ch-6. Transition Graphs.

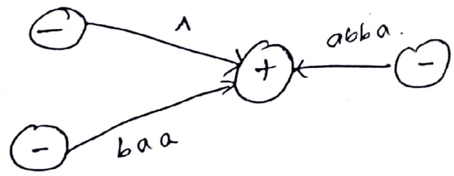
Def^n:- TG is a collection of three things:

1). A finite set of states, atleast one of which is designated as the start state (-) & ~~some~~ some (maybe none) of which are designated as final states (+).

2). An alphabet Σ of possible input letters from which input strings are formed.

3). A finite set of transitions (edge labels) that show how to go from some states to some others based on reading specified substrings of input letter (possibly even the null string Λ).

egs.



Machine that accepts only baa and abba.

Comparison with finite Automata:-

1). TG can read substrings as inputs (eg:- abba above).

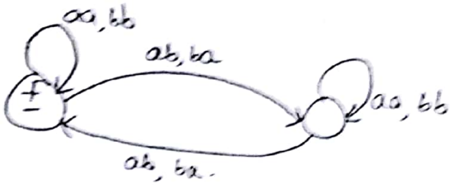
2). Λ can be read as input.

3). start state can be more than one in TG. whereas in finite Automata it ~~can be~~ should be only one.

4). Not necessary that each state displays all outgoing edges for every Σ word.

eg TG that shows all words in which a's occur only in even clumps & that end in three or more b's.

TG for EVEN-EVEN.



Generalized Transition Graph (GTG)

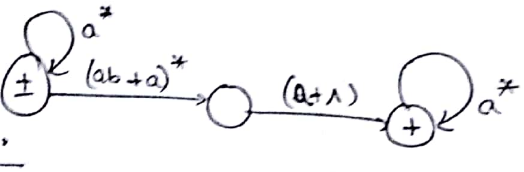
GTG is a collection of three things:

1). A finite set of states, of which at least one is a start state and some (maybe none) are final states.

2). An alphabet Σ of input letters.

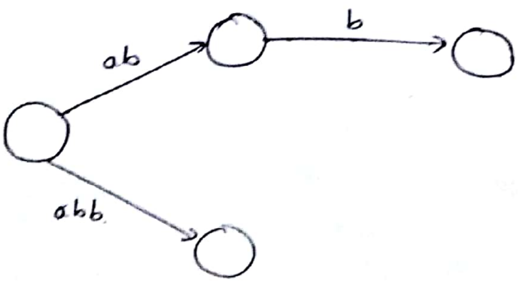
3). Directed edges connecting some pairs of states, each labeled with a regular expression.

eg:- Machine that accepts all strings without a double 'b'.



Non-Determinism.

eg:-



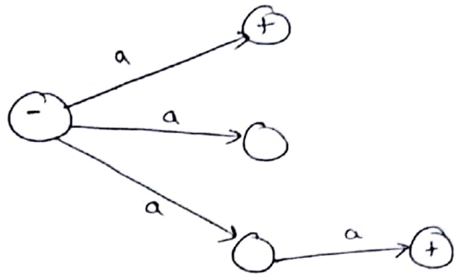
If 'ab' is read as input then 'which way to select' is the issue. It is not definite. Thus, the machine is non-deterministic. Human choice becomes a factor in selecting the path.

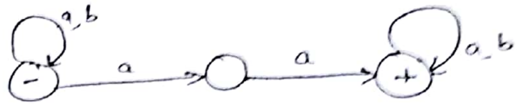
Scanned by CamScanner

Non-Deterministic finite Automata

A NFA is a TG with a unique start state with the property that each of its edge labels is a single alphabet letter.

egs.

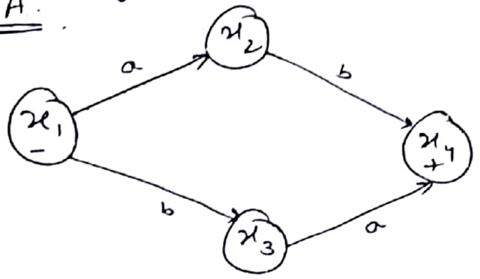




Converting NFA's to FA's.

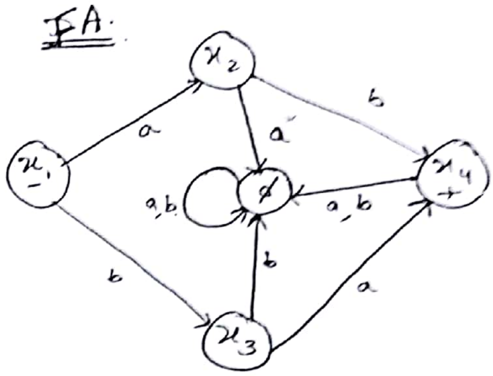
N.FA.

(1)

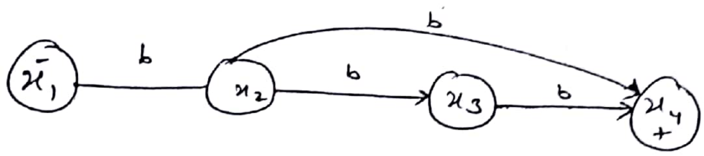


⟹

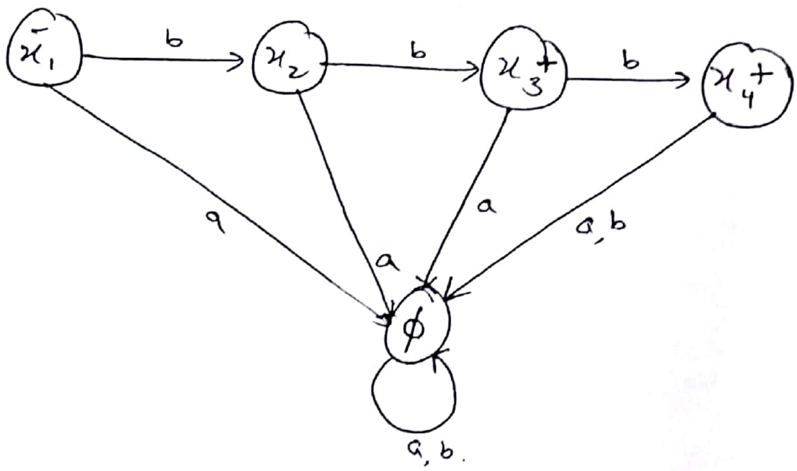
FA.



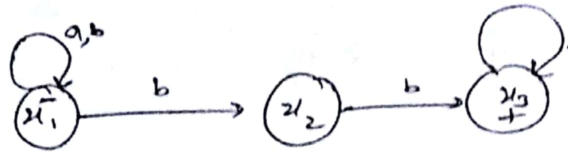
(2) NFA that accepts the lang {bb bbb} is,



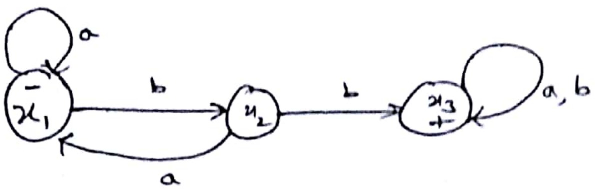
FA.



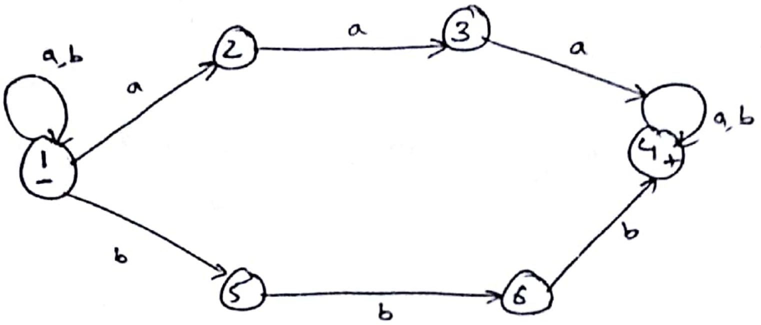
(3). NFA that accepts all inputs with a bb in them is



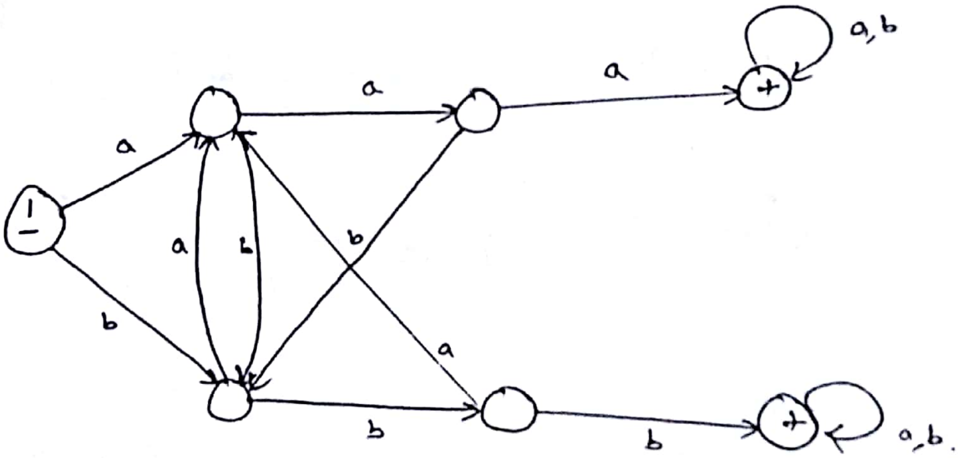
FA.



(4). NFA that accepts all inputs with a triple letter is

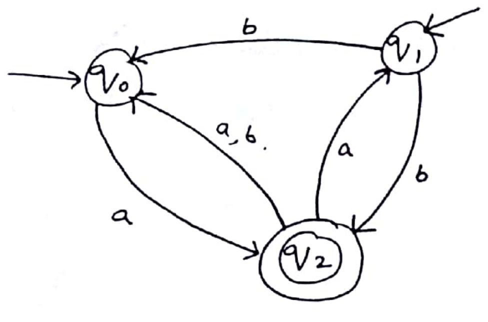


FA.



∴ Conversion of Non-Deterministic Finite Automata to DFA.

eg-1. Convert foll. NDFA to DFA.



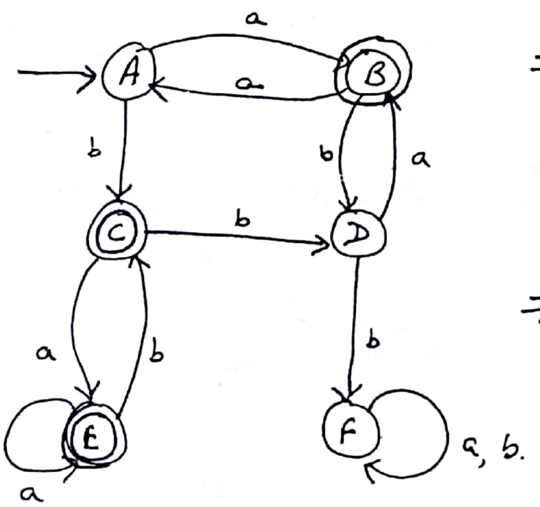
transition Table.

	a	b
q_0	q_2	ϕ
q_1	ϕ	$[q_0, q_2]$
q_2	$[q_0, q_1]$	$\{q_0\}$

Sol^n find successor table, start with 'start states' ie $[q_0, q_1]$

		a	b.
A	$[q_0, q_1]$	q_2	$[q_0, q_2]$
B	q_2	$[q_0, q_1]$	$[q_0]$
C	$[q_0, q_2]$	$[q_0, q_1, q_2]$	$[q_0]$
D	$[q_0]$	q_2	ϕ
E	$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$	$[q_0, q_2]$.
F	ϕ	ϕ	ϕ

DFA [Deterministic finite Automata]



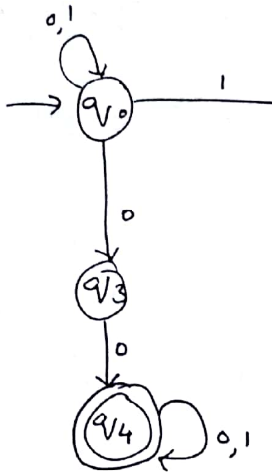
⇒ finate state was q_2 [NDFA]

∴ In DFA, final states are:

B, C, E

⇒ start state is A [set of start state in NDFA ie $[q_0, q_1]$

eg-2. convert NDFA to DFA.



Transition Table.

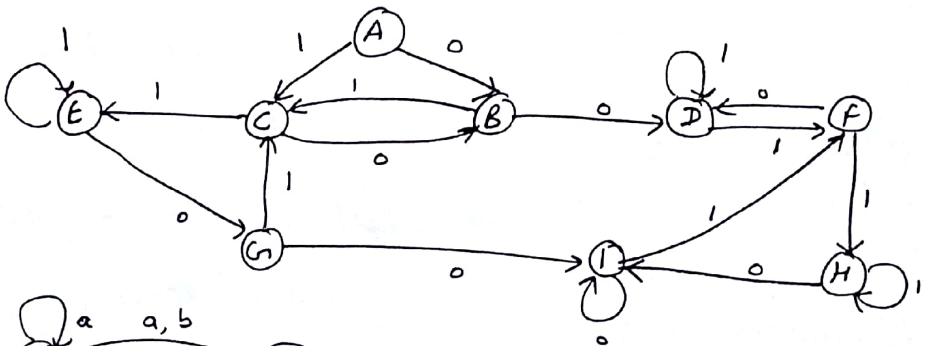
states	0	1
q_0	$[q_0, q_3]$	$[q_0, q_1]$
q_1	-	$[q_2]$
q_2	q_2	$[q_1]$
q_3	q_4	-
q_4	q_4	$[q_4]$

successor Table.

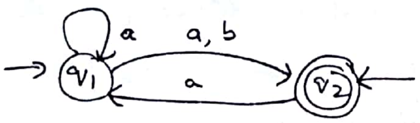
	states	0	1
A	q_0	$[q_0, q_3]$	$[q_0, q_1]$
B	$[q_0, q_3]$	$[q_0, q_3, q_4]$	$[q_0, q_1]$
C	$[q_0, q_1]$	$[q_0, q_3]$	$[q_0, q_1, q_2]$
D	$[q_0, q_3, q_4]$	$[q_0, q_3, q_4]$	$[q_0, q_1, q_4]$
E	$[q_0, q_1, q_2]$	$[q_0, q_3, q_2]$	$[q_0, q_1, q_2]$
F	$[q_0, q_1, q_4]$	$[q_0, q_3, q_4]$	$[q_0, q_1, q_2, q_4]$
G	$[q_0, q_3, q_2]$	$[q_0, q_3, q_4, q_2]$	$[q_0, q_1]$
H	$[q_0, q_1, q_2, q_4]$	$[q_0, q_3, q_2, q_4]$	$[q_0, q_1, q_2, q_4]$
I	$[q_0, q_3, q_2, q_4]$	$[q_0, q_3, q_2, q_4]$	$[q_0, q_1, q_4]$

start state → A

final state → D, E, F, G, H, I

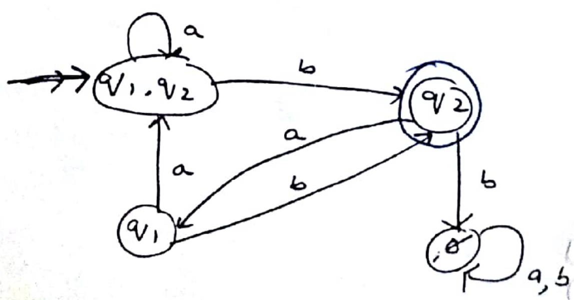


Q3.

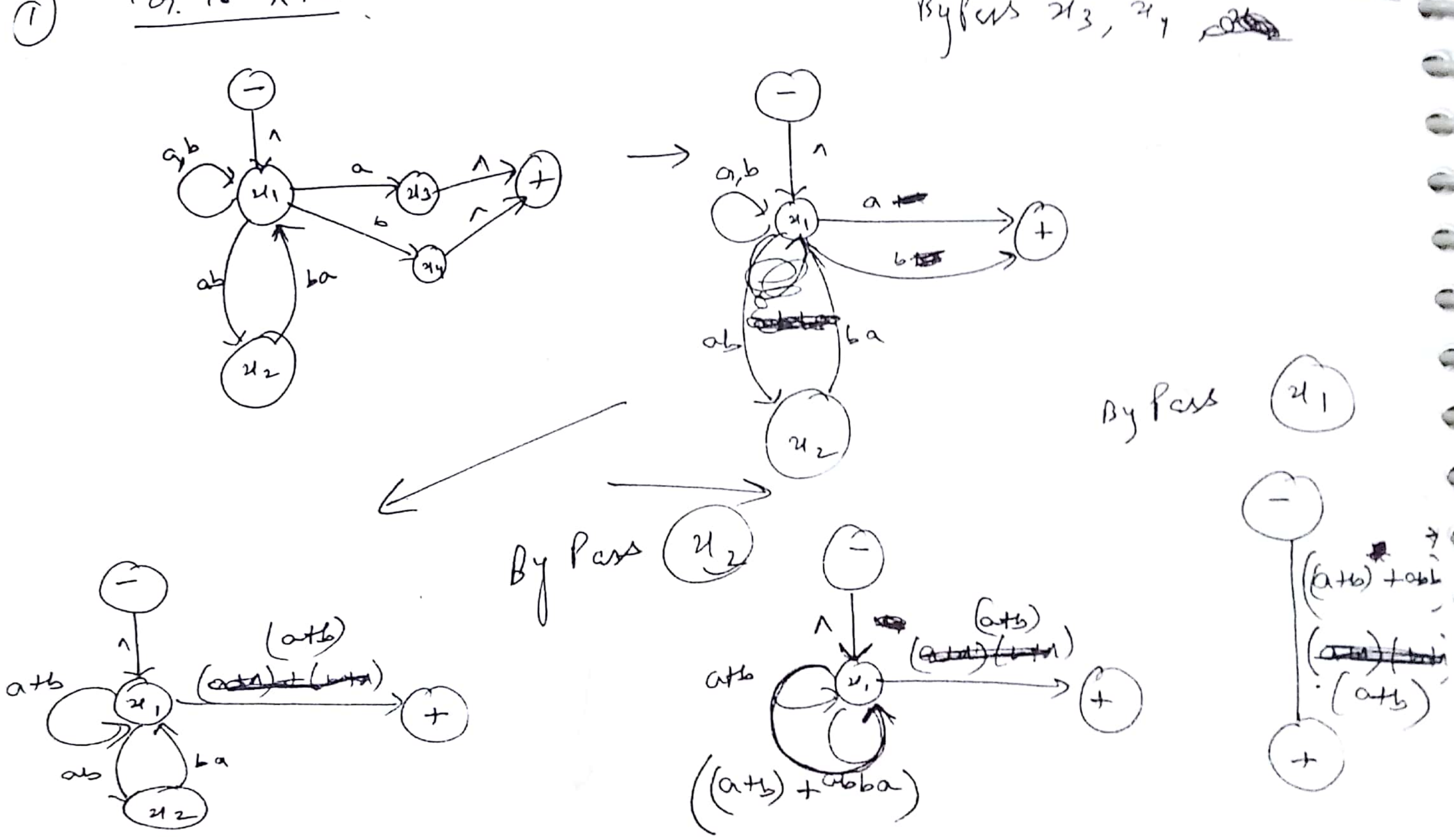


successor table

states	a	b
$[q_1, q_2]$	$[q_1, q_2]$	q_2
q_2	q_1	ϕ
q_1	$[q_1, q_2]$	$[q_2]$

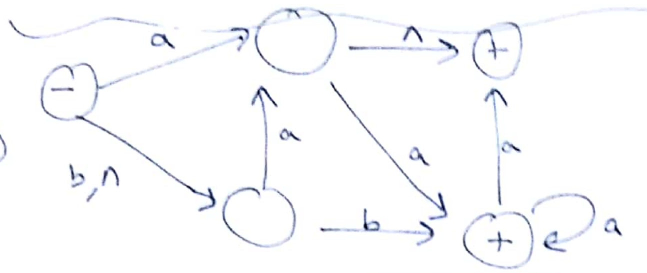


① TG. To R.E.

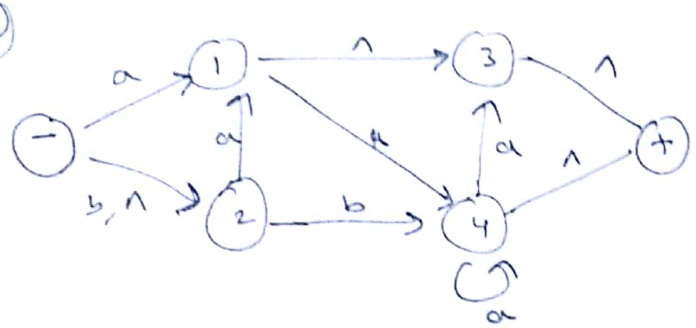

By Pass x_3, x_4
a,b
x_1
a
x_3
b
x_4
ab
ba
x_2
+
a
b
By Pass x_2
By Pass x_1
a+b
(a+b)
((a+b)+abba)
(a+b)

Convert TG into RE.

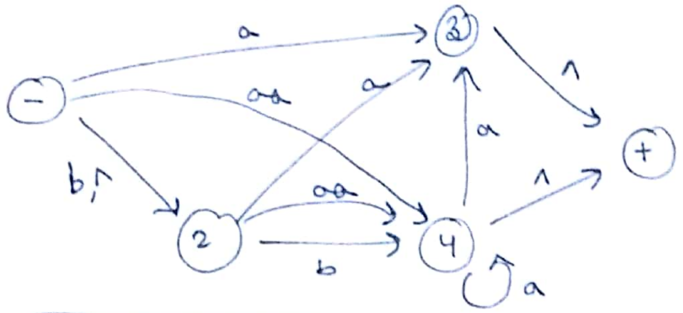
①

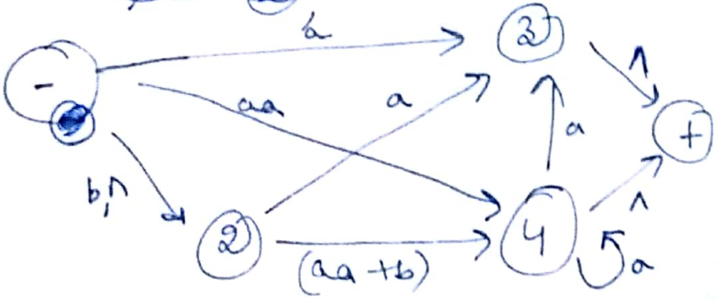


②

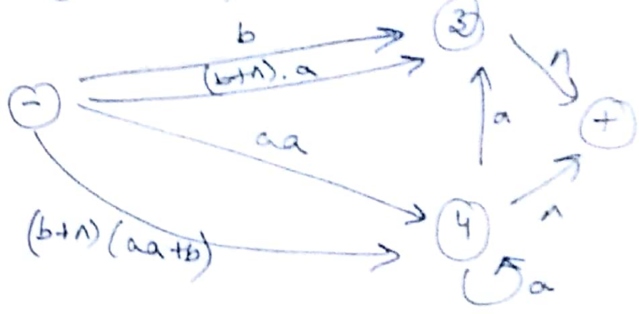


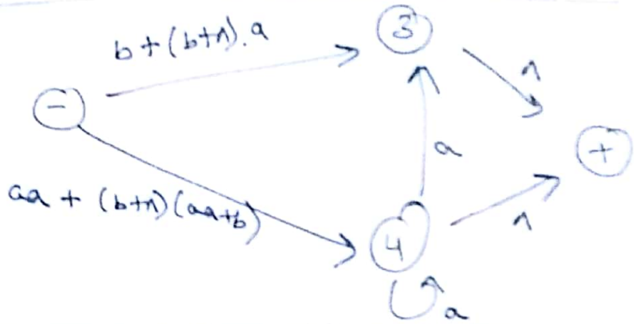
By Pass ①,



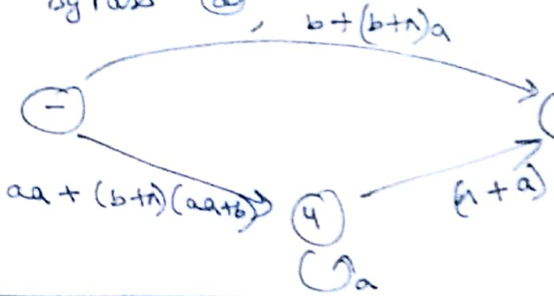


By Pass ②,

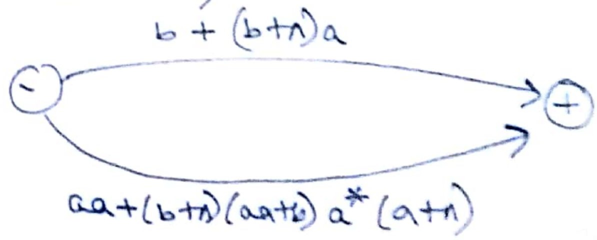




By Pass ②,



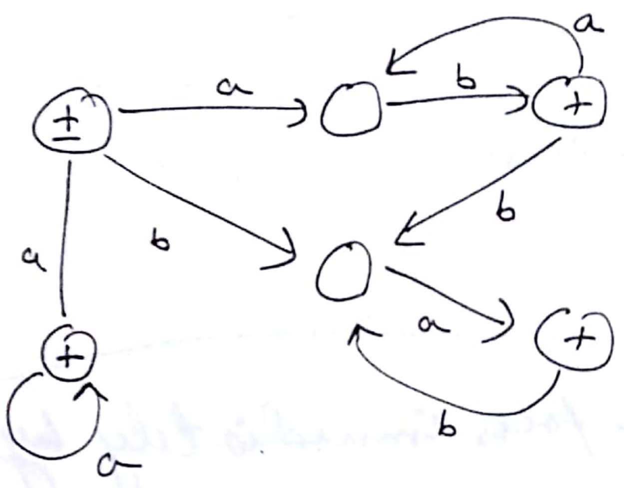
By Pass ④,



ab (misprinted (ab))

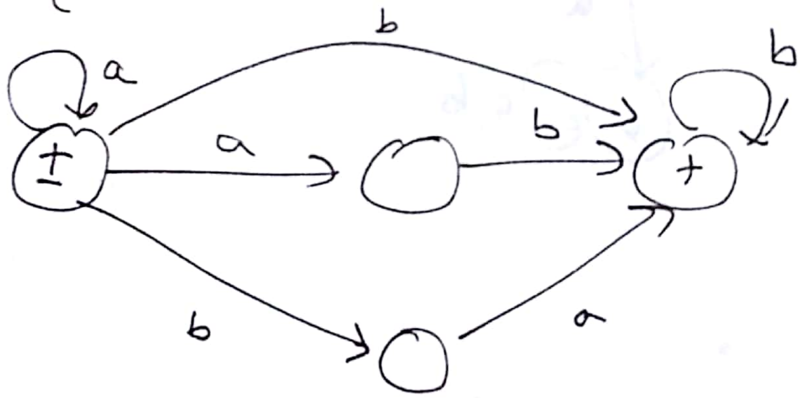
(3) NFA for

$(ab)^*(ba)^* + aa^*$



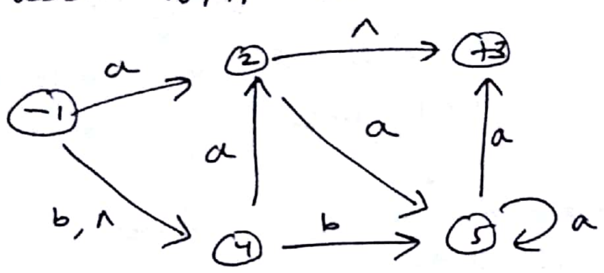
(4) construct FA for

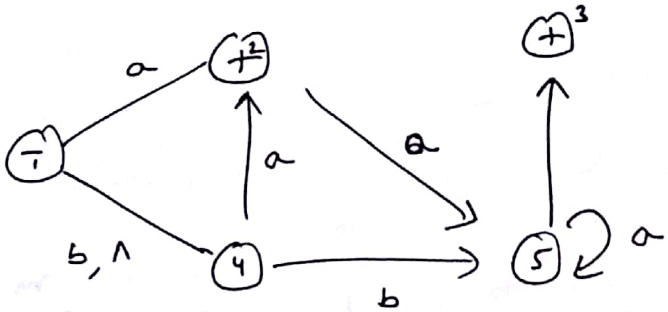
$a^*(ab + ba + \Lambda)b^*$

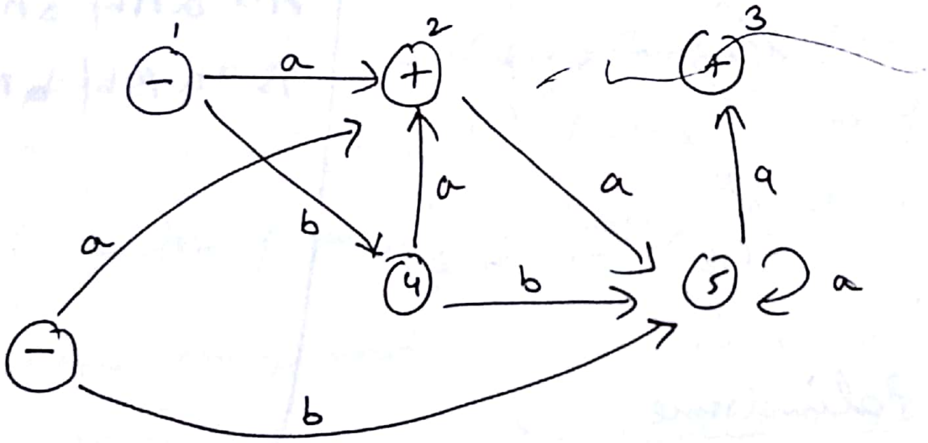


Now convert NFA → DFA

Q Convert NFA to DFA







Now convert NFA to DFA.

Q

$L_1 = (b + ab)^* (a + \Lambda)$

$L_2 = (a+b)^* aa (a+b)^*$ [atleast 2 a's together]

$L_1 \cap L_2 = \Lambda$

for FA → check Previous mail.