Lec 2 : Language & its operations

28/1/24

- Alphabet : is a finite set of symbol, fundamental unit

String : constructed using finite seq. of symbols from alphabet

Language : set of strings from alphabet $|w|$, $|Hello| = 5$

Σ — denotes a string

Σ^* — denotes strings that can be generated from Σ

- Operation on Languages :

① Complement of a Language : $\Sigma^* - L$

② Concatenation of languages : $L_1 = \{x, y, z\}$, $L_2 = \{a, b\}$

$L_1L_2 = \{xa, xb, ya, yb, za, zb\}$

$\Rightarrow L_1L_2 = \{xy \mid x \in L_1, y \in L_2\}$

③ Reverse of language : $L^R = \{w^R \mid w \in L\}$

$L = \{\epsilon, a, b, ac, ab, bc\}$

$L^R = \{\epsilon, a, b, ca, ba, cb\}$

④ Kleene closure or star-closure :

$L^* = L^0 \cup L^1 \cup L^2 \cup \dots$

$L^+ = L^1 \cup L^2 \cup \dots$

$L = \{a, ab, ba\}$

$L^0 = \{\epsilon\}$

$L^1 = \{a, ab, ba\}$

$L^2 = L \cdot L = \{aa, aab, aba, aba, abab, abba, baa, baab, baba\}$

- Null language : Language doesn't consist of any strings

formal language : Concern only syntax not semantic i.e. meaning

- Lexicographic strings : Shorter strings proceed than larger strings

Lec 3 : Regular expression

① Primitive Regular expression

Regular Language	Regular expression
ϕ	ϕ
ϵ	ϵ
$\{a\}$	a
$\{b\}$	b

② If r_1 and r_2 are regular expressions representing regular languages L_1 & L_2 then

a) Union : $L_1 \cup L_2 \longrightarrow r_1 + r_2$

b) Kleene closure : $L_1 \rightarrow r_1$

$L_1^* \rightarrow r_1^*$

c) Concatenation : $L_1 L_2 \rightarrow r_1 r_2$

d) (r_1) is a regular expression representing L_1.

using ① & ② , regular expressions can be obtained.

- Let Alphabet = {0,1}

Regular language	Regular expression
{null} {ε}	ε
{0}	0
{1}	1
{0,1}	0+1
{01}	01
{10}	10
{01, 10}	01+10
{null, 0, 00, 000, ...}	0*

some eg. are missing

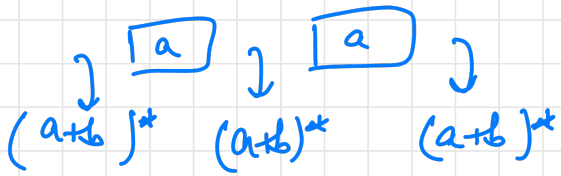
31/1/24

Lec 4 :

eg: Design a regular expression over {a,b} that accept all strings having atleast two a's.

Sol^n $\Sigma = \{a, b\}$

$L = \{ aa, aaa, aab, baa, \ldots \}$



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

eg. RE = ? , substring aa , over $\{a, b\}$

Sol^n

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

eg. $\Sigma = \{a, b\}$, all strings having exactly two a's.

Sol^n $b^* a\ b^* a\ b^*$?

eg. all strings with atmost 2 a's , $\Sigma = (a, b)$

Sol^n $L = \{\varepsilon, a, ab, ba, aa, aab, baa, aba, \ldots \}$?

eg. exhibit the following, (write string upto length 3)

a) $a^* b^* = \{ \ldots \}$

b) $(ab)^* = \{ \ldots \}$

c) $(a+b)^* (ab + ba) = \{ \ldots \}$?

- Application of RE:

Lexical phase of compilation (for identifying the identifier in programming language)

C identifier rule:

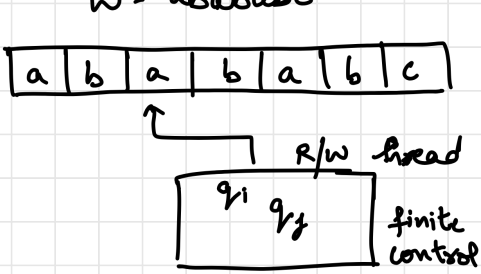
① first symbol can be alphabet or underscore

② Second and onward symbol can be alphabet, digit or underscore

Best the problem is that there can be more than one RE for a particular problem.

Lec 6: Intro to finite automata

- Regular languages can be represented as finite automata.
- Input tape is divided into square.



Initially R/w head points to the leftmost point (square) of the input tape.

- Automata reads symbol from input tape and depending on current state & symbol read, enters into a new state.

Once end of string reaches and the machine is in the final state, we call the string as accepted

If not in the final state after reading input string, the string is rejected

Finite automata

- Deterministic
- Non-deterministic

defined by quintuple

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

Q is a finite non-empty set of states,

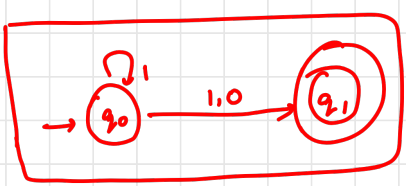
q_0 is an initial state., $q_0 \in Q$

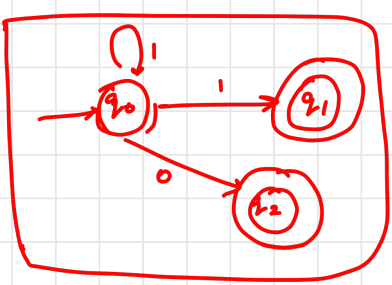
F is set of final states, $F \subseteq Q$

Σ be an alphabet.

δ is a transition f^n.

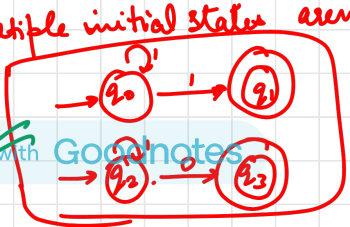
eg: $RE = 1^* (1+0)$

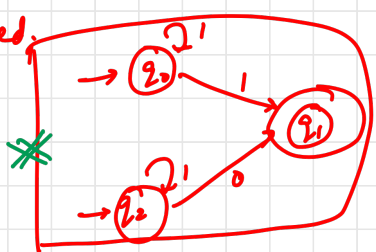




self loop is *.

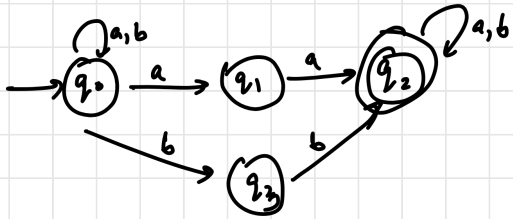
multiple initial states aren't allowed.



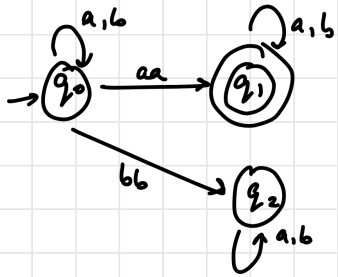


eg: RE = $(a+b)^*(aa+bb)(a+b)^*$

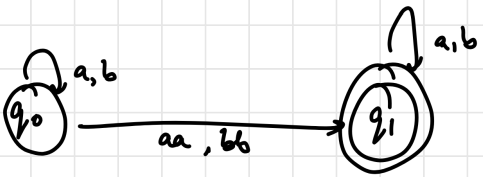
Sol"



or



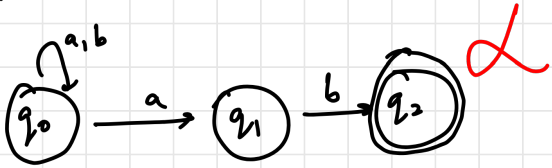
or



1/2/24

eg: $\Sigma = \{a, b\}$ that ends with ab.

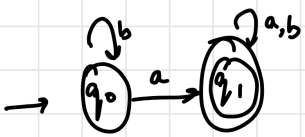
Sol" $(a+b)^*ab$



eg: construct DFA — ① strings containing a.

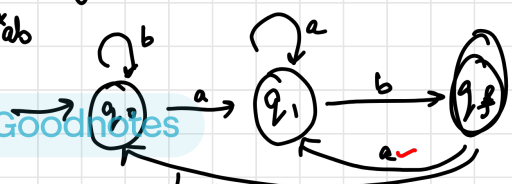
Sol" { a, aa, aaa, ba, ab, abab, ... }

① possibilities bahari padegi



eg: DFA, string ends with ab

Sol" $(a+b)^*ab$



ab, aab, bab, aaab, bbab, baab, abab, aabab, b abab, abbab

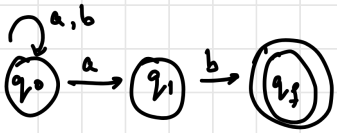
Transition Table : (for the above eg)

	a	b
→ q_0	q_1	q_0
q_1	q_1	q_f
(q_f)	q_1	q_0

- Difference b/w NFA and DFA

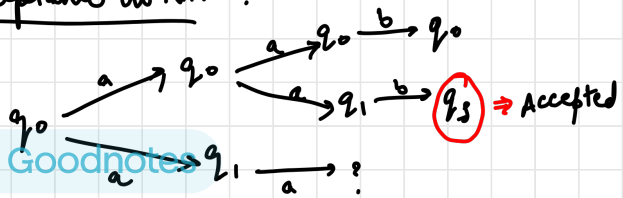
NFA	DFA
$Q \times (\Sigma \cup \{\epsilon\}) \rightarrow P(Q)$	$Q \times \Sigma \rightarrow Q$
Contains null string as an input too	

- examples of NFA :



	a	b
q_0	$\{q_0, q_1\}$	q_0
q_1	-	q_f
q_f	-	-

String acceptance in NFA :



w = aab

a	a	b

↑ q_0 ↑ R/w

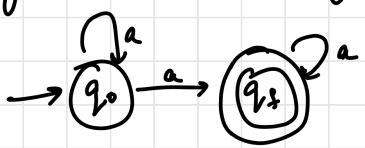
lec 7 : Deterministic finite automata

eg. $\Sigma = \{a\}$, accepts all strings including null string, DFA = ?

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

eg. string over $\{a\}$ with length one or more

Sol^n



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

eg. String over $\{a, b\}$ including null string

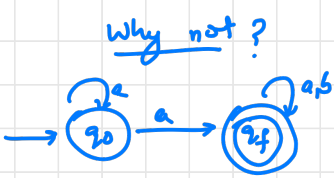
Sol^n $L = \{\epsilon, a, b, ab, ba, \ldots\}$

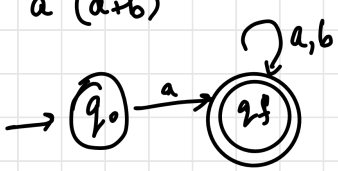
eg. String that start with a, $\Sigma = \{a, b\}$

Sol^n $L = \{a, ab, aab, aaa, aba, abb, \ldots\}$

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

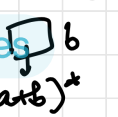
Why not ?



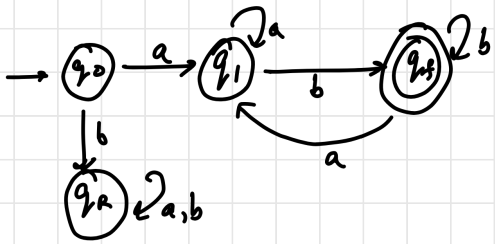


eg. $\Sigma = \{a, b\}$, strings starts with a & end with b

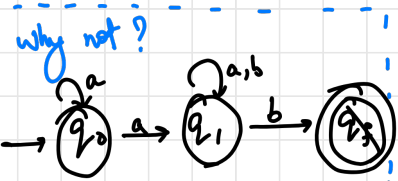
Sol^n



$L = \{ ab, aab, abb, aabb, abab, aaab, abbb, \ldots \}$



why not ?

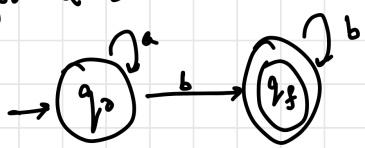


q_R = Rejected state

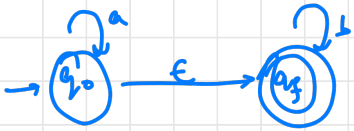
lec 8 : DFA-2

Ex: DFA for $a^* b^*$.

Sol:



or



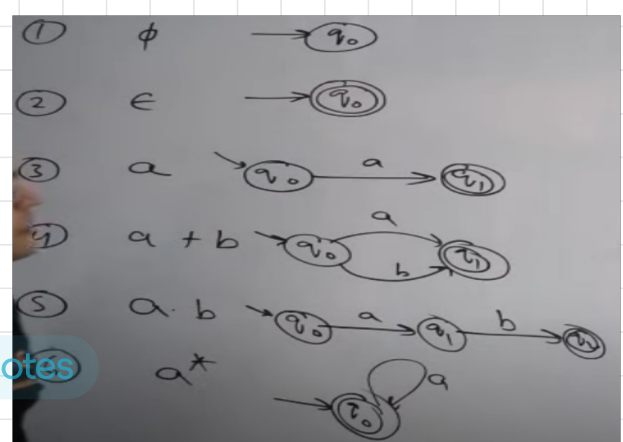
just simply take the transition on ϵ

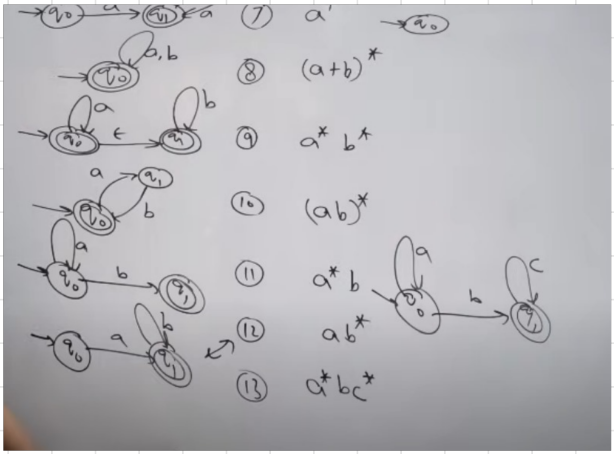
$a^* = \{ \epsilon, a, aa, \ldots \}$

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

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

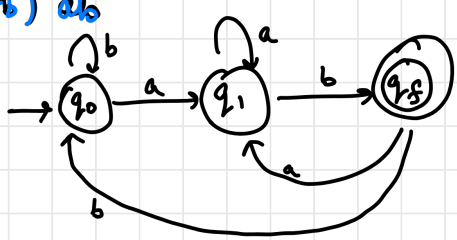
② DFA for $(ab)^*$





eg: DFA for string that end with ab, $\Sigma = (a,b)$

Sol" $(a+b)^*ab$



b on q_0
a on q_1
a & b on q_f

L = { ab, aab, bab, aaab, abab, baab, bbab, ... }

eg: (1) DFA for (1) even no. of a's
(2) " " " b's

(2) over { a, b, c ... z } that accept all strings having substring cat.

(3) DFA for strings that don't end with ab.

(4) DFA for L, $L = \{ w : |w| \bmod 3 = 0 \}$

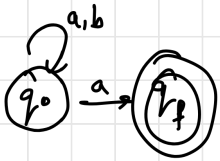
Lec 9 : Non-Deterministic finite automata

2/2/24

eg: NFA for string that ends with a

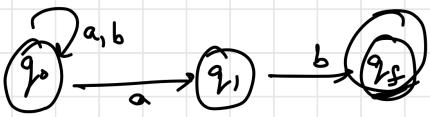
Sol^n 

$(a+b)^*$ a



eg: end with ab , conversion of NFA to DFA

Sol^n $(a+b)^*ab$

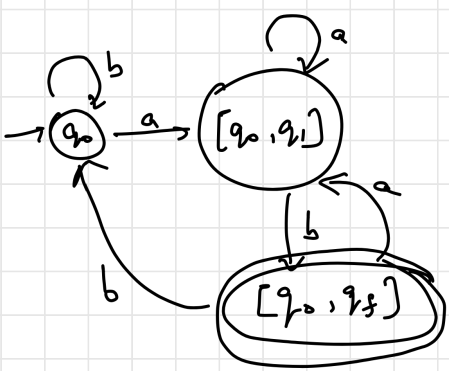
NFA 

Transition diagram

	a	b
→ q_0	q_0, q_1	q_0
q_1	–	q_f
q_f	–	–

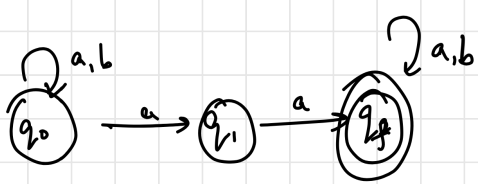
DFA

	a	b
q_0	$[q_0, q_1]$	q_0
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_f]$
$[q_0, q_f]$	$[q_0, q_1]$	q_0



eg: having substring aa

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



NFA

① if NFA has $[q_0, q_1, q_2]$ then DFA will have 2^3 states (Can)

② if " " n states " " " " at max. 2^n states.

Lec 32 : DFA to RE using Arden theorem :

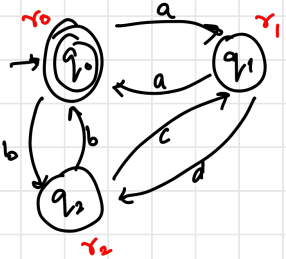
Let Σ be an alphat

P & Q are arbitrary languages over Σ.

$P \subseteq \Sigma^*$ $\quad Q \subseteq \Sigma^*$

then linear equation, $R = RP + Q$ has a sol^n QP^* and this sol^n is unique if $\epsilon \notin P$.

① for a FA, write RE eqns, each eqn corresponding to each state.



$r_0 = \epsilon + r_1 a + r_2 b$ (incoming edges)

$r_1 = a r_0 + r_2 c$

$r_2 = r_1 d + r_0 b$

② Solve using arden theorem.

eg: $E_0 = \epsilon$

$E_1 = E_0 a + E_2 a$

$E_2 = E_1 b + E_2 b$

$E_3 = E_0 b + E_3 (a+b)$

Sol^n $E_1 = \epsilon a + E_2 a \Rightarrow E_1 = E_2 a + a$

$E_2 = (E_2 a + a) b + E_2 b = E_2 (b + ab) + ab$

$E_2 = ab(b+ab)^*$, $E_1 = ab(b+ab)^* a + a$

$E_3 = E_3 (a+b) +$

$E_3 = (ab(b+ab)^* a + a)(a+b)^*$

Lec 11 Regular Grammar :

$x \in (NUT)^+$
$Y \in (NUT)^*$

Grammar G : N, T, S, P

P is finite set of production rules, $x \to y$ where
x is any string over N and T with length one or more
y is any string over N and T including null string

N : Non terminals → A, B, C
T : Terminals → a, b or 0, 1

grammar is regular grammar if it is either left linear or right linear.

Linear : Max one non-terminal in RHS of all productions

Left Linear : $A \to Bx$ or $A \to x$ where $x \in T^*$
(B : non-terminal)

Right Linear : $A \to xB$ or $A \to x$ where $x \in T^*$

eg. write down regular grammar for a^*.

Sol" Now $S \to aS \mid \epsilon$, for right linear language

$S \to \epsilon$
$S \to aS \to a\epsilon \to a$

when S is replaced with ϵ

consider $w = aaa$,

$S \to as \to aas \to aaas \to aaa$

(on replacing $S \to as$, again and again)

Now $S \to Sa \mid \epsilon$ Left Linear grammar

$S \to \epsilon$
$S \to Sa \to Saa \to \epsilon aa = aa$

consider $w = aaa$ (③ ② ①)

$S \to Sa \to Saa \to Saaa \to \epsilon aaa = aaa$

It's like reading characters from right to left like in Urdu.

eg: Regular grammar for $(a+b)^*$

Sol^n Right Linear grammar : $S \to aS \mid bS \mid \epsilon$

$w = abb$ $\quad S \to aS \to abS \to abbS \to abb\epsilon = abb$

① first symbol can be a or b

② 1st " " " " "

③ 2nd " " " " "

Left Linear grammar : ⋮

Lec 12 Regular grammar II :

eg: regular grammar over $\{a,b\}$ that start with a & end with b.

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

at max 1 non terminal is allowed

$S \to aA$
$A \to aA \mid bA \mid b$

Right linear grammar :

$w = aab$ $\quad S \to aA \to aaA \to aab$

Left linear grammar : $S \to Ab$
$A \to Aa \mid Ab \mid a$

eg. regular grammar that have substring aa

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

Sol" Right :

$S \to aA$
$A \to bA \mid a \mid b$

$S \to aS \mid ab \mid aaA$
$A \to aA \mid bA \mid \epsilon$

Left :

$S \to Sa \mid Sb \mid Aaa$
$A \to Ab \mid Aa \mid \epsilon$

eg. strings having different initial & final letter

Sol" $a(a+b)^* b + b(a+b)^* a$

Right :

$S \to aA \mid bB$
$A \to aA \mid bA \mid b$
$B \to aB \mid bB \mid a$

Left :

$S \to Ab \mid Ba$
$A \to Aa \mid Ab \mid a$
$B \to Bb \mid Ba \mid b$

eg: $(ab^*c)^*$

ab^*c

Sol" Right :

$S \to aA \mid \epsilon$
$A \to bA \mid cS$

Left :

$S \to Ac \mid \epsilon$
$A \to Ab \mid Sa$

Lec 19 : properties of null string & null language :

Null language ϕ

Null string $\epsilon \mid \lambda \mid \wedge$

① $\phi^* = \epsilon$

② $\phi^* = \{\epsilon\}$

③ $\epsilon^* = \epsilon$

Identities on Null language :

① $\phi + r = r + \phi = r$

② $\phi r = r\phi = \phi$

Identities on Null string

① $\epsilon^* = \epsilon$

② $\epsilon r = r$

Lec 20 Properties of Regular languages :

if $L_1 \cup L_2$ is regular then L_1 & L_2 are closed under union.

" $L_1 \cap L_2$ is " " " " " " " " intersection.

" $L_1 L_2$ " " " " " " " " concatenation

" L_1^* " " " " is " " Kleene Closure

" $\Sigma^* - L_1$ " " " " is " " complement.

Theorem 1 : Regular languages are closed under union. Proofs

Theorem 2 : " " " " " concatenation

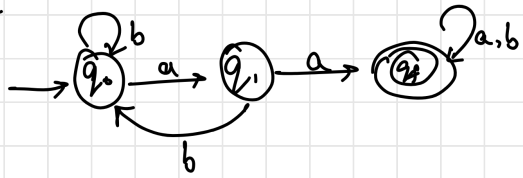
Theorem 3 : " " " " " Kleene Closure

eg: DFA that doesn't contain substring aa.

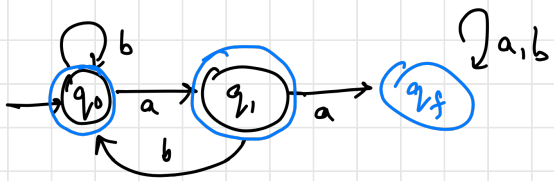
Sol^n $L = \{aa, aab, baa, \dots\}$ $(a+b)^* aa (a+b)^*$

$L' = \{\epsilon, a, b, ba, ab \dots\}$ $(a+b)^*$

DFA for L



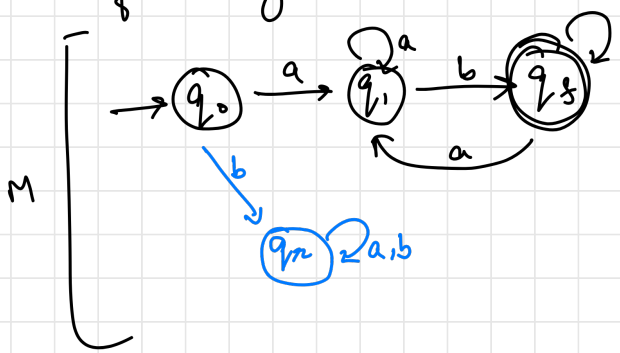
DFA for L'



eg: DFA for string that doesn't start with a & end with b

a (a+b)* b

Sol^n



δ is partial

	a	b
q_0	q_1	–
q_1	q_1	q_f
q_f	q_1	q_f

for M', make initial states final & vice-versa.

Lec 21 : Properties of regular languages 2 :

Theorem : Regular languages are closed under intersection.

eg: Design DFA over {0,1} having atleast two 0's and atmost one 1's.

Sol^n L_1 = atleast 2 0's L_2 = atmost 1 1's