 M

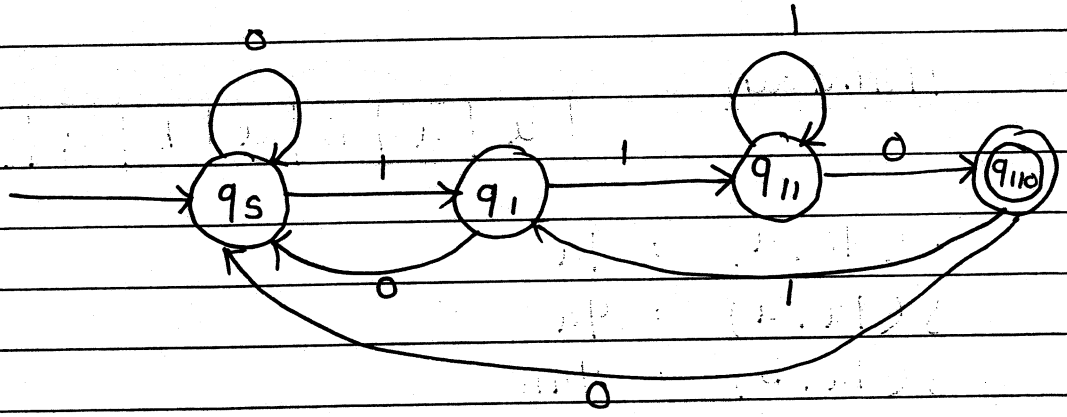
Imp
11

1. $\Sigma = \{0, 1\}$ Accept all string which are ending with 110

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

$$L = \{ 110, \underset{0}{1\,110}, \underset{10}{\underset{01}{\underset{00}{11\,110}}} \}$$

q_s = start state / can see 0
q_1 = can see 1
q_{11} = can see 11
q_{110} = can see 110



q_s	q_1	q_{11}	q_{110}
0 (1)	X0 (11)	(110) X11	XX01 XX00

i/p / States	0	1
q_s	q_s	q_1
q_1	q_s	q_{11}
q_{11}	q_{110}	q_{11}
q_{110}	dc	q_1

ending with problems can never enjoy self loop at final state

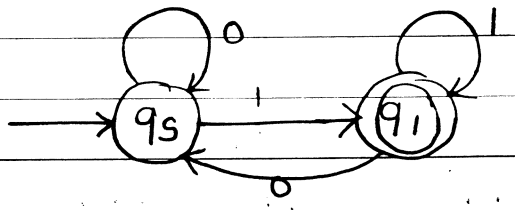
12) $\Sigma = \{0, 1\}$

Accept all string 1) Ending with 1
2) Ending with 01 3) Ending with 11

1) Ending with 1

$L = \{1, 0\underline{1}, 00\underline{1}\}$

10
11



q_s = start state, can see 0

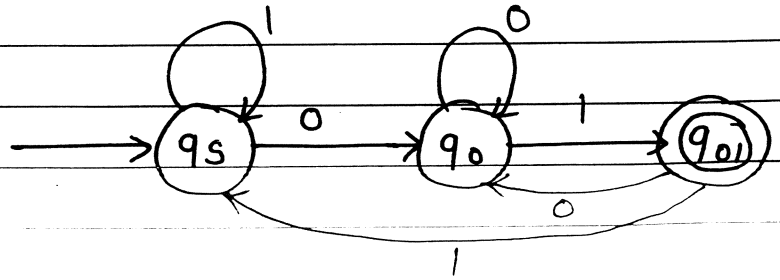
q_1 = can see 1

q_s	q_1	
0 (1)	X0	X1

ii) Ending with 01

$L = \{01, 0\underline{01}, 00\underline{01} \ldots\}$

1
01



q_s = start stat
can see 1

q_0 = can see

q_{01} = can see 0

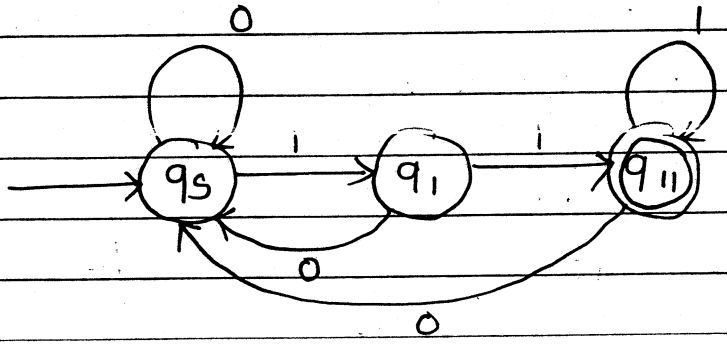
q_s	q_0	q_{01}	
(0) 1	(01)	00	∅X1 ∅X0

substring / ending with = start state / can see (opposite)

classmate
Date
Page

iii) Ending with 11

$L = \{ 11, 011, 111, 0011, 0111, 1011, 1111, \ldots \}$



q_S = start state / can see 0

q_1 = can see 1

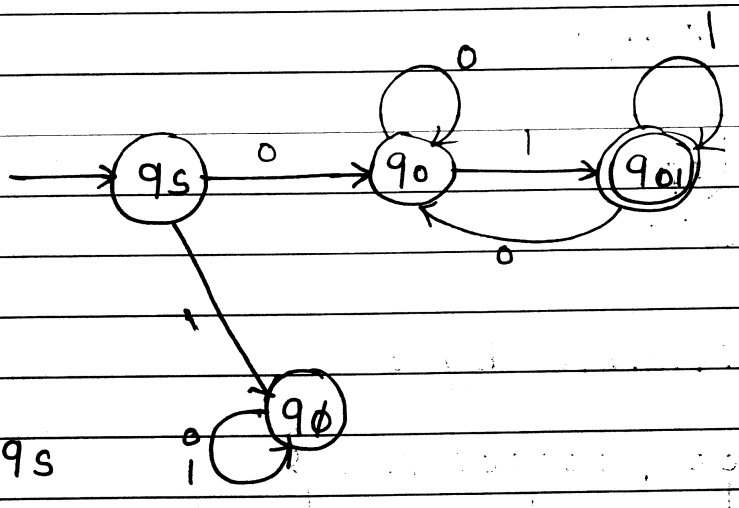
q_{11} = can see 1

q_S		q_1		q_{11}	
0	(1)	10	(11)	✗✗0	✗11

13) Design a DFA to accept all words starting with 0 and ending with 1

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

$L = \{01, 001, 0001, \ldots\}$



q_S = start state

q_0 = can see 0 / starting 0

q_{01} = can see 01, ending 1

q_S

0 1

q_0

01 00

q_{01}

ϕ/1 ϕ/0

i/p / states	0	1
q_S	q_0	q_ϕ
q_0	q_0	q_{01}
q_{01}	q_0	q_{01}
q_ϕ	q_ϕ	q_ϕ

Handrun

0	1	0	0	1

$\delta(q_S, 0) = q_0$

$\delta(q_0, 1) = q_{01}$

$\delta(q_{01}, 0) = q_0$

$\delta(q_0, 0) = q_0$

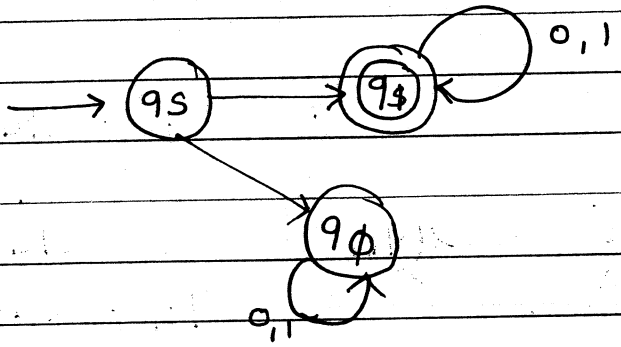
$\delta(q_0, 1) = q_{01}$

Merging / Divide & Conquer

14) Design a DFA to accept all words starting with 1 and length divisible by 3

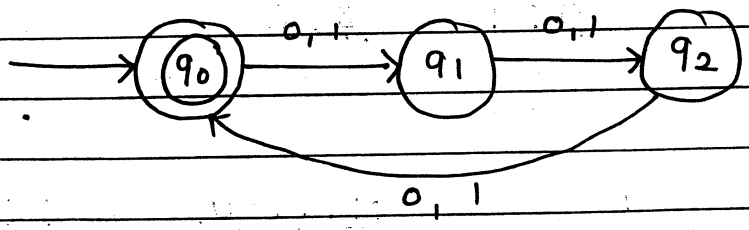
starting with 1

$L = \{1, 10, 11 \ldots\}$



length divisible by 3

$L = \{\varepsilon, 000, 000000 \ldots\}$



q_0 = can see rem 0

q_1 = -11- 1

q_2 = -11- 2

3)12 3)13 3)14

0 1 2

q_0: 0 1

3)1

1

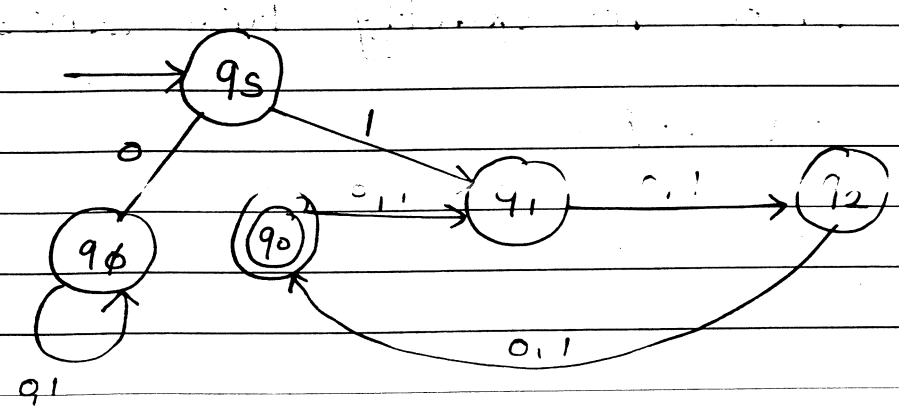
q_1: 0 11

length 2

3)2

2

$$L = \{ 100 \ldots\ldots 100000,\ 101,\ 110,\ 111 \}$$



exactly - rigid constraint
counting of no. of zero's
compulsion - dead state

classmate
Date
Page

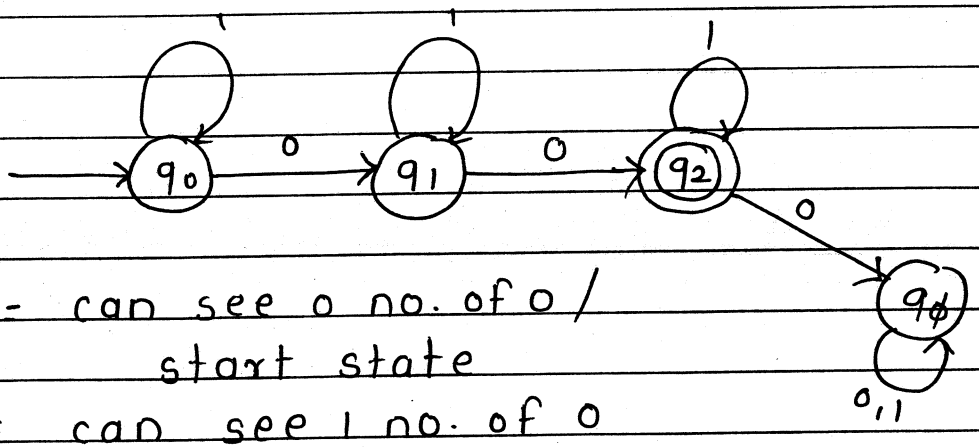
* Some Critical Word Domain

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

Design a DFA to accept all words such that there are exactly two zeros in it

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

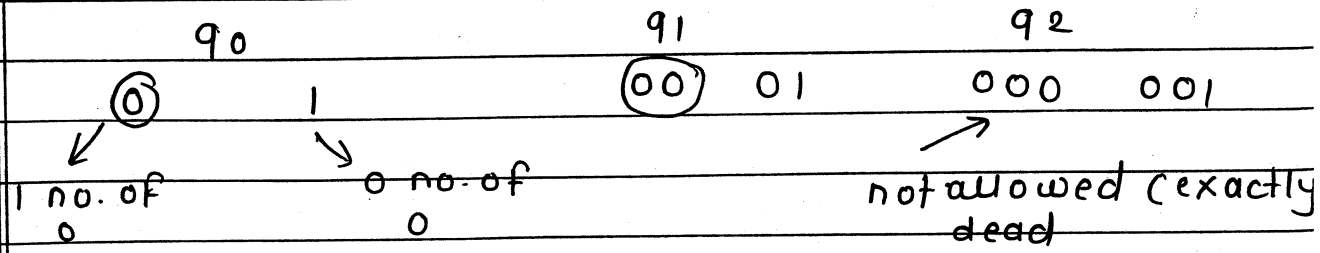
$L = \{ 00, 001, 010, 100, \ldots \}$



q_0 = can see 0 no. of 0 / start state

q_1 = can see 1 no. of 0

q_2 = can see 2 no. of 0



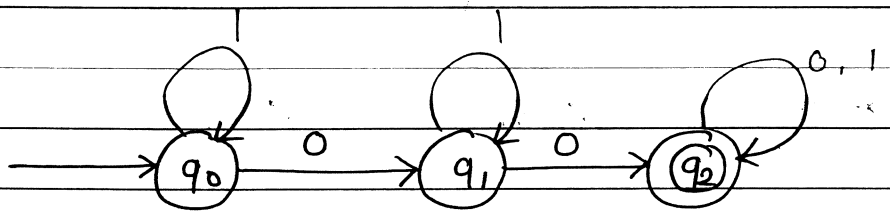
at least - min. two zeros and more than that

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

Design a DFA to accept all words contains at least two zeros in it.

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

$L = \{00, 000, 001, 010, 100, \ldots\}$



q_0 = ss/ can see 0 no. of 0
q_1 = can see 1 no. of 0
q_2 = can see 2 no. of 0

q_0 : 0, 1

q_1 : 00, 01

q_2 : 001, 000

If ε is in L at start, final ss is the final state

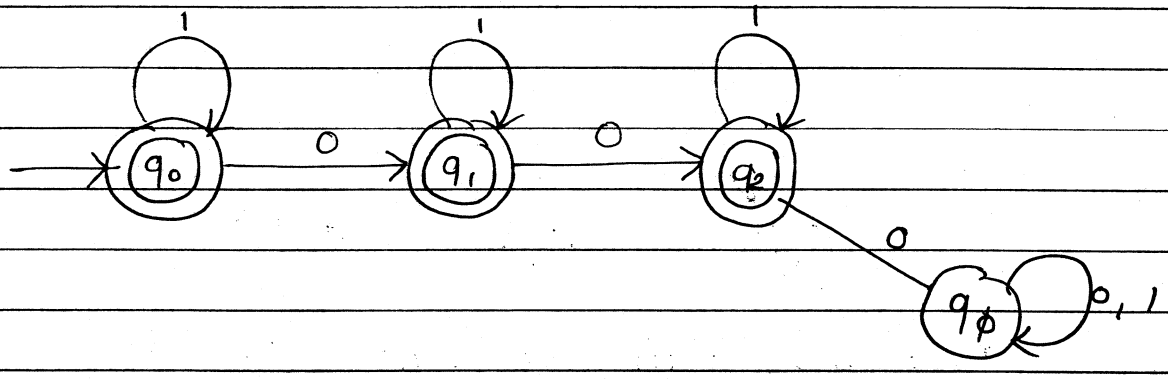
at most = max. 2 0's
min any zero

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

Accept words with <u>at most</u> two zero's

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

$L = \{ \varepsilon, 0, 1, 00, 01, 10, 11, 001, 010, \ldots \}$



q_0 : 0, 1

q_1 : 00, 01

q_2 : 000, 001

q_0 = ss / can see 0 no. of 0's

q_1 = can see 1 no. of 0

q_2 = can see 2 no. of 0

i/p / states	0	1
q_0	q_1	q_0
q_1	q_2	q_1
q_2		
q_ϕ		

Both 0 and 1 will combine to length

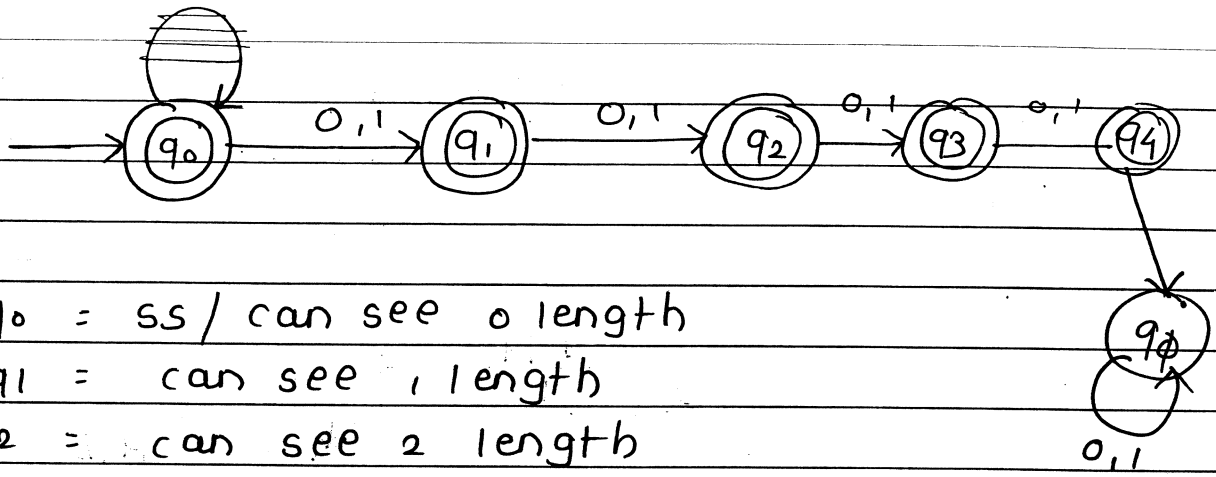
language is finite

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

Accept all words with length at most four

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

$L = \{ \varepsilon, 0, 1, 00, 000, 0000 \}$ finite language



q_0 = ss / can see 0 length

q_1 = can see 1 length

q_2 = can see 2 length

q_0		q_1
0	1	0

When input is two, then fork is used

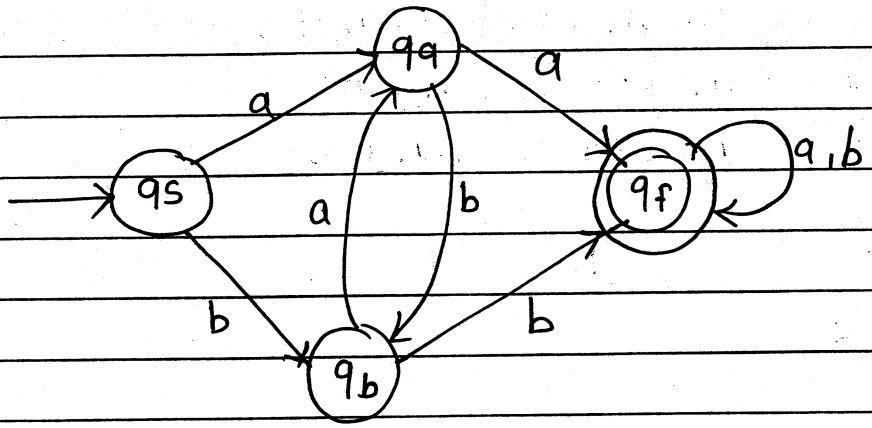
Misllanius Problems

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

Design DFA to accept all double letters

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

$L = \{ aa, bb, aaa, abb, baa, bbb, \ldots \}$



q_s = ss

q_a = can see a

q_b = can see b

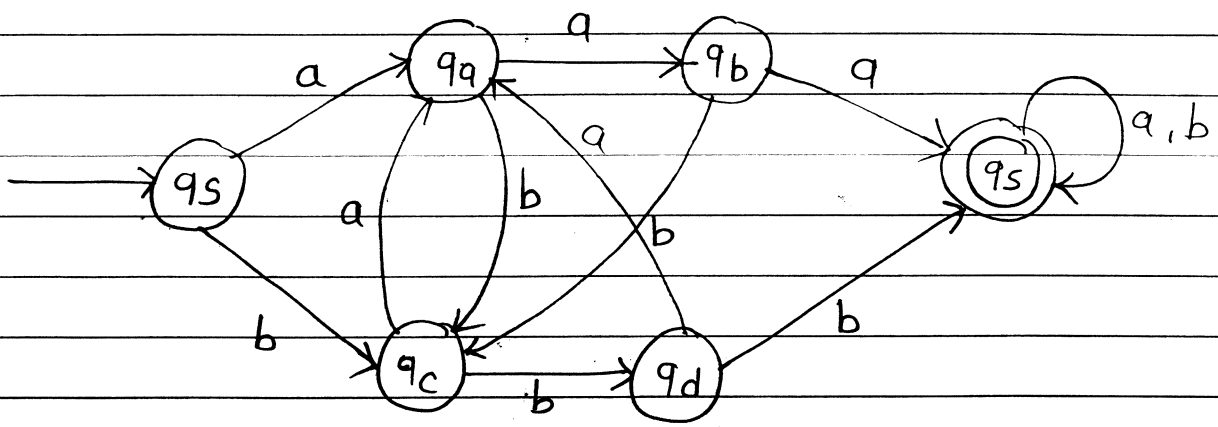
q_f = can see aa / bb

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

Design DFA to accept all triple letters

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

$L = \{ aaa, \underline{aaa}a, \underline{bbb}, \ldots \}$



q_s = s/s

q_a = can see a

q_b = can see aa

q_c = can see b

q_d = can see bb

q_s = can see aaa / bbb

babaabbb

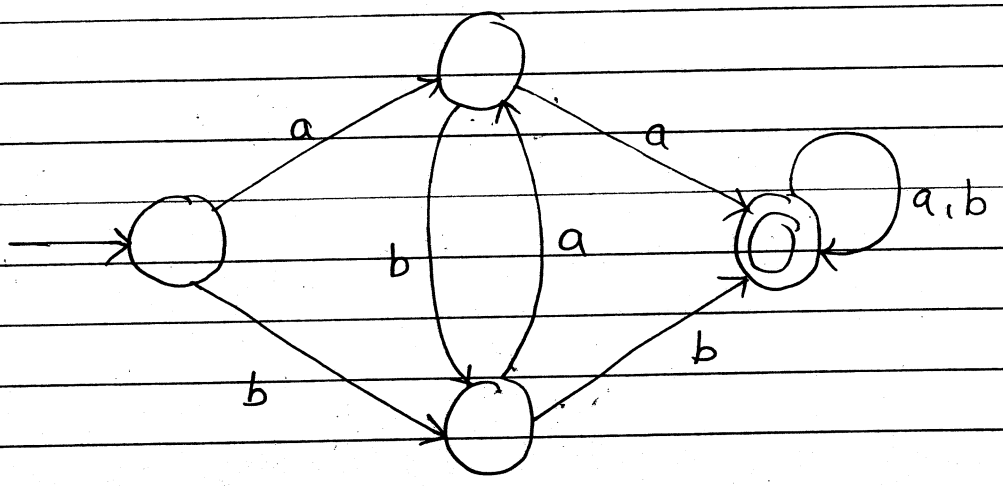
q_a		q_b		q_c		q_d	
(aa)	~~ab~~	(aaa)	~~aaab~~	~~ba~~	bb	~~bbb~~	bbb

*neither - nor problems can be solved by solving either - or and then interchanging the final & non-final states

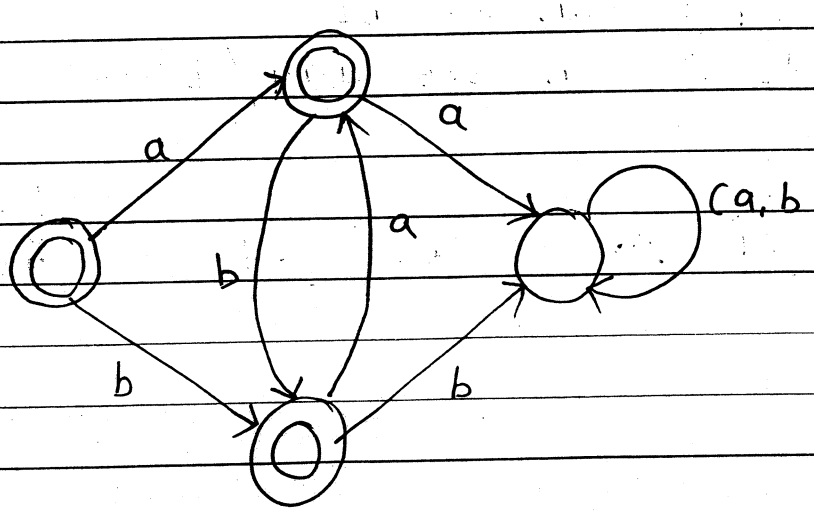
2) $\Sigma = \{a, b\}$

Design DFA to accept all words which consist of neither aa nor bb

Let's design it for accepting either aa or bb



But the expected language needs neither aa nor bb, So, that can be achieved by interchanging final and non final state



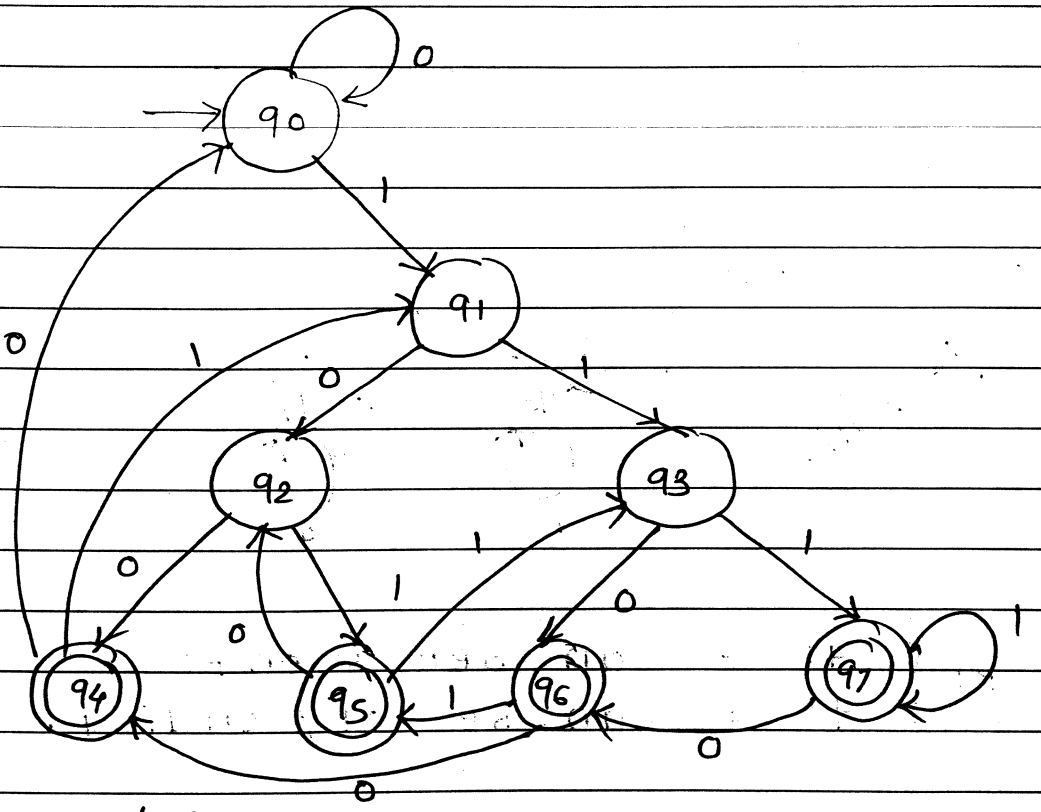
1100100

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

Design DFA to accept all strings, third symbol from the right end is one

3rd	2nd	1st
1	0	0
1	0	1
.	.	.
1	1	1

$L = \{100, 0100, 1100, 101, 0101, \ldots, 110, 0110, \ldots, 111, 0111, \ldots\}$



q_0 =	can see 0/ss
q_1 =	can see 1
q_2 =	can see 10
q_3 =	can see 11
q_4 =	can see 100
q_5 =	can see 101
q_6 =	can see 110
q_7 =	can see 111

q_6: ~~100~~ ~~101~~

q_7: ~~111~~ 110

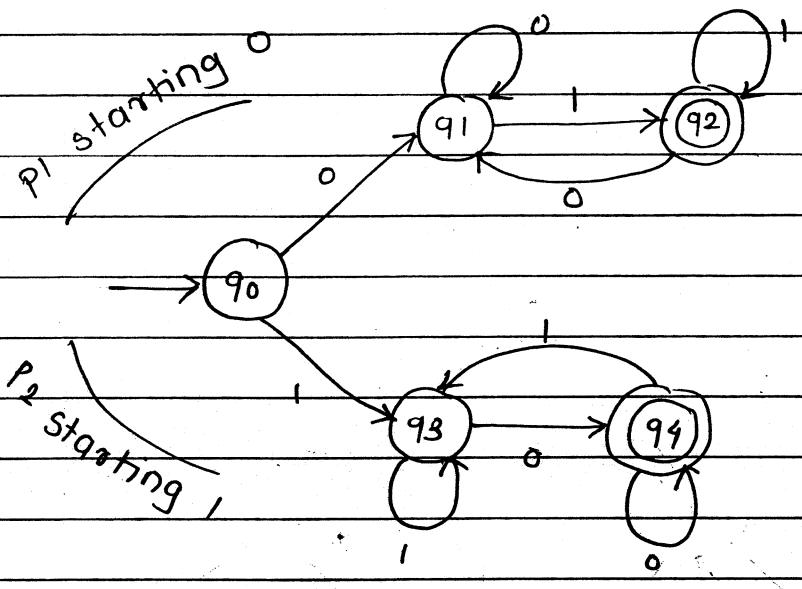
29) $\Sigma = \{0, 1\}$

Accept all strings where first symbol differs from last symbol.

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

$L = \{ 01, 0\,0\,1, \ldots, 10, 1\,0\,0 \}$

Parallel



$q_0 = SS$

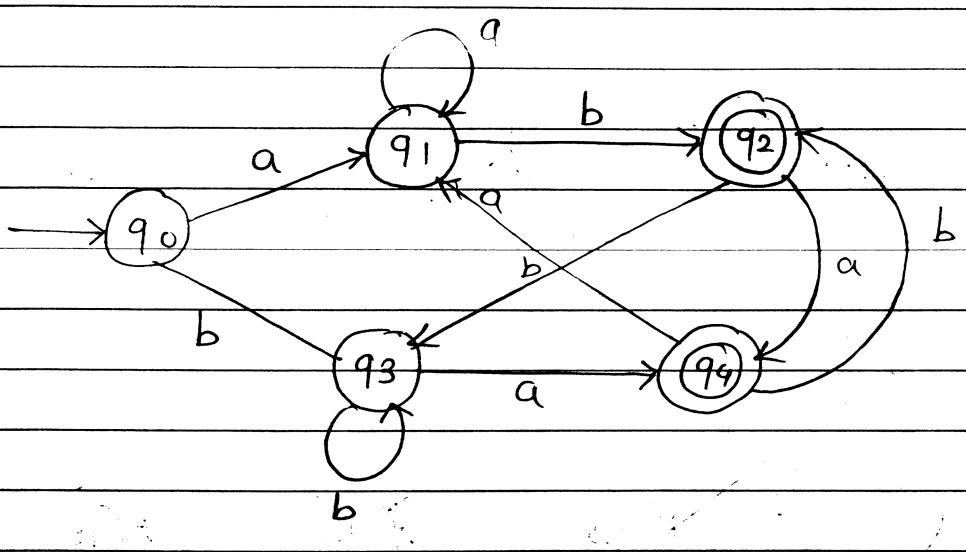
P1 { q_1 = can see 0 on P1
q_2 = can see 01 / ending 1 on P1

P2 { q_3 = can see 1 on P2
q_4 = can see 10 / ending 0 on P2

q_1	q_2	q_3	q_4
$\emptyset 0$ (01)	$\emptyset 1 0$ $\emptyset 1 1$	(10) 11	$\times \emptyset 0$ $\times \emptyset 1$

24) Design DFA to accept strings ending with either BA or AB.

$L = \{ ab, aab, bab, ba, b, aba, bba, \ldots \}$



q_0 = ss

q_1 = can see a

q_2 = can see ab

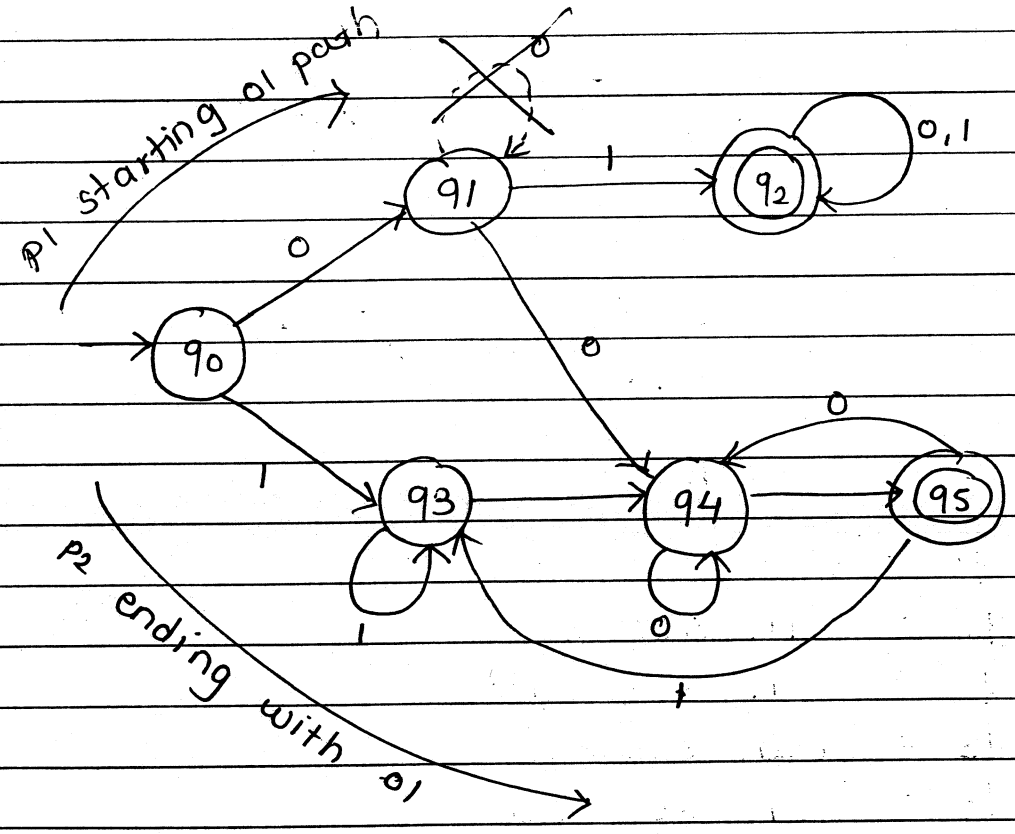
q_3 = can see b

q_4 = can see ba

q_1	q_2	q_3	q_4
~~aa~~ (ab)	~~aba~~	~~abb~~	(ba) ~~bb~~	~~baa~~ ~~bab~~

25) Design DFA for set of all strings such that string either begin with 01 or end with 01 (either – or)

$$L = \{ 01, 011, 001, 101, \ldots \}$$



q_0 = SS

P1 { q_1 = can start 0 on P1
   { q_2 = can see 01 on P1

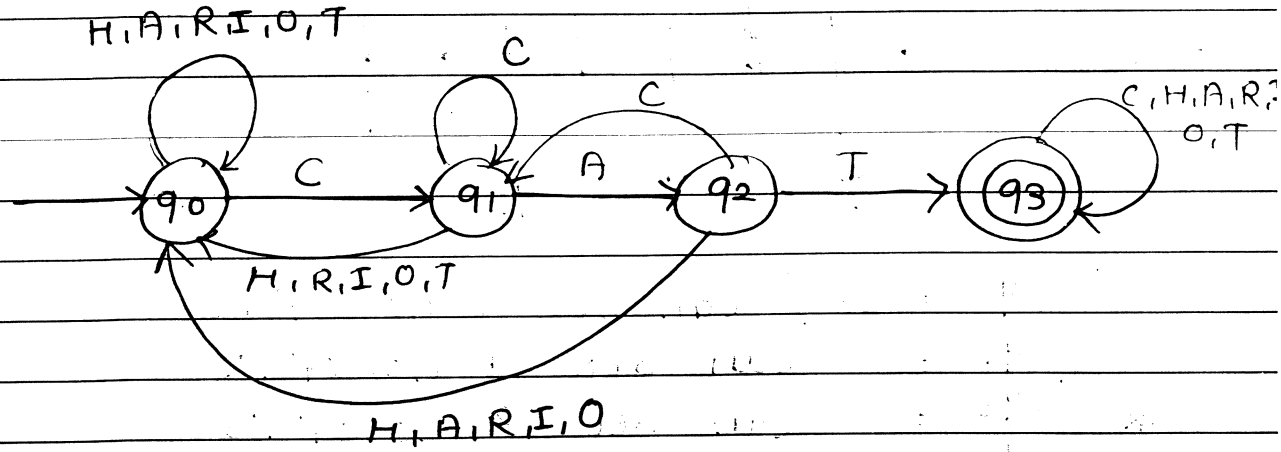
P2 { q_3 = can see starting 1 / 1 on P2
   { q_4 = can see 10 / 0 on P2
   { q_5 = can see 01 / ending 1 on P2

BM vimp most *

26) Design a DFA to recognize the substring 'CAT' over the word 'CHARIOT'

$\Sigma = \{ C, H, A, R, I, O, T \}$

$L = \{ CAT, CAT\_, \_CAT, \dots \}$



q_0 = SS / H A R I O T

q_1 = can see C

q_2 = CA

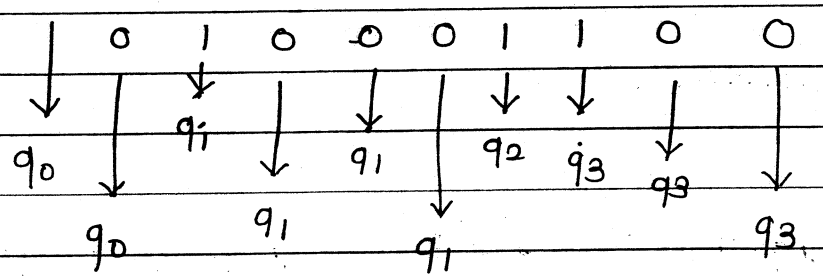
q_3 = CAT

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

Design a DFA to accept all binary numbers where no of 1's is multiple of 3

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

$L = \{111, \text{-----}\}$



q_0 = ss / can see 0 no. of 1's
q_1 = can see 1 no. of 1's
q_2 = can see 2 no. of 1's
q_3 = can see 3 no. of 1's

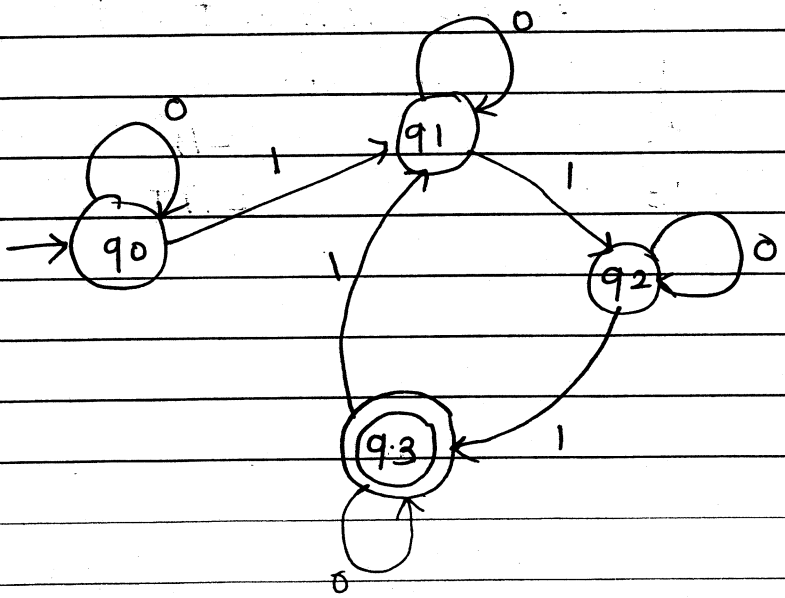
Counting of no. of 1's.

so no elimination tech. used.

q_0
0 1

q_2
110 111

q_3
10 11

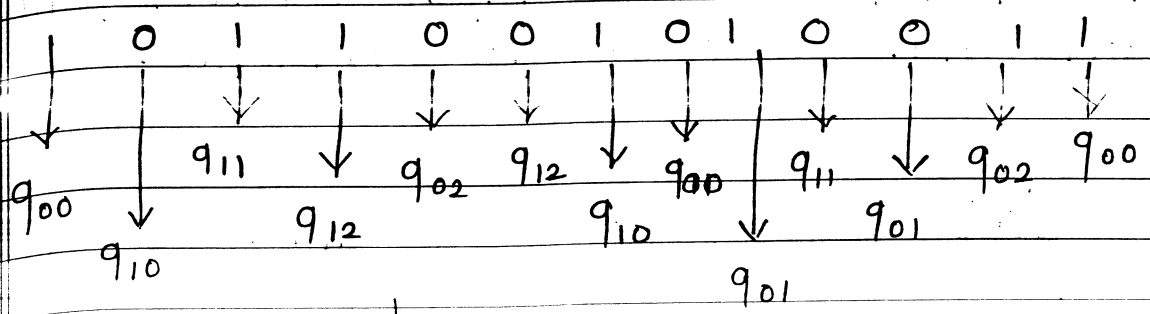


28) Practical Mimp

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

Design DFA to accept no. of 0's divisible by 2 and no. of 1's divisible by 3

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



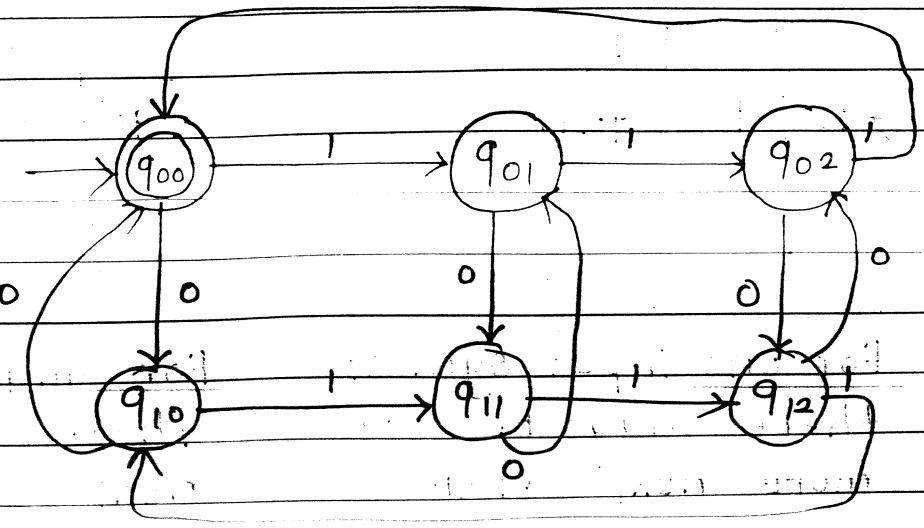
divisible by 2	divisible by 3
no. of 0's	no. of 1's
rem 0	rem 0
rem 1	rem 1
	rem 2
$\Downarrow$	$\Downarrow$
i	j

	0	1
$\rightarrow q_{00}$*	q_{10}	q_{01}
q_{01}	q_{11}	q_{02}
q_{02}	q_{12}	q_{00}
q_{10}	q_{00}	q_{11}
q_{11}	q_{01}	q_{12}
q_{12}	q_{02}	q_{10}

q_{ij}

$i = 0, 1$

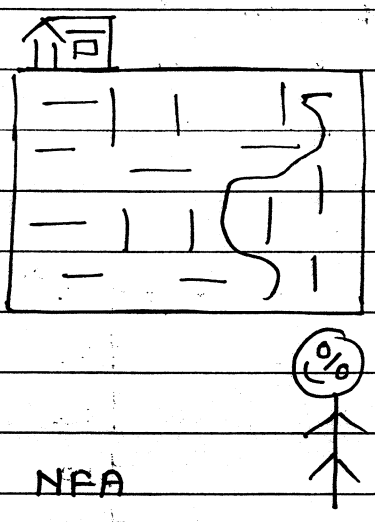
$j = 0, 1, 2$



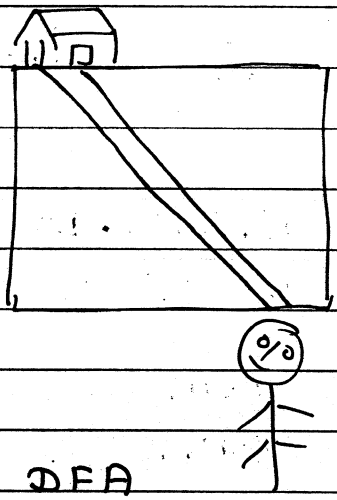
NFA

NFA is a finite automata without output. It is general machine as compared to DFA.

* In a DFA, there is always a deterministic move for every i/p symbol
for NFA, for one input symbol, we may have zero or more transitions

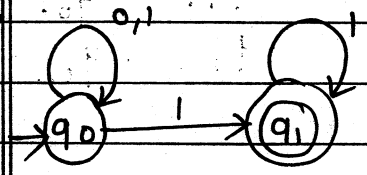


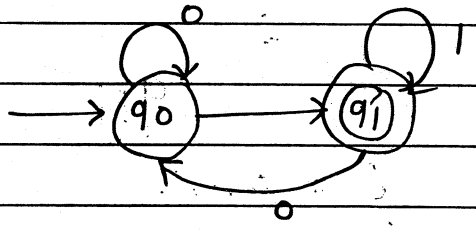
NFA



DFA

1)





2)

	0	1
→ q0	q0	q0, q1
q1	ϕ	q1

	0	1
→ q0	q0	q1
q1*	q0	q1

3) For one input symbol there may be 0 or more next state

For any input system, there is always one next state

4)	It is a general m/c so time complexity of travelling is more	It is a precise machi so time complexity of travelling is less
5)	Easy to design	Tough to design

Technical defination of NFA :

NFA can be defined as 5 tuppel machine $(Q, \Sigma, \delta, q_0, F)$ where

Q = finite set of states

Σ = finite set of input

q_0 = initial set $\quad q_0 \in Q$

F = final set $\quad F \subseteq Q$

DFA : δ : STF

P.S $\times$ I.P $\Rightarrow$ N.S

$Q_n \times \Sigma \Rightarrow Q_{n+1}$

NFA : δ : STF

P.S $\times$ I.P $\Rightarrow$ zero or more states

$Q \times \Sigma \Rightarrow P(Q)$

Equivalence of NFA and DFA

Every NFA can be converted into equivalent DFA.

$L(NFA) = L(DFA)$

NFA $\Rightarrow$ DFA

Given NFA $= (Q, \Sigma, \delta, q_0, F)$	DFA $= (Q', \Sigma', \delta', q_0', F')$
Q = finite no. of states	$Q' = P(Q)$
Σ = finite no. of inputs	$\Sigma' = \Sigma$
q_0 = initial state	$q_0' = \{q_0\}$
F = final state	F' = All sets of Q' where F is present.

P.S $\times$ i/P = N.S

$$\delta'(\{q_0, q_1, \dots q_n\}, q) = \delta(q_0, q) \cup \delta(q_1, q) \cup \dots \cup \delta(q_n, q)$$

classmate
Date
Page

eg 1)

NFA

$Q = \{q_0, q_1\}$

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

$q_0 = q_0$

$F = q_1$

DFA

$Q' = P(Q)$

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

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

$q_0' = \{q_0\}$

$F' = \{q_1\}, \{q_0, q_1\}$

2)

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

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

$q_0 = q_0$

$F = q_2$

$Q' = P(Q)$

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

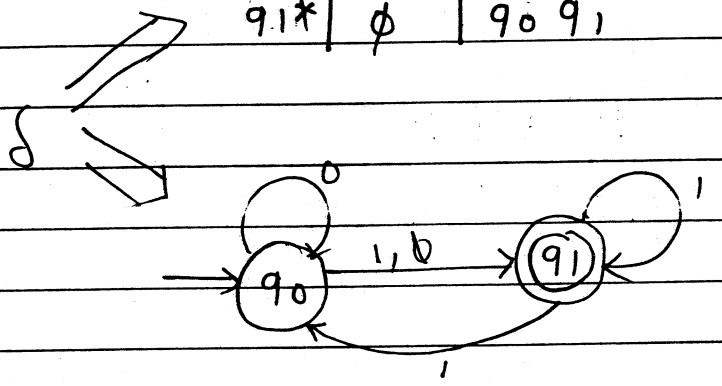
$\Sigma' = \Sigma = \{0, 1\}$

$q_0' = \{q_0\}$

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

29) Convert following NFA into equivalent DFA

	0	1
→ q_0	q_0, q_1	q_1
q_1*	ϕ	$q_0\ q_1$



$Q = \{ q_0, q_1 \}$

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

$q_0 = q_0$

$F = q_1$

Equivalent DFA can be given as :

$DFA = \{ Q', \Sigma', \delta', q_0', F' \}$

where,

$Q' = P(Q) = \{ \phi, \{q_0\}, \{q_1\}, \{q_0, q_1\} \}$

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

$q_0' = \{q_0\}$

$F' = \{q_1\} \{q_0, q_1\}$

$\delta' (\{ state, i/p)$

$= \delta (state, i/p) \cup \delta (state, i/p), ----$

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

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

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

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

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

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

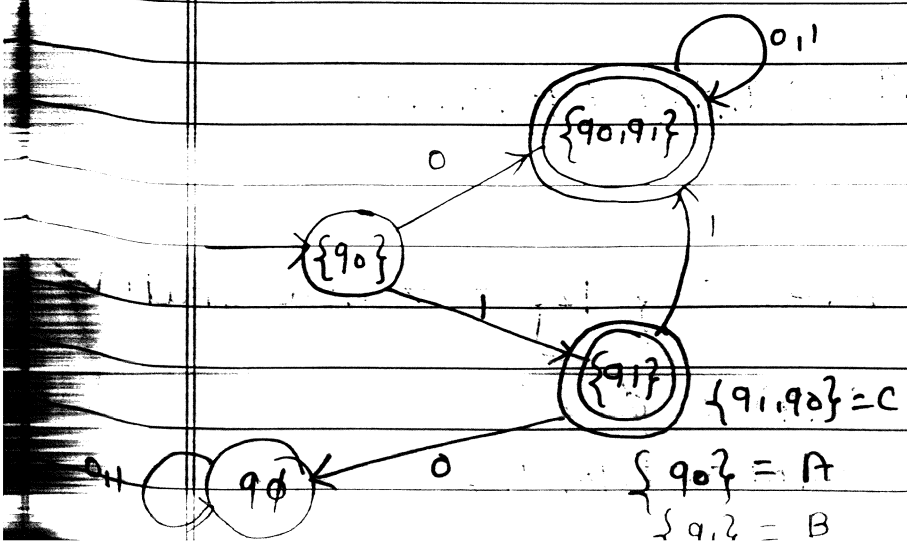
$= \{q_0, q_1\}$

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

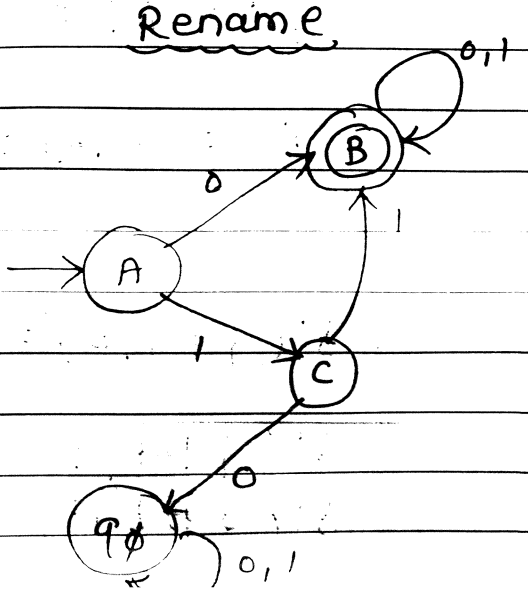
$= \{q_1\} \cup \{q_0, q_1\}$

$= \{q_0, q_1\}$

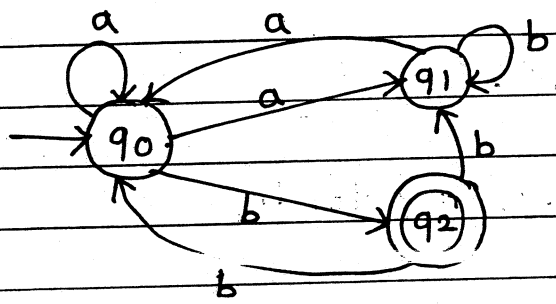
States \ input	0	1
→ $\{q_0\}$	$\{q_0, q_1\}$	$\{q_1\}$
$\{q_1\}^*$	ϕ	$\{q_0, q_1\}$
$\{q_0, q_1\}^*$	$\{q_0, q_1\}$	$\{q_0, q_1\}$



Rename



30) Convert following NFA into equivalent DFA (New Subset Generation Method)



	a	b
→ q_0	$q_0 q_1$	q_2
q_1	q_0	q_1
q_2*	ϕ	$q_0 q_1$

$Q' = P(Q) = \{ \phi, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\} \{q_1, q_2\}, \{q_0, q_2\}, \{q_0, q_1, q_2\} \}$

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

$q_0' = \{q_0\}$

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

$\delta'(\{q_0\}, a) = \delta(q_0, a) = \{q_0, q_1\} \longrightarrow$ new state

$\delta'(\{q_0\}, b) = \delta(q_0, b) = \{q_2\} \longrightarrow$ new state ✓

$\delta'(\{q_2\}, a) = \delta(q_2, a) = \phi$ ✓

$\delta'(\{q_2\}, b) = \delta(q_2, b) = \{q_0, q_1\}$

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

$= \{q_0q_1\} \cup \{q_0\}$

$= \{q_0,q_1\}$

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

$= \{q_2\} \cup \{q_1\}$

$= \{q_1,q_2\} \longrightarrow$ new state

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

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

$= \{q_0\}$ ✓

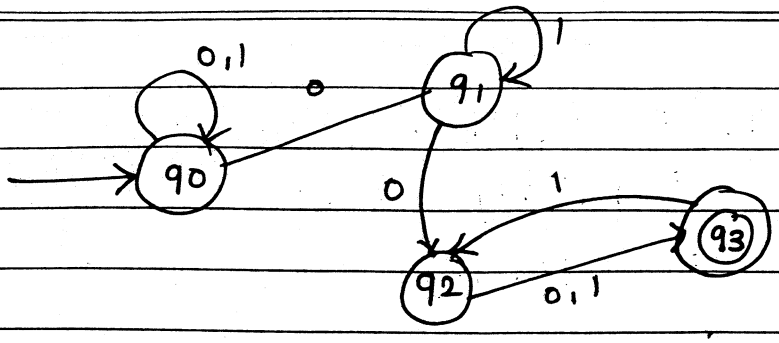
$\delta'(\{q_1,q_2\},b) = \delta(q_1,b) \cup \delta(q_2,b)$

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

$= \{q_0,q_1\}$ ✓

	a	b
$\rightarrow \{q_0\}$	$\{q_0,q_1\}$	$\{q_2\}^*$
$\{q_2\}^*$	ϕ	$\{q_0,q_1\}$
$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_1,q_2\}^*$
$\{q_1,q_2\}^*$	$\{q_0\}$	$\{q_0,q_1\}$

31)



32)

	0	1
P	P q	P
q	r	r
r	S	ϕ
S*	S	S

ε-NFA ⇒ (ε = null move)

ε-NFA = $(Q, \Sigma, \delta, q_0, F)$

Q = finite number of states
Σ = finite number of inputs
q_0 = initial state $q_0 \in Q$
F = final state $F \subseteq Q$

DFA : δ : STF

P.S x i/P ⇒ N.S

$Q \times \Sigma \Rightarrow Q$

NFA : δ : STF

P.S x N/P ⇒ zero / more next state

$Q \times \Sigma \Rightarrow P(Q)$

ε-NFA : δ : STF

P.S x { i/P ∪ ε } ⇒ zero / more next state

$Q \times \{ \Sigma \cup \varepsilon \} \Rightarrow P(Q)$

ε-NFA are the most general machine where machine can move from one state to other state without receiving any input symbol.

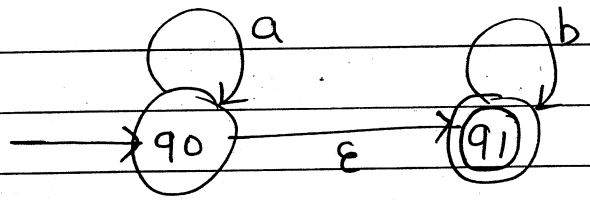
There always a presence of ε if there is a logical path

Anything with ε = ε with anything
= Anything

33) Design ε-NFA for any number of A's followed by any number of B's

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

$L = \{\varepsilon, a, b, aa, ab, ba, \ldots\}$

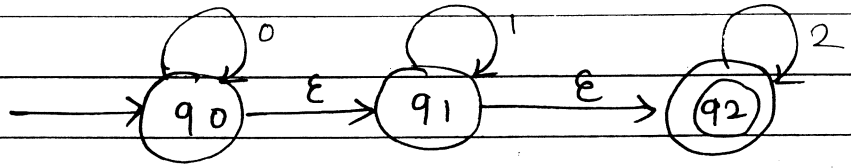


i/p / State	a	b	ε
→ q0	q0	ϕ	q1
q1*	ϕ	q1	ϕ

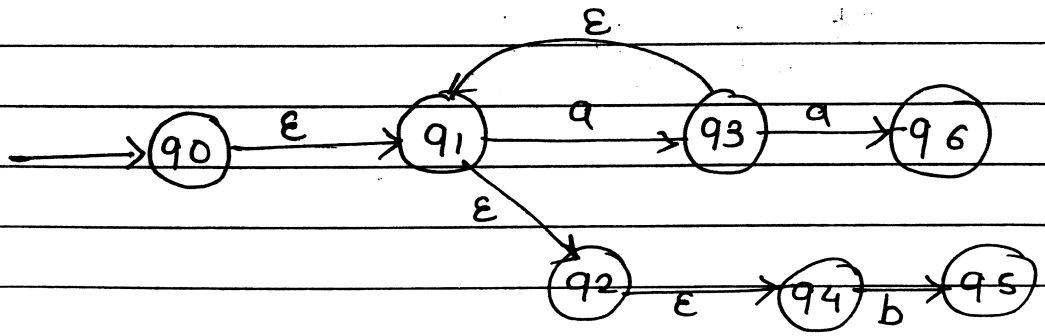
ε - closure of a state

closure = all possibilities

ε - closure of a state is
a set which consists of
the state itself
and all other states to which it can
be reached by any number of
ε moves.



ε closure (q0) = {q0, q1, q2}
ε closure (q1) = {q1, q2}
ε closure (q2) = {q2}



ε closure(q0) = {q0, q2, q4, q5, q1}
ε closure(q1) = {q1, q2, q4}
ε closure(q2) = {q2, q4}
ε closure(q3) = {q3, q1, q2, q4}
ε closure(q4) = {q4}
ε closure(q5) = {q5}
ε closure(q6) = {q6}

classmate Date Page

*Equivalence of ε-NFA to NFA and to DFA

ε NFA
= (Q, Σ, δ, q0, F)

Q = states

Σ = input alphabets

q0 = initial state

f = final state

ε NFA
$Q \times (\Sigma \cup \varepsilon) \Rightarrow P(Q)$
NFA
$Q \times \Sigma \Rightarrow P(Q)$

NFA
$(Q'', \Sigma'', \delta'', q_0'', F'')$

$Q'' = Q$

$\Sigma'' = \Sigma$ w/o ε

$q_0'' = q_0$

$F'' = \{ F \cup q_0$ iff e-clos(q0) consist of F else only F $\}$

$\delta''(state, i/p) = \varepsilon\text{-clos}(\delta(\varepsilon\text{-clo}(state), i/p))$

DFA
$(Q', \Sigma', \delta', q_0', F')$

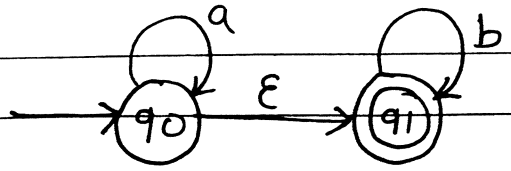
$Q' = P(Q'')$

$\Sigma' = \Sigma''$

$q_0' = \{q_0''\}$

F' = All sets of Q' which consist of F'' into it.

34)



	a	b	ε
→ q_0	q_0	ϕ	q_1
q_1*	ϕ	q_1	ϕ

$Q = \{q_0, q_1\}$

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

$q_0 = q_0$

$F = q_1$

$\varepsilon\text{-clo}(q_0) = \{q_0, q_1\}$

$\varepsilon\text{-clo}(q_1) = \{q_1\}$

$NFA = (Q'', \Sigma'', \delta'', q_0'', F'')$

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

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

$q_0'' = q_0$

$F'' = q_0, q_1$

$\delta''(state, i/p)$

$= \varepsilon\text{-clo}(\delta(\varepsilon\text{-clo}(state), i/p))$

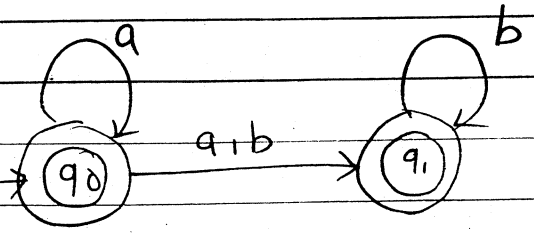
$$\delta''(q_0, a) = \varepsilon\text{-clo}(\delta(\varepsilon\text{-clo}(q_0), a))$$
$$= \varepsilon\text{-clo}(\delta(q_0, q_1), a)$$
$$= \varepsilon\text{-clo}(\delta(q_0, a) \cup \delta(q_1, a))$$
$$= \varepsilon\text{-clo}(q_0 \cup \phi)$$
$$= \varepsilon\text{-clo}(q_0)$$
$$= \{q_0, q_1\}$$

$$\delta''(q_0, b) = \varepsilon\text{-clo}(\delta(\varepsilon\text{-clo}(q_0), b))$$
$$= \varepsilon\text{-clo}(\delta(q_0, q_1), b)$$
$$= \varepsilon\text{-clo}(\delta(q_0, b) \cup \delta(q_1, b))$$
$$= \varepsilon\text{-clo}(\phi \cup q_1)$$
$$= \varepsilon\text{-clo}(q_1)$$
$$= q_1$$

$$\delta''(q_1, a) = \varepsilon\text{-clo}(\delta(\varepsilon\text{-clo}(q_1), a))$$
$$= \varepsilon\text{-clo}(\delta(q_1), a)$$
$$= \varepsilon\text{-clos}(\phi)$$
$$= \phi$$

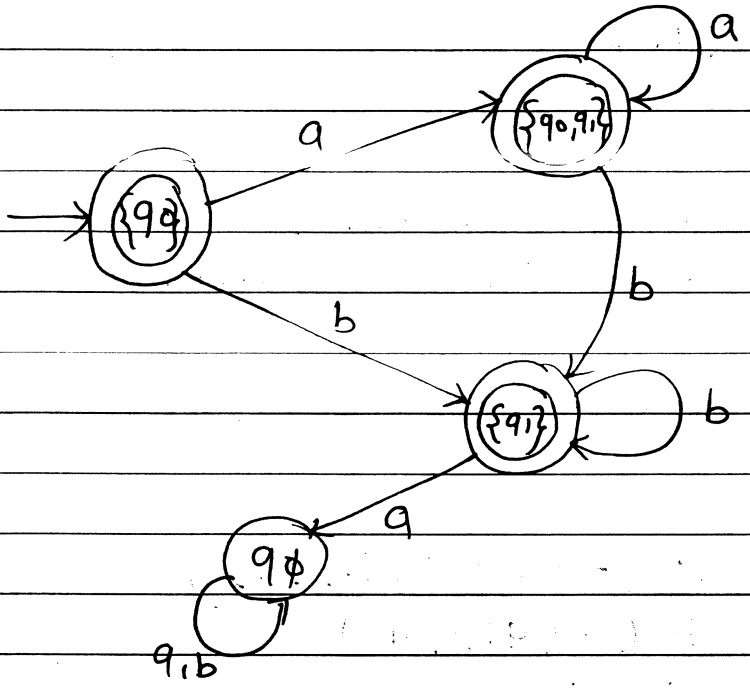
$$\delta''(q_1, b) = \varepsilon\text{-clo}(\delta(\varepsilon\text{-clo}(q_1), b))$$
$$= \varepsilon\text{-clo}(\delta(q_1), b)$$
$$= \varepsilon\text{-clo}(q_1)$$
$$= q_1$$

	a	b
q_0*	q_0, q_1	q_1
q_1*	ϕ	q_1

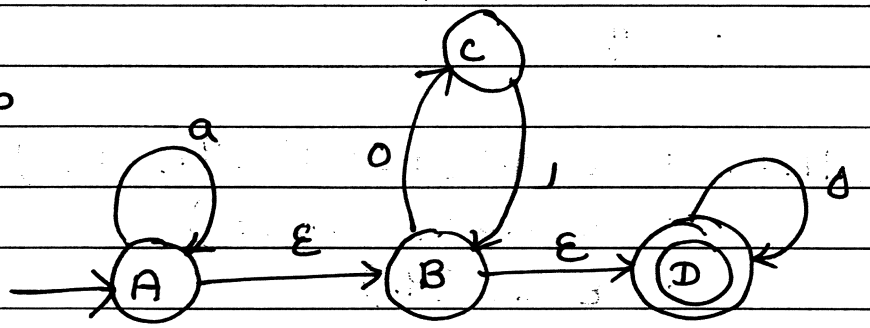


$$L = \{\varepsilon, a, b, aa, ab, bb, \ldots\}$$

	a	b
$\{q_0\}^*$	$\{q_0, q_1\}$	$\{q_1\}$
$\{q_1\}^*$	ϕ	$\{q_1\}$
$\{q_0, q_1\}^*$	$\{q_0, q_1\}$	$\{q_1\}$



35) MIMP



	0	1	ε
A	A	ϕ	B
B	C	ϕ	D
C	ϕ	B	ϕ
D	D	ϕ	ϕ

$\varepsilon\text{-clo}(A) = \{A, B, D\}$

$\varepsilon\text{-clo}(B) = \{B, D\}$

$\varepsilon\text{-clo}(C) = \{C\}$

$\varepsilon\text{-clo}(D) = \{D\}$

$Q'' = \{A, B, C, D\}$

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

$q_0'' = A$

$F'' = D, A$

July

$\delta''(\text{state}, i/p) = \varepsilon\text{-clo}(\delta(\varepsilon\text{-clo}(\text{state}), i/p))$

$\delta''(A, 0) = \varepsilon\text{-clo}(\delta(\varepsilon\text{-clo}(A), 0))$

$= \varepsilon\text{ clo}(\delta\{A, B, D\}, 0))$

$= \varepsilon\text{-clo}(\delta(A, 0) \cup \delta(B, 0) \cup \delta(D, 0))$

$= \varepsilon\text{-clo}(A \cup C \cup D)$

$= \varepsilon\text{-clo}(\{A, C, D\})$

$= \varepsilon\text{ clo}(A) \cup \varepsilon\text{-clo}(C) \cup \varepsilon\text{-clo}(D)$

$= \{A, B, D\} \cup \{C\} \cup \{D\}$

$= \{A, B, C, D\}$

$\delta''(A, 1) = \varepsilon\text{-clo}(\delta(\varepsilon\text{-clo}(A), 1))$

$= \varepsilon\text{-clo}(\delta\{A, B, D\}, 1))$

$= \varepsilon\text{-clo}(\delta(A, 1) \cup \delta(B, 1) \cup \delta(D, 1))$

$= \varepsilon\text{-clo}(\phi \cup \phi \cup \phi)$

$= \varepsilon\text{-clo}(\phi)$

$= \phi$

$$\delta''(B,0) = \varepsilon\text{-clo}(\delta(\varepsilon\text{-clo}(B),0))$$
$$= \varepsilon\text{-clo}(\delta(\{B,D\},0))$$
$$= \varepsilon\text{-clo}(\delta(B,0) \cup \delta(D,0))$$
$$= \varepsilon\text{-clo}(C \cup D)$$
$$= \varepsilon\text{-clo}\{C,D\}$$
$$= \varepsilon\text{-clo}(C) \cup \varepsilon\text{-clo}(D)$$
$$= \{C\} \cup \{D\}$$
$$= \{C,D\}$$

$$\delta''(B,1) = \varepsilon\text{-clo}(\delta(\varepsilon\text{-clo}(B),1))$$
$$= \varepsilon\text{-clo}(\delta(\{B,D\},1))$$
$$= \varepsilon\text{-clo}(\delta(B,1) \cup \delta(D,1))$$
$$= \varepsilon\text{-clo}(\phi \cup \phi)$$
$$= \varepsilon\text{-clo}(\phi)$$
$$= \{\phi\}$$

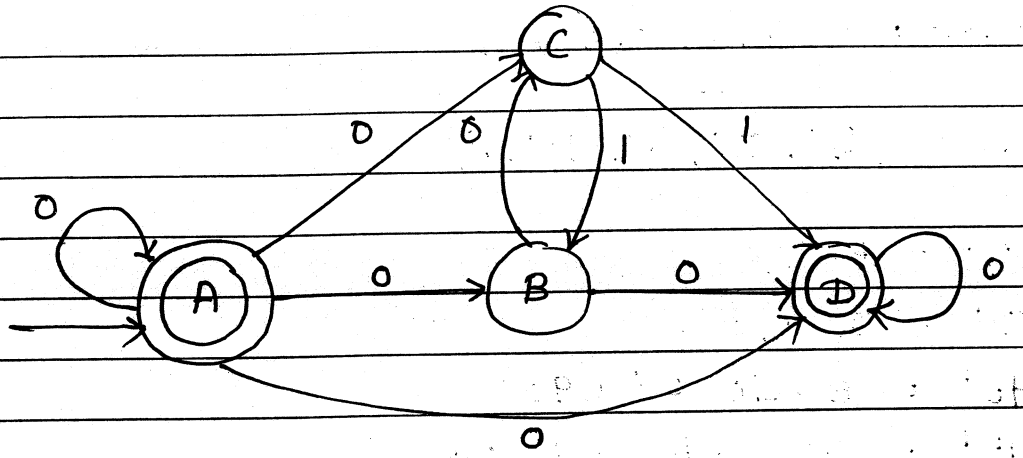
$$\delta''(C,0) = \varepsilon\text{-clo}(\delta(\varepsilon\text{-clo}(C),0))$$
$$= \varepsilon\text{-clo}(\delta(\{C\},0))$$
$$= \varepsilon\text{-clo}(\phi)$$
$$= \{\phi\}$$

$$\delta''(C,1) = \varepsilon\text{-clo}(\delta(\varepsilon\text{-clo}(C),1))$$
$$= \varepsilon\text{-clo}(\delta(\{C\},1))$$
$$= \varepsilon\text{-clo}(B)$$
$$= \{B,D\}$$

$$\delta''(D,0) = \varepsilon\text{-clo}(\delta(\varepsilon\text{-clo}(D),0))$$
$$= \varepsilon\text{-clo}(\delta(\{D\},0))$$
$$= \varepsilon\text{-clo}(D)$$
$$= \{D\}$$

$$\delta''(D,1) = \varepsilon\text{-clo}(\delta(\varepsilon\text{-clo}(D),1))$$
$$= \varepsilon\text{-clo}(\delta\{D\},1))$$
$$= \varepsilon\text{-clo}(\phi)$$
$$= \{\phi\}$$

	0	1
→ A *	A, B, C, D	ϕ
B	C, D	ϕ
C	ϕ	B, D
D	D	ϕ



Direct Method for ε-NFA to DFA

In this method, ε-closure of start state of ε-NFA is considered as start state of DFA
Then, new subset generation method can be used to calculate further states and transitions of DFA
This method may give optimized result (solution with less no. of states)

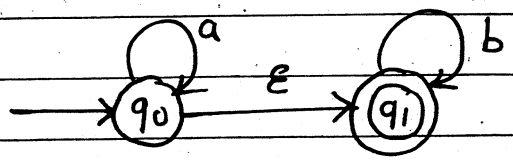
✓ $\epsilon - NFA = (Q, \Sigma, \delta, q_0, F)$

✓ $DFA = (Q', \Sigma', \delta', q_0', F)$

$Q' = P(Q)$
$\Sigma' = \Sigma$ w/o ϵ
q_0' = ε-clo of (q_0)
F' = All sets of Q' where F is present

$\delta'(state, i/p) = \epsilon\text{-clo}(\delta(state), i/p);$

36) Convert ε-NFA into DFA



	a	b	ε
→ q0	q0	ϕ	q1
q1	ϕ	q1	ϕ

Q = {q0, q1}

Σ = {a, b}

q0 = q0

F = 1

ε-clo(q0) = {q0, q1}

ε-clo(q1) = {q1}

DFA = (Q', Σ', δ', q0', F')

Q' = P(Q)

Σ' = Σ w/o ε

q0' = {q0, q1}

F' = All sets where q1 is present

δ'(state, i/p) = ε-clo(δ(state), i/p)

δ'({q0, q1}, a) = ε-clo(δ({q0, q1}, a))

= ε-clo(δ(q0, a) ∪ δ(q1, a))

= ε-clo(q0 ∪ ϕ)

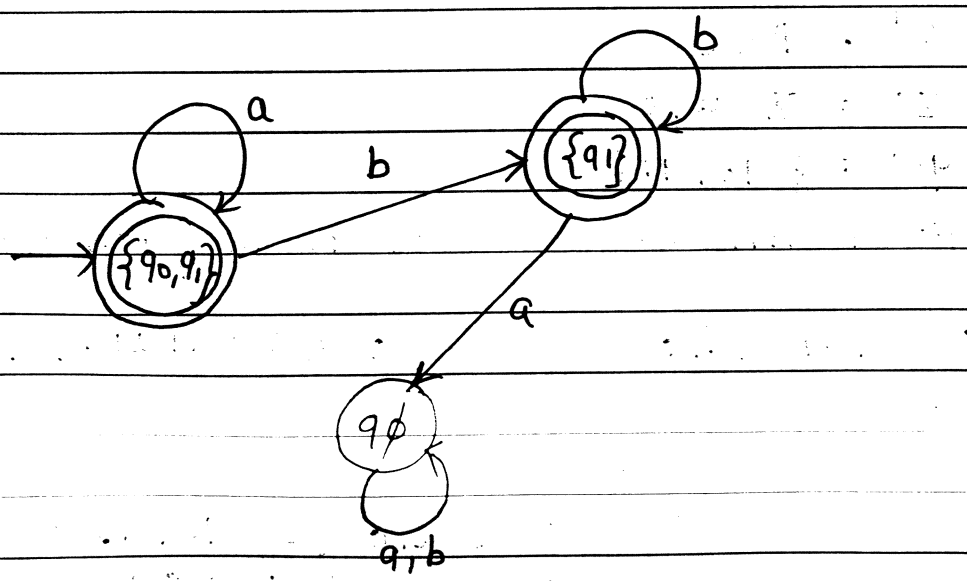
= {q0, q1} ✓

$$\delta'(\{q_0,q_1\}, b) = \varepsilon\text{-clo}(\delta(\{q_0,q_1\}, b))$$
$$= \varepsilon\text{-clo}(\delta(q_0,b) \cup \delta(q_1,b))$$
$$= \varepsilon\text{-clo}(\phi \cup q_1)$$
$$= \varepsilon\text{-clo}(q_1)$$
$$= \{q_1\} \longrightarrow \text{new state}$$

$$\delta'(\{q_1\}, a) = \varepsilon\text{-clo}(\delta(q_1,a))$$
$$= \varepsilon\text{-clo}(\phi)$$
$$= \phi \checkmark$$

$$\delta'(\{q_1\}, b) = \varepsilon\text{-clo}(\delta(q_1,b))$$
$$= \varepsilon\text{-clo}(q_1)$$
$$= \{q_1\} \checkmark$$

	a	b
$\{q_0,q_1\}^*$	$\{q_0,q_1\}$	$\{q_1\}$
$\{q_1\}^*$	ϕ	$\{q_1\}$



FA with Output

FA with output

||

Moore machine — Mealy machine

E.F. Moore (1955) — G.H. Mealy (1956)

⇓

6 Tuple machines

Does not final state

Moore and Mealy machines were developed for sequential circuit design and later on found useful for TOC as FA with O/P

Both the machines provide provide solution for same problem but will produce output in different way.

Moore / Mealy = $(Q, \Sigma, \delta, q_0, \lambda, \Delta)$

Q = finite set of states

Σ = finite set of inputs

Δ = finite set of outputs

q_0 = initial state

δ : STF

$Ps \times i/p \Rightarrow N.S$

$Q \times \Sigma \Rightarrow Q$

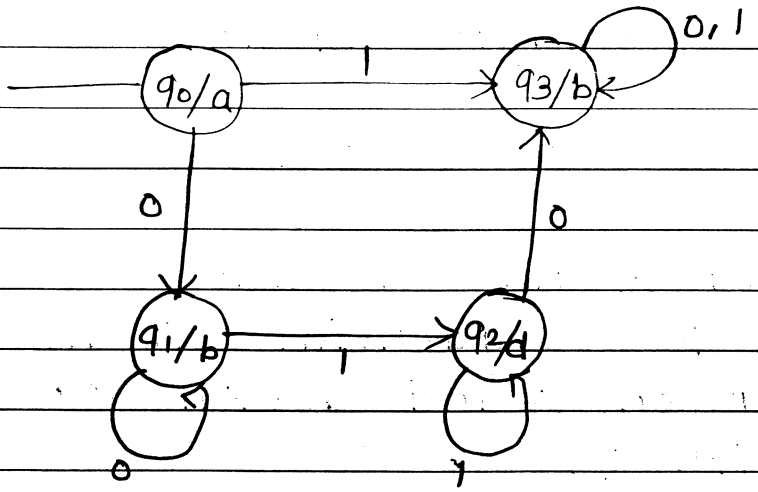
Moore - o/p is insta... state

classmate
Date
Page

λ = MAF / o/p function
(Mapping function / machine function)

- Moore ⇒ o/p is function of P.S
- Mealy ⇒ o/p is function of ps & i/p

Properties of Moore Machine



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

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

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

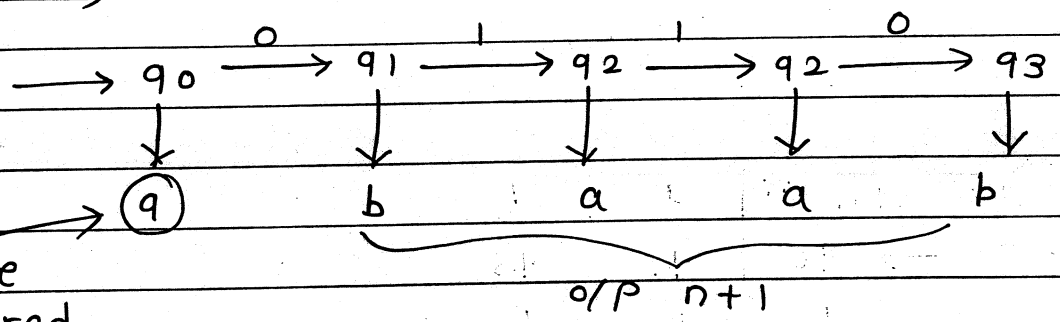
$q_0 = q_0$

states \ i/p	0	1
→ q_0	q_1	q_3
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_3	q_3

PS	O/P
q_0	a
q_1	b
q_2	a
q_3	b

PS	0	1	O/P
q_0	q_1	q_3	a
q_1	q_1	q_2	b
q_2	q_3	q_2	a
q_3	q_3	q_3	b

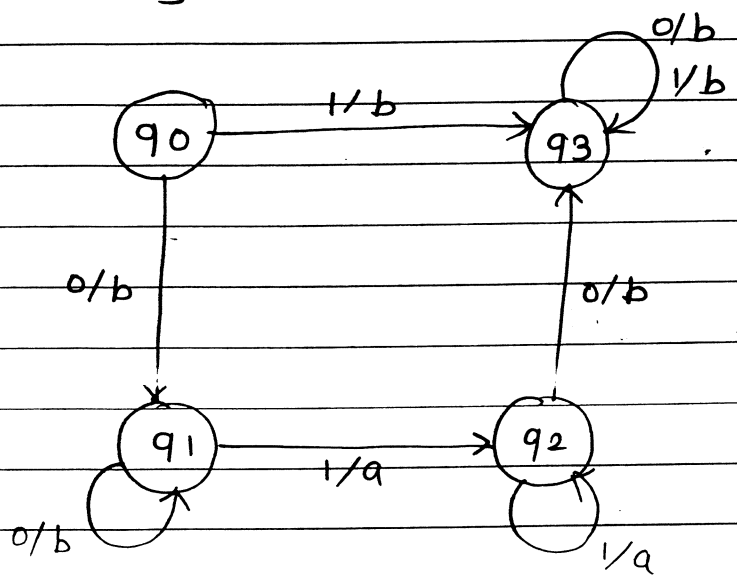
Handrun → 0 1 1 0 $n=4$



* Moore machine always responds to ε in its initial state

mealy = output is on transition line

Mealy Machine



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

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

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

$q_0 = q_0$

δ

state \ i/p	a	1
→ q_0	q_1	q_3
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_3	q_3

λ

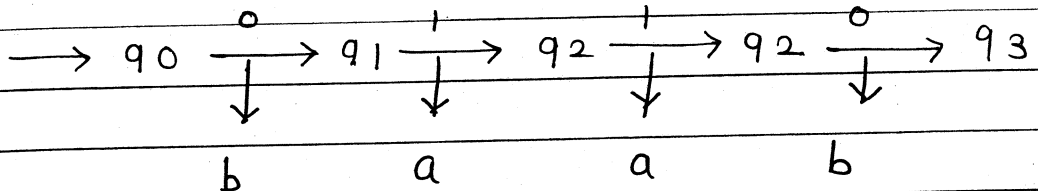
PS \ i/p	0	1
→ q_0	b	b
q_1	b	a
q_2	b	a
q_3	b	b

δ & λ

PS	NS, 1		NS, 0	
	i/p	o/p	i/p	o/p
→ q_0	q_1	b	q_3	b
q_1	q_1	b	q_2	a
q_2	q_3	b	q_2	a
q_3	q_3	b	q_3	b

Handrun → 0 1 1 0

n = 4



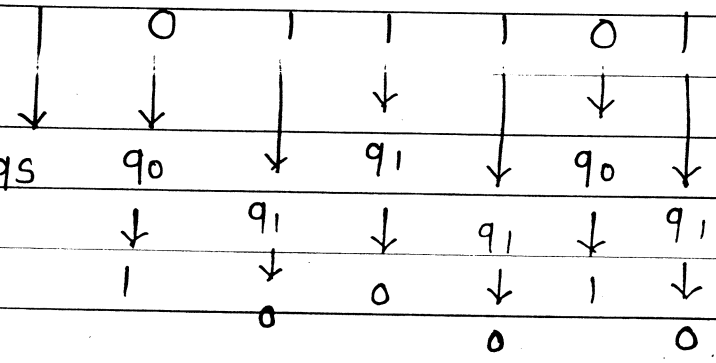
o/p n = 4

Informally, Moore and Mealy machines DFA with output

37) Design Moore and Mealy machines to calculate 1's complement of given binary number

$\Sigma = \{0, 1\}$ → binary

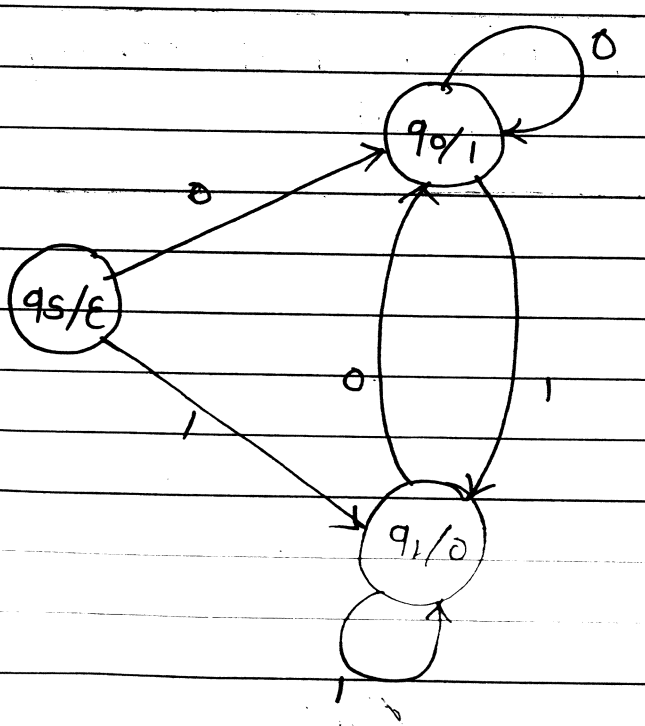
$\Delta = \{0, 1\}$



qs = start state

q0 = can see 0 and produce 1

q1 = can see 1 and produce 0



qs

0 1

q0

ϕ0 ϕ1

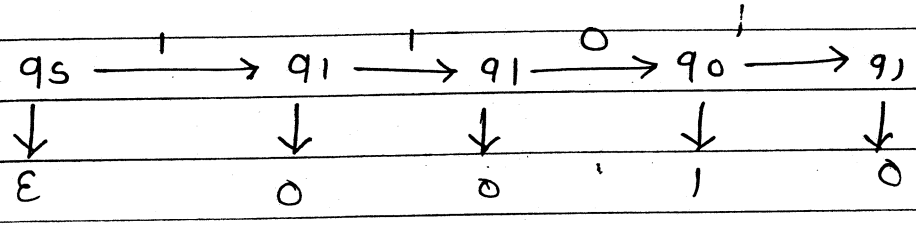
q1

ϕ0 ϕ1

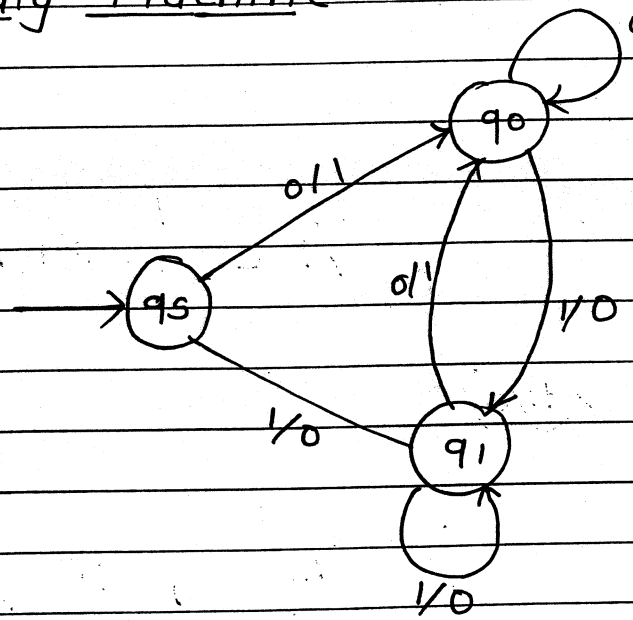
PS	NS 0	NS 1	O/P
q_S	q_0	q_1	ϵ
q_0	q_0	q_1	1
q_1	q_0	q_1	0

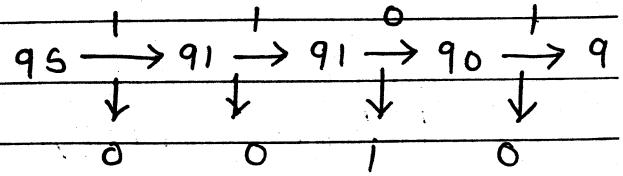
Handrun 1 1 0 1

complement 0 0 1 0



Mealy Machine





	NS i/p 0	O/p	NS i/p 1	O/p
→ q_S	q_0	1	q_1	0
q_0	q_0	1	q_1	0
q_1	q_0	1	q_1	0
	⋮			

substring enjoys self loop in final state

(0) → no final state in moore & Mealy

38) Design Moore and Mealy machines for binary input sequence if it has substring 101, machine produce output A if, it has substring 110 machin produces output B, else machine produces output C

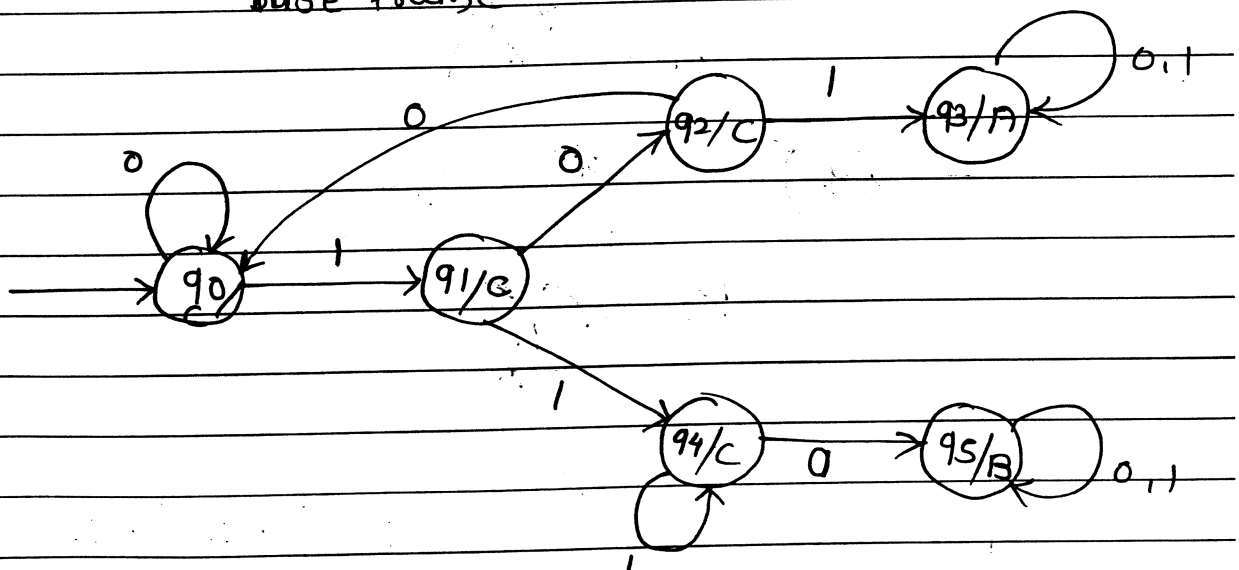
imp / 6M

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

$\Delta = \{A, B, C\}$

$L = \{ 101, 110, 101\,0, 0\,101, 110\,0, 0\,110, \ldots \}$

base frame



q0 = start start / can see 0 / C

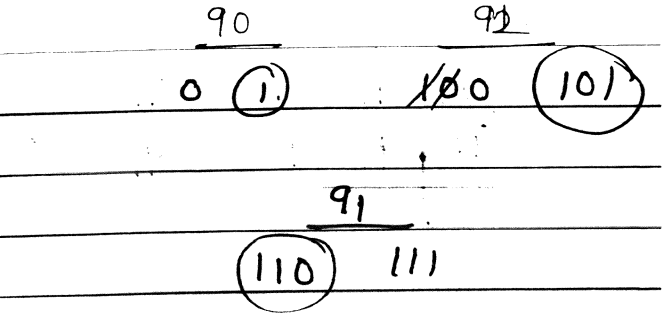
q1 = can see 1 / C

q2 = can see 10 / C

q3 = can see 101 / A

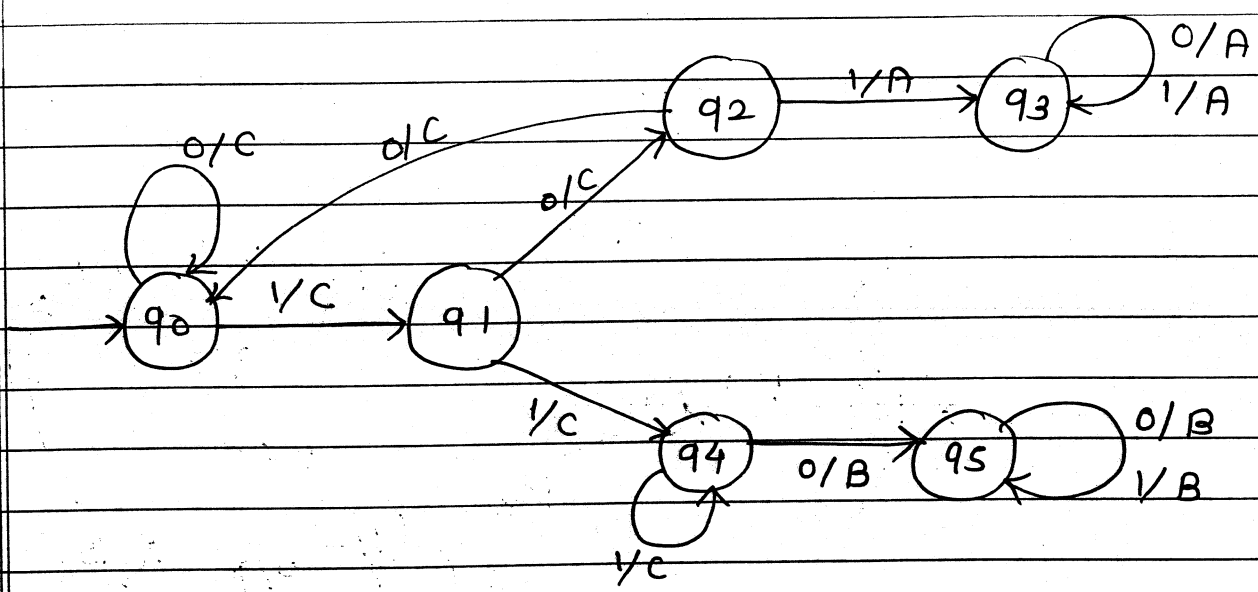
q4 = can see 11 / C

q5 = can see 110 / B



PS	NS		O/P
	0	1	
q_0			
q_1			
q_2			
q_3			
q_4			
q_5			

Mealy Machine

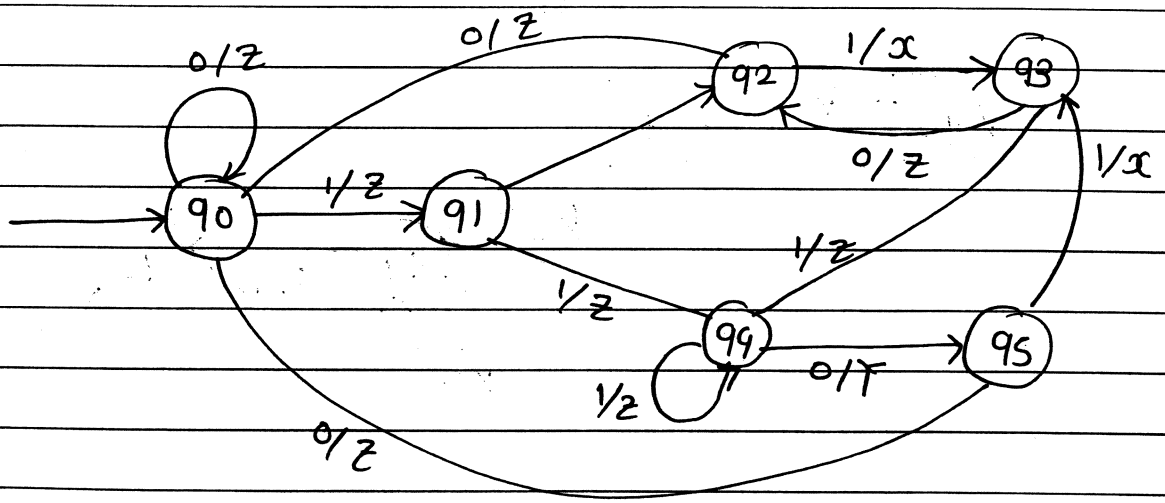


39) Design Mealy machine to accept all string in following way
for input from $(0+1)^*$ if input ends 101 then output = x.
if iput ends with 110 then o/p = y
else o/p = z

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

$\Delta = \{x, y, z\}$

$$L = \left\{ \begin{matrix} 101 \\ 110 \end{matrix} , \begin{matrix} 0101 \\ 1101 \\ 0110 \\ 1110 \end{matrix} , \ldots \right\}$$



q_0 = SS / can see 0/z
q_1 = can see 1 / z
q_2 = can see 10 / z
q_3 = can see 101 / x
q_4 = can see 11 / z
q_5 = can see 110 / y

90	92	93
0 (1)	X00 (101)	X010 X011

94	95
110 111	X100 X101

1 1 0 1 0 1 1 0 Y

Z	Z	Y	X	Z	X	Z	(Y) ← o/p

* Minimization of Mealy Machine

	NS			
	i/p 0	o/p	i/p 1	o/p
q_0	q_0	z	q_1	
q_1	q_2	z	q_4	
q_2	q_0	z	q_3	
q_3	q_2	z	q_4	
q_4	q_5	y	q_4	
q_5	q_0	z	q_3	

Minimization of mealy machine is two step process :

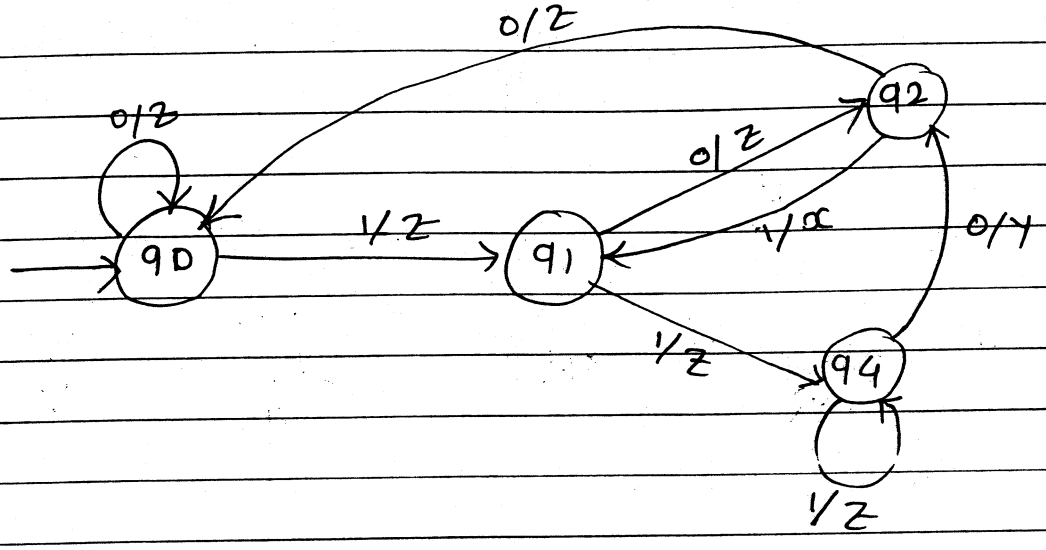
Step1 : Group the states according to similar output

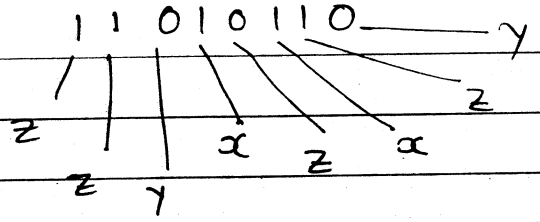
step2 : Regroup according to similar transition

step1 : [q_0, q_1, q_3] [q_2, q_5] [q_4]

step2 : [q_0] [q_1, q_3] [q_2, q_5] [q_4]

	NS			
	0	o/p	1	o/p
→ q_0	q_0	z	q_1	z
q_1	q_2	z	q_4	z
q_2	q_0	z	q_1	x
q_4	q_2	y	q_4	z





40) Design Moore machine that will read sequence made up of a, e, i, o, u and will give output as same character except when I is followed by E, it will produce U

Imp

$$\Sigma = \{A, E, I, O, U\}$$

$$\Delta = \{A, E, I, O, U\}$$

q_s = start state

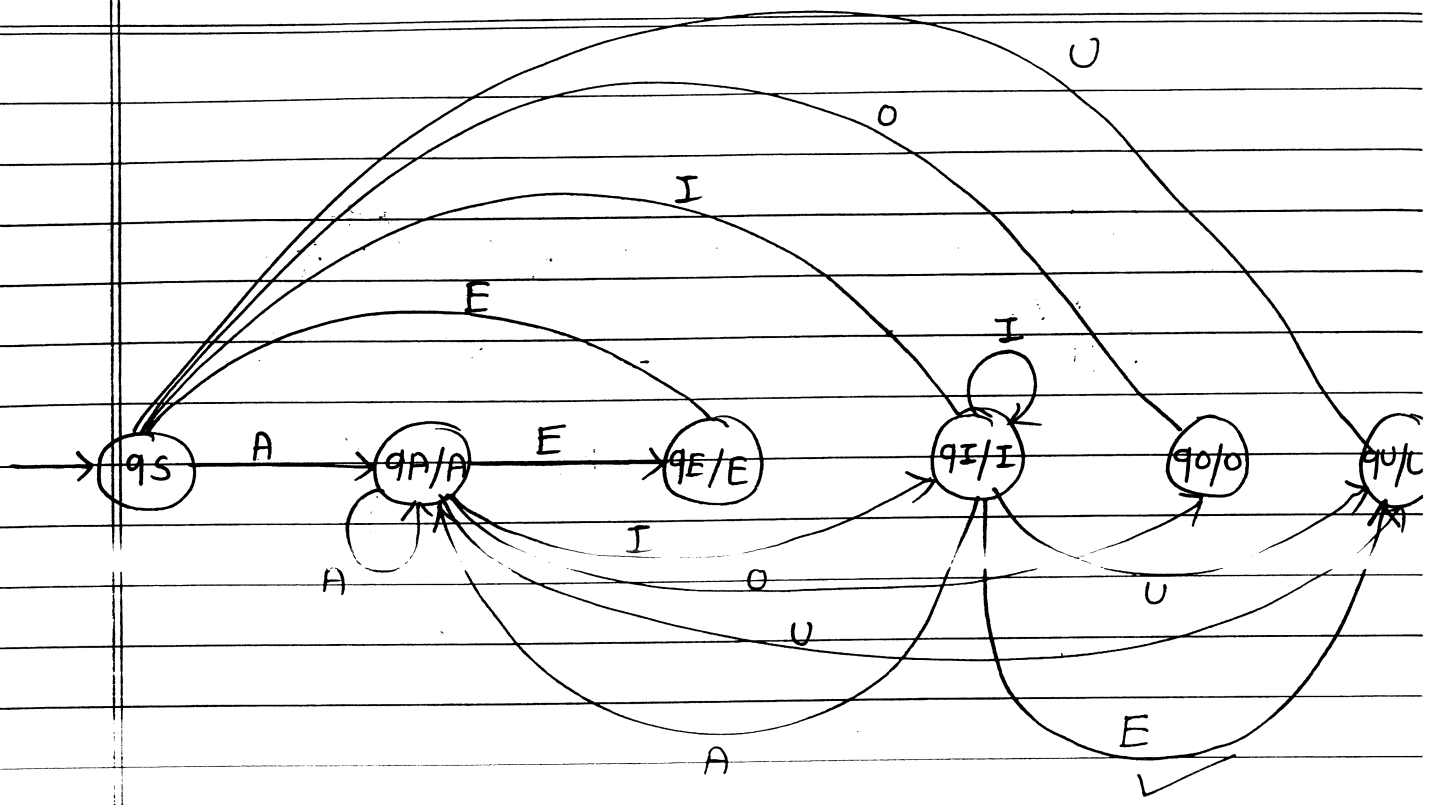
q_A = can see A / A

q_E = can see E / E

q_I = can see I / I

q_O = can see O / O

q_U = can see U / U



Similarly, we can do transitions for qE, qO, and qU.

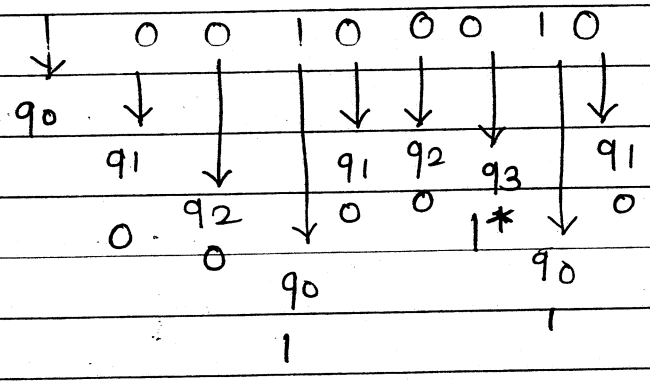
	A	E	I	O	U	O/P
qA	A	E	I	O	U	A
qE	A	E	I	O	U	E
qI	A	U	I	O	U	I
qO	A	E	I	O	U	O
qU	A	E	I	O	U	U

Recounting from 1 comes

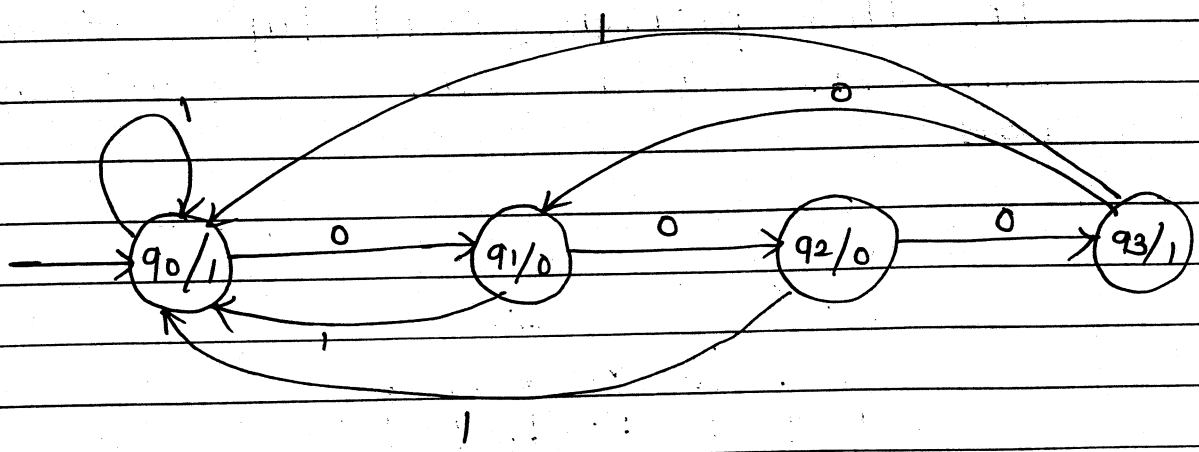
41) Design Moore machine to replace 000 by 001

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

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



q_0 = can see 0 no. of 0's / 1
q_1 = can see 1 no. of 0's / 0
q_2 = can see 2 no. of 0's / 0
q_3 = can see 3 no. of 0's / 1



q_0	q_1	q_2	q_3
0 1	(00) 01	(000) 001	(0000) 0001

42) Design Mealy machine to calculate 2's complement of a number and minimize the machine

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

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

* 1st method

1's complement

\+ 1

2's complement

0 1 1 0 0

1 0 0 1 1

\+ 1

1 0 1 0 0

* 2nd method

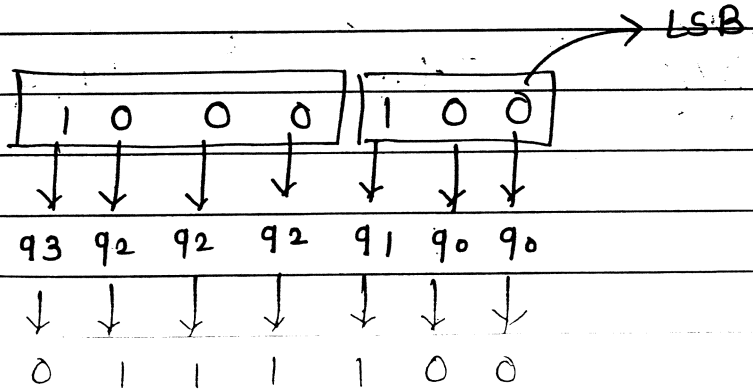
0 1 1 0 0

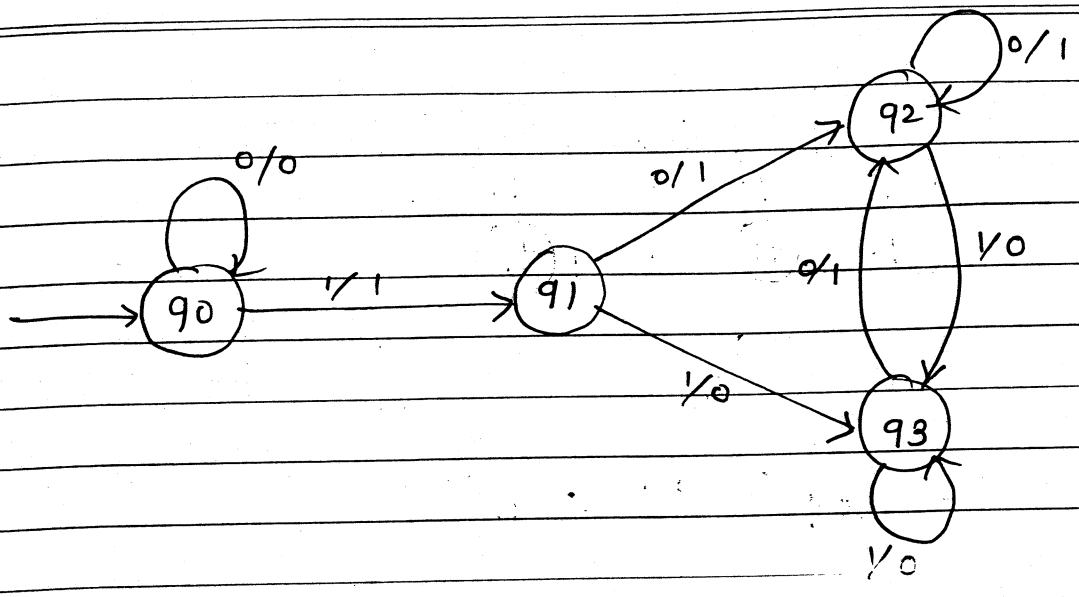
1 0 1

complem. = 0 0

as it until non zero comes

In this problem, read tape of machine will have input symbols starting from LSB





q_0 ⇒ can see zero before non zero ele. and produces output 0

q_1 ⇒ can see 1st non zero element and produces output 1

q_2 ⇒ can see 0 after non zero element and produces o/p 1

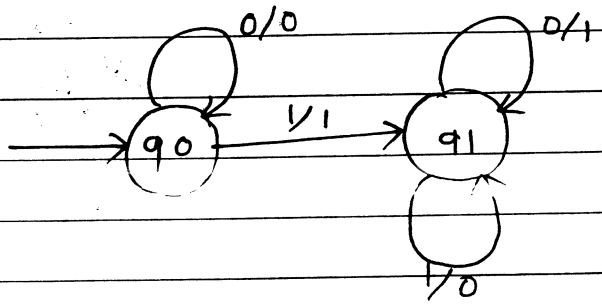
q_3 ⇒ can see 1 after non-zero element and produces o/p 0

	NS			
	0	o/p	1	o/p
q_0	q_0	0	q_1	1
q_1	q_2	1	q_3	0
q_2	q_2	1	q_3	0
q_3	q_2	1	q_3	0

step 1 : $[q_0]$ $[q_1, q_2, q_3]$

step 2 : $[q_0]$ $[q_1, q_2, q_3]$

	0	o/p	1	o/p
→ q_0	q_0	0	q_1	1
q_1	(q_1)	1	(q_1)	0

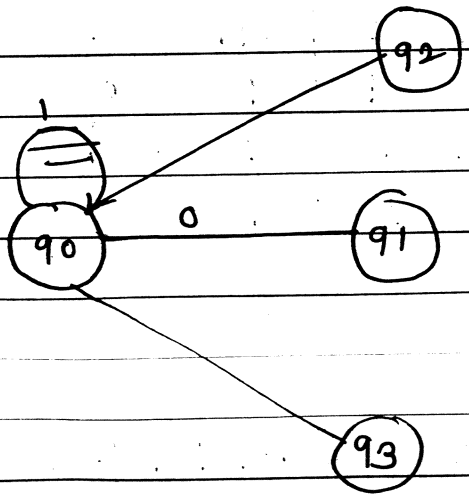


43) Design Moore Mealy machine for $\Sigma = \{0,1\}$. If binary number divisible by 3 then output 'Y' else output is 'N'.

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

$\Delta = \{Y, N\}$

$L = \{0, 11, 110 \ldots\ldots\}$



q_0 = start state / can see 1

	0	1
q_1	01	00

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

* Equivalence of Moore & Mealy M/c

1) Moore to Mealy Conversion

It is a three step process :

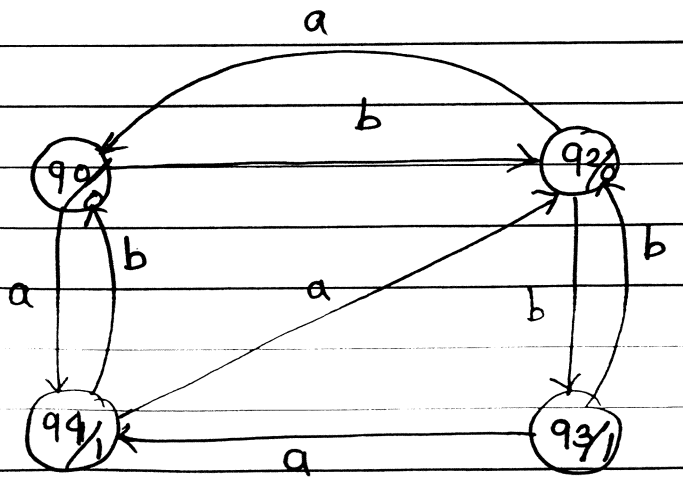
step 1 : Separate output column from moore table

step 2 : Rewrite Mealy table by attaching output in next state column.

step 3 : If possible minimize mealy machine.

44)

	a	b	o/p
→ q_0	q_1	q_2	0
q_1	q_2	q_0	1
q_2	q_0	q_3	0
q_3	q_1	q_2	1



step 1 :

q_0	0
q_1	1
q_2	0
q_3	1

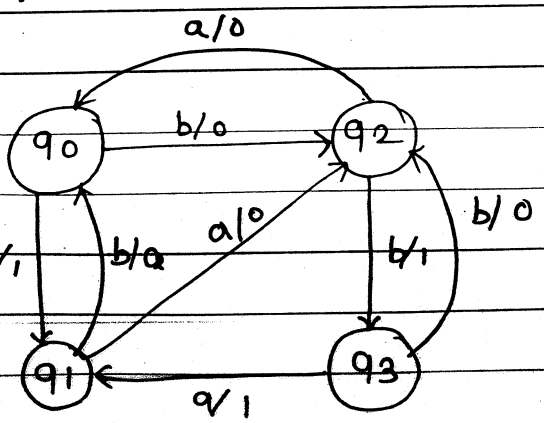
step 2 :

	NS			
	i/p a	o/p	i/p b	o/p
q_0	q_1	1	q_2	0
q_1	q_2	0	q_0	0
q_2	q_0	0	q_3	1
q_3	q_1	1	q_2	0

step 3 : $[q_0, q_3]$ $[q_1]$ $[q_2]$

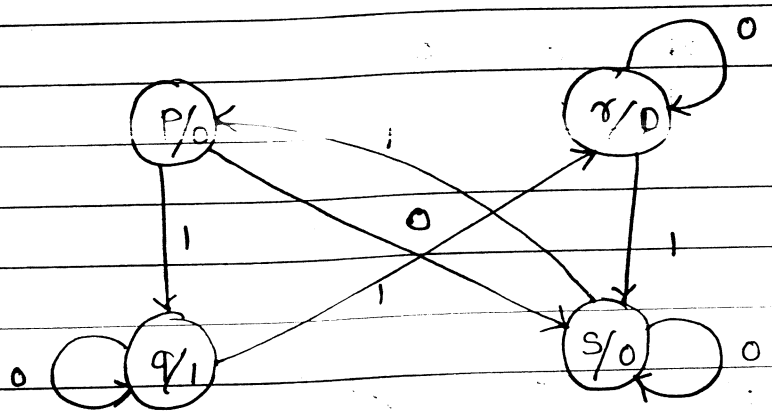
minimization
I) grouping similar o/p $[q_0, (q_3)]$ $[q_1]$ $[q_2$ → can be eliminated
II) Regrouping similar transitions

	NS			
	i/p	o/p	i/p	o/p
q_0	q_1	1	q_2	0
q_1	q_2	0	q_0	0
q_2	q_0	0	q_0	1



45)

	0	1	o/p
p	s	q	0
q	q	r	1
r	r	s	0
s	s	p	0



step 1

	o/p
p	0
q	1
r	0
s	0

step 2 :

	i/p 0	o/p	i/p 1	o/p
p	s	0	q	1
q	q	1	r	0
r	r	0	s	0
s	s	0	p	0

step 3 : [r, s] [p] [q]

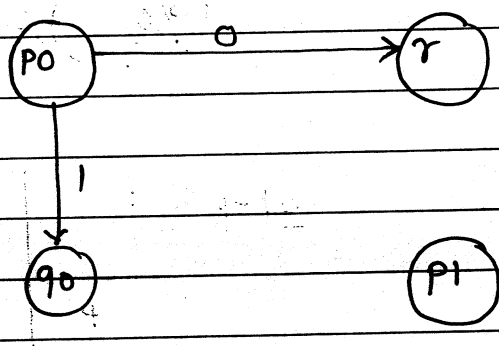
[r] [s] [p] [q] No minimization possible

	NS i/p	o/p	i/p	o/p
p	s	0	q	1
q	q	1	r	0
r	r	0	s	0
s	s	0	p	0

dia. after step2 Mealy
after step3 Minimize

46)

	0	1	o/p
p0	r	q0	ε
p1	r	q0	1
q0	p1	s0	0
q1	p1	s0	1
r	q1	p1	0
s0	s1	r	0
s1	s1	r	1



step 1

	o/p
p0	ε
p1	1
q0	0
q1	1
r	0
s0	0
s1	1

step 2

	NS			
	i/p 0	o/p	i/p 1	o/p
p0	r	0	q0	0
p1	r	0	q0	0
q0	p1	1	s0	0
q1	p1	1	s0	0
r	q1	1	p1	1
s0	s1	1	r	0
s1	s1	1	r	0

step3 : [P0, P1] [q0, q1, s0, s1] [r]

[P0, P1] [q0, q1] [s0, s1] [r]

p q . s r

p	r/0	q/0
q	r/1	s/0
r	q/1	p/1
s	s/1	r/0

H/W 47)

	a	b	o/p
q0	q2	q3	1
q1	q3	q2	0
q2	q1	q2	0
q3	q2	q1	1

2) Mealy to Moore conversion

step1 : Look into the next state column for all states and find states with different output.

step2 : Split state if it has different output and number of spilted states is equal to number of different outputs

step3 : Rearrange Mealy table with splited states

step4 : Write Moore table by removing output from next state column and attaching separate output column

optional → step5 : Response to ε

48)

	0	1
q_0	$q_1/1$	$q_0/0$
q_1	$q_0/1$	$q_2/1$
q_2	$q_2/0$	$q_2/0$

step1 : q_1 with o/p 1

q_0 with o/p 0

q_0 with o/p 1

q_2 with o/p 1

q_2 with o/p 0

classmate
Date
Page

step2 :

q_0 → q_{00} part of q_0 with o/p 0
q_0 → q_{01} part of q_0 with o/p 1

q_1 with o/p 1

q_2 → q_{20} part of q_2 with o/p 0
q_2 → q_{21} part of q_2 with o/p 1

step3 :

q_{00}	q_1/1	q_{00}/0
q_{01}	q_1/1	q_{00}/0
q_1	q_{01}/1	q_{21}/1
q_{20}	q_{20}/0	q_{20}/0
q_{21}	q_{20}/0	q_{20}/0

step4 :

	0	1	o/p → this o/p from step 2
q_{00}	q_1	q_{00}	0
q_{01}	q_1	q_{00}	ϕ
q_1	q_{01}	q_{21}	1
q_{20}	q_{20}	q_{20}	0
q_{21}	q_{20}	q_{20}	1

49)

	0	1
→ q1	q1/1	q2/0
q2	q4/1	q4/1
q3	q2/1	q3/1
q4	q3/0	q1/1

→ diagram is must for table

step 1 q1 with o/p 1

q2 with o/p 0

q2 with o/p 1

q3 with o/p 0

q3 with o/p 1

q4 with o/p 1

step 2 q1 with o/p 1

q2 < q20 part of q2 with o/p 0
q21 part of q2 with o/p 1

q3 < q30 part of q3 with o/p 0
q31 part of q3 with o/p 1

q4 with o/p 1

step 3

q1	q1/1	q20/0
q20	q4/1	q4/1
q21	q4/1	q4/1
q30	q21/1	q31/1
q31	q21/1	q31/1
q4	q20/0	q1/1

step 4

			O/P
91	91	920	1
920	94	94	0
921	94	94	1
930	921	931	0
931	921	931	1
94	930	91	1

diagram :

	0	1
→ p	r, 0	q, 0
q	p, 1	s, 0
r	q, 1	p, 1
s	s, 1	r, 0

step1 : r with o/p 0

p with o/p 1

q with o/p 1

q with o/p 0

s with o/p 0

s with o/p 1

step2 : p with o/p 1

q → q_1 part of q with o/p 1

q → q_0 part of q with o/p 0.

r with o/p 0

s → s_0 part of s with o/p 0

s → s_1 part of s with o/p 1

step3 :

p	r, 0	q_0, 0
q_0	p, 1	s_0, 0
q_1	p, 1	s_0, 0
r	q_1, 1	p_1, 1
s_0	s_1, 1	r, 0
s_1	s_1, 1	r, 0

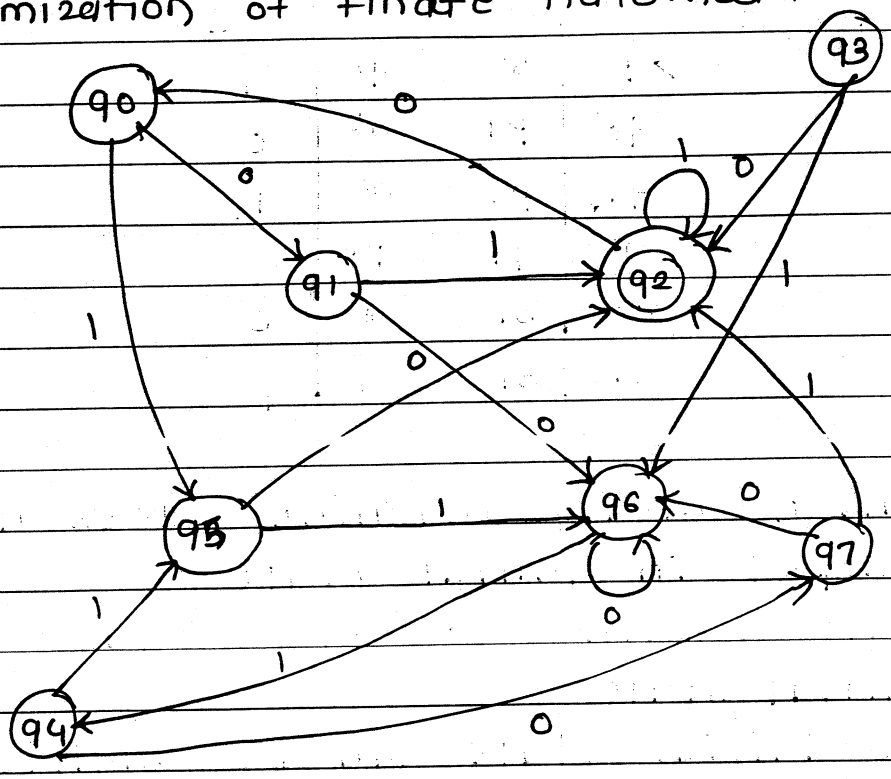
step 4 :

→ P	r	q_0	1
q_0	p_1	s_0	0
q_1	p_1	s_0	1
r	q_1	p_1	0
s_0	s_1	r	0
s_1	s_1	r	1

Given mealy machine does not produce output in initial state :
But resulting Moore produce 1 as o/p in initial state
So initial state can be splitted to respond ε

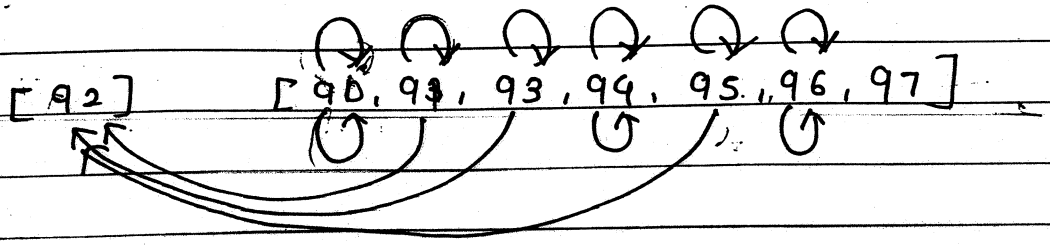
P_ε	r	q_0	ε
P_1	r	q_0	1
q_0	p_1	s_0	0
q_1	p_1	s_0	1
r	q_1	p_1	0
s_0	s_1	r	0
s_1	s_1	r	1

51) Minimization of finite Automata



	0	1
→ q0	q1	q5
q1	q6	q2
q2*	q0	q2
q3	q2	q6
q4	q7	q5
q5	q2	q6
q6	q6	q4
q7	q6	q2

step1 : Group according to final and non final



step2 : Regroup according to transition

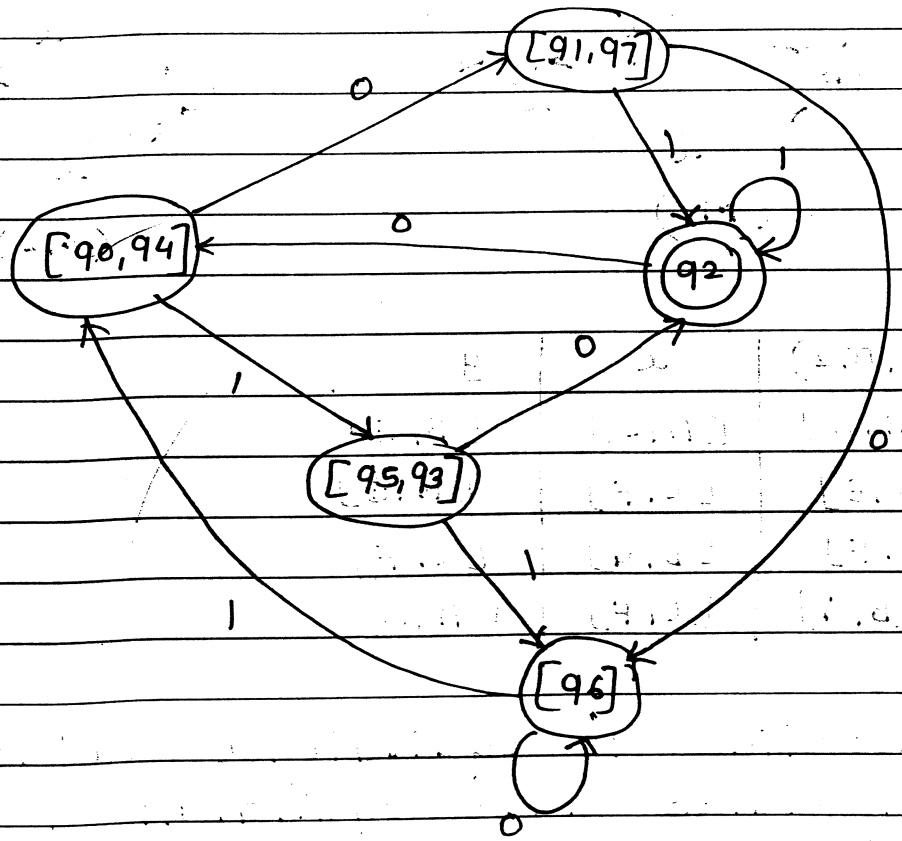
[92]

[90,94,96] [91,97] [93,95]

step3 : Regroup

[92] [90,94] [96] [91,97] [93,95]

---- minimized done

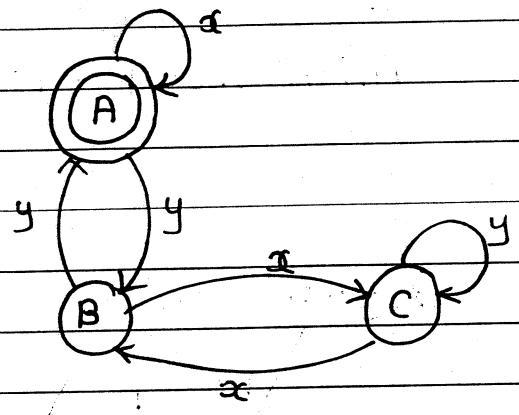


	0	1
→ [90,94]	[91,97]	[93,95]
[91,97]	[96]	[92]
[92]*	[90,94]	[92]
[93,95]	[92]	[96]
[96]	[96]	[90,94]

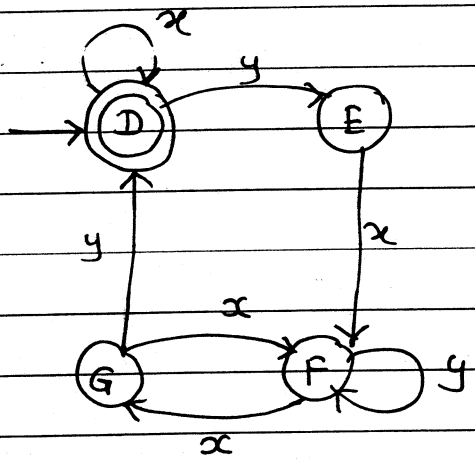
52) FA equivalence

Prove that following two finite automatas are equivalent

The technique used to solve this is known as equivalence methods.



(M1)



(M2)

(m1, m2)	x	y
→ [A, D]	[A, D]	[B, E]
[B, E]	[C, F]	[A, D]
[C, F]	[B, G]	[C, F]
[B, G]	[C, F]	[A, D]

According to equivalence method, all states have equivalent transitions generated

i.e. either final final or non final non final pairs, so, according to equivalence theorem these two finite automatas are equivalent.

2. Regular Expression / Language

Languages accepted by finite automata are known as regular language

L(FA) = Regular language

This languages can be expressed with set theory like notations known as regular expressions

Following notations are used to express the language

lowest ↓ Highest

+ union = choice
• concate = compulsion
* closure = generating all possibilities

FA	RE	
$(Q, \Sigma, \delta, Q_0, F)$	Σ	+
		•
		*

Defination of RE

Let Σ is a finite statet of input alphabets,

1. ϕ is RE
2. ε is RE
3. If $a \in \Sigma$ then a is RE
4. If R1 and R2 are two (sets) RES, then R1 + R2 , R1 • R2 is RE
5. If R1 is RE then (R1)* is RE

By applying above mentioned rules for finite number of times, whatever is formed is also RE.

a^* → represents ...

$a^0 = \varepsilon \quad a^1 = a$

$a^2 = aa$

English to RE

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

language consists of a or b

i.e. $L = \{a, b\}$

$R = a + b$

2) $\Sigma = \{a, b\}$ language consists of one word ab

$RE = a \cdot b$

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

language consists of 0 or more a's / any number of a's

$\Sigma = \{a\}$

$L = \{\varepsilon, a, aa, aaa, \dots\}$

$RE = a^*$

4) $\Sigma = \{a\}$

language consists of at least one a/ one or more a's

$\Sigma = \{a\}$

$L = \{a, aa, aaa, \dots\}$

$RE = a \cdot a^*$ (* = 0)

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

language consists of any combination of a or b

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

$$L = \{\varepsilon, a, b, aa, ab, ba, bb, \ldots\}$$

$$RE = (a+b)^*$$

6) $\Sigma = \{a, b, c\}$

language consists of all words which does not contain b

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

$$L = \{\varepsilon, a, c, aa, cc, ac, ca, \ldots\}$$

$$RE = (a+c)^*$$

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

language consists of non empty string

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

$$L = \{a, b, aa, ab, ba, bb, \ldots\}$$

$$(a+b)\cdot(a+b)^*$$

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

All words starting with 0

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

$L = \{0, 00, 01, 001, 010, 011, 010, \ldots\}$

$RE = 0 \cdot (0+1)^*$

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

All words ending with 1

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

$L = \{1, 01, 11, 011, 101, 111, 011, \ldots\}$

$RE = (0+1)^* \cdot 1$

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

All strings should begin with 1 and end with 0

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

$L = \{10, 100, 110, 1000, 1010, 1100, 1110, \ldots\}$

$RE = 1 \cdot (0+1)^* \cdot 0$

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

All strings which begin with 00 or 11 and ends with 00 or 11

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

$$L = \{00, 11, 00100, 00011, 11000, \ldots\}$$

$$(00 + 11)(0+1)^*(00+11)$$

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

Language consists of even number of a's

$$\Sigma = \{a\}$$

$$L = \{\varepsilon, aa, aaaa, \ldots\}$$

$$RE = (a \cdot a)^*$$

13) $\Sigma = \{a\}$

language consists of odd number of a's

$$\Sigma = \{a\}$$

$$L = \{a, aaa, aaaaa, \ldots\}$$

$$RE = a \cdot (aa)^*$$

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

All words with even length

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

$L = \{\varepsilon, aa, ba, ab, bb, aaaa, bbbb, abab, \dots\}$

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

15) All words with odd length

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

16) $\Sigma = \{a, b, c\}$

All strings with any number of a's followed by any number of b's followed by any number of c's

$RE = a^*\cdot b^*\cdot c^*$

17) $\Sigma = \{a, b, c\}$

All strings with at least one a followed by at least one b followed by at least one c

$RE = a\cdot a^*\cdot b\cdot b^*\cdot c\cdot c^*$

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

language consists of exactly two a's

$L = \{aa, aab$

$RE = b^*ab^*ab^*$

19) All words where third symbol from right end is 1 $\Sigma = \{0, 1\}$

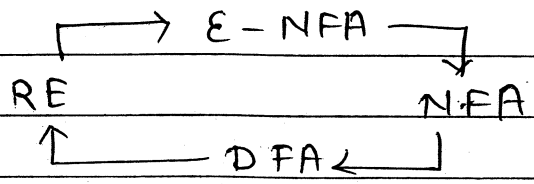
$(0+1)^* 1 (0+1)(0+1)$

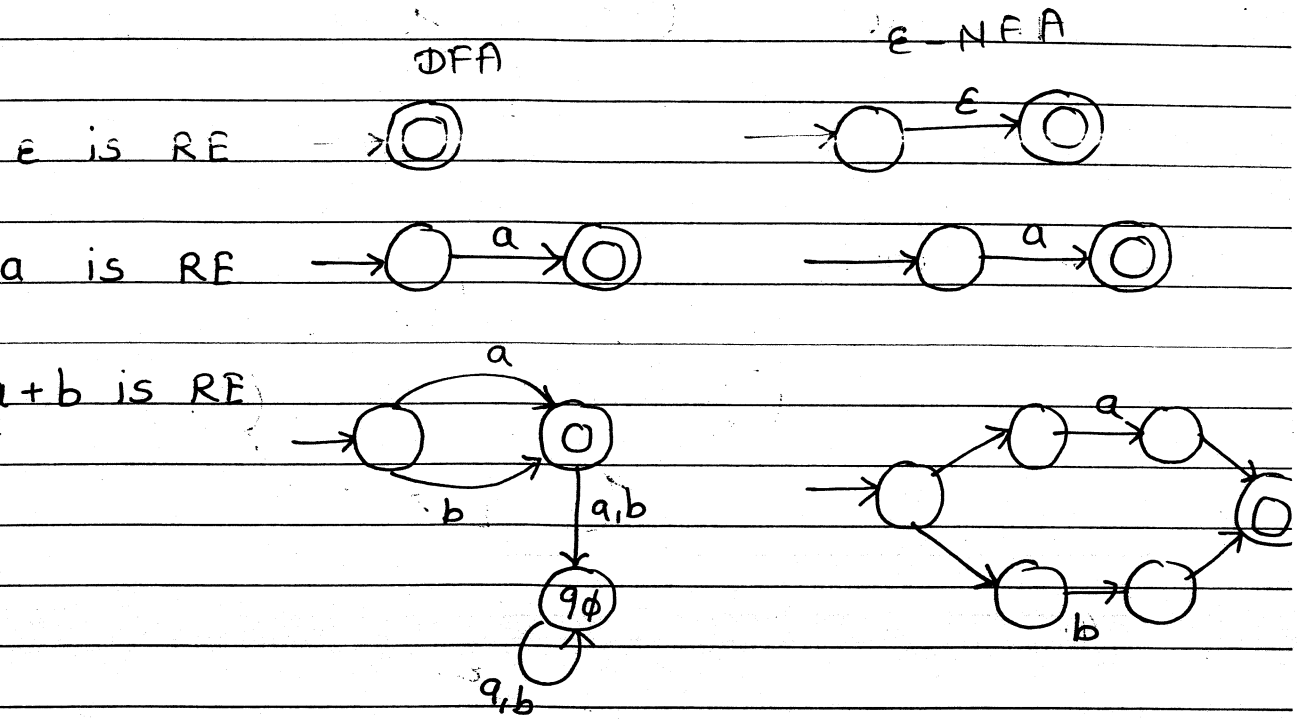
20) All words at least two consecutive 1's

$(0+1)^* 11 (0+1)^*$

21) $\Sigma = \{0, 1\}$ All words

RE and FA Equivalence

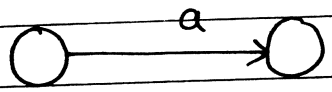




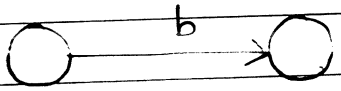
Convert following REs into equivalent NFA

1) (a + b)*

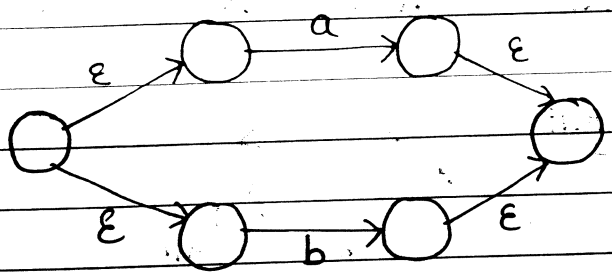
1. a



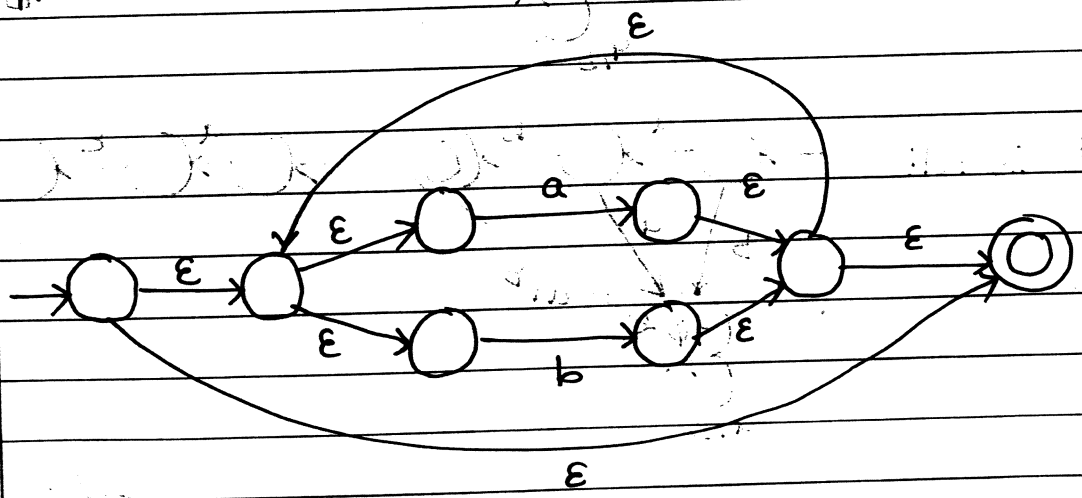
2. b



3: a + b

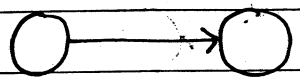


4.

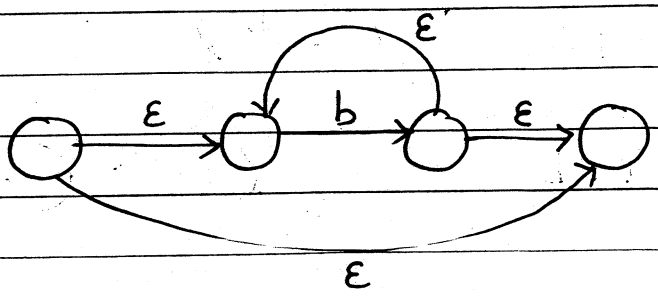


2) $(ab^* + b)$

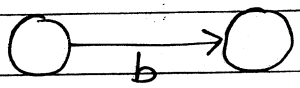
1. a



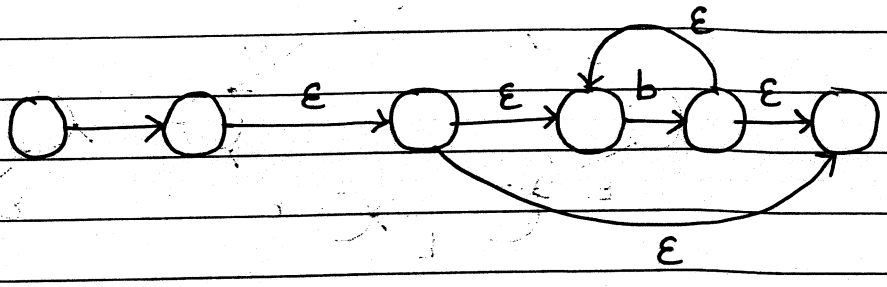
2. b^*



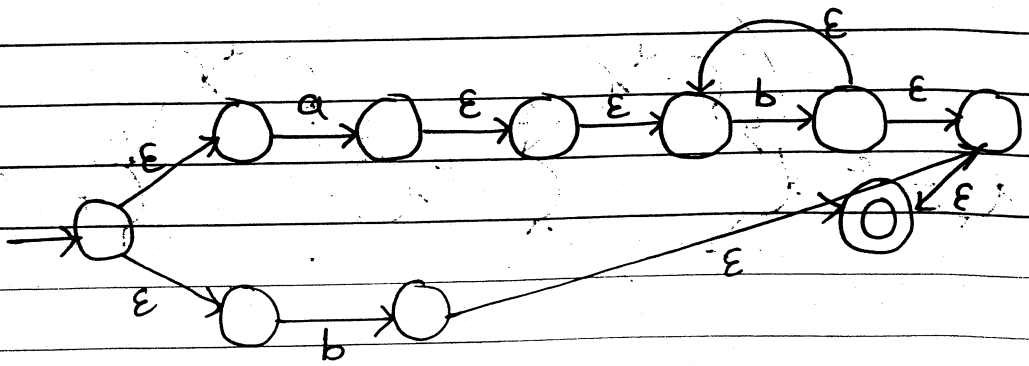
3. b



4. ab^*

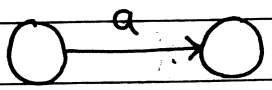


5. $(ab^* + b)$

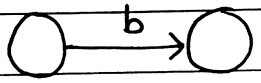


3] $[(a+b)(a+b)]^*$

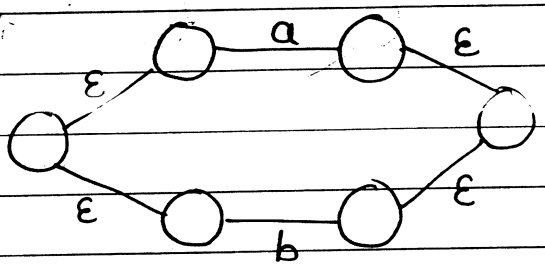
1. a



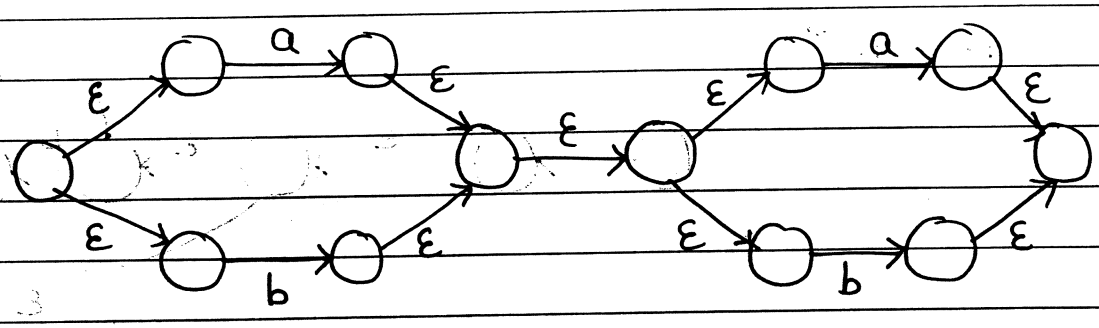
2. b



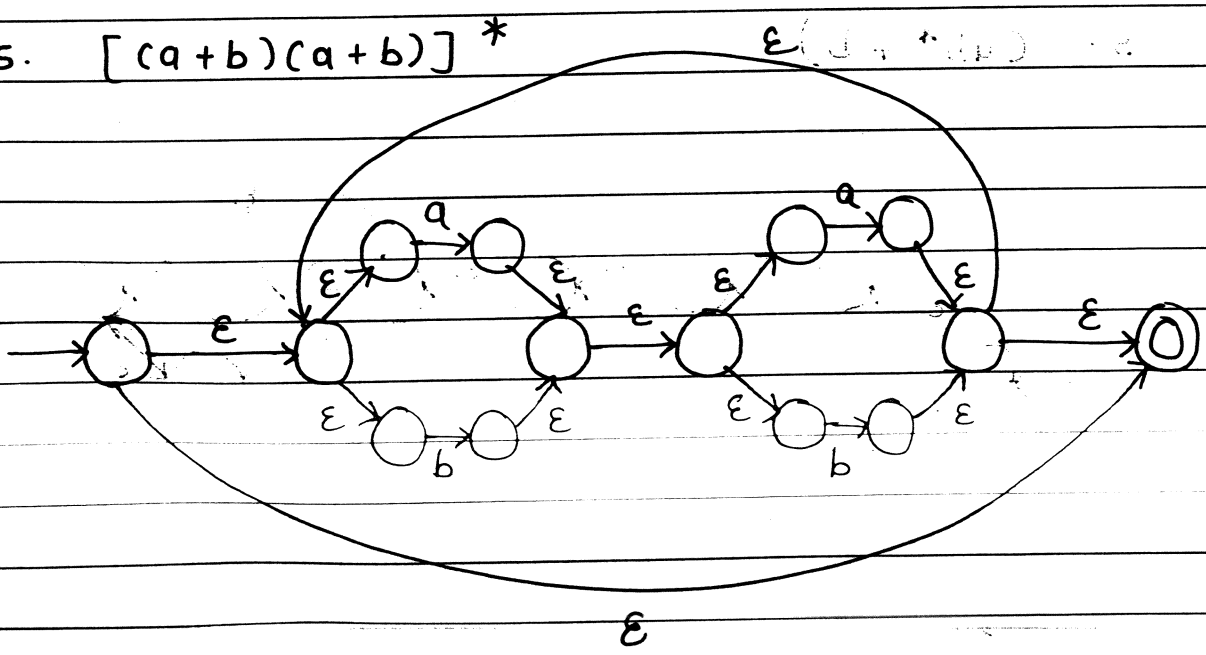
3. a + b



4. $(a+b)(a+b)$

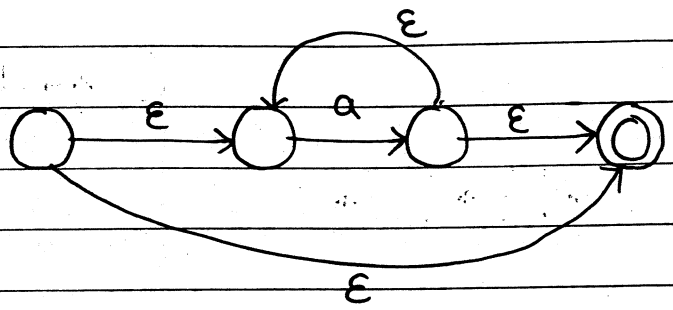


5. $[(a+b)(a+b)]^*$

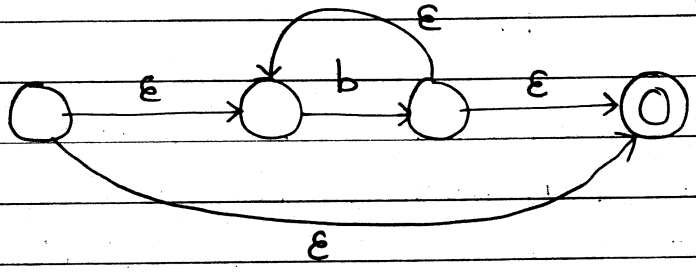


4) $[a^* + b^*]$

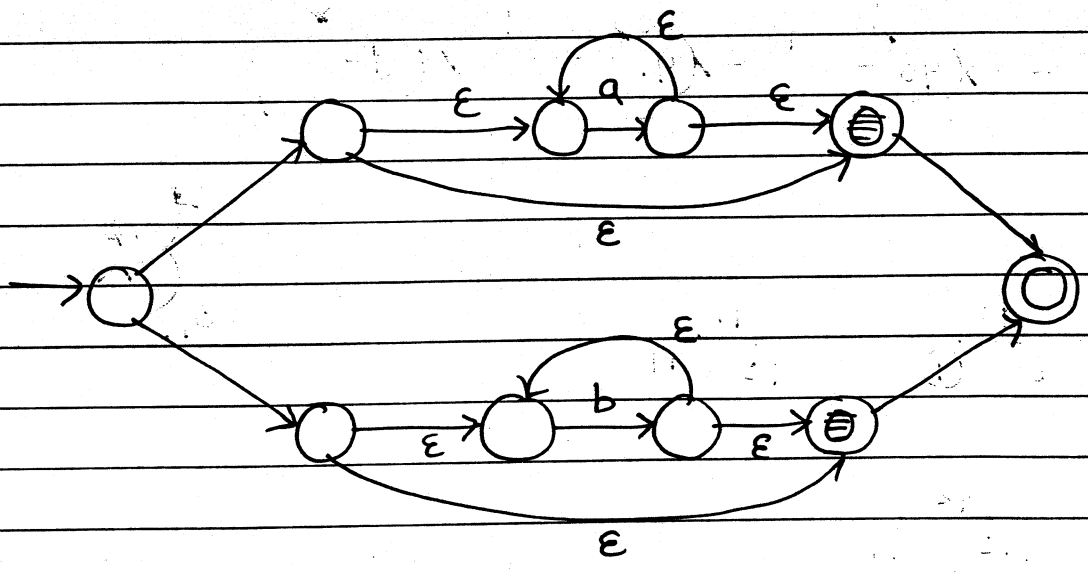
1. a^*



2. b^*



3. $[a^* + b^*]$



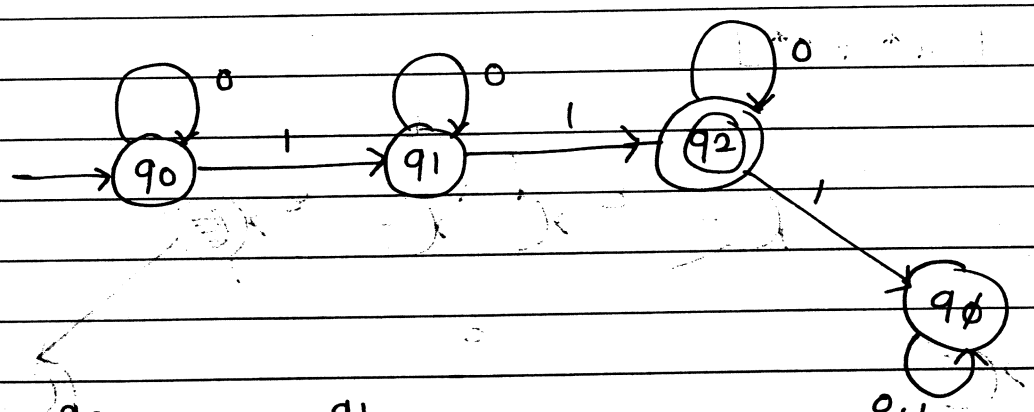
Convert following RES into equivalent DFA

1) $0^*10^*10^*$ (exactly)

$0^{*=0}10^{*=0}10^{*=0}$

$L = \{11, 011, 101, 110, \ldots\}$

q_0 = can see 0 no. of 1's
q_1 = can see 1 no. of 1's
q_2 = can see 2 no. of 1's



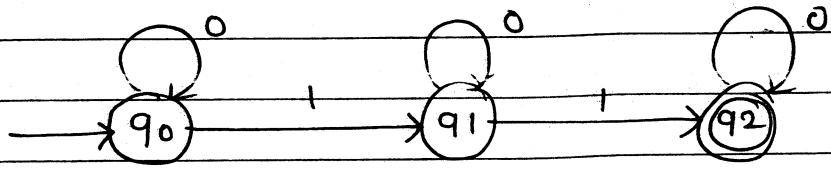
q_0	q_1	
0	10	11

q_2	
110	11

2) $0^*10^*1(0+1)^*$ (at least)

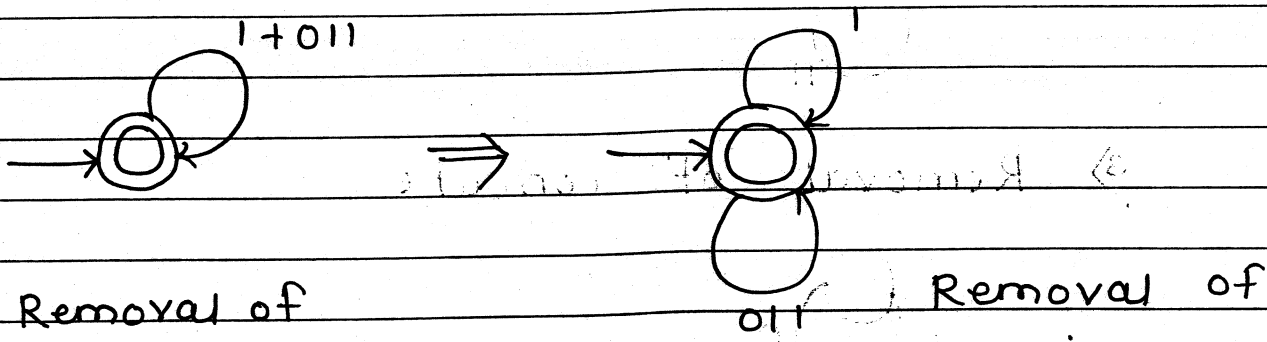
$0^*10^*1(0+1)^* = 0$

$L = \{ 11, 0110, \cdots \}$



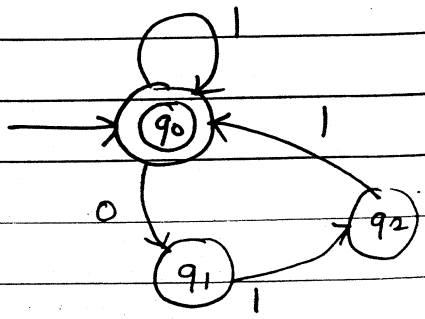
3) $(1+011)^*$

$L = \{ \varepsilon, 1, 11, 011, \cdots \}$

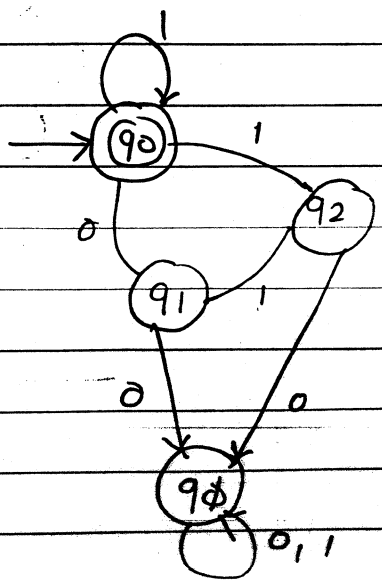


Removal of closure

Removal of union

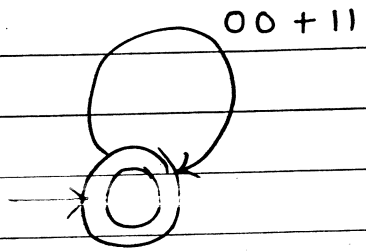


Removal of concate

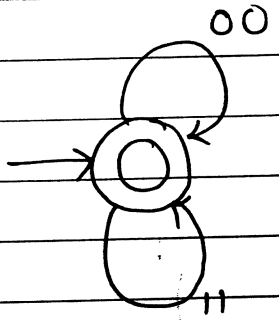


4) $(00+11)^*$

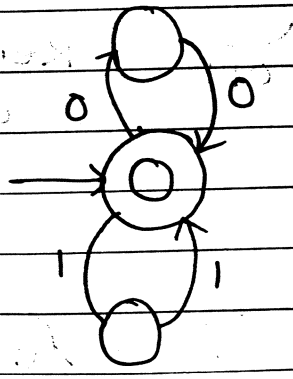
1) Removal of closure



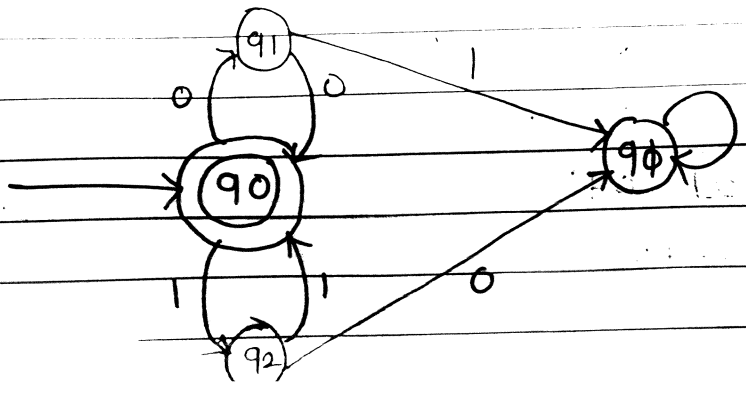
2) Removal of union



3) Removal of concate

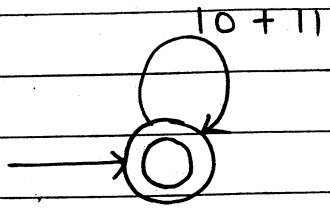


4)

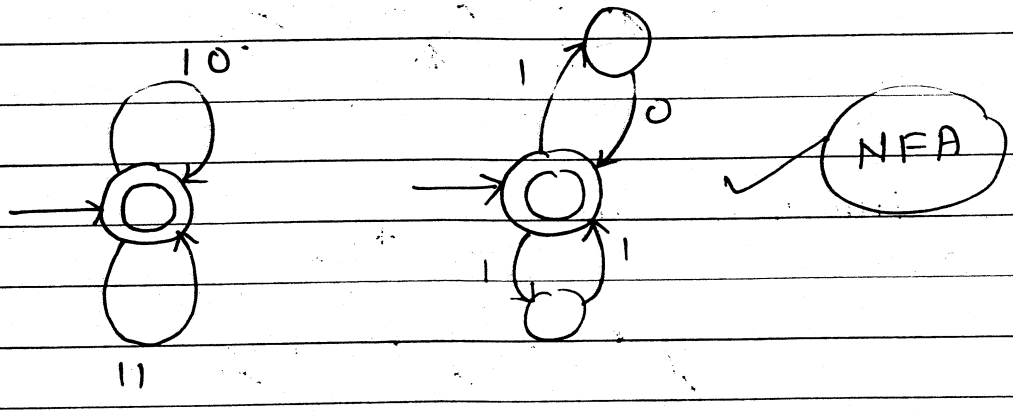


5) $(10 + 11)^*$

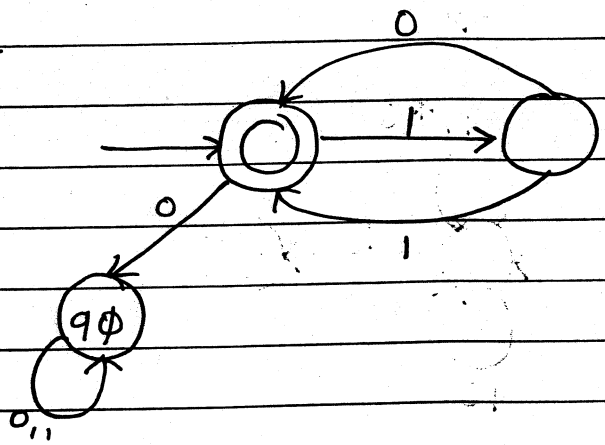
1) Removal of closure

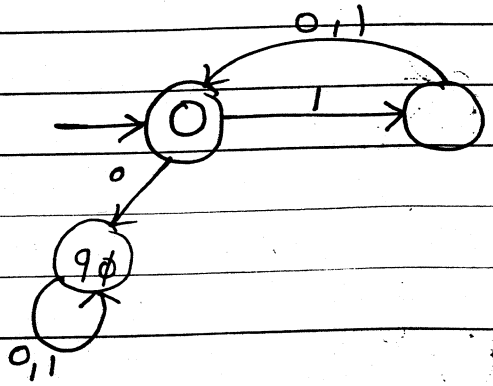


2) Removal of union



DFA

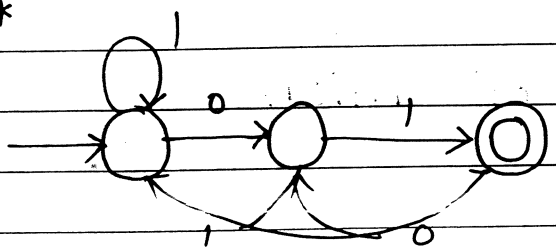




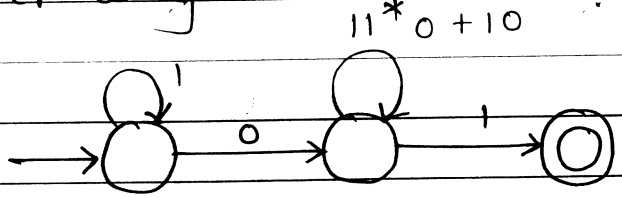
6) Design NFA for given RE

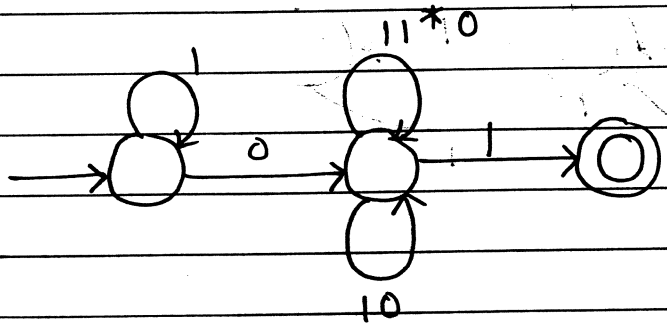
$1*0(11*0+10)*1$

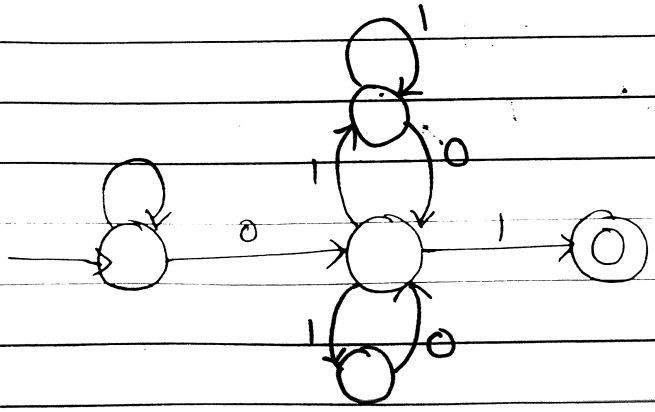
1. $1*$



Another way







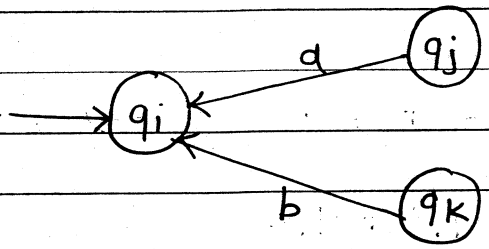
* FA to RE conversion (Arden's Theorem)

Let, P, Q and R are three REs on Σ then if P does not contain ε then all equations of the form

$$R = Q + RP$$

can be reduced to

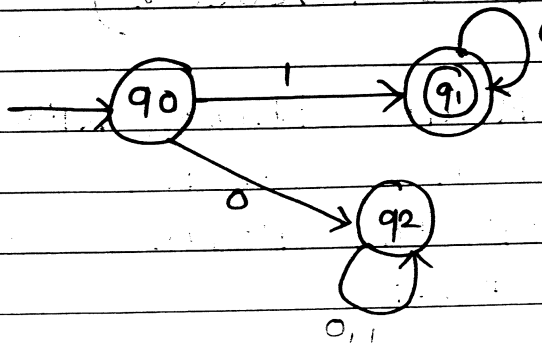
$$R = QP^*$$



$$\boxed{q_i = \varepsilon + q_j a + q_k b}$$

Mimp ? (2015)

Convert following FA into RE.



$q_0 = \varepsilon$ --- (1)

$q_1 = q_0 1 + q_1 0 + q_1 1$ --- (2) [Final]

$q_2 = q_0 0 + q_2 0 + q_2 1$ --- (3) [dead]

$q_1 = q_0 1 + q_1 0 + q_1 1$

putting $q_0 = \varepsilon$

$q_1 = \varepsilon 1 + q_1 0 + q_1 1$ anything with ε = anything

$= 1 + q_1 0 + q_1 1$

$= 1 + q_1 (0+1)$ → common

By Arden's theorm,

$R = Q + RP$

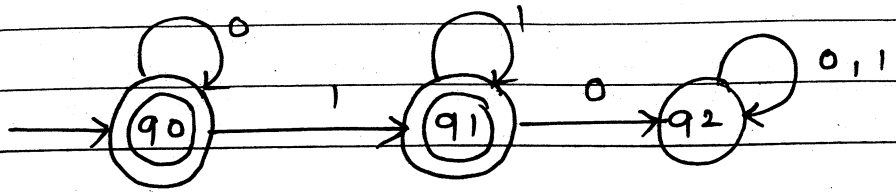
$R = QP^*$

$\therefore \underset{R}{q_1} = \underset{Q}{1} + \underset{R}{q_1} (\underset{P}{0+1})$

$q_1 = 1(0+1)^*$

$\boxed{1(0+1)^*}$

5.



$q_0 = \varepsilon + q_0 0$ ---(1)

$q_1 = q_0 1 + q_1 1$ ---(2) [dead]

$q_2 = q_1 0 + q_2 0 + q_2 1$ ---(3)

$q_0 = \varepsilon + q_0 0$

$R \quad = \quad Q \quad R \quad P \qquad R = QP^*$

$q_0 = \varepsilon \cdot 0^*$

$$\boxed{q_0 = 0^*}$$

$q_1 = q_0 1 + q_1 1$

$\underset{R}{q_1} = \underset{Q}{0^* 1} + \underset{R}{q_1} \underset{P}{1}$

$$\boxed{q_1 = 0^* 1 1^*}$$

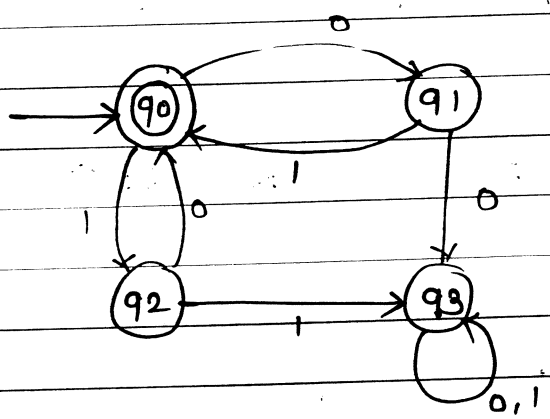
When there are multiple final states in given FA then resulting RE will be union of REs of both all final states

$$\boxed{RE = 0^* + 0^* 1 1^*}$$

classmate
Date
Page

Vvvimp

Prove that following FA accepts equal number of 0's and 1's such that each prefix has at most one more 0 than number of 1's or at most one more 1 than number of 0's



$q_0 = \varepsilon + q_2 0 + q_1 1$ → final

$q_2 = q_0 1$

$q_3 = q_0 0$

$q_3 = q_2 1 + q_1 0 + q_3 0 + q_3 1$ → dead

$$q_0 = \varepsilon + q_2 0 + q_1 1$$

$$q_0 = \varepsilon + q_0 10 + q_0 01$$

$$\underset{R}{q_0} = \underset{Q}{\varepsilon} + \underset{R}{q_0}\underline{\underline{(10+01)}}_{P}$$

$$q_0 = \varepsilon . (10+01)^*$$

$$\boxed{q_0 = (10+01)^*}$$

So, the given FA accepts equal number of 0's and 1's

Prefix state means the state prior to final state

So, q_1 and q_2 are prefix states

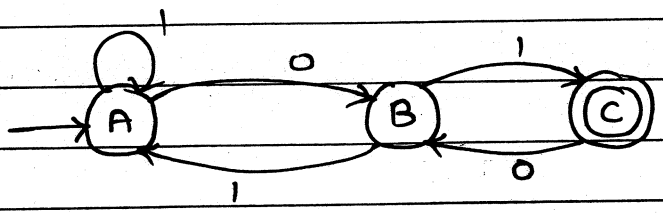
$$q_1 = q_0 0$$
$$= (10+01)^*.0$$

So, q_1 prefix has at most one more 0 than number of 1's

$$q_2 = q_0 1$$
$$= (10+01)^*.1$$

So, q_2 prefix has at most one more 1 than number of 0's

8.



$A = \varepsilon + B1 + A1$ ---(1)

$B = A0 + C0$ ---(2)

$C = B1$ ---(3) (final)

$$\underset{R}{A} = \underset{Q}{\varepsilon} + \underset{R}{B1} + \underset{P}{A1}$$

$$A = (\varepsilon + B1).1^*$$

$$A = \varepsilon.1^* + B11^*$$

$$\boxed{A = 1^* + B11^*}$$

$B = A0 + C0$

$B = (1*B|1^*)0 + B10$

$B = 1^*0 + B|1^*0 + B10$

$B = \underbrace{1^*0}_{R} + \underbrace{B}_{Q}\underbrace{(11^*0 + 10)}_{P}$

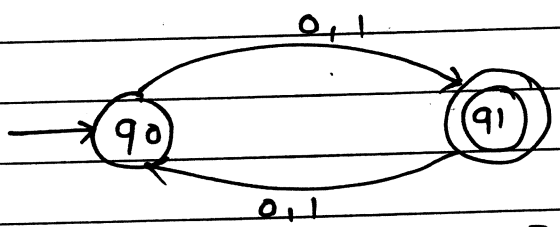
$$\boxed{B = 1^*0\,(11^*0 + 10)^*}$$

$C = B1$

$$\boxed{C = 1^*0\,(11^*0 + 10)^*\cdot 1}$$

H/W

Arden's theorm



Ans → $[(0+1)][(0+1)\cdot(0+1)]^*$

classmate
Date
Page

* RE identities AND Equivalence

Just similar to set theorem, boolean algebra, logic calculus, RE also has some identities which can be used to minimize regular expression and to show that two REs are equivalent.

Identities

1. $\phi + R = R + \phi = R$

2. $\phi \cdot R = R \cdot \phi = \phi$

3. $\varepsilon \cdot R = R \cdot \varepsilon = R$

Idempotent (how it behaves with itself)

4. $R + R = R$ (we have choice)

Commutation (interchange)

5. $R + P = P + R$ $\quad R \cdot P = P \cdot R$ ✗
as there is compulsion of following

Distribution

6. $P(Q + R) = PQ + PR$

7. $(P + R)Q = PQ + RQ$

Association : (can only happen with similar entities)

8. $P + (Q + R) = (P + Q) + R$

9. $P(Q \cdot R) = (P \cdot Q) \cdot R$

Laws of closure

10. $\phi^* = \varepsilon$

11. $\varepsilon^* = \varepsilon$

12. $R^* \cdot R^* = R^*$

13. $\varepsilon + R \cdot R^* = R \cdot R^* + \varepsilon = R^*$

14. $(R^*)^* = R^*$

15. $R \cdot R^* = R^* \cdot R = R^+$

16. $(P + Q)^* = (P^* + Q^*)^* = (P^* \cdot Q^*)^*$

17. $(P^* \cdot Q)^* = \varepsilon + (P + Q)^* \cdot Q$

18. $(P \cdot Q)^* \cdot P = P \cdot (Q \cdot P)^*$

1. $R + R = R$

$LHS = R + R$

$= R[\varepsilon + \varepsilon]$

$= R.\varepsilon$

$= R$

$= RHS$

2. $(PQ)^*P = P(QP)^*$

$LHS = (PQ)^*P$

$= (PQ)^{*=1}P$

$= \{\varepsilon, PQ, PQPQ, PQPQPQ, \ldots\}P$

$= \{\varepsilon.P, PPQP, PQPQP, \ldots\}$

$= P\{\varepsilon, QP, QPQP, \ldots\}$

$= P.(Q.P)^*$

$= RHS$

3. $R^*R^* = R^*$

$LHS = R^*R^*$

$= \{\varepsilon, R, RR, RRR, \ldots\}.\{\varepsilon, R, RR, RRR, \ldots\}$

$= \{\varepsilon, R, RR, RRR, \ldots\}$

$= R^*$

$= RHS$

4. $(P+Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$

$(P+Q)^* = \{\varepsilon, P, Q, PP, PQ, QP, \dots\}$

$(P^* + Q^*)^* = [\{\varepsilon, P, PP, \dots\} + \{\varepsilon, P, PP, \dots\}]^*$

$= \{\varepsilon, P, Q, PP, PQ, QP, \dots\}$

$(P^* . Q^*)^* = [\{\varepsilon, P, PP \dots\} . \{\varepsilon . Q, QQ, \dots\}]^*$

$= \{\varepsilon, P, Q, PP, \dots$
PQ
QP
$QQ\}$

5. Prove that following two REs are equal

$P = (1+011)^*$

$q = \varepsilon + 1^*(011)^*(1^*(011)^*)^*$

$q = \varepsilon + 1^*(011)^*(1^*(011)^*)^*$

Assume $A = 1^*(011)^*$

$q = \varepsilon + A \cdot A^*$

By using identity $\varepsilon + R \cdot R^* = R^*$

$q = A^*$

$q = (1^* . (011)^*)^*$

By using identity $(P^*.Q^*)^* = (P+Q)^*$

$q = (1+0p_1)^*$

$q = P$

Hence proved

6. $P = 0^*1(0+10^*1)^*$

$Q = (1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1)$

$= (1+00^*1)\{E + (0+10^*1)^*(0+10^*1)\}$

Assume $0+10^*1 = A$

$= (1+00^*1)\{E + A^*A\}$

$= (1+00^*1)(E+A^*)$ → By identity $R.R^* = R^*$

$= (1+00^*1)(E+0+10^*1)$

$= (1+00^*1)(0+10^*1)$

$= (E+00^*)1(0+10^*1)$

$R.R^*$

$= 0^*1(0+10^*1)^*$

$= P$

classmate

Date

Page

$$Q = \underbrace{(1+00*1)}_{A} + \underbrace{(1+00*1)}_{A}\underbrace{(0+10*1)^*}_{B}\underbrace{(0+10*1)}_{B}$$

$$Q = A + AB^*B$$

$$Q = A(\varepsilon + B^*B)$$

$$\downarrow$$

$$\varepsilon + R^*R = R^*$$

$$Q = A \cdot B^*$$

$$Q = (1+00*1) \cdot (0+10*1)^*$$

$$= (\varepsilon +$$

classmate
Date
Page

* Pumping Lemma → Rule
Generating string

languages accepted by FA are known as Regular languages and can be expressed by RE

There are some languages which are not regular in nature and can not be accepted by FA.

Pumping lemma provides necessary conditions for a language to become regular language by pumping some strings into it.

This can be used to prove that language is not regular if it does not follow necessary condition

It uses proof by contradiction

statement of pumping lemma

Let M is a finite automata with n no. of states $M = (Q, \Sigma, \delta, q_0, F)$

$|Q| = n$

which accepts regular language L

Let w is a word, $w \in L$

where

$|w| > n$

then w can be splitted as $w = xyz$

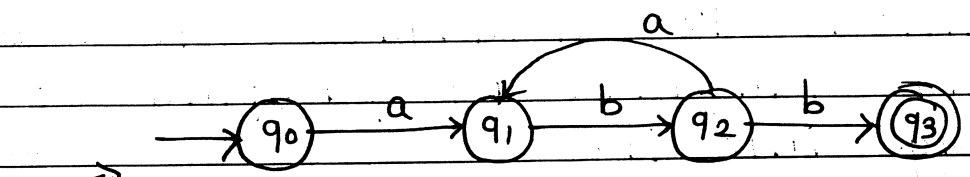
where

i) $|y| > 0$

ii) $|xy| < n$

iii) $xy^i z \in L$ if y value is pumped $i \geq 0$ times

Understanding of pumping lemma



$\delta \Rightarrow$

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

$|Q| = n = 4$

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

$q_0 = q_0$

$F = q_3$

$L = \{\overset{3}{abb}, \overset{5}{ababb}, \overset{7}{ababbabb}, \dots\}$

$w = \underset{x}{a}\,\underset{y}{ba}\,\underset{z}{bb}$

i) $|y| > 0$

ii) $|xy| < n$

iii) $i = 0$ $\quad \underset{x}{a}\,\underset{z}{bb} \in L$

$i = 1$ $\quad \underset{x}{a}\,\underset{y}{ba}\,\underset{z}{bb} \in L$

$i = 2$ $\quad \underset{x}{a}\,\underset{y}{ba}\,\underset{y}{ba}\,\underset{z}{bb} \in L$

Prove that following language is not regular

① $L = \{ a^n b^n \mid n \geq 0 \}$

L = Number of a's followed by equal no. of b

$L = \{ \varepsilon, ab, aabb, aaabbb, ----- \}$

step1 : Assume that given language L is a regular language and accepted by FA with n states

step2 : Let $w = aaabbb \in L$

where $|w| > n$ ← no. of states of FA

This word can be splitted into x, y z components in following three cases

case1 : y value consists of only a's

$$w = \underset{x}{a}\underset{y}{a}\underset{z}{abbb}$$

where i) $|y| > 0$

ii) $|xy| < n$

iii) $xy^i z \in L$ for all $i \geq 0$

so,

After pumping y value for $i \geq 0$ times, we get

$i = 0 \quad \underset{x}{a}\underset{z}{bbb} \notin L$

$i = 1 \quad \underset{x}{a}\ \underset{y}{aa}\ \underset{z}{bbb} \in L$

$i = 2 \quad \underset{x}{a}\ \underset{y}{aa}\ \underset{y}{aa}\ \underset{z}{bbb} \notin L$

So, in this case number of a's are more than b's after pumping y value

case2 y value consists of only b's

$w = \underset{x}{a}\ \underset{y}{bb}\ \underset{z}{bbb} \qquad \underset{x}{aaa}\ \underset{y}{bb}\ \underset{z}{b}$

where i) $|y| > 0$

ii) $|xy| < n$

iii) $xy^iz \in L$ for all $i \geq 0$

So, after pumping y value for $i \geq 0$ times we get

$i = 0 \quad \underset{x}{aaa}\ \underset{z}{b} \notin L$

$i = 1 \quad \underset{x}{aaa}\ \underset{y}{bb}\ \underset{z}{b} \in L$

$i = 2 \quad \underset{x}{aaa}\ \underset{y}{bb}\ \underset{y}{bb}\ \underset{z}{b} \notin L$

case3 y value consists of a and b both

$w = \underset{x}{aa}\ \underset{y}{ab}\ \underset{z}{bb}$

where i) $|y| > 0$

ii) $|xy| < n$

iii) $xy^iz \in L$ For all $i \geq 0$

So, after pumping y value for $i \geq 0$ time we get,

$i = 0$ $\underset{x}{a}\underset{z}{abb} \in L$

$i = 1$ $\underset{x}{a}\underset{y}{aa}\underset{z}{bbb} \in L$

$i = 2$ $\underset{x}{aa}\underset{y}{ab}\underset{y}{ab}\underset{z}{bb} \in L$

Generated words are not following language nature

step3 : So, in all above 3 cases after pumping y value generated words does not belong to language

Hence, our assumption in step ① that language is regular is not true

Therefore According to Contradiction our assumption was wrong.

So, language is not regular.

(2) $L = \{ a^p \mid \text{where } p \text{ is prime} \}$

$L = \{ aa, aaa, aaaaa, \text{-----} \}$

$\underset{x}{a}\underset{y}{aaa}\underset{z}{a}$

$i = 0 \quad \underset{x}{a}\underset{y}{aa} \in L$

$i = 1 \quad a\ aa\ aa \in L$

$i = 2 \quad a\ aa\ aa\ aa \in L$

$i = 3 \quad a\ aa\ aa\ aa\ aa \notin L$

(3) $L = \{ \omega \omega^R \mid \omega \in (a,b)^* \}$

even palindrome → $\omega = \{ \varepsilon, a, b, aa, ab, ba, bb, \text{---} \}$

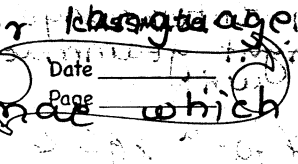
$L = \{ \varepsilon, aa, bb, aaaa, abba, baab, bbbb, \text{---} \}$

(4) $L = \{ \omega\omega \mid \omega \in (a,b)^* \}$

$\underset{\omega}{abb}\underset{\omega}{bbabb} \notin L$

FA Regular language

language is accepted by FA called as regular language & can be expressed with regular expr.



It can also be represented by type 3 grammar which is known as regular grammar

so it's always possible to convert FA into Regular Grammar & vice versa

left linear Grammar

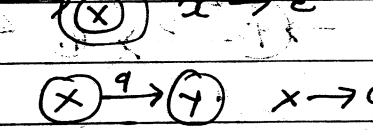
$X \to aX$
$X \to Xa$
$X \to q$
$X \to \varepsilon$

LLG: $X \to Xa$, $X \to a$, $X \to \varepsilon$

RLG: $X \to aX$, $X \to q$, $X \to \varepsilon$

FA - RLG
RLG - FA
FA - LLG
LLG - FA
RLG - LLG
LLG - RLG

* FA to RLG Rules



$x \to \varepsilon$

$x \to aY$

$x \to qY$

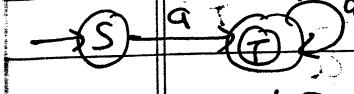
(1)

FA $(Q, \Sigma, \delta, q_0, F) \Rightarrow (V_N, \Sigma, P, S)$

Rename $q_0 = S$

$q_1 = T$

$q.(q+b)^*$

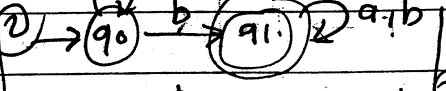


$S \to qT \mid q$

$T \to qT \mid q$

$T \to bT \mid b$

(2)



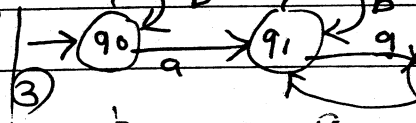
$S \to aS \mid a$

$S \to bS \mid bb$

$T \to aT \mid a$

$T \to bT \mid b$

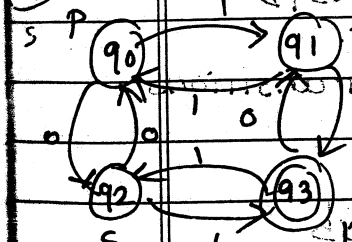
(3)



$S \to bS$
$S \to aT$
$T \to bT$
$T \to aR \mid a$
$T \to bT$

$R \to bR \mid b$
$R \to aT$

(4)



$P \to 1Q$
$Q \to 1P$
$P \to 0S$
$S \to 0P$

$Q \to 0R \mid 0$
$R \to 0Q$
$S \to 1R \mid 1$
$R \to 1S$

RLG → FA

(1)

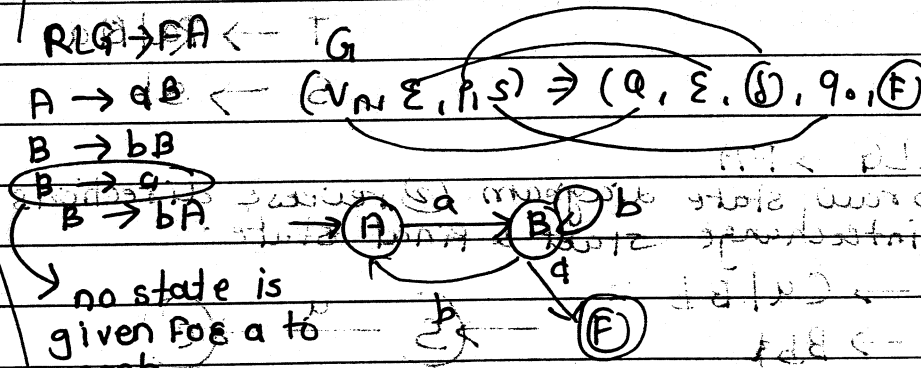
$A \to qB$
$B \to bB$
$B \to q$
$B \to bA$

no state is given for a to reach

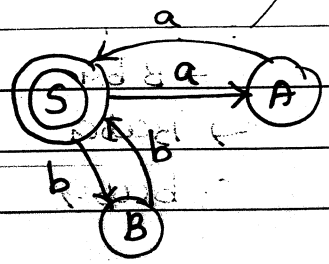
(2)

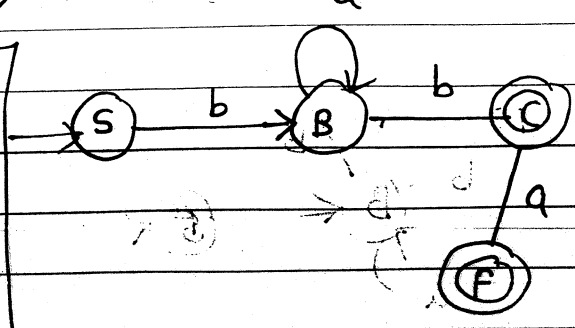
$S \to bB$
$B \to bC \mid aB \mid b$
$C \to a$

G $(V_N, \Sigma, P, S) \Rightarrow (Q, \Sigma, \delta, q_0, F)$?



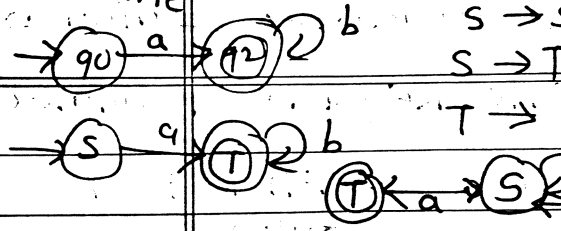
$S \to aA \mid \varepsilon$
$S \to bB$
$A \to aS \mid a$
$B \to bS \mid b$





FA - LLG

Step 1 Interchange start & final state ② Reverse direction of all transitions ③ write right grammar in LLG form.



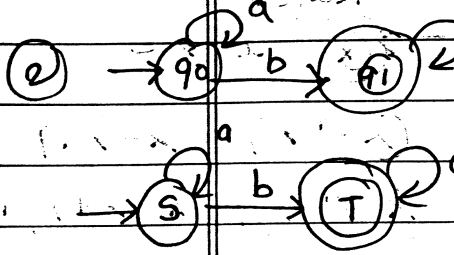
$S \to Sb$
$S \to Ta \mid a$
$T \to \varepsilon$

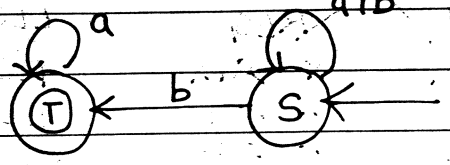
$S \to a$
$S \to Sb$
$S \to ab$

$S \to Sb$
$S \to Sbb$
$S \to abb$

$S \to Ta$
$\to \varepsilon . a$
$\to a$

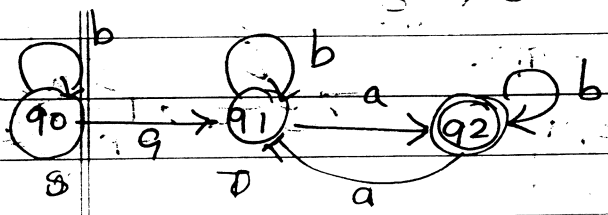
$S \to Sb$
$\to Tab$
$\to \varepsilon . ab$

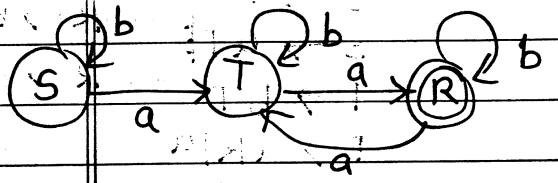
② 

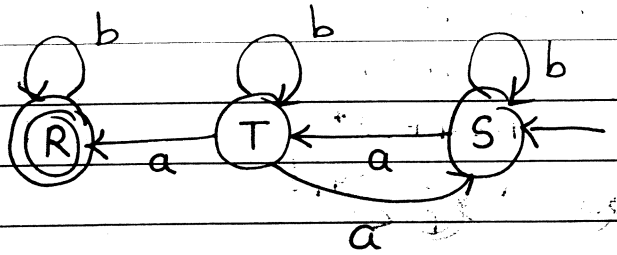


$T \to Ta \mid a$
$S \to Sa$

$S \to Sb$
$S \to Tb \mid b$

③ 





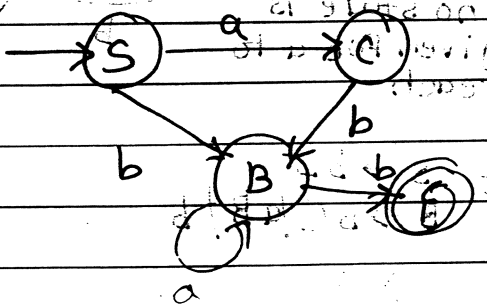
$R \to b$
$T \to Tb$
$T \to Ra \mid Baa$
$S \to Sb$

$S \to Ta$
$T \to Sa$
$R \to Rb \mid b$

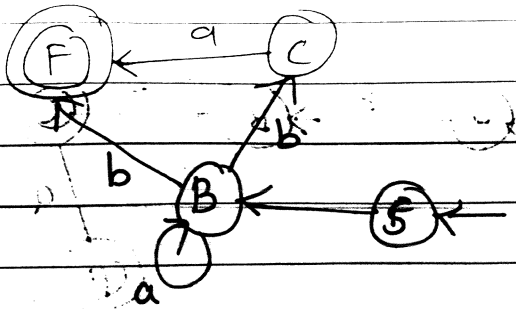
LLG → FA

① Draw state diagram ② reverse directions of all transitions ③ interchange start & final state.

$S \to Ca \mid Bb$
$C \to Bb$
$B \to Ba \mid b$



$S \to Bb$
$\to bb$
$S \to Ca$
$\to Bbq$
$\to bba$

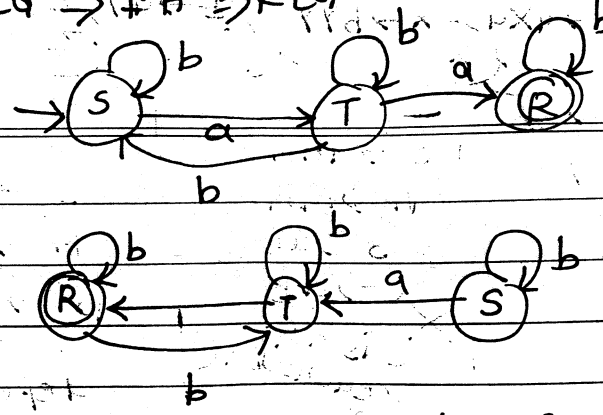


$S \to Ca$
$\to Bba$
$\to Baaba$
$= baabq$

Convert RLG into LLG.

RLG → S → Sb
S → Tq
T → Tb
T → Sb
T → Ra | a
R → Rb | b

LLG ⇒ RA ⇒ RLG



S → bS
S → qT
T → bT
T → qR | q
R → bR | b
R → qT

Grammar plays a vital role in syntax & symantic analysis phase of compiler. So, to incorporate grammar into compiler, the grammar must be 1) simplified 2) normalized 3) ambiguity free

① simplification of grammar - Grammar can be simplified by using following three steps: ① removal of useless symbols ② removal of ε production ③ removal of unit productions

(*) Removal of useless symbols

← non generating — Non reachable

The symbol which can not be terminated by terminals either by directly or indirectly

The symbol which can not be reached from start symbol

1) S → AB | SS | a A → aA B → bB
A → aA
B → bB

NT = { S, A, B }

S → a ✓ A & B are non generating

S → AB
→ aAB

③ S → A
A → aA | b

{ S, A } ✓ ✓

② S → Aq | Bb | a | b
A → Aa | a
B → bB

B is non generating

S → Aq | a | b
A → Aq | q

④ S → AB | CA
B → BC | AB
A → a
C → aB | b

B non gener

S A ⑤ C

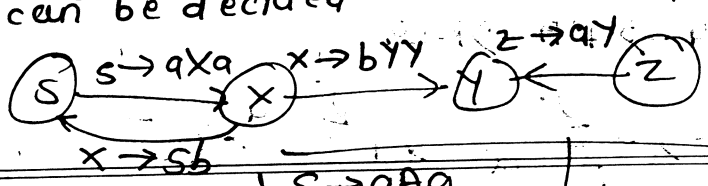
B → Bb
Bb

⑤ S → aAq
A → Sb | bCC
C → abb
C → qC

S A C

Non reachability can be decided with the help of dependancy graph

$S \to aXa$
$X \to Sb \mid bYY$
$Y \to abb$
$Z \to aY$



S, X, Y

$S \to aXa$
$X \to Sb \mid bYY$
$Y \to abb$

$S \to aAa$
$A \to bBB$
$B \to ab$
$C \to aB$

C is not reachable

$S \to aS \mid AB$
$A \to bA$
$B \to AA$

all are reachable

dependancy graph is must to exam.

(2) Removal of ε production

ε production means presence of null move & because of it compiler may transit from one state to another state without receiving any symbol. so, the presence of ε in a grammar is hampering / hazardous for compiler development.

the symbol with ε production is known nullable symbols

$S \to AB$
$A \to aA \mid \varepsilon$
$B \to bB \mid \varepsilon$
$A \to \varepsilon$, $B \to \varepsilon$ } nullable (direct)

$S \to AB$
$S \to \varepsilon$ → nullable (indirect)

if language consists of ε.

If $\varepsilon \in L$ then ε can be removed from entire grammar except start symb ($S \to \varepsilon$)

If $\varepsilon \notin L$ then ε can be removed from entire grammar

(2) $S \to aX$
$X \to aX \mid bX \mid \varepsilon$

$X \to \varepsilon$ | X is nullable (direct)
S is not nullable

As $\varepsilon \notin L$, ε can be removed from entire grammar

So, after replacing nullable symbo effect grammar becomes

$S \to aX$	$X \to aX$	$X \to bX$
$S \to aX$ (a.ε)	$X \to aX$ (a.ε)	$X \to bX$
$S \to a$	$X \to a$	$X \to b$

$S \to aX \mid a$
$X \to aX \mid a \mid b$

(3) $S \to aXb$
$X \to aXb \mid \varepsilon$

$X \to \varepsilon \Rightarrow X$ is nullable

$S \to aXb$
S is not nullable
$\varepsilon \notin L$ so ε can be removed from entire grammar

$S \to aXb$	$X \to aXb$
$S \to aXb$	$X \to aXb$
$S \to ab$	$X \to ab$

Grammar
$S \to aXb \mid ab$
$X \to aXb \mid ab$

GNF ONE NT → one terminal followed by zero or more non terminal.

$S \to a$ ✓

$S \to aAbX$

$S \to aAB$ ✓

$S \to aBCD$ ✓

$S \to aaBCX$

1. $S \to aSb$
   $S \to ab$

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

Step1 For every terminal symbol, introduce new NT

Terminals = $\{a, b\}$

$A \to a$

$B \to b$

$S \to ASB$

$S \to AB$

$A \to a$
$B \to b$ | GNF

step2 Introduce an order among NT by renaming

$S = A1$

$A = A2$

$B = A3$

$A1 \to A2A1A3$

$A1 \to A2A3$

$A2 \to a$
$A3 \to b$ } GNF

step3 lemma1 and lemma2 can be applied if rules are not in GNF form

$A_i \rightarrow A_j\gamma$ where $(j > i)$

$A_1 \rightarrow A_2 A_1 A_3$ $(j > i)$

$A_i \quad A_j \; \gamma$

$A_1 \rightarrow A_2 A_3$ $(j > i)$

$A_i \quad A_j \; \gamma$

$A_2 \rightarrow a$

$A_3 \rightarrow b$

step4 Reverse substitution

$A_3 \rightarrow b$

$A_2 \rightarrow a$

$A_1 \rightarrow A_2 A_3$

a

$A_1 \rightarrow a A_1 A_3$

step5 As lemmas are not applied, z rules are not generated.

So, the grammar in GNF form,

$A_3 \rightarrow b$

$A_2 \rightarrow a$

$A_1 \rightarrow aA_3$

$A_1 \rightarrow aA_1A_3$

lemma 2 remove left recursion

$A \rightarrow Aq$

$A \rightarrow \beta i$ $\quad Z \rightarrow \alpha i$

$A \rightarrow \beta i Z$ $\quad Z \rightarrow \alpha i Z$

e.g. $A \rightarrow Aa$

$A \rightarrow b$ $\quad Z \rightarrow a$

$A \rightarrow bZ$ $\quad Z \rightarrow aZ$

step4

$A_1 \rightarrow A_2A_3$

$A_2 \rightarrow A_3A_1$

$A_2 \rightarrow A_4$

$A_3 \rightarrow A_5 \mid A_4A_3A_2 \mid A_5Z \mid A_4A_3A_2Z$

$A_4 \rightarrow b$

$A_5 \rightarrow a$

$Z \rightarrow A_1A_3A_2 \mid A_1A_3A_2Z$

steps Reverse substitution.

$A_5 \rightarrow a$

$A_4 \rightarrow b$

$A_3 \rightarrow a \mid bA_3A_2 \mid aZ \mid bA_3A_2Z$

$A_2 \rightarrow b \mid bA_3A_2A_1 \mid aZA_1 \mid bA_3A_2ZA_1 \mid aA_1$

$A_1 \rightarrow bA_3 \mid bA_3A_2A_1A_3 \mid aZA_1A_3 \mid bA_3A_2ZA_1A_3 \mid aA_1A_3$

$Z \rightarrow$ as substitute

lemma1 which are not in the form of

$A_i \rightarrow A_j\gamma$ where $j > i$

e.g. $\underset{A_i}{A_3} \rightarrow \underset{A_j}{A_1}\ \underset{\gamma}{A_2A_3}$

all b productions $A_1 \rightarrow A_4A_3A_2$

$A_1 \rightarrow A_4A_1A_2$

$A_3 \rightarrow A_4A_3A_2A_2A_3$ $(j > i)$

$A_3 \rightarrow A_4A_1A_2A_2A_3$ $(j > 1$

$Z \rightarrow B$

$Z \rightarrow BZ$

e.g (2) $S \to SS$
$S \to aSb$
$S \to ab$

step1 : $A \to a$
$B \to b$

$S \to SS$
$S \to ASB$
$S \to AB$
$A \to a$
$B \to b$ } GNF

step2 : $S = A1$
$A = A2$
$B = A3$

$A1 \to A1A1$
$A1 \to A2A1A3$
$A1 \to A2A3$
$A2 \to a$
$A3 \to b$ } GNF

step3 :

$A1 \to A1A1$ → left recursion (lemma 2)

$\underline{A1} \to \underline{A2}\ \underline{A1}\ \underline{A3}$
$Ai \quad Aj \quad \gamma$ $\quad Ai \to Aj\gamma \ (j>i)$

$\underline{A1} \to \underline{A2}\ \underline{A3}$ $\quad Ai \to Aj\gamma \ (j>i)$
$Ai \quad Aj \quad \gamma$

$A2 \to a$
$A3 \to b$ } GNF

lemma2 application.

$\underline{A1} \to \underline{A1}\ \underline{A1}$ $\quad [\gamma = 1]$
$A \quad A \quad \gamma$

$A1 \to A2A1A3$ $\quad S[S=2]$
$A1 \to A2A3$

$A1 \to A2A1A3$ ✓ $\quad A1 \to A2A3$
$A1 \to A2A1A3Z$ $\quad A1 \to A2A3Z$

$Z \to A1$
$Z \to A1Z$

After step3 grammar is

$A1 \to A2A1A3 \mid A2A1A3Z \mid A2A3 \mid A2A3Z$
$A2 \to a$
$A3 \to b$
$Z \to A1 \mid A1Z$

step4 Reverse substitution $\quad A3 \to b$
$A2 \to a$
$A1 \to aA1A3 \mid aA1A3Z \mid aA3 \mid aA3Z$
$Z \to aA1A3 \mid aA1A3Z \mid aA3 \mid aA3Z$
$Z \to aA1A3Z \mid aA1A3ZZ \mid aA3Z \mid aA3Z$

e.g. $S \to AB$
$A \to BS \mid b$
$B \to SA \mid a$

step1

$X \to b$
$Y \to a$

$S \to AB$
$A \to BS$
$A \to X$
$B \to SA$
$B \to Y$
$X \to b$ } GNF
$Y \to a$ }

step2

$S = A1$
$A = A2$
$B = A3$
$X = A4$
$Y = A5$

step3

$A1 \to A2A3$ — in other cases $Ai \to Aj\gamma$, $j > i$

$A2 \to A3A1$

$A2 \to A4$

$\boxed{A3 \to A1A2}$ — lemma1 can be applied

$A3 \to A5$

$A4 \to b$ } GNF
$A5 \to a$ }

Applying lemma1

$\underline{A3} \to \underline{A1}A2$
$A \quad\quad B$

$\underline{A1} \to \underline{A2}A3$
$B \quad\quad \beta$

$\underline{A3} \to \underline{A2A3}A2$ Again lemma1 to be apply
$A \quad\quad \beta$

$\underline{A2} \to \underline{A3A1}$
$\beta \quad\quad \beta 1$

$A2 \to A4$ $\boxed{A3 \to A3A1A3A2}$ lemma2 to be apply

$A3 \to A4A3A2$ $Ai \to Aj\gamma$ $(j > i)$

$\underline{A3} \to \underline{A3}A1A3A2$ $\gamma = 1$
$A \quad\quad A$

$A3 \to AS$ } $S = 2$
$A3 \to A4A3A2$ }

$A \to \beta$ $A3 \to AS$ $A3 \to A4A3A2$
$A \to \beta Z$ $A3 \to AsZ$ $A3 \to A4A3A2Z$

$Z = A1A3A2$
$Z = A1A3A2Z$