Page No.:
Date: / /

Unit 2 : Nondeterminism and Kleene's Theorem

* Non deterministic finite automata :

NFA stands for Non-determinstic finite automata.

NFA is a state machine consisting of states and transition that can either accept or reject a finite string.

It is easy to construct NFA than DFA for given regular language.

The finite automata are called NFA when there exist many paths for specific input from the current state to the next state.

Every NFA is not DFA, but each NFA can be translated into DFA.

NFA also has five states,

Q : finite set of states

Σ : finite set of input symbol

q_0 : initial state

F : final state

δ : Transition function.

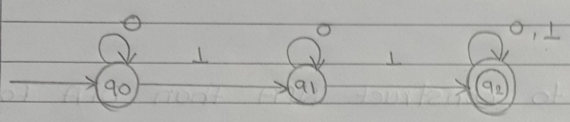
$$\delta : Q \times \Sigma \rightarrow 2^{Q}$$

The power to be several states at once.

In deterministic automata only single power or single transition.

String ending with 01 :
In deterministic finite automata (DFA)

① Only single transition function will be there

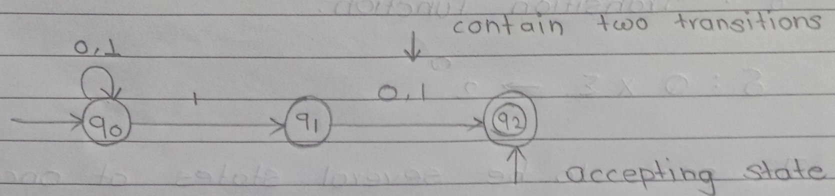


⑪ Instead of accepting state only single transition

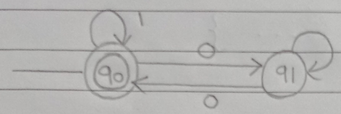
	0	1
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_2	q_2

In non deterministic automata,

It will contain two transition, or two arcs at a single state and accepting state most of the time should be empty.



* check weather Following is NFA / DFA



String acceptance / Reject

① 1
② 101
③ 1^*
④ 11001
⑤ 010011
⑥ $0^* 1^*$
⑦ 0^*
⑧ 1^*

state diagram

		0	1 ← alphabets
states	q_0	q_1	q_0
	q_1	q_0	q_1

$$\left.\begin{array}{l} ①\ \delta(q_0, 0) = q_1 \\ ②\ \delta(q_0, 1) = q_0 \\ ③\ \delta(q_1, 0) = q_0 \\ ④\ \delta(q_1, 1) = q_1 \end{array}\right\} \text{Transitions}$$

① 1^*

$1^* = \{1, 11, 111, 1111, \ldots\ldots\}$

a) $1 \Rightarrow \delta(q_0, 1) = q_0$

b) $11 \Rightarrow \delta(q_0, 1) = q_0$

$\delta(q_0, 1) = q_0$

$1^* = \{ \wedge, 1, 11, 111, 1111 \text{ ------} \}$

$= \{ 1, 11, 111, 1111, \text{ ------} \}$

1^* it is accepted

② 0^*

$0^* = \{ \wedge, 0, 00, 000, 0000 \text{ ------} \}$

$= \{ 0, 00, 000, 0000 \text{ ------} \}$

a) $0 \Rightarrow \delta(q_0, 0) = q_1$ ✗

b) $00 \Rightarrow \delta(q_0, 0) = q_1$

$\delta(q_1, 0) = q_0$ ✓

c) $000 \Rightarrow \delta(q_0, 0) = q_1$

$\delta(q_1, 0) = q_0$

$\delta(q_0, 0) = q_1$ ✗

d) $0000 \Rightarrow \delta(q_0, 0) = q_1$

$\delta(q_1, 0) = q_0$

$\delta(q_0, 0) = q_1$

$\delta(q_1, 0) = q_0$ ✓

0^* it is accepted by FA

③ 101

$\delta(q_0, 1) = q_0$

$\delta(q_0, 0) = q_1$

$\delta(q_1, 1) = q_1$

101 is not accepted.

④ 11001

$\delta(q_0, 1) = q_0$
$\delta(q_0, 1) = q_0$
$\delta(q_0, 0) = q_1$
$\delta(q_1, 0) = q_0$
$\delta(q_0, 1) = q_0$

11001 is accepted

⑤ 010011

$\delta(q_0, 0) = q_1$
$\delta(q_1, 1) = q_1$
$\delta(q_1, 0) = q_0$
$\delta(q_0, 0) = q_1$
$\delta(q_1, 1) = q_1$
$\delta(q_1, 1) = q_1$

010011 it is not accepted

⑥ $0^* 1^*$

$0^* = \{\Lambda, 0, 00, 000, \ldots\ldots\}$

$1^* = \{\Lambda, 1, 11, 111, \ldots\ldots\}$

$0^* 1^* = \{\Lambda, \Lambda 1, \Lambda 11, \Lambda 111, 0\Lambda, 01, 011, 0111, 00\Lambda, 001,$
$0011, 00111, 000\Lambda, 0001, 00011, 000111 \}$

$= \{ 1, 11, 111, 0, 01, 011, 0111, 00, 001, 0011, 00111, 000, 0001, 00011, 000111, ---- \}$

$0^* 1^* = \{ 0, 1, 00, 01, 11, ---- \}$

a) $0 \Rightarrow \delta(q_0, 0) = q_1$

b) $1 \Rightarrow \delta(q_0, 1) = q_0$

c) $00 \Rightarrow \delta(q_0, 0) = q_1$

$\delta(q_1, 0) = q_0$

d) $0001 \Rightarrow \delta(q_0, 0) = q_1$

$\delta(q_1, 0) = q_0$

$\delta(q_0, 0) = q_1$

$\delta(q_1, 1) = q_1$

Hence, it is not accepted by FA

Draw the state diagram using following table & string accepted by FA 110001 and 101000

State \ IP	0	1
$\rightarrow q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

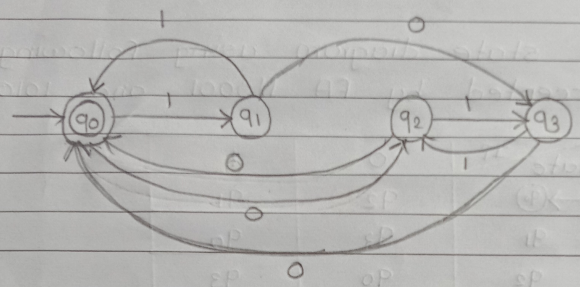
$$
\begin{aligned}
\delta(q_0, 110001) &= \delta(q_1, 10001) \\
&= \delta(q_0, 0001) \\
&= \delta(q_2, 001) \\
&= \delta(q_0, 01) \\
&= \delta(q_2, 1) \\
&= q_3
\end{aligned}
$$

It is not accepted by FA.

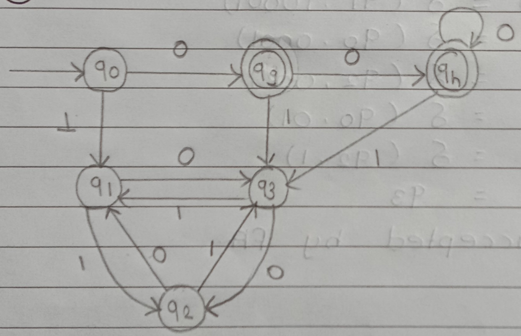
(2) 101000

$$
\begin{aligned}
\delta(q_0, 101000) &= \delta(q_1, 01000) \\
&= \delta(q_3, 1000) \\
&= \delta(q_2, 000) \\
&= \delta(q_0, 00) \\
&= \delta(q_2, 0) \\
&= q_0
\end{aligned}
$$

It is accepted by FA



②



$q_0 \Rightarrow$ Initial state

$Q = \{q_0, q_1, q_2, q_3, q_g, q_h\}$

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

$F = \{q_g, q_h\}$

Page No.:
Date: / /

Transition table

	0	1
$\rightarrow q_0$	q_g	q_1
q_1	q_3	q_2
q_2	q_1	q_3
q_3	q_2	q_1
q_g	q_h	q_3
q_h	q_h	q_3

Ex : ① 10101

$|s_1| = 5$

$\delta(q_0, 10101)$

$\delta(q_1, 0101)$

$\delta(q_3, 101)$

$\delta(q_1, 01)$

$\delta(q_3, 1)$

$= q_1$ Non final state

② $s_2 = 0^* 11$

$0^* 11 = \{\wedge 11, 011, 0011, 00011, \ldots\}$

$s_{12} = 11$

$\delta(q_0, 11)$

$\delta(q_1, 1)$

$= q_2$ Not accepting

Page No.:
Date: / /

③ $S_3 = 0^*$

$0^* = \{0, 00, 000\}$

$\delta(q_0, 0) = q_g$ Accepted

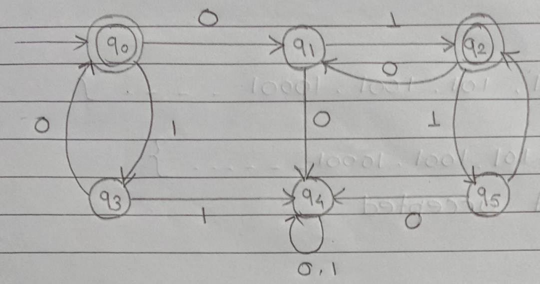


fig : state diagram

	0	1
→ q_0	q_1	q_3
q_1	q_4	q_2
q_2	q_1	q_5
q_3	q_0	q_4
q_4	q_4	q_4
q_5	q_4	q_2

fig : transition table

① 11111

$|s_1| = 5$

$$\begin{aligned}
\delta(q_0, 11111) &= \delta(q_3, 1111) \\
&= \delta(q_4, 111) \\
&= \delta(q_4, 11) \\
&= \delta(q_4, 1) \\
&= q_4
\end{aligned}$$

Not accepted.

Page No.:
Date: / /

② 10^*1

$S_2 = \{ 1\wedge1, 101, 1001, 10001 ----\}$

$= \{ 11, 101, 1001, 10001 ------\}$

It is not accepted

③ 0101

$|S_3| = 4$

$\delta(q_0, 0101) = q_2$

$\delta(q_0, 0101)$

$\therefore \delta(q_1, 101)$

$\therefore \delta(q_2, 01)$

$\therefore \delta(q_1, 1)$

$= q_2$ Accepting state

Lucky	Page No.:
Date : / /	

* NFA with Null Transitions

A nondeterministic finite automata with Λ (null) transitions [abbreviated NFA -Λ) is 5 tuple $(Q, \Sigma, q_0, A, \delta)$ where Q and Σ are finite sets $q_0 \in Q$, $A \subseteq Q$

$$\delta' : Q \times (\Sigma \cup \{\Lambda\}) \rightarrow 2^{Q}$$

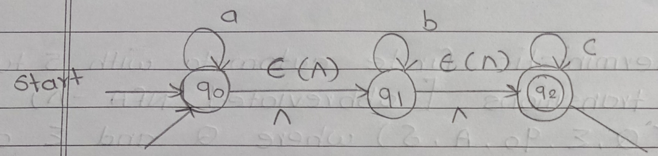
① It contain empty inputs and they called as null transitions.

② If the finite automata $M = \{Q, \Sigma, q_0, A, \delta\}$ is automata with Λ null transitions then auto--mata without M' null transitions will be $M' = \{Q, \Sigma, q_0, A', \delta'\}$

NFA with null transitions is allowed to make transition not only on input form the alphabet but also with null input. i.e., without any symbol. This transition without input is called null transitions.

Null transition makes the designing of NFA simpler as it reduces the complexity of the design and in some cases reduces the number of states too.

Ex :



initial state   accepting state

fig : automata with $\wedge$ (null) transitions

In above fig. it contain 4 inputs a, b, c with $\wedge$ so we construct transition table

status	a	b	c	$\wedge$
$\rightarrow q_0$	$\{q_0\}$	ϕ	ϕ	$\{q_1\}$
q_1	ϕ	$\{q_1\}$	ϕ	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$	ϕ

NFA with ϵ / $\wedge$ Epsilon Transitions

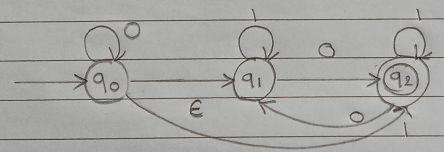
It allows FA to jump from one state to another state without consuming any input symbol

Add one column for ϵ moves

Easier for NFA transitions.

Epsilon Transition are those can be reached from state q by repeatelly making ϵ transition includ-ing itself.

Ex: Transaction Diagram



Transaction Table

State	Alphabets		
	0	1	ϵ
q_0	q_0	q_2	$\{q_0, q_1\}$
q_1	q_2	q_1	ϕ
q_2	q_1	q_2	ϕ

* Computational Tree for NFA

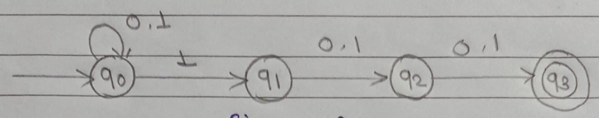
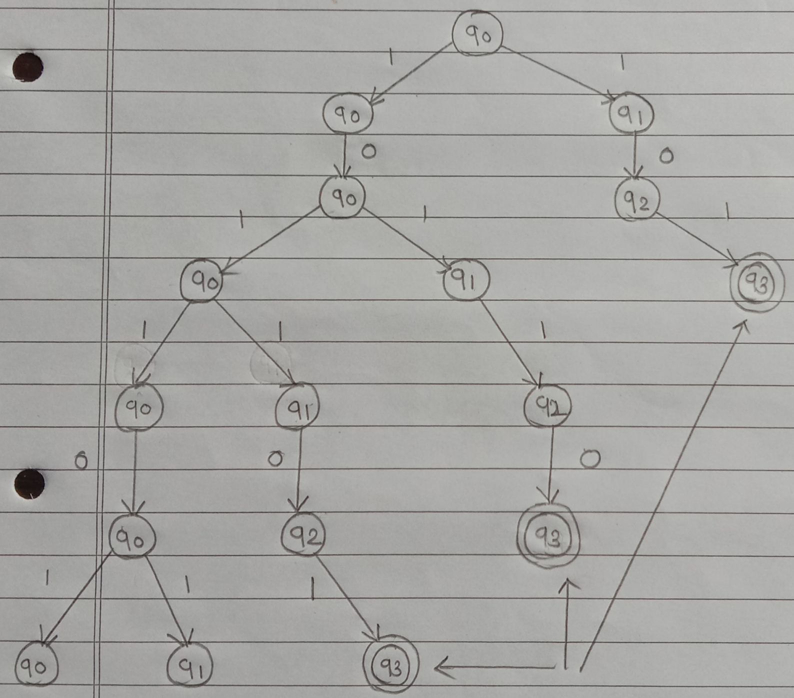


fig : NFA



Accepting states

Lucky	Page No.:
Date : / /	

* Difference between DFA and NFA

	DFA	NFA
①	DFA stands for Determ--inistic finite automata.	NFA stands for non-determin-istic finite automata.
②	Empty string transition, DFA can not use	NFA can use empty string transition.
③	DFA can be understood as one machine	NFA can be understood as multiple machine
④	DFA is more difficult to construct	NFA is easier to construct
⑤	Time needed for execut-ing input string is less	Time needed for executing input string is more.
⑥	All DFA are NFA	Not all NFA are DFA
⑦	DFA requires more space	NFA require less space
⑧	Dead state may be required	Dead state is not required
⑨	$\delta : Q \times \Sigma \rightarrow Q$ i.e., next possible state belongs to Q.	$\delta : Q \times \Sigma \rightarrow 2^Q$ ie, next possible state belongs to power set of Q

10	In DFA, the next possible state is distinctly set	In NFA, each pair of set and input symbol can have many possible next states.
11	DFA rejects the string in case it terminates in a state that is different from the accepting state.	NFA rejects the string in the event of all branches dying or refusing the string

Page No.:
Date: / /

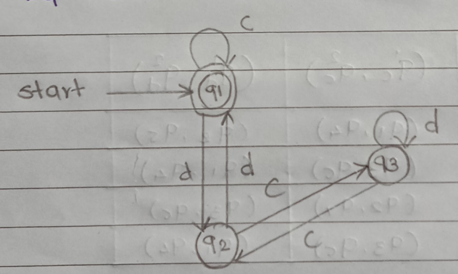
* Equivalance of FA's

Two FA's over ε are equivalent, if they accept the same set of string over ε.

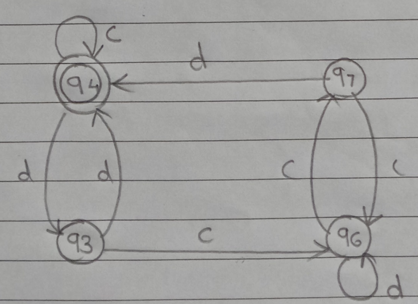
They are called to be non equal if they contain one transaction at x transaction reaches at final position and other does not.

The method called as comparision method.

Ex : Equivalent finite automata



automata M1



automata M2

Page No.:
Date: / /

In two automata M1 and M2, we have q_1 and q_4 as a accepting and initial state.

They contain transactions c and d at each level

Now in M1 and M2 we know that $\Sigma = \{c, d\}$

M1 having start state q_1 and
M2 having start state q_4

comparision table :

(q, q')	(q'_c, q^2_c)	(q'_d, q^2_d)
(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
(q_2, q_5)	(q_3, q_6)	(q_1, q_4)
(q_3, q_6)	(q_2, q_7)	(q_3, q_6)
(q_2, q_7)	(q_3, q_6)	(q_1, q_4)

Here the pairs in 2nd and 3rd row are identical or appear in comparision row so they are equivalent

* kleene's theorem , (Part 1)

Part 1, Moore (Machines)

Developed by moore in 1995 by scientiest Moore

Moore machine is 6 tuple , $(Q, \Sigma, q_0, \delta, \Delta, \lambda)$

Where (Q, Σ, q_0, δ) symbols from finite automata

Δ output (symbol) alphabet / Tape symbol

λ mapping from Q to Δ.
i.e., contain all outputs

If moore machine contain the symbols from $q_1, q_2, \ldots, q_n \; \forall \; n \geq 0$ then
$\lambda(q_1), \lambda(q_2), \ldots \lambda(q_n)$ will contain all the strings.

Ex : consider moore machine $M = (Q, \Sigma, \Delta, \delta, q_0, \lambda)$
Where,

$$Q = \{q_0, q_1, q_2, q_3\}$$

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

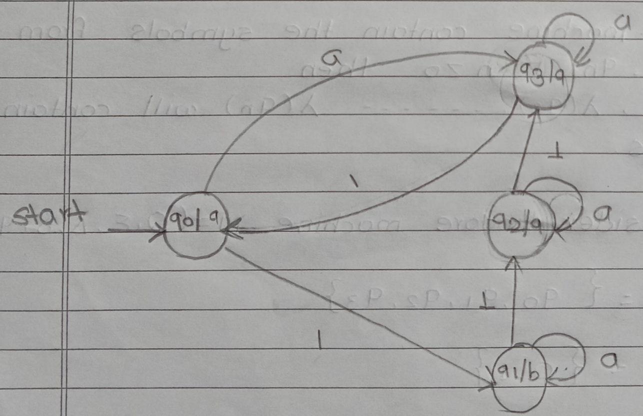
$$\Delta = \{a, b\}$$

Now from given data we can construct the moore machine as follows.

state	δ		output λ
	0	1	
q_0	q_3	q_1	a
q_1	q_1	q_2	b
q_2	q_2	q_3	a
q_3	q_3	q_0	a

Now we consider input string 0111 then transcation will be

$$q_0 \xrightarrow{0} q_3 \xrightarrow{1} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2$$



From the transition diagram the output string will be <u>aaaba</u>

Lucky Page No.:
Date: / /

* kleenes theorem

Any regular Language can be accepted by finite automata.

Proof :

If a language accepted by finite automata then it will accepted by also nondeterminstic finite automata with null transition.

It will contain the operations like union, kleene and concatenation.

If L_1 and L_2 are two regular Languages then by induction hypothesis, We get,

$L_1 \cup L_2$, L_1L_2 and L_1^*

Now consider if L_1 and L_2 are recognized NFA-Λ with M_1 and M_2 then

$M_i = (Q_i, \Sigma, q_i, A_i, \delta_i)$

Union :

$M_u = (Q_u, \Sigma, q_u, A_u, \delta_u)$

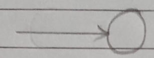
$\forall\ Q_u = Q_1 \cup Q_2 \cup \{q_u\}$

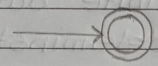
$A_u = A_1 \cup A_2$

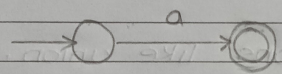
Here Q_u is accepted by either Q_1 or Q_2

$A_u = A_1 \cup A_2$

NFA - Λ for three basic regular language

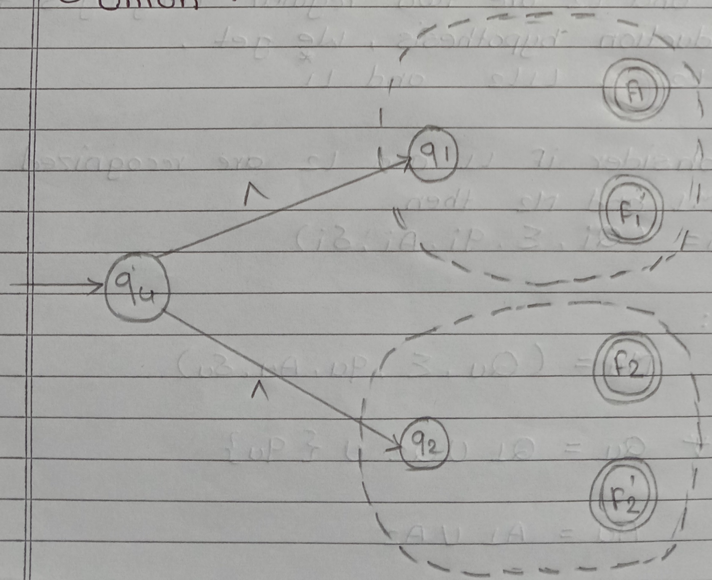
①  union

②  concatenation

③ 

kleene

① Union :



Page No.:
Date: / /

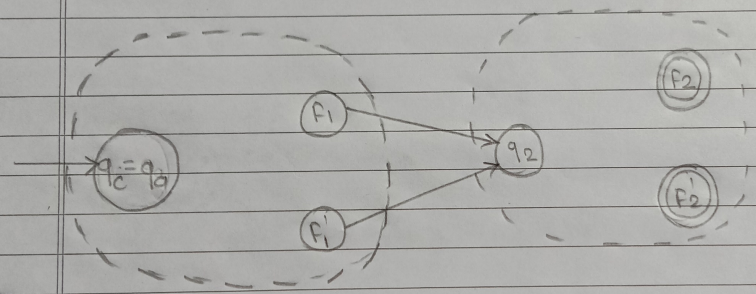
② Concatenation :

$M_c = (Q_c, \Sigma, q_c, A_c, \delta_c)$

Here no need of new state in it

$q_c = q_1$ initial state

$A_2 = A_c$ Accepting state



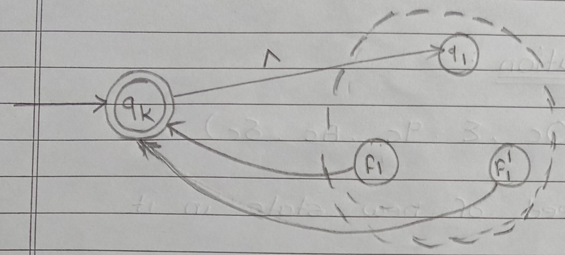
③ kleene :

$M_k = (Q_k, \Sigma, q_k, A_k, \delta_k)$

q_k new state not in Q_1

$Q_k = Q_1 \cup \{q_k\}$

$A_k = \{q_k\}$


Λ
q_1
q_k
f_1
f_1'

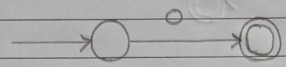
Ex : construct the NFA - Λ for ,

① $(00+1)^* (10)^*$

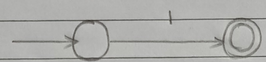
$(00+1)^*$

a) 00 $\Rightarrow$

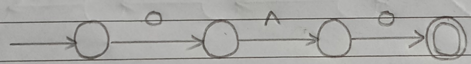
0 :



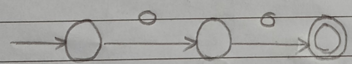
1 :



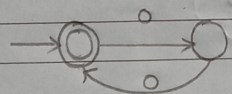
NFA - Λ for 00 ,



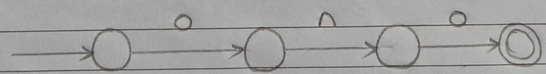
OR



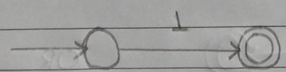
OR



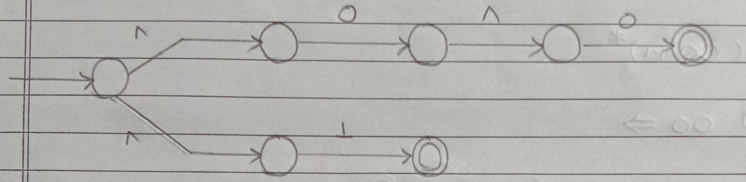
b) 00 :



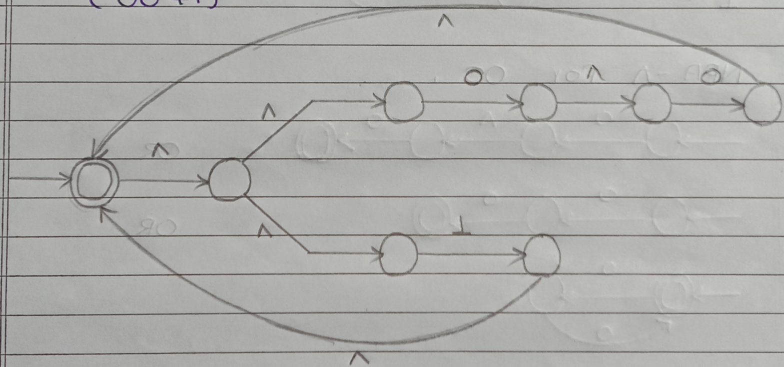
c) 1 :



d) (00+1)

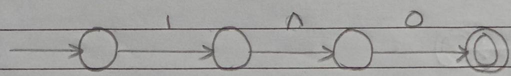


e) (00+1)*

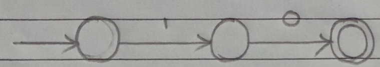


f) (10)*

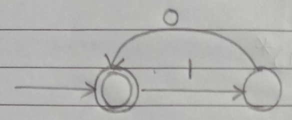
g) 10



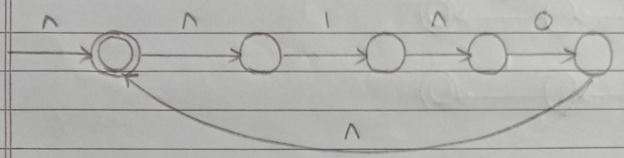
OR



OR



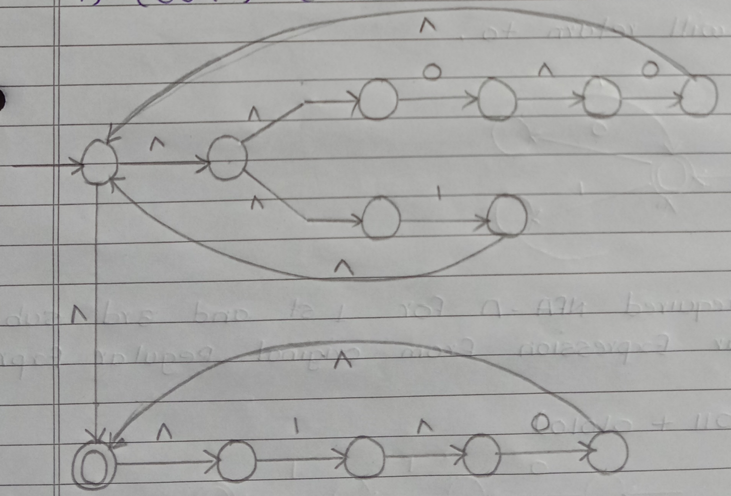
h) $(10)^*$



Now as we having 2 NFA - Λ we have to concatenate each other.

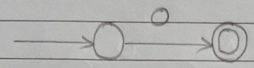
$$\underline{\underline{(00+1)^* (10)^* \neq (10)^* (00+1)^*}}$$

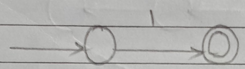
i) $(00+1)^* \cdot (10)^*$



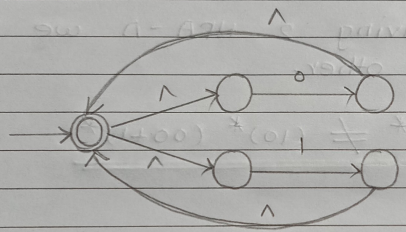
② $(0+1)^* \, (011+01010) \, (0+1)^*$

a) $(0+1)^*$

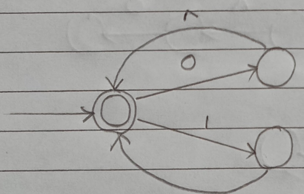
0 : 

1 : 

$(0+1)^*$:

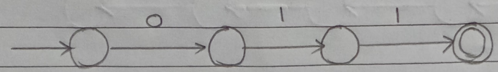


It will return to,

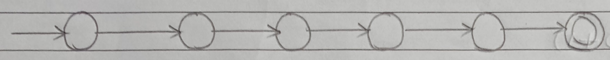


The required NFA-Λ for 1st and 3rd sub Regular Expression from original Regular Expression

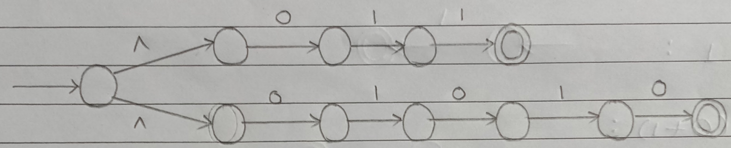
b) (011 + 01010)

011 : 

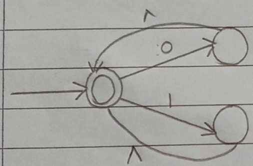
01010 :



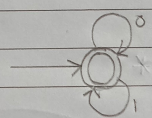
(011 + 01010) :



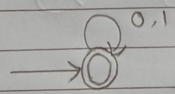
final result,



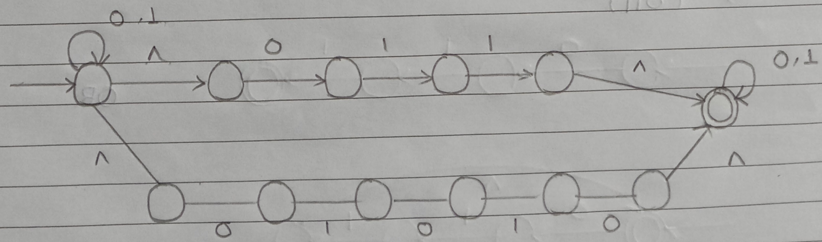
OR



OR



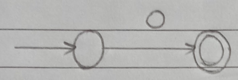
(0+1)*



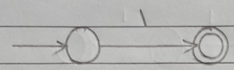
③ (0+1) (01)* (011)*

a) (0+1)

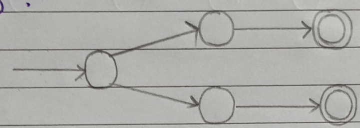
0 :



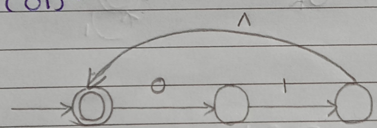
1 :



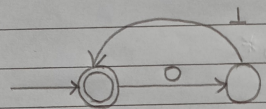
(0+1) :



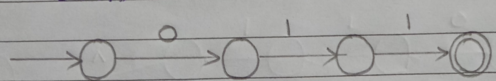
b) (01)*



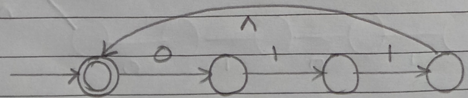
OR



c) (011)*

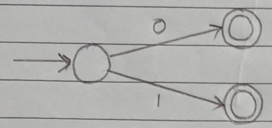


OR



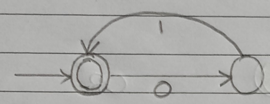
final result :

①



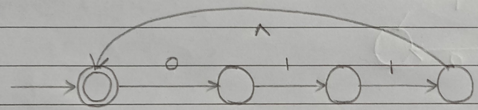
(0+1)

②

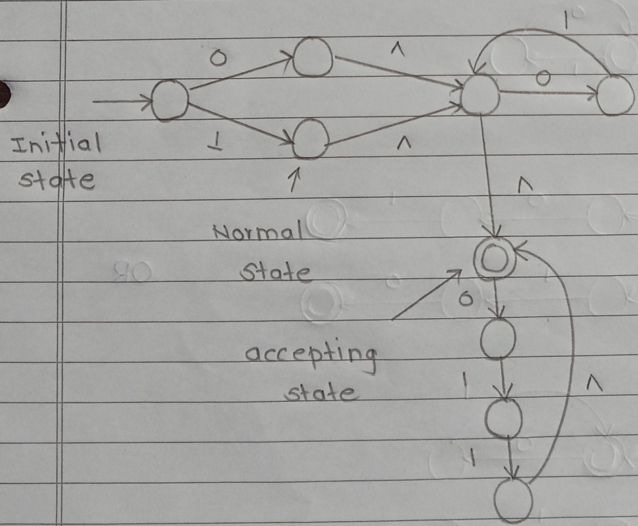


(01)*

③

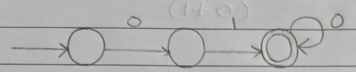


(011)*



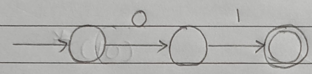
② $010^* + 0(01+10)^*11$

a) 010 :

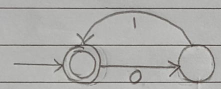


b) $(01+10)^*$

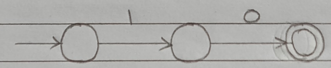
01 :



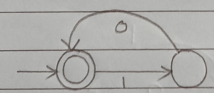
OR



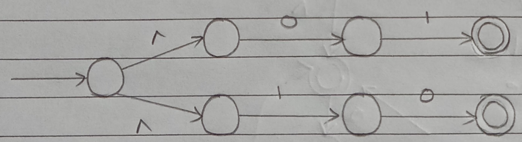
10 :



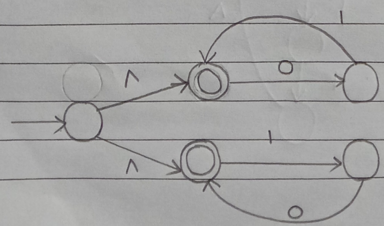
OR



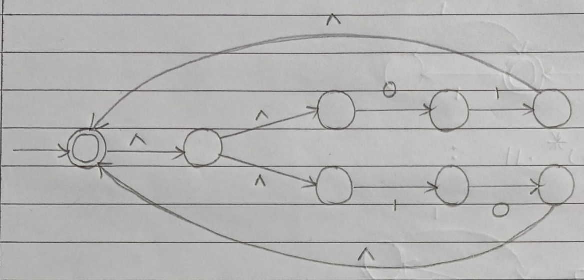
(01 + 10) :



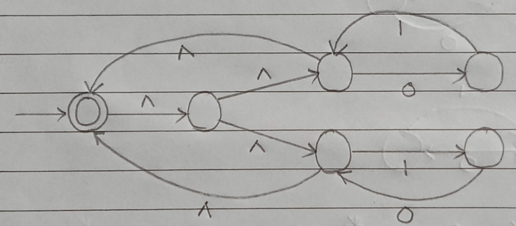
OR



$(01+10)^*$:

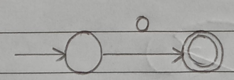


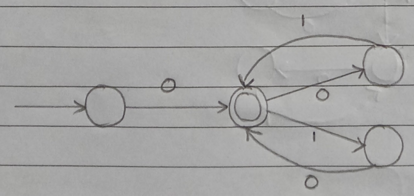
OR



c) $0(01+10)^*$

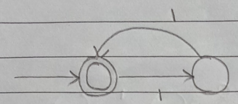
0 :



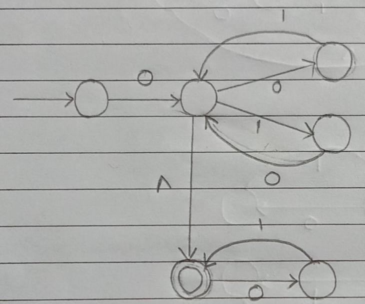


d) $0(01+10)^* \cdot 11$

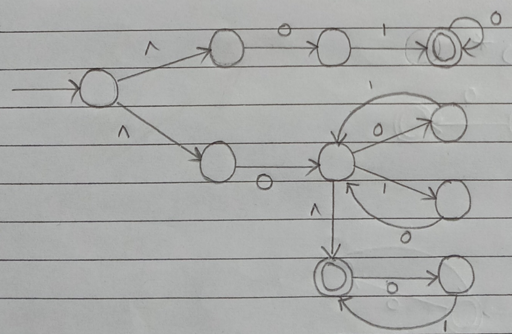
11 :



$0(01+10)^* \cdot 11$:



e) $(010^*) + (0(01+10)^* \cdot 11)$



OR