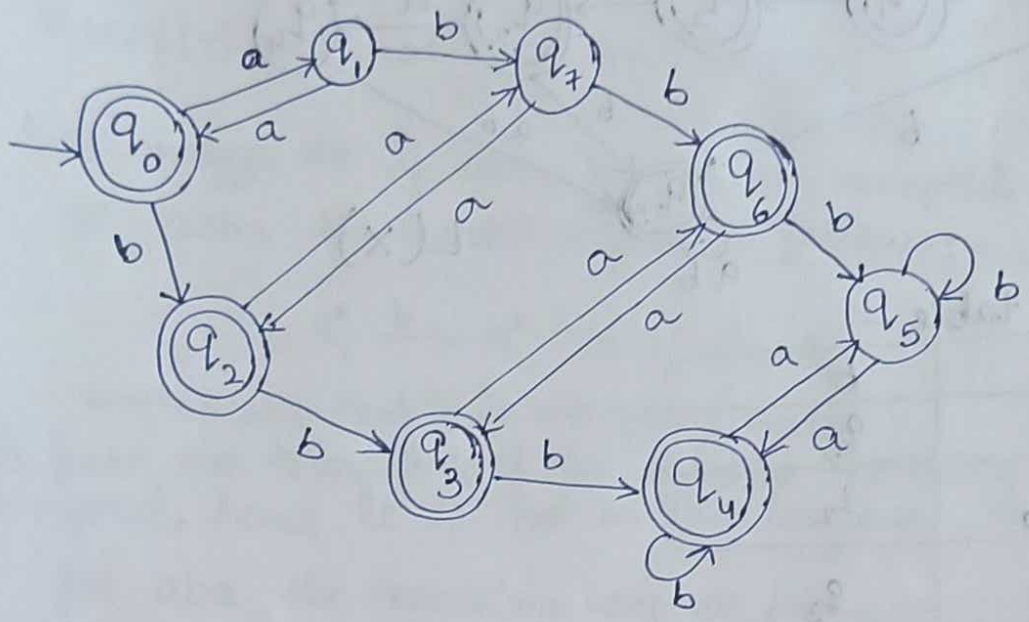
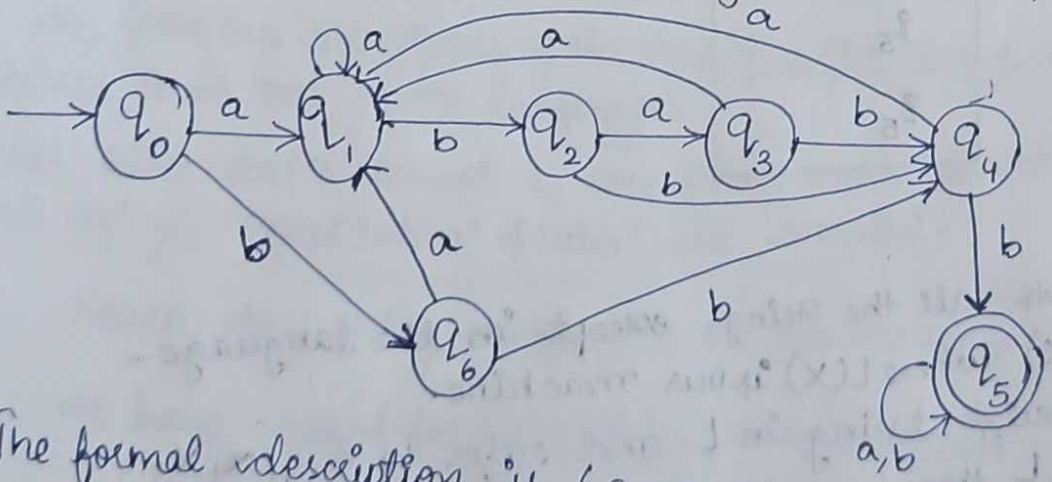
MANOGNA YADLAMUDI
CWID: A20551908

1. Construct DFA's over {a, b}:

(a) {w | w contains exactly two b's or an even number of a's}



(b) {w | w contains as substring ababb and/or bbb}



The formal description is $(Q, \Sigma, \delta, q_0, F)$,

$Q \Rightarrow$ set of states $\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$

$\Sigma \Rightarrow$ input alphabets $\{a, b\}$

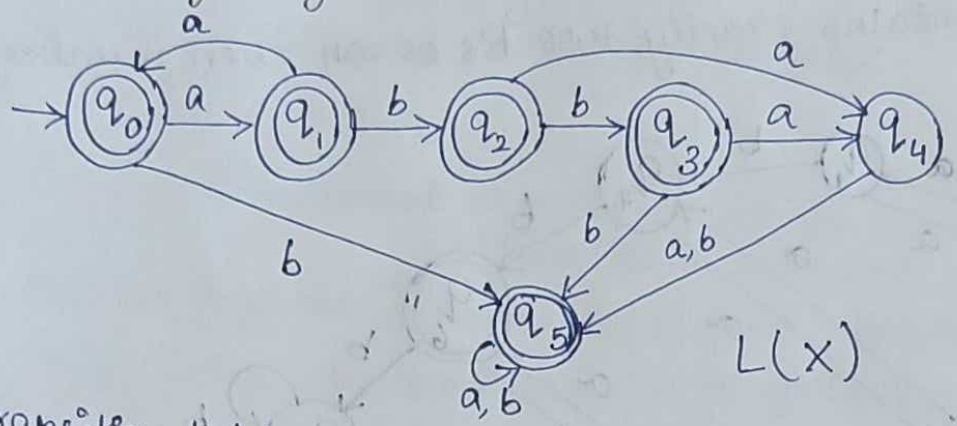
$\delta \Rightarrow$ transition function

$q_0 \Rightarrow q_0$ is start state

$F \Rightarrow$ final state $\{q_5\}$

δ	a	b
q_0	q_1	q_6
q_1	q_1	q_2
q_2	q_3	q_4
q_3	q_1	q_4
q_4	q_1	q_5
q_5	q_5	q_5
q_6	q_1	q_4

(c) $\{w / w$ is any string except the two strings abba and aba$\}$



transition table –

δ	a	b
q_0	q_1	q_5
q_1	q_0	q_2
q_2	q_4	q_3
q_3	q_4	q_5
q_4	q_5	q_5
q_5	q_5	q_5

Proof –

(1) DFA accepts all the strings ~~except~~ in the language –

$L \subseteq L(X)$ where $L(X)$ is our machine.

→ characterize string in L and refer to δ of X.

if $w \in L$, then

- $w \neq$ abba and aba

for $w \neq$ abba ⇒ any strings but not 'abba' are accepted

In this case, the string 'abba' will end in state q_4 of the machine. state q_4 is not the accepting state. Thus, $L(X)$ has all strings but not 'abba'.

for $w \neq$ aba ⇒ any strings but not 'aba' are accepted

In this case, string 'aba' will end in state q_4 of the machine. state q_4 is not the accepting state. Therefore $L(X)$ has all strings but not 'aba'.

Hence, we can say that both conditions arrive to state q_4, so, all strings but not the two are accepted.

we proved that any string in L is accepted by X.

(2) Any string accepted is in the language.

§ $\varepsilon \in L(X)$ & $\varepsilon \in L$

$\forall w \in L(X)$ such that $w \neq$ abba and aba

$L(X) \neq$ abba and aba

lets analyze the if abba & aba are accepted or not.

for abba the transitions are as follows:

$$q_0^* \xrightarrow{a} q_1^* \xrightarrow{b} q_2^* \xrightarrow{b} q_3^* \xrightarrow{a} q_4$$

* → indicates that it is accepting state

So, from our transitions of the machine the string is not accepted, hence it is not in the language.

for aba the transitions are as follows:

$$q_0^* \xrightarrow{a} q_1^* \xrightarrow{b} q_2^* \xrightarrow{a} q_4$$

* → indicates that it is accepting state

So, from our transitions of the machine the string is not accepted hence it is not in the language.

All other states except q_4 are final states which implies all strings except 'abba' & 'aba' are accepted.

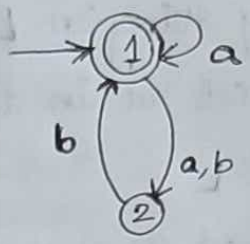
Hence, Any string accepted is in the language.

we have proved for both sides, therefore DFA recognizes exactly the specified language i.e,

$\{ w \mid w$ is any string except the two strings abba and aba$\}$

$\therefore L = L(X)$

2. Convert NFA to DFA



Steps:

① Convert NFA to its transition table
② Create DFA's start state
③ Create DFA's transition table
④ Create DFA's final state
⑤ Construct the DFA

Step 1 - Converting NFA to its equivalent transition table.

	a	b
→1*	1,2	2
2	–	1

'→' indicates start state
'*' indicates final state

Step 2 - Create DFA's start state

As 1 is NFA's start state
Start state for DFA is also 1

Step 3 - Create DFA's transition table

$\delta'(\{1\}, a) = \delta(1,a) = \{1,2\}$

$\delta'(\{1\}, b) = \delta(1,b) = \{2\}$

$\delta'(\{2\}, a) = \delta(2,a) = \phi \Rightarrow$ DFA doesnot contain empty transition therefore, lets introduce a dead state $\{3\}$.

$\delta'(\{2\}, a) = \delta(2,a) = \phi = \{3\}$

$\delta'(\{1,2\}, a) = \delta(1,a) \cup \delta(2,a) = \{1,2\} \cup \phi = \{1,2\}$

$\delta'(\{1,2\}, b) = \delta(1,b) \cup \delta(2,b) = \{2\} \cup \{1\} = \{1,2\}$

$\delta'(\{2\}, b) = \delta(2,b) = \{1\}$

$\delta'(\{\phi\}, a) = \delta(\phi,a) = \phi = \{3\}$

$\delta'(\{\phi\}, b) = \delta(\phi,b) = \phi = \{3\}$

Transition table

δ	a	b
→1	{1,2}	{2}
2	{3}	{1}
3	{3}	{3}
1,2	{1,2}	{1,2}

Step 4 - Create DFA's final state

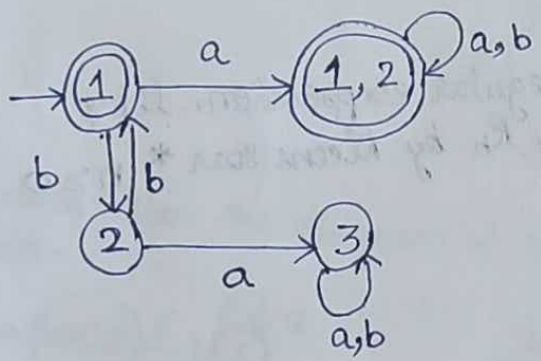
As 1 is NFA's final state, the final state for DFA is all states that have NFA's final state i.e, 1 state

The transition table can be written as -

'*' → indicates final state

δ	a	b
→1*	{1,2}	{2}
2	{3}	{1}
3	{3}	{3}
{1,2}*	{1,2}	{1,2}

Step 5 - Construct the DFA



3. Strings for RE -

1. $\Sigma^*00\Sigma^*1\Sigma^* \simeq (0+1)^*00(0+1)^*1(0+1)^* \simeq (0 \cup 1)^*00(0 \cup 1)^*1(0 \cup 1)^*$

Strings that are members of the language -

001, 100110, 1110001101

Strings that are not members of the language -

00, 1, 101

2. $(1 \cup 10 \cup 00)^*$

Strings that are members of the language -

ε, 110, 100

Strings that are not members of the language -

0, 10001, 01

3. $(111)^*$

Strings that are members of the language -

ε, 111, 111111111111

Strings that are not members of the language -

1, 0, 01

4. Regular Expression for the language

(a) $\{w \mid$ w contains exactly two b's, or an even number of a's$\}$

Regular expression -

$$b^*(ab^*ab^*)^* \cup a^*ba^*ba^*$$

Proof -

let L_n be the language, R_n be the Regular expression. let's define L_n as the language generated by R_n by kleene star $*$, $n \geq 0$ concatenations.

Base case:

$n = 0$

$R_0 = \varepsilon$

$L_0 = \{\varepsilon\} = \{w \mid w$ contains exactly two b's, or an even number of a's$\}$.

Inductive step:

let $L_k = \{w \mid w$ contains exactly two b's, or an even number of a's$\}$. $k \geq 0$

let L_{k+1} is a language generated by regular expression R_{k+1}

Here replace $*$ by $k+1$ concatenations.

$$R_{k+1} = b^*(ab^*ab^*)^{k+1} \cup a^*ba^*ba^*$$

lets check by breaking the Regular expression.

$b^*(ab^*ab^*)^{k+1}$ it accepts even number of a's.

$a^*ba^*ba^*$ it accepts only 2 b's.

Proof -

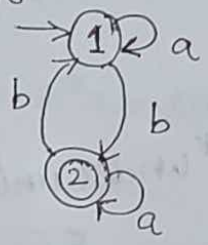
Here $b^*(ab^*ab^*)^{k+1}$ contains at least two a's or even occurances of a's and $a^*ba^*ba^*$ represents exactly two b's.

∴ It satisfies the condition for the language.

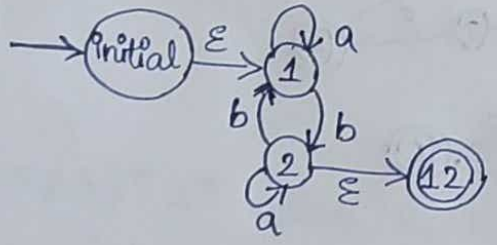
Here $K \geq 0$, $L_K = \{w / w$ contains exactly two b's or an even number of a's $\}$

∴ L_K is equal to R_6.

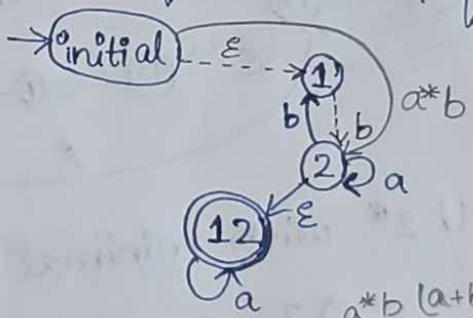
5. DFA to Regular Expression.



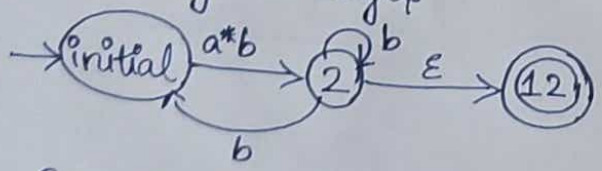
→ Normalize the diagram & add new state for initial & final state.



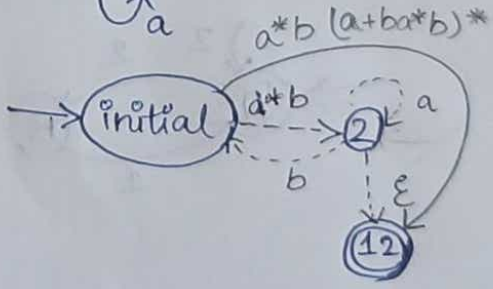
Remove state 1



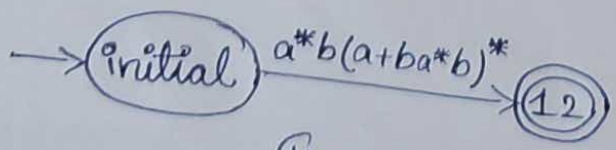
Removing old edges



Remove state 2



Removing old edges



The equivalent Regular expression for DFA is

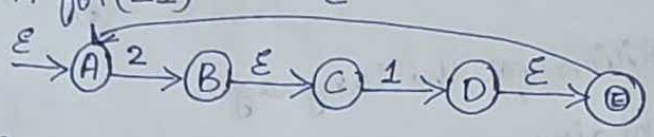
$a^*b(a+ba^*b)^*$

6. Construct NFA for Regular Expressions

1. $(21)^* \cup 2^*$

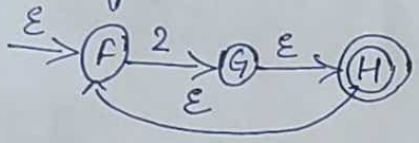
Step 1 -

NFA for $(21)^*$



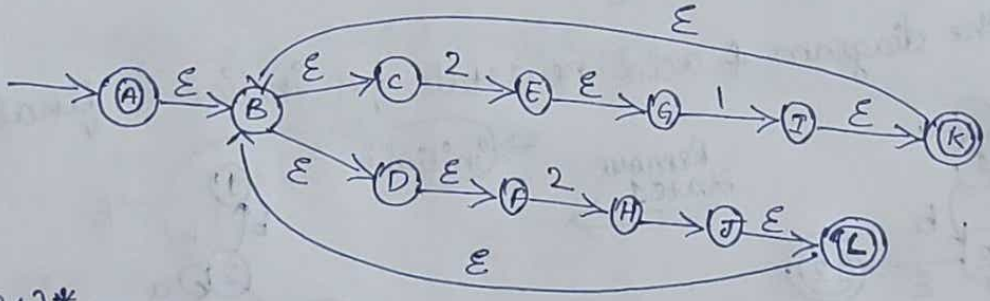
Step 2 -

NFA for 2^*

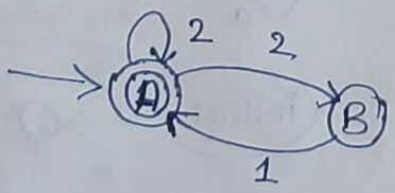


Step 3 -

combining $(21)^*$ with 2^* under union (OR) condition.



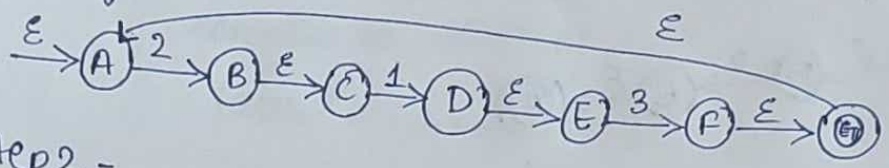
$(21)^* \cup 2^*$ with minimal 'ε' transitions.



2: $(213)^* 2^*$

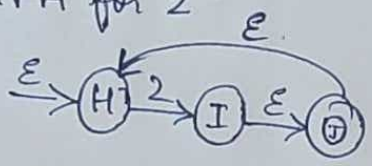
Step 1 -

NFA for 213^*.



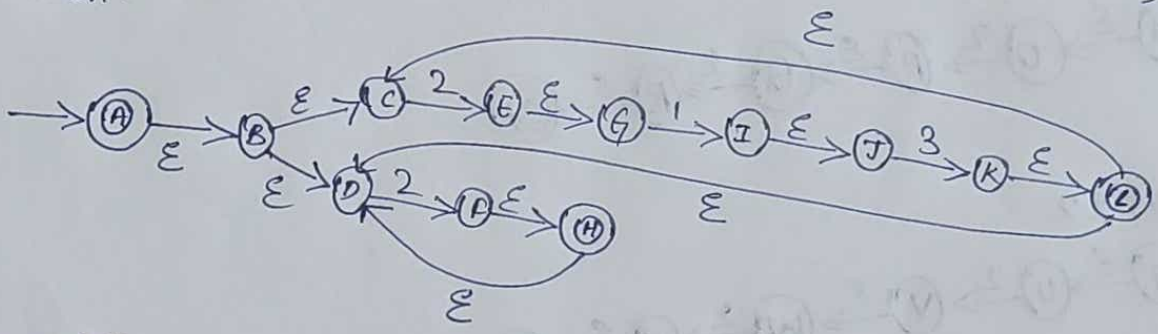
Step 2 -

NFA for 2^*.

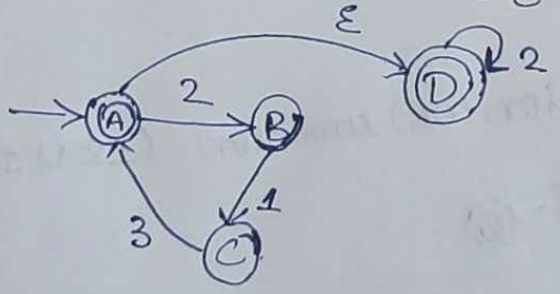


Step 3 -

Combining $(213)^*$ with 2^* under concatenation (AND) condition.



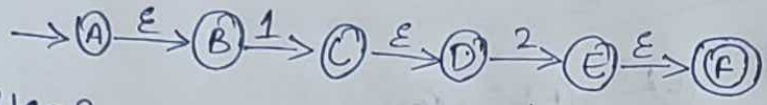
$(213)^* 2^*$ with minimal 'ε' transitions



3. $(12 \cup 11)^* (21 \cup 22)^*$

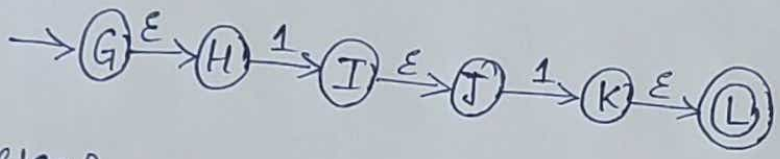
Step 1 -

~~12 → NFA~~ NFA for 12



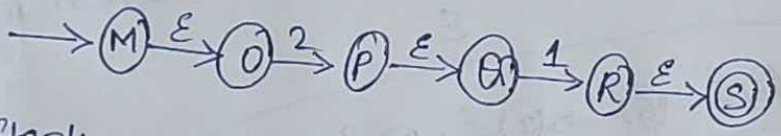
Step 2 -

NFA for 11



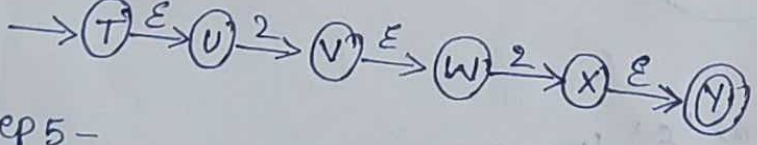
Step 3 -

NFA for 21



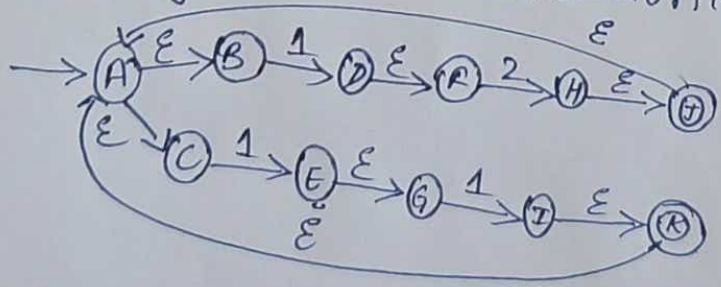
Step 4 -

NFA for 22



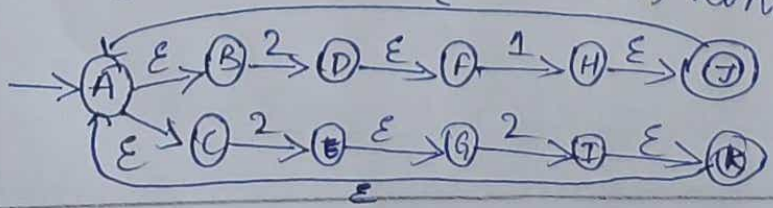
Step 5 -

combining 12 & 11 under union (OR) condition $(12 \cup 11)^*$



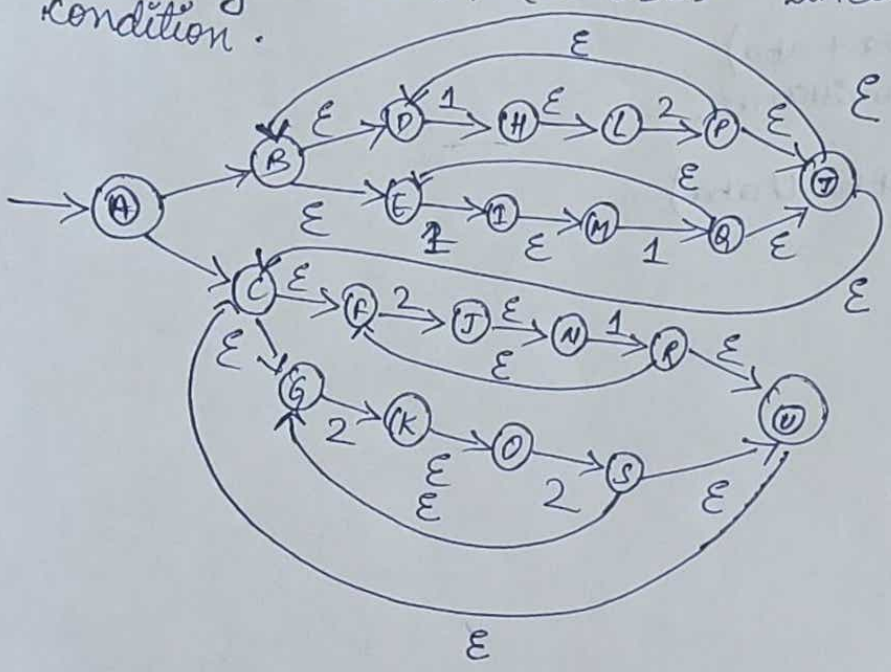
Step 6 -

combining 21, 22 under union (OR) condition $(21 \cup 22)^*$

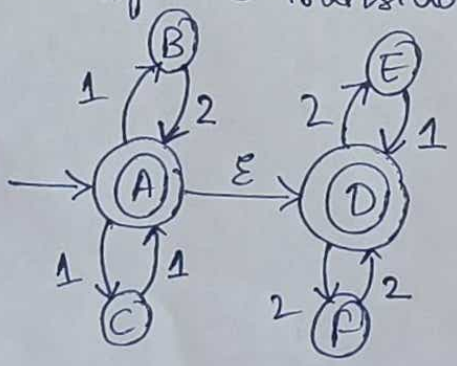


Step 7 -

combining $(12 \cup 11)^*$, $(21 \cup 22)^*$ under concatenation (AND) condition.



NFA with few 'ε' transitions



4. Regular Expression for the language

(b) $\{w \mid w$ contains as substring ababb and/or bbb $\}$

Regular Expression -

$$[[(a+b)^* ababb (a+b)^*) + ((a+b)^* bbb (a+b)^*)]^*$$

can also be written as,

$$[((a \cup b)^* ababb (a \cup b)^* \cup ((a \cup b)^* bbb (a \cup b)^*)]^*$$

can also be written as

$$[(\Sigma^* ababb \Sigma^*) \cup (\Sigma^* bbb \Sigma^*)]^*$$

(c) $\{w | w$ is any string except the two strings abba and aba$\}$

Regular expression-

$(a+b)^* - (abba + aba)$

can also be written as,

$(a \cup b)^* - (abba \cup aba)$