Automata theory:

is a theoretical branch of Computer Science and Mathematics, which mainly deals with the logic of computation with respect to simple machines referred to as automata

Automata enables the scientists to understand how machines compute the functions and solve problems.

vation: To develop methods to describe and analyse the dynamic behaviour of discrete systems

Ex. Alphabets: $\Sigma = \{0, 1\}$ is an alphabet of binary digits
$\Sigma = \{0, 1, \ldots 9\}$ is an alphabet of decimal digits
$\Sigma = \{a, b, c\}$
$\Sigma = \{A, B, C, \ldots Z\}$

→ $\Sigma = \{a, b\}$

No. of strings of length 2 that can be generated over the alphabet $\{a, b\}$

aa, ab, ba, bb

Length of string $|w| = 2$

No. of string = 4

Conclusion:-

For alphabet $\{a, b\}$ with length n, number of strings can be generate = 2^n

Language: A language is a set of strings chose form some Σ^*

OR

A language is a subset of Σ^*.

A language which can be formed over Σ can be finite or infinite.

Ex L_1 = { Set of all strings of length 2 }
= {aa, ab, ba, bb..} Finite lang

L_2 = { Set of all strings which starts with 'a' }
= {a, aa, ab, aba, abb, aaa, abb....} Infinite

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

Σ^0 = Set of all strings over Σ of length 0. $\{\epsilon\}$

Σ^1 = Set of all strings over Σ of length 1. $\{a, b\}$

Σ^2 = Set of all strings over Σ of length 2. {aa, ab, ba, bb}

$|\Sigma^2| = 4$ and $|\Sigma^3| = 8$

Inductive proofs:

The Principle of mathematical Induction:-
Suppose we have some statement P(n) and we want to demonstrate that P(n) is true for all $n \in N$.

If we provide proofs for P(0), P(1) ... P(k) where k is some large number ...

In order to show that $\forall n$, P(n) holds, it suffices to establish the following two properties

a) Base case : show that P(0) holds
b) Induction step: If P(n-1) holds / implies, show that P(n) holds.

Eg. No. 4
Ex. 1.1 Prove that $\sum_{i=0}^{n-1} i^2 = \frac{(n-1)\,n\,(2n-1)}{6}$ — (1)

$$\sum_{i=0}^{n} i^2 = \frac{n(n+1)\,(2(n+1)-1)}{6}$$

$$\sum_{i=0}^{n} i^2 = \frac{n(n+1)\,(2n+1)}{6}$$

Since $$\sum_{i=0}^{n} i^2 = \sum_{i=0}^{n-1} i^2 + n^2$$

we are given

$$\sum_{i=0}^{n-1} i^2 = \frac{(n-1)\,n\,(2n-1)}{6}$$

$$\therefore \frac{(n-1)\,n\,(2n-1)}{6} + n^2 = \frac{n(n+1)(2n+1)}{6}$$

Date
Page

www.the mathpage.

Ex. 2 $1+2+3+\cdots+n = \frac{1}{2}n(n+1)$

for $n=0$ $P(0)=0$

$n=1$ $P(1)=1$

$n=2$ $P(2)=2$

$n=3$ $1+2+3 = \frac{1}{2}\,3.4^{2}$

$= 6$

$n=4$ $1+2+3+4 = \frac{1}{2}\,4.5$

$= 10$

$n=k$

$$1+2+\cdots+k = \frac{k(k+1)}{2}$$

$n=(k+1)$

$$1+2+3+\cdots+(k+1) = \frac{(k+1)(k+2)}{2} \quad \text{---(2)}$$

$$1+2+3\cdots k+(k+1) = k\cdot\frac{(k+1)}{2} + k+1$$

$$= \frac{k\cdot(k+1)+2(k+1)}{2}$$

$$= \frac{(k+1)(k+2)}{2}$$ taking $(k+1)$ as common

	DFA	NFA
1)	Deterministic FA	Non Deterministic FA
2)	$\delta : Q \times \Sigma \to Q$	$\delta : Q \times \Sigma \to 2^Q$
3)	Cannot use ϵ transition	Can use ϵ transition
4)	All DFA are NFA	Not all NFA are DFA

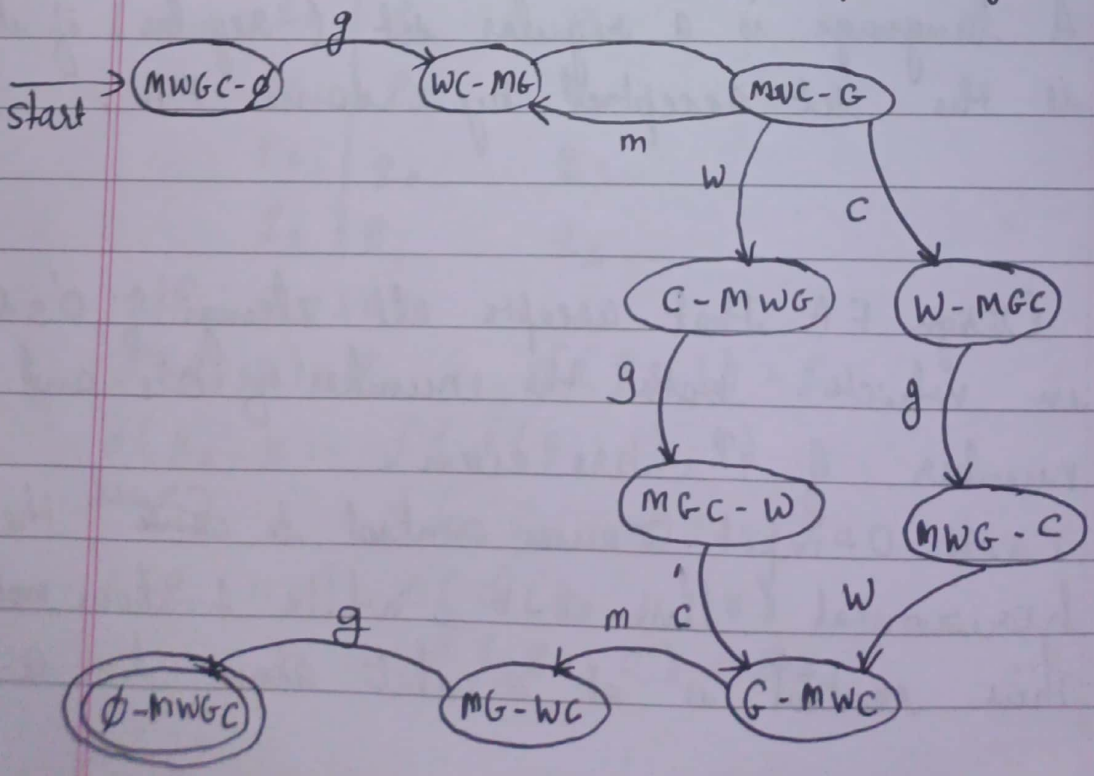
Unit II

Regular languages & FA.

A finite automaton is a mathematical model of a system with discrete I/P & O/P.
The system can be in any one of a finite number of internal configurations or "states".

Ex. Switching Circuits
Text Editers
Lexical Analysers.
Human Brain 2^{35}.
Pattern Recognition

Ex. A man with wolf, goat & cabbage is on the left bank of a river. There is a boat large enough to carry the man & only one of the other three. However, if the man leaves the wolf & goat unattended on either shore, the wolf will surely eat the goat. Illy if goat & cabbage are left unattended, the goat will eat the cabbage. Is it possible to cross the river without goat or cabbage being eaten.



FA is denoted by M

FA is denoted by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

where $Q \rightarrow$ A finite set of states.

Σ is a finite input alphabet

q_0 in Q is the initial state

$F \subseteq Q$ is set of Final states.

δ is the transition fun^n mapping $Q \times \Sigma$ to Q.

ie. $\delta(q, a)$ is a state for each state q & I/P symbol a.

$\bar{\delta}$ from $Q \times \Sigma^*$ to Q.

1) $\bar{\delta}(q, \epsilon) = q$

2) $\delta(q, wa) = \delta(\bar{\delta}(q, w), a)$

A string x is said to be accepted by FA $M = (Q, \Sigma, \delta, q_0, F)$ if $\delta(q_0, x) = p$ for some p in F.

The language accepted by M, designated $L(M)$ is the set $\{x \mid \delta(q_0, x)$ is in $F\}$.

A language is a regular set (regular) if it is the set accepted by some FA.

Ex. Design FA that accepts all strings of 0's & 1's in which both the number of 0's and number of 1's are even.

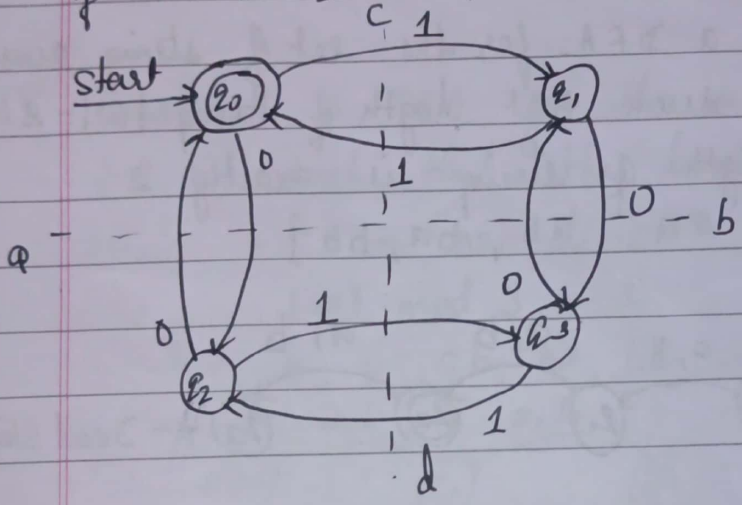
Each 0-input causes control to cross the horizontal line a-b, while 1 does not. Thus control is at a state above line a-b

Construct FA accepting set of strings over {0,1} such that ~~length of~~ $0(w) \bmod 2 = 0$ & $1(w) \bmod 2 = 0$ i.e. no. of 0's or no. of 1's should be divisible by 2 where 'w' is any string.

iff I/P seen so far contains an even number of 0's.

IIIly Control is at a state to left of the vertical line c-d iff I/P contains even no. of 1's.

Thus control is at q_0 iff there are both even number of 0's and even number of 1's in the I/P.



$Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{0, 1\}$, $F = \{q_0\}$

δ

	Inputs 0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

I/P 11010 1

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

$\delta(q_0, 11) = \delta(\delta(q_0, 1), 1)$

$= \delta(q_1, 1) = q_0$ ∴ 11 is in LM

$\delta(q_0, 110) = \delta(\delta(q_0, 11), 0)$

$= \delta(q_0, 0) = q_2$

...

$\delta(q_0, 1101) = q_3$

$\delta(q_0, 11010) = q_1$

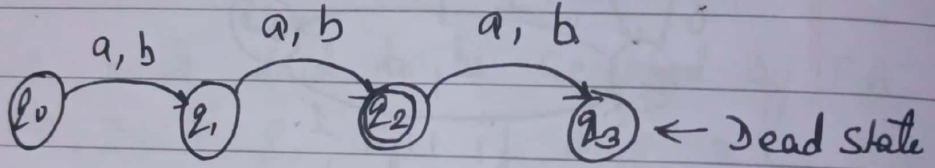
finally $\delta(q_0, 110101) = q_0$

1	1	0	1	0	1
q_0	q_1	q_0	q_2	q_3	q_1	q_0

$\therefore$ 110101 is in L(m).

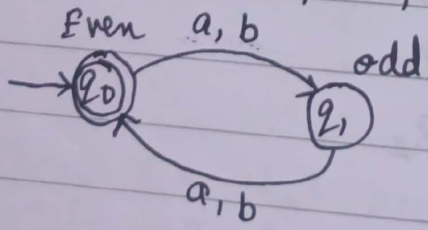
1) Construct a DFA for the set of string over {a, b} such that length of string $|w| = 2$ ie. length of string is exactly 2

Sol^n $L = \{aa, ab, ba, bb\}$

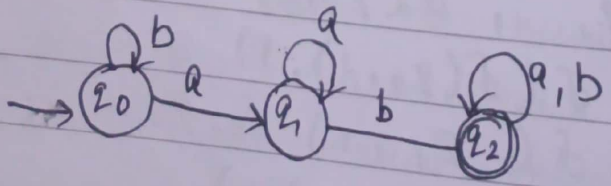


2) Construct a DFA for the set of string over {a, b} such that length of the string |w| is divisible by 2 i.e. $|w| \bmod 2 = 0$

$L = \{\epsilon, aa, ab, ba, bb, aaaa, bbbb \ldots\}$

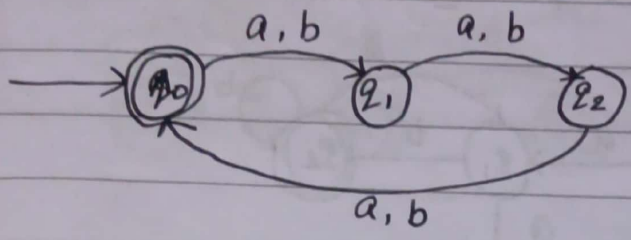


3



3) Construct a DFA for the set of string over {a, b} such that length of the string |w| mod 3 = 0.

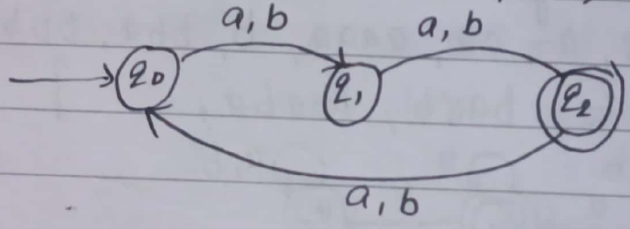
$L = \{ \epsilon, aaa, aab, aba, abb, aaaaaa, bbbbbb \ldots \}$



4) Construct a DFA for the set of string over {a, b} such that the length of the string |w| is not divisible by 3.

ie. $|w| \bmod 3 = 1$

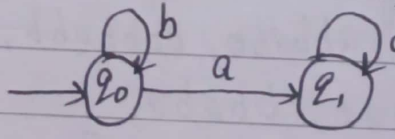
$L = \{ a, b, aa, ab, ba, bb, aaaa, bbbb \ldots \}$



5) Construct a DFA accepting set of string over {a, b} where each string contains 'a' as substring.

$L_1 = \{ a, aa, ab, ba, \ldots \}$

$L_2 =$

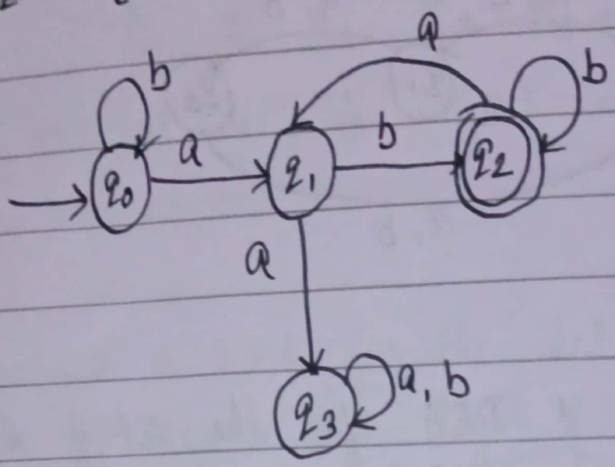


6) Construct a DFA accepting set of string over {a, b} where each string contains 'ab' as substring.

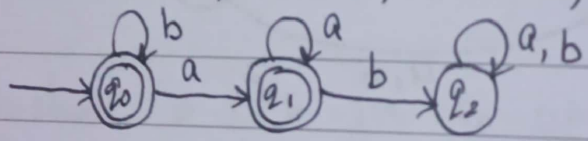
$L_1 = \{ ab, aab, abb, bab \ldots \}$

$L_2 = \{ aba, bba, bbbaaa, \ldots \}$

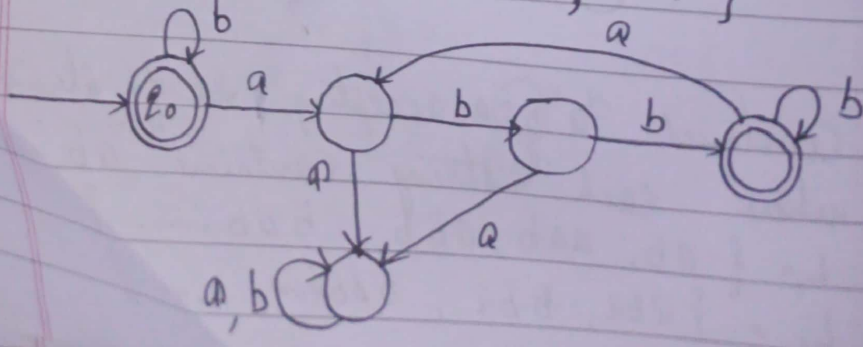
Construct DFA accepting set of strings over {a, b} in which every a is followed by
$L_1 = \{\epsilon, ab, abab, abbb, ababababab...\}$
$L_2 = \{ba, baab, bbaba....$



8) Construct DFA accepting set of strings over {a, b} in which every 'a' is never followed by 'b'
$L_1 = \{\epsilon, a, aa, aaaa, b, bba, bbbba...\}$
$L_2 = \{ba, baab, bbaba, ...\}$



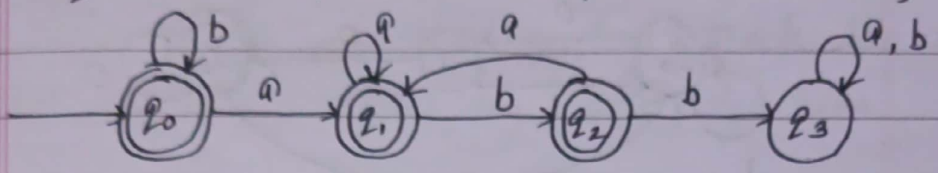
9) Construct a DFA accepting set of strings over {a, b} in which every 'a' is followed by a 'bb'
$L_1 = \{\epsilon, abb, abbabb, bbbbabb, abbabbabb...\}$
$L_2 = \{ba, baab, bbaba, ...\}$



10. Construct a DFA in which every 'a' is never followed by bb.

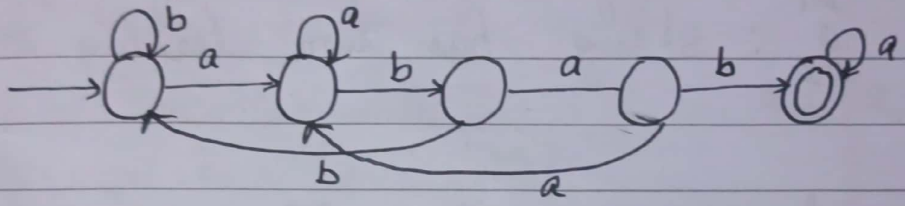
$L_1 = \{\epsilon, a, aa, aabaa, b, bb, bbbbba \ldots\}$

$L_2 = \{abb, babbabb, bbaba \ldots\}$

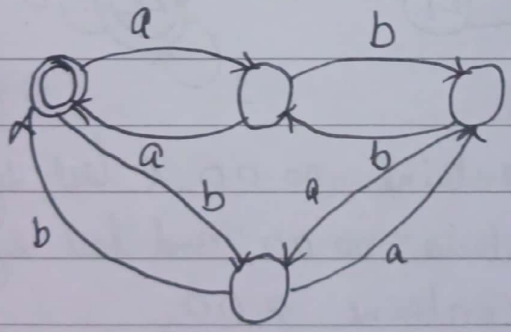


11. Construct a DFA accepting

$w \in \{a,b\}^*$: w has abab as a substring



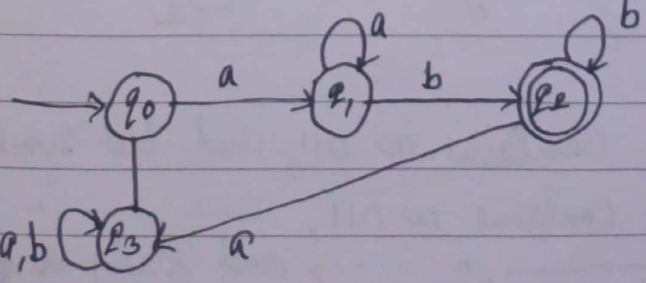
12 $w = \{(a,b)^*$: w has an odd number of a's and an even number of b's$\}$



13. $w = \{(a,b)^*$: $a^n b^m$ where n, m is greater than or equal to 1

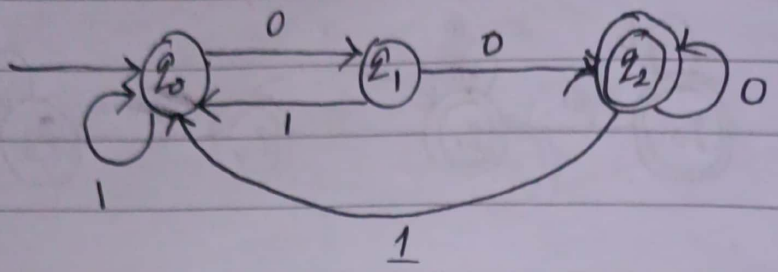
$L_1 = \{ab, aab, abb, aabb, aaabbb, aaabbb \ldots\}$

$L_2 = \{\epsilon, a, b \ldots\}$



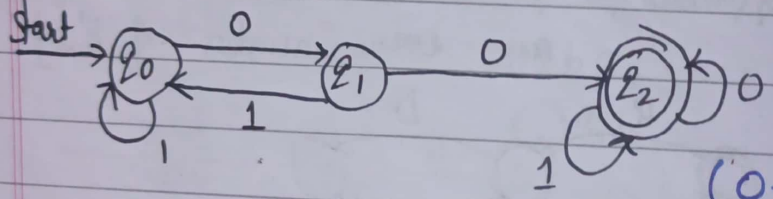
4) Give a DFA accepting the language over
48 the alphabet {0, 1}

i) The set of all strings ending in 00.



q_0 : string has no trailing zeros
q_1 : String has one trailing zeros
q_2 : string has two trailing zeros

ii) The set of all string with two consecutive 0's

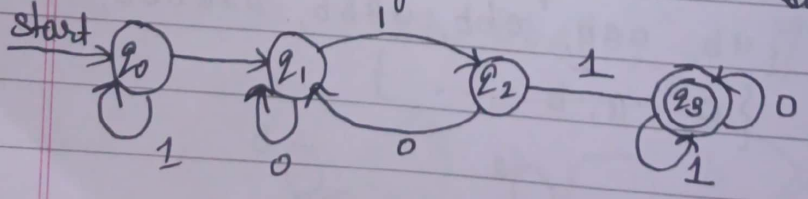


$(0+1)^*00(0+1)^*$

q_0 : String contains no 00, & last symbol was a one
q_1 : String contains no 00, and last symbol was a zero
q_2 : String contains a 00

. . . . .

iii) The set of strings with 011 as a substring



q_0 : String contains no 011, and last symbol was a one
q_1 : String contains no 011, " " zero
q_2 : " " ; and string so far ends 01
q_3 : String contains 011.

Ex 2.2.5
Page 54.

re $(0+1)^* 1 (0+1)^9$

19 Give a DFA accepting the following language over the alphabet $\{0, 1\}$.
The set of all strings whose tenth symbol from right is a '1'.

Proof: The states of the machine track the last 10 symbols read by the machine

$$Q = \{ x_1 x_2 \cdots x_n \mid 0 \le n \le 10, \text{ and } x_i \in \{0,1\} \text{ for all } i \in \{1, \ldots n\} \}$$

$q_0 = \epsilon$ (the empty string)

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

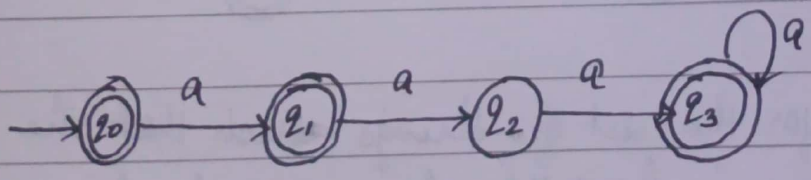
F (the accepting states) $= \{ \mid x_2 \ldots x_{10} \mid x_i \in \{0,1\}$ for all $i \in \{2 \ldots 10\} \}$

$$\delta(x_1 x_2 \ldots x_n, y) = \begin{cases} x_1 x_2 \cdots x_n y & \text{if } n < 10 \\ x_2 \cdots x_n y & \text{if } n = 10 \end{cases}$$

16. Construction of DFA accepting set of strings over $\{a\}$ in which $\{a^n \mid n \ge 0, n \ne 2$
ie 'n' should be greater than 0 and not equal to 2.

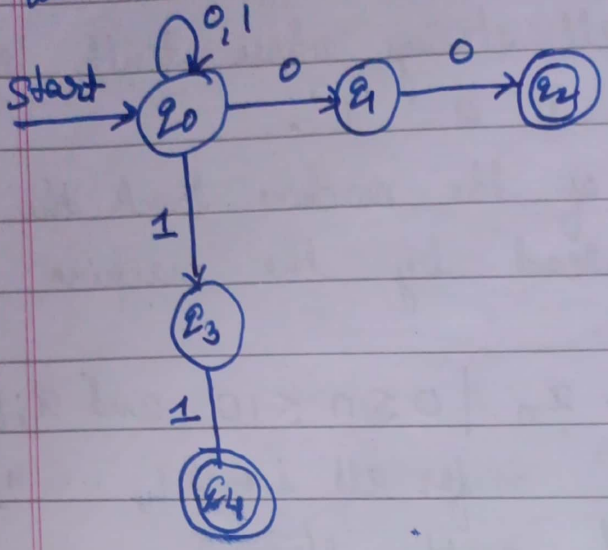
$L_1 = \{\epsilon, a, aaa, aaaa, aaaaa, \ldots\}$

not accepted $L_2 = \{aa, aaaa \ldots\}$ $\quad \neq a^2$



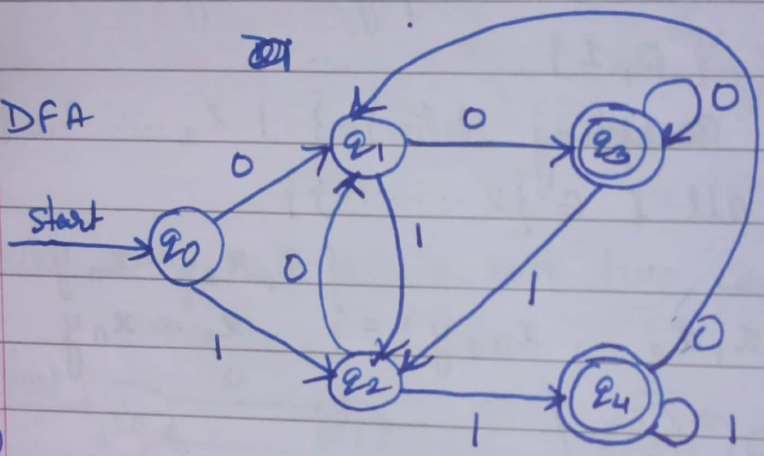
NFA

17. Set of all strings over {0,1} ending in 00 or 11



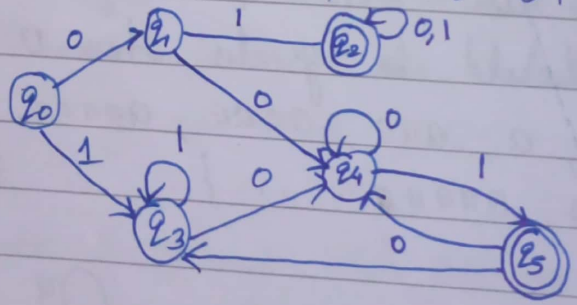
$(0+1)^*(00+11)$

DFA



2.2.5 (c)

18. DFA for the set of all strings that either begin or end (or both) with 01



d) DFA for the set of strings such that the number of 0's is divisible by 5 and the number of 1's is divisible by 3.

Number of zero's % 3 = 0

Number of One's % 5 = 0

$\therefore L = \{ \epsilon, 000, 11111, 00011111, 00111101, 11111000, 10101011, 000000011111 \cdots \}$

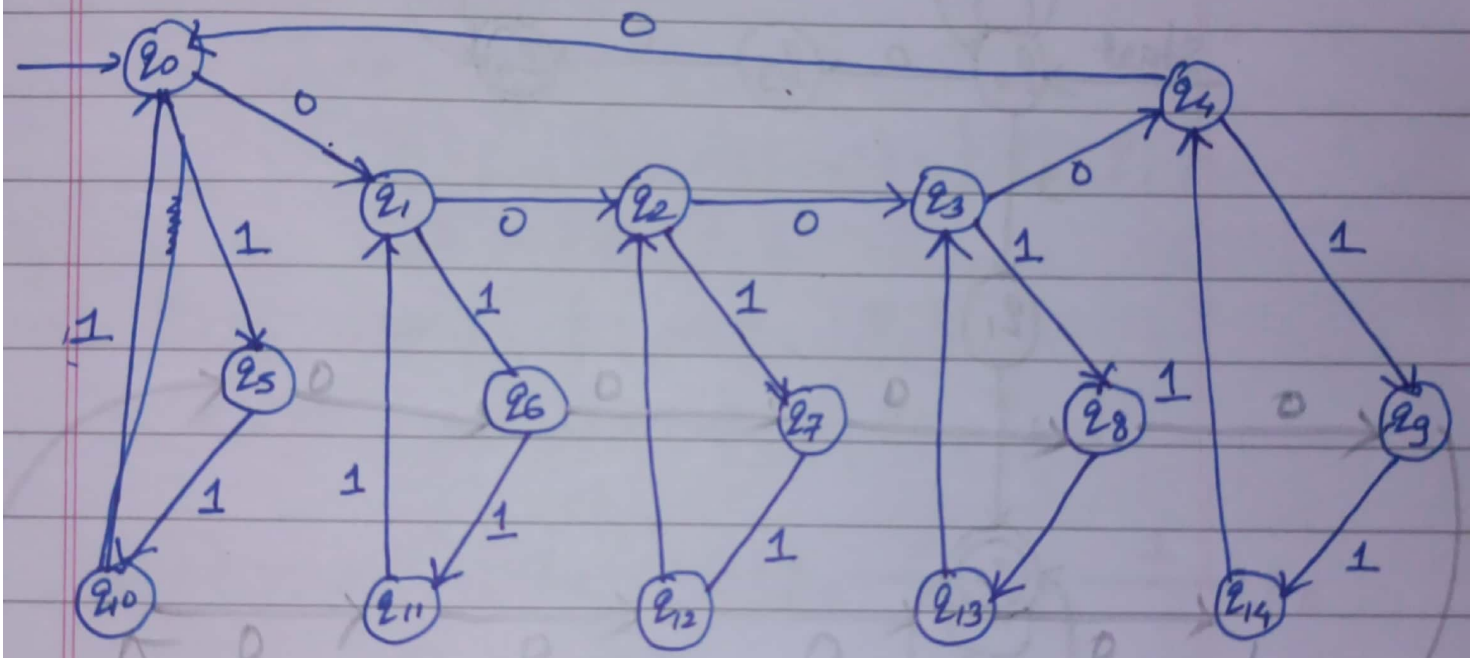
So, the strings accepted by DFA will be of length 0, 3, 5, 8, 11, 13, 14, 16 .....

i.e. $3x + 5y \quad x, y >= 0$

Modulo 3 gives remainder as (0, 1, 2)

& Modulo 5 gives remainder as (0, 1, 2, 3, 4)

$\therefore$ there will be $3 \times 5 = 15$ states for the automaton ..



NDFA : Non Deterministic FA

Consider modifying the finite automaton model to allow zero, one or more transition from a state on the same I/P symbol. This new model is called NFA.

Design an NFA that accepts strings with either two consecutive 0's or two consecutive 1's

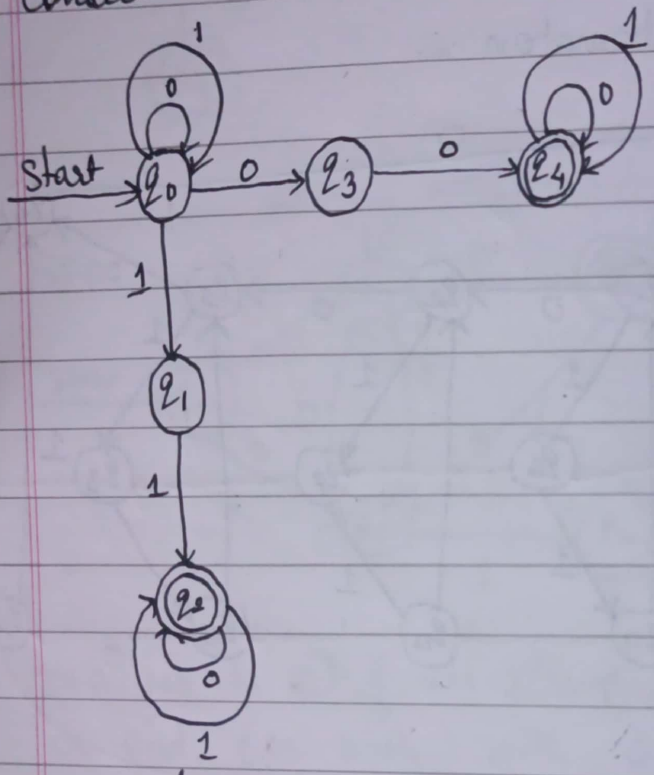


Fig: The transition diagram for NFA.

Formally we denote NFA by 5-tuple $(Q, \Sigma, \delta, q_0, F)$

Whe Q = no of states

Σ = input

q_0 = start state

F = final state

δ is a mapping from $Q \times \Sigma = 2^Q$ (power set of

States	0	1
q_0	$\{q_0, q_3\}$	$\{q_0, q_1\}$
q_1	ϕ	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$
q_3	$\{q_4\}$	ϕ
q_4	$\{q_4\}$	$\{q_4\}$

Input: 01001

$\delta(q_0, 0) = \{q_0, q_3\}$

$\delta(q_0, 01) = \delta(\delta(q_0, 0), 1)$

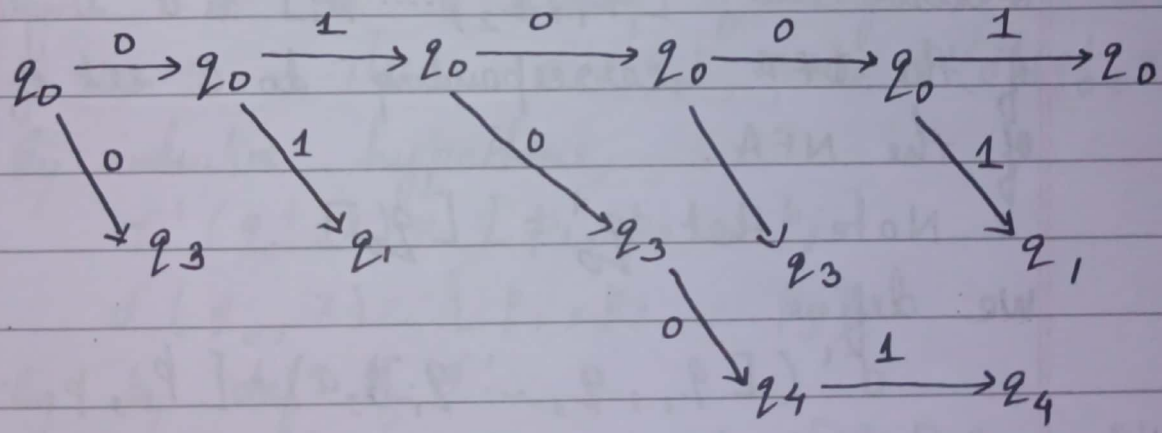
$= \delta(\{q_0, q_3\}, 1)$

$= \delta(q_0, 1) \cup \delta(q_3, 1) = \{q_0, q_3\}$

111 $\delta(q_0, 010) = \{q_0, q_3\}$

$\delta(q_0, 0100) = \{q_0, q_3, q_4\}$

& $\delta(q_0, 01001) = \{q_0, q_1, q_4\}$



The Proliferation of states of NFA. with input 01001

$\delta(q_0, 01001) = \delta(q_0, 1001)$

$= \delta(q_0, 001)$

$= \delta(q_3, 01)$

$= \delta(q_4, 1)$

$= \delta(q_4)$

Equivalence of DFA's and NFA's

Date
Page

orem 2-1 Let L be a set accepted by NFA. Then there exist a DFA that accepts L.

Proof: Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA accepting L.

Define a DFA, $M' = (Q', \Sigma, \delta', q_0', F')$ as follow
The states of M' are all subsets of the set of states of M.

ie. $Q' = 2^Q$.

M' will keep track in its state of all the state M could be in at any given time
F' is the set of all states in Q' containing final state of M.
An element of Q' will be denoted by $[q_1, q_2 \cdots q_i]$, where $q_1, q_2 \cdots q_i$ are in Q.

Observe that $[q_1, q_2, \cdots q_i]$ is a single state of the DFA corresponding to a set of states of the NFA.

Note that $q_0' = [q_0]$

We define

$$\delta'([q_1, q_2 \cdots q_i], a) = [p_1, p_2 \cdots p_j]$$

iff $\delta(\{q_1, q_2 \cdots q_i\}, a) = \{p_1, p_2 \cdots p_j\}$

i.e. δ' applied to an element $[q_1 q_2 \cdots q_i]$ of Q' is computed by applying δ to each state of Q represented by $[q_1 q_2 \cdots q_i]$.

On applying δ to each of $q_1, q_2 \cdots q_i$ and taking the union, we get some new set of

states $p_1, p_2 \cdots p_j$. This new set of states has a representative $[p_1, p_2 \cdots p_j]$ in Q' and that element is the value of $\delta'([q_1, q_2 \cdots q_i], a)$

By induction on the length of the input string x that

$$\delta'(q_0', x) = [q_1, q_2, \cdots q_i]$$

iff $\quad \delta(q_0, x) = \{q_1, q_2, \cdots q_i\}$

Basis: The result is trivial for $|x| = 0$, since $q_0' = [q_0]$ and x must by ϵ.

Induction: Suppose that the hypothesis is true for inputs of length m or less.
Let xa be a string of length $m+1$ with a in Σ. Then $\delta'(q_0', xa) = \delta'(\delta'(q_0', x), a)$
By inductive hypothesis,

$$\delta'(q_0', x) = [p_1, p_2 \cdots p_j]$$

iff $\quad \delta(q_0, x) = \{p_1, p_2 \cdots p_j\}$

By by def^n of δ'

$$\delta'([p_1, p_2, \cdots p_j], a) = [r_1, r_2 \cdots r_k]$$

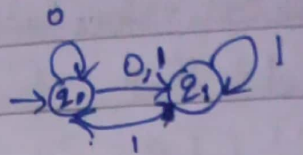
iff $\quad \delta(\{p_1, p_2, \cdots p_j\}, a = \{r_1, r_2 \cdots r_k\}$

thus $\quad \delta'(q_0', xa) = [r_1, r_2, \cdots r_k]$

iff $\quad \delta(q_0, xa) = \{r_1, r_2 \cdots r_k\}$

which establishes the inductive hypothesis.
To complete the proof, we have only to add that $\delta'(q_0', x)$ is in F' exactly when (q_0, x) contains a state of Q that is in F,
Thus $L(M) = L(M')$

Ex. Let $M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$ be an NFA where



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

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

$\delta(q_1, 0) = \emptyset$

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

We can construct a DFA $M' = (Q, \{0, 1\}, \delta', [q_0],$ accepting $L(M)$ as follows.

$2^Q = \{\emptyset, [q_0], [q_1], [q_0 q_1]\}$

Q consists of all subsets of $\{q_0, q_1\}$

we denote the elements of Q by $[q_0]$, $[q_1]$ $[q_0, q_1]$ and $\emptyset$.

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

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

Likewise

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

$\delta'([q_1], 0) = \emptyset$

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

Naturally $\delta'(\emptyset, 0) = \delta'(\emptyset, 1) = \emptyset$

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

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

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

$= \{q_0, q_1\}$

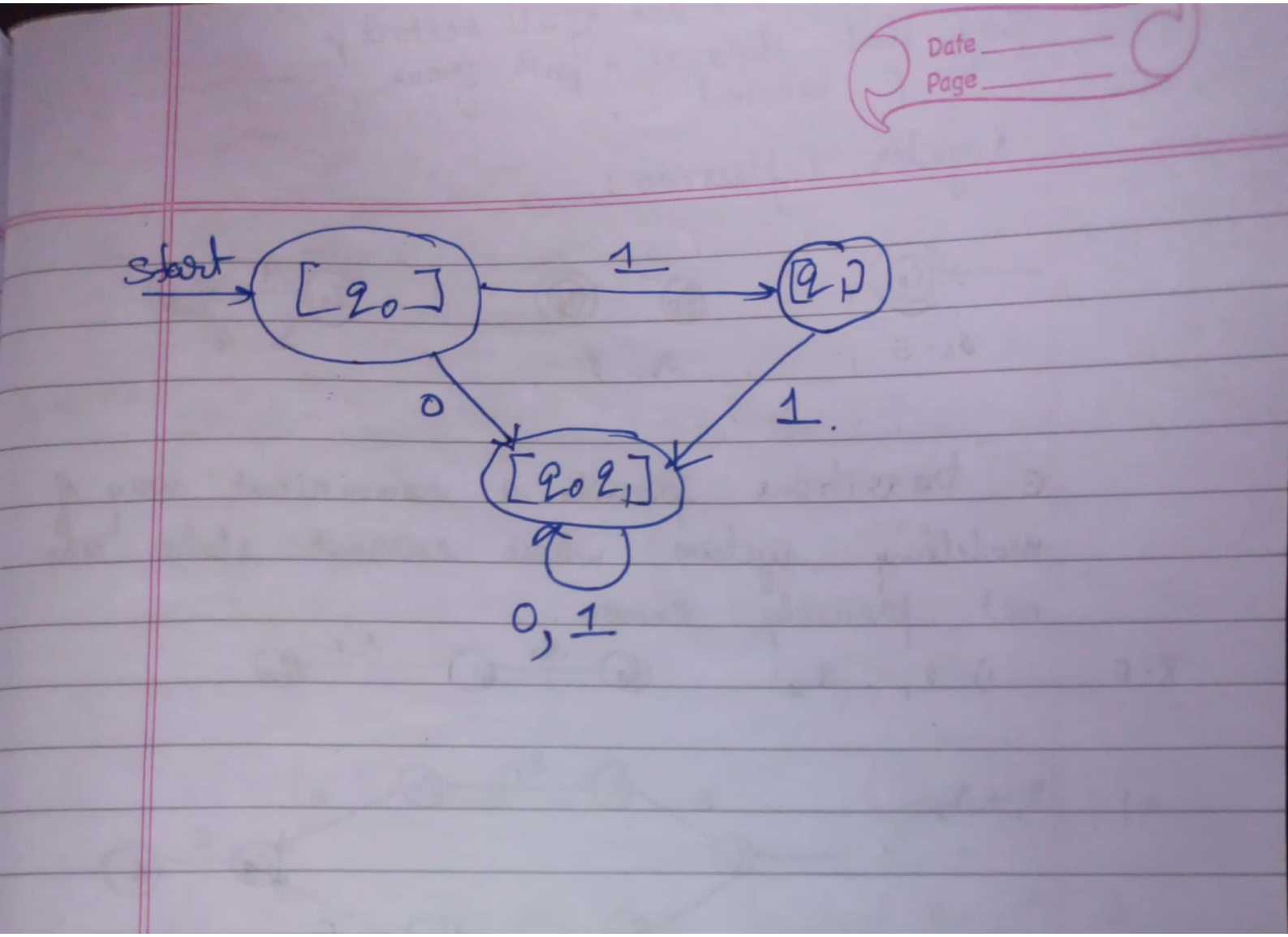
& $\delta'([q_0, q_1], 1) = [q_0, q_1]$

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

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

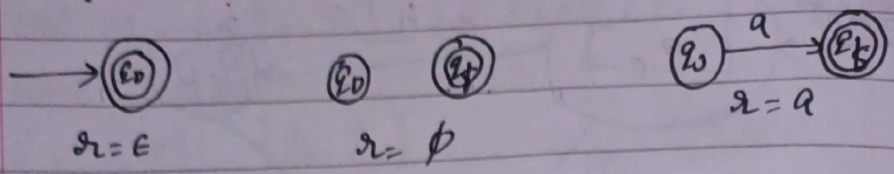
The set F of final state is

$\{[q_1], [q_0, q_1]\}$.



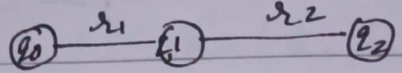
losure (q) : denotes the set of all vertices p such that there is a path from q to p labeled ∈.

Regular expression :

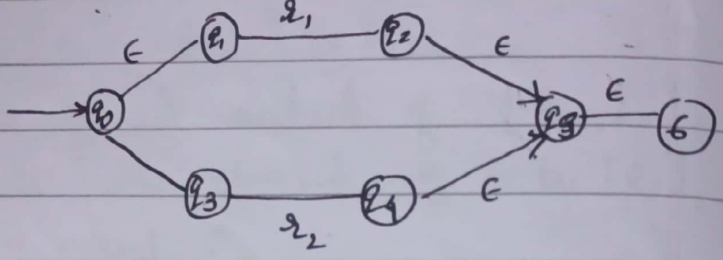


∈ transitions provide a convinient way of modelling systems whose current states are not precisely known.

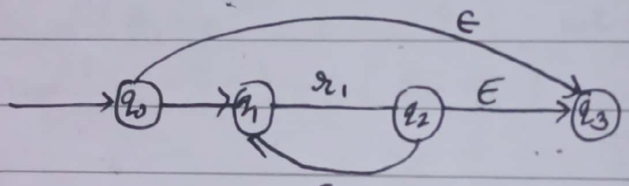
R.E 1) $r_1 . r_2$



2) $r_1 + r_2$



3) r_1^*



$r_1^* = \{ \in, a, aa \ldots \}$

Ex. 1) All the strings of 0's & 1's $(0+1)^*$

2) All strings of 0s and 1's with atleast two consecutive 0's. $(0+1)^* 00 (0+1)^*$

3) All strings of 0's & 1's beginning with 1 and not having two consecutive 0's. $(1+10)$

4) All strings of 0's & 1's that do not have consecutive 0's. $(0+\epsilon)(1+10)^*$

5) All strings of 0's & 1's ending in 011. $(0+1)$

6) Any no. of 0's followed by any no. of 1's followed by any no. of

7. The string in $0^* 1^* 2^*$ with atleast one of each symbol. $0^+ 1^+ 2^+$ or $00^* 11^* 22^*$

Pg. No. 28

Let Σ be a finite set of symbols and let L_1, L_2 be the set of strings from Σ^*. The concatenation of L_1 & L_2 denoted by $L_1 L_2$ is the set $\{xy \mid x$ is in L_1, and y is in $L_2\}$.

L_1 followed by L_2

$L^0 = \{\epsilon\}$ and $L^i = L L^{i-1}$ for $i \geq 1$.

The kleen closure of L, denoted by L^* is

$$L^* = \bigcup_{i=0}^{\infty} L_i$$

& Positive closure of L, denoted by L^+ is the set

$$L^+ = \bigcup_{i=1}^{\infty} L^i$$

Ex. Let $L_1 = \{10, 1\}$ and $L_2 = \{011, 11\}$

Then $L_1 L_2 = \{10011, 1011, 111\}$

$\{10, 11\}^* = \{\epsilon, 10, 11, 1010, 1011, 1110, \ldots\}$

If Σ is an alphabet the Σ^* denotes all strings of symbols in Σ.

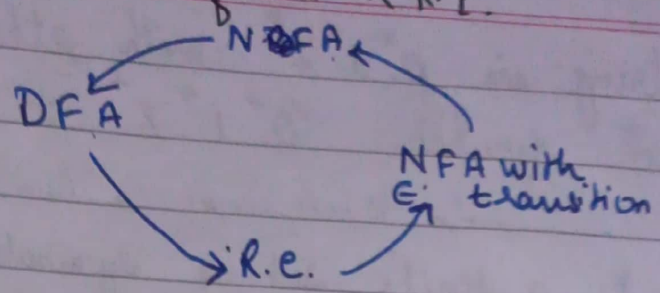
7. All strings starting & ending with a's and any no. of b's in betn.

$L(r) = \{aa, aba, abba, abbba, \ldots\}$

$r = a \cdot b^* \cdot a$

R.E. Regular Grammar.

Equivalence of FA & R.E.



Construct an NFA for the r.e $01^* + 1$.

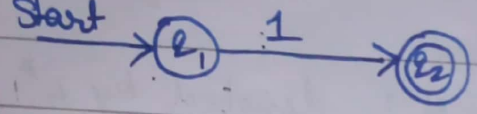
By our precedence rule $0(1)^* + 1$

$\therefore r_1 = 01^*$, $r_2 = 1$

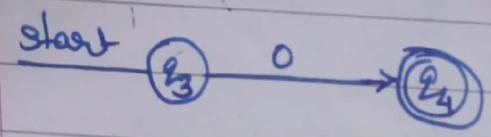
& $r_1 = r_3 r_4$ where $r_3 = 0$ and $r_4 = 1^*$

$r_4 = r_5^*$

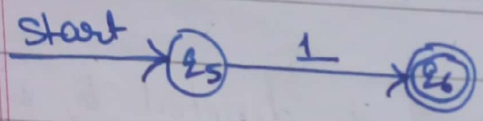
r_2



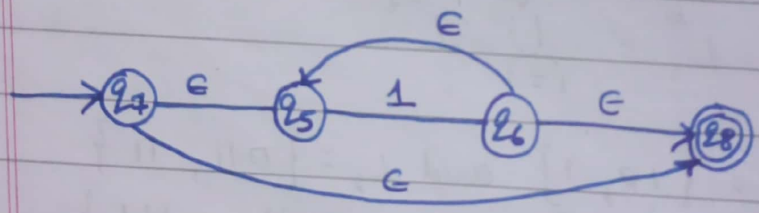
r_3



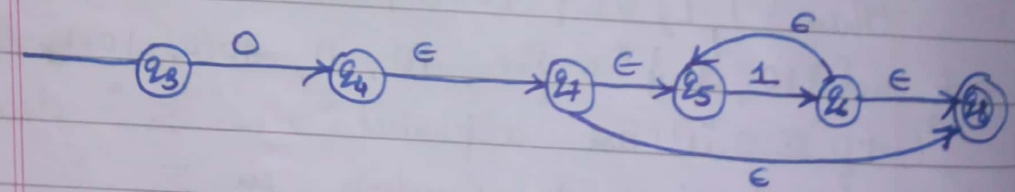
r_5



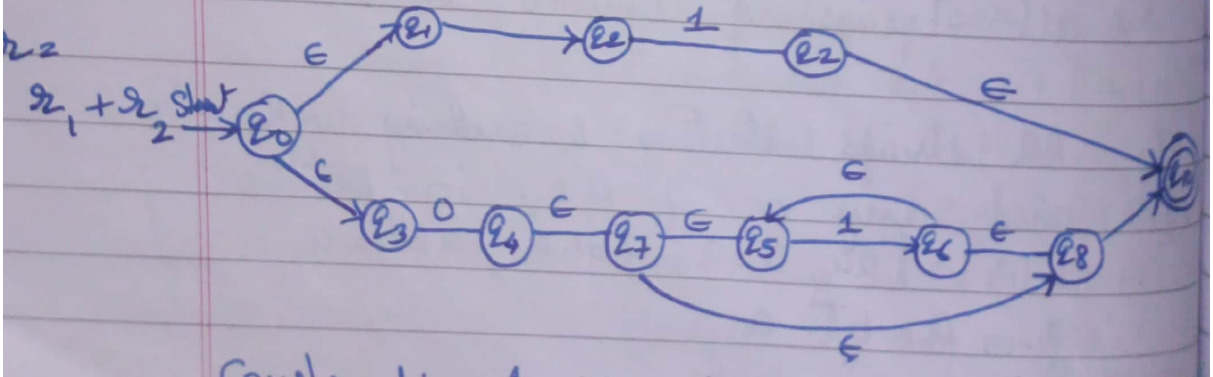
r_4



$r_1 = r_3 r_4$



$r_1 + r_2$



Construction of NFA for r.e. $01^* + 1$

Ex. Describe in English the language represented by following expressions.

i) $(a+ab)^*$: Every b is preceded by an a

ii) $(a+b)^*a(a+b)^*$: The string contains atleast one a.

iii) $(a^*ab^*ab^*)+b^*$: The string contains atleast two aa or a string of only b's

iv) $a^+b^*c^+$: One or more a's followed by any number of b's followed by one or more number of c's.

Ex. Find r.e. for the following: $\{0,1\}^*$

i) The language of all strings containing exactly two 0's

$1^*01^*01^*$

ii) String containing at least two 0's

$1^*01^*0(1+0)^*$

iii) String not ending in 01

$(1+0)^*(00+11+10)$

Ex Show that $(a^*b^*)^* = (a+b)^*$

r_1 r_2

$L(r_1) = \{(\epsilon, a, aa \ldots) \cdot (\epsilon, b, bb, \ldots)\}^*$

$= \{\epsilon, a, b, aa, bb, \ldots\}^*$

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

$L(r_2) = \{\epsilon, a, b, aa, ab, ba, bb \ldots\}$

$\therefore (a^*b^*)^* = (a+b)^*$

III/4 $(a^*+b^*)^* = (a+b)^*$

$(ab)^*a = a(ba)^*$

8 $(a.b)^* \neq a^*.b^*$

g.No: 32

Theorem 2.4

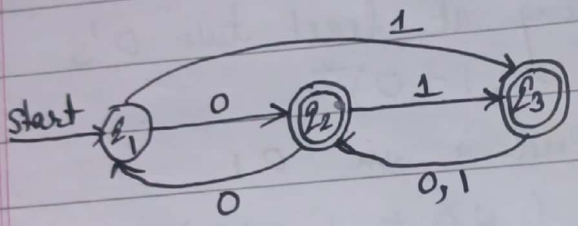
Let R_{ij}^{k} denote set of all strings x such that $\delta(q_i, x) = q_j$ and if $\delta(q_i, y) = q_i$

ie. R_{ij}^{k} is the set of all strings that take FA from state q_i to state q_j without going through state numbered higher than k.

$$R_{ij}^{k} = R_{ik}^{k-1} (R_{kk}^{k-1})^{*} R_{kj}^{k-1} \cup R_{ij}^{k-1},$$

$$R_{ij}^{0} = \begin{cases} \{a \mid \delta(q_i, a) = q_j\} & \text{if } i \neq j \\ \{a \mid \delta(q_i, a) = q_j\} \cup \{\epsilon & \text{if } i = j \end{cases}$$

Ex:



	k = 0	k = 1	k = 2
R_{11}^{k}	ϵ		
R_{12}^{k}	0		
R_{13}^{k}	1		
R_{21}^{k}	0		
R_{22}^{k}	ϵ		
R_{23}^{k}	1		
R_{31}^{k}	ϕ	ϕ	
R_{32}^{k}	0+1		
R_{33}^{k}	ϵ		

$R_{11}^{0} = \epsilon$, $R_{12}^{0} = 0$, $R_{13}^{0} = 1$, $R_{21}^{0} = 0$

$R_{22}^{0} = \epsilon$, $R_{23}^{0} = 1$, $R_{31}^{0} = \phi$, $R_{32}^{0} = 0+1$

$R_{33}^{0} = \epsilon$

$\phi r = \phi$ $\quad r \cdot r^* = r^+$

$r + r = r$ $\quad (P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$

$$r_{11}^1 = R_{11}^0 \cdot (R_{11}^0)^* R_{11}^0 \cup R_{11}^0$$

$$= \epsilon \cdot \epsilon^* \epsilon \cup \epsilon = \epsilon$$

$$r_{12}^1 = R_{11}^0 (R_{11}^0)^* R_{12}^0 \cup R_{12}^0$$

$$= \epsilon \cdot \epsilon^* \epsilon \cup 0 = 0$$

$$r_{13}^1 = R_{11}^0 (R_{11}^0)^* R_{13}^0 \cup R_{13}^0$$

$$= \epsilon \cdot \epsilon^* 1 \cup 1 = 1$$

$$r_{21}^1 = R_{21}^0 \cdot (R_{11}^0)^* R_{11}^0 \cup R_{21}^0$$

$$= 0 \cdot (\epsilon)^* \cdot \epsilon \cup 0 = 0$$

$$R_{22}^1 = R_{21}^0 \cdot (R_{11}^0)^* \cdot R_{22}^0 \cup R_{22}^0$$

$$= 0 \cdot (\epsilon)^* \cdot 0 \cup \epsilon \approx 0 \cdot (\epsilon)^+ 0 + \epsilon$$

$$\approx \epsilon + 00$$

$$R_{23}^1 = r_{21}^0 \cdot (r_{11}^0)^* \cdot r_{13}^0 \cup r_{23}^0$$

$$= 0 \cdot (\epsilon)^* \cdot 1 \cup 1 = 1 + 01$$

$$r_{11}^2 = r_{12}^1 \cdot (r_{22}^1)^* r_{21}^1 \cup r_{11}^1$$

$$= 0 \cdot (\epsilon + 00)^* \cdot 0 \cup \epsilon = 0 \cdot (00)^* = (00)^*$$

$$r_{12}^2 = r_{12}^1 \cdot (r_{22}^1)^* \cdot r_{22}^1 \cup r_{12}^1 = 0 (\epsilon + 00)^* (0 + \epsilon)$$

$$= 0 (00)^*$$

$$r_{13}^2 = r_{12}^1 \cdot (r_{22}^1)^* \cdot r_{23}^1 \cup r_{13}^1$$

$$= 0 \underbrace{(\epsilon + 00)^*}_{00^*} \underbrace{(1 + 01)}_{(\epsilon + 0)1} + 1$$

$$= 0 \underbrace{(00)^* (\epsilon + 0)}_{0^*} 1 + 1$$

$$= 0 \, 0^* 1 + 1$$

$$= 0^* 1$$

$r_{22}^2, \; r_{23}^2, \; r_{31}^2, \; r_{32}^2, \; r_{33}^3$

$$R_{12}^{3} = R_{13}^{2}\left(R_{33}^{2}\right)^{*} R_{32}^{2} + R_{12}^{2}$$

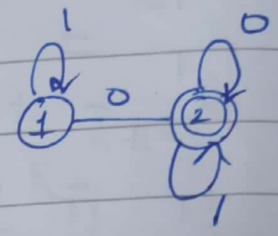
i)

$$= 0^{*}1\,(\epsilon + (0+1)0^{*}1)^{*}\,(0+1)(00)^{*} + 0(00)^{*}$$

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

& $R_{13}^{3} =$

Q. Obtain r.e for the DFA



Soln

$R_{11}^{(0)}$ $\epsilon + 1$

$R_{12}^{(0)}$ 0

$R_{21}^{(0)}$ $\emptyset$

$R_{22}^{(0)}$ $\epsilon + 0 + 1$

$$R_{11}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)} \cdot R_{11}^{(0)*} R_{11}^{(0)}$$

$$= (\epsilon+1) + (\epsilon+1)\cdot(\epsilon+1)^{*}(\epsilon+1)$$

$$= 1^{*}$$

$R_{11}^{(1)} = 1^{*}$

$R_{12}^{(1)} = 1^{*}0$

$R_{21}^{(1)} = \emptyset$

$R_{22} = \epsilon + 0 + 1$

$$R_{11}^{(2)} = R_{11}^{(1)} + R_{12}^{(1)} \cdot R_{22}^{(1)*} R_{21}^{(1)}$$

$$= 1^{*} + 1^{*}0(\epsilon+0+1)^{*}\emptyset$$

$$= 1^{*} + \emptyset$$

$$= 1^{*}$$

$$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)}\left(R_{22}^{(1)}\right)^{*} R_{22}^{(1)}$$

$$= 1^{*}0 + 1^{*}0(\epsilon+0+1)^{*}(\epsilon+0+1)$$

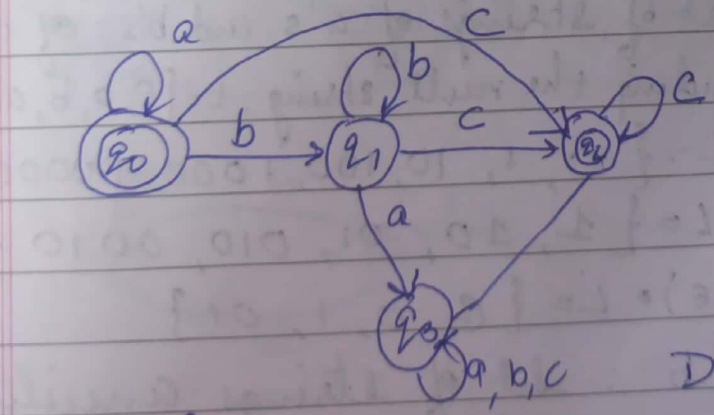
$$= 1^{*}0(0+1)^{*}$$

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

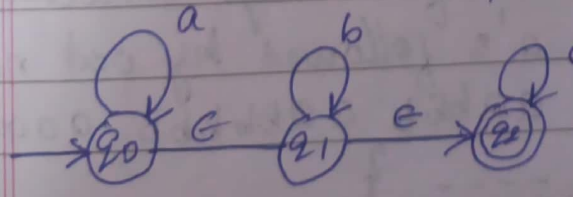
Identities for r.e.

1) $\phi + r = r$ (The identity for Union)
2) $\phi \cdot r = r \cdot \phi = \phi$
3) $\epsilon \cdot r = r \cdot \epsilon = r$ (The identity for Concatenation)
4) $\epsilon^* = \epsilon$
5) $r + r = r$ Idempotent Law
6) $r^* \cdot r^* = r^*$
7) $r \cdot r^* = r^*$
8) $(r^*)^* = r^*$
9) $\epsilon + r \cdot r^* = r^* = \epsilon + r^* r$
10) $(pq)^* p = p(pq)^*$
11) $(p+q)^* = (p^* q^*)^* = (p^* + q^*)^*$
12) $(p+q)^* r = pr + qr$ Right Distributive Law
13) $r(p+q) = rp + rq$ Left Distributive Law

$L = \{a^n b^m c^q \mid n, m, q \geq 0\}$



DFA:



ε - NFA

$r = a^* b^* c^*$

Write r.e. {0,1}

The set of all strings with at most one pair of consecutive 0's and at most one pair of consecutive 1's.

→ $(11)^*$: Set consisting of even no's of 1's including empty string, $L = \{\epsilon, 11, 1111, 111111, \ldots\}$

→ $(a+b)^*$: Set of strings of a's and b's of any length including the null string. $L = \{\epsilon, a, b, aa, ab, ba$

→ $(0+10^*)^*$ $L = \{0, 1, 10, 100, 1000, 10000, \ldots\}$

→ (0^*10^*) $L = \{1, 10, 01, 010, 0010 \ldots\}$

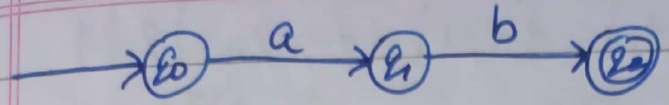
→ $(0+\epsilon)\cdot(1+\epsilon)^*$ $L = \{\epsilon, 0, 1, 01\}$

→ $(aa)^*(bb)^*b$: Set of strings consisting of even numbers of a's followed by odd no's of
$L = \{b, aab, aabbb, aabbbbb, aaaab,$
$aaaabbb \ldots\}$

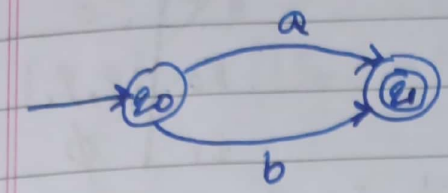
→ $(aa+ab+ba+bb)^*$: strings of a's & b's of even length can be obtained by concatenat any comb^n of strings aa, ab, ba & bb
$L = \{aa, ab, ba, bb, aaab, aaba, \ldots\}$

xKDn9U4SE2M
youtube

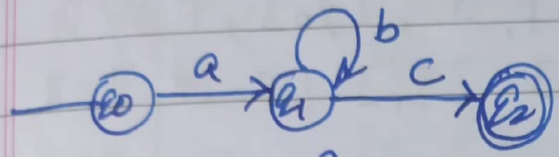
classmate
Date
Page

1. 

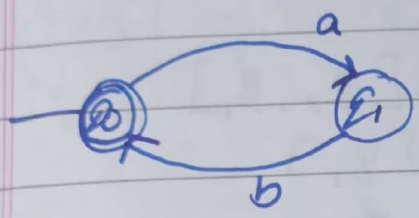
ab

2. 

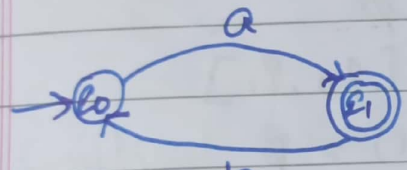
$a+b$

3. 

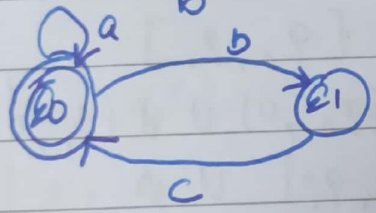
ab^*c

4. 

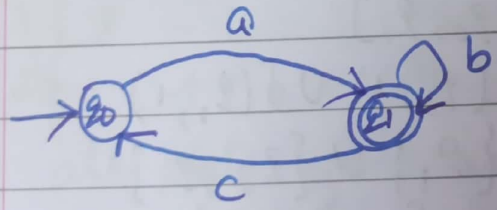
$(a \cdot b)^*$

5. 

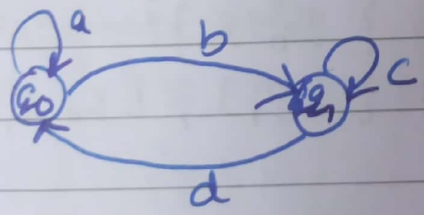
$a(ba)^*$

6. 

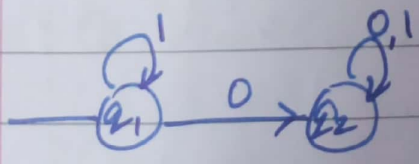
$(a^* + (bc)^*)^*$
$(a+bc)^*$

7. 

$a(b+ca)^*$

8. 

$a^*b(c+da^*b)^*$

9. 

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

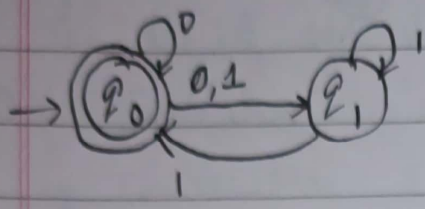
(Method I)

Eg. No. 23

NFA to DFA.

x. 2.5

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



Q \ Σ	0	1
q_0	$\{q_0, q_1\}$	$\{q_1\}$
q_1	ϕ	$\{q_0, q_1\}$

$M' = (Q, \{0,1\}, \delta', [q_0], F)$

Let Q be $[q_0]$ $[q_1]$ $[q_0, q_1]$ & ϕ

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

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

$\delta'([q_1], 0) = \phi$

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

turally $\delta'(\phi, 0) = \delta'(\phi, 1) = \phi$

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

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

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

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

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

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

$= \{q_0 q_1\}$

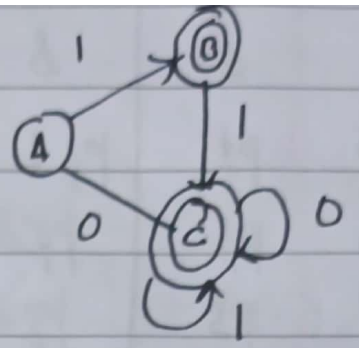
Set of final states $\{[q_1], [q_0, q_1]\}$

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

$M' = (Q, \Sigma, \delta', [q_0], F)$ where $\delta' = Q' \times \Sigma \to$

$Q' = \{[q_0][q_1][q_0, q_1]$

Q \ Σ	0	1
$[q_0]$	$[q_0, q_1]$	$[q_1]$
$[q_1]$	-	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$



Ex 2.9 Construct DFA equivalent to NFA's

	0	1
p	p, q	p
q	r	r
r	s	-
s	s	s

$Q' = \{ p, q, r, s, pq, pr, ps, qr, qs, rs, pqr, pqs, prs, qrs, pqrs \}$

As S is the final state all states having 's' as one of the constituents are final states for resultant DFA.

$F' = \{ s, ps, qs, rs, pqs, prs, qrs, pqrs \}$

$\delta'(p, 0) = pq$

$\delta'(q, 1) = r$

$\delta'(s, 1) = s$

$\delta'(pqr, 0) = [\delta(p, 0) \cup \delta(q, 0) \cup \delta(r, 0)]$

$= \{ p, q \} \cup \{ r \} \cup \{ s \} = [pqrs]$

	Q	Σ 0	1
1	p	pq (5)	p (1)
2	q	r (3)	r (3)
3	r	s (4)	-
* 4	s	s (4)	s (4)
5	pq	pqr (11)	pr (6)
6	pr	pqs (12)	p (1)
* 7	ps	pqs (12)	ps (7)
8	qr	rs (10)	r (3)
9	qs	rs (10)	rs (10)
* 10	rs	s (4)	s (4)
11	pqr	pqrs (15)	pr (6)
12	pqs	pqrs (15)	prs (13)
* 13	prs	pqs (12)	ps (7)
14	qrs	rs (10)	rs (10)
15	pqrs	pqrs (15)	prs (13)

(10) (4)
(13) (7)
(14) (9)
(15) (12)

DFA - Minimization

1. Replace one non-final state by equivalent non-final state.
2. Replace final state by its equivalent final state only.
3. Do not replace initial state by any other state.
4. Do not replace one final state by non-final state or non-final state by a final state.
5. Replacing state A by state B means deleting all entries related to state A.

Wherever we find any transition where A is the next-state, we replace it by B and state A can be ~~dated~~ deleted.

6- Repeat for more steps.

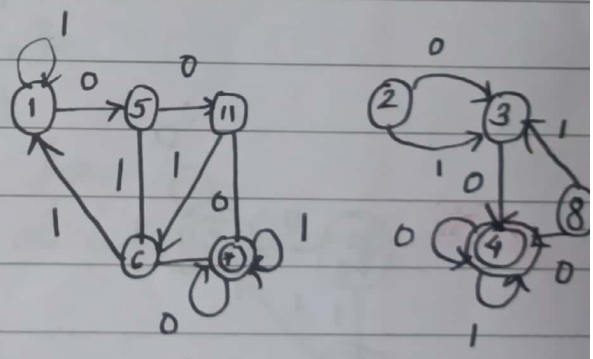
	Q' \ Σ	0	1
	1	5	1
	2	3	3
	3	4	–
*	4	4	4
	5	11	6
	6	12	1
*	7	12	7
	8	4 •	3
*	9	4 •	4
	11	12 *	6
*	12	12 *	7

(4) (9) → both & final

(7) (12) → both & final

Final

	Q' \ Σ	0	1
	1	5	1
	2	3	3
	3	4	–
*	4	4	4
	5	11	6
	6	7•	1
*	7	7•	7
	8	4	3
	11	7•	6



Exclude part 2

Change the labels by A, B, C, D, E.

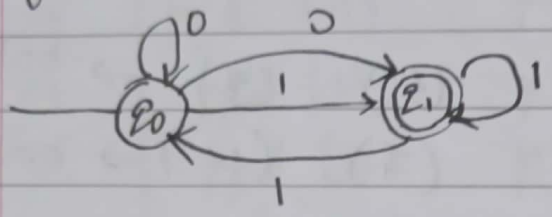
• Modified entries due to replacement.

* final states

Method (II)

It starts finding the transitions one state at a time. If the next state of a given transition is a new comb^n. then it gets added to the set of state for the resultant DFA.

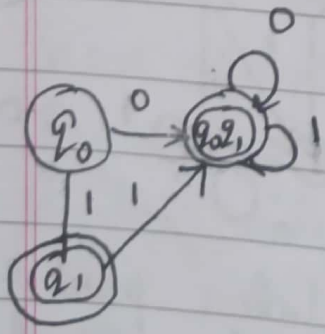
Ex



As there is a transition from q_0 to q_0, q_1 on 0
∴ add $[q_0 q_1]$.

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

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



FA with ϵ moves

NFA with & without ϵ-moves

$\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$

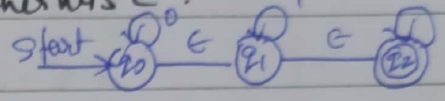
Pg.No. 26

Theorem If L is accepted by an NFA with ϵ transitions then L is accepted by an NFA without ϵ.

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

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

$$F' = \begin{cases} F \cup \{q_0\} & \text{if } \epsilon \text{ closure } (q_0) \text{ contains } F \\ F & \text{otherwise.} \end{cases}$$



Ex

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

Fig:- State Transition table for NFA without ϵ transition

$\hat{\delta}(q, a) = \epsilon$ - closure $(\delta(\hat{\delta}(q, \epsilon), a)$

$\hat{\delta}(q, \epsilon) = \epsilon$ - closure (q)

ϵ - closure $(q_0) = \{q_0, q_1, q_2\}$

ϵ - closure $(q_1) = \{q_1, q_2\}$

ϵ-closure $(q_2) = \{q_2\}$

q_2 is at zero distance from q_0.

q_2 is at zero distance from q_1.

$\therefore F = \{q_2\}$

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

$\delta'(q_0, 0) = \epsilon$ closure $\delta(\hat{\delta}(q_0, \epsilon), 0))$

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

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

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

$= \epsilon$ - closure (q_0)

$= [q_0, q_1, q_2]$

$\delta'(q_0, 1) = \epsilon\text{-closure}\ \delta(\hat{\delta}(q_0, \epsilon), 1))$

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

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

$= \epsilon\text{-closure}(\phi \cup \{q_1\} \cup \phi)$

$= \epsilon\text{-closure}(q_1)$

$= [q_1, q_2]$

$\delta'(q_0, 2) = \epsilon\text{-closure}\ \delta(\hat{\delta}(q_0, \epsilon), 2))$

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

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

$= \epsilon\text{-closure}(\phi \cup \phi \cup \{q_2\})$

$= \epsilon\text{-closure}(q_2)$

$= \{q_2\}$

$\delta'(q_1, 0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_1, \epsilon), 0))$

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

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

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

$= \phi$

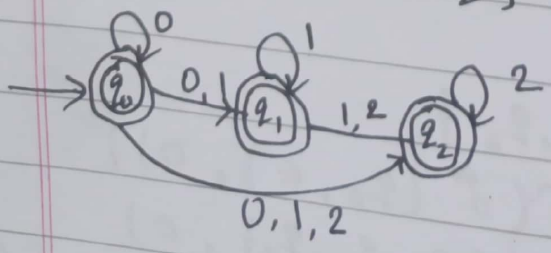
114

$\delta'(q_1, 1) = \{q_1, q_2\}$

$\delta'(q_1, 2) = \{q_2\}$

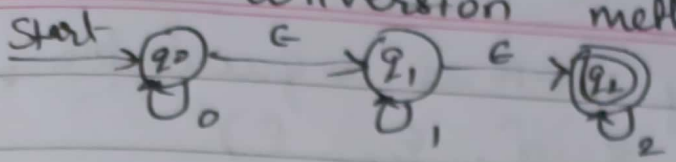
$\delta'(q_2, 0) = \phi$ $\quad \delta'(q_2, 1) = \phi$

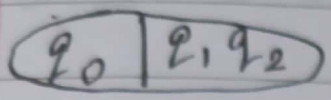
$\delta'(q_2, 2) = \{q_2\}$



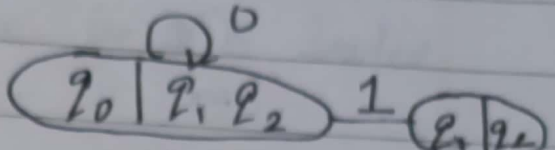
Indirect conversion method :

Ex.



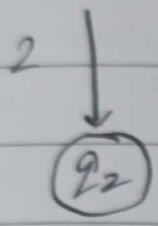


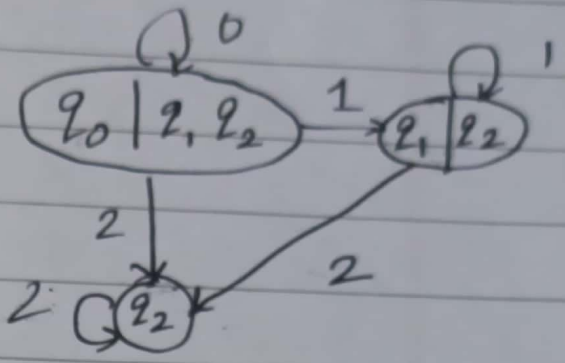
(a)



(b)

$q_0 | q_1 q_2 \rightarrow q_1 q_2$





Date
Page

2DFA

A two-way DFA is a quintuple

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

where δ is maping from $Q \times \Sigma$ to $Q \times \{$

Finite Automata with output.

FA is a mathematical model of FSM which does not include info regarding O/P. It generates only two outcomes

The def^n of FA includes limited power of FSM as language acceptor. The FSM with output have two types that generate O/P.

1. moore Machine 2. mealy m/c.

1. moore m/c: A moore m/c is a machine with finite no. of states and for which the O/P symbol at any given time depends upon only the current state of the m/c.

A moore m/c is six-tuple

$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

λ = Machine Function; $\lambda : Q \rightarrow \Delta$

Δ is an O/P alphabet

Ex. Construct a moore m/c to find out the residue-modulo-3 for binary numbers

Sol^n It is observed that if i is written in binary is followed by a 0, the resulting string has the value 2i and if i in binary is followed by a 1, the resulting string has a value 2i+1.

5 - residue 2

i = 1 value 1

write 0 after i ie 10 value 2×

i = 400 value = 4

1.0 = 1000 value 8

i = 1 value 1

i.1 = 11 value 2×1+1 = 3

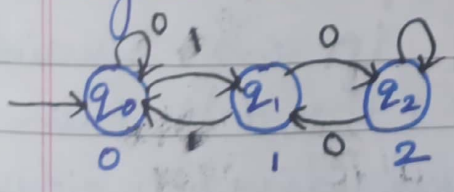
i = 100

i.1 = 1001 2×4+1 = 9

If we divide a no. by 3 we get different remainder values 0, 1 and 2

Remainder value (R) (Remainder from previous step)	0	1	2
where we write 0 after R /3	2R mod 3	0	2	1
when we write 1 after R & divide /3	(2R+1) mod 3	1	0	2

Read the I/P from left to right, one digit at a time. The digit we read is divided by a divisor. The next digit is then concatenated to the remainder of this division to form the next no. to be divided. We continue this process till all the digits in the I/P string are exausted.



Q \ Σ	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

Ex I/P 10110 states q_0, q_1, q_2, q_2, q_1

O/P is 0 1 2 2 1

Design an FSM for divisibility by 3 for decimal numbers.

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

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

$O = \{0, 1\}$ no. is divisible by 3 O/P is 1
no is not —"— O/P is 0

If we divide a no. by 3 we get 0, 1 and 2 as remainder.

$S = \{S_0, S_1, S_2\}$

MAF: $I \times S \rightarrow O$ STF: $I \times S \rightarrow S$

S \ I	(0,3,6,9)	(1,4,7)	(2,5,8)
S_0	1	0	0
S_1	0	0	1
S_2	0	1	0

S \ I	0,3,6,9	1,4,7	2,5,8
S_0	S_0	S_1	S_2
S_1	S_1	S_2	S_0
S_2	S_2	S_0	S_1

At every step in the division of any multi-digit decimal input, the remainder from the previous division step is concatenated to the next I/P digit to form the number to be considered for division in next step.

i.e. If m/c is in S_1 i.e. one-remainder state and the i/p is (1, 4, 7) then next step considers for division either 11, 14 or 17 resp. Now if we divide these numbers by 3, we get a

Final state is either S_0, S_1 or S_2

remainder 2. Hence the m/c moves from S_1 to S_2 with O/P 0. This indicates that the no. 11, 14, 17 is <u>not</u> divisible by 3.

111 If m/c is in S_2 and I/P is (1, 4, 7) then the numbers formed are 21, 24 or 27 which are divisible by 3.

∴ the m/c moves to zero remainder state S_0 with O/P 1. This indicates that the no. 21, 24, 27 is divisible by 3

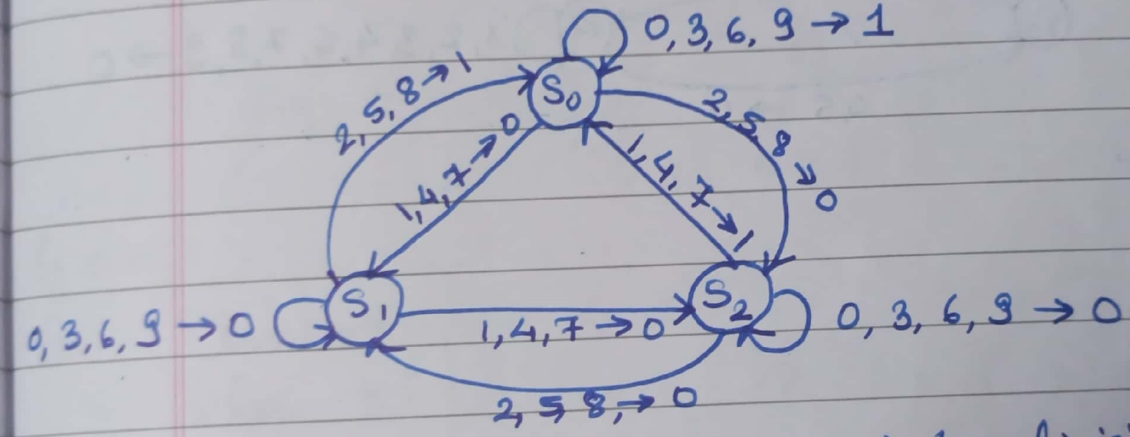


Fig:- Transition graph (TG) for divisibility by 3 tester

Ex. 112

Current state	I/P	Next State	Output
S_0	1	S_1	0
S_1	1	S_2	0
S_2	2	S_1	1

Non-Divisible

141 6

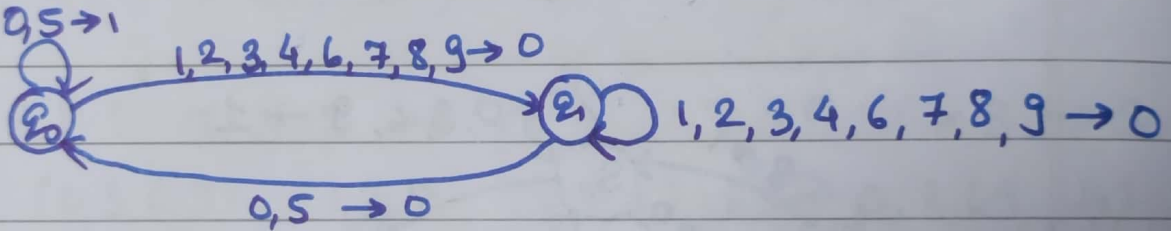
Current state	I/P	Next state	O/P
S_0	1	S_1	0
S_1	4	S_2	0
S_2	1	S_0	1
S_0	6	S_0	1

Divisible

Table :- Transition matrix for divisibility by 3 tester

Current state \ Next state	S_0	S_1	S_2
S_0	0/1, 3/1, 6/1, 9/1	1/0, 4/0, 7/0	2/0, 5/0, 8/0
S_1	2/1, 5/1, 8/1	0/0, 3/0, 6/0	1/0, 4/0, 7/0
S_2	1/1, 4/1, 7/1	2/0, 5/0, 8/0	0/0, 3/0, 6/0, 9/0

Ex. Design an FSM for divisibility by 5-tester for decimal no.



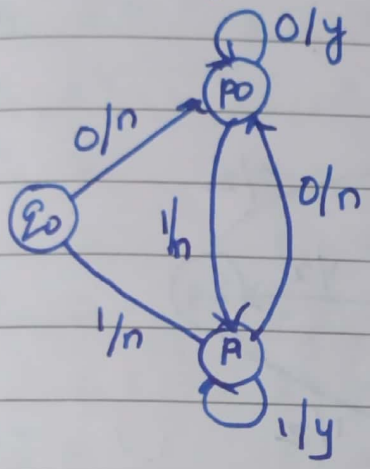
Mealy machine

A mealy m/c is a six tuple $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where λ maps from $Q \times \Sigma$ to Δ.

pg 43

Ex. Design a mealy m/c that accepts the language consisting of string from Σ^*, where $\Sigma = \{0, 1\}$ and ending with double '0's or double 1's.

Soln



$M = (\{q_0, P_0, P_1\}, \{0, 1\}, \{y, n\}, \delta, \lambda, q_0)$

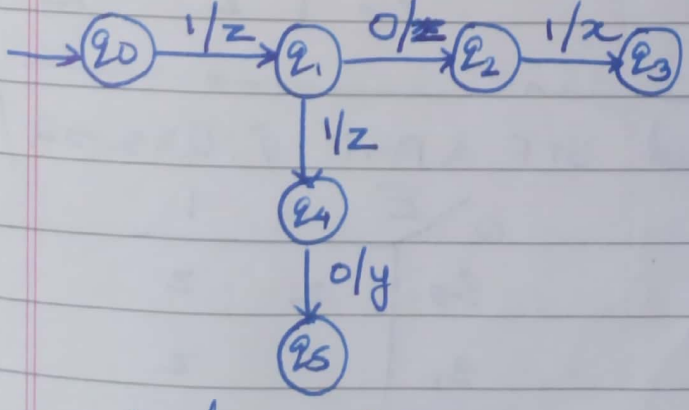
I/P: 01100 nnyny.

Fig: A mealy m/c.

pg.52

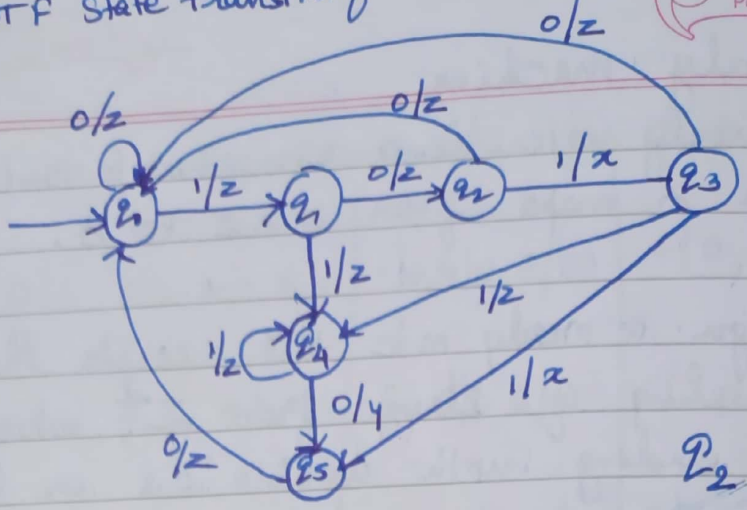
Ex. Construct moore & mealy m/c for the I/P from Σ^* where $\Sigma = (0, 1)$, if the I/P ends in '101', the o/p should be x, if the input ends in '110' o/p should be y otherwise o/p should be z.

Soln



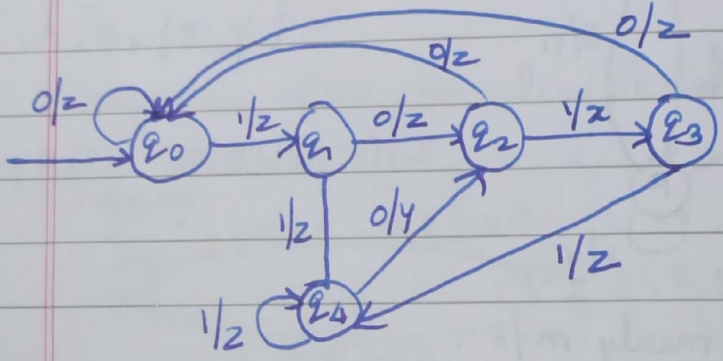
Mealy m/c.

$I \times S \to O$ MAF machine fun is an onto
$I \times S \to S$ STF state transition fn



q_2 & q_0 are equivalent

1c)



STF $\delta : Q \times \Sigma \to Q$ | MAF $\lambda : Q \times \Sigma \to \Delta$

Q \ Σ	0	1
q_0	q_0	q_1
q_1	q_2	q_4
q_2	q_0	q_3
q_3	q_0	q_4
q_4	q_5	q_4
q_5	q_0	q_3

Q \ Σ	0	1
q_0	z	z
q_1	z	z
q_2	z	x
q_3	z	z
q_4	y	z
q_5	z	x

Table: Reduced STF & MAF $\delta : Q \times \Sigma \to Q$ | $\lambda : Q \times$

Q \ Σ	0	1
q_0	q_0	q_1
q_1	q_2	q_4
q_2	q_0	q_3
q_3	q_0	q_4
q_4	q_2	q_4

Q \ Σ	0	1
q_0	z	z
q_1	z	z
q_2	z	x
q_3	z	z
q_4	y	z

moore machine

out of the state $[q_0, x]$, $[q_0, y]$ & $[q_0, z]$ any one can be considered as the initial state and other two can be omitted in order to get the final answer.

STF $\delta' : Q' \times \Sigma \to Q'$ MAF : $\lambda' : Q' \to \Delta$

Q' \ Σ	0	1	Q	Δ
$[q_0, x]$	$[q_0, z]$	$[q_1, z]$	$[q_0, x]$	x
$[q_0, y]$	$[q_0, z]$	$[q_1, z]$	q_0, y	y
$[q_0, z]$	$[q_0, z]$	q_1, z	q_0, z	z
$[q_1, x]$	q_2, z	q_4, z	q_1, x	x
$[q_1, y]$	q_2, z	q_4, z	q_1, y	y
$[q_1, z]$	q_2, z	q_4, z	q_1, z	z
$[q_2, x]$	q_0, z	q_3, x	q_2, x	x
$[q_2, y]$	q_0, z	q_3, x	q_2, y	y
$[q_2, z]$	q_0, z	q_3, x	q_2, z	z
$[q_3, x]$	q_0, z	q_4, z	q_3, x	x
$[q_3, y]$	q_0, z	q_4, z	q_3, y	y
$[q_3, z]$	q_0, z	q_4, z	q_3, z	z
$[q_4, x]$	q_2, y	q_4, z	q_4, x	x
$[q_4, y]$	q_2, y	q_4, z	q_4, y	y
$[q_4, z]$	q_2, y	q_4, z	q_4, z	z

Cheap
easy to use
very fast

Reactive
Low response time

Moore to mealy conversion

oore $M_1 = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

ealy $M_2 = (Q, \Sigma, \Delta, \delta, \lambda', q_0)$

where $\lambda'(q, a) = \lambda(\delta(q, a))$ $\forall (q \in Q \& \forall$

Associate O/P of next state $\lambda(\delta(q,a))$ with present
mealy m/c $\lambda'(q, a)$

oore : O/P depends on m/c's state

ealy : O/P depends on m/c's state & input sy

∴ the difference is λ.

x : Convert moore to mealy

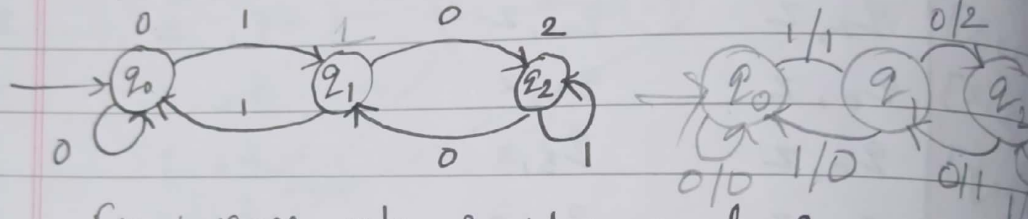


fig : moore m/c residue - mode 3

As per the conversion rule

$\lambda'(q, a) = \lambda(\delta(q, a))$

∴

$\lambda'(q_0, 0) = \lambda(\delta(q_0, 0)) = \lambda(q_0) = 0$

$\lambda'(q_0, 1) = \lambda(\delta(q_0, 1)) = \lambda(q_1) = 1$

$\lambda'(q_1, 0) = \lambda(\delta(q_1, 0)) = \lambda(q_2) = 2$

$\lambda'(q_1, 1) = \lambda(\delta(q_1, 1)) = \lambda(q_0) = 0$

$\lambda'(q_2, 0) = \lambda(\delta(q_2, 0)) = \lambda(q_1) = 1$

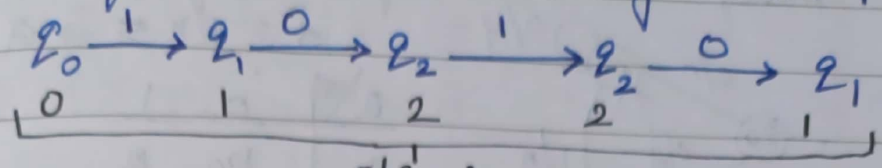
$\lambda'(q_2, 1) = \lambda(\delta(q_2, 1)) = \lambda(q_2) = 2$

Create MAF of the equivalent $\lambda' : Q \times \Sigma$

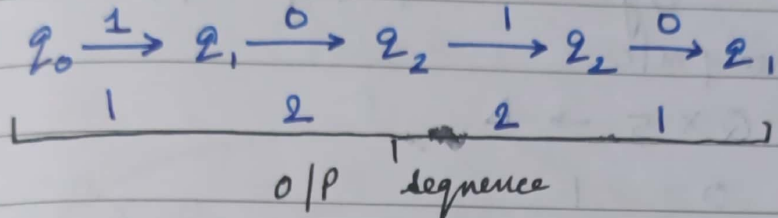
Q \ Σ	0	1
q_0	0	1
q_1	2	0
q_2	1	2

Ex: Vending m/c is an FA which recognizes coins of different denominations.

I/P sequence 1010 length $n = 4$



For moore m/c length of O/P seq = 5 $(n+1)$



For mealy m/c length of O/P seq = 4 = n

Mealy to Moore:

Mealy $M_1 = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

Moore $M_2 = ([Q \times \Delta], \Sigma, \Delta, \delta', \lambda', [q_0, b_0])$

where b_0 is an arbitrarily selected member of Δ

$\delta'([q, b], a) = [\delta(q, a), \lambda, (q, a)]$

$\lambda'([q, b]) = b$

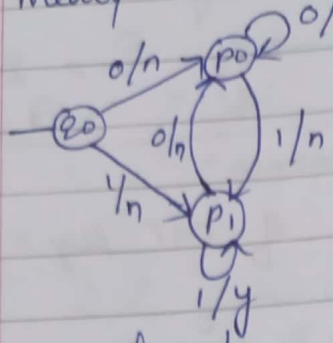
As we cannot determine the O/P symbol that the equivalent Moore m/c holds in each state, so we create all possible combinations of the state symbols & O/P symbols $Q \times \Delta$.

$\therefore \lambda'([q, b], a) = [\delta(q, a), \lambda(q, a)]$

The O/P symbol b has no role to play in determining next state ie.

$[q, b_1]$, $[q, b_2]$ makes transition to same next state.

Ex. Mealy m/c to accept strings ending with 01



$\delta : Q \times \Sigma \to Q$

Q \ Σ	0	1
q_0	p_0	p_1
p_0	p_0	p_1
p_1	p_0	p_1

Fig: mealy m/c

$\lambda : Q \times \Sigma \to \Delta$

Q \ Σ	0	1
q_0	n	n
p_0	y	n
p_1	n	y

Let Q' be the moore m/c

$Q' = Q \times \Delta = \{[q_0, n], [q_0, y], [p_0, n], [p_1, n], [p_1$

1. $\delta'([q_0, n], 0) = \delta(q_0, 0), \lambda(q_0, 0) = [p_0, n]$
   $\delta'([q_0, n], 1) = \delta(q_0, 1), \lambda(q_0, 1) = [p_1, n]$
   $\lambda'([q_0, n]) = n$
2. $\delta'([q_0, y], 0) = \delta(q_0, 0), \lambda(q_0, 0) = [p_0, n]$
   $\delta'([q_0, y], 1) = \delta(q_0, 1), \lambda(q_0, 1) = [p_1, n]$
   $\lambda'([q_0, y]) = y$
3. $\delta'([p_0, n], 0) = [\delta(p_0, 0), \lambda(p_0, 0)] = [p_0, y]$
   $\delta'([p_0, n], 1) = [\delta(p_0, 1), \lambda(p_0, 1)] = [p_1, n]$
   $\lambda'([p_0, n]) = n$
4. $\delta'([p_0, y], 0] = [p_0, y]$
   $\delta'([p_0, y], 1) = [p_1, n]$, $\lambda'([p_0, y]) = y$
5. $\delta'([p_1, n], 0) = [p_0, n]$
   $\delta'([p_1, n], 1) = [\delta(p_1, 1), \lambda(p_1, 1)] = [p_1, y]$
   $\lambda'([p_1, n]) = n$

11/4

6. $\delta'([p_1, y], 0) = [p_0, n]$

$\delta'([p_1, y], 1) = [p_1, y]$

$\lambda'([p_1, y]) = y$

Note that $[q_0, n]$ & $[q_0, y]$

$[p_0, n]$ $[p_0, y]$

$[p_1, n]$ $[p_1, y]$ have same behaviour.

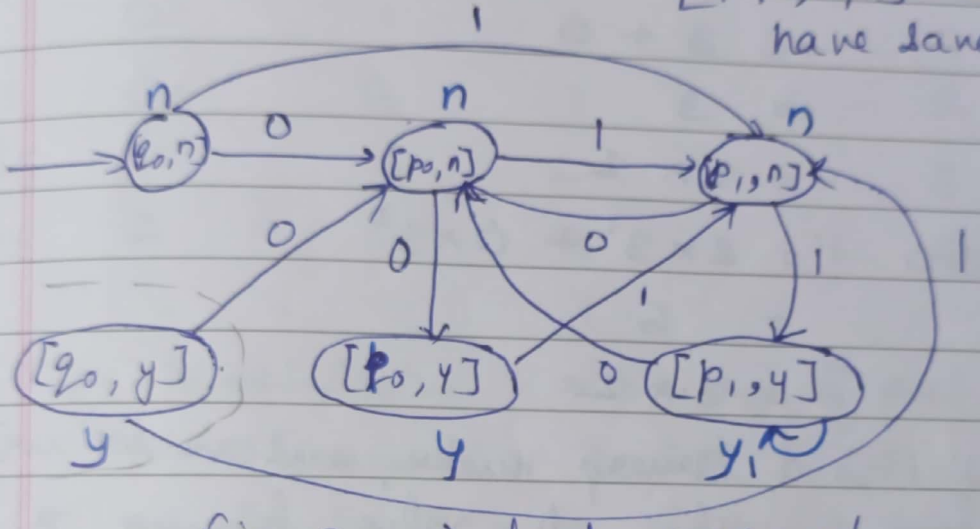


fig Equivalent moore m/c

Relabel:

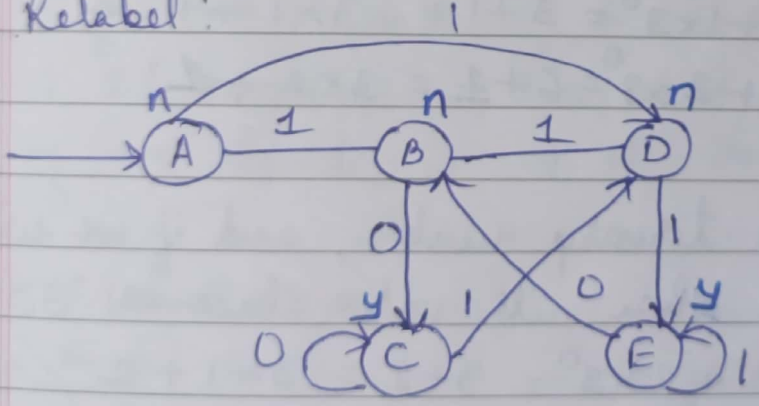


Fig: minimized moore m/c

Mealy & moore m/c

1) Have No Final state
2) Produce o/p from an I/P String
3) No - non-determinism

Pg. 52

Q. For i/p from $(0+1+2)^*$ print the residue modulo 5 of the input treated as a ternary (base 3, with digits 0, 1, 2

Soln Some properties of ternary numbers:

1. If i is a ternary number, and if we write 0 after it then it become 3i

Ex 1. $0 = 1 \times 3^1 + 0 \times 3^0$

$= 3 + 0$

$= 3$

$= 3 \times 1$

Illy 2. $0 = 2 \times 3^1 + 0 \times 3^0$

$= 6$

$= 3 \times 2$

2. If i is a ternary number, and we we write after it then its value becomes $3i+1$

Ex $1.1 = 1 \times 3^1 + 1 \times 3^0 = 3 + 1 = 3 \times 1 + 1$

$2.1 = 2 \times 3^1 + 2 \times 3^0 = 6 + 1 = 3 \times 2 + 1$

3. If i is a ternary number, and if we write 2 after it, then its value becomes $3i+2$

Ex. $1.2 = 1 \times 3^1 + 2 \times 3^0 = 3 + 2 = 3 \times 1 + 2$

$2.2 = 2 \times 3^1 + 3 \times 3^0 = 6 + 2 = 3 \times 2 + 2$

As we are dividing a ternary number by 5 we have five different values for the remainders 0, 1, 2, 3, and 4.

The remainder values are expressed in decimal (base 10) number format as usual

Table ($\lambda: Q \to \Delta$)

Moore m/c. $Q = \{q_0, q_1, q_2, q_3, q_4\}$

Q	q_0	q_1	q_2	q_3	q_4
Δ	0	1	2	3	4

Depending on the remainder in the previous state, if the next digit is 0, 1, or 2 then the next remainder value can be obtained.

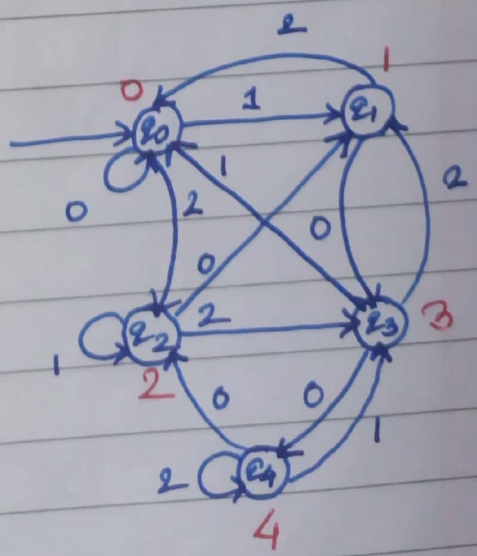
R	0	1	2	3	4
3R mod 5	0	3	1	4	2
3R+1 mod 5	1	4	2	0	3
3R+2 mod 5	2	0	3	1	4

If remainder is R = 3, and if next digit 0 then
3.0 (ternary) = 3 × 3 = 9 (decimal)
If we divide 9 by 5, the remainder is 4

Ily if R = 1 and if 2 is next digit then
1.2 (ternary) = 3 × 1 + 2 = 5 (decimal)
If we divide 5 by 5, the remainder is 0

STF ($\delta: Q \times \Sigma \to Q$)

Q \ Σ	0	1	2
q_0	q_0	q_1	q_2
q_1	q_3	q_4	q_0
q_2	q_1	q_2	q_3
q_3	q_4	q_0	q_1
q_4	q_2	q_3	q_4



3.8 DFA Minimization

Q \ Σ	0	1
a	b	f
b	g	c
c	a	c
d	c	g
e	h	f
f	c	g
g	g	e
h	g	c

Draw a table and put a x for all combinations of final and non-final state. ie. combinations of c with all other states.

b							
c	x	x					
d			x				
e	—		x				
f			x	—			
g			x				
h	—		x			x	
	a	b	c	d	e	f	g

Start with last column i.e. g, h

$\delta(g,0) = g$ $\quad \delta(g,1) = e$

& $\delta(h,0) = g$ $\quad \delta(h,1) = c$ $\Big\rangle$ (e, c)

for pair (e, c) there is already an 'x' marked i.e the two state are not equivalent
∴ put x for the pair g, h

(d, f)

$\delta(d, 0) = \delta(f, 0) = c$

$\delta(d, 1) = \delta(f, 1) = g$

As both are equal, do not put x for (d, f) as they are equivalent.

(b, h)

$\delta(b, 0) = \delta(h, 0) = g$

$\delta(b, 1) = \delta(h, 1) = c$

∴ As both are equal, do not put x for ~~(g, c)~~ (b, h)

(a, e)

$\delta(a, 0) = b$ $\quad \delta(a, 1) = f$ (b, h) is equivalent

$\delta(e, 0) = h$ $\quad \delta(e, 1) = f$

∴ a & e are equivalent hence do not put 'x' for (a, e)

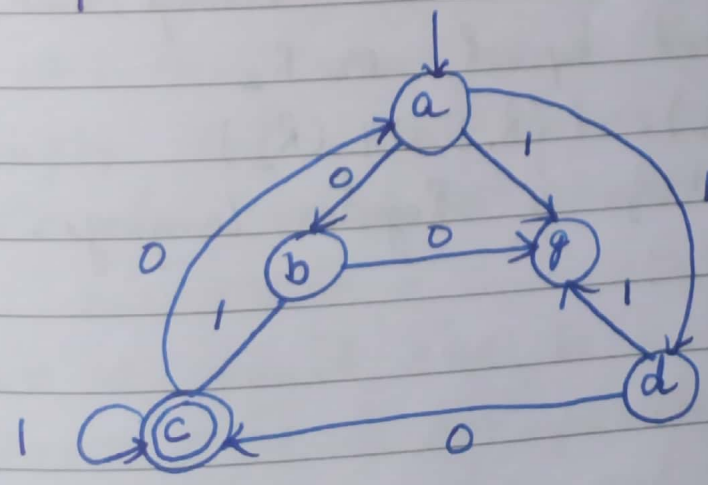
Table :- Reduce Transition table.

Q \ Σ	0	1
a	b	d.
b	g	c
c	a	c
d	c	g
g	g	a.

$d \equiv f$

$b \equiv h$

$a \equiv e$



Minimized DFA

Regular sets and their closure properties

A regular set or a regular language is a language accepted by some FA that can be represented by r.e. Regular sets are thus closed under the same operations that are used to construct r.e. $+, \cdot, *$.

Def^n The sets of r.e. over Σ is defined as

1. Every finite set of words over Σ is a regular set.
2. If U & V are regular sets over Σ; then $U \cup V$ & $U \cdot V$ are regular sets.
3. If S is regular set over Σ then S^* is regular.

Closure Properties of Regular sets.

3.1. Regular languages are closed under union

Proof: Let L_1 & L_2 be two r.l.

$L_1 = L(R_1)$ & $L_2 = L(R_2)$

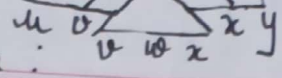
Now $R_1 + R_2$ is a r.e.

$\therefore L(R_1 + R_2)$ is a regular language.

& $R_1 + R_2$ denotes all the strings that are either denoted by R_1 or R_2

$L(R_1 + R_2) = L(R_1) \cup L(R_2)$

Hence $L_1 \cup L_2$ is Regular Language.



Pumping Lemma:

It is an essential property of r.l. It was first articulated by Y. Bar Hillel, Micha A. Perles & Eli Shamir in 1961. It is useful tool for disproving the regularity of given language.

Ex. Prove that $L = \{0^{i^2} \mid i \text{ is an integer } i \geq 1\}$ which consist of all strings of 0's whose length is a perfect square is non-regular.

Soln Given $i \geq 1$

For $i = 1$, $0^{i^2} = 0^{1^2} = 0$ length $= 1^2$

$i = 2$, $0^{i^2} = 0^{2^2} = 0000$ length $= 2^2$

$i = 3$, $0^{i^2} = 0^{3^2} = 000000000$ length $= 3^2$

Length of each string is perfect square

1. Assume that L is regular language. Let n be the constant of pumping lemma.
2. Choose a sufficiently large string z such that $z = 0^{l^2}$ for some large $l > 0$,

$|z| = l^2 \geq n$.

$z = u\,v\,w$

3. $u\,v^i\,w$ for all $i \geq 0$ is in L.

$|v| \geq 1$ which means that v cannot be empty and must contain one or more symbols. Consider when v contains single symbol

$\therefore z = u\,v\,w = 0^{l^2}$ ie. 0 in z is perfect square $u\,v^2\,w$ should also be a member of L. However this is not possible, as v contains only a single symbol, and adding one to the perfect square length would not always

yield perfect square length.
Thus pumping v would yield strings with non-square lengths.
$\therefore uv^2w$ is not a member of L.
This contradicts our assumption that L is regular.
Let v = 0000 (40's) or 000000000 (9 0's)
v^2 = 000...0 (8 0's) or 00...0 (18 0's)
the length does not remain a perfect square.

Let v = 000 (3 0's) v^2 = 000000 (6 0's)
which does not have a perfect square length.
$\therefore$ $L = \{0^{i^2} \mid i \geq 1\}$ is non-regular.

Ex. $L = \{a^n b^n \mid n > 0\}$. Prove that the language is non-regular.

$L = \{a^m b^m \mid m > 0\}$

Let $z = a^l b^l$ for some large $l \geq 0$.

$|z| = 2l \geq n$

$\therefore z = uvw$

As per the pumping lemma, every uv^iw for all $i \geq 0$ is in L. $|v| \geq 1$

$z = uvw = a^l b^l$

Let v = ab or aabb

$\therefore v^2$ = abab or v^2 = aabbaabb

a's are followed by b's and not vice versa

$\therefore uv^2w$ is not a member of L

Hence uv^2. L is non-regular.

The pumping length depends on the language, not any specific FSM. Long enough that some FSM will have a cycle.

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

$\therefore L = \{ \epsilon, 00, 11, 0101, 0000, 1010, 1111 \ldots \}$

1. Assume that L is a regular language. Let n be constant of pumping Lema
2. $z = xx$ where $x = 0^p 1^q$

   $|z| = 2(p+q) \geq n$

   Write $z = uvw$
3. As per the pumping lema, every string uv^iw for all $i \geq 0$ is in L.

   $|v| \geq 1$.

   $z = uvw = xx = 0^p 1^q 0^p 1^q$

   $z = 0^p 1^q 0^p 1^q$ after pumping $v = 0$ once. we get $z_1 = 0^p 1^q 00 0^p 1^q$ which cannot be represented as concatenation of two equal sub-strings

   $\therefore uv^2w$ is not a member of L as it modifies the string from xx to $xvvx$ rather than $xvxv$. This contradicts our assumption that L is regular.

1) Assume that A is Regular
2) It has to have a Pumping Length
3) All string longer than P can be pumped $|s| \geq$
4) Now find a string 's' in A such that $|s| \geq$

$L = \{ a^n b^n \mid n > 0 \}$

$P = 7$ — a a a a a a a a b b b b b b b — $|uv| \leq p$

Case 1 — u | v | w — $6 < 7$

Case 2 — a a a a a a a b b, b b b b b b b — u | v | w

Case 3 — a a a a a a a b b b b b b b — $|uv| = 9$ — u | v | w

Case 1 — $uv^i w = uv^2 w$

a a a a a a a a a a a b b b b b b b

$11 \neq 7$

Case 2 — $uv^2 w =$ a a a a a a a b b b b b b b b b b

$7 \neq 11$

Case 3 — $uv^2 w =$ a a a a a a a a b b a a b b b b b b b

$|uv| < P$ & $P = 7$

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

$P = 7$ — 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 — u | v | w

$uv^i w$ — 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 $\notin A$

$|v| > 0$

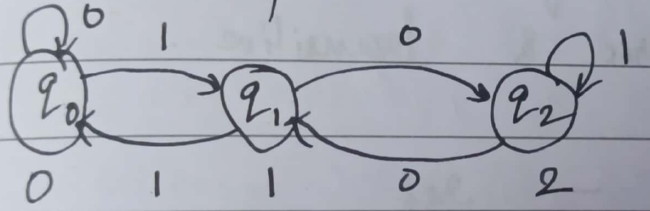
$|uv| \leq P = 7$

Myhill Nerode Theorem:

This theorem provides a necessary and sufficient cond^n for a language to be regular.

Given a r.l. L and a pair of strings x and y let the equivalence classes of ~~strings~~ and the no. of states in the smallest DFA recognizing L is equal to rel^n R_L be defined as $x R_L y$, iff, for all z in Σ^*, xz is in L exactly when 'yz' is in L. the R_L is of finite index.

Ex Consider binary no.s divisible by 3.



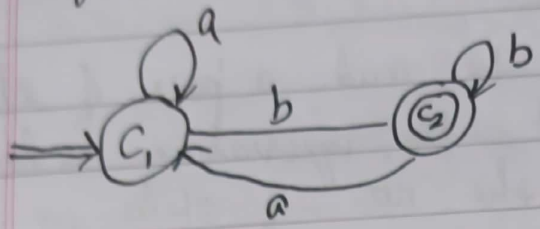
There are three equivalent classes based on remainder values 0, 1, 2. The minimal state DFA that can be obtained for such a language also has 3 states Hence the language is regular.

Proof If there are n states, then we partition the set of strings into n subsets, where S_i is the set of string that, when given as input to automaton M causes it to end in state i.

For a language L, defined over an alphabet Σ, L partitions Σ^* into distinct classes

If L generates finite no. of classes then L is regular.

Q. Let L be set of all string ending with b. defined over $\Sigma = \{a, b\}$



C_1 = set of all string ending in a
C_2 = set of all strings ending in b

Since these are finite classes L is regular.

NPTEL

Rel^n is equivalence when rel^n is reflexive symmetric & transitive.

$x R x$ — ref
$x R_y = y R x$ - symmetric
$x R_y \wedge y R_z = x R_z$ - Transitive.

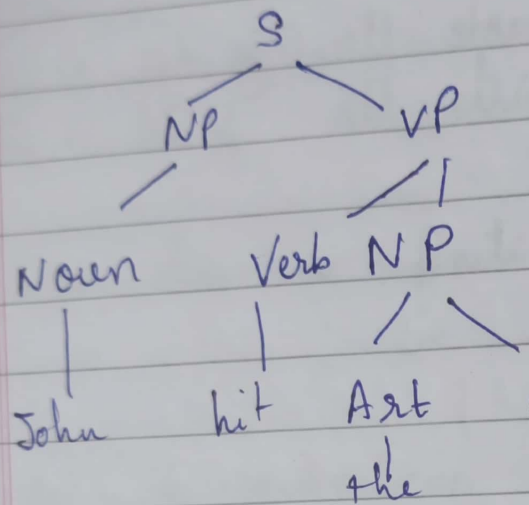
q_0 q_1 . . . .

q_{n-1}

Atmost 'n' equivalence classes.
no. of equivalence classes will be atmost the no. of states.

Grammar

S → NP & VP
NP → Noun | Art Noun
VP → Verb & NP
Noun → John | ball
Verb → hit | Ricked
Art → a | the



2

F:- Any CFL without ε is generated by a gramm in which all productions are of the form.

CNF : A → BC A → a

Here A, B, C are variables & a is terminal

Ex 1 S → aSa | bSb | a | b | aa | bb

Soln S → ASA | BSB | a | b | AA | BB
A → a
B → b

Introduce R_1 & R_2
S → A R_1 | B R_2
R_1 → SA
R_2 → SB
A → a
B → b

Ex. 2 $S \to bA \mid aB$

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

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

Sol^n Introduce R_1 & R_2

$R_1 = a$

$R_2 = b$

$S \to R_2A \mid R_1B$

$A \to R_2AA \mid R_1S \mid a$

$B \to R_1BB \mid R_2S \mid b$

$R_1 \to a$

$R_2 \to b$

Introduce R_3 & R_4

$S \to R_2A \mid R_1B$

$A \to R_2R_3 \mid R_1S$

$B \to R_1R_4 \mid R_2S \mid b$

$R_1 \to a$

$R_2 \to b$

$R_3 \to AA$

$R_4 \to BB$

Ex. 3 $S \to ABA$

$A \to aA \mid \in$

$B \to bB \mid \in$

$L(G') = L(G) - \in$

$\therefore S \to ABA \mid AB \mid BA \mid AA \mid A \mid B$

$A \to aA \mid a$

$B \to bB \mid b$

Sol^n $S \to AR_1 \mid AB \mid BA \mid AA \mid R_2A \mid a \mid R_3B \mid \in$

$A \to R_2A \mid a$

$B \to R_3B \mid b$

$R_1 \to BA$

$R_2 \to a$ $R_3 \to b$

Griebach Normal Form:

$A \to a\alpha$, $A \to a$

$a \to T$

$A \Rightarrow$ Non Terminal

α is a (possibly empty) string containing only non-terminals.

(1) Convert the following grammar to GNF

$S \to ABA \mid AB \mid BA \mid AA \mid A \mid B$

$A \to aA \mid a$

$B \to bB \mid b$

$S \to A_1$

$A \to A_2$

$B \to A_3$

Replace the leading A's by their right hand size productions

$S \to aABA \mid aBA \mid aAB \mid bBA \mid bA \mid aAA \mid aA \mid aA \mid a \mid bB \mid b$

$A \to aA \mid a$

$B \to bB \mid b$.

(2) $S \to CA \mid BB$

$B \to b \mid SB$

$C \to b$

$A \to a$

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

We get

$A_1 \to A_2A_3 \mid A_4A_4$

$A_4 \to b \mid A_1A_4$

$A_2 \to b$

$A_3 \to a$

If Production is of the form $A_i \to A_jx$ then $i < j$ & should never be $i \geq j$

$\therefore A_4 \to A_1A_4$ $\therefore$ Replace A_1

$\therefore A_4 \to b \mid A_2 A_3 A_4 \mid A_4 A_4 A_4$

$i \quad < \quad j$

Replace A_2 — is problematic

$A_4 \to b \mid b A_3 A_4 \mid A_4 A_4 A_4$

↓

Left Recursion

To remove Left Recursion introduce a new variable B_4

$B_4 \to A_4 A_4 B_4 \mid A_4 A_4$

Write A_4 with B_4

$\therefore A_4 \to b \mid b A_3 A_4 \mid b B_4 \mid b A_3 A_4 B_4$

Now we get-

$A_1 \to A_2 A_3 \mid A_4 A_4$

$A_4 \to b \mid b A_3 A_4 \mid b B_4 \mid b A_3 A_4 B_4$

$B_4 \to A_4 A_4 B_4 \mid A_4 A_4$

$A_2 \to b$

$A_3 \to a$

A_1 is not in GNF

$A_1 \to b A_3 \mid b A_4 \mid b A_3 A_4 A_4 \mid b B_4 A_4 \mid b A_3 A_4 B_4 A_4$

$A_4 \to b \mid b A_3 A_4 \mid b B_4 \mid b A_3 A_4 B_4$

$B_4 \to b A_4 B_4 \mid b A_3 A_4 A_4 B_4 \mid b B_4 A_4 B_4 \mid b A_3 A_4 B_4 A_4 B_4 \mid b A_4 \mid b A_3 A_4 A_4 \mid b B_4 A_4 \mid b A_3 A_4 B_4 A_4$

$A_2 \to b$

$A_3 \to a$

Construct a CFG for $\{a^{2n} bc \mid n \geq 1\}$

The grammar G.

$S \rightarrow A\ b\ c$

$A \rightarrow a\ a\ A \mid a\ a$ for $n \geq 1$

The no. of a's generated is always even.

$L = \{a^n b^m c^k \mid n = m$ or $m \leq k$; $n \geq 0$, $m \geq 0, k \geq 0\}$

n. (a) $n = m$

(b) $m \leq k$

(a) $S_1 \rightarrow AC$

$A \rightarrow aAb \mid \epsilon$ $n = m$

$C \rightarrow cC \mid \epsilon$ generates any no of c's

b) $S_2 \rightarrow BDE$

$B \rightarrow aB \mid \epsilon$ generates any no. of b's

$D \rightarrow bDc \mid \epsilon$ for $m = k$

$E \rightarrow cE \mid \epsilon$ for $m < k$

$S \rightarrow S_1 \mid S_2$

Q.3. Remove useless symbol.

$G = \{(S, A), (1, 0), P, S\}$

$S \rightarrow 10 \mid 0S1 \mid 1S0 \mid A \mid SS$

Soln $S \Rightarrow A \Rightarrow ?$

$S \rightarrow 10 \mid 0S1 \mid 1S0 \mid SS$

Remove useless symbol

Q.4. $G = \{(S, A, B), (a), P, S\}$

P is $S \rightarrow AB \mid a$

$A \rightarrow a$

Drop B

Drop A also

$S \rightarrow a$.

Unit Production

$A \Rightarrow x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow B$

If $B \rightarrow \alpha_1 \mid \alpha_2 \mid \cdots$ then create productions

$A \rightarrow \alpha_1 \mid \alpha_2 \mid \cdots$

Ex ① $G = \{(A, B), (a, b), P, A\}$

$A \rightarrow B$

$B \rightarrow a \mid b$

Soln $A \rightarrow a \mid b$

Ex ② $S \rightarrow A \mid bb$

$A \rightarrow B \mid b$

$B \rightarrow S \mid a$

Soln Add $A \rightarrow S \mid a$

$\therefore S \rightarrow A \mid bb$

$A \rightarrow S \mid a \mid b$

Ex (3) $S \to a \mid Xb \mid aYa \mid b \mid aa$

$X \to Y$

$Y \to b \mid X$

Sol? Delete $X \to Y$

$S \to a \mid Yb \mid aYa \mid b \mid aa$

$Y \to b$

CFL

$L(G) = \{ w \mid w \in T^*, \text{ is derivable from } S \}$

41

x. (1) Let G be a CFG

$S \to aSb \mid ab$

Strings derived from S

$S \Rightarrow ab$

$S \Rightarrow aSb$

$\Rightarrow aaSbb$

$\Rightarrow aaabbb$

$\therefore L(G) = \{ a, b, aabb, aaabbb, aaaabbbb, \}$

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

.2 Find CFL

$S \to aB \mid bA$

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

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

ol? Derive the string abab, abba, aabb, bbaaba ----

$\therefore L(G) = \{ x \mid x$ containing equal number of a's & b's.

Find CFG

Q. 1 $L = \{ a^i b^j c^k \mid i = j + k \}$

same ↑ ↑ same → $k = 0$

$S \to a\,S\,c \mid A \mid \epsilon$ → $i = j = k = 0$

$A \to a\,A\,b \mid \epsilon$

↓ equal ↓ equal

S ⇒ aSc

→ aaScc

→ aaAbcc

→ aaaAbcc

→ aaaabbcc

Q. 2 $L = \{ a^{m+n} b^m c^n \mid n, m \geq 0 \}$

Sol^n $S \to a\,S\,c \mid A \mid \epsilon$

$A \to a\,A\,b \mid \epsilon$

Q. 3 $L = \{ 0^{m,n} 0^{m+n} \mid m, n \geq 0 \}$

$S \to 0\,S\,0 \mid A \mid \epsilon$

$A \to 1\,S\,0 \mid \epsilon$

Q. 4 $L = \{ a^n b^m a^n \mid n \geq 0, m \geq 1 \}$

$S \to a\,S\,a \mid B$

$B \to b\,B \mid b$

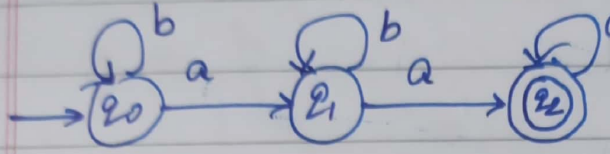
~~Ans~~ $L = \{ a^n b^m c^k \mid n = m$ or $m \leq k, n \geq 0, m \geq 0, k \geq 0 \}$

A right linear grammar is one in which each production is of the following form $X \to aY$, $X \to a$

Regular Grammar

Q. Generate a Regular grammar
$L = \{ w \mid w$ contains atleast two 'a' symbol$\}$

:- $L = \{ aa, abaa, abaaab, baab, baa, aab ---- \}$



q_0 $S \to bS$; Any No. of b

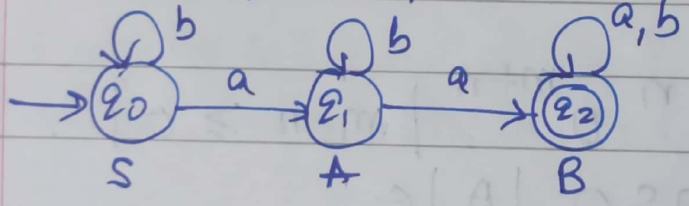
q_1 $S \to aA$; Atleast one a symbol.

$A \to bA$; any no. of b followed by string that contains at least one a symbol

$A \to aB$

To produce all strings are $\{a, b\}^*$

$B \to \in / \lambda \mid aB \mid bB$



∴ Gramar is

$S \to bS$

$S \to aA$

$A \to bA$

$A \to aB$

$B \to \lambda$

$B \to aB$

$B \to bB$

This is right-linear grammar

Q. Consider the foll CFG that defines a very small subset of valid English sentences.
Terminal symbol have foll. interpretations
a = "a", t = "the", d = "dog", r = "ran".
V = { S, Article, Noun, Subject, Verb }
T = { a, d, r, t }
S = S

P: S → Subject Verb
Subject → Article Noun
Article → a | t
Noun → d
Verb → r

Q. what set of strings does this grammar produ
Ans. { adr, tdr }

Q. Using the interpretation of terminal symbols, what English sentences do these string represent?
A A Dog ran, The dog ran

Type 2 grammar :- CFG
$A \to \alpha$ where A is a non-terminal &
α is sentential form i.e. $\alpha \in (V \cup T)^*$
& α may also be equal to ϵ.
The start symbol of the grammar can also appear on RHS
NPDA can accept the whole class of CFL
G = {(S), (a, b), P, S}
P: S → a Sa | bSb | a | b

* Write the grammar for generating the variable name, which can be given by the r.e (letter) (letter / digit)* which means, a letter followed by any number of letters or digits.

:- Let L be the NT to produce letters & D be the NT to produce digits.

∴ G is

$S \to LS'$

$S' \to LS' \mid DS' \mid \epsilon$

$L \to a \mid b \mid \dots \mid z \mid A \mid B \mid C \dots \mid Z$

$D \to 0 \mid 1 \mid 2 \dots \mid 9$

Show the derivation of flag 12.

Ex. Describe the CFL generated by G

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

where P is

$S \to aSa \mid bSb \mid a \mid b \mid \epsilon$

Solve Derive strings for aa, bb, bab, bbb

$L = \{\epsilon, a, b, aa, bb, aaa, aba, bbb, bab$

∴ Language consisting of all palindrome over $\Sigma = \{a, b\}$

Ex. Write a grammar for the language over $\Sigma = \{a, b\}$ containing at least one occurrence of 'aa'.

$S \to XaaX$

$X \to aX \mid bX \mid \epsilon$

$(a+b)^* aa (a+b)^*$

Chomsky Heirarchy

The class of phase structure grammars is very large. However, imposing certain constraints on production rules, different classes of phrase structure grammar can be obtained. The heirarchy thus obtained is called Chomsky Hierarchy by Noam Chomsky in 1956.

Chomsky suggest four different classes of phrase structure grammar:

1. Type 0 — Unrestricted
2. Type 1 — Context-Sensitive
3. Type 2 — CFG
4. Type 3 — Regular grammar.

217

Regular grammar.

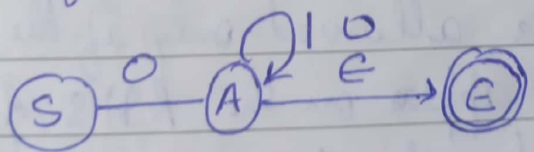
$A \to wB$ or $A \to w$

Ex The language $0(10)^*$ is.

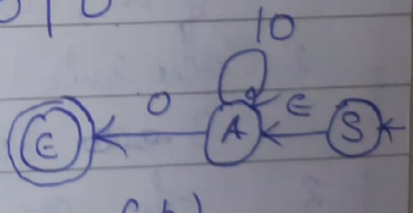
$S \to 0A$

$A \to 10A \mid \epsilon$

& Left Linear $S \to S10 \mid 0$



(a)



(b)

Restriction

i) The LHS should contain only 1 NT.

ii) The RHS can contain at most one NT symbol, which is allowed to appear as the rightmost symbol or the leftmost symbol.

The languages generated using this grammar is called as regular language (RL). RL can be expressed by r.e

Unrestricted grammar

1. Type 0: There are no restrictions on the productions of a grammar of this type

$\alpha \to \beta$; $\alpha \neq \epsilon$

α & β are sentential form i.e any combination of terminals & NT

$\alpha, \beta \in (V \cup T)^*$ & $\alpha \neq \epsilon$

Semi-Thue grammar.

The language is recognized by TM.

Also known as recursively enumerable lang.

Ex $G = [\{A, B, C\}, \{a, b, c\}, P, A\}$

P is

$A \to AB$

$AB \to Bc$

$B \to A cD$

$Ac \to abc$

$D \to \epsilon$

2. Type 1: Context Sensitive Restrictions are

1. For each production of the form $\alpha \to \beta$, the length of β is atleast as much as the length of α, except for $S \to \epsilon$
2. The rule $S \to \epsilon$ is allowed only if the start symbol S does not appear on the RHS of any production
3. The grammar has the form $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$ where the replacement of a NT A by β is allowed only if α_1 precedes A and α_2 succeeds A. α_1 & α_2 may not be empty.

$G = \{(A, B, C), (a, b), P, A\}$

$A \to AB$

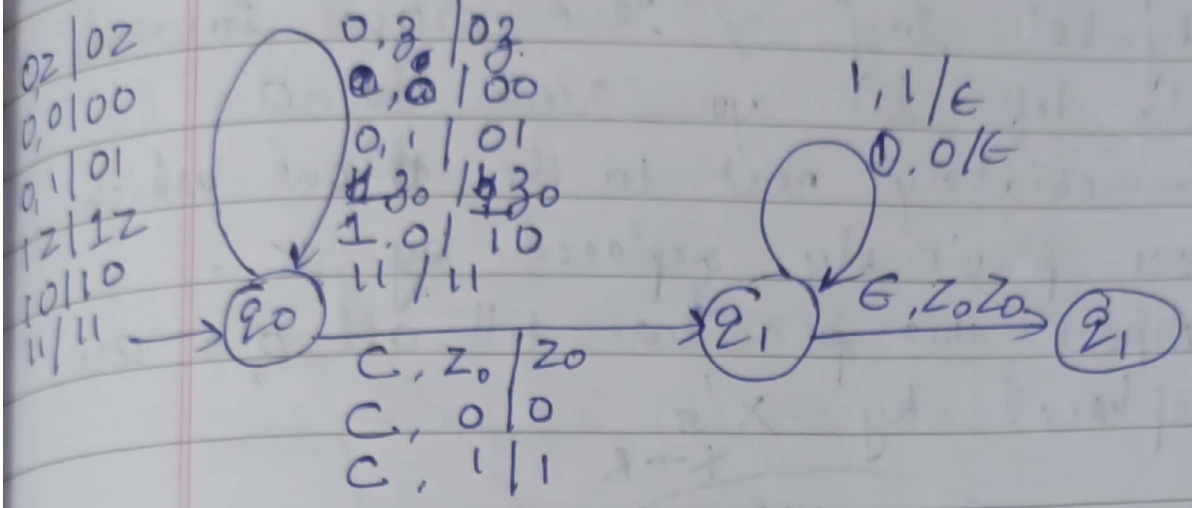
$AB \to$

PDA

$L = \{ w c w^R \mid w \in \{0,1\}^* \}$

0	1	1	c	1	1	0

w $\quad$ w^R

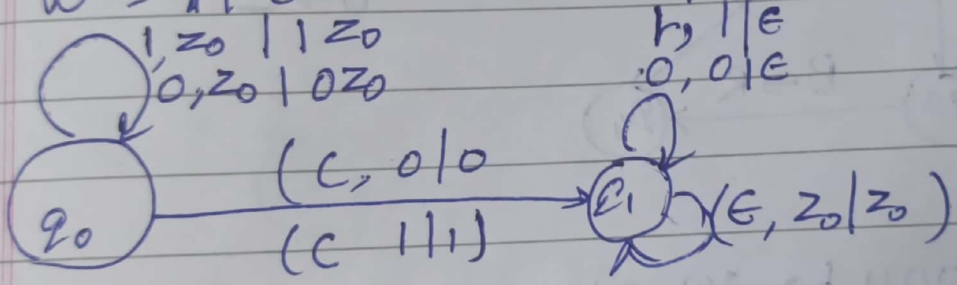


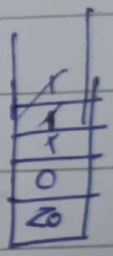
abb

OR

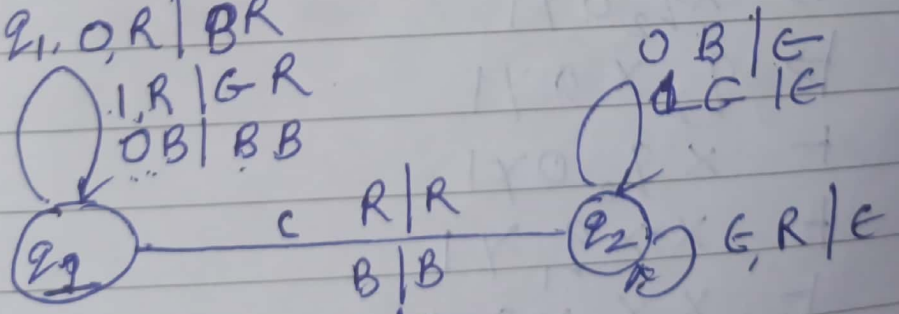
$w = 011$

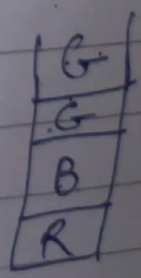
$w^R = 110$





OR/BR
OB/BB
OG/BG
1R/GR
1B/GB
1G/GG



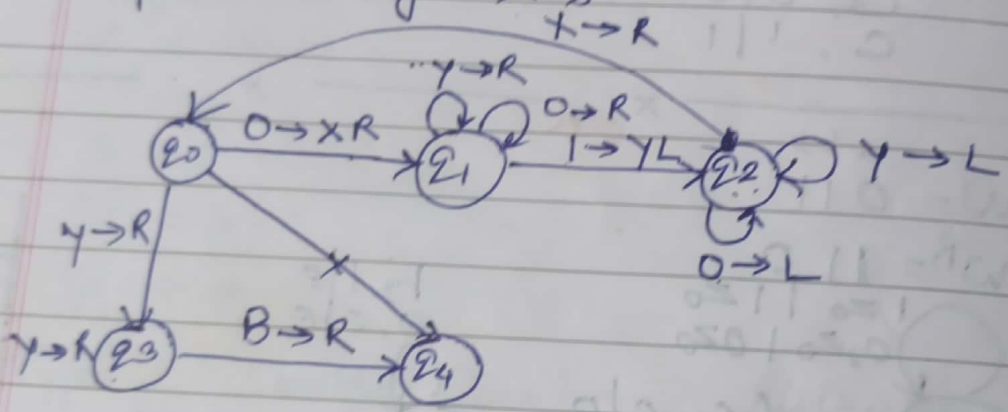


Design a TM for $0^n 1^n$ for $n \geq 0$

Algo: 1) Replace '0' by 'X' and move towards right till you reach first 1.

2) Replace this first symbol by some other symbol say 'Y' and move towards the left till you reach the 0 immediately next to the one which was previously replaced by X.

3) Repeat this procedure till all 0's are replaced by X's



$q_0 0011 \vdash X q_1 011$

$\vdash X 0 q_1 11$

$\vdash X q_2 0 Y 1$

$\vdash q_2 X 0 Y 1$

$\vdash X q_0 0 Y 1$

$\vdash X X q_1 Y 1$

$\vdash X X Y q_1 1$

$\vdash X X q_2 Y Y$

$\vdash X q_2 X Y Y$

$\vdash X X q_0 Y Y$

$\vdash X X Y q_3 Y$

$\vdash X X Y Y q_3$

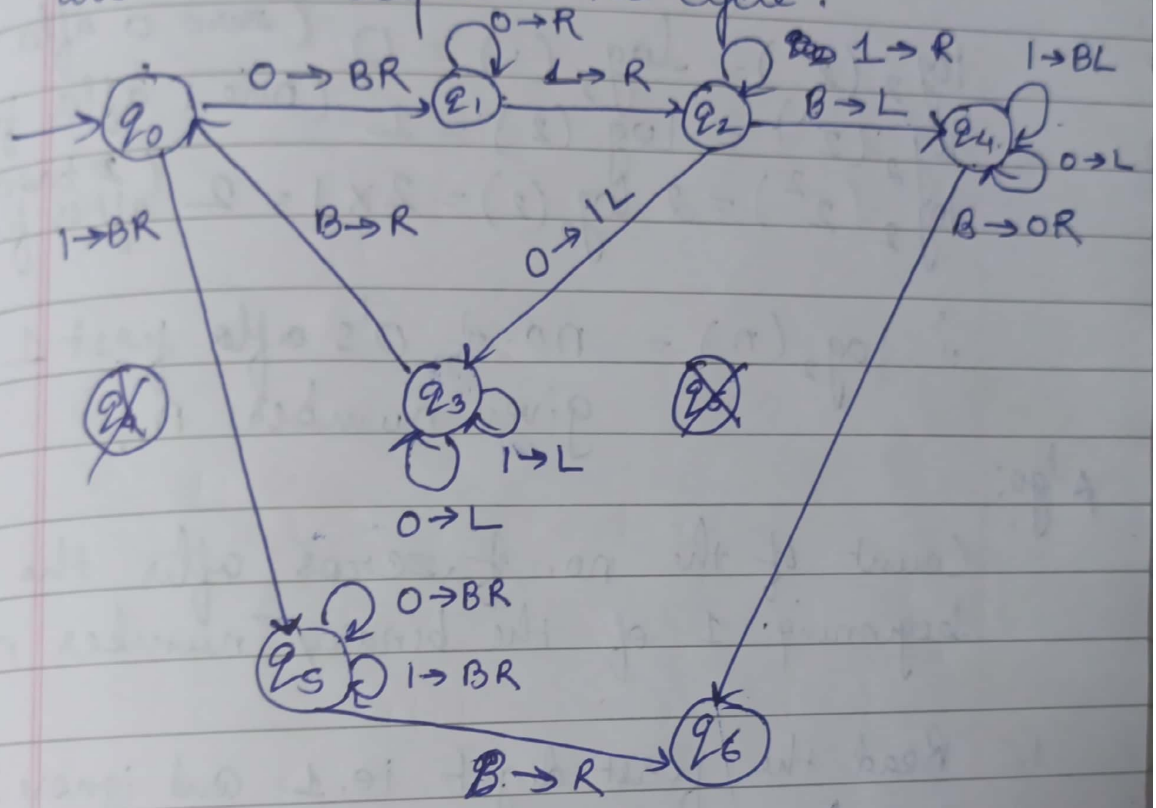
$\vdash X X Y Y B q_4$

Design a TM for $m-n$, $m \geq n$ & 0 if $m < n$.

Algo 1) Start with $0^m 1 0^n$ & halt with 0^{m-n}.

2) Repeatedly replace leading 0 by blank then search right for a 1 followed by a 0 and change 0 to 1.

3) M moves left until it encounters a blank and then repeats the cycle.



$q_0\,0010 \vdash B q_1 010$

$\vdash B0 q_1 10$

$\vdash B01 q_2 0$

$\vdash B0 q_3 11$

$\vdash q_3 B011$

$\vdash B q_0 11$

$\vdash BB q_1 11$

$\vdash BB1 q_2 1$

$\vdash BB11 q_2$

$\vdash BB1 q_4 1$

$\vdash BB q_4 1$

$\vdash B q_4 B$

3 Design a TM to find $\log_2 n$ when n is binary no. & a perfect power of 2
(n binary)

Soln $2^0 = 1 = 1$
$2^1 = 2 = 10$
$2^2 = 4 = 100$
$2^3 = 8 = 10000$

$\log_2(2^0) = \log_2(1) = 0$ (zero 0 after 1)
$\log_2(2^1) = \log_2(2) = 1$ (one 1 after first 1)
$\log_2(2^2) = 2\log_2(2) = 2 \times 1 = 2$ (2 two 0's after first 1)

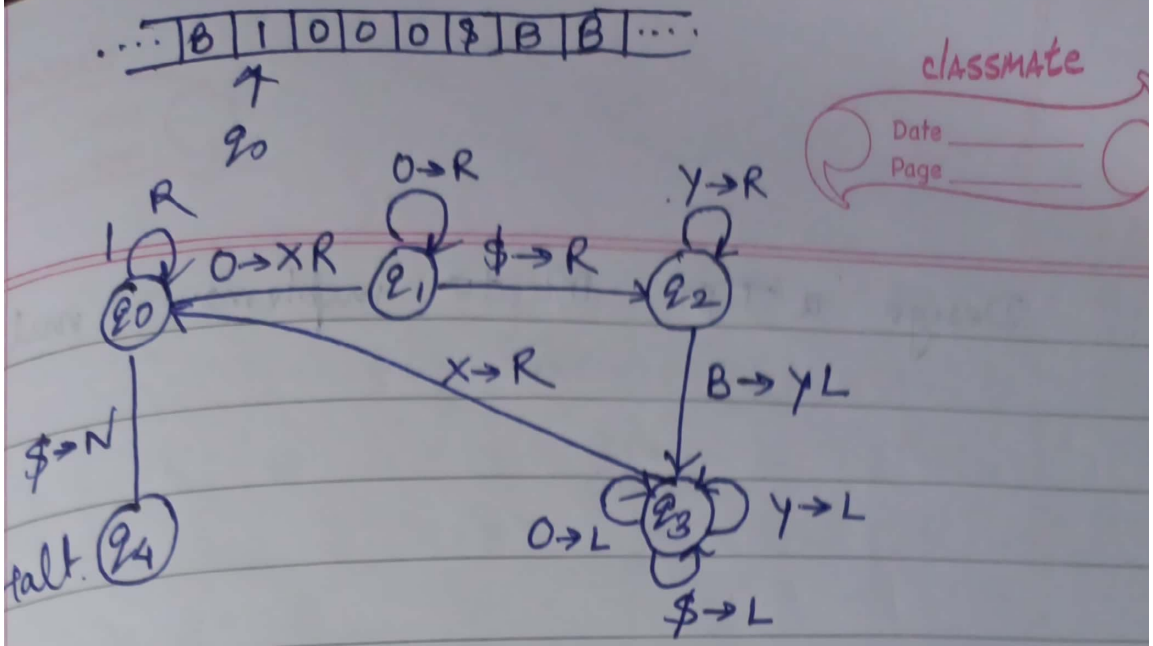
$\therefore \log_2(n) =$ no. of 0's after first 1 in given number n.

Algo:

Count of the no. of zeros after the beginning 1 of the binary number n

1. Read the first digit ie. 1 and ignore it.
2. Read the following 0 from the string that follows 1, replace it by a symbol 'X' & write a symbol 'Y' after the right end marker
3. Repeat the procedure till all the 0's that follow 1 get replaced by X's
4. The no. of Y's written after the right-end marker gives the required value of $\log_2(n)$.

The I/P string is in binary format and the resultant value is in unary format



Q.No.3

Sentence → Subject predicate
Subject → noun phrase
noun phrase → <adjective> <noun phrase>
noun phrase → <noun>

predicate → verb phrase
verb phrase → <verb> <adverbial phrase>
<verb>
adverbial phrase → <preposition> <noun phrase>
<adjective> <noun phrase>

CFG's are useful for describing arithmetic expressions, with arbitrary nesting of balanced parenthesis & block structure in pgm languages.

CFG is a finite set of variables (non-terminals or syntatic categories) each of which represent a language.

The language represented by the variables are described recursively in terms of each other & primitive symbols called terminals.

The rules relating the variables are called productions.

→ Rule Tree, Parse tree, Syntax Tree

Derivation tree is a graphical representation of how the sentence has been derived using grammar.

Ex 1

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

$P = S \Rightarrow aSb$

$S \Rightarrow ab$

$S \Rightarrow aSb$

$\Rightarrow aaSbb$

$\Rightarrow a^3 S b^3$

$\Rightarrow a^{n-1} S b^{n-1}$

$\Rightarrow a^n b^n$.

The strings in $L(G)$ are $a^n b^n$ for $n \geq 1$.

$\therefore L(G) = \{a^n b^n \mid n \geq 1\}$.

No. 83

2 Consider the following CFG

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

Ps $S \Rightarrow aAS \mid a$ (1) (2)

$A \Rightarrow SbA \mid SS \mid ba$ (3) (4) (5)

Leftmost Derivation

$S \Rightarrow \underline{a}AS$ (1)

$\Rightarrow aS\underline{b}AS$ 3

$\Rightarrow aab\underline{A}S$ 2

$\Rightarrow aabba\underline{S}$ 5

$\Rightarrow aabbaa$ 2

Right most

$S \Rightarrow aA\underline{S}$ 1

$\Rightarrow a\underline{A}a$ 2

$\Rightarrow aSb\underline{A}a$ 3

$\Rightarrow aSbbaa$ 5

$\Rightarrow aabbaa$ 2

Ex. 3 $S \to aSb \mid ab$

Generate Leftmost & Rightmost Derivation for aaabbb

$S \to aSb$ Rule 1

$\Rightarrow aaSbb$ 1

$\Rightarrow aaabbb$ 2

87. A CFG G such that some word has two parse trees is said to be Ambigious.

Let $G = (V, T, P, S)$ be a gramar. A symbol X is useful if there is a derivation $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$ for some α, β and w, where w is in T^*. Otherwise X is useless.

(1) Some terminal string must be derivable from X &

(2) X must occur in some string derivable from S.

A CFL is the language generated by a CFG G

$L(G) = \{ w \mid w \in T^*$, & is derivable from Start Symbol $S \}$

G Productions

A production of the form $A \to G$, where A is a Non Terminal is known as G production.

If ∈ is a member of L(G) for a given grammar G then we cannot eliminate all ∈ productions from G, whereas if ∈ is not in L(G), then we can remove all the ∈ productions.

PDA to CFG (V, T, P, S)

① Construction of set of Non Terminal
$V = (\{S\} \cup [q, z, q'] \mid q, q' \in Q, z \in \Gamma)$

② S Production
$S \to [q_0, z_0, q] \to (q', \in)$ $q \in Q$

③ For Pop operation
$\delta(q, a, z) \to (q', \in)$

$[q, z, q'] \to a$

④ For Push Operation & No operation
$\delta(q, a, z) \to (q_1, z_1 z_2 \cdots z_n)$

$[q, z\ q'] \to a[q_1, z_1, q_2][q_2, z_2, q_3]$
$\cdots\cdots\cdots [q_n, z_n, q']$

$\delta(q_0, 0, z_0) = \{q_0, Xz_0\}$

$[q_0\ z_0\ q_0] \to 0[q_0, X, q_0][q_0, z_0, q_0]$
$[q_0\ z_0\ q_1] \to 0[q_0\ X, q_1][q_1, z_0, q_0]$

PDA's are used to accept the language generated by CFG.

Theorem: A language is CFG, iff some PDA recognizes it.

Proof 1. Given a CFG, show how to construct a PDA that recognizes it

Part. 2. Given a PDA, show how to construct a CFG that recognizes the same language.

Ex. Given a grammar

$S \to \overset{①}{BS} \mid \overset{②}{A}$

$A \to \overset{③}{0A} \mid \overset{④}{\in}$

$B \to BB\overset{⑤}{1} \mid \overset{⑥}{2}$. Build a PDA.

Sol?: Build a leftmost derivation

$\to S$

$\cdot\ BS$ Using rule 1

$\underline{B}B1S$ Using rule 5

$2\underline{B}1S$ 6

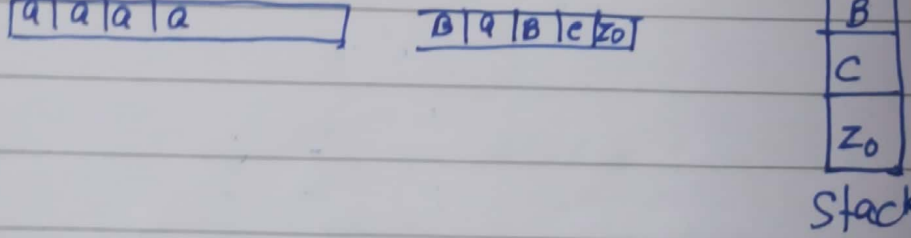
$22\mid\underline{B}$ 6

$221A$ 2

$221\in$ 4

221 ,

General Form - $aaaaBaBCB$



Left Most Derivation $S \to aaaa\ B\,a\,B\,C \to$

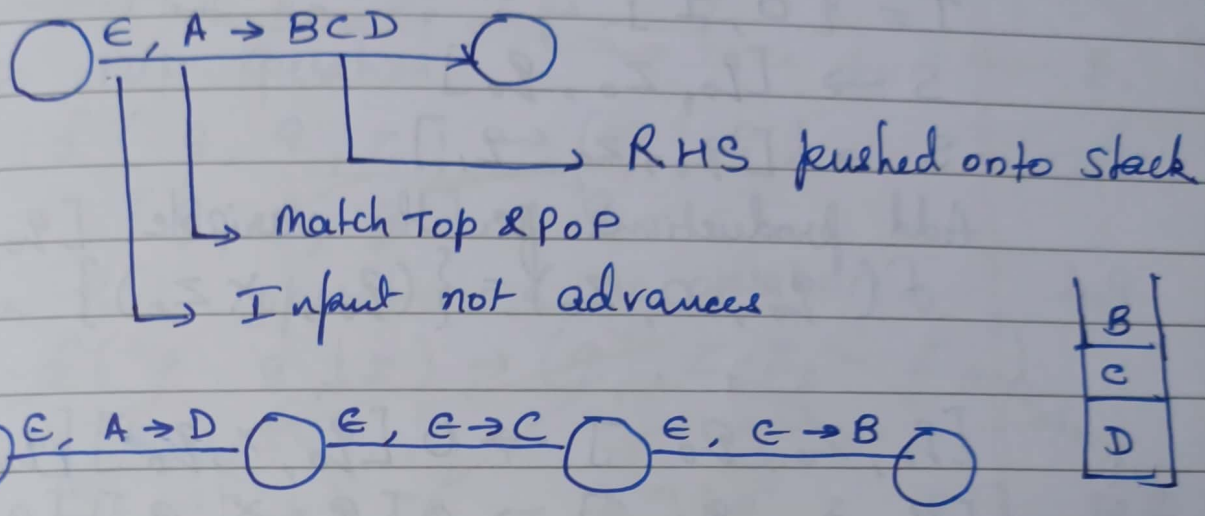
At each step expand left most derivation

$B \to ASA \times BA$, Pop B & Push RHS

$\to aaaa\ A\,S\,A\,\times\,B\,A\,a\,B\,C$

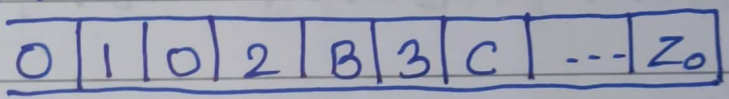
→ Match Stack Top to a Rule
→ Pop stack
→ Push RHS of Rule onto the stack

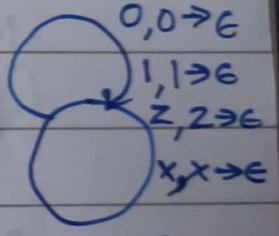
Rule A → BCD



Rule A → 0102B3C

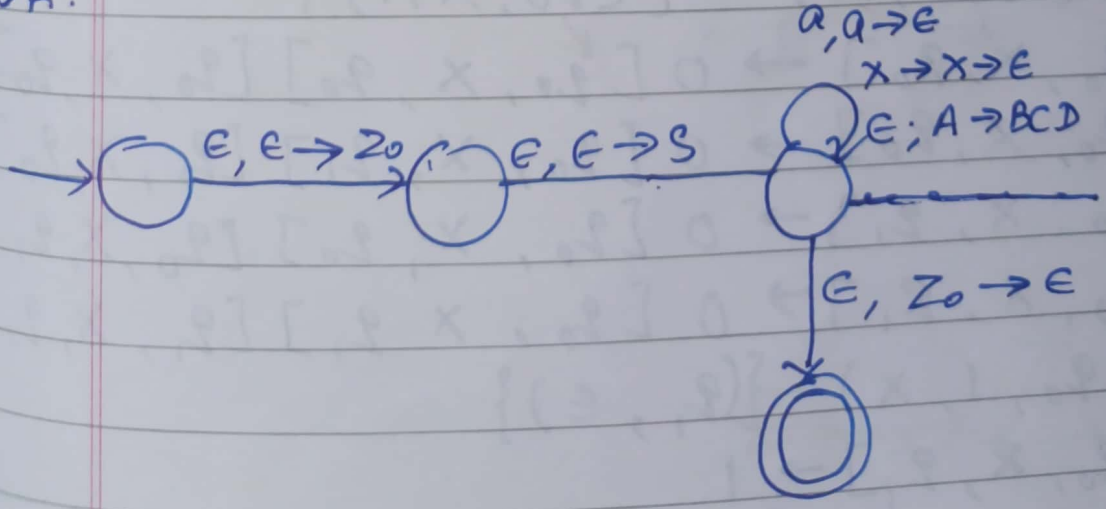
Pop A & Push LHS





If we have a terminal symbol just advance and do not push anything.

PDA.



Ex. 5.3. CFG $G = (V, T, P, S)$.

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

$T = \{ 0, 1 \}$

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

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

Add productions for the variable $[q_0, Z_0, q_0]$

$\delta(q_0, 0, Z_0) = \{(q_0, X Z_0)\}$

$[q_0, Z_0, q_0] \to 0 [q_0, X, q_0] [q_0, Z_0, q_0]$

$[q_0, Z_0, q_0] \to 0 [q_0, X, q_1] [q_1, Z_0, q_0]$

Next, the production for $[q_0, Z_0, q_1]$ are

$\delta(q_0, 0, Z_0) = \{(q_0, X Z_0)\}$

$[q_0, Z_0, q_1] \to 0 [q_0, X, q_0] [q_0, Z_0, q_1]$

$[q_0, Z_0, q_1] \to 0 [q_0, X, q_1] [q_1, Z_0, q_1]$

Next

$\delta(q_0, 0, X) = \{(q_0, XX)\}$

$[q_0, X, q_0] \to 0 [q_0, X, q_0] [q_0, X, q_0]$

$[q_0, X, q_0] \to 0 [q_0, X, q_1] [q_1, X, q_0]$

$[q_0, X, q_1] \to 0 [q_0, X, q_0] [q_0, X, q_1]$

$[q_0, X, q_1] \to 0 [q_0, X, q_1] [q_1, X, q_1]$

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

$[q_0, X, q_1] \to 1$

$\delta(q_1, \epsilon, Z_0) = \{(q_1, \epsilon)\}$ Pop

$[q_1, Z_0, q_1] \to \epsilon$

$\delta(q_1, \epsilon, X) = \{(q_1, \epsilon)\}$ $[q_1, X, q_1] \to \epsilon$

$\delta(q_1, 1, X) = \{(q_1, \epsilon)\}$ $[q_1, X, q_1] \to 1$

PDA TO CFG

$F = \phi$ No Final state

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

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

Steps 1. Construction of set of Non-Terminal

$V = \{S\} \cup \{[q, Z, q'] \mid q, q' \in Q, Z \in T\}$

2. i) S-Production → Initial state, $q' = q_0, q_1$
   → Initial stack Symbol

$S \rightarrow [q_0, Z_0, q], \quad q \in Q$

→ once q_0 & then q_1

ii) For Pop operation

$\delta(q, a, Z) \rightarrow (q', \in)$

$[q, Z, q] \rightarrow a$

iii) for Push operation & No operation

$\delta(q, a, Z) \rightarrow (q_1, Z_1 Z_2 \cdots Z_m)$

New state generated

$[q, Z, q'] \rightarrow a[q_1, Z_1, q_2][q_2, Z_2, q_3] \cdots [q_m, Z_m, q]$

q_0, q_1

where $q', q_2, q_3 \cdots q_m \in Q$