Theory of Computing:

Complexity theory

Compatability theory

Set memership

Automata theory

1. ⇒ Machine:
   - → Finite automated
   - → Push down automaton
   - → Turning machine

Grammar = set of rules:

Chomsky Hierarchy:

Language = set of strings constructed using certain tools.

Regular expression / any subset L of Σ^*

$G = \{ V, \Sigma, P, S \}$

V = set of non terminals

Σ = alphabet set (Terminal)

P = protection

S = start symbol.

Σ = { symbols } = can't expand further

{ 0, 1, 2 .... 9 }

a − z

V = set of NT can expand further.

$G = \{ A - Z \}$

$P \Rightarrow \alpha \rightarrow \beta$

↓

have atleast one non terminal

Production = have any rule

start symbol

P. $\alpha \rightarrow \beta$ → Producer → expansion

$\alpha \in \{ V^+ \cup \Sigma \}^*$ + atleast one non terminal

$\beta \in \{ V \cup \Sigma \}^+$ * vice versa

Recap:

1. Automata theory
2. Theory of computing.
3. Complexity.
4. Computability.

ε, λ = empty string

epsilon, lambda.

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

Σ^* = Language

* concatenation
* length
* Reversal

* $+$ one or more repetition

(1 or more) repetition

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

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

Σ^+ one or more repetition.

$\Sigma^+ = \{a, b, aa, ab, ba, bb \ldots\}$

$\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$

$\Sigma^+ = \Sigma^* - \{\varepsilon\}$

special language:

$\{\}$ The empty set contain no string.

$\{\varepsilon\}$ A language contain one string ie empty string.

$\{\} \neq \{\varepsilon\}$

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

$L = \{x \mid x \text{ is } \Sigma^* \text{ and contains an even no of 0's}\}$

$= \{010, 001, \ldots 1001010 \ldots\}$.

Language rules:

* $L\{\varepsilon\} = \{\varepsilon\}L = L$.

Eg: $L = \{00, 01, 10, 11\}$

$\{\varepsilon\}L = \{00, \varepsilon01, \varepsilon10, 11, \varepsilon\}$

$= \{00, 01, 10, 11\}$

* $(L_1L_2)(L_3) = L_1(L_2L_3)$

( )

Regular expression

. concatenation

| or

* 0 or more

+ → 1 or more.

All symbol in capital letters.

Precedence:

( ) * . | (nothing b/w 2 symbol then it is concatenation)

+

* $L^+ = LL^*$

$L_1 = \{a, b\}$

$L_2 = \{c, d\}$

$L_3 = \{e, f\}$

$L_1 L_2 = \{ac, ad, bc, bd\}$

$L_1 L_2 L_3 = \{ace, acf, ade, adf, bce, bcf, bde, bdf\}$

$L_1^* = \{\varepsilon, a, b, aa, ab, ba, bb\}$

$L_1^+ = L_1^* - \varepsilon$

$L_4 L_1^* \cup_4 = \{a, b, ab, aa, ba, bb\}$. Total string = 8

* $L^+ = L^* - \{\varepsilon\}$

* $L = \{\} = \phi$

* $L\phi = \phi L = \phi$

* Union

* Intersection

* Difference

*

* α, β, γ contain terminal & nonterminal.

* 4 tuple of $G = \{V, \Sigma, P, S\}$

① $L = \{a^n | b^n | n \geq 0\}$

$= \{\varepsilon, ab, aabb, aaabbb \ldots\}$

P: $S \to \varepsilon | asb$ → production.

↓ repetition

$S \to \varepsilon | asb$

$S \to asb$ → alternates

↓

to get aabb

$S \to \varepsilon | aa\,sbb$

$S \to asb$

$S \to a\,asbb$

$\to aa\varepsilon bb$

$\to aabb.$ to get aabb

② $L = \{a^n b^{n+1} | n \geq 0\}$

$= \{b, abb, aabbb, aaabbbb \ldots\}$

P: $S \to b | asb.$

Derive aabbb.

$S \to b | asb.$

$S \to a\,sb$ → asb

$\to aa\,sbb$ → b.

$\to aabbb.$

③ $L = \{ a^m b^n \mid m \text{ may not equal to } n \}$

$m, n \geq 0$

$= \{ a, b, abb, aab, aabbb, aaabb \ldots \}$

Derive abbb

P $(S \rightarrow a \mid b \mid asb)$

$S \rightarrow asb$

$S \rightarrow a$

$S \rightarrow a \mid b \mid asb$

$\{a, b, abb, aaabbb \ldots\}$

$asb.$

$aabb.$

$aaabbb.$

$\{a \mid b \mid asb\}$

$asb.$

$aabb.$

$aaabbb.$

$abbb.$

③ P : $S \rightarrow sb \mid as \mid \varepsilon$

$S \rightarrow aS$

$\rightarrow asb$

$\rightarrow asbb$

$\rightarrow asbbb$

$\rightarrow a \varepsilon bbb$

$\rightarrow abbb$

④ P : $S \rightarrow \varepsilon \mid aA \mid bB \mid asb.$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

$S \rightarrow asb$

$\rightarrow abBb$

$\rightarrow abbb$

⑤ $\Sigma = \{a, b\}$

$L = \Sigma^*$

$\Sigma^* = \{\varepsilon, a, b, aa, bb, aab, ba, \ldots\}$

$P: S \rightarrow \varepsilon \mid aS \mid bS \mid Sa \mid Sb \mid aSb$

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

$L = \{aba, baab, abbba, \ldots\}$

$L = \{abaaba, babbbab\}$

$P: S \rightarrow \varepsilon \mid aSa \mid bSb \mid aS \mid bS \mid Sa \mid Sb$

⑥ $G = \{V, \Sigma, P, S\}$

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

$L = \{$palindrome$\}$.

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

$V = \{S\}$

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

abba
abbba
baaab.

$\alpha \rightarrow \beta$

$\beta \in \{V \cup \Sigma\}^*$

$\alpha \in \{V^+ \cup \Sigma\}^*$

Left most derivation

i) Recursively enurable grammar $\alpha \rightarrow \beta$

no restriction, Turning machines

$L = \{(ba)^n \mid n >= 1\}$, $A \rightarrow SCA \mid \varepsilon$, $C \rightarrow ab$, $Sab \rightarrow ba$

ii) Context sensitive grammar $|\alpha| \leq |\beta|$

Length $\alpha \leq$ length β linear bounded automation

iii) Context free grammar $|\alpha| = 1$

push down automation non terminal

iv) Regular grammar $A \rightarrow a$

$A \rightarrow aB$

finite state automation $A \rightarrow a \mid aB$

one non terminal on left side.

right only terminal or terminal followed by non terminal.

ii) Context sensitive grammar: (recursive language)

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

$S \rightarrow abc \mid aAbc$

$Ab \rightarrow bA$

$Ac \rightarrow Bbcc$

$bB \rightarrow Bb$

$aB \rightarrow aa \mid aaA$

$L = \{(ba)^n b^m c : n >= 0\}$

$AB \rightarrow AbBC$

$A \rightarrow bcA$

$B \rightarrow b.$

Eg: $a^n b^n c^n$

iii) Context free language:

$L = \{wcw' : w \in (a,b)^*\}$

$S \rightarrow asa \mid bsb \mid c$

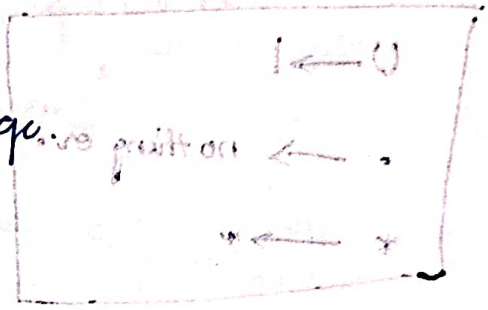
Eg: $a^n b^n$

iv) Regular Language: $A \rightarrow aIaB$

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

$S \rightarrow as|bs|\varepsilon$.

Eg: a^*



Write grammars to Language.

Find Language for G

Chomsky Hierarchy

Identify type of grammar.

Regular expressions and NFA

Precedence ( ) * . + 1 2 3 4

$0^*.1.0^*.1.0^* \longrightarrow 0^*10^*10^*$

minimum 1 one should present.

ε	11
0	0101
00	001001

RE - definition:

Basis 1: $L(a) = \{a\}$

2: $L_1 = \phi = \{\}$ set can be empty.

$R_{L_1} = \phi$

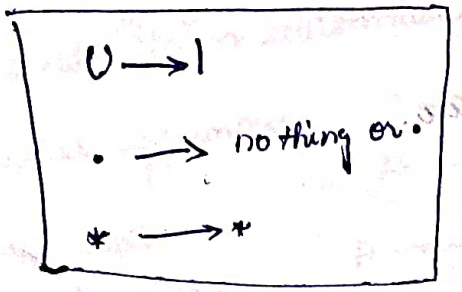
3: $L_2 = \{\varepsilon\} = \{\varepsilon\}$

$R_{L_2} = \varepsilon$

3) $L_3 = \{a, b\}$ $L_3 = \{a\}$ — a

$R_{L_3} = a \mid b$ $R_{L_3} = a$.

Induction:



1. $L(E_1 | E_2) = L(E_1) \cup L(E_2)$

$R(L_1 \cup L_2) = R(L_1) | R(L_2)$

$R_1 + R_2$

2. $R(L_1 \cdot L_2)$

$= R(L_1) \cdot R(L_2)$

$= R_1 \cdot R_2$

3. $R(L^*) = (R(L))^*$

Regular expression Examples:

$L_1 = \{a\}$

$L_2 = \{\varepsilon\}$

E_1 & E_2 are RE for L_1 & L_2

$E_1 = a$

$E_2 = \varepsilon$

$L_1 \cup L_2 = \{a, \varepsilon\}$

$L_1 \cdot L_2 = \{a\}$

$L_1^* = \{\varepsilon, a, aa \ldots\}$

$E_1 | E_2$

$= \{a, \varepsilon\}$

$E_1 \cdot E_2 = a$

$E_1^* = a^* = \{\varepsilon, a, aa \ldots\}$

$L(01) = \{01\}$

$L(01+0) = \{01, 0\}$

$L(0(1+0)) = \{01, 00\}$

$L(0^*) = \{\varepsilon, 0, 00, 000, \ldots\}$

* $L((0+10)^*(\varepsilon+1))$ = all strings of 0's & 1's without two consecutive 1's.

$(0+10)^*(\varepsilon+1)$

/ | \

ε 0+10 00, 010, 100, 1010

1, 01, 101, 0

001, 100,

010

10101

* All 0's and 1's having 101 as substring.

$L(0+1)^* = \{\varepsilon, 0, 1, 00, 01, 10, 11 \ldots\}$

$L((0+1)^*101)^*$

* All 0's & 1's having 101 as a subsequence

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

* All strings of 0's and 1's having a number of 0's that is a multiple of 3

$1^*(1^*01^*01^*01^*)^*1^*$

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

$1^*(01^*01^*01^*)^*$

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

1) RE for $(00)^*(\varepsilon+0)$

$(00)^*(\varepsilon+0) = (00)^*\cdot\varepsilon + (00)^*0$

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

$(00)^*$ → even no. of zeros

0 → odd no of zeros

$= 0^*$

2) $p + (5-5)^*(xyz)$?

$\{3,4,5\}$

~~p+4+y~~ ii) $p6y$ iii) $3xyz$ ~~iv) p35z~~

~~v) p35 35 35 x . vi) pp)35.~~

3) a) $(r+s)^* = (s+r)^*$

b) $(r^*)^* = r^*$

c) $(r^*s^*)^* = (r+s)^*$

$r=a, \quad s=b$

~~1) $(00)^*(\varepsilon+0) = (00)^*\cdot\varepsilon + 0$~~

$r^* = \{\varepsilon, aa, aaa \ldots\}$

$s^* = \{\varepsilon, b, bb, bbb \ldots\}$

$(r^*s^*)^* \{\varepsilon, a, b, ba, ab, abb \ldots\}$

$(r+s) = (a,b)+0)$

$(r+s)^* = \{\varepsilon, a, b, aa, bb, ab, ba \ldots\}$

Regular expression :

1) $L + M = M + L$.

2) $(L + M) + N = L + (M + N)$

3) $(LM)N = L(MN)$.

4) $\phi + L = L + \phi = L$.

5) $\varepsilon L = L\varepsilon = L$.

6) $\phi L = L\phi = \phi$.

7) $L(M+N) = LM + LN$

8) $(M+N)L = ML + NL$

9) $L + L = L$.

10) $(L^*)^* = L^*$.

11) $\phi^* = \varepsilon$.

12) $\varepsilon^* = \varepsilon$.

13) $(xy)^* x = x(yx)^*$.

* Strings of even length. ~~$\Sigma = \{0,1\}$~~

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

* Strings of odd length.

~~$(01)^+ \quad (10)^+ \quad (\phi)^+$~~ & $0^*(10^*10^*)1$

* String ending with 1 and not containing 00

$(0+\varepsilon)\,(10+11)^*1$ $= \{\varepsilon, 10, 11, 1011, 1110 \ldots\}1$

$(10)^*1^*$ $\{1, 101, 111, 10111, 11101 \ldots\}$

Possibilities
(00, 10, 11, 01)

* $L = \{1^n 0^m \mid n+m \text{ is even}\}$

* Even number of zeros followed by odd no of 1's.

* Identifier $(\_|l)(l|d)^*$ l = letters
d = digits.

$l = \{a, b, c \ldots z\}$ Identifiers start with _ or letters).

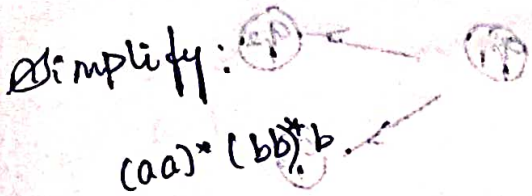
$l = [a-z]$

$d = \{0, 1, 2 \ldots 9\}$

$(\_|l)(l|d)^*$

- $(0+1+00)^* = (0+1)^*$
- $1^*(011)^+[\lambda+(011)^+]^* = (1+011)^*$

Simplify:



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

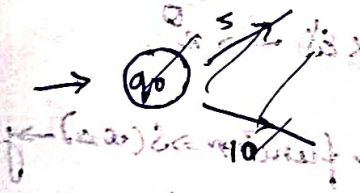
Grammar:

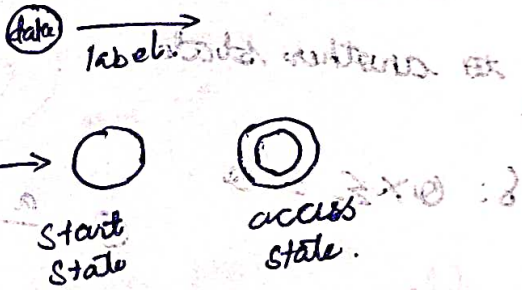
2. CH
3. Regular expression.

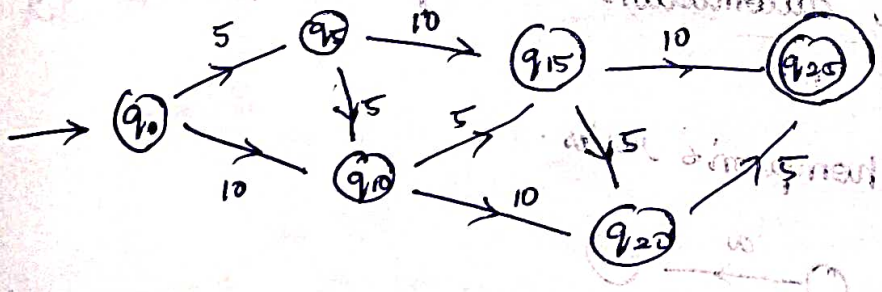
Automation:

Machine accept regular expression are finite automata

Parking payment (RS 10, RS 5). Parking cost 25 rs







Finite automation (FA) is a 5-tuple $(Q, \Sigma, q_0, A, \delta)$

Q = finite set of states.

Σ = finite input alphabets.

$q_0 \in Q$ is a initial state.

$A \subseteq Q$ is the set of accepting state.

$\delta : Q \times \Sigma \to Q$ is the transistion function.

→ Epsilon transision.

NFA

$\delta : Q \times (\Sigma \cup \varepsilon) \to 2^Q$

can transist from same state & input to move than 2 state.

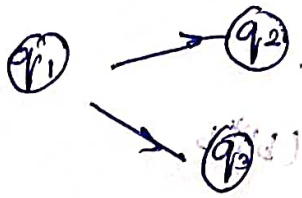
Finite automation

Deterministic FA

Non Deterministic FA

$Q \times \Sigma \rightarrow Q$

$Q \times \{\Sigma \cup \varepsilon\} \rightarrow 2^Q$



will accept epsilon.

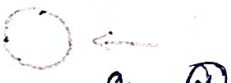
not accept epsilon

from one state to acc i/p transition to another state

from one state to acc translation to multiple state.

$\delta : Q \times \Sigma \rightarrow Q$

$\delta : Q \{\Sigma \times \varepsilon\} \rightarrow 2^Q$

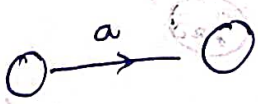


Transition function $\Rightarrow \delta(x, a) \rightarrow y$

Construct finite automation to regular expression.

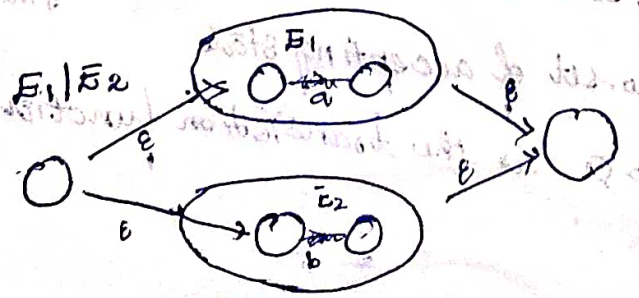
NFA

Thompson's rules.

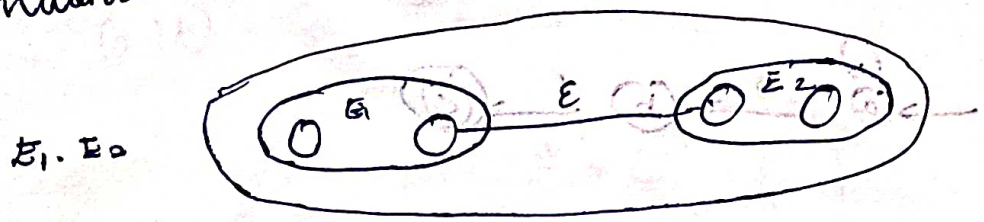


ϕ (no connection)

union $E_1 | E_2$

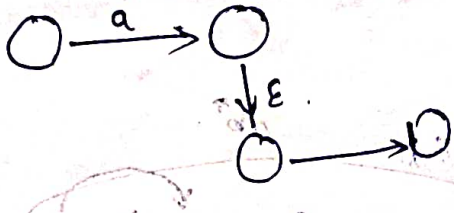


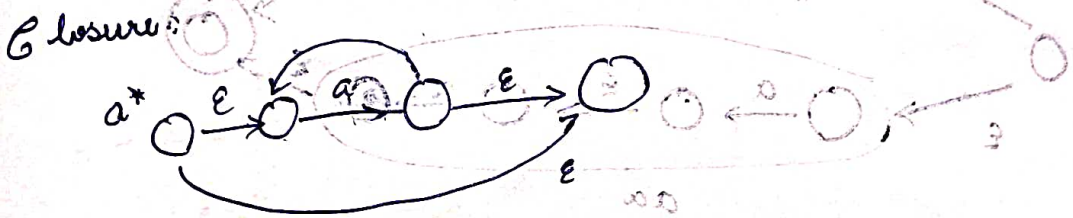
concatenation:

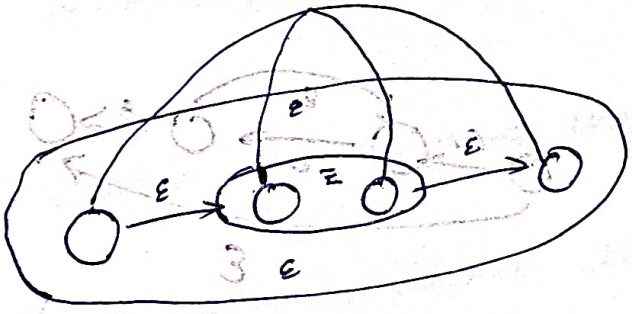


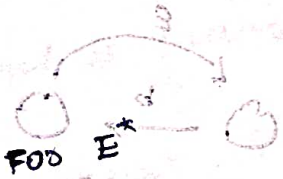
E1. E2

a. b





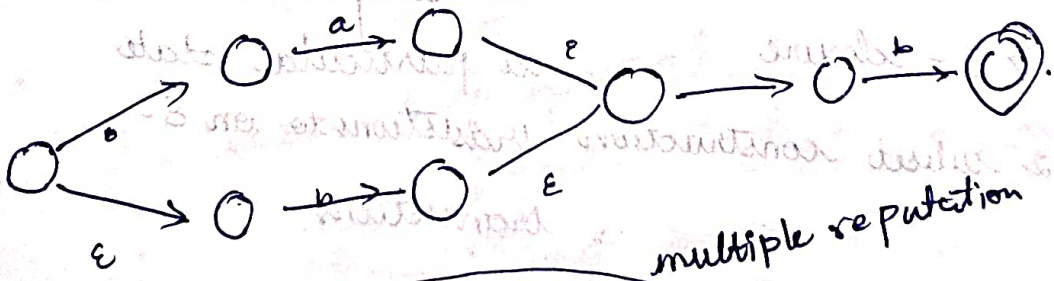




RE → NFA

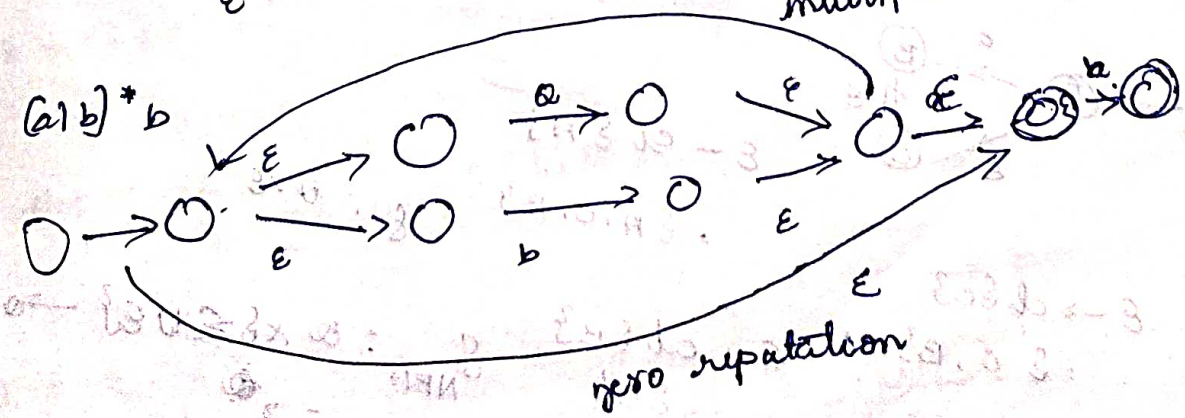
from start to acc state the string is acceptable

(a|b) b

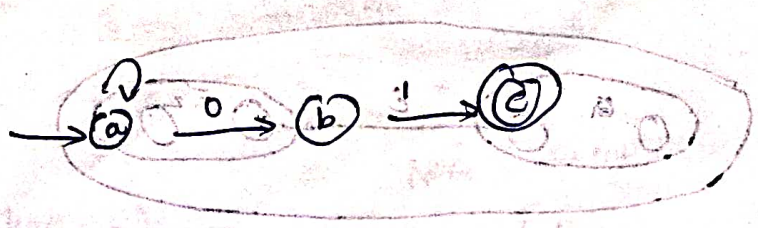


(a|b)* b

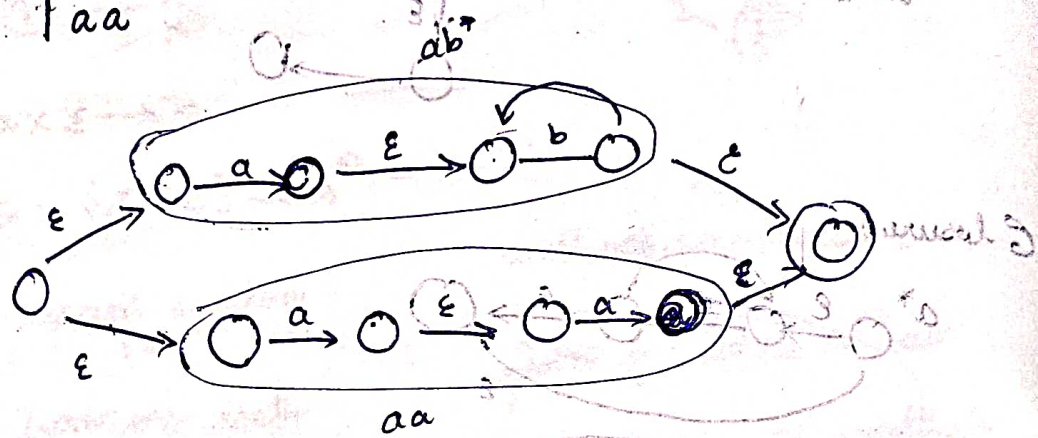
multiple repetation

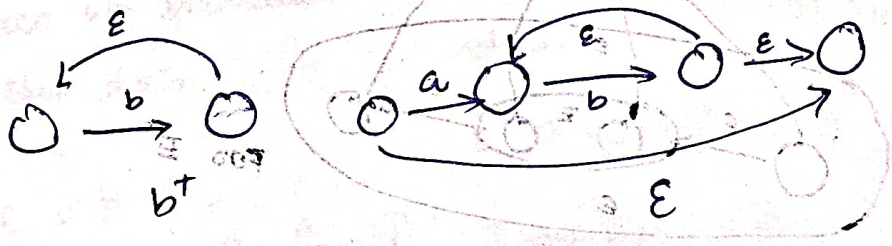


0,1



$ab^* + aa$



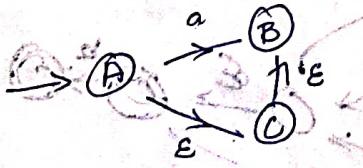


NFA → DFA:

ε - closure

subset construction

is set of state to which a particular state transitions to on ε - transitions.



$\varepsilon - cl\{A\}$

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

$\varepsilon \to cl\{C\}$

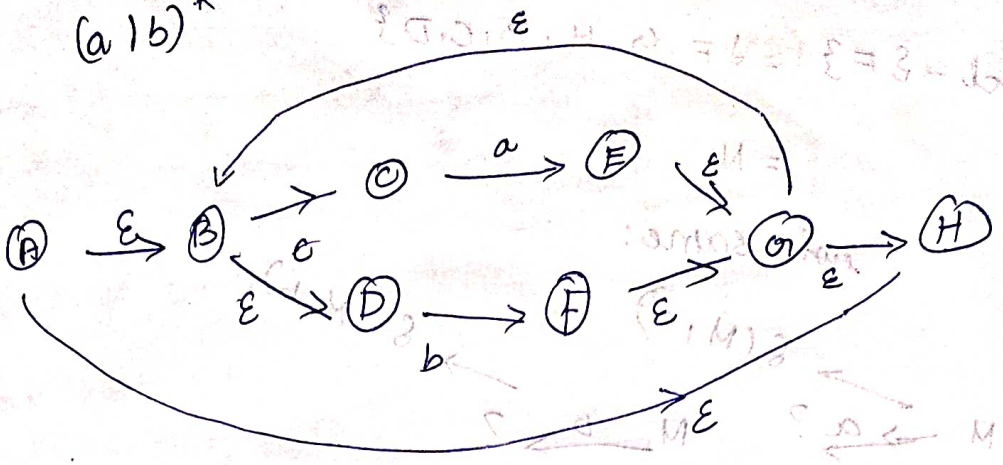
$= \{C, B\}$ note: $\varepsilon - cl\{B\}$

$= \{B\}$

$\delta_{NFA} : Q \times \{\Sigma \cup \varepsilon\} \to$

$= 2^Q$

$(a|b)^*$



$\varepsilon - cl\{F\} = \{F, G, H, B, C, D\}$.

$\varepsilon - cl\{B\} = \{B, D, C\}$.

$\varepsilon - cl\{A\} = \{A, B, C, D, H\}$

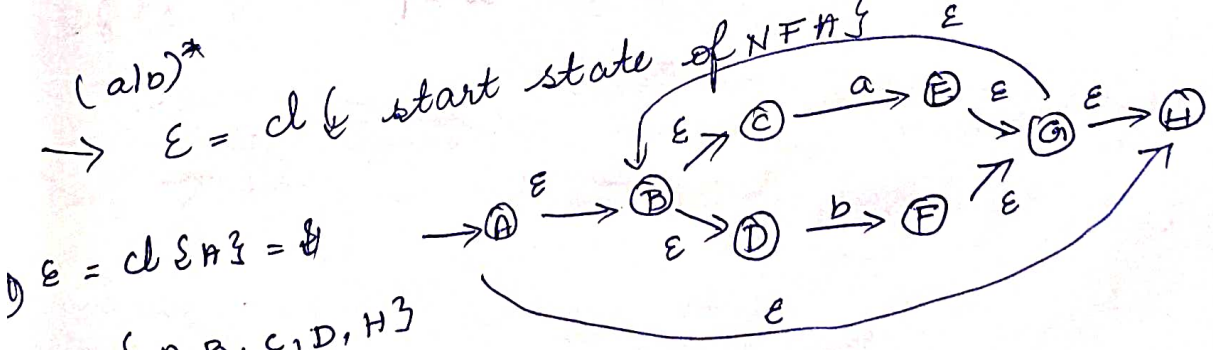
ε = without accepting a input it transits

Non determinism : without the input it move to other state.

δ DFA $\xrightarrow{a \times z.}$ Q

$(a|b)^*$

→ ε = cl ↓ start state of NFA $\}$



$\varepsilon = cl\{A\} = \phi$

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

states of DFA = L

L → ? $\quad \delta(L, a)$

L → ? $\quad \delta(L, b)$

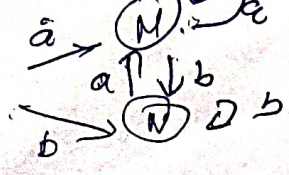
a. C → E

b. D → F

① $\varepsilon \to cl\{E\} = \{E, G, B, H, C, D\}$

= M

②



ε-cl $\{F\}$ = {F, G, H, B, C, D}

= N

not same:

step ③ M —a→ ? $\delta(M,a)$; M —b→ ? $\delta(M,b)$

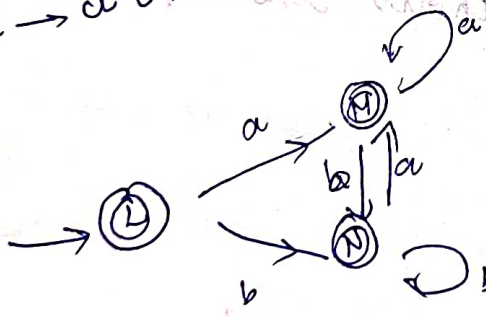
ε-cl $\{G\}$ = M. (from: M.)

ε-cl $\{F\}$ = N

step ④ N —a→ ? $\delta(N,a)$; N —b→ ? $\delta(N,b)$

$\varepsilon \rightarrow$ cl $\{E\}$ = H.

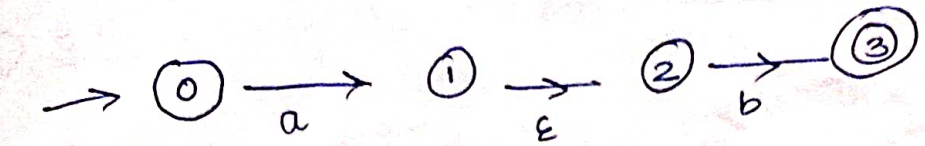
$\varepsilon \rightarrow$ cl $\{F\}$ = N.



Start state = L.

accept state = L, M, N.

RE = a, b.



NFA

construct DFA

Start state = 0.

1) ε cl {0} = {0} = A

Label with next input = A

2) $\delta(A, a) = A \xrightarrow{a} ?$ ε cl {1} = {1, 2} = B

$\delta(A, b) = A \xrightarrow{b} ?$ ε cl {3} = ϕ

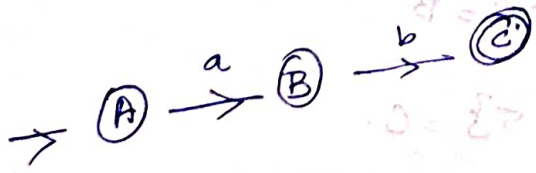
ε cl {1} $\neq$ ε cl {0} so we give new label as B.

3) δ {B, a) = ϕ

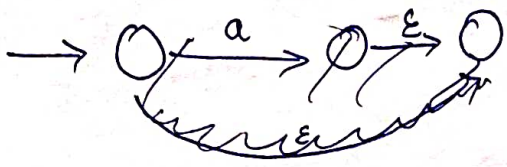
δ (B, b) = ε cl {3} = {3} = C.

4) δ {c, a} = ϕ

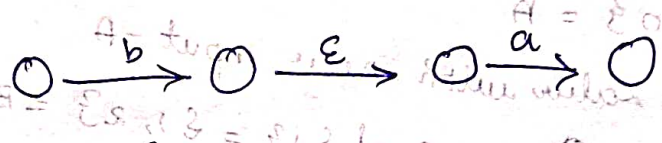
δ (c, b) = ϕ.



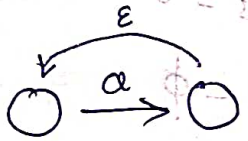
① $a^+ | ba$

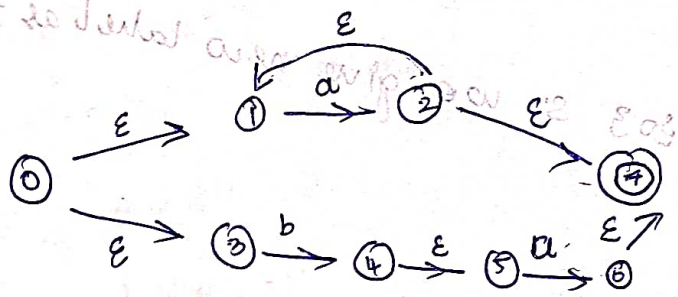


ba



a^+





starting state = 0.

① $\varepsilon\text{-cl}(0) = \{0, 1, 3\} = A$.

② $\delta(A, a) = \varepsilon\text{-cl}(2) = \{2, 7\} = B$

$\delta(A, b) = \varepsilon\text{-cl}(4) = \{4, 5\} = C$.

③ $\delta(B, a) = \varepsilon\text{-cl}(2) = \{2, 7, 1\} = B$

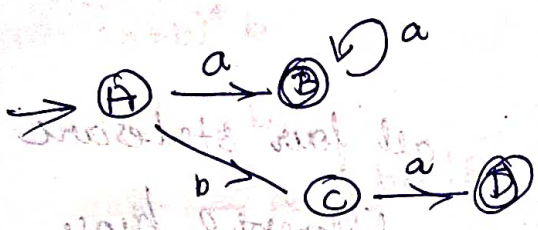
$\delta(B, b) = \phi$.

(4) $\delta(C, a) = \varepsilon cl(6) = \{6, 7\} = D$

$\delta(C, b) = \varepsilon cl\ \phi$ ——

$\delta(D, a) = \phi$

$\delta(D, b) = \phi$



Transition state.

state / i/p	a	b
A_{NA}	B_A	C_{NA}
B^*	B_A	ϕ
C_{NA}	D	ϕ
D^*	ϕ	ϕ

	a	b
$\rightarrow\{0,1,3\}$	$\varepsilon cl\{2\} = \{2,7\} = B$	$\varepsilon cl\{4\} = \{4,5\} = C$

Minimisation of DFA reduces the complexity.

Non accepting $= \{A, C\}$

Accepting state $= \{B, D\}$

A, B are not same since it doesn't transist to the same state ie (A, b) goes to NA, (B, b) to ϕ.

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

non accepting | accepting

$\{A, C\}$

$A \neq C$

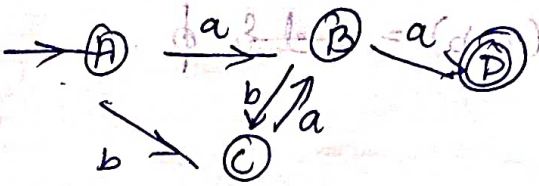
$\{B, D\}$

$B \neq D$.

Where as $A \neq C$, $B \neq D$. It is already in minimal form. We could'nt minimise it further

(1)

	a	b
A	B NA	C NA
B	D A	C NA
C	B NA	ϕ
D*	ϕ	ϕ



all four states are different & hence it is in the minimised form.

{A, B, C, D}

NA

{A, B, C} {D}.

Show the transition for i/p string 'abaa'.

$\delta(A, abaa)$

$= \delta(B, baa)$

$= \delta(C, aa)$

$= \delta(B, a)$

$=$ D in accepting state :

$\delta(\delta(\delta(\delta(A, a), b), a), a))$

B

C

B

D

Answers: Tutorial : 1

① $\Sigma = \{a, b\}$

end with b.

$(a+b)^* b$.

$S \rightarrow Ab$.

$A \rightarrow aA \mid bA \mid \varepsilon$.

$\boxed{A \rightarrow aA \mid bA \mid b}$

② $\Sigma = \{a, b\}$

$L = \{a^n b^m \mid n \neq m ; n, m \geq 0\}$.

$S \rightarrow a \mid b \mid aA \mid bA$

③ a & b a / b → even → a → even
odd b - odd.

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

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

$S \rightarrow aA$
$\rightarrow aaA$.
$\rightarrow aabA$
$\rightarrow aabab$.

b count is odd & a count is even

$\{b, aba, baa, aab, \ldots\}$

$S \rightarrow b \mid aBa \mid Baa \mid aaB$.

~~$B \rightarrow aBa \mid b \mid Bas \mid aaB$.~~

$S \rightarrow asa \mid bsb \mid bsa \mid asb \mid a \mid b$.

$S \rightarrow asa$
$\rightarrow absba$
$\rightarrow abasaba$
$\rightarrow abababa$.

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

$S \longrightarrow AAS \mid A$

$A \longrightarrow a \mid b$.

4) $(a+b)^*$

$S \longrightarrow \varepsilon \mid abs$.

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

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

$= (a+b)^*$.

ii) $\phi (a+b)^* = \phi$

iii) $\phi + b^* = b^*$

iv) $\varepsilon + (a^* + b^*)^* = (a+b)^*$

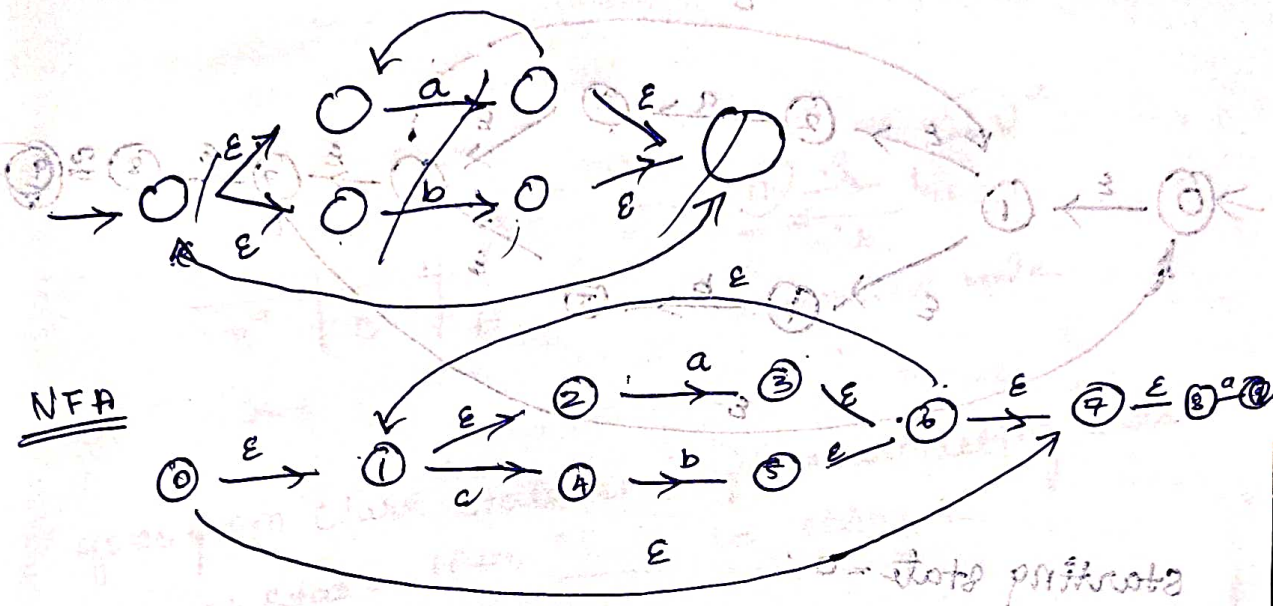
6) $\{a+ab\}^*$

$\Rightarrow ab \rightarrow$ not a grammar.

$(a+b)^* a$

Construct a NFA, DFA, min DFA

Show the moves of FA for the string 'abaa'.



NFA

DFA

Starting state = 0.

$\varepsilon cl\{0\} = \{0, 1, 2, 4, 7, 8\} = A.$

$\delta(A, a) = \varepsilon cl\{3\} = \{3, 6, 7, 8, 1\} = B$

$\varepsilon cl\{a\} = \phi\{a\}$

$\delta(A, b) = \varepsilon cl\{5\} = \{5, 6, 7, 8\} = C$

$\delta(B, a) = \phi\{\varepsilon cl\{a\}\} = \{a\} = D.$

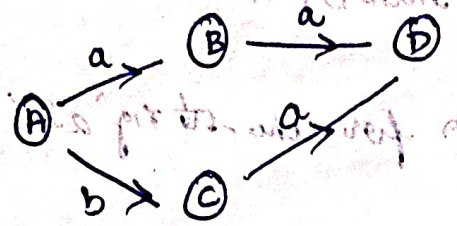
$\delta(B, b) = \phi$

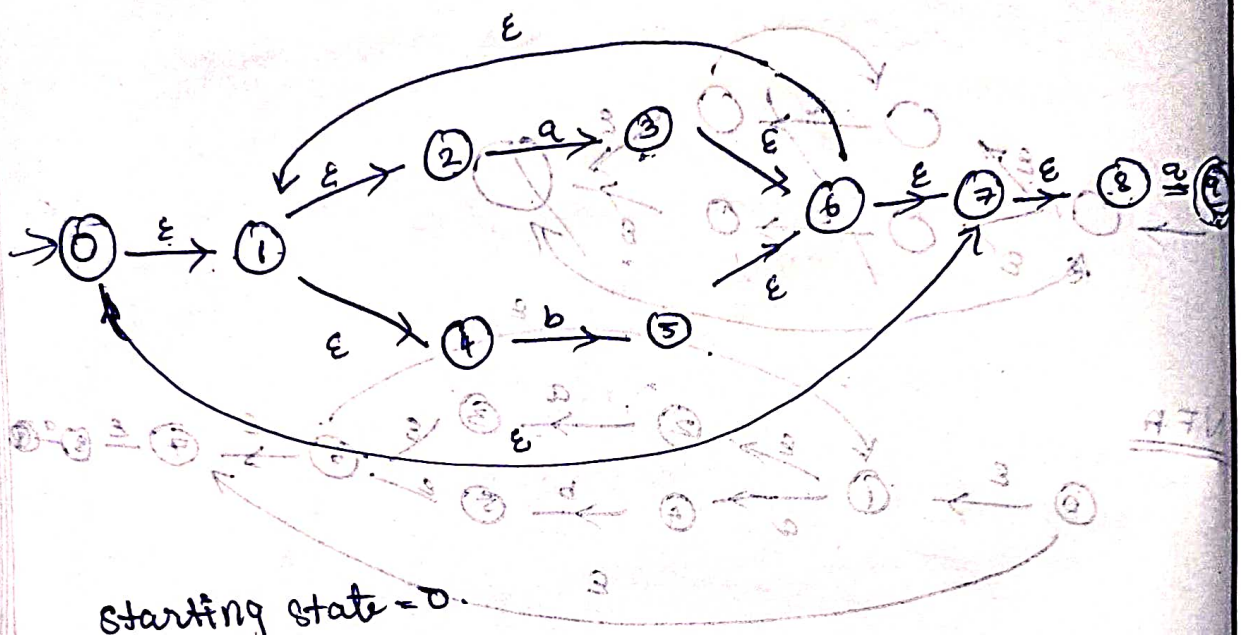
$\delta(C, a) = \phi \quad \varepsilon cl\{a\} = D.$

$\delta(C, b) = B \; \phi.$

$\delta(D, a) = \phi$

$\delta(D, b) = \phi$





starting state = 0.

$\varepsilon\text{-cl}\{0\} = \{0, 1, 2, 4, 7, 8\} = A$

$\delta(A, a) = \varepsilon\text{-cl}\{3, 9\} = \{3, 6, 7, 8, 1, 2, 4, 9\} = B$

$\delta(A, b) = \varepsilon\text{-cl}\{5\} = \{5, 6, 1, 2, 4, 7, 8\} = C$

$\delta(B, a) = \varepsilon\text{-cl}\{3, 9\} = B.$

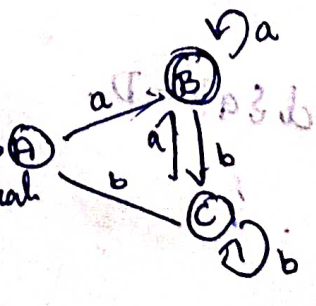
$\delta(B, b) = \varepsilon\text{-cl}\{5\} = C.$

$\delta(C, a) = B$

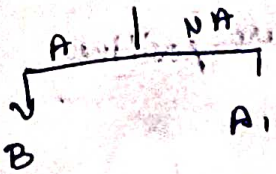
$\delta(C, b)$

	a	b
A_{NA}	B_A	C_{NA}
B_A^*	B_A	C_{NC}
C_{NA}	B_A	C_{NC}

are equal

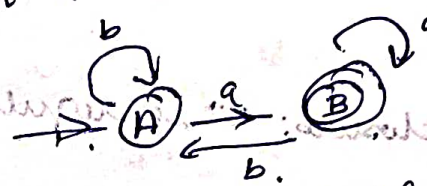


{A, B, C}



A and C are equal we can remove any one. Here we remove C and replace those place with A

	a	b
A	B	A
B	B	A



Minimised state.

goes from start state and after completing ends in accept state then it is string is accepted.

~~$\delta(A, aba)$~~ $= \delta(A, aba)$

$= \delta(B, ba)$

$= \delta(A, a)$

$= B$

Accepting.

if it is $\delta(A, bb)$

$= \delta(A, b)$

$= A$

It would be non accepting

Regular languages are closed under
$\cup, \cap, \cdot, ', -$, complement, reversal, Kleen closure, positive closure.

Closure properties:

L & M are regular languages for regular expression

RE's:

* Kleen closure: R^* is regular expression whose language is L^*

* Positive closure: R^+ is RE whose language is L^+.

* Union: $L \cup M$ is closed given by RE $R+S$.

Intersection: $L \cap M = \overline{\bar{L} \cup \bar{M}}$

(+ Using Demorgan's Law.

+ Union & Complement are closed under regular languages.

Concatenation: RS is a regular expression whose language is LM.

Completion: If L is regular language, so is $\bar{L}$

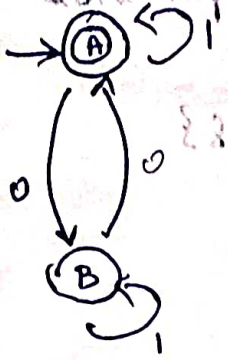
Difference is closed $\bar{L} = \Sigma^* - L$.

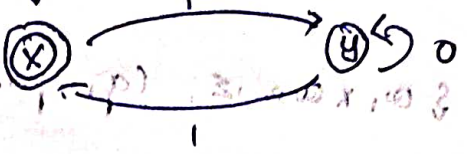
$L - M = L \cap \bar{M}$

$L \cap M = \overline{\bar{L} \cap \bar{M}}$

$\bar{L} = \Sigma - L$

Example : L: Even no of zeros, M = even no of ones.





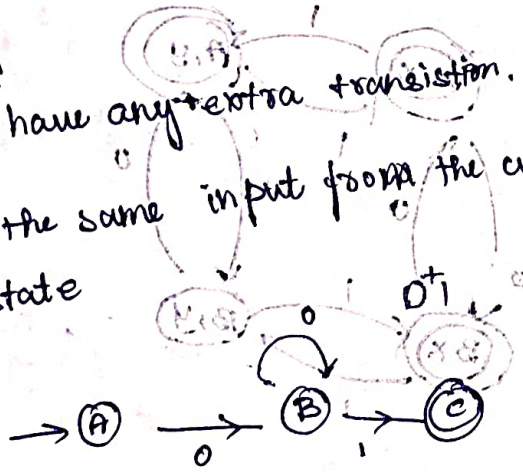
L	0	1
A*	B	A
B	A	B

M	0	1
X*	Y	X
Y	X	Y

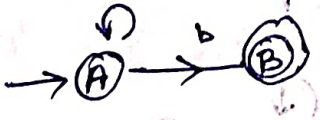
DFA

1) Not have any extra transistion.

2) For the same input from the current state it goes to only one state



a*b

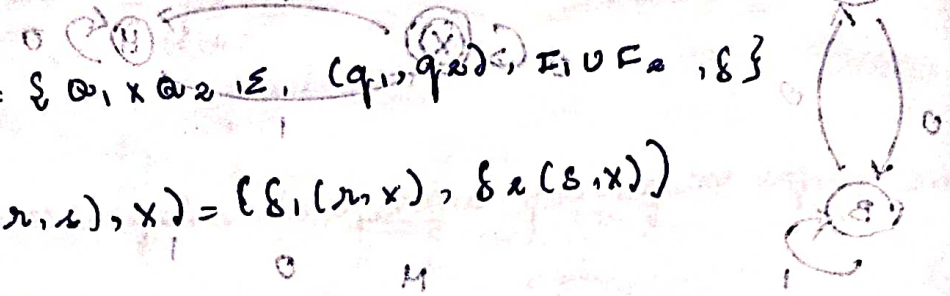


L even no of 0's. 1001
1010

	0	1

Union:

$L = \{Q_1, \Sigma, q_1, F_1, \delta_1\}$; $M = \{Q_2, \Sigma, q_2, F_2, \delta_2\}$



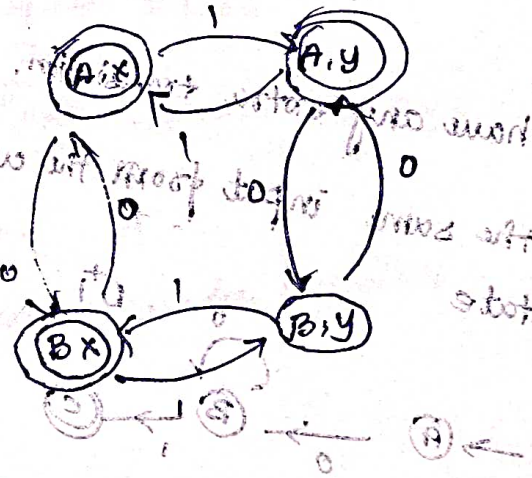
$L \cup M = \{Q_1 \times Q_2, \Sigma, (q_1, q_2), F_1 \cup F_2, \delta\}$

$\delta((r,s), x) = (\delta_1(r,x), \delta_2(s,x))$

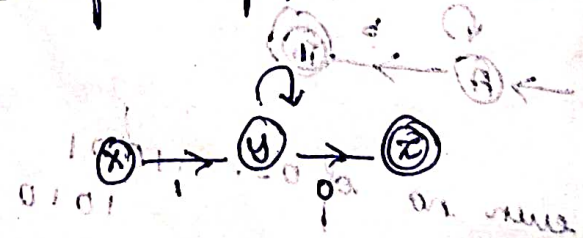
L	0	1
A^*	B	A
B	A	B

M	0	1
x	y	x
y	y	x

L ∪ M	0	1
AX^*	BX	AY
AY^*	BY	AX
BX^*	AX	BY
BY	AY	BX



Finite state contains any one of finite state.



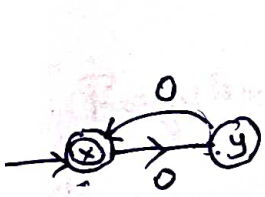
M	0	1
x	ϕ	y
y	z	y
z	ϕ	ϕ

AX	ϕ	ϕ
AY	BZ	ϕ
AZ	ϕ	ϕ
BX	ϕ	CY
BY	BZ	CY
BZ	ϕ	ϕ
CX	ϕ	ϕ
CY	ϕ	ϕ
CZ	ϕ	ϕ

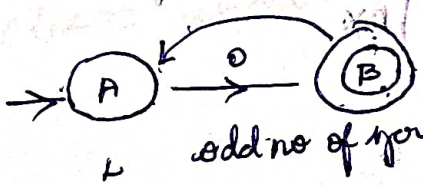
$\Sigma = \{0, 1\}$

$L = (00)^*0$

$M = (00)^*$



even no of zeros



odd no of zeros.

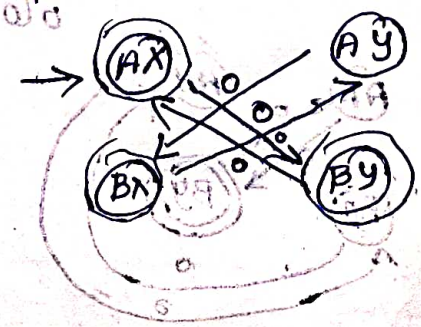
	0
A	B
B*	A

	0
M	y
y	x

combine automated M states AY; AX, BX, BY.

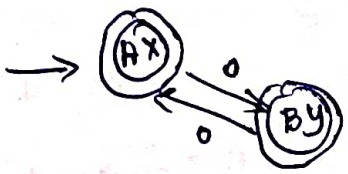
	0
AX*	BY
AY	BX
BX*	AY
BY*	AX

for union accept state might change



We see where that start from start state and goes to accepting stat.

BX & AY are eliminated because they can't move from start state.



$\{00, 0000\} \cap \{0, 000, 00000\} = \phi$.

$L \cap H$	0
→ AX	BY
AY	BX
BX*	AY
BY	AX

No common between even & odd no of 0 zeros so ϕ.

$\bar{L} - M = L \cap \bar{M}$.

complement is accepting become non accepting & non accepting become accepting.

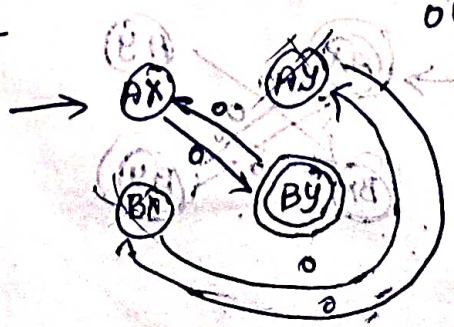
$\bar{M}$	0
x	y
y*	x

$\bar{M}$ =

$0(00)^*$ $\bar{M}$ become odd no of zeros.

$L \cap \bar{H}$	0
→ AX	BY
AY	BX
BX	AY
BY*	AX

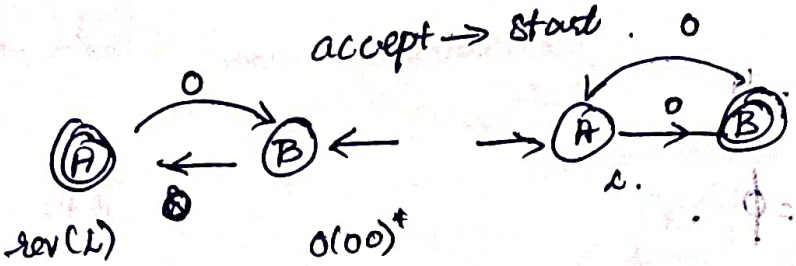
$0(00)^*$



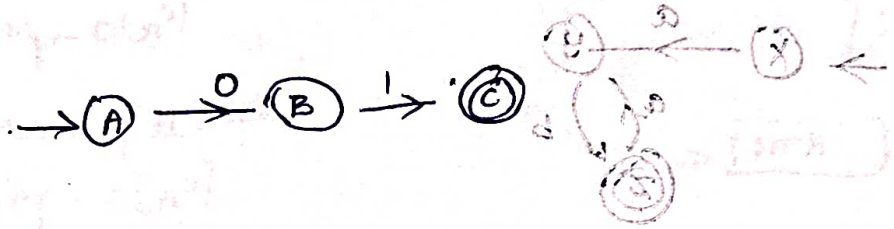
AY & BX are eliminated

Reversal:

rev(L) = start → accept & reverse transition.

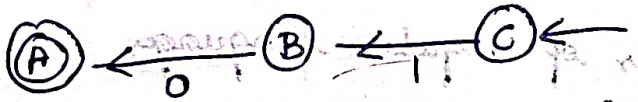


L = 01



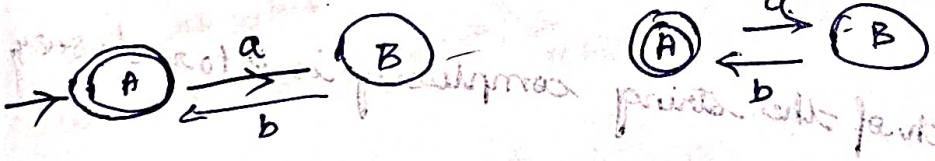
L^R = ?

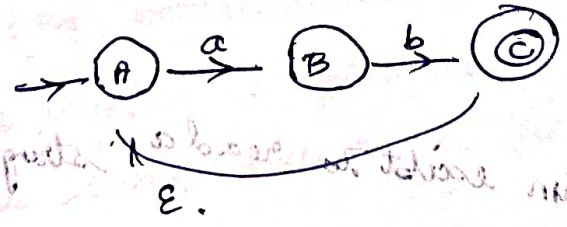
L^R = 10.



Regular automata is closed under reversal too.

$L = (ab)^*$ $(ab)^+$





ε cl {A} = {A} = X

$\delta(x, a)$ = ε cl {B} = {B} = Y.

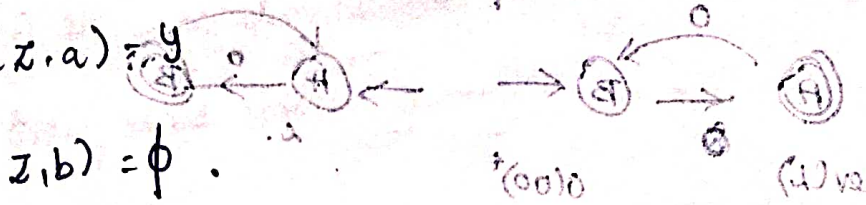
$\delta(x, b)$ = ϕ

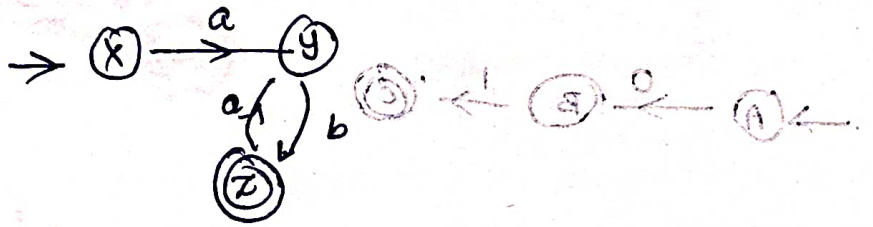
$\delta(y, a) = \phi$

$\delta(y, b) = \varepsilon$ d $\{c\}$

$\delta(z, a) = y$

$\delta(z, b) = \phi$.





Decision properties of Regular languages

Pumping Lemma.

→ algorithm that takes a formal description of a language & tells whether some properties hold or not

If n is length of the string complexity is to read a string

O(n)

Yes, the algorithm exist to read a string.

Testing DFA with string n - O(n)

deterministic testing for NFA - $O(n^3)$ - $= n(n^2)$

$= n^3$.

epsilon closure (non-determinism)

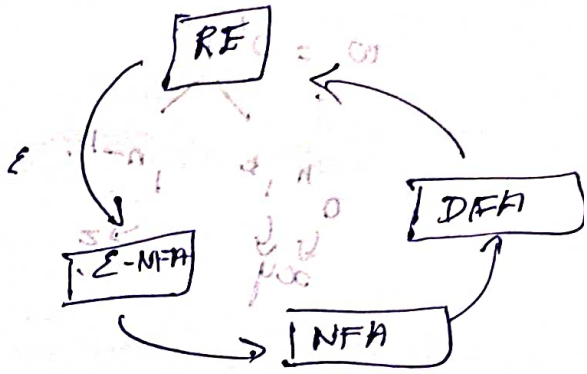
NFA – DFA:

-> ε - closure of 1 state

complexity - $O(n^2)$

-> ε - closure of n state

complexity - $O(n^3)$



->Each set of DFA is powerset of NFA (2^n)

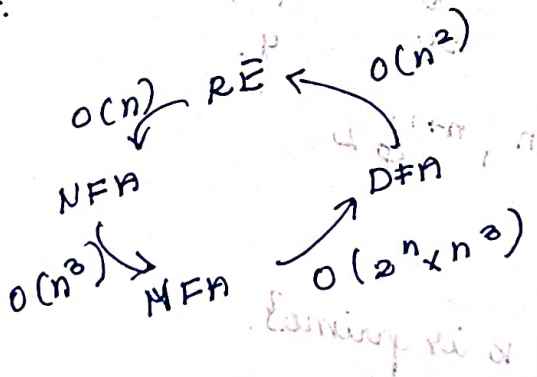
There are n state of DFA

pumping lemma:

FA → RE

not regular

$w = x y^2 z$.



prove using contradiction:

assume it regular. & split the string, check whether same else prove not regular.

1> $|xy| \leq n$

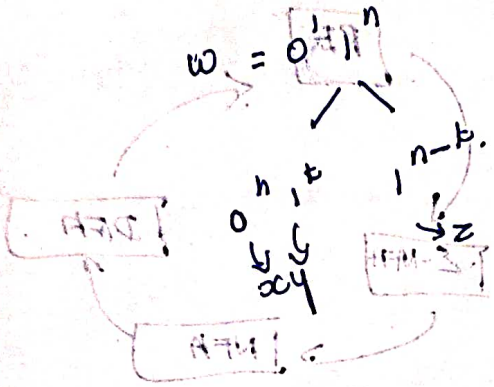
2> $|y| > 0$

3> for all $i \geq 0$, $xy^i z$ is in L

Using pumping lemma:

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

① $L = \{0^n 1^n \mid n \geq 1\}$



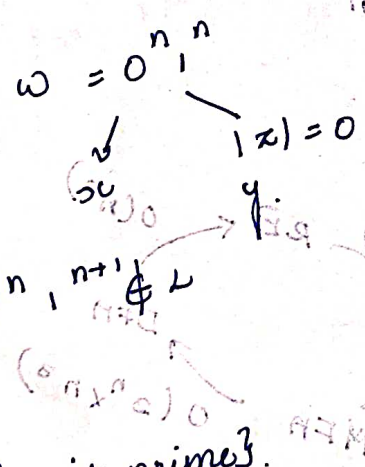
$\{0^n 1^k \mid n \geq 1\}$

no of 0 = no of 1

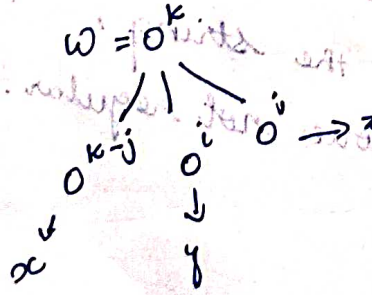
$= 0^{n-k} 1^{n-k}$

$= 0^n 1^{n+1} \notin L$

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



② $L = \{0^n \mid n \text{ is prime}\}$.



$0^{k-i-j}\ 0^{i+1}\ 0^j$

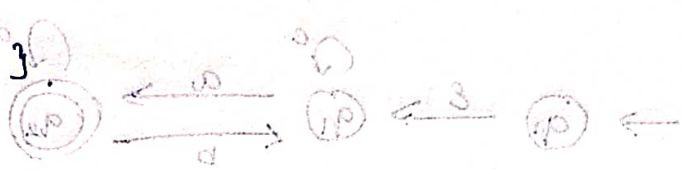
$0\ 0^{k+1} \notin L$.

1001 => palindrome

$w \in \{0,1\}$

3) ww^r is not regular.

4) $\{0^{i,j} \mid i > j\}$.



② ww^r

$w = 001$

$w^r = 100$

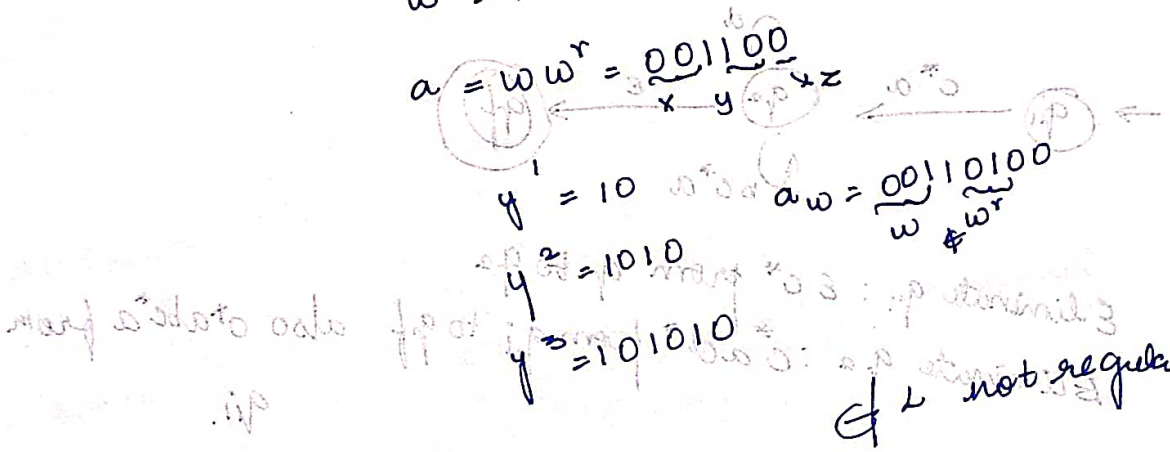
$a = ww^r = \underbrace{00}_{x}\underbrace{1}_{y}\underbrace{100}_{z}$

$y^1 = 10$ $\quad aw = \underbrace{001}_{w}\underbrace{10100}_{w^r}$

$y^2 = 1010$

$y^3 = 101010$

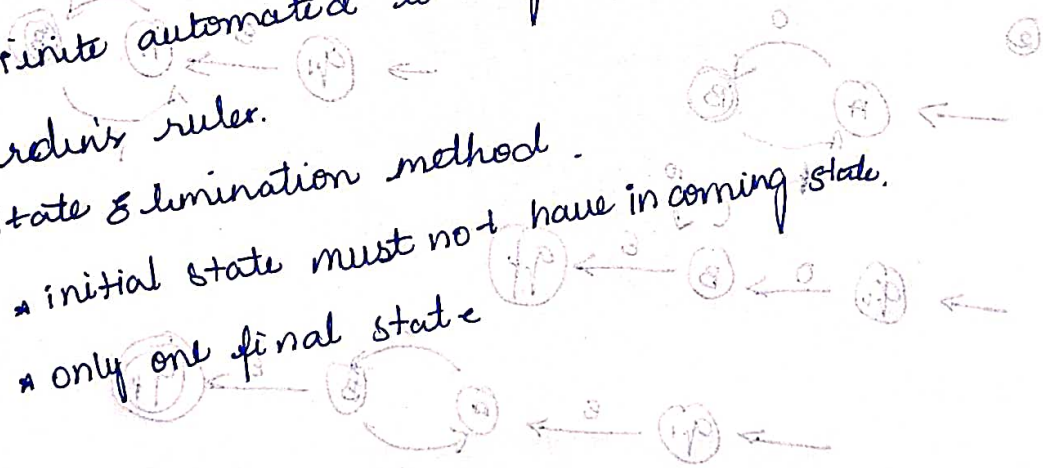
$\notin L$ not regular



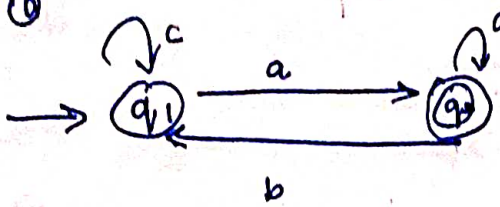
FA to RE

Finite automated to Regular Expression.

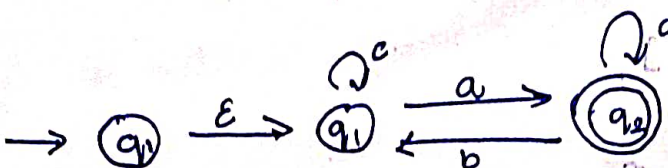
* Arden's rule.
* State Elimination method.
  * initial state must not have incoming state.
  * only one final state



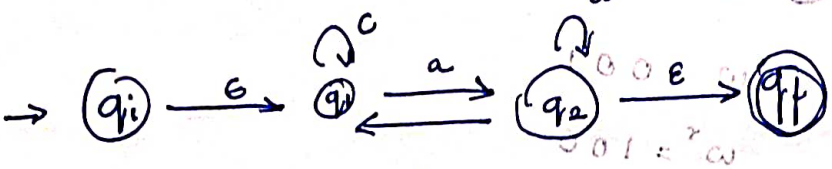
①

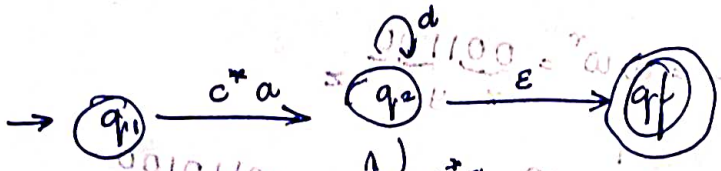


i) create initial state q_i



ii) create a new final state q_f.



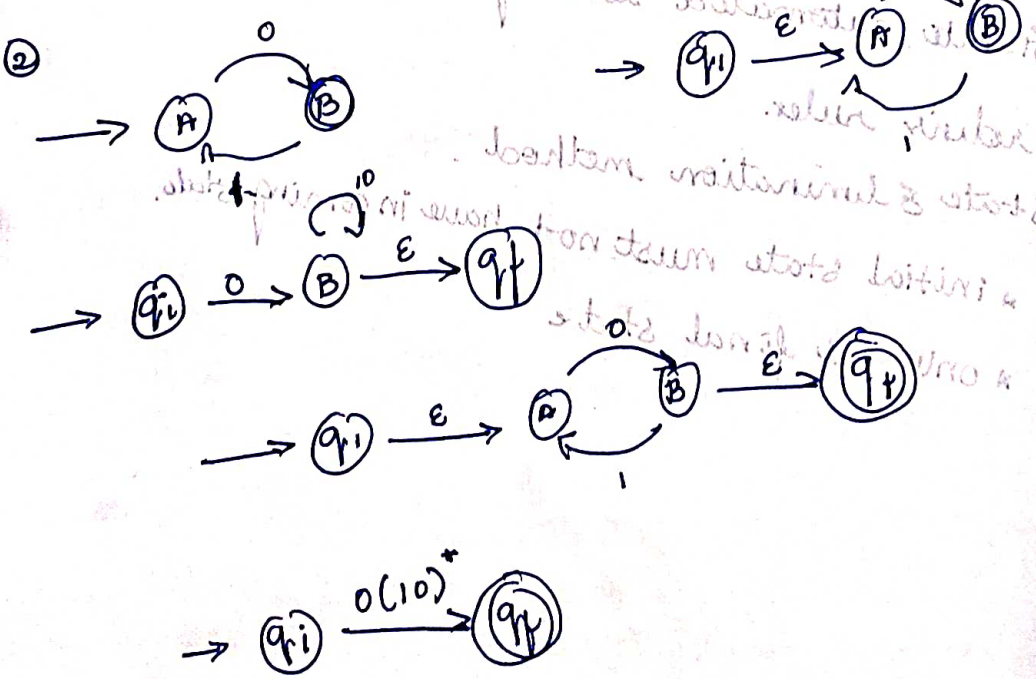


Eliminate q_1 : εc^* from q_i to q_2

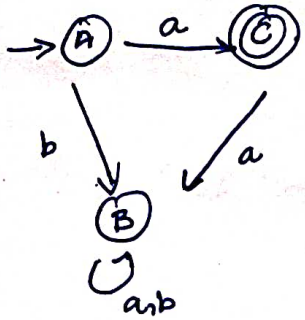
Eliminate q_2 : c^*ad^* from q_i to q_f also c^*abc^*a from q_i to q_i.

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

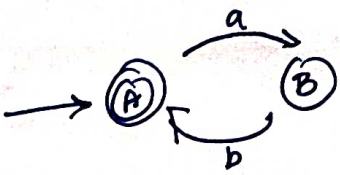
②



Example: 1



Example: 2



statement

RE

ε NFA

εNFA – DFA.

string

traverse

output

2 RE

find $\cup, \cap, |$, complement reversal.

Derive the RE