day / date:

THEORY OF AUTOMATA

gc : ow6txxj

day / date: 21-8-23
Monday

Lecture # 1

set:

$A = \{1,2,3\}$ → finite

$N = \{1,2,3,\ldots\}$ → infinite

$\{1,2,3\} = \{3,2,1\}$ ⇒ order doesn't matter

$\{1,1,2,3\}$ ⇒ multiset (repetition allowed) ⇒ order doesn't matter

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

sequence:

$(3,2,1) \neq (1,2,3)$ ⇒ order matters

union:

$A \cup B$

intersection:

$A \cap B$

compliment:

$\bar{A} = U - A$

subset:

$\subseteq$

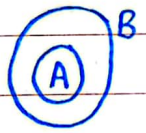
$A = \{1,1,2,3,4\}$

cardinality $= |A| = 5$

$1 \subseteq A$

proper subset:

$\subsetneq$



Null set:

$|A| = \emptyset = 0$

$|B| = \{\emptyset\} = 1$

Power set:

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

$P(A) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{2,3\}, \{3,1\}, \{1,2,3\}\}$

$2^{A \to \text{cardinality}} = 2^3$

Ordered pair

order matters

$(a,b) \neq (b,a)$

Unordered pair

order doesn't matter

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

Cartesian Product

$A \times B = \{(i,j) \mid i \in A \text{ and } j \in B\}$

Lecture # 2

day / date: 23-8-23 wednesday

Alphabet, String & definition notation

Alphabet : Denoted by 'Σ'

An alphabet is a finite non-empty set of symbols which is used to represent the input of the machine.

Common Alphabet :

$\Sigma = \{0,1\}$ → set of binary numbers

$\Sigma = \{a, b, c, \ldots z\}$ → small characters of alphabets

$\Sigma = \{$ ASCII numbers of alphabets$\}$

Strings :

Strings are the sequence of symbol of the alphabet.

e.g '010010' is the sequence from the alphabet $\Sigma = \{0,1\}$,
'ababac' is the sequence from the alphabet $\Sigma = \{a,b,c\}$

Empty string : Every alphabet 'Σ' has a special character called empty string with 0 occurence of symbol denoted by λ, ε

Power of an alphabet :

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

$\Sigma^0 = \{\varepsilon\}$ (epsilon)

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

$\Sigma^2 = \{00, 01, 10, 11\}$

$\vdots$

Σ^k

$\Sigma = \{\overset{i}{01}, \overset{j}{10}\}$

Power set:

$2^2 = 4$

$\{0101, 0110, 1001, 1010\}$

Kleen star / Kleen closure : (contains epsilon)

Kleen star is the unary operator denoted by Σ^*

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots \Sigma^\infty$$

$$\Sigma^* = \bigcup_{i=0}^{i=\infty} \{w \mid |w| = i\}$$

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

$\Sigma^* = \{\varepsilon, 0, 1, 00, 11, 10, 01, 000, 111, 001, 011, \ldots\}$

day / date:

Kleen Plus:

denoted by Σ^+

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \dots \Sigma^\infty$

$\Sigma^+ = \bigcup_{i=1}^{i=\infty} \{ w \mid |w| = i \}$

suppose L1 {10, 1}
L2 {011, 11}
Q. How many L = L1 L2 ?
a) 4 {10011, 1011, 1011, 111}
b) 3
c) 2
d) none of these

Concatination of the string:

A = 0100
B = 1100
AB = 01001100
substring

* agr A = {0, 1} B = {1, 0} hota tau saray sets bantay

* function $f: D \rightarrow R$ (input, output)

Set formal notation:

$X = \{ 0^n 1^n \mid n \geq 0 \}$

Relations:
=, >, <, ≥, ≤

$0^0 1^0 = 0 = \varepsilon$
$0^1 1^1 = 01$ $\quad 010 \notin X$
$0^2 1^2 = 0011$
$0^4 1^4 = 00001111 \in X$

Quiz:

L = {ab, aa, baa}

1) abaa baaa baa
2) aaaa baaaa
3) baaaaa baaaab
4) baaaaabaa

Q. which of the strings are in L^*

A) 1, 2 and 3
B) 2, 3 and 4
C) 1, 2 and 4
D) 1, 3 and 4

because 3 contains last element b which is not part of any set

Q2. The number of substring that can be formed by
a b e f b g h n m p

A) 10
B) 55
C) 45
D) 56

$n = 10$

$\frac{n(n+1)}{2}$

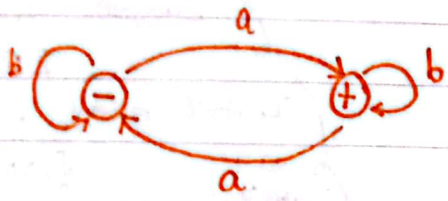
$\frac{5 \times 10(11)}{2} = 55 + \varepsilon = 56$

C

Section # M

Lecture # 1

day / date: 24-8-23
Thursday

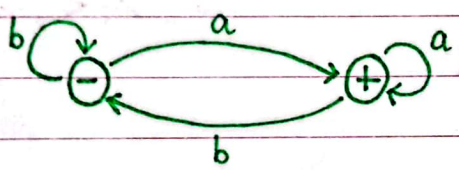


A → null string
Finite Automata
Finite State Machine

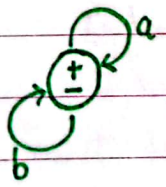
- If in any model we need acceptance of null string the starting and ending should be same.

- every state should have two outgoings.
  + yes/end final − no/start/non-final

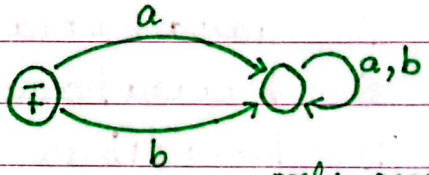
ending on 'a' state diagram:



- every model has a single start state, there can never be a no start.
- all set of words should be executed, but should be 'yes'

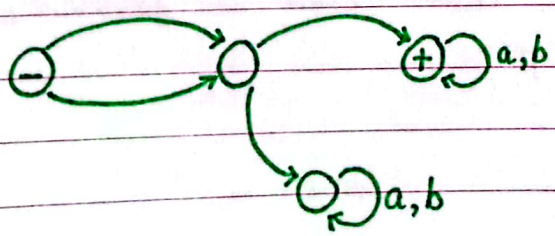


will accept all words



only accepts null

second letter always 'b' : _b___



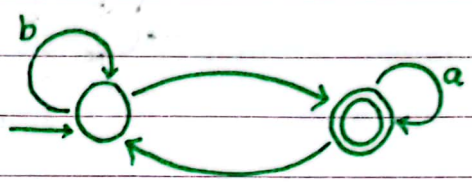
Lecture 2

day / date: 26-8-25 Saturday

Finite Automata

ending on a: ——a

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



ababab NO

we'll consider a string Yes if it ends on final state.

◎ final state

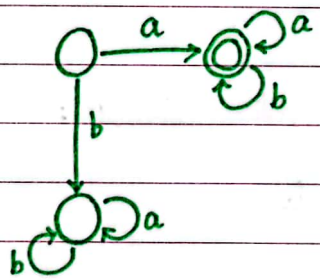
in this specific example all strings ending with 'a' are YES.

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

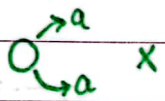
set of string = { $\wedge$ or ε ✓, a ✓, b ✓, aa, ab,

concatenate this with all elements of set and repeat.

starting with a: a——

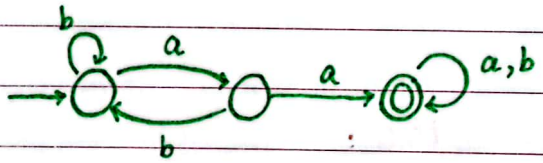


- every state should have all possible outcomes there should be
- unique outcomes

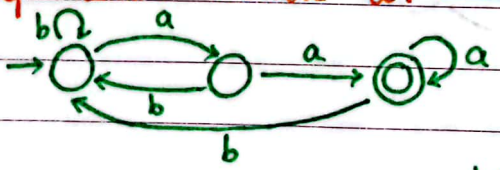


- for incoming there is no rule.
- always make such automata first which is going to be accepted
- there would always be one extra state to the required no of elements.

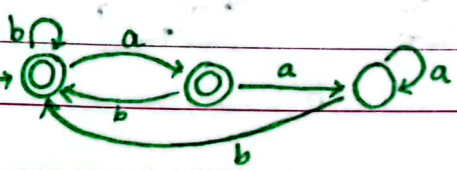
double a: ——aa——



ending with double a: ——aa

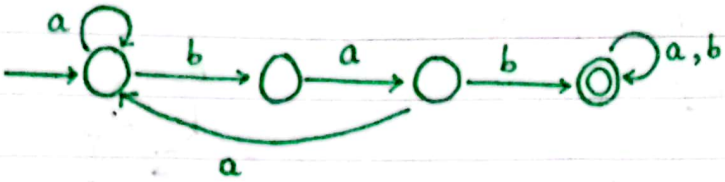


not ending with double a:

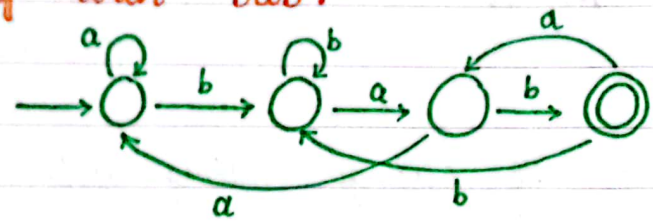


day / date:

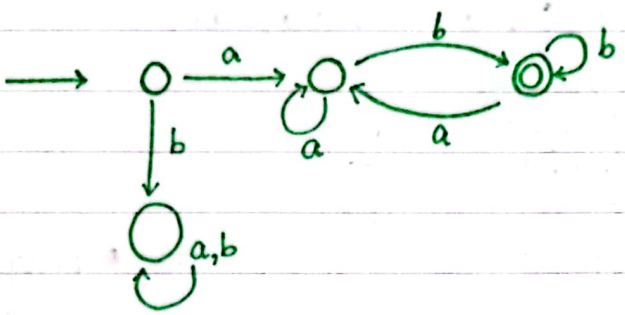
containing bab : —bab—



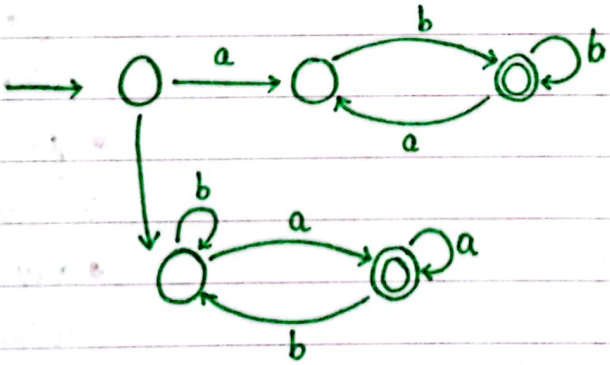
ending with bab : —bab



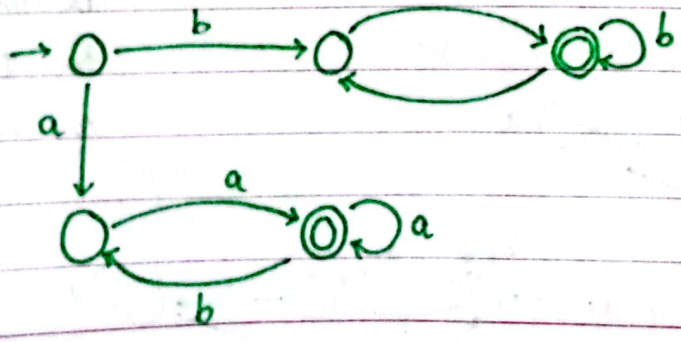
starting with a and ending at b :



starting and ending with different letters :



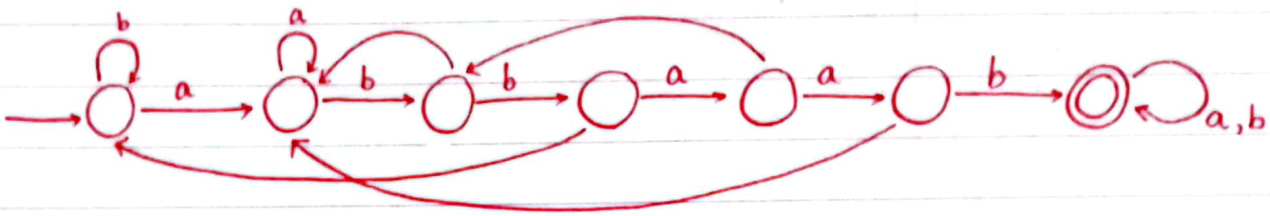
starting and ending with same letters :



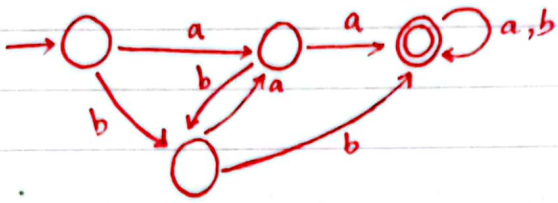
Lecture #3

day / date: 31-8-23
Thursday

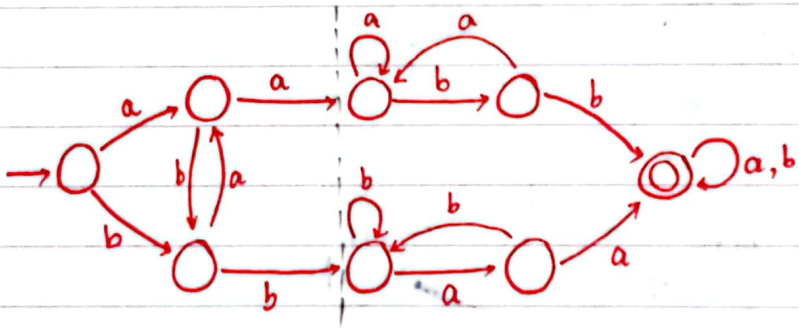
containing —abbaab— :



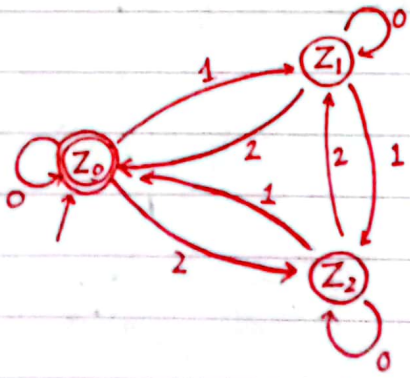
containing aa or bb :



containing aa and bb :



mod 3 :

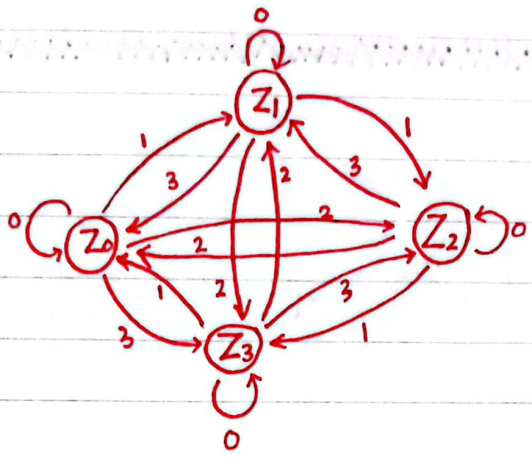


input {0,1,2}

says yes to the strings when sum is divisible by 3

mod 4 :

day / date:



{0,1,2,3}

LECTURE # 4

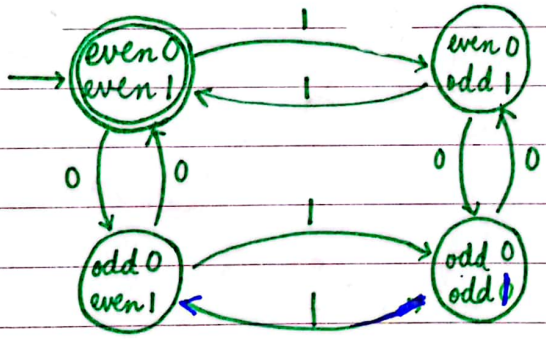
2-9-23
Saturday

Even-Even :

L = { w | w has both an even # of 0's and even # of 1's }

status = { ∧, 00, 11, 0000, 0101, 1010, 0110, 1001, ... }

even-even
even-odd
odd-even
odd-odd

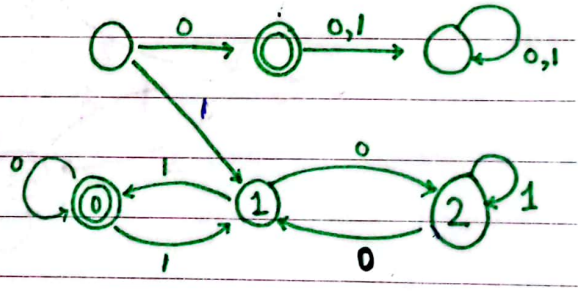


Decimal equivalent should be a divisible of 3 :

x 0 = x*2+0 = 2

x 1 = x*2+1 = 3

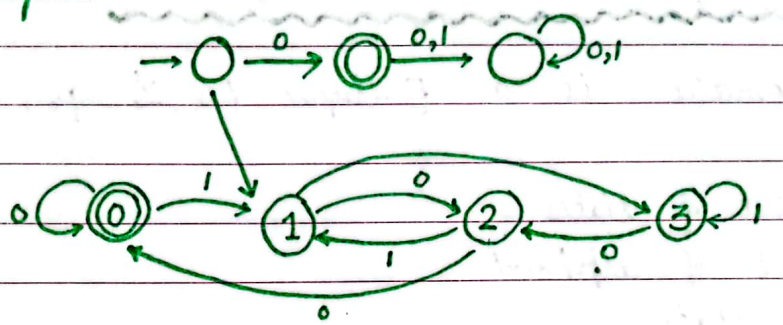
(11)0 = $(11)_2$ *2+0 = (3*2)+0 = 6



∴ 0,1,2 are remainders

day / date:

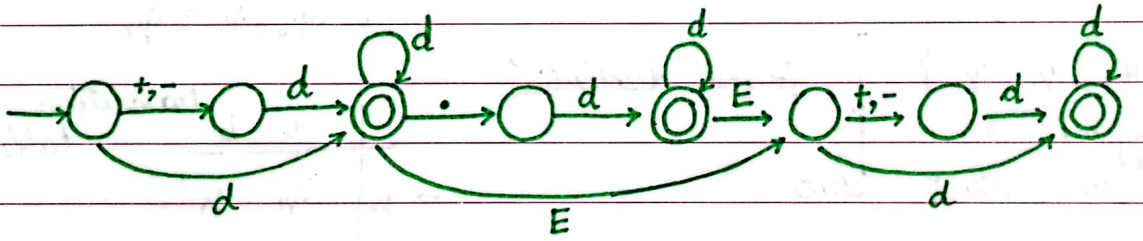
Divisible by 4 :



Regular language : that can be recognized by finite automata.

* finite automata does not have memory : so it cannot detect a string with equal no of a's and b's or a string which is a palindrome.

floating numbers :



35
+35
35.8
-35.8
0.3E1
0.3E-2
-3E8

Lecture # 5

day / date: Thursday 7-9-23

Specifying deterministic DFA : Notations for DFA

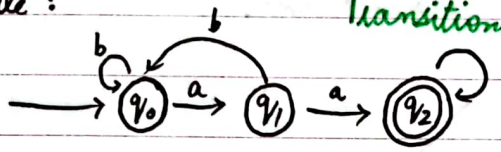
A finite automata is a 5-tuple $(Q, \Sigma, q_0, A, \delta)$

Q = finite set of states
Σ = finite set of alphabet
q_0 = start state
A = final state set
δ = transition function

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

Example :

Transition Diagram



$\delta(q_0, a) = q_1$
$\delta(q_0, b) = q_0$
$\delta(q_2, a) = q_2$
$\delta(q_2, b) = q_2$

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b\}$
$A = \{q_2\}$
q_0 is the start state

} formal description

transition table

	a	b
→ q_0	q_1	q_0
q_1	q_2	q_0
(*) q_2	q_2	q_2

δ^* → extended transition function
$\delta^*(q_0, x) = p$
$q_0 \in Q$
$x \in \Sigma^*$
$p \in Q$

$\Sigma = \{a, b\}^*$
Σ^* = set of all words

day / date:

$\delta^*(q_0, x) \Rightarrow \delta^*(x, ya)$

$\delta(q_0, a_1 a_2 a_3 \ldots a_n)$

$\delta(q_0, a_1) = q_1$

$\delta(q_1, a_2) = q_2$

$\delta(q_2, a_3) = q_3$

$\delta(\Sigma_{n-1}, a_n) = q_n$

Recursive definition of transition function:

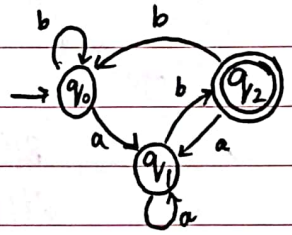
$M = (Q, \Sigma, q_0, \Lambda, \delta)$ be an FA

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

abba b

1. Base: $\delta^*(q, \epsilon) = q$
2. $\delta^*(q, x) = \delta^*(q, ya) = \delta(\delta^*(q, y), a)$ → Recursive definition

Example:



	a	b
$-q_0$	q_1	q_0
q_1	q_1	q_2
$+q_2$	q_1	q_0

$\delta^*(q_0, abab)$
$= \delta(\delta^*(q_0, aba), b)$
$= \delta(\delta(\delta^*(q_0, ab), a), b)$
$= \delta(\delta(\delta(\delta^*(q_0, a), b), a), b)$
$= \delta(\delta(\delta(\delta(\delta^*(q_0, \epsilon), a), b), a), b)$

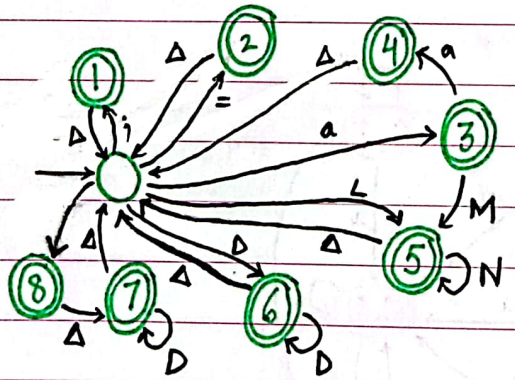
① abab = z → accept

② baa → reject

lexical analyzer:

Sum = Sum + 3.12 ;

(=) [Sum] (+) [3.12] ; Sum



M = letter and digits but no A

N = letter and digits

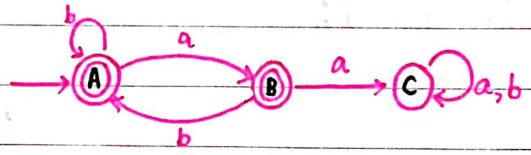
day / date: 9-9-23
Saturday

Lecture # 6

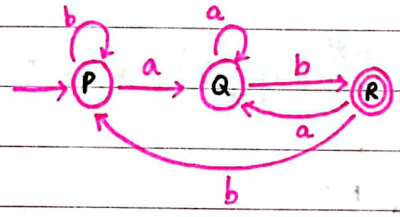
Union, Intersection and Difference

L_1: aa is not a substring
L_2: ends with ab

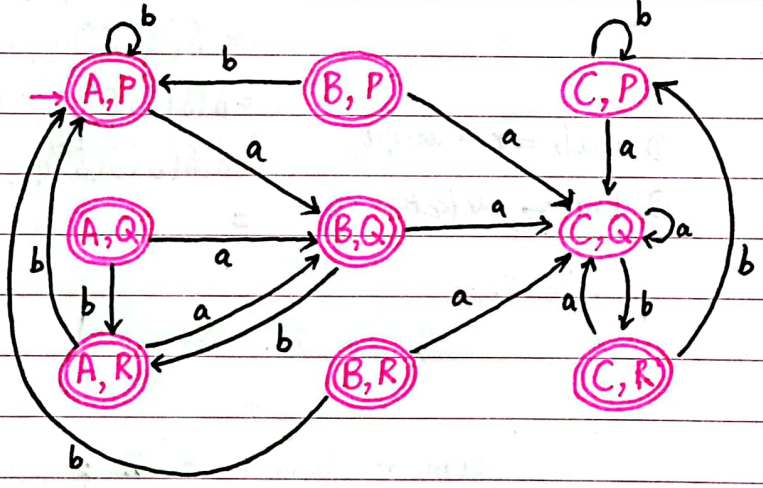
L_1 :



L_2 :



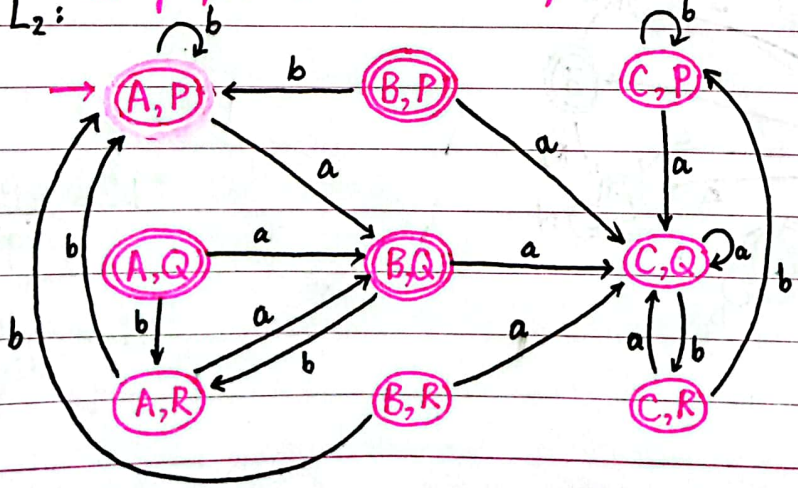
$L_1 \cup L_2$:



$\delta((A,P),a) = (B,Q)$
$\delta((A,P),b) = (A,P)$
$\delta((A,Q),a) = (B,Q)$
$\delta((A,Q),b) = (A,R)$
⋮

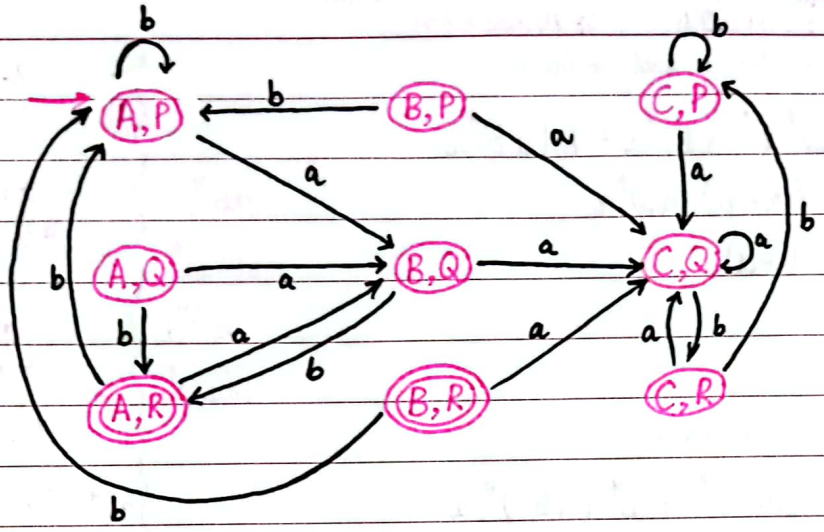
* A,Q is extra stage since there is no incoming edge so we will never come here.

$L_1 - L_2$: *only final states will change.



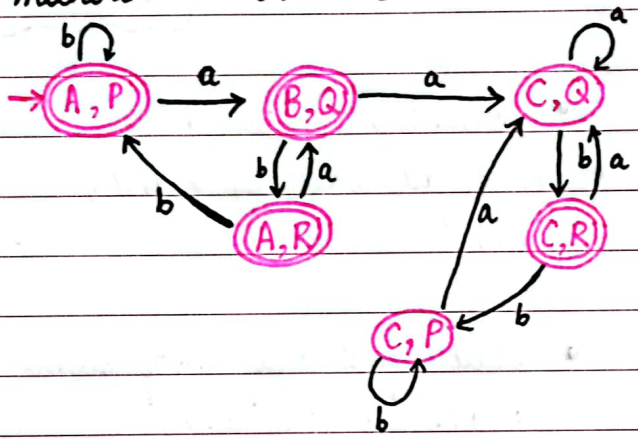
day / date:

$L_1 \cap L_2$:



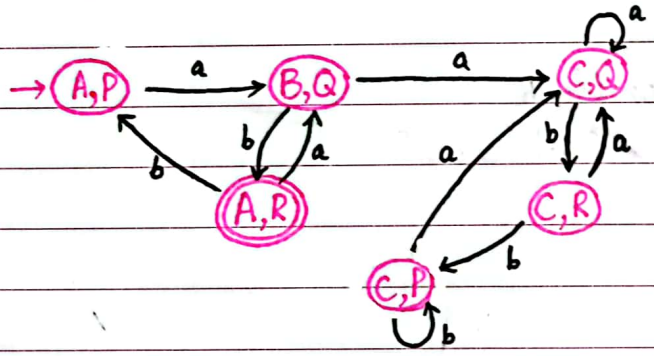
Another method : Good one

$L_1 \cup L_2$:

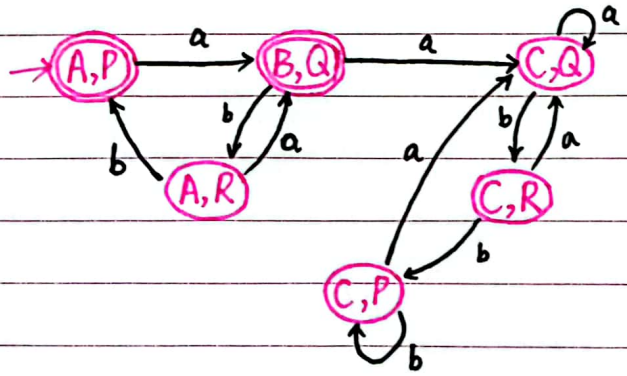


	a	b
(A,P)	(B,Q)	(A,P)
(B,Q)	(C,Q)	(A,R)
(C,Q)	(C,Q)	(C,R)
(A,R)	(B,Q)	(A,P)
(C,R)	(C,Q)	(C,P)
(C,P)	(C,Q)	(C,P)

$L_1 \cap L_2$:



$L_1 - L_2$:



day / date:

Regular Expression *imp

- b ababab a (without double a and double b) :

$$\wedge(ab)^* + b(ab)^* + \wedge(ab)^*a + b(ab)^*a$$
$$= (\wedge+b)(ab)^* + (\wedge+b)(ab)^*a$$
$$= (\wedge+b)(ab)^*(\wedge+a)$$

$(ab)^*$
$b(ab)^*$
$(ab)^*a$
$b(ab)^*a$

a.b
⇒ joining a with b

a+b
⇒ union

a^*
⇒ Kleen star

$a^* = \{\wedge, a, aa, aaa, \ldots\}$
$ab^* = \{\wedge, ab, abab, \ldots\}$
$a^+ = \{a, aa, aaa, \ldots\}$

- bbb ababab a (allowing double b)

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

$b^*(ab^+)^* + b^*(ab^+)^*.a$
$= b^*(ab^+)^*(\wedge+a)$

- abababab

a^*b^*
$(ab)^*$
ab^*
b^*b^*

- set of all words by a and b

$(a+b)^*$

- start and end with same letter

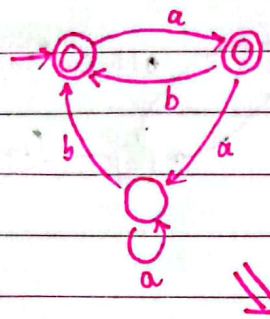
$a(a+b)^*a + b(a+b)^*b$

- containing double a or double b

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

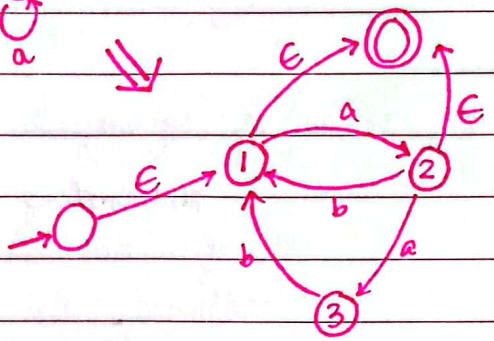
State Elimination

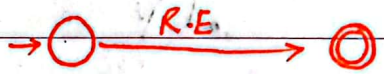
day / date: 7-10-23

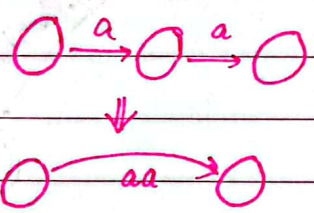


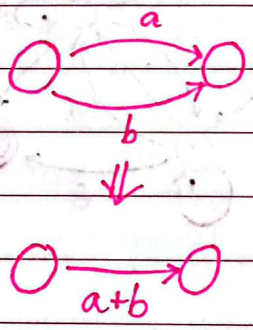
start state : unique start with no incoming

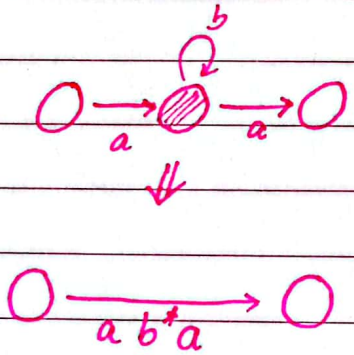
final state : unique final with no outgoing







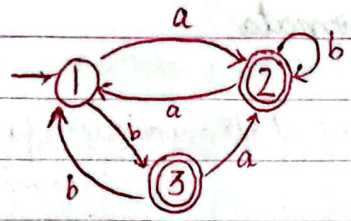




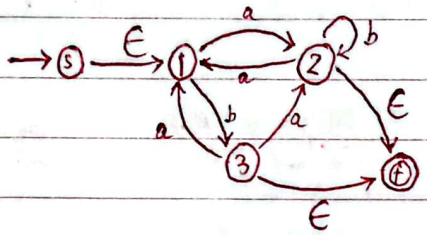
1. eliminate edge
2. incoming • inside • outgoing

* If we combine two terms this is called generalised NFA.
* NFA, NFA^ ⇒ 1 outgoing
* generalisedNFA ⇒ more than 1 outgoing

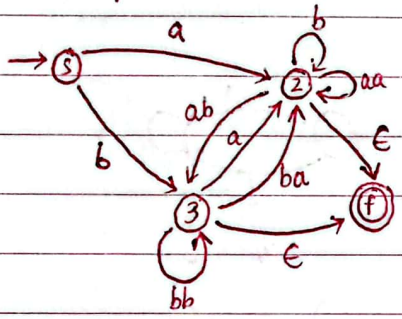
day / date:



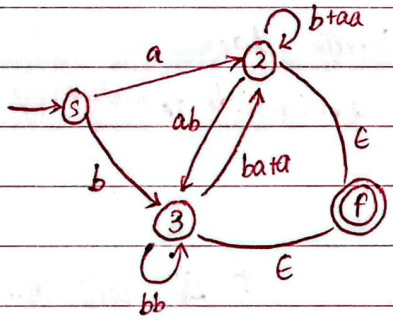
introducing start and final states



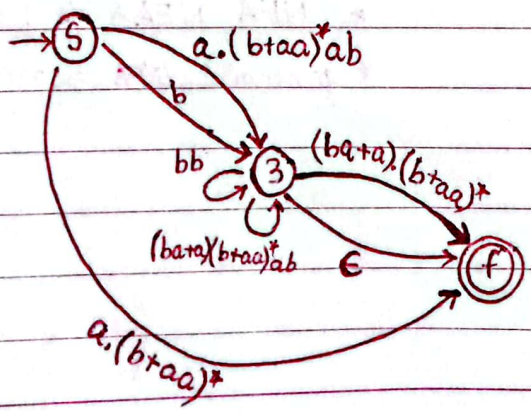
eliminating state 1



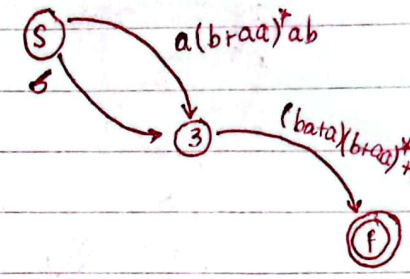
simplifying edges



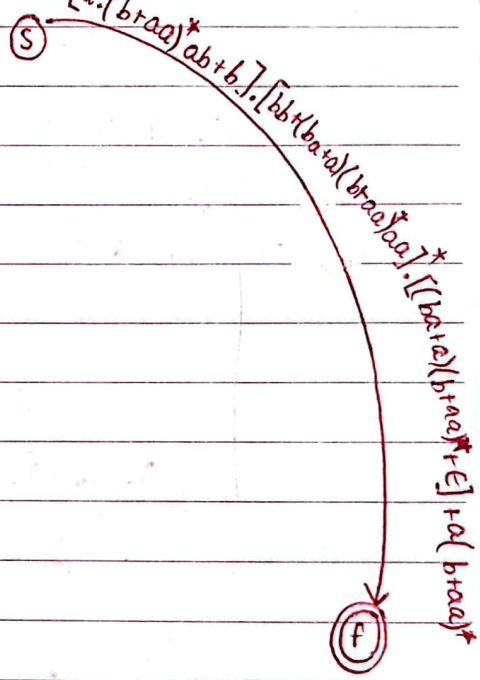
eliminating state 2



simplifying edges



eliminating state 3



day / date:

join First ki final to second ka start and ~~remove first ki final~~ make first ki final state a non-final one and second ki start into non-start one.

Pumping lemma: koi language regular hai ya ni.

if loop ⇒ regular
finite ⇒ regular

Grammar

day / date:

- Starting with a

$S \rightarrow sA$
$A \rightarrow aA \mid bA \mid \epsilon$

- $a^n b^n$ $\quad \{\wedge, ab, aabb, \ldots\}$

$S \rightarrow aSb \mid \epsilon$

- even palindrome

$S \rightarrow aSa \mid bSb \mid \wedge$

- odd palindrome

$S \rightarrow aSa \mid bSb \mid a \mid b$

- $a^m b^n c^k \mid m,n,k \geq 0$

$S \rightarrow ABC$
$A \rightarrow aA \mid \wedge$
$B \rightarrow bB \mid \wedge$
$C \rightarrow cC \mid \wedge$

- $a^n b^m c^n \mid m,n \geq 1$

$S \rightarrow aSc \mid B$
$B \rightarrow bB \mid b$

$a^n b^n c^n \mid n \geq 0$

$G_1 = (V_1, \Sigma, S_1, P_1)$
$G_2 = (V_2, \Sigma, S_2, P_2)$
$G_U = (V_U, \Sigma, S_U, P_U)$
$V_U = V_1 \cup V_2 \cup \{S_U\}$
Σ
$S = S_U$
$P_U = P_1 \cup P_2 \cup \{S_U \rightarrow S_1 \mid S_2\}$

$V_1 \cap V_2 = \emptyset$ (if G_1 and G_2 has same alphabets, re-label them)

day / date:

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

$S_1 \rightarrow aA$

$A \rightarrow aA \mid bA \mid \Lambda$

$S_2 \rightarrow Bb$

$B \rightarrow aB \mid bB \mid \Lambda$

$S_0 \rightarrow S_1 \mid S_2$

$S_1 \rightarrow aA$

$A \rightarrow aA \mid$

Cfl kay liye alphabete ka independent hona zaroori hai

$a^n . b^n$ X

$A \rightarrow aA \mid \Lambda$

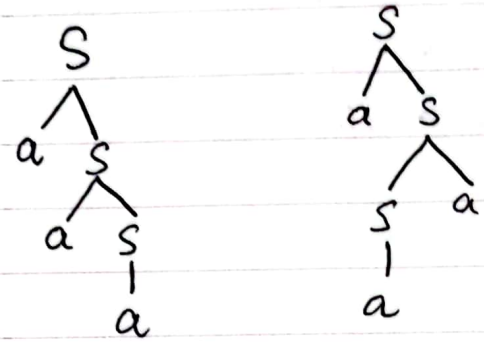
$b \rightarrow bB \mid \Lambda$

day / date:

Ambiguity

* ek hee word kay liye 2 derivations.
* compiler can't understand it, wo deterministic hota hai.

$$S \rightarrow aS \mid Sa \mid a$$



if we want aaa

$S \Rightarrow aS$
$\Rightarrow aaS$
$\Rightarrow aaa$

$S \Rightarrow aS$
$\Rightarrow aSa$
$\Rightarrow aaa$

$$S \rightarrow aSb \mid abS \mid \wedge$$

$$\Downarrow$$

$$S \rightarrow ST \mid \wedge$$
$$T \rightarrow aTb \mid abT \mid ab$$