Date:.....

DFA

formal def of DFA

Q - set of states
Σ - alphabet
δ - transition function
q_0 - stand state
F - set of final states

M accepts string $w = w_1 \cdots w_n$ each $w_i \in \Sigma$ if there is a sequence of states $r_0 \cdots r_n \in Q$ where

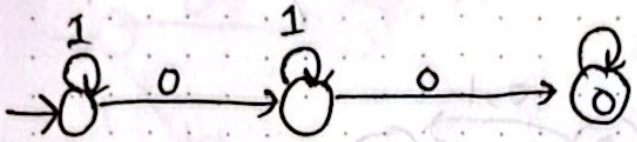
$r_0 = q_0$
$r_i = \delta(r_{i-1}, w)$
$r_n \in F$

⊛ If there exists a DFA for a language then it's called Regular language.

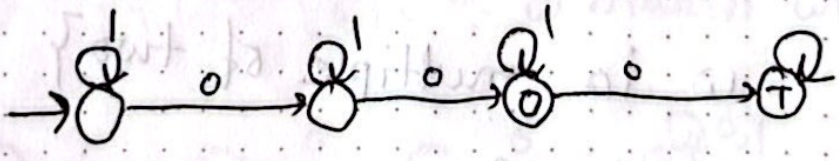
Date:.....

BRAC UNIVERSITY
Inspiring Excellence

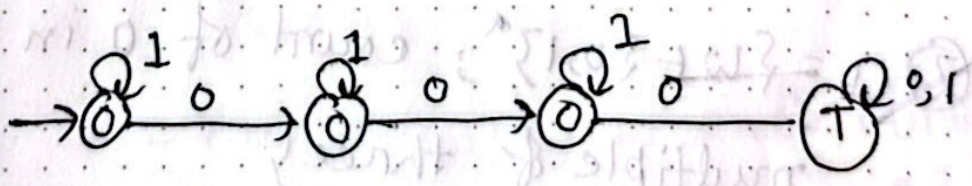
① $L = \{ w \in \{0,1\}^* ;$ w contains at least two 0's$\}$



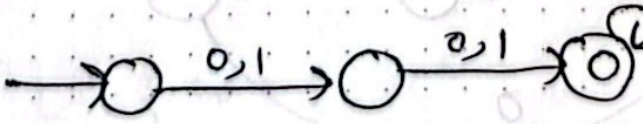
② $L = \{ w \in \{0,1\}^* ;$ w contains exactly two 0's$\}$



③ $L = \{ w \in \{0,1\}^* ,$ w contains at most two 0's$\}$

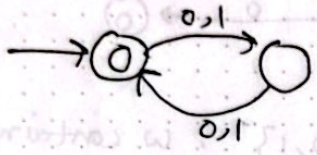


④ {length of w is at least two}

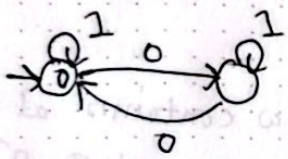


Date:.....

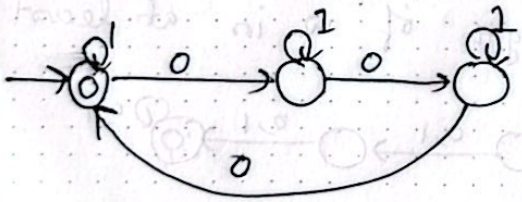
⑤ $L = \{w \in \{0,1\}^* ;$ length of w in even / multiple of 2$\}$



⑥ $L = \{w \in \{0,1\}^*$ count of 0 in w is multiple of two$\}$

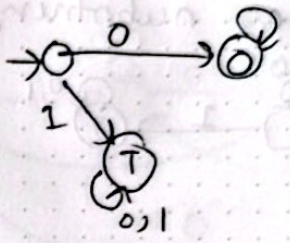


⑦ $L = \{w \in \{0,1\}^* ;$ count of 0 in w is multiple of three$\}$

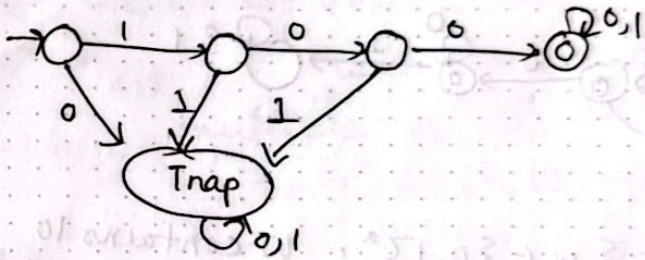


Date:.....

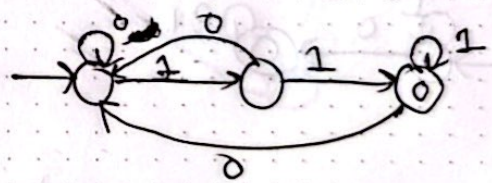
⑧ $L = \{w \in \{0,1\}^* ;$ w starts with 0$\}$



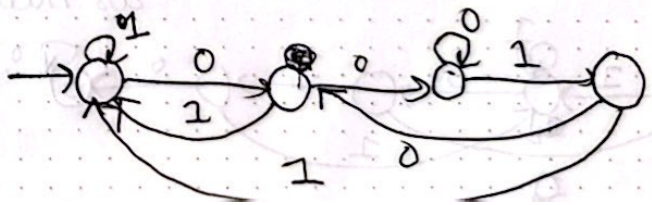
⑨ $L = \{w \in \{0,1\}^* ;$ w starts with 100$\}$



⑩ $L = \{w \in \{0,1\}^* ;$ w ends with 11$\}$

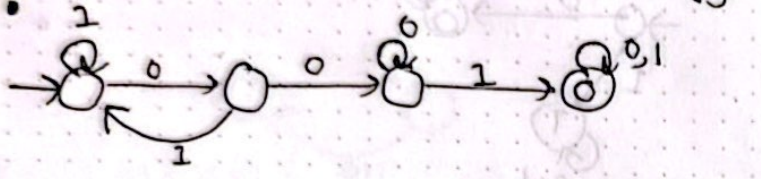


⑪ $L = \{w \in \{0,1\}^* ;$ w ends with 101$\}$

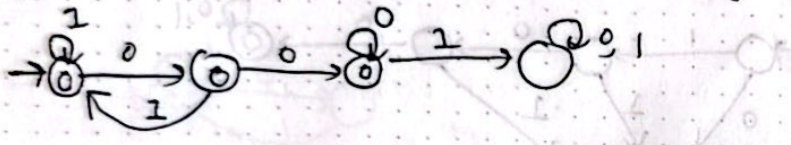


Date:.....

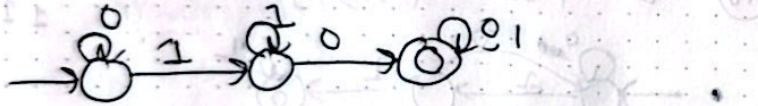
(12) $L = \{ w \in \{0,1\}^* ;$ w contains 001 as substring$\}$



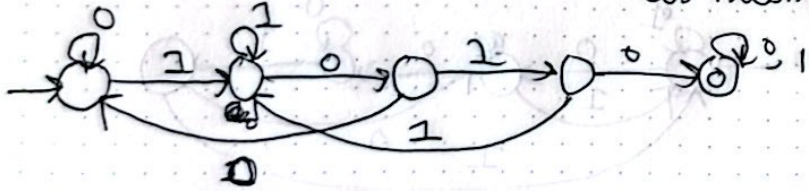
(13) $L = \{ w \in \{0,1\}^* ;$ w doesn't contain 001 as substring$\}$



(14) $L = \{ w \in \{0,1\}^* ;$ w contains 10 as substring$\}$

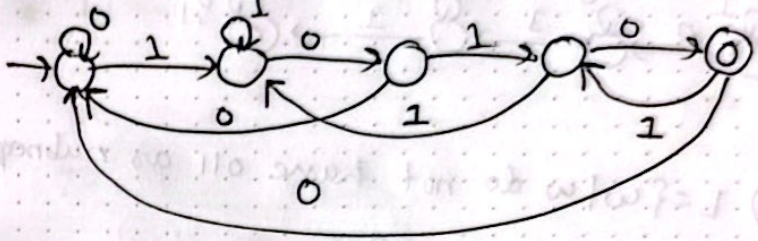


(15) $L = \{ w \in \{0,1\}^* ;$ w contains 1010 as substring$\}$



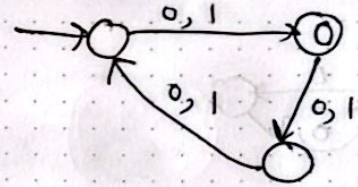
Date:.....

(16) $L = \{ w \in \{0,1\}^* ;$ w ends with 1010$\}$



(17) $L = \{ w \in \{0,1\}^* ;$ the length of w is one more than multiple of 3$\}$

1, 4, 7, 10, 13 - - - ..



3 reminders

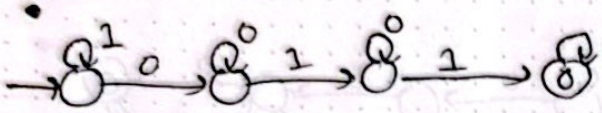
% = 0

% = 1

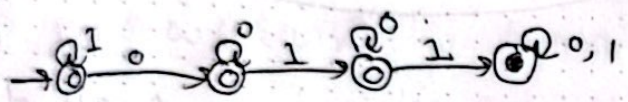
% = 2

Date:.....

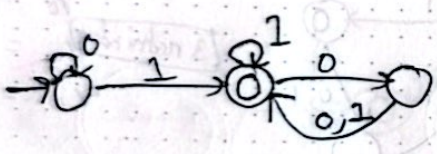
(18) L={w | w have 011 as subsequence}



(19) L={w | w do not have 011 as subsequence}

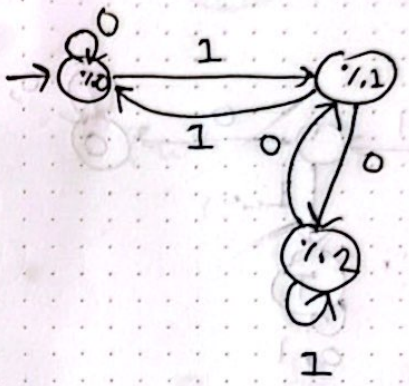


(20) L={w | w contains at least one 1 and an even number of 0's follow the last 1}

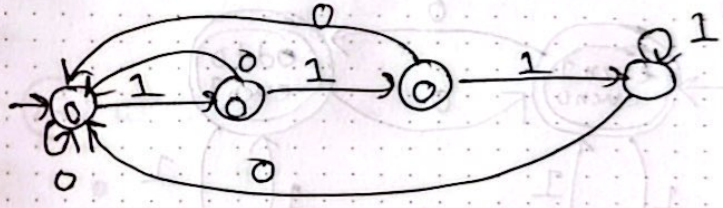


Date:.....

(21) L={w | w when interpreted as a binary number is divisible by 3}

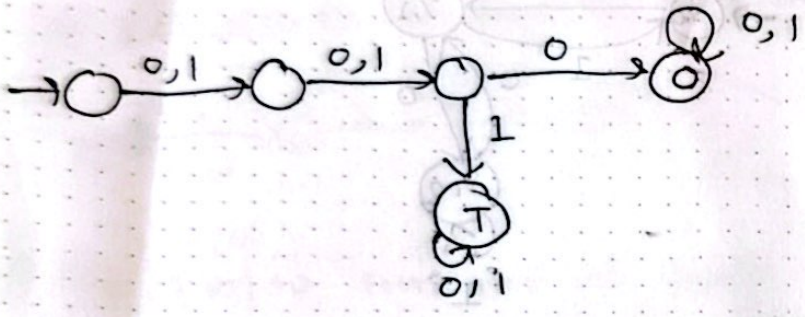


(22) strings that end with at most two 1's [!(at least 3 1's)]

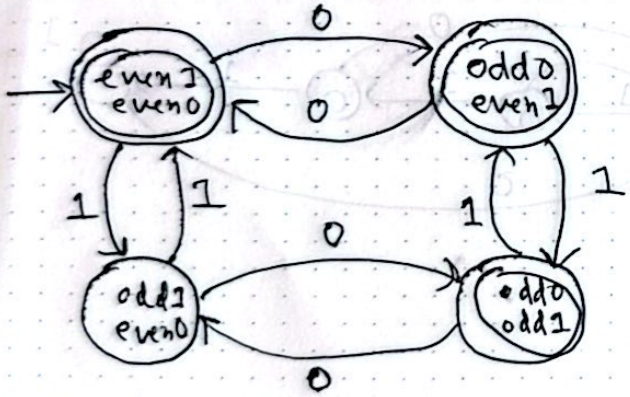


Date:.....

(23) String that has 0 at third position

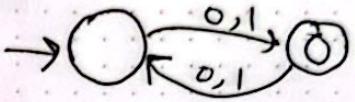


(24) {w | w have an even number of 1's or odd number of 0's}

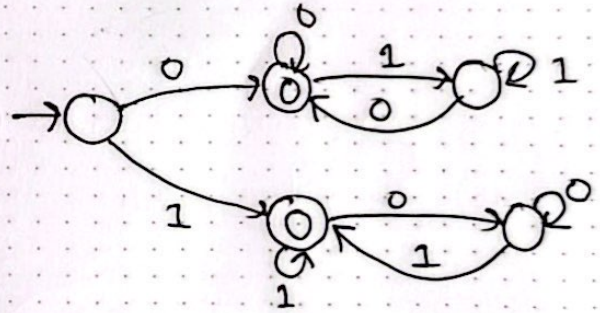


Date:.....

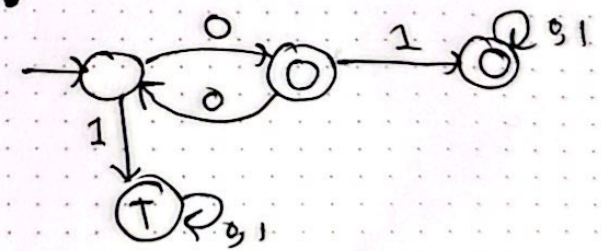
(25) length of w is odd



(26) L = {w ∈ {0,1}* ; w starts & ends with same symbol}



(27) w starts with odd number of 0's



BRAC UNIVERSITY
Inspiring Excellence

Date:.....

Regular Operations

Operations that can be applied to some regular languages and the resulting language still be regular ~~tag~~ language are called regular operations.

Three regular operations:

- Union : $\{x \mid x \in A \text{ or } x \in B\} = A \cup B$
- Concatenation : $\{xy \mid x \in A \text{ and } y \in B\} = A \cdot B$
- Star : $\{x_1 x_2 \cdots x_k \mid k \geq 0 \text{ \& each } x_i \in A\} = A^*$

[Regular languages is closed under the regular operations.]

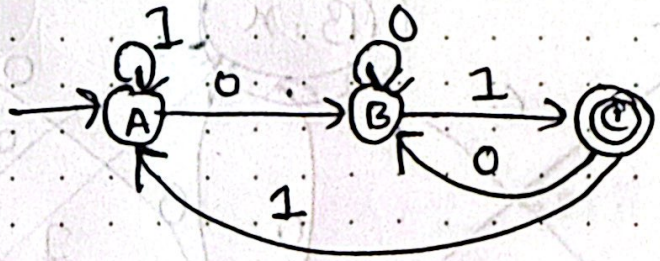
Date:...../...../.....

BRAC UNIVERSITY
Inspiring Excellence

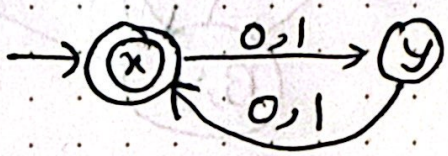
Example $A = \{w \in \{0,1\}^* : w \text{ ends with } 01\}$

$B = \{w \in \{0,1\}^* : \text{length of } w \text{ is even}\}$

$M_A =$



$M_B =$



$L = A \cup B$

~~We can M~~

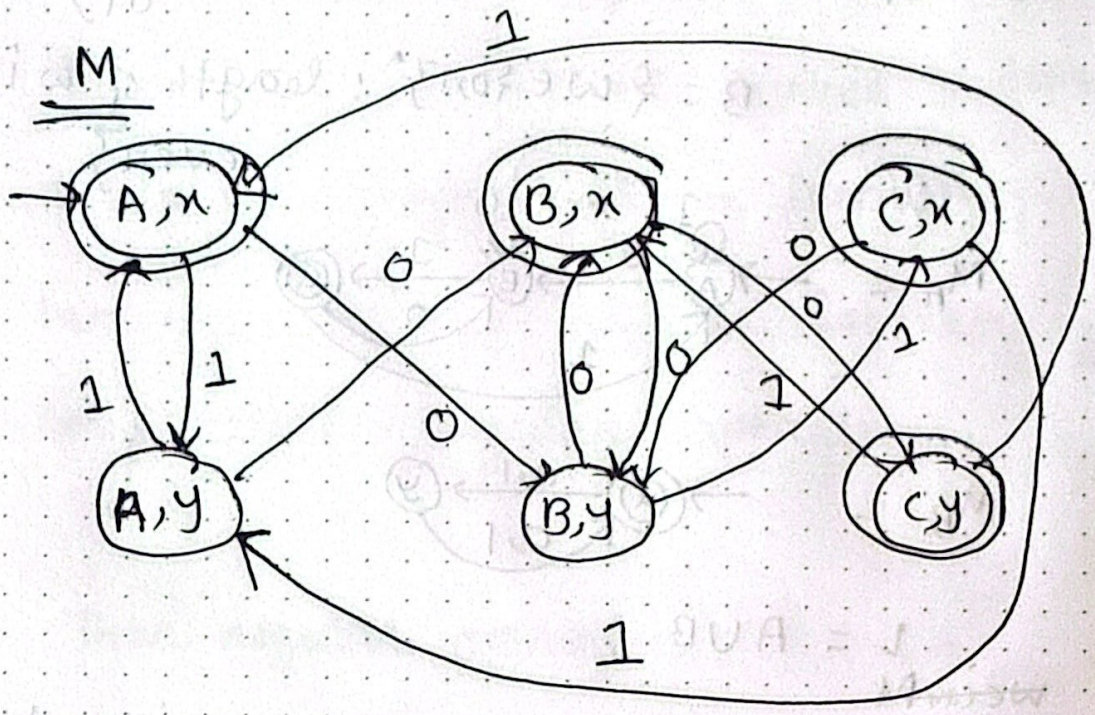
we can make M that recognized L by using cross product.

BRAC UNIVERSITY
Inspiring Excellence

Date:.....

no. of states in M = total states in M_A X total states in M_B.

M



Start state: That contains both the ~~start~~ state of M_A & M_B
(A, x)

Accepting state → That contains either or both the final state of M_A & M_B. (A,x), (B,x), (C,x), (C,y))

Date:.....

BRAC UNIVERSITY
Inspiring Excellence

if A∩B then accepting state will be only state which contains both the final state of M_A & M_B

(C,X)

but not case will be same.

Concatenation: In case of it, instead of constructing automata M to accept it's input if either M_1 or M_2 accept, it must accept if input can be broken into two piece, where M_1 accepts the first piece & M_2 accept the second piece. Problem with this is that M doesn't know where to break it's input.

Solution to this is Nondeterminism!!