AUTOMATA AND COMPILER DESIGN

Formal Language and Regular

→ Languages definition languages regular expressions.

→ Finite automata NFA, DFA

→ Conversion of regular expression to NFA

→ NFA to DFA

→ context to free grammar and passing derivation parsetrees.

→ Application of finite automata.

Compiler Design:

→ It translates the Lexical Analysis, Syntax analysis and symantic analysis. These 3 are front end.

→ code optimization and code generation. These 2 are backend.

→ Compiler Design translates the error detection and recovery [principles of Compiler design]

Automata:

→ Automata is a theoratical branch of computer science and mathematical

→ It is the study of abstract machines and computational problems. which can be

solved using the machines.

→ Abstract machine is called automata. ~~with~~ a

→ Automata in a finite no. of states is called Finite Automata.

Languages:

1. Set:

→ Set means collection of objects. These objects are elements.

→ All elements are enclosed in curly brackets {}

2. Empty set:

→ The set which does not have any element.

→ It is represented by ∅.

3. Null set:

→ Represented by ε or ^

4. Power set:

→ It is a set of all the subsets of its elements.

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

It is represented by 2^n

n = no. of elements

Here, $2^n \Rightarrow 2^3 = 8$.

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

Operations on set:

→ UNION

→ INTERSECTION

→ DIFFERENCE

→ COMPLIMENT

Union:

→ Combination of 2 sets.

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

$A \cup B = \{1,2,3,4\}$.

Intersection:

→ Common elements of both the sets

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

$A \cap B = \{1,2\}$

Difference

→ Represented by (-)

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

$A - B = \{1\}$

Compliment:

→ $\bar{A}$ is the compliment operation

→ Here, U is the Universal set

$U = \{10,20,30,40,50\}$ $A = \{10,20\}$

$\bar{A} = U - A$

$\bar{A} = \{30,40,50\}$.

Introduction to defining Language:

→ Alphabet:

* Alphabet is a collection of symbols which cannot be defined formally.

$$S = \{a, b, c, \ldots, z\}$$

→ String:

* String is a collection of symbols from alphabet.

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

$$\{\epsilon, aa, bb, ab, ba, aba, \ldots\}$$

* Infinite no. of strings can be created within the set.

→ Language:

* Language is a collection of appropriate string.
* Language is defined by using input set.
* It is denoted by Σ

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

Operations on strings:

→ concatenation

→ Transpose

→ Palindrome.

Concatenation:

→ Two strings combine and form a single string.

$A = \{white\}$

$B = \{house\}$

$AB = \{whitehouse\}$

Palindrome:

→ The string can be read from left to right as well as right to left

Eg: MAM, MADAM

Transpose:

→ Transpose means reverse operation

$XA = (XA)^T = AX$

$XA = \{123\}$

$(XA)^T = AX = \{321\}$

CLOSURE:

→ There are 2 types of closures:

* star (or) kleen
* Positive closure

Star (or) kleen closure:

→ This is a set of strings of any length including null string (ε)

$$\Sigma^* = \{ \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots\ldots \cup \Sigma^n \}$$

$$\Sigma = \{0\}$$

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

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

$$\Sigma^* = \{\varepsilon, 1, 2, 12, 21, 111, \ldots\ldots\}$$

Positive closure:

→ Positive closure means without including null string.

$$\Sigma^+ = \{ \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots\ldots \Sigma^n \}$$

$$\Sigma = \{0\}$$

$$\Sigma^+ = \{0, 00, 000, \ldots\ldots\}$$

Finite Automata Model:

→ It represents the mathematical model of a system with certain ~~output~~ input, finally it gives the certain output.

$$Q, \Sigma, \delta, q_0, F$$

$Q \Rightarrow$ Finite set of states.

$\Sigma \Rightarrow$ It represents the input set of input alphabet

$\delta \Rightarrow$ Transition function ~~are~~ or mapping function.

$q_0 \Rightarrow q_0$ is the initial state or start state

$$q_0 \in Q$$

F ⇒ Final state which is a non empty set.

Acceptance of strings and Languages:

→ strings and Languages are accepted by finite automata.

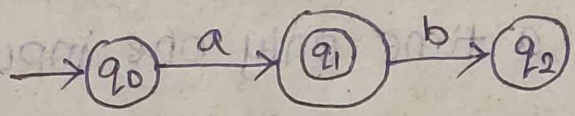
O ⇒ state

→ ⇒ connecting from one state to another state

→O ⇒ start or initial state

(O) ⇒ final state

Eg:



$Q \rightarrow \{q_0, q_1, q_2\}$

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

$\delta \rightarrow$

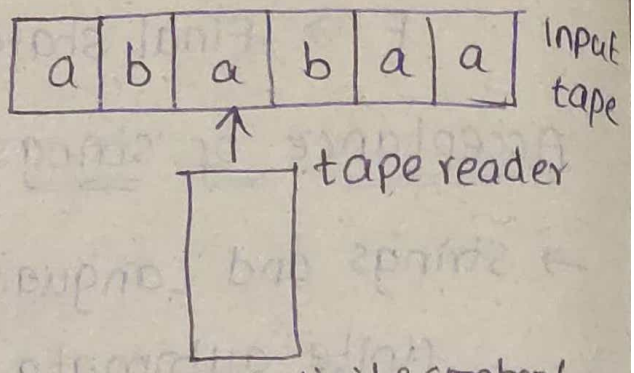
$q_0 \rightarrow \{q_0\}$

$F \rightarrow \{q_1\}$

Transition table:

	a	b
q_0	q_1	$\emptyset$
((q_1))	$\emptyset$	q_2
q_2	$\emptyset$	$\emptyset$

1. Input Tape
2. Tape Reader
3. Finite Control



1. Input Tape:

→ Input tape is divided into cells

→ Each cell is containing input symbol from input alphabet

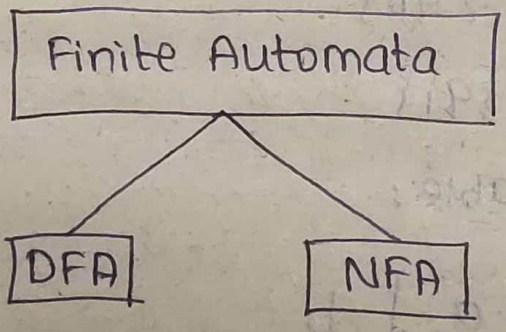
2. Tape Reader:

→ It reads the cells one by one from left to right; at a time only one input symbol is read.

3. Finite control:

→ It decides the next state, particular input from input tape.

Types of automata:

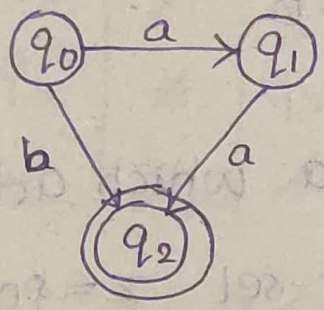


Deterministic finite automate

Non deterministic finite automata.

Deterministic Finite Automata (DFA):

→ Finite Automata (FA) is called DFA if there is only one path for specific input from current state to next state.



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

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

$q_0 = \{q_0\}$

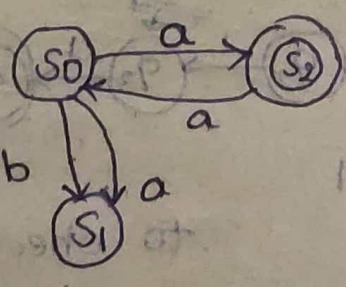
$F = \{q_2\}$

Transition table:

	a	b
q_0	q_1	q_2
q_1	q_2	$\emptyset$
(q_2)	$\emptyset$	$\emptyset$

Non Deterministic Finite Automata (NFA):

→ Finite Automata (FA) is called NFA if there exists many paths for specific input from current state to next state



$Q = \{S_0, S_1, S_2\}$

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

$q_0 = \{S_0\}$

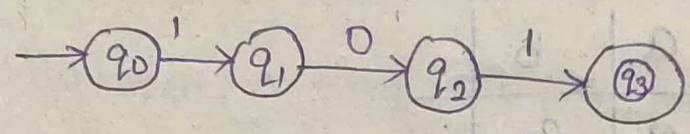
$F = \{S_2\}$

Transition Table:

S_0	a	b
S_0	$\{S_2, S_1\}$	S_1
S_1	$\emptyset$	$\emptyset$
(S_2)	S_0	$\emptyset$

Q: Design finite automata which acts as only input 101 over the set $\Sigma = \{0,1\}$

DFA is



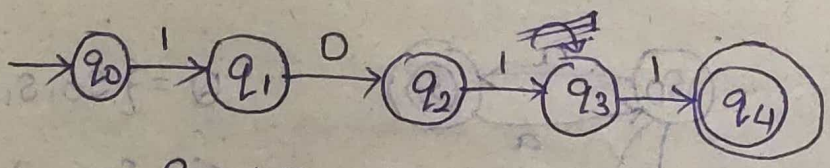
$q_0\,101$

$q_1\,01$

$q_2\,1$

(q_3)

Q: Design finite automata which acts as only input 1011 over the set $\Sigma = \{0,1\}$



$q_0\,1011$

$q_1\,011$

$q_2\,11$

$q_3\,1$

q_4

to check whether string is accepted or not

Transition table

	0	1
q_0	∅	q_1
q_1	q_2	∅
q_2	∅	q_3
q_3	∅	q_4
q_4	∅	∅

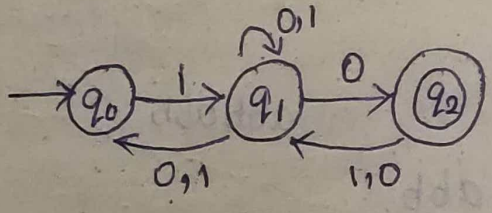
Q: Design finite automata. The string start with 1 end with 0

string should start with 1 & end with 0

1 [either 1(0/1)0] 0

Possibilities

10
1100
1010
1110
1000
110
100



self loop can take either 0/1 'n' no. of times

Tuples: -

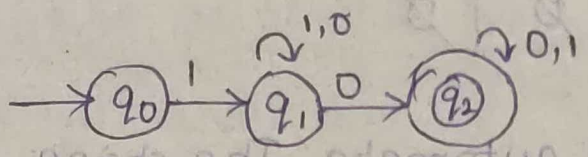
2 verifications.

Transition table

→ Design finite automata with odd no. of 1's and any no. of 0's.

Possibilities:

1 0
1 0 0 0
1 1 1 0
1 1 1 1 1 0



1⇒ Design DFA to accept the string a's & b's with a,b,b.

2⇒ Design finite automata the string ends with double zero

3⇒ Design finite automata where string start with 0 and end with 1.

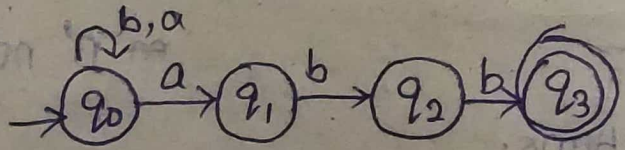
4⇒ Design finite automata where string ends with 01

(1)

Possibilities

abbabb

a(or)b abb



q_0 abbabb

q_0 bbabb

q_0 babb

q_0 abb

q_1 bb

q_2 b

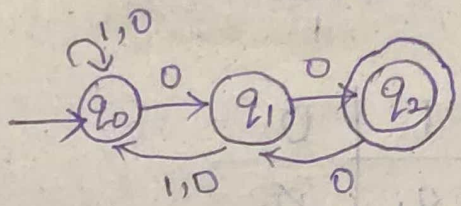
q_3

(2) string ends with double zero

possibilities: 10100

1100

100



q_0 10100

q_0 0100

q_0 100

q_0 00

q_1 0

q_2

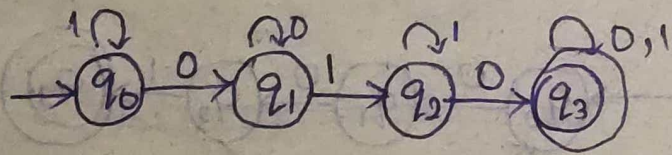
→ Design DFA to accept odd or even numbers represented using binary notations.

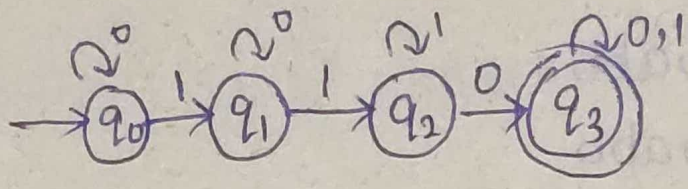
possibilities

Here, we can take any no. of 0's & any no. of 1's

010

0011 etc.



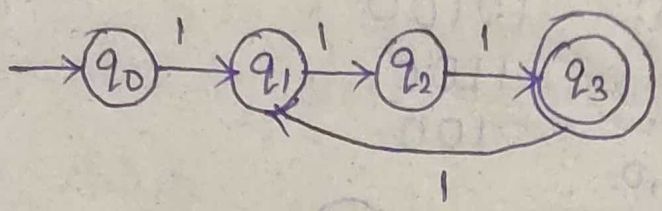


→ Design Finite automata to accept unary number divisible by 3

The number 3 can be written in 111, 111111, ...

Possibilities

111
111111

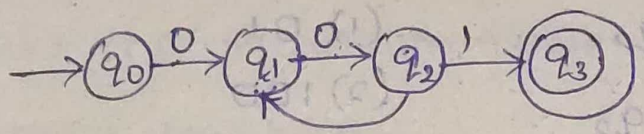


	1	0
q_0	q_1	∅
q_1	q_2	∅
q_2	q_3	∅
q_3	q_1	∅

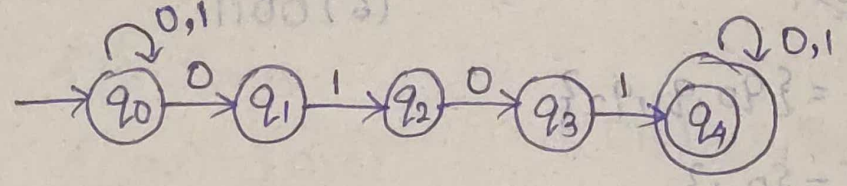
→ Design DFA to accept even no. of 0's followed by single 1

Possibilities:

001
00001
0000001



→ Construct NFA for the language consisting a substring 0101



either 0(or)1 | 0101 | either 0(or)1

→ Design DFA for accepting all the strings

$$L = \{0^m 1^n / m \geq 0 \text{ \& } n \geq 1\}$$

The language in which all the zeroes followed by all the 1's.

$$L = \{0^m 1^n / m \geq 0 \text{ and } n \geq 1\}$$

if $m=0, n=1$

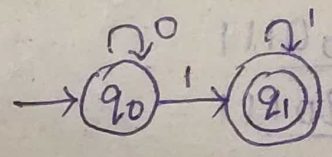
$L = 0^m 1^n$

$L = 0^0 1^1$

$L = 0^m 1^n$

$m=2, n=3$

$0^2 1^3 \Rightarrow 00111$



$\Rightarrow$ $M = (\{q_0, q_1, q_2\} \{0,1\} \delta\ q_0 \{q_1\})$

$\delta(q_0, 0) = q_0$

$\delta(q_0, 1) = q_1$

$\delta(q_1, 0) = q_0$

$\delta(q_1,1) = q_2$

$\delta(q_2,0) = q_2$

$\delta(q_2,1) = q_1$

(1) 01

(2) 100

(3) 101

(4) 0111

(5) 11001

(6) 00111

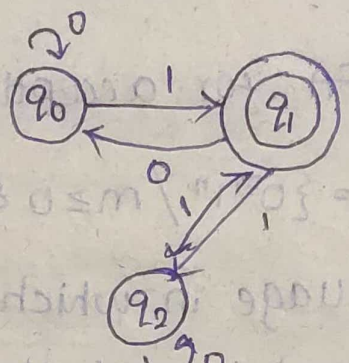
Tuples:

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

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

$q_0 = q_0$

$F = \{q_1\}$



$q_0 01$

$q_0 1$

$q_1 \Rightarrow$ accepted

$q_0 100$

$q_1 \Rightarrow$ not accepted.

$q_0 101$

$q_1 01$

$q_0 1$

$q_1 \Rightarrow$ accepted

$q_0 0111$

$q_0 111$

$q_1 11$

$q_2 1$

$q_1 \Rightarrow$ accepted.

$q_0 11001$

$q_1 1001$

$q_2 001$

$q_2 01$

$q_2 1$

$q_1 \Rightarrow$ accepted

$q_0 0011$

$q_0 011$

$q_0 11$

$q_1 1$

q_2

not accepted

→ Draw the transition table & transition diagram for NFA

$M = (\{A, B, C, D\}, \{0, 1\}, \delta, \{C\}, \{A, C\})$

$\delta(A, 0) = \delta(A, 1) = \{A, B, C\}$

$\delta(B, 0) = B$

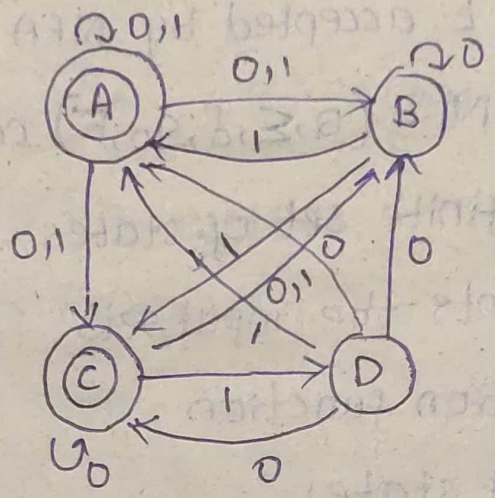
$\delta(B, 1) = \{A, C\}$

$\delta(C, 0) = \{B, C\}$

$\delta(C, 1) = \{B, D\}$

$\delta(D, 0) = \{A, B, C, D\}$

$\delta(D, 1) = \{A\}$



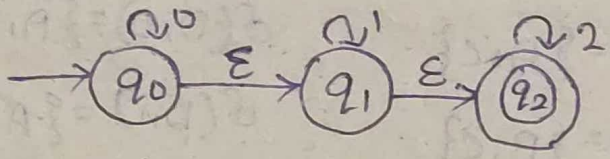
Transition table:

	0	1
A	{A, B, C}	{A, B, C}
B	{B}	{A, C}
C	{B, C}	{B, D}
D	{A, B, C, D}	{A}

NFA with ε transitions :

→ The ε transition means if NFA, they are given in order to ~~or~~ more from one state to another state without having any input symbol.

For example,



Acceptance of Language:

→ The Language L accepted by NFA with ε denoted by $M = (Q, \Sigma, \delta, q_0, F)$ can be defined as Q is the finite set of states

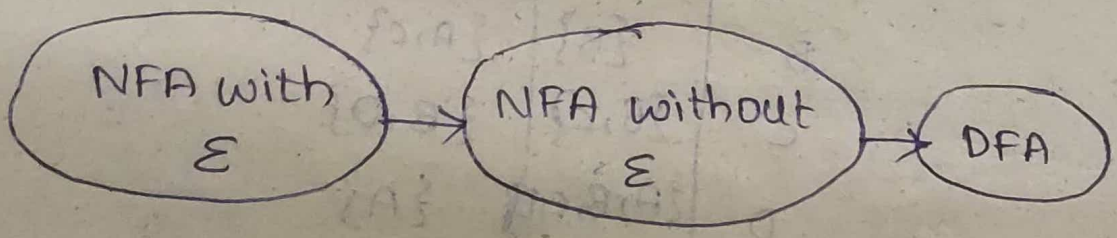
Σ represents the input sets

δ ⇒ Transition function

q_0 ⇒ initial state

F ⇒ Final state

conversions and equivalence:



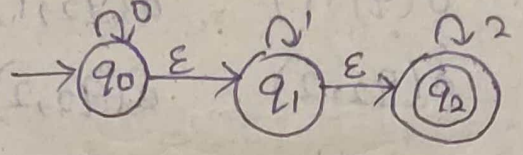
1 ⇒ In this method, we try to remove all the ε's transitions from given NFA

Find all ε transitions from each state from Q that will be called as ε closure set of $\{q_i\}$ $q_i \in Q$

2⇒ Then transition can be obtained, the transition means an ε closure or transition moves

→ step 2 is repeated for each input symbol & each state of given NFA.

Q: Convert the given NFA with ε to without ε



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

$\Sigma = \{0, 1, 2\}$

$q_0 = q_0$

$F = \{q_2\}$

Transition Table:

	0	1	2	ε
q_0	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_1	ϕ	q_2
(q_2)	ϕ	ϕ	q_2	ϕ

self state + ε reachable states

ε closure $(q_0) = \{q_0, q_1, q_2\}$

ε closure $(q_1) = \{q_1, q_2\}$

ε closure $(q_2) = \{q_2\}$

→ Now, find out the transition of each & every input symbol

$\delta(q_0, 0)$	$\delta(q_1, 0)$	$\delta(q_2, 0)$
$\delta(q_0, 1)$	$\delta(q_1, 1)$	$\delta(q_2, 1)$
$\delta(q_0, 2)$	$\delta(q_2, 2)$	$\delta(q_2, 2)$

NOTE: To find the transition

$$\delta(q_0, a) = \varepsilon\text{closure}(\delta(\hat{\delta}(q_0, \varepsilon), a))$$

$q_0 \Rightarrow$ any state

$a \Rightarrow$ any input symbol

$\hat{\delta} \Rightarrow$ delta cap

$\hat{\delta}(q_0, \varepsilon) \Rightarrow \varepsilon$ closure(q_0)

$$\delta(q_0, 0) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_0, \varepsilon), 0))$$

$$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_0), 0))$$

$$= \varepsilon\text{-closure}(\delta(q_0, q_1, q_2), 0))$$

$$= \varepsilon\text{-closure}(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$$

$$= \varepsilon\text{-closure}(q_0 \cup \emptyset \cup \emptyset)$$

$$= \varepsilon\text{-closure}(q_0)$$

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

$\delta(q_0,1) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_0,\varepsilon),1))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_0),1))$

$= \varepsilon\text{-closure}(\delta(q_0,1) \cup \delta(q_1,1) \cup \delta(q_2,1))$

$= \varepsilon\text{-closure}(\emptyset \cup q_1 \cup \emptyset)$

$= \varepsilon\text{-closure}(q_1)$

$= \{q_1, q_2\}$

$\delta(q_0,2) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_0,\varepsilon),2))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_0),2))$

$= \varepsilon\text{-closure}(\delta(q_0,2) \cup \delta(q_1,2) \cup \delta(q_2,2))$

$= \varepsilon\text{-closure}(\emptyset \cup \emptyset \cup q_2)$

$= \varepsilon\text{-closure}(q_2)$

$= \{q_2\}$

$\delta(q_1,0) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_1,\varepsilon),0))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_1),0))$

~~$\times\{= \varepsilon\text{-closure}(\delta(q_0,0) \cup \delta(q_1,0) \cup \delta(q_2,0))\}\times$~~

$= \varepsilon\text{-closure}(\delta(q_1,q_2),0))$

$= \varepsilon\text{-closure}(\delta(q_1,0) \cup \delta(q_2,0))$

$= \varepsilon\text{-closure}(\emptyset \cup \emptyset)$

$= \varepsilon\text{-closure}(\emptyset)$

$= \emptyset$

$\delta(q_1,1) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_1,\varepsilon),1))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_1),1))$

$= \varepsilon\text{-closure}(\delta(q_1,q_2),\emptyset))$

$= \varepsilon\text{-closure}(\delta(q_1,1) \cup \delta(q_2,1))$

$= \varepsilon\text{-closure}(q_1 \cup \emptyset)$

$= \varepsilon\text{-closure}(q_1)$

$= \{q_1, q_2\}$.

$\delta(q_1, 2) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_1, \varepsilon), 2))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_1), 2))$

$= \varepsilon\text{-closure}(\delta(q_1, q_2), 2))$

$= \varepsilon\text{-closure}(\delta(q_1, 2) \cup \delta(q_2, 2))$

$= \varepsilon\text{-closure}(\emptyset \cup q_2)$

$= \varepsilon\text{-closure}(q_2)$

$= \{q_2\}$.

$\delta(q_2, 0) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_2, \varepsilon), 0))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_2), 0))$

$= \varepsilon\text{-closure}(\delta(q_2), 0))$

$= \varepsilon\text{-closure}(\delta(q_2, 0))$

$= \varepsilon\text{-closure}(\emptyset)$

$= \emptyset$

$\delta(q_2, 1) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_2, \varepsilon), 1))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_2), 1))$

$= \varepsilon\text{-closure}(\delta(q_2), 1))$

$= \varepsilon\text{-closure}(\delta(q_2, 1))$

$= \varepsilon\text{-closure}(\emptyset)$

$= \emptyset$

$\delta(q_2,2) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_2,\varepsilon),2))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_2),2))$

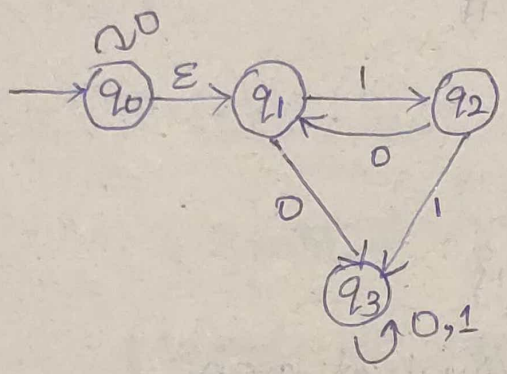
$= \varepsilon\text{-closure}(\delta(q_2),2))$

$= \varepsilon\text{-closure}(\delta(q_2,2))$

$= \varepsilon\text{-closure}(q_2)$

$= \{q_2\}.$

(2)



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

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

$q_0 = \{q_0\}$

$F = \{q_1\}$

$\varepsilon\text{-closure}(q_0) = \{q_0, q_1\}$

$\varepsilon\text{-closure}(q_1) = \{q_1\}$

$\varepsilon\text{-closure}(q_2) = \{q_2\}$

$\varepsilon\text{-closure}(q_3) = \{q_3\}$

$\delta(q_0,0) = \varepsilon\text{-closure}(\delta(\hat{\delta}(q_0,\varepsilon)\,a))$

$\varepsilon\text{-closure}(\delta(\hat{\delta}(q_0,\varepsilon),0)$

$\varepsilon\text{-closure}(\delta(q_0,q_1),0)$

$\varepsilon\text{-closure}(\delta(q_0,0) \cup \delta(q_1,0))$

$\varepsilon\text{-closure}(q_0) \cup \varepsilon\text{-closure}(q_3)$

$\{q_0,q_1\} \cup \{q_0\}$

$\{q_0,q_1,q_3\}$

NFA to DFA Conversion:

→ NFA to DFA; given an NFA $M=(Q,\Sigma,\delta,q_0,F)$, Now, we have to convert it into DFA, then $M'=(Q',\Sigma',\delta',q_0',F')$

→ Let $M=\{\{q_0,q_1\},\{0,1\},\delta,q_0,\{q_1\})$

$\delta(q_0,0)=\{q_0,q_1\}$

$\delta(q_0,1)=\{q_1\}$

$\delta(q_1,0)=\emptyset$

$\delta(q_1,1)=\{q_0,q_1\}$

construct its equivalent DFA

Sol Given, $M=(Q,\Sigma,\delta,q_0,F)$

Let the DFA be $M'=(Q',\Sigma',\delta',q_0',F')$

Now, the δ' function is obtained by

$\delta'(q_0,0)=[q_0,q_1]$

$\delta'(q_0,1)=[q_1]$

$\delta(q_1,0)=[\emptyset]$

$\delta(q_1,1)=[q_0,q_1]$

Transition table (NFA):

	0	1
q_0	$\{q_0,q_1\}$	$\{q_1\}$
q_1	$\emptyset$	$\{q_0,q_1\}$

→ $\{q_0, q_1\}$ is the new state, now we have to find out the $(\delta(q_0,q_1),0)$ and $(\delta(q_0,q_1),1)$

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

$\delta(q_0,0) \cup \{\delta(q_1,0)$

$\{q_0,q_1\} \cup \{\emptyset\}$

$[q_0,q_1]$

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

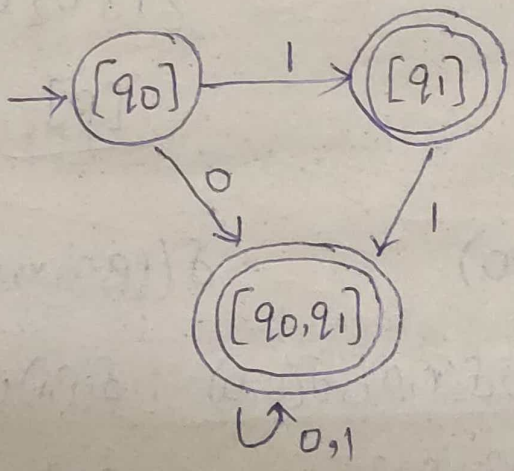
$\delta(q_0,1) \cup \delta(q_1,1)$

$\{q_1\} \cup \{q_0,q_1\}$

$[q_0,q_1]$

Transition table (DFA):

	0	1
$[q_0]$	$[q_0,q_1]$	$[q_1]$
$[q_1]$	$[\emptyset]$	$[q_0,q_1]$
$[q_0,q_1]$	$[q_0,q_1]$	$[q_0,q_1]$



Q. Construct NFA to DFA

	0	1
p	{p, q}	{p}
q	{r}	{r}
r	{s}	∅
(s)	{s}	{s}

→ NFA

$\Rightarrow$ ~~$\delta((P,q),0)$~~ $\delta((P,q),1)$

$\delta(P,0) \cup \delta(q,0)$ $\delta(P,1) \cup \delta(q,1)$

$\{P,q\} \cup \{r\}$ $\{P\} \cup \{r\}$

~~$\{P,q,r\}$~~ ~~$\{P,r\}$~~

new state $\leftarrow$ $[P,q,r]$ $[P,r] \rightarrow$ new state

$\Rightarrow$ $\delta((P,q,r),0)$ $\delta((P,q,r),1)$

$\delta(P,0) \cup \delta(q,0) \cup \delta(r,0)$ $\delta(P,1) \cup \delta(q,1) \cup \delta(r,1)$

$\{P,q\} \cup \{r\} \cup \{s\}$ $\{P\} \cup \{r\} \cup \{\emptyset\}$

$[P,q,r,s]$ $[P,r]$

$\Rightarrow$ $\delta((P,r),0)$ $\delta((P,r),1)$

$\delta(P,0) \cup \delta(r,0)$ $\delta(P,1) \cup \delta(r,1)$

$\{P,q\} \cup \{s\}$ $\{P\} \cup \{\emptyset\}$

$[P,q,s]$ $[P]$

$\Rightarrow$ $\delta((P,q,r,s),0)$ $\delta((P,q,r,s),1)$

$\delta(P,0) \cup \delta(q,0) \cup \delta(r,0), \delta(s,0)$ $\delta(P,1) \cup \delta(q,1) \cup \delta(r,1) \cup \delta(s$

$\{P,q\} \cup \{r\} \cup \{s\} \cup \{s\}$ $\{P\} \cup \{r\} \cup \{\emptyset\} \cup \{s\}$

$[P,q,r,s]$ $[P,r,s]$

$\Rightarrow$ $\delta((P,q,s),0)$ $\delta(P,q,s),1)$

$\delta(P,0) \cup \delta(q,0) \cup \delta(s,0)$ $\delta(P,1) \cup \delta(q,1) \cup \delta(s,1)$

$\{P,q\} \cup \{r\} \cup \{s\}$ $\{P\} \cup \{r\} \cup \{\emptyset\}$

$[P,q,r,s]$ $[P,r]$

$\Rightarrow$ $\delta((P,r,s),0)$ $\quad$ $\delta((P,r,s),1)$

$\delta(P,0) \cup \delta(r,0) \cup \delta(s,0)$ $\quad$ $\delta(P,1) \cup \delta(r,1) \cup \delta(s,1)$

$\{P,q\} \cup \{s\} \cup \{s\}$ $\quad$ $\{P\} \cup \{\emptyset\} \cup \{s\}$

$[P,q,s]$ $\quad$ $[P,s]$

Transition table for

$\delta((P,s),0)$ $\quad$ $\delta((P,s),1)$

$\delta(P,0) \cup \delta(s,0)$ $\quad$ $\delta(P,1) \cup \delta(s,1)$

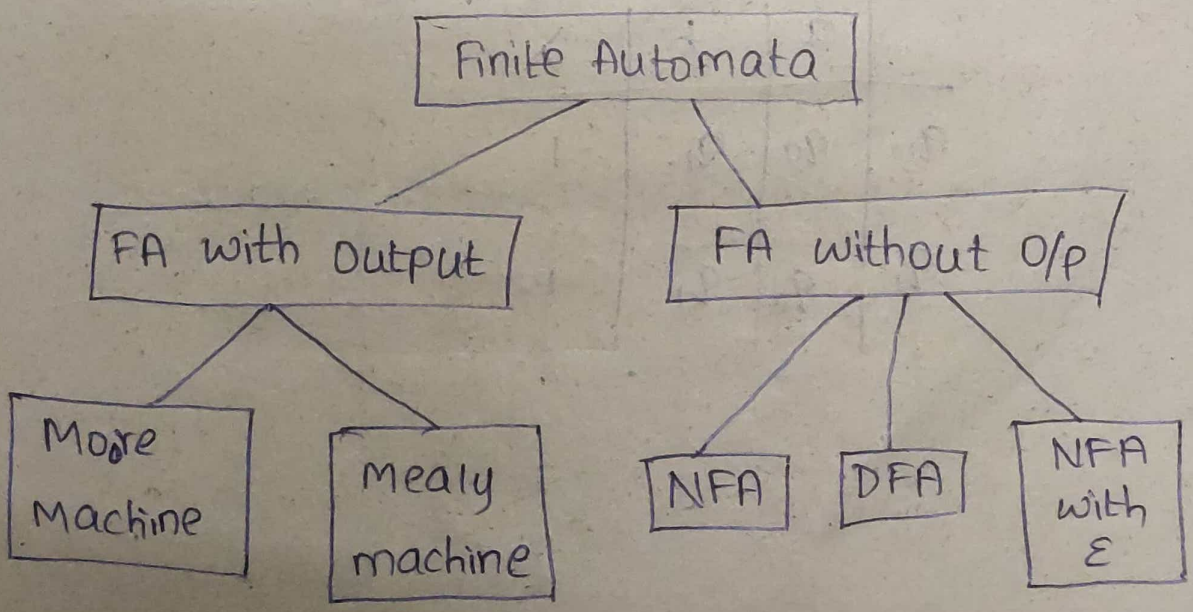
$\{P,q\} \cup \{s\}$ $\quad$ $\{P\} \cup \{\emptyset\}$

$[P,q,s]$ $\quad$ $[P]$.

Transition table for DFA:

	0	1
P	{P,q}	{P}
q	{r}	{r}
r	{s}	$\emptyset$
s	{s}	{s}

Finite Automata:



Moore Machine:

→ It is a final state machine in which the next state is decided by current state & current I/p symbol.

→ The output symbol depends only upon present state of the machine.

→ There are 6 inputs - $Q, \Sigma, \delta, q_0, \Delta, \lambda$

Q - finite set of states which is non empty set.

Σ - I/p alphabet

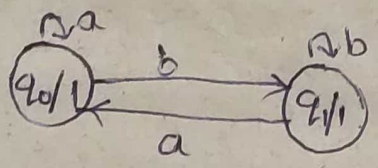
δ - Transition function (or)

q_0 - initial/start state

Δ - output alphabet

λ - output function.

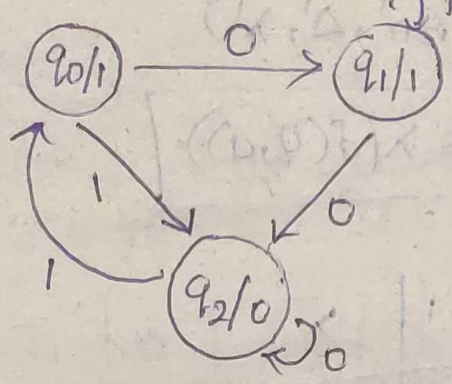
→ λ says that every state o/p is associated



	a	b	λ
q_0	q_0	q_1	1
q_1	q_0	q_1	1

→ Draw the transition diagram.

	0	1	λ
q_0	q_1	q_2	1
q_1	q_2	q_1	1
q_2	q_2	q_0	0

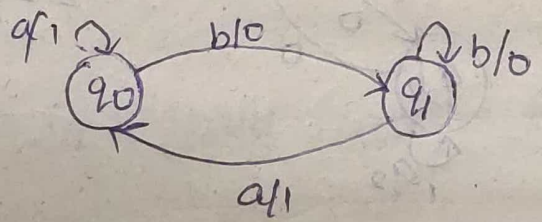


Mealy Machine:

→ The output symbol depends on present input symbol & present state of the machine.

→ There are 6 tuples - $Q, \Sigma, \delta, q_0, \Delta, \lambda$

$$\lambda : Q \times \Sigma \rightarrow \Delta$$



status	Input a	o/p	Input b	o/p
q_0	q_0	1	q_1	0
q_1	q_0	1	q_1	0

Conversions :

Method for conversion of More machine to Mealy machine:

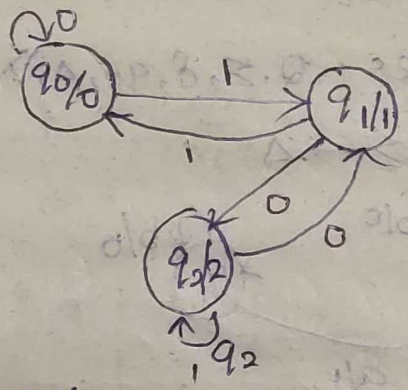
$M=(Q,\Sigma,\delta,q_0,\Delta,\lambda)$ be the more machine equivalent, mealy machine can be represented by

$$M'=(Q',\Sigma',\delta',q_0',\Delta',\lambda')$$

$$\boxed{\lambda'(q,a)=\lambda(\delta(q,a))}$$

Eg:

Q	0	1	λ
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2



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

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

$\lambda'(q_0,0)=\lambda(\delta(q_0,0))=\lambda(q_0)=0$

$\lambda'(q_0,1)=\lambda(\delta(q_0,1))=\lambda(q_1)=1$

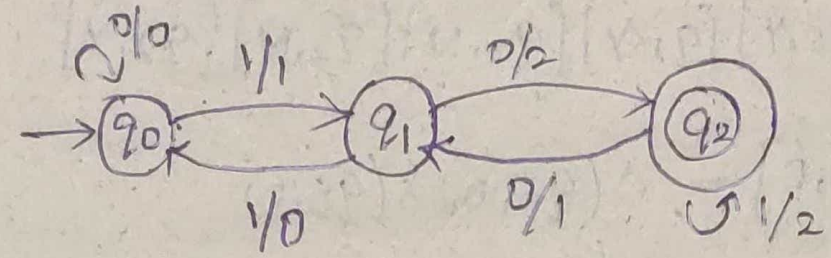
$\lambda'(q_1,0)=\lambda(\delta(q_1,0))=\lambda(q_2)=2$

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

$\lambda'(q_2,0)=\lambda(\delta(q_2,0))=\lambda(q_1)=1$

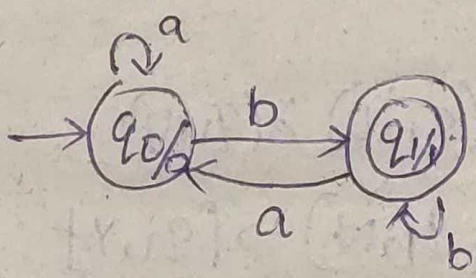
$\lambda'(q_2,1)=\lambda(\delta(q_2,1))=\lambda(q_2)=2.$

	Input 0	O/P	Input 1	O/P
q_0	q_0	0	q_1	1
q_1	q_2	2	q_0	0
q_2	q_1	1	q_2	2



Q

	a	b	x
q_0	q_0	q_1	0
q_1	q_0	q_1	1

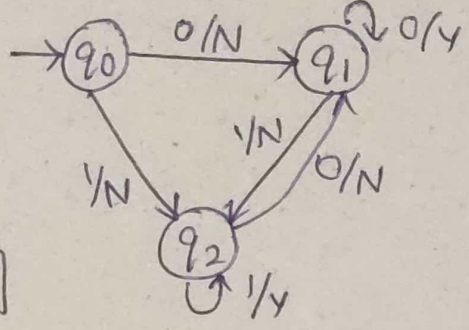


Conversion of Mealy machine to more machine:

$$\delta' = [[q,b]a] = [\delta(q,a)\,\lambda(q,a)]$$

$$\lambda' = [q,b] = b.$$

The states for more machine will be



$[q_0,N]$ $[q_0,Y]$ $[q_1,Y]$, $[q_1,N]$

$[q_2,N]$ $[q_2,Y]$

$$\delta'[[q_0,N],0] = \delta(q_0,0)\,\lambda(q_0,0)$$

$$= [q_1,N]$$

$$\lambda'[q_0,N] = N$$

$$\delta'[[q_0,N],1] = \delta(q_0,1)\,\lambda(q_0,1)$$

$$= [q_2,N]$$

$$\lambda'[q_0,N] = N.$$

$$\delta'[[q_1,N],0] = \delta(q_1,0)\,\lambda(q_1,0)$$

$$= [q_1,Y]$$

$$\delta'[[q_1,N],1] = \delta(q_1,1)\,\lambda(q_1,0)$$

$$= [q_2,N]$$

$$\delta'[[q_1,Y],0] = \delta(q_1,0)\,\lambda(q,0)$$

$$= [q_1,Y]$$

$$\delta'[[q_1,Y],1] = \delta(q_1,1)\,\lambda(q_1,1)$$

$$= [q_2,N]$$

$$\delta[[q_2,N],0] = \delta(q_2,0)\lambda(q_2,0)$$
$$= [q_1,N]$$
$$\delta[[q_2,N],1] = \delta(q_2,1)\lambda(q_2,0)$$
$$= [q_2,Y]$$
$$\delta[[q_2,Y],0] = \delta[q_2,0)\lambda(q_2,0)$$
$$= [q_1,N]$$
$$\delta[(q_2,Y),1] = \delta[q_2,1]\lambda(q_2,1)$$
$$= [q_2,Y]$$

states	0	1	λ
$[q_0,N]$	$[q_1,N]$	$[q_2,N]$	N
$[q_0,Y]$	$[q_1,N]$	$[q_2,N]$	Y
$[q_1,N]$	$[q_1,Y]$	$[q_2,N]$	N
$[q_1,Y]$	$[q_1,Y]$	$[q_2,N]$	Y
$[q_2,Y]$	$[q_1,N]$	$[q_2,Y]$	N
$[q_2,Y]$	$[q_1,Y]$	$[q_2,Y]$	Y

REGULAR SET:

→ Set is represented by a regular expression is called regular set.

REGULAR EXPRESSION:

→ Let Σ be an alphabet which is used to denote the inputs and the regular expressions over Σ can be defined as follows.

* $\emptyset$ is a regular expression which denotes empty set
* ε is a regular expression which denotes null string
* For each a in Σ, a is the regular expression & denotes the set a i.e., {a}
* If ~~and~~ r and s are regular expressions denoting the languange L1 & L2

$r+s = L1 \cup L2$ i.e., Union

$rs = L1L2$ i.e., concatenation

$r^* = L1^*$ i.e., closure

REGULAR LANGUAGE:

* The Language accepted by regular expression is called regular language

Q: Write the regular expressions for the language accepting all combination of a's over the set $\Sigma = \{a\}$

$$R.E = a^*$$

$$R.L = \{\varepsilon, a, aa, aaa, \ldots\}$$

Q: Regular Expression for the language containing no. of a's & b's

$$R.E = (a+b)^*$$

$$R.L = \{\varepsilon, a, b, aa, bb, ab, ba, aba \ldots\}$$

Q: Construct RE for the language accepting all the strings the string end with 00 over the set $\Sigma = \{0,1\}$

Possibilities

00
100
0100

combination of 0's & 1's 00

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

$$\{0, 1, 00, 11, 01, 10, 100 \ldots\} 00$$

$$\Rightarrow \{\varepsilon, 00, 100, \ldots\}$$

Q: String start with 1 end with 0

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

→ Write a regular expression to denote or accept all the strings which begin or end with 00 or 11

Sol The regular expression can be categorized into 2 sub parts then L1 = The string which begin with 00 or 11. L2 = The string which end with 00 or 11.

$L_1 = \{$ begin 00 or 11 $\}$ $(00+11)(0+1)^*$

$L_2 = \{$ End 00 or 11 $\}$ $(0+1)^*(00+11)$

$L_1 + L_2$

$$[(00+11)(0+1)^*] + [(0+1)^*(00+11)]$$

→ Construct the R.E to denote a language L over the $\Sigma = \{0\}$ having even length of string

$R.E = (00)^*$

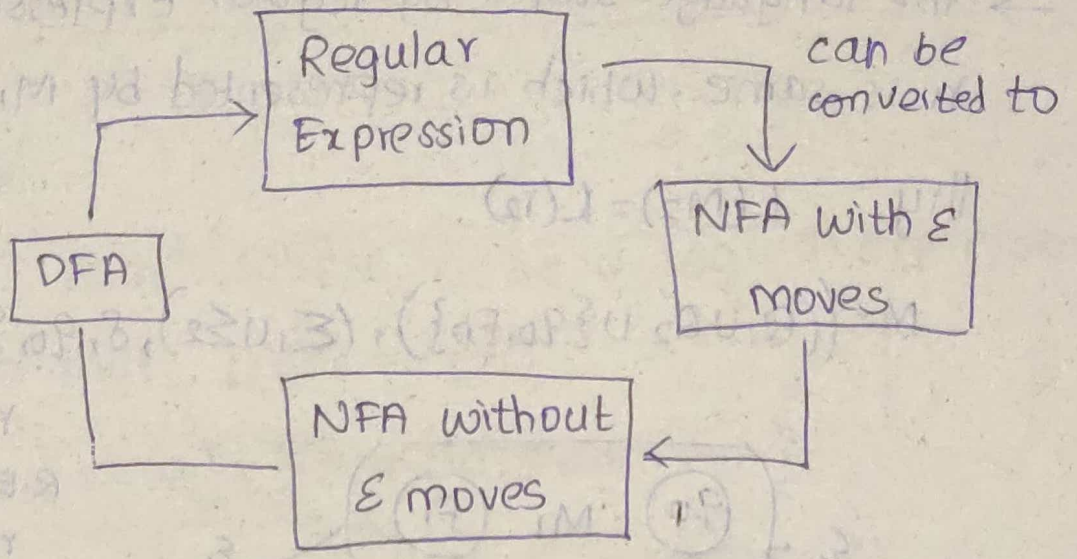
$R.L = \{\varepsilon, 00, 0000, 000000, \ldots\}$

→ odd length of string

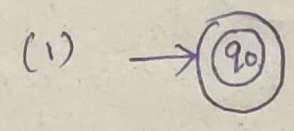
$R.E = 0(00)^*$

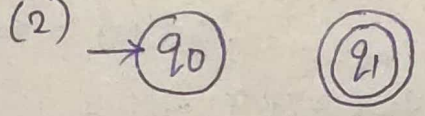
$R.L = \{\varepsilon 0, 000, 00000, \ldots\}$

Constructing FA for given regular expression:

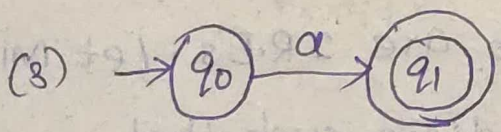


→ Equivalence of NFA & R.E.

(1)  $r = RE = \varepsilon$

(2)  $r = R.E = \emptyset$

No path to final state.

(3)  $r = R.E = a$

Any type of regular expressions there are only 3 cases that are possible. Union
Concatenation
closure.

1. Union:

→ Let $r = r_1 + r_2$ where r_1 & r_2 are Regular expressions there exists 2 NFA's.

$$M_1 = \{Q_1, \Sigma_1, \delta_1, \{F_1\}, \{q_0\}\}$$

$$M_2 = \{Q_2, \Sigma_2, \delta_2, \{F_2\}, \{q_0\}\}$$

$L(M_1) = L(r_1)$

→ The language states by regular expression r_1 is same, which is represented by M_1

Ilry $L(M_2) = L(r_2)$

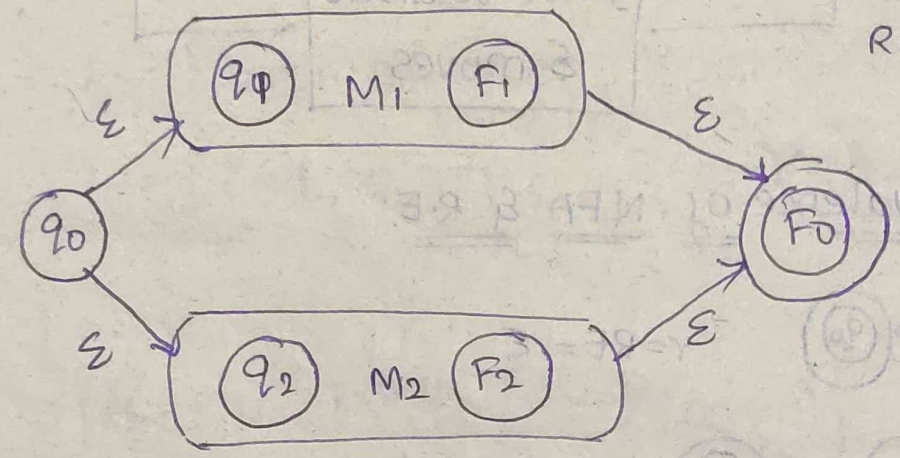
$M = ((Q_1 \cup Q_2 \cup \{q_0, F_0\}), (\Sigma_1 \cup \Sigma_2), \delta, q_0, \{F_0\})$

$r = 0+1$

$R.E = r_1 + r_2$

$r_1 = 0$

$r_2 = 1$



Concatenation

→ Let $r = r_1 r_2$ where r_1 & r_2 are 2 R.E's. Let M_1 & M_2 denotes the 2 machines such that

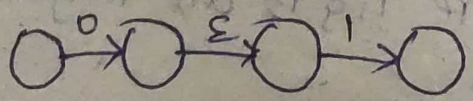
$L(M_1) = L(r_1)$

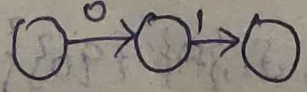
$L(M_2) = L(r_2)$

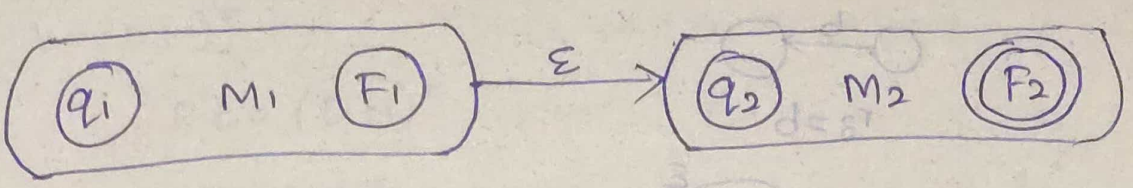
$r = 01$

$r = r_1 r_2$

$r_1 = 0, r_2 = 1$



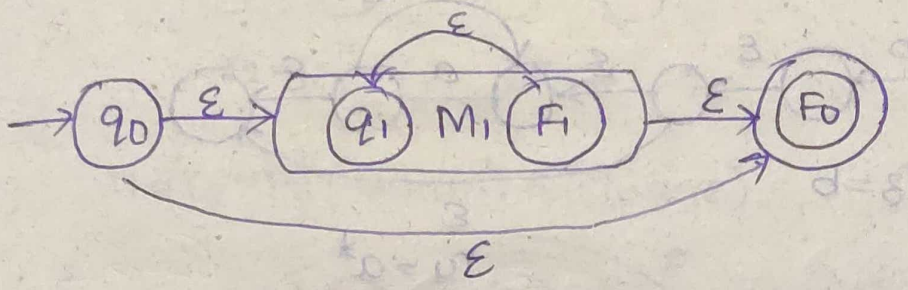




closure

→ Let $r = r_1^*$ where r_1 be a regular expression such that $L(M_1) = L(r_1)$

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



⇒ $b+ba^*$

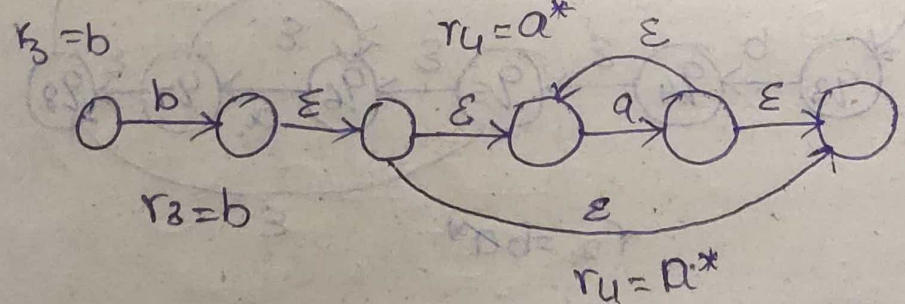
$R.E = b+ba^*$

$r = r_1 + r_2$

$r_1 = b \quad r_2 = ba^*$ (concatenation)

$r_2 = r_3 r_4$

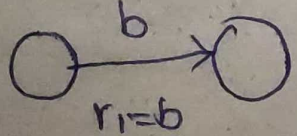
$r_3 = b \quad r_4 = a^*$

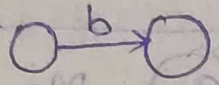


concatenation.

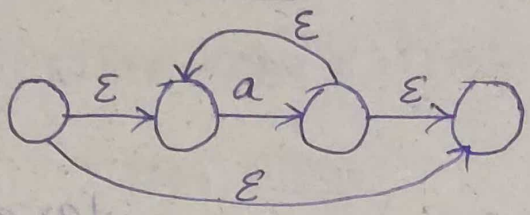
$r_2 = ba^*$

Individual diagrams:



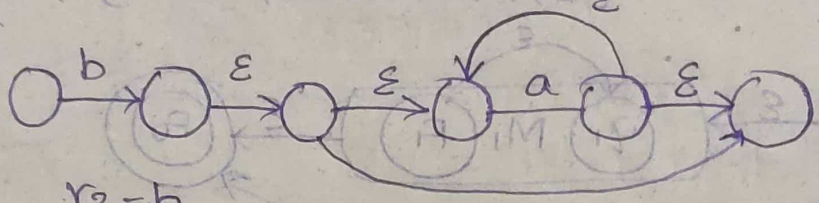


$r_3 = b$



$r_4 = a^*$

→ $r_2 = ba^*$ (concatenation)

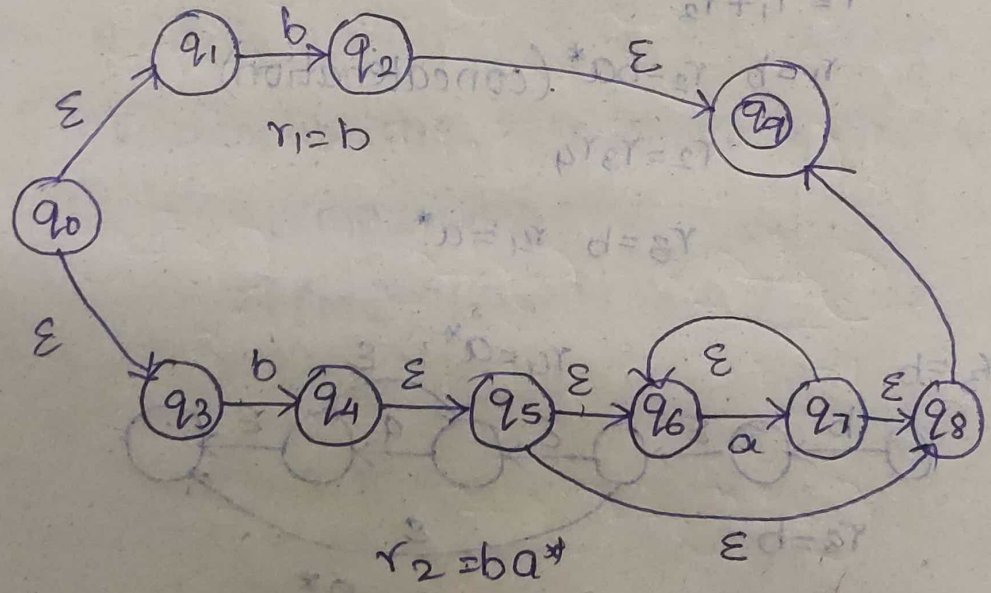


$r_3 = b$

$r_4 = a^*$

→ $r = r_1 + r_2$ $\quad r_2 = ba^*$

$r = b + ba^*$ (union)



$r_1 = b$

$r_2 = ba^*$

$r_1 = b + ba^*$

$\Rightarrow r = (0+1)^*$

$R.E = (0+1)^*$

~~$r = r_1 + r_2$~~

$r = (r_1)^*$

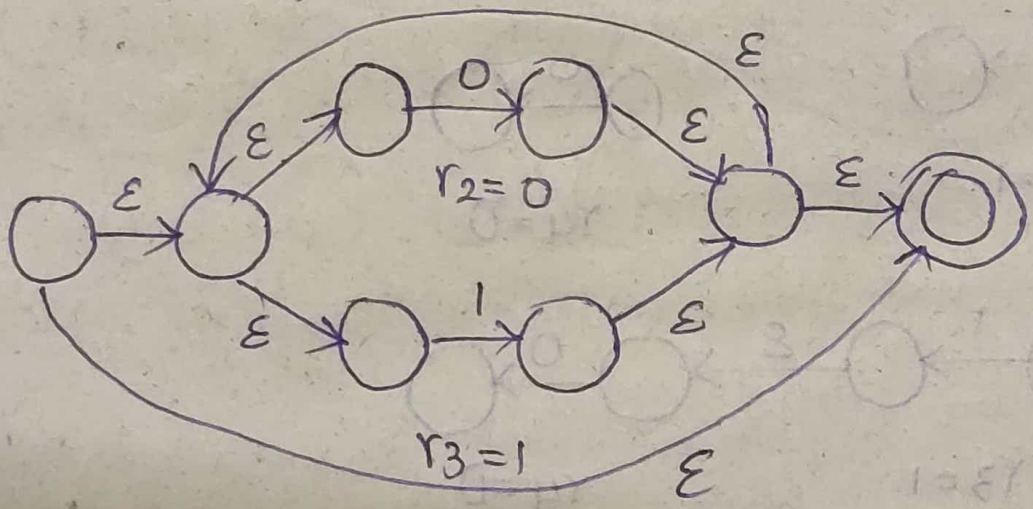
$r_1 = 0 + 1$

$r_1 = r_2 + r_3$

$r_2 = 0 \quad r_3 = 1$

$r_2 = 0$ $r_3 = 1$

$r_1 = (r_2 + r_3)$ (union) & (closure)



Derivation Tree:

→ Derivation Tree is a graphical representation of the given production rule for a given CFG (context Free Grammar) the derivation can be done to obtain some string given set of production rule, the derivation tree is called passed tree.

→ The root node is always a indicating the start symbol.

→ The derivation is always read from left to right

→ The leaf nodes are always terminal node.

→ The Interial nodes are always the non-terminal node.

Q: ~~Given~~ Draw derivation tree for the given production rule

$S \rightarrow bSb$

$S \rightarrow a/b$

CFG = (V, T, P, S)

V = non-terminal (upper case letters)

T = terminal (lower case letters including operators)

P = Production rules

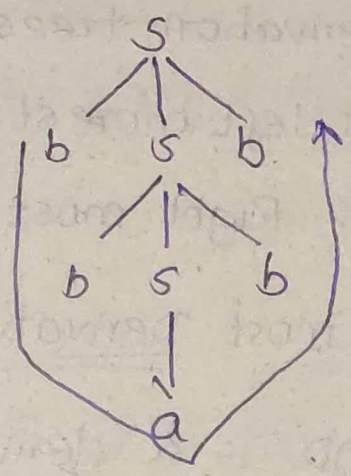
S = start symbol.

$V = \{S\}$

$T = \{a, b\}$

$P = \{ S \to bSb$
$S \to a$
$S \to b \}$

$S = \{S\}$



bbabb

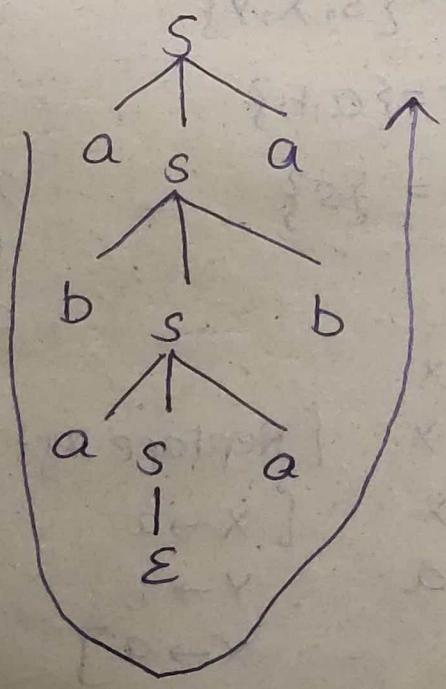
→ Draw the derivation tree for the abaaba for CFG $P = \{ S \to asa$
$S \to bsb$
$S \to a/b/\varepsilon \}$.

$V = \{S\}$

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

$P = \{ S \to asa$
$S \to bsb$
$S \to a/b/\varepsilon \}$

$S = \{S\}$



abaaba

→ Derivation trees are 2 types

1. left most derivation (LMD)
2. Right most derivation (RMD)

Leftmost Derivation (LMD):

→ LMD is a derivation in which the left most non-terminal is replaced first from sentencial form

Right most Derivation (RMD):

→ RMD is a derivation in which the right most non-terminal is replaced first from sentencial form.

Q: Write the Left & Right derivation for given Production rules

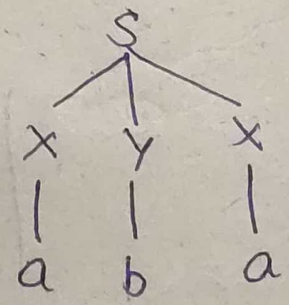
$$P = \{ S \to XYX \quad X \to a \quad Y \to b \}$$

$$V = \{S, X, Y\}$$

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

$$S = \{S\}$$

$\$/P = aba$



LMD:

$S \to XYX$

aYX [Replace left most terminal]

abX [$X \to a$

aba $Y \to b$

$X \to a$]

RMD:

$S \to XYX$ [Replace right most terminal]

XYa [$X \to a$

Xba $Y \to b$

aba $X \to a$]

Q: Write LMD & RMD using CFG given by

$P = \{ S \to aB/bA$

$A \to a/as/bAA$

$B \to b/bs/aBB \}$

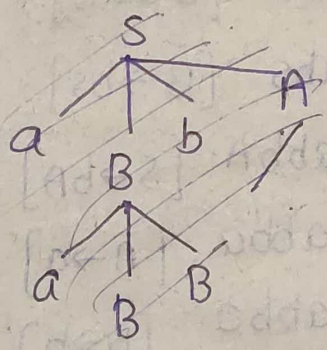
aabbabba

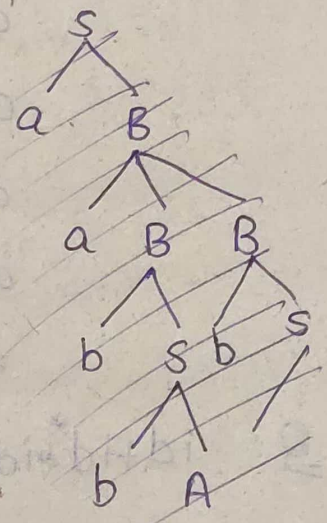
i/p string

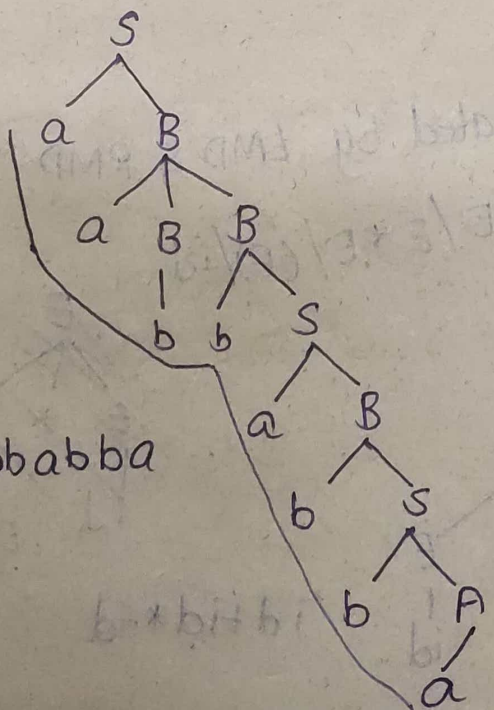
$V = \{S, A, B\}$

$T = \{a, b\}$

$s = \{s\}$







aabbabba

LMD: $S \to aB$

$aaBB \quad [B \to aBB]$

$aabB \quad [B \to b]$

$aabbS \quad [B \to bS]$

$aabbaB \quad [S \to aB]$

$aabbabS \quad [B \to bS]$

$aabbabbA \quad [S \to bA]$

$aabbabba \quad [A \to a]$

RMD

$S \to aB$

$S \to aaBB \quad [B \to aBB]$

$aaBbS \quad [B \to bS]$

$aaBbaB \quad [S \to aB]$

$aaBbabS \quad [B \to bS]$

$aaBbabbA \quad [S \to bA]$

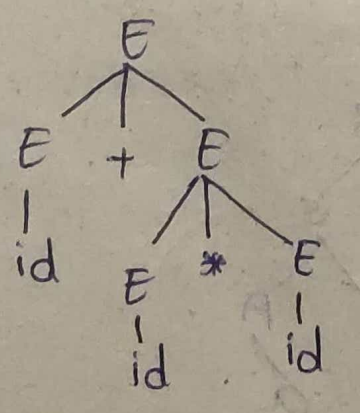
$aaBbabba \quad [A \to a]$

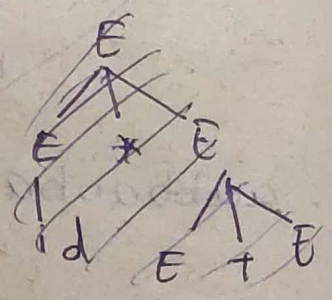
$aabbabba \quad [B \to b]$

Q: $id + id * id$

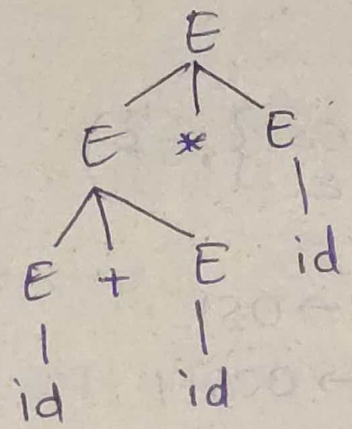
can be generated by LMD & RMD

$E \to E+E / E*E / (E) / id$





$id + id * id$



id+id*id

(2 diagrams are possible)

Q: Construct CFG for the given language

$L = a^n b^{2n}$ $n \geq 1$

if $n=1$

$L = a^1 b^{2(1)} \rightarrow abb$

if $n=2$

$L = a^2 b^{2(2)} \rightarrow aabbbb$

$S \Rightarrow abb$ $\quad S \rightarrow \varepsilon$

$S \rightarrow asbb$

$V = \{S\}$

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

$s = \{S\}$

Q: construct CFG for the languge

$L = a^n b^n$ $n \geq 1$

if $n=1$

$L = a^1 b^1 \rightarrow ab$

if $n=2$

$L = a^2 b^2 \rightarrow aabb$

$S \rightarrow \varepsilon$

$S \rightarrow ab$

$S \rightarrow asb$

$V = \{S\}$

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

$s = \{S\}$

→ Find the language where

$$G = (\{S\}, \{0,1\}, \{S \to 0S1, S \to \varepsilon\}, S)$$

$P = S \to 0S1$
$S \to \varepsilon$

$S \to 0S1$
$S \to 0\varepsilon 1 \to 01$

$S \to 0S1$
$S \to 00S11$
$00\varepsilon 11$
0011

$$L = 0^n 1^n, \quad n \geq 1$$

→ Construct the CFG having any no. of a's over the $\Sigma = \{a\}$.

$G = (V, T, P, S)$
R.E = a^*

$RL = \{\varepsilon, a, aa, aaa, \ldots\}$

$P \to \{S \to \varepsilon$
$S \to as\}$

$V = \{S\}$
$T = \{a, \varepsilon\}$
$S = \{S\}$

$S \to as$
$S \to a\varepsilon$
$S \to a$

$S \to as$
$S \to aas$
$S \to aa\varepsilon$
$S \to aa.$

→ construct the CFG for RE $(0+1)^*$

R.E = $(0+1)^*$
$RL = \{\varepsilon, 0, 1, 00, 11, 10, 01, 010 \ldots\}$
$S \to \varepsilon$

$S \to 0S$
$S \to 1S$

$S \to 0S$
$S \to 0\varepsilon$
$S \to 0$

$S \to 1S$
$S \to 1\varepsilon$
$S \to 1$

$S \to 0S$
$S \to 00S$
$S \to 00\varepsilon$
$S \to 00$

$V = \{S\}$

$T = \{0, 1, \varepsilon\}$

$S = \{S\}$

→ construct CFG which consists of all the strings having atleast 1 occurance of 000.

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

$(0+1)^* \, 000 \, (0+1)^*$

$P \rightarrow \{$

$S \rightarrow ATA$

$A \rightarrow 0A \mid 1A \mid \varepsilon \}$

$T \rightarrow 000\}$

$V = \{S, A, T\}$

$T = \{0, 1, \varepsilon\}$

$S = \{S\}.$

Applications Of finite Automata:

→ Software for designing & checking the behaviour of digital circuit.

→ Software for scanning large bodies of text example, web pages, pattern matching.

→ Software for verifying systems of all types that having finite no. of states.

Eg: 1. Stock Network Transaction

2. Communication Network protocal.

→ Language processing, compiler construction, computer networks, video games, design & digital circuit Bio medical problem solving.