1/5/21:

Unit-1

Natural Language:-

* The language used to communicate with others.
* It is highly flexible.
* It is powerful.
* No special ~~tool~~ effort is needed to learn it.
* It is ambiguous (confusion).

ex: Hindi, English, Telugu e.t.c.

Formal Language:-

* They have strict rules and values to be followed.
* They have semantics and syntaxes which are predefined.
* It is unambiguous.
* It is designed to communicate with machine.

Automata Theory:-

* It is used to implement Formal language.
* It is a mathematical model for automata theory.

Applications of Formal Languages:-

1. Text processing.
2. Compiler design.
3. Hardware design.

Alphabet:-

An alphabet is non-empty and finite set of symbols.

ex: $X = \{0, 1\}$ is alphabet.

$Y = \{0, 1, 2, \ldots\}$ is not alphabet.

String:-

It is a finite sequence of symbols from some alphabet.

ex: "xyz" is string over alphabet $\Sigma = \{a, b, c, \ldots, x, y, z\}$.

Language :-

Language is a collection of strings.

ex: ① $L_1 = \{01, 0011, 000111\}$ is a language over alphabet $\{0,1\}$.

② $L_2 = \{\epsilon, 0, 00, 000, \dots\}$ is a language over alphabet $\{0\}$.

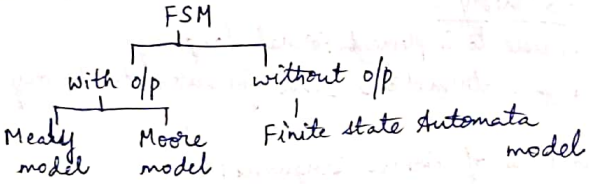
③ $L_3 = \{0^n 1^n 2^n : n \geq 1\}$ is a language over alphabet $\{0,1,2\}$.

↳ for n=1 012
for n=2 001122
for n=3 000111222 . [ϵ is empty string].

Finite-state Machine :-

The finite state system represents a mathematical model of a system with some input. The model finally gives some output.

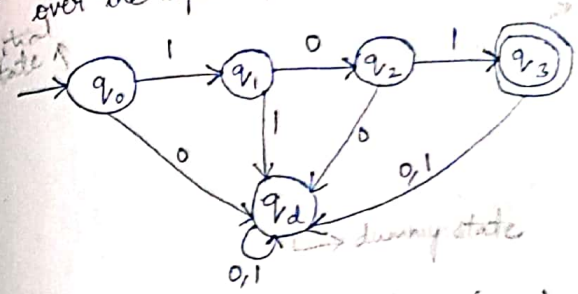
ex: Elevator, switch



A finite automation is formerly defined as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where, (for DFA).

Q is a finite set of states which is non-empty.
Σ is the input alphabet
q_0 is the initial state
F is a set of final states and $F \subseteq Q$.
δ is a transition function or mapping function $Q \times \Sigma \to Q$ using this the next state can be determined depending on the current input.

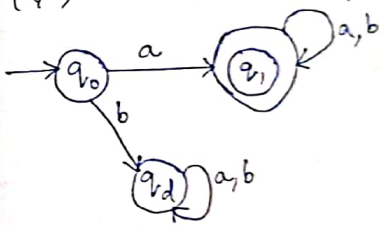
Q1: Design the DFA which accepts the string 101 over the input 0,1.



$Q = \{q_0, q_1, q_2, q_3, q_d\}$
$\Sigma = \{0, 1\}$
$q_0 = q_0$
$F = q_3$

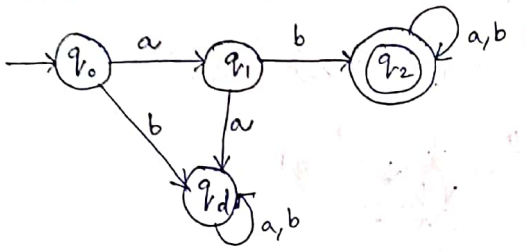
$\delta(q_0, 1) \to q_1$
$\delta(q_0, 0) \to q_d$
$\delta(q_1, 0) \to q_2$
$\delta(q_1, 1) \to q_d$
$\delta(q_2, 1) \to q_3$
$\delta(q_2, 0) \to q_d$
$\delta(q_3, 1) \to q_d$, $\delta(q_3, 0) \to q_d$,
$\delta(q_d, 0) \to q_d$, $\delta(q_d, 1) \to q_d$.

Q2: Construct a DFA which accepts the string over (a,b) and all the strings starts with "a".



$Q = \{q_0, q_1, q_d\}$.
$\Sigma = \{a, b\}$.
$\delta(q_0, a) \to q_1$, $\delta(q_0, b) \to q_d$
$\delta(q_1, a) \to q_1$, $\delta(q_1, b) \to q_1$
$\delta(q_d, a) \to q_d$, $\delta(q_d, b) \to q_d$
$q_0 = q_0$
$F = q_1$

Q3 :- Construct a DFA which accepts the string over (a,b) and all strings start with "ab".

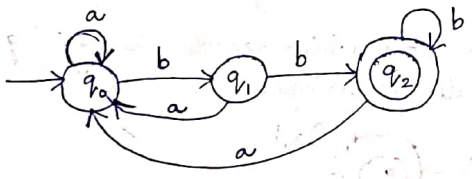


H.W:

① Design a DFA which accepts the string which ends with bb over the inputs $\{a,b\}$.

② Design a DFA that accepts the string containing ab as substring over the input (a,b).

①



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

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

$\delta(q_0, a) \rightarrow q_0$, $\delta(q_0, b) \rightarrow q_1$

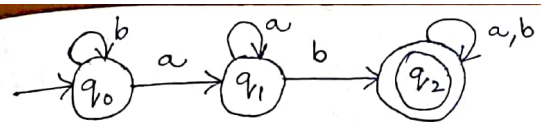
$\delta(q_1, a) \rightarrow q_0$, $\delta(q_1, b) \rightarrow q_2$

$\delta(q_2, a) \rightarrow q_0$, $\delta(q_2, b) \rightarrow q_2$

$q_0 = q_0$

$F = q_2$

②



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

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

$\delta(q_0, a) \rightarrow q_1$, $\delta(q_0, b) \rightarrow q_0$.

$\delta(q_1, a) \rightarrow q_1$, $\delta(q_1, b) \rightarrow q_2$.

$\delta(q_2, a) \rightarrow q_2$, $\delta(q_2, b) \rightarrow q_2$

$q_0 = q_0$

$F = q_2$.

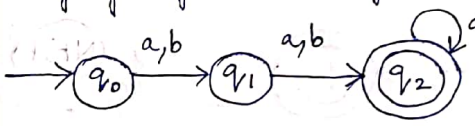
H.W:

① Design a DFA which takes the length of string ≥ 2 over input alphabet $\{a, b\}$.

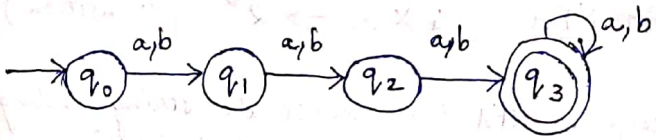
② Design a DFA which takes the length of string ≥ 3 over input alphabet $\{a, b\}$.

③ Construct a DFA which accepts all the strings of length divisible by 3 over $\{a,b\}$.

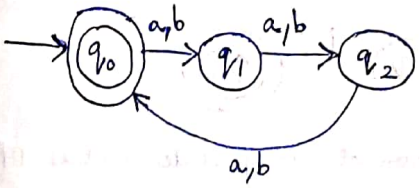
①



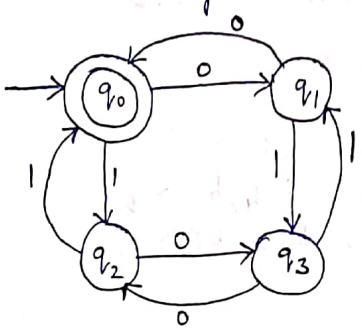
②



③

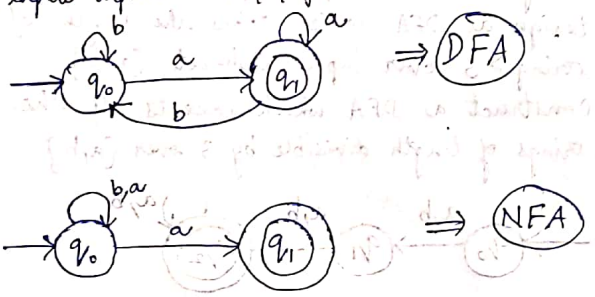


Q1: Design a DFA that accepts the string which contains even no. of 0's and even no. of 1's.



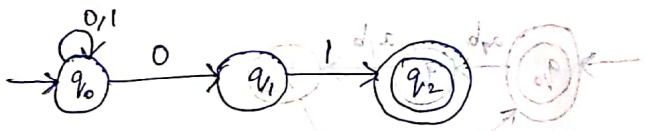
---

① Design a NFA which ends with "a" where the input alphabet is {a,b}.



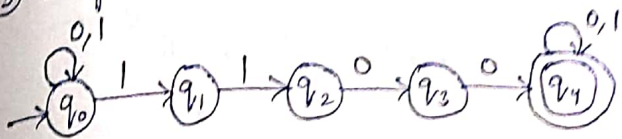
For NFA, $Q \times \Sigma \rightarrow 2^Q$ (transition).

② Design a NFA that access the string ending with "01" where input alphabet is {0,1}.



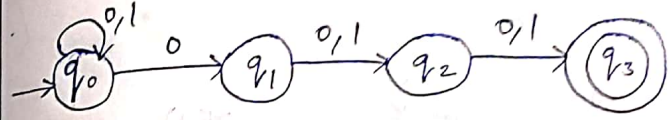
No need of transition at every state unlike DFA.

③ Design a NFA that contains "1100" as sub string.

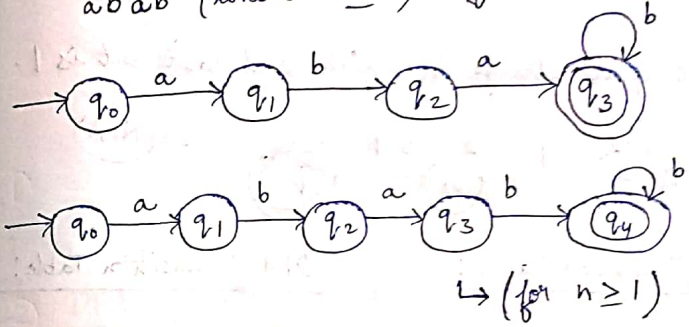


H.W.

① Design a NFA with input alphabet {0,1} where the third symbol from right is 0.



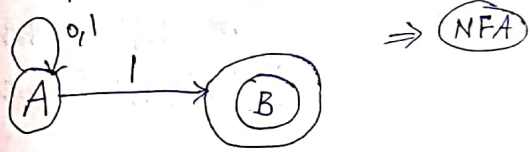
Q: ① Construct a NFA accepting the language $ab\,ab^n$ (where $n \geq 0$).



↳ (for $n \geq 1$)

Conversion of NFA to DFA :-

① Set of all strings that ends with 1 where the input alphabets are {0,1}.



NFA

Transition Table:

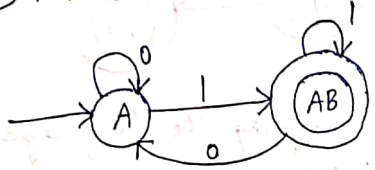
	0	1
→ A	{A}	{A, B}
(B)	∅	∅

DFA Transition Table:

Same as NFA

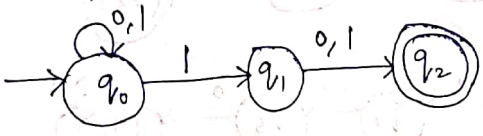
	0	1
→ A	{A}	{AB}
(AB)	{A}	{AB}

[∵ $\delta(AB,0) = \delta(A,0) \cup \delta(B,0)$
$\delta(AB,1) = \delta(A,1) \cup \delta(B,1)$]



⇒ DFA

② Set of all strings in which 2nd last bit is 1.



⇒ NFA

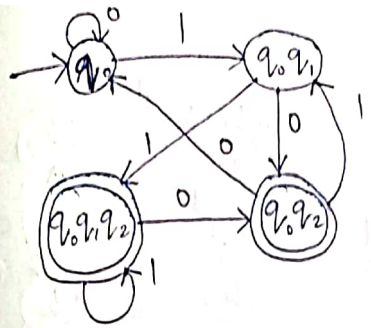
NFA Transition Table:

	0	1
→ q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
(q_2)	∅	∅

↓ q_2 is final here
So all combinations of q_2 will be final here

DFA Transition Table:

	0	1
→ q_0	$\{q_0\}$	$\{q_0 q_1\}$
$q_0 q_1$	$\{q_0 q_2\}$	$\{q_0 q_1 q_2\}$
($q_0 q_2$)	$\{q_0\}$	$\{q_0 q_1\}$
($q_0 q_1 q_2$)	$\{q_0 q_2\}$	$\{q_0 q_1 q_2\}$



⇒ DFA

* No. of states in DFA ≥ No. of states in NFA.
* Max. no. of states in DFA = $2^{\text{no. of states in NFA}}$

Equivalence of 2 finite state machines:-

H.W.

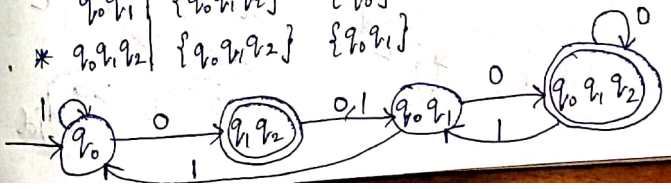
① Convert the following NFA to a DFA.

δ	0	1
→ q_0	$\{q_1, q_2\}$	$\{q_0\}$
q_1	$\{q_0, q_1\}$	∅
* q_2	$\{q_1\}$	$\{q_0, q_1\}$

final state

Sol": DFA Transition Table:

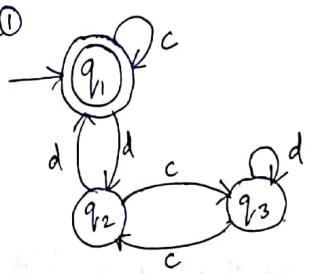
	0	1
→ q_0	$\{q_1, q_2\}$	$\{q_0\}$
* $q_1 q_2$	$\{q_0 q_1\}$	$\{q_0 q_1\}$
$q_0 q_1$	$\{q_0 q_1 q_2\}$	$\{q_0\}$
* $q_0 q_1 q_2$	$\{q_0 q_1 q_2\}$	$\{q_0 q_1\}$



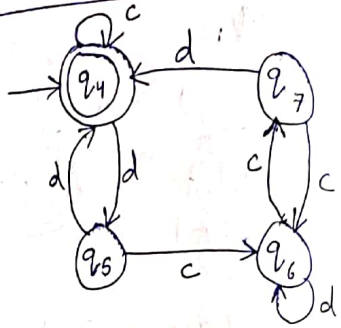
DFA

Equivalence of 2 finite state machines :-

Q. ①



(a)



(b)

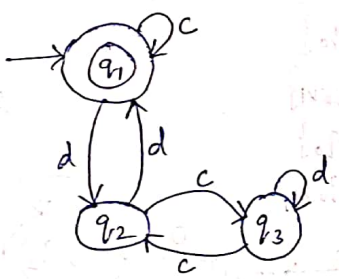
Are 'a' and 'b' machines equivalent ?

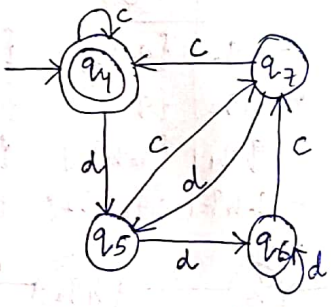
sol^n:

States	c	d
(q_1, q_4)	(q_1, q_4) f f	(q_2, q_5) i i
(q_2, q_5)	(q_3, q_6) i i	(q_1, q_4) f f
(q_3, q_6)	(q_2, q_7) i i	(q_3, q_6) i i
(q_2, q_7)	(q_3, q_6) i i	(q_1, q_4) f f

Hence the given machines are equivalent.

②





sol^n:

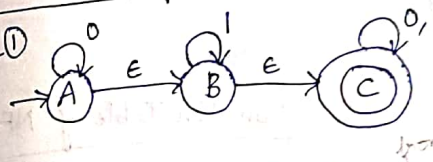
States	c	d
(q_1, q_4)	(q_1, q_4) f f	(q_2, q_5) i i
(q_2, q_5)	(q_3, q_7) i i	(q_1, q_6) f i

Upon giving input d to q_2 it moves to final state q_1 but q_5 moves to intermediate state q_6.

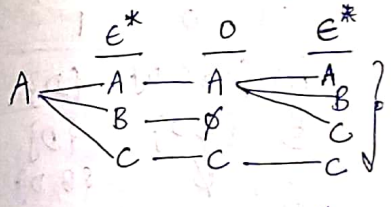
Hence the given machines are not equivalent.

Conversion of ϵ-NFA to NFA :-

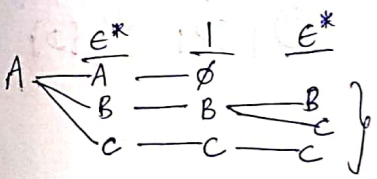
Q. ①



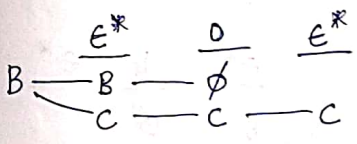
Transition Table of NFA

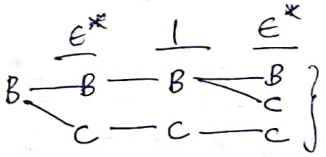


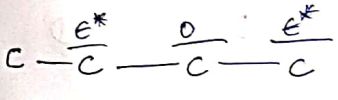
	0	1
→ A	{A, B, C}	{B, C}
B	{C}	{B, C}
C	{C}	{C}

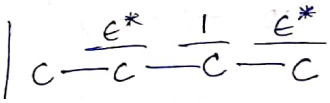


Any state which reaches final state with ϵ is also final state

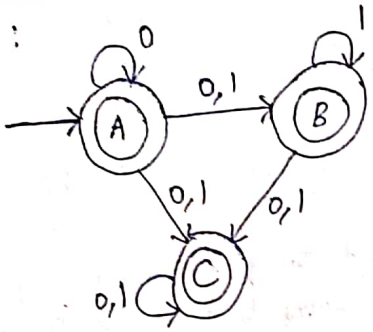




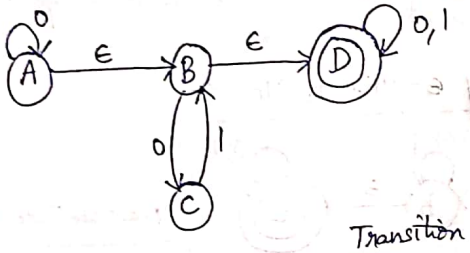




NFA :

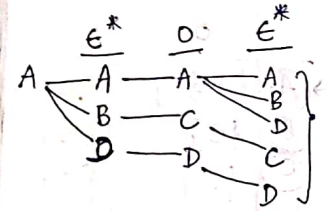


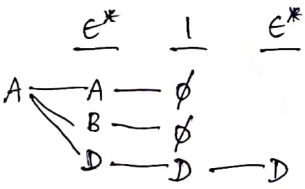
②

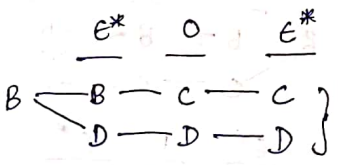


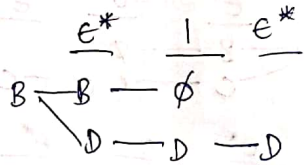
Transition Table of NFA :-

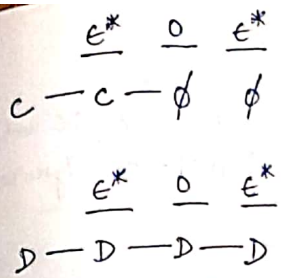
	0	1
→ (A)	{A, B, C, D}	{D}
(B)	{C, D}	{D}
C	∅	{B, D}
(D)	{D}	{D}

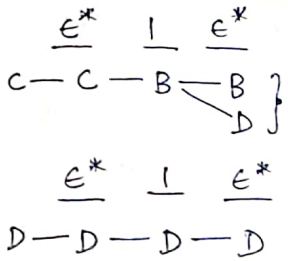




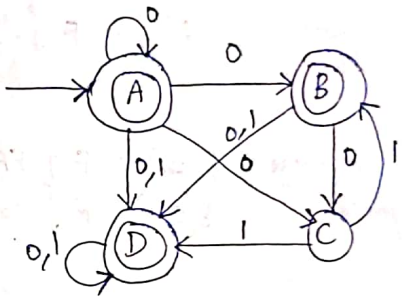








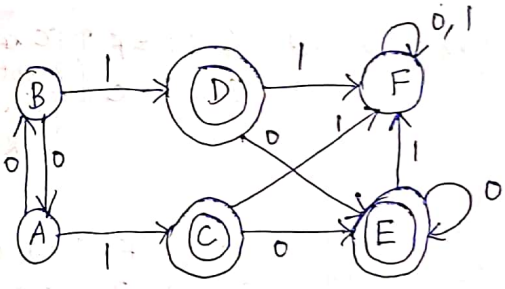
NFA :



Minimisation of DFA :-

Method 1 : & MYHILL NERODE (Table filling method)

Q1 :-



A B C D E F

A
B
C ✓ ✓
D ✓ ✓
E ✓ ✓
F ✓ ✓ ✓ ✓ ✓

put ✓ when 1 state is final & other is not.

Now consider the unchecked boxes.

BA

$\delta(B,0) = A$ } AB is unchecked
$\delta(A,0) = B$ } So ignore.

$\delta(B,1) = D$ } DC unchecked
$\delta(A,1) = C$ } Ignore

DC

$\delta(D,0) = E$ } EE doesn't exist
$\delta(C,0) = E$ } So ignore.

$\delta(D,1) = F$ } FF doesn't exist.
$\delta(C,1) = F$ } Ignore

EC

$\delta(E,0) = E$ } ED not there.
$\delta(C,0) = D$ } Ignore..

$\delta(E,1) = F$ } FF not there
$\delta(C,1) = F$ } So ignore.

ED

$\delta(E,0) = E$ } EE not there.
$\delta(D,0) = E$ } Ignore

$\delta(E,1) = F$ } FF not there
$\delta(D,1) = F$ } Ignore

FA

$\delta(F,0) = F$ } FB is unchecked.
$\delta(A,0) = B$ } So ignore.

$\delta(F,1) = F$ } FC is checked
$\delta(A,1) = C$ } & obtained from FA. So check FA.

FB

$\delta(F,0) = F$ } FA is checked.
$\delta(B,0) = A$ } So check FB.

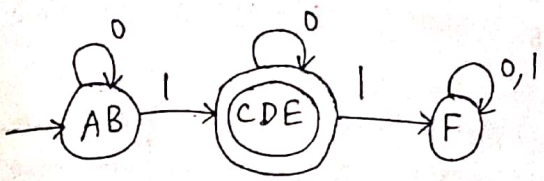
$\delta(F,1) = F$ } FD checked.
$\delta(B,1) = D$ } Also FB is checked

From table, unchecked states are,

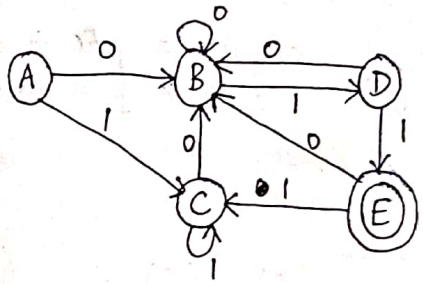
(A,B) (D,C) (E,C) (E,D) → can be combined.

(A,B) (D,C,E) (F) → is added to include all states.

∴ (A,B) (C,D,E) (F).



(2).



A B C D E

A
B ✓
C ✓
D ✓ ✓ ✓
E ✓ ✓ ✓ ✓

Now consider the unchecked states.

BA

$\delta(B,0) = B$ } BB ignore.
$\delta(A,0) = B$ }

$\delta(B,1) = D$ } DC unchecked
$\delta(A,1) = C$ } Ignore.

CA

$\delta(C,0) = B$ } BB ignore
$\delta(A,0) = B$ }

$\delta(C,1) = C$ } CC ignore.
$\delta(A,1) = C$ }

CB

$\delta(C,0) = B$ } BB ignore
$\delta(B,0) = B$ }

$\delta(C,1) = C$ } CD ignore.
$\delta(B,1) = D$ }

DA

$\delta(D,0) = B$ } BB ignore
$\delta(A,0) = B$ }

$\delta(D,1) = E$ } EC is checked
$\delta(A,1) = C$ } So check DC

DB

$\delta(D,0) = B$ } BB ignore
$\delta(B,0) = B$ }

$\delta(D,1) = E$ } ED is checked
$\delta(B,1) = D$ } So check DE

DC

$\delta(D,0) = B$ } BB ignore
$\delta(C,0) = B$ }

$\delta(D,1) = E$ } EC is checked
$\delta(C,1) = C$ } So check DC

(cont.)

From table, unchecked states are,

(B,A) (C,A) (C,B)

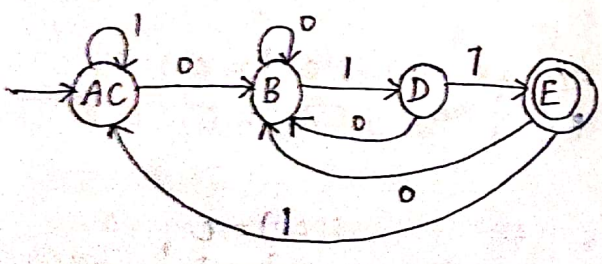
(A,B,C) (D) (E).

	0	1
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

From transition table, AC on input 1 goes to C but B goes to D. So ABC do not form a state.

Hence split (A,C), (B). ✗

∴ (A,C) (B) (D) (E).



Procedure / steps :-

1. Draw a table for all pairs of states (P,Q).
2. Mark all pairs where $P \in F$ and $Q \notin F$.
3. If there are any unmarked pairs (P,Q) such that $[\delta(P,x), \delta(Q,x)]$ is marked, then mark [P,Q] where 'x' is an input symbol.
   Repeat this until no more markings can be done.
4. Combine all the unmarked pairs and make them a single state in the minimized DFA.

(In the exam, first write these steps and then solve the given problem)

(cont)....

Now repeat the procedure for unchecked states...

BA

$\delta(B,0) = B$ } BB ignore
$\delta(A,0) = B$ }

$\delta(B,1) = D$ } DC is checked.
$\delta(A,1) = C$ } So check BA

CA

$\delta(C,0) = B$ } BB ignore
$\delta(A,0) = B$ }

$\delta(C,1) = C$ } CC ignore
$\delta(A,1) = C$ }

CB

$\delta(C,0) = B$ } BB ignore
$\delta(B,0) = B$ }

$\delta(C,1) = C$ } CD is checked.
$\delta(B,1) = D$ } So check CB

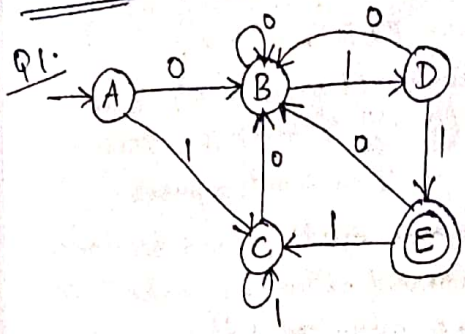
∴ Now, from table,

(A,C) (B) (D) (E)

and then the minimised DFA is

Method 2 : Equivalence Method.

Q1:



Transition Table:

	0	1
→A	B	C
B	B	D
C	B	C
D	B	E
(E)	B	C

0-Equivalence divide by final & non-final states
{A, B, C, D} {E}

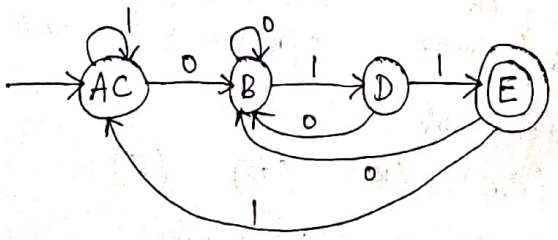
1-Equivalence refer from 0 equ.
{A, B, C} {D} {E}

2-Equivalence refer from 1 equ.
{A, C} {B} {D} {E}

3-Equivalence refer from 2 equ.
{A, C} {B} {D} {E}

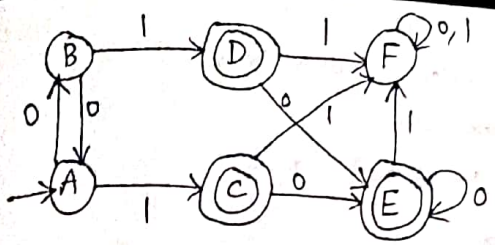
Now the last two equivalences (here 2, 3) became equ.
Hence we can stop here.

∴ (A, C) (B) (D) (E).



Splitting based on inputs & corr. transitions leading to same group or not. (Like partition table STLD).

Q2.



Transition Table:

	0	1
→A	B	C
B	A	D
(C)	E	F
(D)	E	F
(E)	E	F
F	F	F

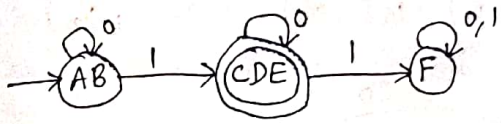
0-Equivalence
{A, B, F} {C, D, E}

1-Equivalence
{A, B} {F} {C, D, E}

2-Equivalence
{A, B} {F} {C, D, E}

Now we can stop.

∴ (A, B), (C, D, E), (F)



(3) Construct the minimised DFA by using the two methods.

	0	1
→ q_0	q_1	q_5
q_1	q_6	q_2
(q_2)	q_0	q_2
q_3	q_2	q_0
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

Equivalence Method :-

Transition table is given in qs.

0 - Equivalence

$\{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}$ $\{q_2\}$

1 - Equivalence

$\{q_0, q_4, q_6\}$ $\{q_3, q_5\}$ $\{q_1, q_7\}$ $\{q_2\}$

2 - Equivalence

$\{q_0, q_4\}$ $\{q_6\}$ $\{q_3, q_5\}$ $\{q_1, q_7\}$ $\{q_2\}$.

3 - Equivalence

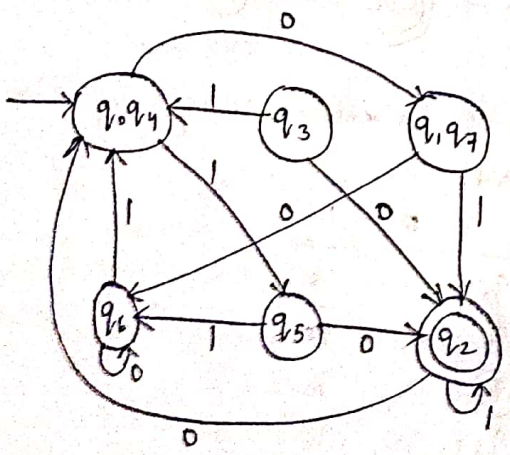
$\{q_0, q_4\}$ $\{q_6\}$ $\{q_3\}$ $\{q_5\}$ $\{q_1, q_7\}$ $\{q_2\}$.

4 - Equivalence

$\{q_0, q_4\}$ $\{q_6\}$ $\{q_3\}$ $\{q_5\}$ $\{q_1, q_7\}$ $\{q_2\}$.

So stop here.

$\therefore$ (q_0, q_4) (q_6) (q_3) (q_5) (q_1, q_7) (q_2)



My Hill Nerode Method :-

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_0								
q_1	✓							
q_2	✓	✓						
q_3	✓	✓	✓					
q_4		✓	✓	✓				
q_5	✓	✓	✓	✓	✓			
q_6	✓	✓	✓	✓	✓	✓		
q_7	✓		✓	✓	✓	✓	✓	

Consider the unchecked states.

q_1q_0

$\delta(q_1, 0) = q_6$ } ignore
$\delta(q_0, 0) = q_1$ }

$\delta(q_1, 1) = q_2$ }
$\delta(q_0, 1) = q_5$ } check q_1q_0

q_3q_0

$\delta(q_3, 0) = q_2$ } check
$\delta(q_0, 0) = q_1$ } q_3q_0

$\delta(q_3, 1) = q_0$ } ignore.
$\delta(q_0, 1) = q_5$ }

q_3q_1

$\delta(q_3, 0) = q_2$ } check
$\delta(q_1, 0) = q_6$ } q_3q_1

$\delta(q_3, 1) = q_0$ } check
$\delta(q_1, 1) = q_2$ } q_3q_1.

q_4q_0

$\delta(q_4, 0) = q_7$ } ignore
$\delta(q_0, 0) = q_1$ }

$\delta(q_4, 1) = q_5$ } ignore.
$\delta(q_0, 1) = q_5$ }

q_4q_1

$\delta(q_4, 0) = q_7$ } ignore
$\delta(q_1, 0) = q_6$ }

$\delta(q_4, 1) = q_5$ } check
$\delta(q_1, 1) = q_2$ } q_4q_1

q_4q_3

$\delta(q_4,0) = q_7$ } check
$\delta(q_3,0) = q_2$ } q_4q_3

$\delta(q_4,1) = q_5$ } Ignore
$\delta(q_3,1) = q_0$ }

q_5q_0

$\delta(q_5,0) = q_2$ } check
$\delta(q_0,0) = q_1$ } q_5q_0

$\delta(q_5,1) = q_6$ } Ignore
$\delta(q_0,1) = q_5$ }

q_5q_1

$\delta(q_5,0) = q_2$ } check
$\delta(q_1,0) = q_6$ } q_5q_1

$\delta(q_5,1) = q_6$ } check
$\delta(q_1,1) = q_2$ } q_5q_1

q_5q_3

$\delta(q_5,0) = q_2$ } Ignore
$\delta(q_3,0) = q_2$ }

$\delta(q_5,1) = q_6$ } Ignore
$\delta(q_3,1) = q_0$ }

q_5q_4

$\delta(q_5,0) = q_2$ } check
$\delta(q_4,0) = q_7$ } q_5q_4

$\delta(q_5,1) = q_6$ } Ignore
$\delta(q_4,1) = q_5$ }

q_6q_0

$\delta(q_6,0) = q_6$ } Ignore
$\delta(q_0,0) = q_1$ }

$\delta(q_6,1) = q_4$ } check
$\delta(q_0,1) = q_5$ } q_6q_0

q_6q_1

$\delta(q_6,0) = q_6$ } Ignore
$\delta(q_1,0) = q_6$ }

$\delta(q_6,1) = q_4$ } check
$\delta(q_1,1) = q_2$ } q_6q_1

q_6q_3

$\delta(q_6,0) = q_6$ } check
$\delta(q_3,0) = q_2$ } q_6q_3

$\delta(q_6,1) = q_4$ } Ignore
$\delta(q_3,1) = q_0$ }

q_6q_4

$\delta(q_6,0) = q_6$ } Ignore
$\delta(q_4,0) = q_7$ }

$\delta(q_6,1) = q_4$ } check
$\delta(q_4,1) = q_5$ } ~~Ignore~~ q_6q_4

q_6q_5

$\delta(q_6,0) = q_6$ } check
$\delta(q_5,0) = q_2$ } q_6q_5

$\delta(q_6,1) = q_4$ } check
$\delta(q_5,1) = q_6$ } q_6q_5

q_7q_0

$\delta(q_7,0) = q_6$ } check
$\delta(q_0,0) = q_1$ } q_7q_0

$\delta(q_7,1) = q_2$ } check
$\delta(q_0,1) = q_5$ } q_7q_0

q_7q_1

$\delta(q_7,0) = q_6$ } Ignore
$\delta(q_1,0) = q_6$ }

$\delta(q_7,1) = q_2$ } Ignore
$\delta(q_1,1) = q_2$ }

q_7q_3

$\delta(q_7,0) = q_6$ } check
$\delta(q_3,0) = q_2$ } q_7q_3

$\delta(q_7,1) = q_2$ } check
$\delta(q_3,1) = q_0$ } q_7q_6

q_7q_4

$\delta(q_7,0) = q_6$ } Ignore
$\delta(q_4,0) = q_7$ }

$\delta(q_7,1) = q_2$ } check
$\delta(q_4,1) = q_5$ } q_7q_4

q_7q_5

$\delta(q_7,0) = q_6$ } check
$\delta(q_5,0) = q_2$ } q_7q_5

$\delta(q_7,1) = q_2$ } check
$\delta(q_5,1) = q_6$ } q_7q_5

q_7q_6

$\delta(q_7,0) = q_6$ } Ignore
$\delta(q_6,0) = q_6$ }

$\delta(q_7,1) = q_2$ } check
$\delta(q_6,1) = q_4$ } q_7q_6

Now repeat the same procedure.

$q_4 q_0$

$\delta(q_4, 0) = q_7$ } Ignore
$\delta(q_0, 0) = q_1$ }

$\delta(q_4, 1) = q_5$ } Ignore
$\delta(q_0, 1) = q_5$ }

$q_5 q_3$

$\delta(q_5, 0) = q_2$ } Ignore
$\delta(q_3, 0) = q_2$ }

$\delta(q_5, 1) = q_6$ } Check
$\delta(q_3, 1) = q_0$ } $q_6 q_0$

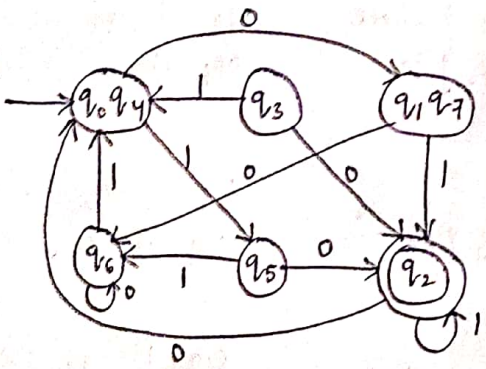
$q_7 q_1$

$\delta(q_7, 0) = q_6$ } Ignore
$\delta(q_1, 0) = q_6$ }

$\delta(q_7, 1) = q_2$ } Ignore
$\delta(q_1, 1) = q_2$ }

∴ The unchecked states from table are,
(q_4, q_0) $(q_7 q_1)$.

∴ $(q_0 q_4)$ $(q_1 q_7)$ (q_3) (q_5) (q_6) (q_2)



Moore and Mealy Machines :-

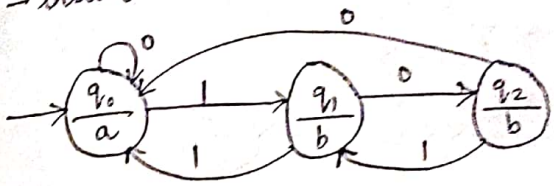
Moore Machine :-

It is a finite state machine where the output depends only on present state.

$\{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$ — 6 tuples in Moore machine

Q – set of states
Σ – input symbols
Δ – o/p symbols
δ – i/p transition function
λ – o/p transition function
q_0 – initial state

ex:



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

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

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

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

q_0 — 0 → q_0, 1 → q_1
q_1 — 0 → q_2, 1 → q_0
q_2 — 0 → q_0, 1 → q_1

$\lambda : Q \rightarrow \Delta$

$q_0 \rightarrow a$
$q_1 \rightarrow b$
$q_2 \rightarrow b$

$q_0 \rightarrow q_0$ (initial state).

Transition Table :-

Current state	Next state 0	Next state 1	o/p
→ q_0	q_0	q_1	a
q_1	q_2	q_0	b
q_2	q_0	q_1	b

(p.t.o)

→ Suppose 00110 is the input given. What is o/p?

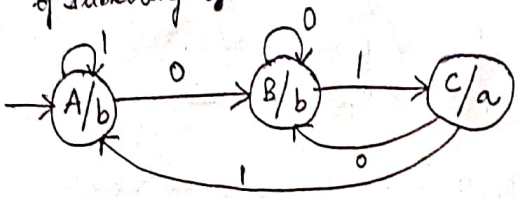
00110
a a b a a

[irrespective of input first write the o/p of initial state]

→ 011
a a b a

∴ n → n+1
i.e if there are n inputs, there will be n+1 outputs.

① Construct a Moore machine that takes set of strings over {0,1} and counts the no. of occurences of substring "01".

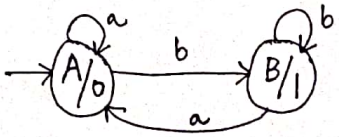


o/p 'a' comes for each occurence of '01'.

→ Suppose 011001 is i/p given. Then o/p is..
b b a b b a

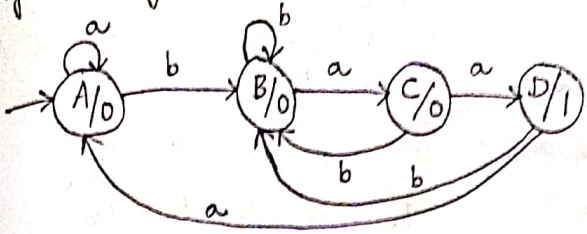
→ 00 1
b b b a

② Construct a Moore Machine that takes sets of strings over {a,b} and counts no. of b's.



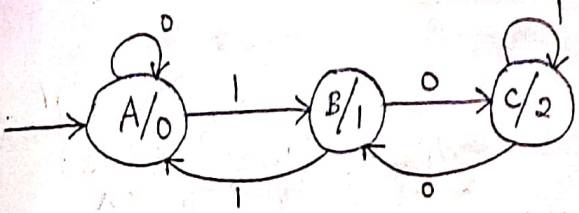
→ abb
0 0 1 1

③ Construct a Moore machine that takes set of all strings over {a,b} and counts no. of occurences of substring 'baa'.



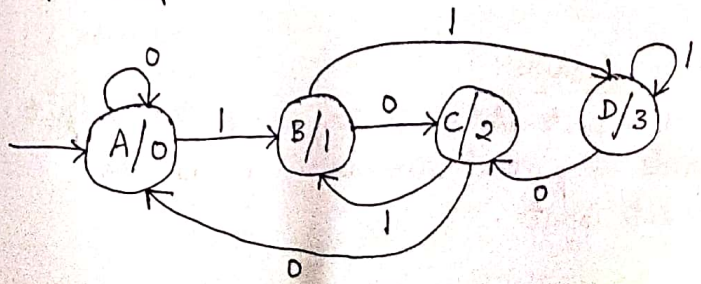
H.W:

① Construct a Moore machine that takes binary numbers as input and produces residue modulo 3 as output.



0 (0) - 1 (01) - 2 (10) -
3 (011) - 4 (100) - 5 (101) -
6 (110) - 7 (111) - 8 (1000) -

② Construct a Moore Machine that takes binary numbers as input and produces residue modulo 4 as output.

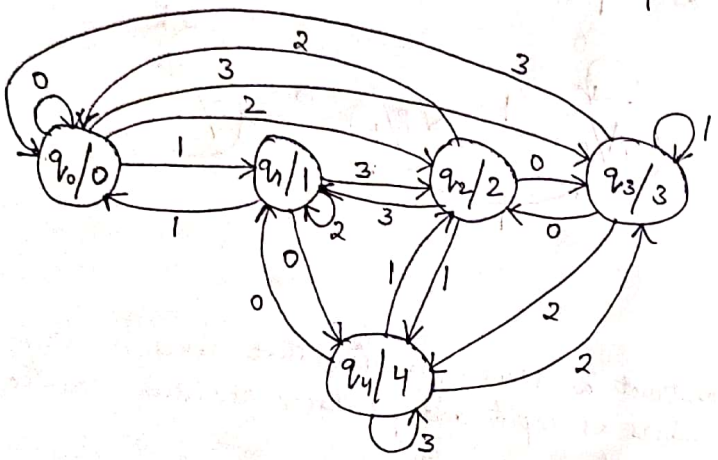


① Construct a Moore Machine that takes gray code numbers as input and produces residue modulo 5 as output.

i/p : 0 1 2 3
o/p : 0 1 2 3 4

Transition table: (using trick)

	0	1	2	3
q_0	q_0	q_1	q_2	q_3
q_1	q_4	q_0	q_1	q_2
q_2	q_3	q_4	q_0	q_1
q_3	q_2	q_3	q_4	q_0
q_4	q_1	q_2	q_3	q_4



Mealy Machine:-

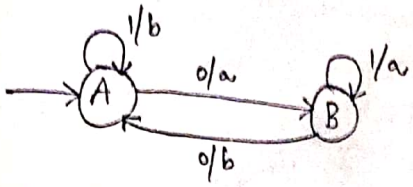
It is a finite state machine where the output depends on present state as well as the present input.

$(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ — 6 tuples in Mealy Machine

$\delta : Q \times \Sigma \to Q$

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

ex:

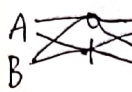


$Q \to \{A, B\}$

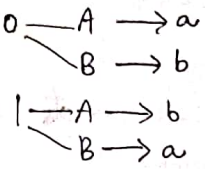
$\Sigma \to \{0, 1\}$

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

δ :



λ :



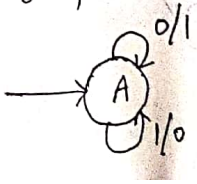
q_0 : A

Transition Table:-

Current State (CS)	Next State (NS)			
	i/p = 0		i/p = 1	
	State	o/p	State	o/p
A	B	a	A	b
B	A	b	B	a

① Construct a Mealy machine that produces 1's complement of any binary input strings.

i/p : 0 1
o/p : 0 1



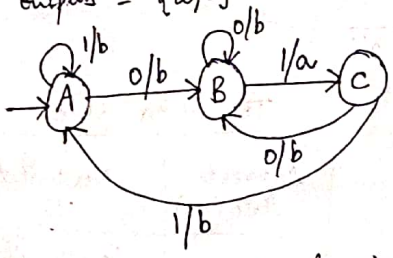
∴ $n \longrightarrow n$

i.e if there are n inputs there will be n o/p

(2) Construct a Mealy machine that prints 'a' whenever the sequence '01' is encountered.

inputs = {0,1}

outputs = {a,b}



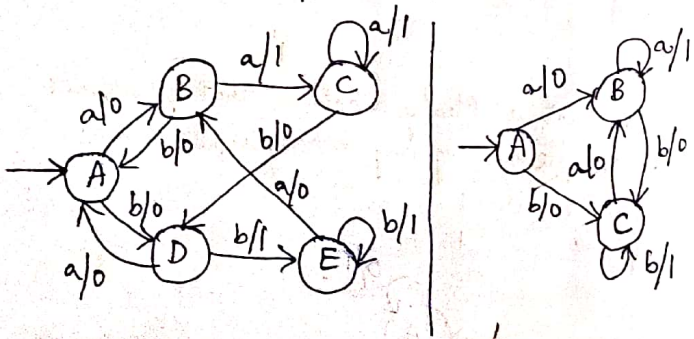
→ Suppose 011 as input given

bab

→ 100 → 1001

bbb bbba

(3) Design a Mealy machine that accepts 'aa' or 'bb' in the ending of string.



Choose the one with less no. of states ↓

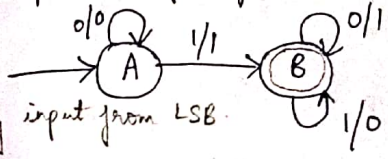
(4) Design a Mealy machine to find the 2's complement of a given number.

001 → 111

111 ✓

01 → 11

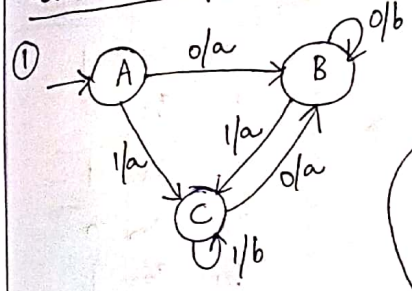
11 ✓

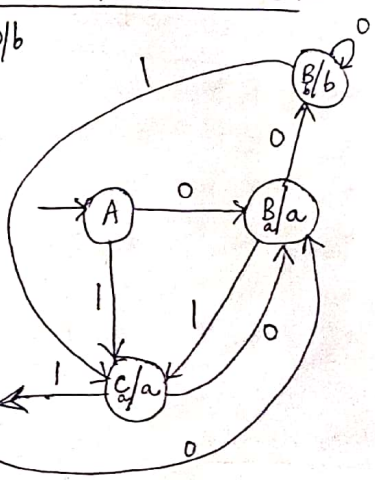


Giving input from LSB

Conversion of Mealy machine to Moore Machine :-

(1)





Me → 3 ✓

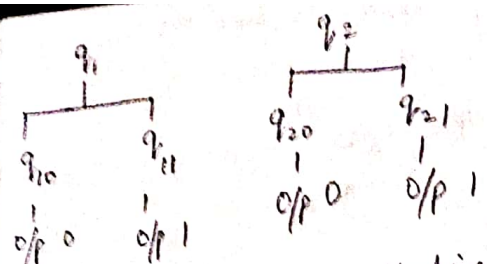
Mo → 5 ✓

$S \times n = 3 \times 2 = 6$ (max in Moore). here $5 < 6$ ✓

S → states in Me, n → no. of inputs.

(2) → Transition Table for Mealy Machine :

State	a	b
→ q_0	$q_3, 0$	$q_1, 1$
q_1	$q_0, 1$	$q_3, 0$
q_2	$q_2, 1$ *	$q_2, 0$ *
q_3	$q_1, 0$	$q_0, 1$



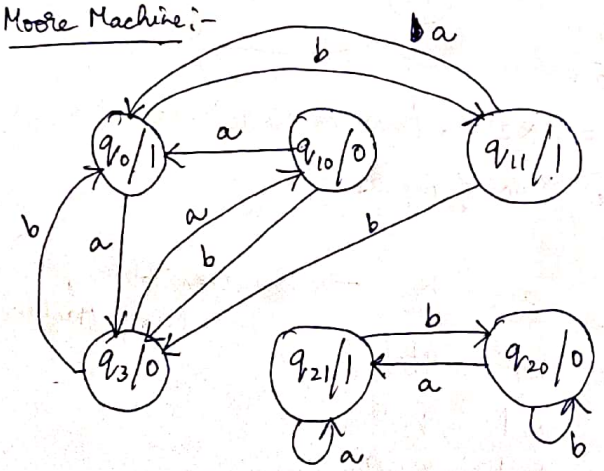
Transition table for Moore Machine:-

State	a	b
→ q_0	q_3, 0	q_{11}, 1
q_{10}	q_0, 1	q_3, 0
q_{11}	q_0, 1	q_3, 0
q_{20}	q_{21}, 1	q_{20}, 0
q_{21}	q_{21}, 1	q_{20}, 0
q_3	q_{10}, 0	q_0, 1

→

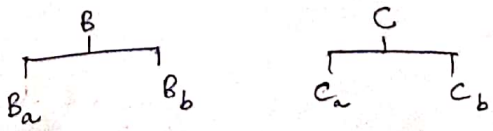
State	a	b	o/p
q_0	q_3	q_{11}	1
q_{10}	q_0	q_3	0
q_{11}	q_0	q_3	1
q_{20}	q_{21}	q_{20}	0
q_{21}	q_{21}	q_{20}	1
q_3	q_{10}	q_0	0

Moore Machine:-



Q(1): Transition table for Mealy Machine:-

State	0	1
→ A	B_a	C_a
B	B_b	C_a
C	B_a	C_b



Transition Table for Moore Machine:-

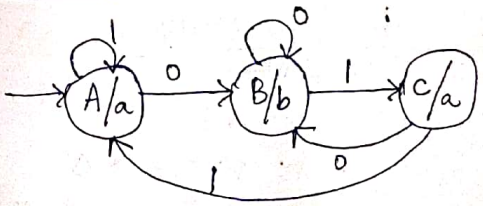
State	0	1
→ A	B_a, a	C_a, a
B_a	B_b, b	C_a, a
B_b	B_b, b	C_a, a
C_a	B_a, a	C_b, b
C_b	B_a, a	C_b, b

→

State	0	1	o/p
A	B_a	C_a	—
B_a	B_b	C_a	a
B_b	B_b	C_a	b
C_a	B_a	C_b	a
C_b	B_a	C_b	b

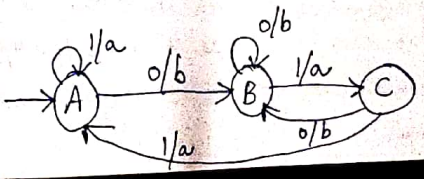
Conversion of Moore machine to Mealy Machine:-

(1)



– Moore

Mealy:



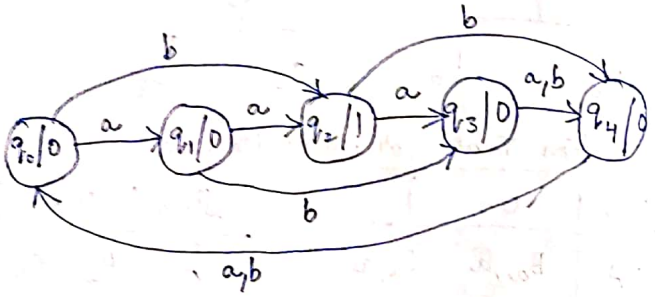
Moore Transition Table:-

State	0	1	o/p
→A	B	A	a
B	B	C	b
C	B	A	a

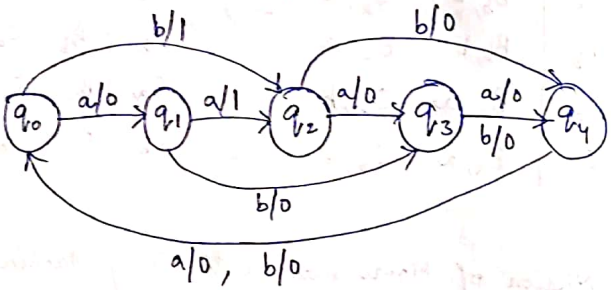
Mealy Transition Table:-

State	0		1	
	NS	o/p	NS	o/p
→A	B	b	A	a
B	B	b	C	a
C	B	b	A	a

(2)



Mealy:



Moore	Mealy
1. Definition.	1. Definition.
2. $n \rightarrow n+1$	2. $n \rightarrow n$
3. $\lambda : Q \rightarrow \Delta$	3. $\lambda : Q \times \Sigma \rightarrow \Delta$
4. When converting Moore → Mealy No change in no. of states.	4. When converting Mealy → Moore there is increase/change in states.
5. o/p is synchronous with clock. (Slower).	5. o/p is asynchronous. (Faster)

31/5/21.

Unit-2.

Q1: Write down the regular expression where $\Sigma = a, b$, having,

(i) no string → RE - ϕ

(ii) length 0 → ϵ

(iii) length 1 → $(a+b)$

(iv) length 2 → $aa + ab + ba + bb$
$a(a+b) + b(a+b)$
$= (a+b)(a+b)$

(v) length 3 → $(a+b)(a+b)(a+b)$

(vi) atmost 1
$(\epsilon + a + b)$ → $\{\epsilon, a, b\}$
R.E → Regular set

(2) Write the regular expression for all strings of 0's and 1's.

Sol^n: First write the language for 0's and 1's.
$\{\epsilon, 0, 1, 01, 00, 10, 11, 000, \text{---}\}$.
Regular expression is $(0+1)^*$

(3) Set of all strings of 0's and 1's ending with 00.

Sol^n: Language $\{00, 000, 100, 1000, 0100, 01000$
$\{\epsilon, 0, 1, 10, 01, 11, 00, \text{--}\}00$
Regular expression is $(0+1)^*00$

(4) Set of all strings of 0's and 1's beginning with 0 and ending with 1.

Sol^n: Language $\{01, 001, 011, 0001, 0101, \ldots\}$
$0\{\epsilon, 0, 1, 00, 10, 11, 01, \ldots\}1$

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

(5) Set of all strings with atleast 2 consecutive 0's.

Sol^n: Language $\{00, 001, 100, 000, \ldots\}$

R.E is $(0+1)^*00(0+1)^*$

(6) $\Sigma = a, b$, Write R.E where the length of strings is atleast 2.

Sol^n: Language $\{bb, ba, bab, babb, baaba, \ldots\}$

R.E is $\underbrace{(a+b)(a+b)}_{\text{for 2}}(a+b)^*$

(7) Write R.E where $\Sigma = a, b$, having length atmost 2.

Sol^n: $L = \{\epsilon, a, b, aa, bb, ab, ba\}$

R.E is $(\epsilon+a+b)(\epsilon+a+b)$.

(8) Write R.E where $\Sigma = a, b$ having length of string is even.

Sol^n: $L = \{\epsilon, aa, ab, abab, ababbb, \ldots\}$

R.E is $((a+b)(a+b))^*$ → if we put 0 in place of * then length 0.

(9) Write R.E where $\Sigma = a, b$ having length of string is odd.

Sol^n: $L = \{a, b, aaa, bbb, \ldots\}$

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

(10) Write R.E where $\Sigma = a, b$ and whose length is divisible by 3.

Sol^n: $L = \{\epsilon, aaa, bbb, aabbab, \ldots\}$

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

Identity Rules for Regular Expressions :-

P and Q are two equivalent R.Es (i.e P and Q represent the same set of strings), then to simplify the R.Es, the foll. identity rules can be used :

1) $\phi + R = R$, $\epsilon + R = R + \epsilon$
2) $\epsilon R = R\epsilon = R$
3) $R + R = R$
4) $RR^* = R^*R = R^+$
5) $\epsilon + RR^* = R^* = \epsilon + R^*R$
6) $\phi R = R\phi = \phi$
7) $\epsilon^* = \epsilon$, $\phi^* = \epsilon$
8) $(R^*)^* = R^*$
9) $(PQ)^*P = P(QP)^*$
10) $(P+Q)R = PR + QR$
11) $R(P+Q) = RP + RQ$
12) $(P+Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$

Q① Write down the R·E whose length is congruent to 2 mod 3.
↳ any number divided by 3 must give remainder 2

sol^n: Lengths will be 2, 5, 8, 11, ...
R·E is $(a+b)(a+b).((a+b)(a+b)(a+b))^*$
↓ +2 ↓ divisible by 3

$* = 0$ len = 2
$* = 1$ len = 5
$* = 2$ len = 8.

② R·E for length congruent to 3 mod 4.
sol^n: Lengths will be 3, 7, 11, 15, ...
R·E is $(a+b)(a+b)(a+b)((a+b)(a+b)(a+b)(a+b))^*$

③ Write R·E where $\Sigma = a, b$ whose no. of a's are exactly 2.
sol^n: $L = \{ aa, aab, baa, aabb, aba, \ldots \}$
R·E is $b^*ab^*ab^*$

④ Write R·E whose having atleast 2 a's.
sol^n: $L = \{ aa, abaa, baa, ababa, \ldots \}$
R·E is $b^*ab^*a(a+b)^*$.

⑤ Write R·E having atmost 2 a's.
sol^n: $L = \{ ab, bb, aab, a, abab, \ldots \}$
R·E is $b^*(\varepsilon+a)b^*(\varepsilon+a)b^*$

$\varepsilon\varepsilon$ — 0 a's
εa — 1 a
aa — 2 a's

⑥ Write R·E whose no. of a's is even.
sol^n: $L = \{ b, aab, baab, babaaba, \ldots \}$
R·E is $b^* + (b^*ab^*ab^*)^*$

⑦ Write R·E for set of all strings beginning with 1 or 0 but not having consecutive 0's.
sol^n: $L = \{ 10, 101, 010, 0101, \ldots \}$
R·E for string that does not contain 2 consecutive 0's is $(1+10)^*$
~~R·E starting with 0 or 1 is $(0+1)$~~
Final R·E is $(0+\varepsilon)(1+10)^*$ [∵ for 10 to be acceptable]
$\varepsilon 10 = 10$

⑧ Write R·E whose set of strings is 0's followed by 1, followed by 2 such that atleast one 0 followed by atleast one 1 which is followed by atleast one 2.
sol^n: R·E is $0^+1^+2^+$ (we cannot put * since atleast 012 has to be there (ϵ is not accepted)).

⑨ Write a R·E where $\Sigma = 0, 1$ whose last 2 symbols are same.
sol^n: $L = \{ 00, 11, 000, 011, 100, \ldots \}$
R·E is $(0+1)^*(00+11)$

⑩ Write down the R·E in which every 0 is

followed by atleast 2 two 1's.

Soln: L = {1011, 011, 011011, --}

R.E is $(1+011)^*$

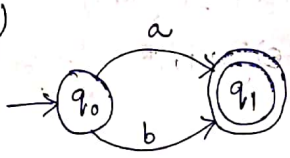
[∵ $(0+1)^*$ contains all strings of 0 and 1's with no condition.

but in this qs 0 must be followed by 11.

so $(011+1)^*$]

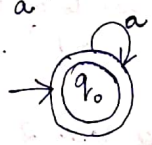
Regular expression to Finite Automata :-

(1) $(a+b)$

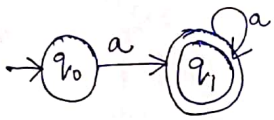


(2) (ab)

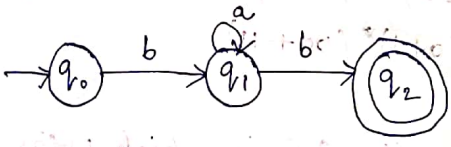
(3) a^*



(4) a^+

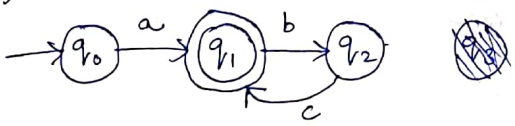


(5) ba^*b

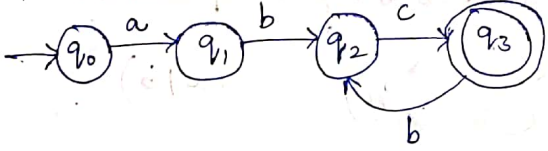


(6) $(a+b)c$

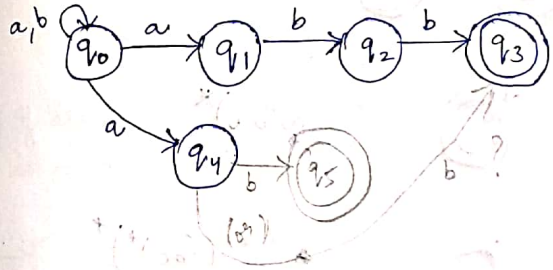
(7) $a(bc)^*$



(8) $a(bc)^*+$



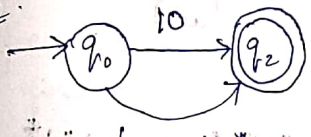
(9) $(a+b)^*(abb + a^+b)$



(10) $10 + (0+11)0^*1$

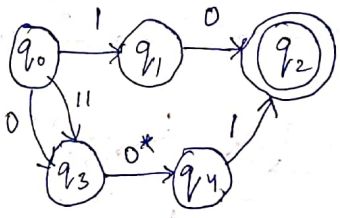
1 2

1.

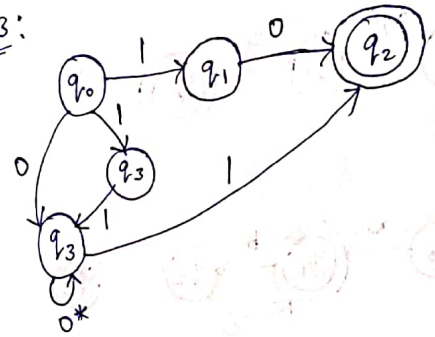


$(0+11)0^*1$

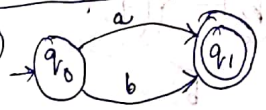
2.



Step 3:



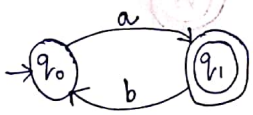
Finite Automata to Regular Expression :-

1. 

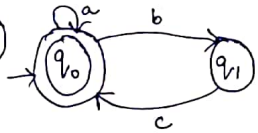
$(a+b)$

2. 

ab^*c

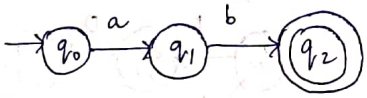
3. 

$a(ba)^*$

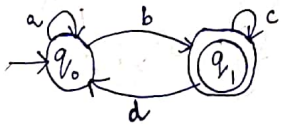
4. 

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

$= (a+bc)^* = (a^*(bc)^*)^*$ ✓ ✓

5. 

(ab)

6. 

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

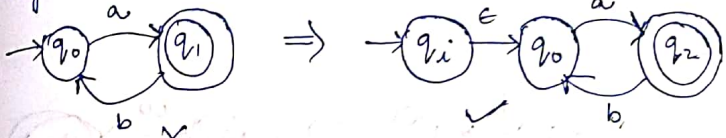
$= a^*b(c+(da^*b))^*$

Methods to convert Finite Automata to R.E :-

1. State Elimination method.
2. Arden's theorem.

State Elimination method :-

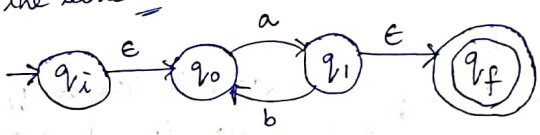
1) Initial state should not contain any incoming edges.



✓ ✗

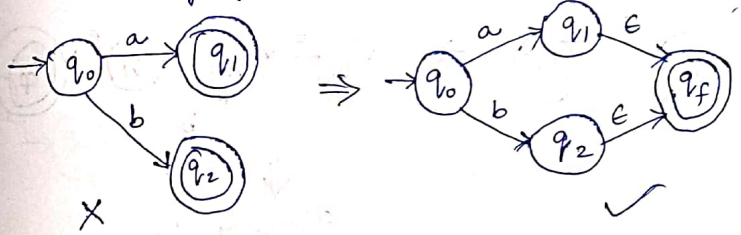
2) Final state should not contain any outgoing edge.

→ for the same ex:



✓

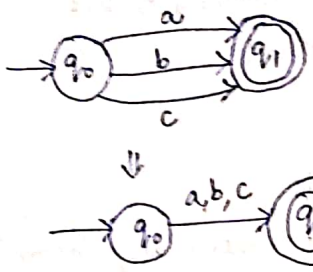
3) If there exists more than 1 finite state, make it is a single finite state.



✗ ✓

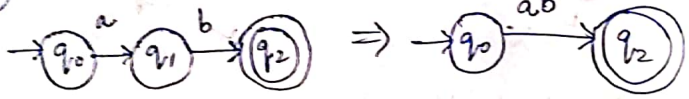
4) Eliminate all the states except the initial and final state (if possible).

ex: (1)



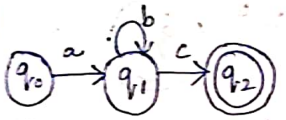
Hence the R·E is $(a+b+c)$.

(2)



Hence R·E is (ab).

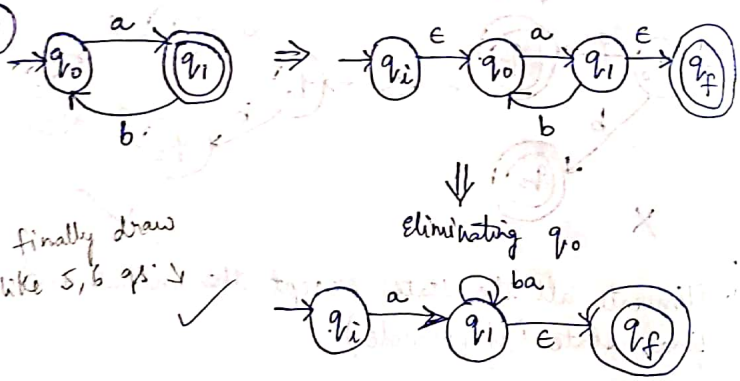
(3)



⇒ cannot remove q_1 as there is a loop on it.

Hence R·E is ab^*c. → finally draw like q5, 6

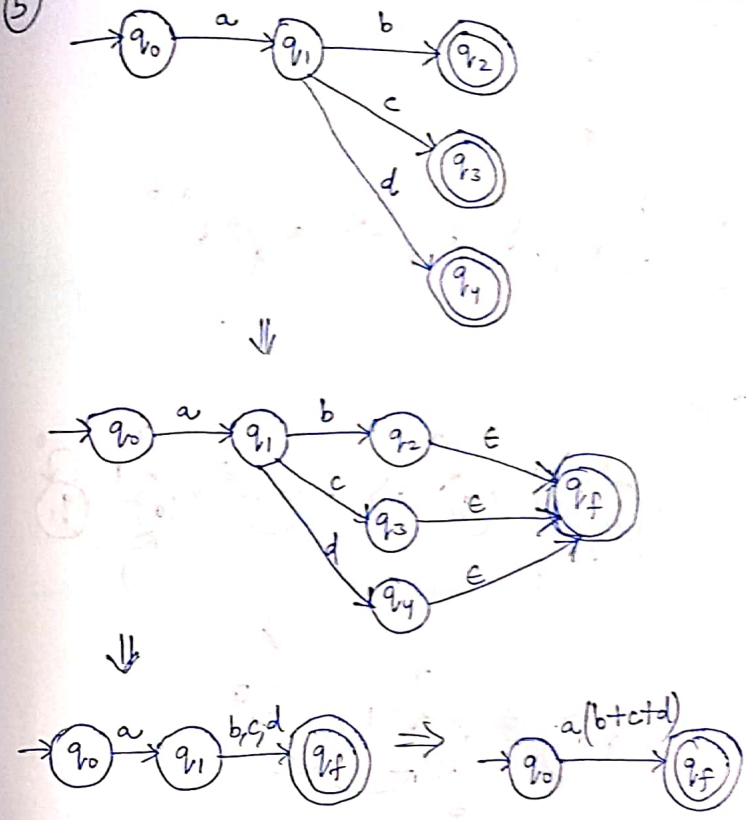
(4)



finally draw like 5, 6 qs ↓ ✓

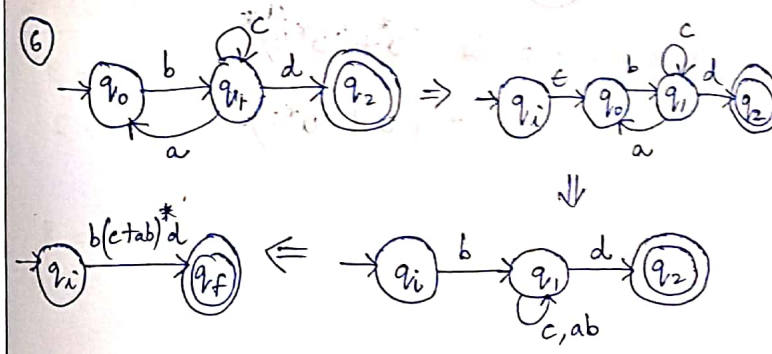
Hence R·E is → $a(ba)^*$.

(5)



∴ R·E is $a(b+c+d)$.

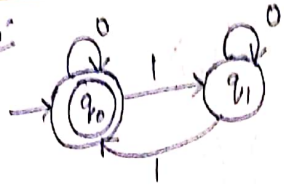
(6)



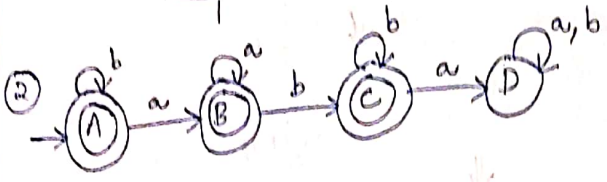
∴ R·E is $b(c+ab)^*d$

H.W:

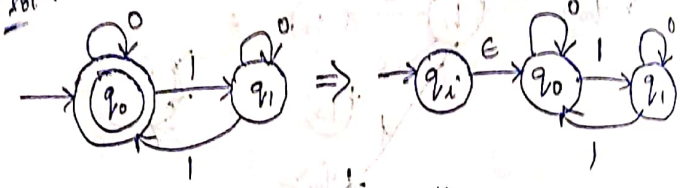
①

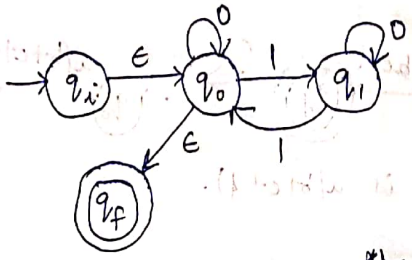


②



① Soln:

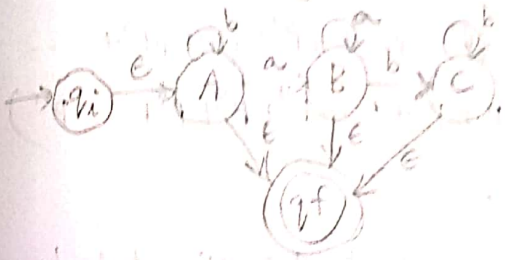


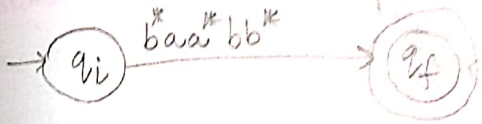


Hence the R.E is $(0^*+(10^*1)^*)^*$

$= (0+10^*1)^*$

② Soln:





$(b^*+\epsilon)(aa^*+\epsilon)(bb^*+\epsilon)$

$b^*(\epsilon+aa^*)(\epsilon+bb^*)$

Arden's Theorem:-

$R = Q + RP$ (P does not belong to null)

$R = QP^*$

$R = Q+RP$ → Proof:

$= Q+(Q+RP)P$

$= Q + QP + RP^2$

$= Q + QP + (Q+RP)P^2$

$= Q + QP + QP^2 + RP^3$

$= Q + QP + QP^2 + (Q+RP)P^3$

$= Q + QP + QP^2 + QP^3 + RP^4$

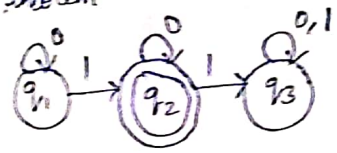
$\vdots$

$= Q + QP + QP^2 + QP^3 + QP^4 + QP^5 + QP^6 + \dots$

$= Q(\epsilon + P + P^2 + P^3 + P^4 + P^5 + P^6 + \dots P^n)$

$\therefore \boxed{R = QP^*}$.

① Convert finite automata to R.E using Arden's theorem.



$q_{r1} = \epsilon + q_{r1}0 \quad \text{—①}$

$q_{r2} = q_{r2}0 + q_{r1}1 \quad \text{—②}$

$q_{r3} = q_{r2}1 + q_{r3}0 + q_{r3}1 \quad \text{—③}$

Consider eq.① $\Rightarrow$ $\underset{R}{q_{r1}} = \underset{}{\epsilon} + \underset{R}{q_{r1}}\underset{P}{0}$

$R = Q + RP$

$q_{r1} = \epsilon \cdot 0^* = 0^* \quad \text{—④}$

Now, $q_{r2} = q_{r2}0 + q_{r1}0$

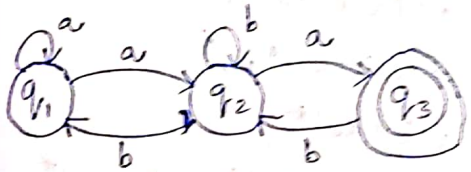
$\underset{R}{q_{r2}} = \underset{R\ P}{q_{r2}0} + \underset{Q}{0^*1}$

$q_{r2} = 0^*10^*$

q_{r3} is a dead state. So delete it.

$\therefore$ R.E is 0^*10^*

②



$q_{r1} = \epsilon + q_{r1}a + q_{r2}b \quad \text{—①}$

$q_{r2} = q_{r2}b + q_{r1}a + q_{r3}b \quad \text{—②}$

$q_{r3} = q_{r2}a \quad \text{—③}$

Consider eq.①, $\underset{R}{q_{r1}} = \underset{}{\epsilon} + \underset{Q}{q_{r2}b} + \underset{R\ P}{q_{r1}a}$ ✗

$q_{r1} = (\epsilon + q_{r2}b)a^* = q_{r2}ba^*$

Now, $\underset{R}{q_{r2}} = \underset{Q}{q_{r1}a + q_{r3}b} + \underset{R\ P}{q_{r2}b}$

$q_{r2} = (q_{r1}a + q_{r3}b)b^*$ ✗

put ③ in ②,

$q_{r2} = q_{r2}b + q_{r1}a + q_{r2}ab$

$\underset{R}{q_{r2}} = \underset{Q}{q_{r1}a} + \underset{R\ P}{q_{r2}(b+ab)}$

$q_{r2} = q_{r1}a(b+ab)^* \quad \text{—④}$

put ④ in ①,

$q_{r1} = \epsilon + q_{r1}a + q_{r1}a(b+ab)^*b$

$q_1 = \epsilon + q_1(a + a(b+ab)^*b)$

$R \quad Q \quad R \quad P$

$q_1 = \epsilon(a+(a(b+ab)^*b)^*$

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

$q_2 = q_1 a(b+ab)^*$

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

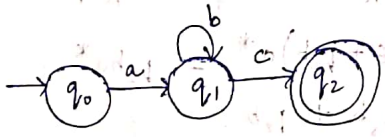
$q_3 = q_2 a$

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

$q_3 = a^2(b+ab)^*(a+a(b+ab)^*b)^*$

↳ is R.E

(3)



$q_0 = \epsilon$ —①

$q_1 = q_0 a + q_1 b$ —②

$q_2 = q_1 c$ —③

Now, $q_1 = a + q_1 b$

$R \quad Q \quad R \quad P$

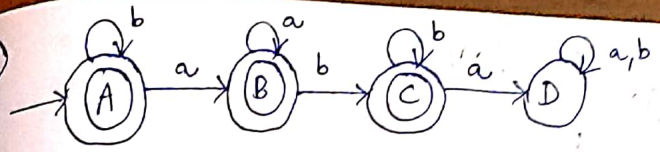
$q_1 = ab^*$

$q_2 = q_1 c$

$= ab^* c$

∴ The R.E is ab^*c.

(4)



$A = Ab + \epsilon$ —①

$B = Aa + Ba$ —②

$C = Bb + \epsilon b$ —③

eq.① ⇒ $A = Ab + \epsilon$

$R \quad R\,P \quad Q$

$A = \epsilon b^* \Rightarrow A = b^*$ —④

eq.② ⇒ $B = Aa + Ba$

$B = b^*a + Ba$

$R \quad Q \quad R\,P$

$B = b^*aa^*$ —⑤

eq.③ ⇒ $C = Bb + cb$,

$C = b^*aa^*b + Cb$

$R \quad Q \quad R\,P$

$C = b^*aa^*bb^*$ —⑥

Final R.E is $A+B+C$ [∵ A, B, C are final states].

$= b^* + b^*aa^* + b^*aa^*bb^*$

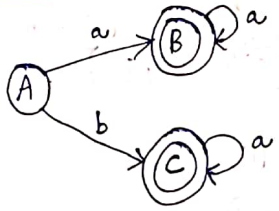
$= b^*(aa^* + \epsilon + aa^*bb^*)$

$= b^*(aa^*(bb^* + \epsilon) + \epsilon)$

$= b^*(aa^*b^* + \epsilon)$

[∵ $bb^* + \epsilon = b^*$].

(5)



$A = \epsilon$ — (1)

$B = Aa + Ba$ — (2)

$C = Ab + Ca$ — (3)

eq. (2) $\Rightarrow$ $B = \epsilon a + Ba$

$B = a + Ba$

$R = Q + RP$

$B = aa^*$ — (4)

eq. (3) $\Rightarrow$ $C = Ab + Ca$

$= \epsilon b + Ca$

$C = b + Ca$

$R = Q + RR$

$C = ba^*$ — (5)

Final R.E. is $B + C$

$= aa^* + ba^*$ //

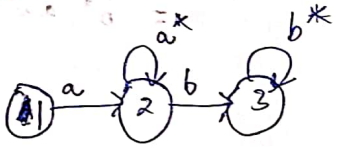
---

* Finite languages → is accepted by finite automata → if it is regular language.

* Infinite language → cannot be accepted if it is not regular.

Ex: $a^n b^m \;/\; n, m \geq 1$

ab
aabb
aab
abb



no memory required, because n, m are independent of each other.

Hence it is Regular language.

Ex: a^* is finite language because we can obtain a finite automata.

(1) $a^n b^n \;/\; n \leq 100$

$a^n b^n \;/\; n \geq 1$

2 → aabb

3 → aaabbb

Here no. of a's & b's are same and infinite.

So it is not regular language.

(2) $a^n b^m c^k \;/\; n, m, k \geq 1$

↓ after taking n a's, the b does not depend on n a's & same is for c. So no memory required.

∴ It is regular language.

(3) $ww^R \;/\; |w| = 2, \quad \Sigma = \{a, b\}$.

Given that length of w is 2. ⟹ → length is finite.

if $w = aa \rightarrow aaaa$

if $w = ab \rightarrow abba$

if $w = bb \rightarrow bbbb$

if $w = ba \rightarrow baab$

(w^R is reverse of w)

∴ It is regular language.

(4) $ww \;/\; w \in (a,b)^*$

if $w = abb \rightarrow abbabb$

if $w = aa \rightarrow aaaa$

∴ the length can be infinite.

So it is not regular language.

(5) $a^i b^{2j} \mid i, j \geq 1$

i	j	
1	1	→ abb
1	2	→ abbbb

This is also ~~independent~~ infinite, but the no of b's do not depend on a.

∴ The given language is regular language.

(6) $a^n \mid n$ is even

n=0 → a^0
n=2 → aa
n=4 → aaaa

$\{a^0, a^2, a^4, \ldots\}$

∴ Regular.

(∵ we have finite automata & also we have a pattern)

(7) $a^n \mid n$ is odd

n=1 → a
n=3 → aaa
n=5 → aaaaa

$\{a, a^3, a^5, a^7, \ldots\}$.

∴ Regular.

(8) $a^n \mid n$ is prime

$\{a^2, a^3, a^5, a^7, a^{11}, a^{13}, \ldots\}$

No pattern.

So, not Regular language.

(9) $a^{n^2} \mid n \geq 1$

$\{a^1, a^4, a^9, a^{16}, \ldots\}$

No pattern.

So, Not R.L.

(10) $a^{2^n} \mid n \geq 1$

$\{a^2, a^4, a^8, a^{16}, \ldots\}$

we can't construct any finite automata using this pattern.

∴ It is not R.L.

(11) $a^n b^{n+m} c^m \mid n, m \geq 1$

The no of b's are dependent on no of a's and no of c's.

And it can be infinite and can't be stored in memory.

∴ It is not R.L.

* (If there is dependence, then not R.L).

(12) $a^n b^{n+m} \mid n, m <= 100000$

Here n, m are finite. So finite memory is only required.

∴ The given lang is R.L.

Pumping Lemma :-

If L is an infinite lang then there exists some +ve integer 'n' such that any string $w \in L$ has length greater than equal to 'n' i.e.;

$|w| >= n$ then we can divide into 3 parts $w = xyz$ satisfy the foll conditions

(i) for each $i >= 0$, $xy^i z \in L$.

(ii) $|y| > 0$

(iii) $|xy| <= n$.

If a given lang passes the Pumping Lemma conditions then we can't decide (R.L / N.R.L) but if the conditions fail then lang is not R.L.

(1) Prove the lang. $a^n b^{2n} \mid n \geq 0$ by using P.L.

for $n=3 \rightarrow$ aaabbbbbb, x | y | z, xyz (can divide in any manner such that conditions satisfy)

$xy^iz \rightarrow xy^2z$ (taking $i=2$)

↳ a aabbbbb aabbbbbb b

3 5 2 6

$\notin L$ [$\because a^n \neq b^{2n}$]

$\therefore$ Not R.L.

(2) $a^n b^n \mid n >= 1$

$\rightarrow a^n b^n$ $n=5$ (say). $|xy| \leq n$

aa aaa bbbbb

x y z $5 \leq 5$ ✓.

$xy^iz \rightarrow xy^2z$

aaa aa aa bbbbb

7 5 $\notin L$ [$a^n \neq b^n$].

$\therefore$ Not R.L.

(3) $a^n \mid n \geq 1$

$\rightarrow a^n$, $n=5$ (say)

aaa aa

x y z

$xy^iz \rightarrow xy^2z$ (taking $i=2$)

aaa aa aa $\in L$

7

P.L conditions passed. So cannot decide R.L or not R.L.

(4) $L = ww$, w belongs to $(a,b)^*$

if $w = aba \rightarrow$ ab aaba

x y z

$xy^iz \rightarrow xy^4z$ (taking $i=4$).

abbbbaaba $\notin L$.

$\therefore$ Not R.L.

Closure Properties of Regular Sets :- (L & M are 2 R.L)

1. Union of two regular languages is regular. ($L \cup M$)
2. Intersection of two regular languages is regular ($L \cap M$)
3. Complement of a language is regular. ($\Sigma^* - L$).
4. Difference of two regular languages is regular. ($L - M$)
5. Reversal of a R.L is regular.
   ex: if $L = \{001, 110\}$, then $L(R) = \{100, 011\}$
6. Closure of a R.L is regular (L^*).
7. Concatenation of R.L's is regular. (LM).
8. Homomorphism of a R.L is regular.
   A homomorphism is the substitution of strings for symbols.

$$h(L) = \{ h(w) \mid w \in L \}$$

$h: \Sigma$ to Y^*

ex: $\Sigma = \{0,1\}$, $Y = \{a,b\}$

Let $h(0) = aa$

$h(1) = bb$

If $L = \{00, 11\}$

then $h(L) = \{aaaa, bbbb\}$.

Inverse Homomorphism :-

where $\Sigma = \{0, 1\}$
$\Psi = \{a, b\}$.

$h(0) = a$
$h(1) = b$
$h(2) = ab$
$h(L) = \{abab\}$

$\Rightarrow h^{-1}(L) = \{\frac{0}{a}\frac{1}{b}\frac{0}{a}\frac{1}{b}, \frac{2}{ab}\frac{2}{ab}, \frac{0}{a}\frac{1}{b}\frac{2}{ab}, \frac{2}{ab}\frac{0}{a}\frac{1}{b}\}$

So, $h(h^{-1}(L)) = \{abab, abab, abab, abab\}$
$= \{abab\}$.

↑ Q. If the lang. is regular, then inverse of homomorphism is also regular.

Applications of R.L :-

1. Lexical Analysis.
2. Finding patterns.

Unit - 3

RG ↓ R.L ↓ FA

A grammar is regular if it has rules of form $A \to a$ (or) $A \to Ba$ (or) $A \to \epsilon$

16/6/21

Unit - 3

Grammar :-

Grammar is a standard way of representing lang.

$G = \{V, T, P, S\}$ → 4 tuples.

V - Variables (upper case)
T - terminals (lower case)
P - Production rule
S - Start symbol

$S \to aSb \mid \epsilon$ → production.

a, b : terminals , S : variable

Consider $S \to aSb$ (S on replacing with aSb)
$aaSbb \to aaaSbbb$
if $S = \epsilon$ then $aabb$ $aaabbb$

Hence the lang. generated by given grammar is $a^n b^n$.

1. $S \to SS$
   $S \to aSb$
   $S \to bSa$
   $S \to \epsilon$
   $\Rightarrow S \to SS \mid aSb \mid bSa \mid \epsilon$

Sol^n: $S \to SS$
$aSb\,aSb \to abab$
$abSab\,abSab \to abab\,abab$

$S \to SS$
$aSb\,bSa \to abba$
$S \to SS$
$bSa\,bSa \to baba$

$S \to SS$
$aSb\epsilon \to ab$

∴ The lang. is $(ab + ba)^*$

Regular Grammar (RG) :-

Left Linear Grammar (LLG)

$A \to a$

$A \to Ba$

$A \to a \mid Ba$

Right Linear Grammar (RLG)

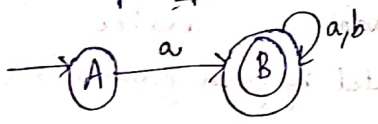
$A \to a$

$A \to aB$

$A \to a \mid aB$

ex: $S \to aBb$
= is not a R.G because it is neither LLG nor RLG.

(1) Convert given finite automata to R.G.

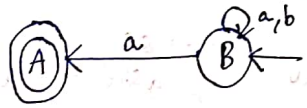


$A \to aB \mid a$ → we add | input because B is final state.

$B \to aB \mid bB \mid a \mid b$

Now convert this into LLG.

Step 1: Reverse the finite automata. (i.e initial & final states are exchanged & arrow directions are also reversed).



$B \to aA \mid a \mid bB \mid aB$

$A \to \varepsilon$

Step 2: Now just change as follows.

$B \to Aa \mid a \mid Bb \mid Ba$

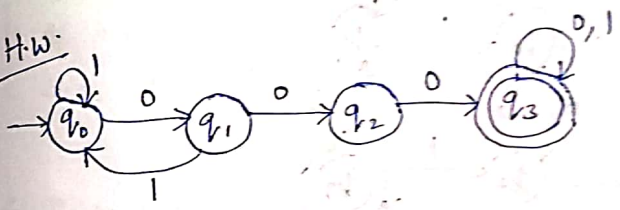
$A \to \varepsilon$

Hence this is the LLG.

Rules to convert finite automata to R.G :-

* If there is a transition of the form $\delta(q, a) \to p$, then add as production "$q \to ap$" if p is non-final state.
* If there is a transition of the form $\delta(q, a) \to p$, then add the production "$q \to ap \mid a$" if p is final state.

H.W.



$q_0 \to 1q_0 \mid 0q_1$

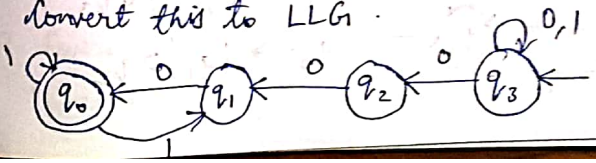
$q_1 \to 1q_0 \mid 0q_2$

$q_2 \to 0q_3 \mid 0$

$q_3 \to 0q_3 \mid 0 \mid q_3 \mid 0 \mid 1$

} RLG.

Convert this to LLG.



$q_3 \rightarrow 0q_3 \mid 1q_3 \mid 0q_2$

$q_2 \rightarrow 0q_1$

$q_1 \rightarrow 0q_0 \mid 0$

$q_0 \rightarrow 1q_0 \mid 1 \mid 1q_1$

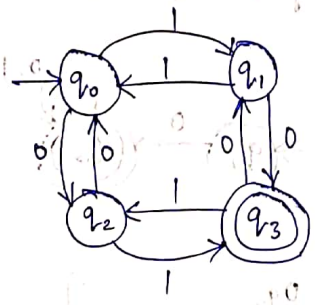
Now interchange, $q_3 \rightarrow q_3 0 \mid q_3 1 \mid q_2 0$

$q_2 \rightarrow q_1 0$

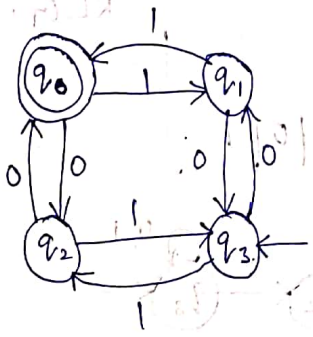
$q_1 \rightarrow q_0 0 \mid 0$

$q_0 \rightarrow q_0 1 \mid 1 \mid q_1 1$

Q1. Find LLG for the given DFA.



To convert into LLG, reverse the DFA.



$q_3 \rightarrow 0q_1 \mid 1q_2$

$q_2 \rightarrow 0q_0 \mid 0 \mid 1q_3$

$q_1 \rightarrow 1q_0 \mid 1 \mid 0q_3$

$q_0 \rightarrow 1q_1 \mid 0q_2$

LLG $\Rightarrow$ $q_3 \rightarrow q_1 0 \mid q_2 1$

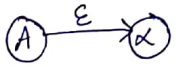
$q_2 \rightarrow q_0 0 \mid 0 \mid q_3 1$

$q_1 \rightarrow q_0 1 \mid 1 \mid q_3 0$

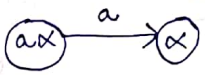
$q_0 \rightarrow q_1 1 \mid q_2 0$

Conversion of R.G to Finite Automata :-

1. First convert given LLG into RLG.
2. Take all possible suffixes of r.h.s of rule as states of FA.
   ex: In $S \rightarrow 01S$, possible suffixes are $[01S]$, $[1S]$, $[S]$, $[\varepsilon]$.
3. Draw FA with start state as initial state and $[\varepsilon]$ as final state.
4. If A is a non-terminal in the grammar and $A \rightarrow \alpha$ is a production, then add the transition.



5. If a is a terminal and α is some possible suffix of the form T^* or T^*V, then add a transition by $\delta([a\alpha], a) = [\alpha]$.



6. This NFA corresponds to RLG. If given NFA is LLG then follow 7 & 8 steps, otherwise stop.
7. Reverse the NFA M' to get M such that,
   - Initial state of M' is made final state of M.

- Final state of M' is initial state of M.
- Reverse all edges of M' and add to M.

8. Obtained NFA M corresponds to given LLG.

(1) $S \longrightarrow aA \mid b$

$A \longrightarrow aA \mid a$

Sol^n: This is R.L.G.

Suffixes are, $[aA]$, $[A]$, $[b]$, $[a]$.

Rule 1 (R1):

$\delta([S], \epsilon) = [aA], [b]$

$\delta([A], \epsilon) = [aA], [a]$

Rule 2 (R2):

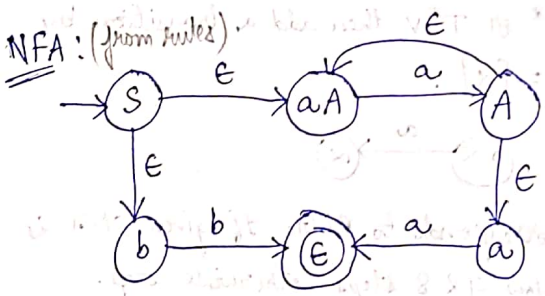
$\delta([aA], a) = [A]$

$\delta([a], a) = [\epsilon]$

$\delta([b], b) = [\epsilon]$

} from productions given in q^s

NFA: (from rules)



(2) $S \longrightarrow Ab \mid ab$

$A \longrightarrow Ab \mid Bb$

$B \longrightarrow Ba \mid a$

Sol^n: Since this is LLG, convert to R.L.G.

$S \longrightarrow bA \mid ba$

$A \longrightarrow bA \mid bB$

$B \longrightarrow aB \mid a$

Suffixes are, $[bA]$, $[A]$, $[ba]$, $[a]$, $[bB]$, $[B]$, $[aB]$

R1:

$\delta([S], \epsilon) = [bA], [ba]$

$\delta([A], \epsilon) = [bA], [bB]$

$\delta([B], \epsilon) = [aB], [a]$

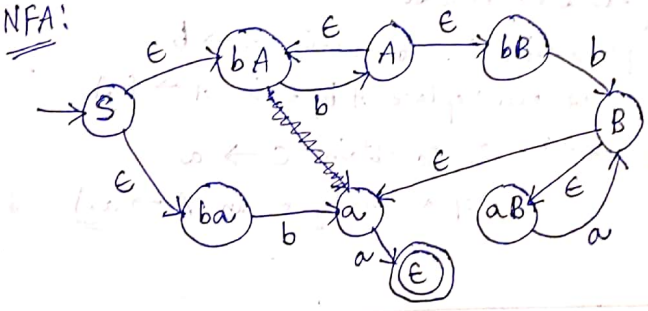
R2:

$\delta([bA], b) = [A]$, $\delta([ba], b) = [a]$

$\delta([a], a) = \epsilon$, $\delta([bB], b) = [B]$.

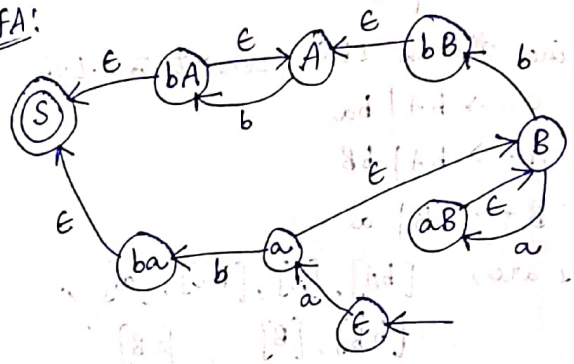
$\delta([aB], a) = [B]$.

NFA:



The obtained NFA is for R-L-G.
But in qs. it is LLG. So reverse the NFA.

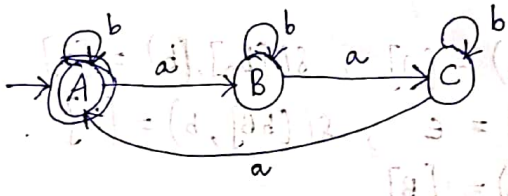
∴ NFA:



① Draw the FA for the foll:

A → aB | bA | b
B → aC | bB
C → aA | bC | a

} RLG



A is final state because,
in production 1, to obtain A → b
we can replace A by ε in A → bA
in production 3, to obtain C → a
we can replace A by ε in A → aA

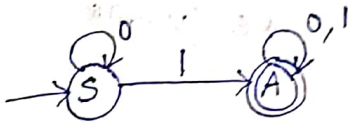
If it is LLG.

LLG → RG → FA (Reverse) → F ✓

② Draw the FA for the foll:

S → 0S | 1A | 1
A → 0A | 1A | 0 | 1

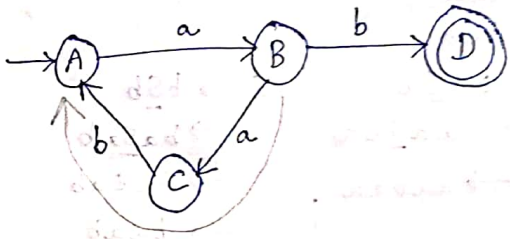
} RLG.



A is final state, to obtain S → 1, take S → 1A, A = ε
to obtain A → 0, take A → 0A, A = ε
to obtain A → 1, take A → 1A, A = ε

③ A → aB
B → abA | b

} RLG.



C is taken for B → abA.
D is taken because, A → aB
→ ab [∵ B → b].
Hence ab is the min. string to be accepted.

Conversion of LLG to RLG:-

LLG → R → (Reverse) FA → F → g (obtain the grammar)

R.G

Context-free Grammar:-

A grammar is context free if for (V,T,P,S) tuples,

$$A \rightarrow x$$

where $A \in V$, $x \in (V \cup T)^*$

(1) $S \rightarrow aS$
$S \rightarrow \epsilon$

Soln:

$S \rightarrow aS$ | $S \rightarrow aS$
$\rightarrow a\epsilon$ | $\rightarrow aaS$
$\rightarrow a$ | $\rightarrow aa\epsilon$
| $\rightarrow aa$

Hence lang. generated is a^*.

(2) $S \rightarrow \epsilon | a | b$
$S \rightarrow aSa$
$S \rightarrow bSb$

Soln:

$S \rightarrow aSa$ | $S \rightarrow bSb$
$\rightarrow aaSaa$ | $\rightarrow baSab$
$\rightarrow aabaa$ | $\rightarrow ba\epsilon ab$
| $\rightarrow baab$

$S \rightarrow aSa$
$\rightarrow aba$

Hence the lang. generated by this grammar is Word and word reverse i.e. Palindrome.

(3) $S \rightarrow SaSbS | SbSaS | \epsilon$

Soln:

$S \rightarrow SaSbS$ | $S \rightarrow SaSbS$
$\rightarrow \epsilon a \epsilon b \epsilon$ | $\rightarrow SaSbSaSbS$
$\rightarrow ab$ | $bSaSbS$
| $\rightarrow abaabbab$

$S \rightarrow SbSaS$ | $S \rightarrow SbSaS$
$\rightarrow \epsilon b \epsilon a \epsilon$ | $\rightarrow SbSaSbSbSaS$
$\rightarrow ba$ | $aSbSaS$
| $\rightarrow babbaaba$

Hence the lang. generated by this grammar is no. of a's = no. of b's.

(4) $S \rightarrow aS$
$S \rightarrow bS$
$S \rightarrow a$
$S \rightarrow b$

Soln:

$S \rightarrow aS$ | $S \rightarrow aS$ | $S \rightarrow aS$
$\rightarrow aa$ | $\rightarrow ab$ | $\rightarrow aaS$
| | $\rightarrow aaa$

$S \rightarrow bS$ | $S \rightarrow bS$ | $S \rightarrow bS$
$\rightarrow bb$ | $\rightarrow ba$ | $\rightarrow bbS$
| | $\rightarrow bbb$

Hence the lang is $(a+b)^+$

(1) Generate grammar (context-free) for the R.E $(011+1)^*(01)^*$

Soln: First consider $(011+1)^*$ → C

$A \to CA \mid \epsilon$

$C \to 011 \mid 1$

Now consider $(01)^*$ → B

$B \to DB \mid \epsilon$

$D \to 01$

∴ Finally,

$S \to AB$

$A \to CA \mid \epsilon$

$C \to 011 \mid 1$

$B \to DB \mid \epsilon$

$D \to 01$

check by constructing strings,

$S \to AB$

$\to CAB$

$011 \epsilon\epsilon$

$\to 011$ ✓

(2) Construct the context-free grammar for which the 1st & last symbol differ where the inputs are $\{0,1\}$.

sol^n: The R-E is,

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

$S \to 0A1 \mid 1A0$

$A \to 0A \mid 1A \mid \epsilon$ [we can't write $0 \mid 1$, because we have $*$].

ex: $S \to 0A1$

$\to 0\epsilon1$

$\to 01$ ✓

(3) Write the context-free grammar for $a^n b^{2n} \mid n \geq 1$.

sol^n: $a^n b^{2n} = a^n (bb)^n$

$S \to aSbb \mid abb$

[∵ abb, aabbbb, aaabbbbbb, ... are acceptable].

(4) $a^m b^n \mid m, n \geq 1$

sol^n: ~~$S \to aSb \mid a \mid b \mid ab \mid \epsilon$~~

$S \to aAbB$

$A \to aA \mid \epsilon$

$B \to bB \mid \epsilon$

Derivations of Context-free Grammar:-

It is a process of defining the string out of grammar by application of rules starting from the start symbol.

ex: (1) $S \to aS$, $S \to \epsilon$ } Derive a^4 from given grammar

sol^n: $S \to aS$

$\to aaS$

$\to aaaS$

$\to aaaa\epsilon$ ✓ we will get a^4.

(2) $S \to aS \mid bS \mid a \mid b$

$S \to aS$

$S \to bS$

$S \to a$

$S \to b$

sol^n: $S \to aS$

$\rightarrow$ abS
$\rightarrow$ abbS
$\rightarrow$ abbaS
$\rightarrow$ abbab

Left most derivation :-

The left most non-terminal in a string is the first non-terminal that we encounter when we scan from left to right.

(or)

Replacing the left most variable first at every step gives the left most derivation.

ex: S $\rightarrow$ baXaS | ab
X $\rightarrow$ Xab | aa

Derive baaaabaab using left most derivation.

sol^n: S $\rightarrow$ baXaS
$\Rightarrow$ baXabaS
$\Rightarrow$ baaaabaS
$\Rightarrow$ baaaabaab ✓

Right most derivation :-

Replacing the right most variable first at every step gives the RMD.

ex: S $\rightarrow$ baXaS | ab
X $\rightarrow$ Xab | aa

Derive baaaabaab using RMD.

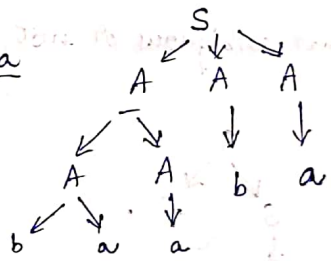
sol^n: S $\rightarrow$ baXaS
$\Rightarrow$ baXaab
$\Rightarrow$ baaaabaab ✓

Derivation Tree :-

Process shown pictorially as a tree to illustrate how a word derived from CFG. It is also called syntax trees / parse trees.

ex: S $\rightarrow$ AAA | AA
A $\rightarrow$ AA | aA | Ab | a | b
} Derive baaba

sol^n:
baaba



$\rightarrow$ (This is LMD)

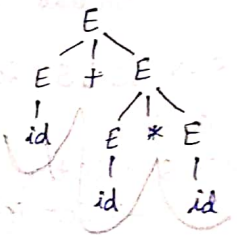
Ambiguous Grammar :-

If there exists more than one path stream or there exists more than one LMD or more than one RMD then the grammar is said to be ambiguous.

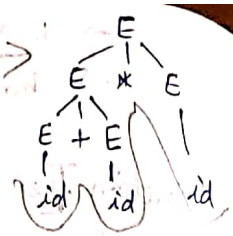
ex: ① E $\rightarrow$ id | E+E | E*E | E-E

LMD : (Lets derive id+id*id)

E $\Rightarrow$ E+E
$\Rightarrow$ id+E
$\Rightarrow$ id+E*E
$\Rightarrow$ id+id*E
$\Rightarrow$ id+id*id



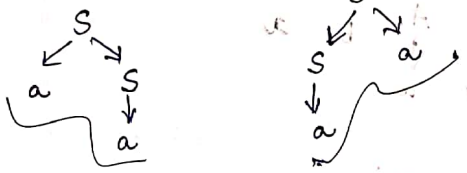
Now consider $E \Rightarrow E*E$
$\Rightarrow E+E*E$
$\Rightarrow id+E*E$
$\Rightarrow id+id*E$
$\Rightarrow id+id*id$



For 2 strings we got same productions. So the given grammar is ambiguous.
(Can prove using derivations or derivation tree).

(2) $S \rightarrow aS \mid Sa \mid a$
Is the given grammar ambiguous or not?

Sol^n: Let's derive aa.



Hence grammar is ambiguous.

(3) $S \rightarrow aSa \mid bSb \mid a \mid b \mid \epsilon$ Ambiguous or not?

Sol^n: Consider bab
↳ can be derived in only 1 way
i.e $S \rightarrow bSb$
$\rightarrow bab$ ✓ So, it is not ambiguous.

(4) $S \rightarrow iCtS \mid ictSeS \mid a$
$C \rightarrow b$

Sol^n: $S \rightarrow iCtS$
$\rightarrow ictictSeS$
$\rightarrow ibtibtaea$ ✓

Now, $S \rightarrow iCtSeS$
$\rightarrow iCtiCtSea$
$\rightarrow ibtibtaea$ ✓

2 strings → same productions. So the grammar is ambiguous.

Simplification / minimisation of grammar:-

3 steps:-
* Elimination of useless symbols.
* Elimination of ϵ production
* Elimination of unit productions (like $A \rightarrow B$, both A, B non-terminals).

Elimination of useless symbols:-

A symbol is useless if it cannot derive a terminal or is not reachable from start symbol.

ex: (1) $S \rightarrow ABa \mid BC$
$A \rightarrow aC \mid BCC$
$C \rightarrow a$
$B \rightarrow bcc$
$D \rightarrow E$
$E \rightarrow d$
$F \rightarrow e$

} Eliminate useless symbols.

Sol^n: S — A — B — C

D E F (×)

So finally $S \rightarrow ABa \mid BC$
$A \rightarrow aC \mid BCC$
$C \rightarrow a$
$B \rightarrow bcc$

} after removing useless symbols.

(2) $S \rightarrow AB \mid CA$
$S \rightarrow BC \mid AB$
$A \rightarrow a$

$C \rightarrow aB \mid b$

Sol^n: There is no production from B. So it is useless.

So after removing productions containing B,

$S \rightarrow CA$
$A \rightarrow a$
$C \rightarrow b$

③ $S \rightarrow aS \mid A \mid BC$
$A \rightarrow a$
$B \rightarrow aa$
$C \rightarrow aCb$

Sol^n:

Finally, $S \rightarrow aS \mid A$
$A \rightarrow a$

Elimination of ∈ production :-

① $S \rightarrow ABaC$
$A \rightarrow BC$
$B \rightarrow b \mid \epsilon$
$C \rightarrow D \mid \epsilon$
$D \rightarrow d$

1. Nullable variables. {B, C, A}.

$S \rightarrow ABaC \mid BaC \mid AaC \mid ABa \mid aC \mid Aa \mid Ba \mid a$

(putting A, B, C as ∈ — at a time 2 ∈ — derived from previous ones)

$A \rightarrow BC \mid C \mid B$
$B \rightarrow b$
$C \rightarrow D$
$D \rightarrow d$

② $S \rightarrow aA$
$A \rightarrow BB$
$B \rightarrow aBb \mid \epsilon$

Sol^n: Nullable variables {B, A}

$S \rightarrow aA \mid a$
$A \rightarrow BB \mid B$
$B \rightarrow aBb \mid ab$

Elimination of unit production :-

① $S \rightarrow A \mid bb$
$A \rightarrow B \mid b$
$B \rightarrow S \mid a$

Sol^n: $S \rightarrow a \mid b \mid bb$

$A \rightarrow a \mid b \mid bb$

$B \rightarrow a \mid b \mid bb$.

② $S \rightarrow Aa \mid B$
$B \rightarrow A \mid bb$
$A \rightarrow a \mid bc \mid B$

Sol^n: $S \rightarrow Aa \mid bb \mid a \mid bc$
$B \rightarrow bb \mid a \mid bc$
$A \rightarrow a \mid bc \mid bb$

③ $S \rightarrow aA \mid aBB$
$A \rightarrow aAA \mid \epsilon$
$B \rightarrow bB \mid bbC$
$C \rightarrow B$

Sol^n: Eliminating null productions

$S \rightarrow aA \mid a \mid aBB$
$A \rightarrow aAA \mid aA \mid a$
$B \rightarrow bB \mid bbC$
$C \rightarrow B$.

3 sol^n: (cont).

eliminate useless symbols.

(B, C because they don't derive anything).

$S \rightarrow aA \mid a$
$A \rightarrow aAA \mid aA \mid a$ ✓

↳ No unit productions involved. So this is the final grammar.

Elimination of Left Recursion:-

If production is of form $A \rightarrow A\alpha \mid \beta$

then add

$A \rightarrow \beta A'$
$A' \rightarrow \alpha A' \mid \epsilon$

ex: ① $E \rightarrow E+T \mid T$

(A: E; α: $+T$; β: T)

$A \rightarrow E$
$\alpha \rightarrow +T$
$\beta \rightarrow T$

$\therefore E \rightarrow TE'$
$E' \rightarrow +TE' \mid \epsilon$

② $S \rightarrow S0S1S \mid 01$

$\alpha = 0S1S$
$\beta = 01$

$\therefore S \rightarrow 01S'$
$S' \rightarrow 0S1SS' \mid \epsilon$

③ $S \rightarrow (L) \mid x$
$L \rightarrow L,S \mid S$

$\therefore L \rightarrow SL'$
$L' \rightarrow ,SL' \mid \epsilon$
$\rightarrow S \rightarrow (L) \mid x$ (No change bez no LR)

If production is of the form $A \rightarrow A\alpha \mid \beta_1 \mid \beta_2 \mid \ldots$

then add

$A \rightarrow \beta_1 A' \mid \beta_2 A'$
$A' \rightarrow \alpha A' \mid \epsilon$

If production is of the form,

$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots A\alpha_n \mid \beta$

then add

$A \rightarrow \beta A'$
$A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \ldots \alpha_n A' \mid \epsilon$

Elimination of Left Factoring:-

$S \rightarrow aA \mid aB$
$NDG \rightarrow DG$ (Non-deterministic grammar → deter. grammar)

ex: ① $S \rightarrow aS \mid aB \mid a \mid b$

If $A \rightarrow \alpha\beta_1 \mid \alpha\beta_2 \mid \alpha\beta_3 \mid \gamma$ ✓

add

$A \rightarrow \alpha A' \mid \gamma$
$A' \rightarrow \beta_1 \mid \beta_2 \mid \beta_3$

$S \rightarrow aS' \mid a \mid b$
$S' \rightarrow S \mid \beta$

② $A \rightarrow aSb \mid aSc \mid b$

sol^n: $A \rightarrow aSA' \mid b$

$A' \rightarrow$ ~~as~~ $b \mid c$

Normal forms:-

1. Chomsky Normal Form :-(CNF) :-

A CFG is in in CNF if each of ~~its~~ the production has one of the 2 forms.

① Should contain 2 variables on right side.

ex: $A \rightarrow BC$.

② Should contain 1 terminal.

ex: $A \rightarrow a$

Ex: ① Convert the foll: grammar to CNF.

$S \rightarrow bA \mid aB$

$A \rightarrow bAA \mid aS \mid a$

$B \rightarrow aBB \mid bS \mid b$

sol^n: (i) replacing every terminal by variable.

$S \rightarrow C_bA \mid C_aB$

$A \rightarrow C_bAA \mid C_aS \mid a$ → Should not replace bcz it is in CNF.

$B \rightarrow C_aBB \mid C_bS \mid b$ →

$C_a \rightarrow a$

$C_b \rightarrow b$

Now converting those which are not in CNF.

$S \rightarrow C_bA \mid C_aB$ (No change bcz it is in CNF)

$A \rightarrow C_bD \mid C_aS \mid a$

$B \rightarrow C_aE \mid C_bS \mid b$

$C_a \rightarrow a$

$C_b \rightarrow b$

$D \rightarrow AA$

$E \rightarrow BB$

② $S \rightarrow AB \mid aB$
$A \rightarrow aab \mid \epsilon$
$B \rightarrow bbA$
} Convert this GFG to CNF.

sol^n: Removing ϵ production,

$S \rightarrow AB \mid B \mid aB$

$A \rightarrow aab$

$B \rightarrow bbA \mid bb$

Removing unit productions,

$S \rightarrow AB \mid bbA \mid bb \mid aB$

$A \rightarrow aab$

$B \rightarrow bbA \mid bb$

$S \rightarrow AB \mid C_bC_bA \mid C_bC_b \mid C_aB$

$A \rightarrow C_aC_aC_b$

$B \rightarrow C_bC_bA \mid C_bC_b$

$C_a \rightarrow a$

$C_b \rightarrow b$

$S \rightarrow AB \mid C_{bb}A \mid C_bC_b \mid C_aB$

$A \rightarrow C_{aa}C_b$

$B \rightarrow C_{bb}A \mid C_bC_b$

$C_a \rightarrow a$

$C_b \rightarrow b$

$C_{aa} \rightarrow C_aC_a$

$C_{bb} \rightarrow C_bC_b$

(3) $S' \rightarrow S$

$S \rightarrow ASB \mid \epsilon$

$A \rightarrow aAS \mid a$

$B \rightarrow SbS \mid A \mid bb$

Sol": Removing ϵ productions,

$S \rightarrow ASB \mid AB$

$A \rightarrow aAS \mid aA \mid a$

$B \rightarrow SbS \mid bS \mid Sb \mid b \mid A \mid bb$

Removing unit productions,

$S \rightarrow ASB \mid AB$

$A \rightarrow aAS \mid aA \mid a$

$B \rightarrow SbS \mid bS \mid Sb \mid b \mid aAS \mid aA \mid a \mid bb$

$S \rightarrow DB \mid AB$

$A \rightarrow C_aD \mid C_aA \mid a$

$B \rightarrow SC_bS \mid C_bS \mid SC_b \mid b \mid C_aD \mid C_aA \mid a \mid C_bC_b$

$C_a \rightarrow a$

$C_b \rightarrow b$

$D \rightarrow AS$

$S \rightarrow DB \mid AB$

$A \rightarrow C_aD \mid C_aA \mid a$

$B \rightarrow ES \mid C_bS \mid SC_b \mid b \mid C_aD \mid C_aA \mid a \mid C_bC_b$

$C_a \rightarrow a$

$C_b \rightarrow b$

$D \rightarrow AS$

$E \rightarrow SC_b$

Procedure to convert CFG to CNF :-

1. Eliminating null and unit productions.
2. Include the productions of the form $A \rightarrow BC \mid a$ as it is.
3. Eliminate string of terminals on RHS of production if it exists / agrees one of the following.

   $S \rightarrow a_1a_2a_3$ ($C_1 \rightarrow a_1$, $C_2 \rightarrow a_2$, $C_3 \rightarrow a_3$)

   Introduce non-terminals for every terminal.

4. To restrict the no. of variables on the RHS, if the production is $Y \rightarrow X_1X_2X_3X_4 \cdots$

   $Y \rightarrow X_1R$ ($R = X_2X_3X_4 \cdots$)

   $R \rightarrow X_2R_1$ ($R_1 = X_3X_4 \cdots$)

   $R_1 \rightarrow X_3X_4$.

We need to include new variables by combining two-two variables.

Can write in any way but RHS should contain 2 variables

Greibach normal form :- (GNF) :-

A CFG is said to be in GNF if all the productions are in the form of

$A \rightarrow aX$

where $a \in T$ (terminal), $X \in V^*$ (any no. of variables).

ex: $A \rightarrow a$ ✓
$A \rightarrow aB$ ✓

Procedure to convert CFG to GNF :-

1. Elimate null and unit productions and construct CNF.
2. Rename the variables (like $A_1, A_2, A_3, \ldots$).
3. For production $A_i \rightarrow A_j \alpha$:
   (i) $j > i$ → leave as it is.
   (ii) $j = i$ → apply left recursion.
   (iii) $j < i$ → substitution rule.

ex: $S \rightarrow XB$ ($S = A_1$, $X = A_2$, $B = A_3$)
$B \rightarrow A \mid a$ ($A_4 = A$)

Q1:

$S \rightarrow XA \mid BB$
$B \rightarrow b \mid SB$
$X \rightarrow b$
$A \rightarrow a$

Sol^n: No null and unit productions.
The grammar is already in CNF.

Now, Rename the variables,

$j > i$ $A_1 \rightarrow A_2 A_3 \mid A_4 A_4$ — ①
$j < i$ $A_4 \rightarrow b \mid A_1 A_4$ — ②
$A_2 \rightarrow b$ GNF — ③
$A_3 \rightarrow a$ — ④

Replaced
S with A_1
X with A_2
A with A_3
B with A_4

Consider $A_4 \rightarrow A_1 A_4 \mid b$ $j < i$
So apply substitution.

↳ $A_4 \rightarrow A_2 A_3 A_4 \mid A_4 A_4 A_4 \mid b$ — ⑤ [∵ from ①]
↳ $A_4 \rightarrow bA_3 A_4 \mid A_4 A_4 A_4 \mid b$ — ⑥
GNF by subs. by L.R we need to solve
↳ $A_4 \rightarrow A_4 A_4 A_4 \mid bA_3 A_4 \mid b$

[$A \rightarrow A\alpha \mid \beta_1 \mid \beta_2 \Rightarrow A \rightarrow \beta_1 A' \mid \beta_2 A'$
$A' \rightarrow \alpha A' \mid \epsilon$].

$A_4 \rightarrow bA_3 A_4 Z \mid bZ$
$Z \rightarrow A_4 A_4 Z \mid \epsilon$
Remove ϵ,
$\Rightarrow A_4 \rightarrow bA_3 A_4 Z \mid bA_3 A_4 \mid bZ \mid b$
$Z \rightarrow A_4 A_4 Z \mid A_4 A_4$

∴ $A_1 \rightarrow A_2 A_3 \mid A_4 A_4$ — not in GNF — 1'
$A_4 \rightarrow bA_3 A_4 Z \mid bA_3 A_4 \mid bZ \mid b$ ✓ GNF — 2'
$Z \rightarrow A_4 A_4 Z \mid A_4 A_4$ — not in GNF — 3'
$A_2 \rightarrow b$ ✓ GNF — 4'
$A_3 \rightarrow a$ ✓ GNF — 5'

For $A_1 \rightarrow A_2 A_3 \mid A_4 A_4$

even though $j > i$ we use substitution because it is not in GNF.

$A_1 \rightarrow bA_3 \mid bA_3A_4ZA_4 \mid bA_3A_4A_4 \mid bZA_4 \mid bA_4$

[∵ from 1' and 2'].

Consider now, $Z \rightarrow A_4A_4Z \mid A_4A_4$

Now substitute,

$Z \rightarrow bA_3A_4ZA_4Z \mid bA_3A_4A_4Z \mid bZA_4Z \mid bA_4Z \mid bA_3A_4ZA_4 \mid bA_3A_4A_4 \mid bZA_4 \mid bA_4$.

∴ Final productions in GNF are:

$A_1 \rightarrow bA_3 \mid bA_3A_4ZA_4 \mid bA_3A_4A_4 \mid bZA_4 \mid bA_4$

$A_4 \rightarrow bA_3A_4Z \mid bA_3A_4 \mid bZ \mid b$

$Z \rightarrow bA_3A_4ZA_4Z \mid bA_3A_4A_4Z \mid bZA_4Z \mid bA_4Z \mid bA_3A_4ZA_4 \mid bA_3A_4A_4 \mid bZA_4 \mid bA_4$.

$A_2 \rightarrow b$

$A_3 \rightarrow a$

Q2. $S \rightarrow AA \mid a$

$A \rightarrow SS \mid b$

Sol^n: Grammar is in CNF.

So rename the variables:

$j > i$ $A_1 \rightarrow A_2A_2 \mid a$ — ① | Replaced S with A_1, A with A_2

$j < i$ $A_2 \rightarrow A_1A_1 \mid b$ — ② |

Consider $A_2 \rightarrow A_1A_1 \mid b$ $j < i$

So apply substitution.

$A_2 \rightarrow A_2A_2A_1 \mid aA_1 \mid b$

[$A \rightarrow A\alpha \mid \beta_1 \mid \beta_2 \Rightarrow A \rightarrow \beta_1A' \mid \beta_2A'$, $A' \rightarrow \alpha A' \mid \epsilon$]

$A_2 \rightarrow aA_1Z \mid bZ$

$Z \rightarrow A_2A_1Z \mid \epsilon$

Remove ϵ,

$A_2 \rightarrow aA_1Z \mid aA_1 \mid bZ \mid b$

$Z \rightarrow A_2A_1Z \mid A_2A_1$

∴ $A_1 \rightarrow A_2A_2 \mid a$ not in GNF — 1'

$A_2 \rightarrow aA_1Z \mid aA_1 \mid bZ \mid b$ ✓ GNF — 2'

$Z \rightarrow A_2A_1Z \mid A_2A_1$ not in GNF — 3'

Take $A_1 \rightarrow A_2A_2 \mid a$ Substitution

$A_1 \rightarrow aA_1ZA_2 \mid aA_1A_2 \mid bZA_2 \mid bA_2 \mid a$ ✓

Take $Z \rightarrow A_2A_1Z \mid A_2A_1$ Substitution

$Z \rightarrow aA_1ZA_1Z \mid aA_1A_1Z \mid bZA_1Z \mid bA_1Z \mid aA_1ZA_1 \mid aA_1A_1 \mid bZA_1 \mid bA_1$ ✓

∴ Final productions in GNF are,

$A_1 \rightarrow aA_1ZA_2 \mid aA_1A_2 \mid bZA_2 \mid bA_2 \mid a$

$A_2 \rightarrow aA_1Z \mid aA_1 \mid bZ \mid b$

$Z \rightarrow aA_1ZA_1Z \mid aA_1A_1Z \mid bZA_1Z \mid bA_1Z \mid baA_1ZA_1 \mid aA_1A_1 \mid bZA_1 \mid bA_1$

Pumping Lemma for CFG :-

If A is a CFG then A has pumping length 'p' such that the string is divided into 5 features/pieces.

$S = uvxyz$

such that,

(i) $uv^ixy^iz : \rightarrow \in A$

(ii) $|vy| > 0$

(iii) $|vxy| \leq p$

Q1. $S = a^nb^nc^n \mid n \geq 0$

Prove that given lang. is not context-free.

Soln: We prove by contradiction i.e. assuming that it is CFG.

Take $n=4$, $\underbrace{aaaa}_{u}\ \underbrace{bbbb}_{v}\ \underbrace{c}_{x}\ \underbrace{c}_{y}\ \underbrace{ccc}_{z}$ (can take in any manner).

Take $i=3$, uv^ixy^iz.

$\Rightarrow \underbrace{aaaa}_{4}\underbrace{bbbbbb}_{6}\underbrace{ccccccc}_{6} \notin A$.

P.L 1st condition failed.

∴ The given grammar is not CFG.

[Also we took $p=3$ such that (ii) (iii) are satisfied bcz $|vy| = 2 > 0$ ✓

$|vxy| = 3 = p = 3$ ✓].

Closure Properties of CFL (Context-free Languages) :-

1. CFL are closed under union.
2. CFL are closed under concatenation.
3. CFL are closed under closure.
4. They are not closed under intersection.
5. They are not closed under compliment.

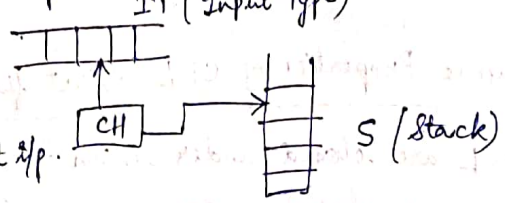
[Explanation same as given for properties of CFG].

12/7/21:

Unit-4

Push Down Automata (PDA):-

Since there are a few dis. adv. of F.A we go for PDA.



There are 7 tuples.

$M = (Q, \Sigma, \delta, \Upsilon, q_0, Z_0, F)$.

F.A dis-adv. → very finite amount of memory.

$a^n b^n \mid n>0$ → cannot construct finite automata.

PDA is a F.A with extra memory called stack.

→ $M = (Q, \Sigma, \delta, \Upsilon, q_0, Z_0, F)$.

- Q – no. of states.
- Σ – input alphabet.
- δ – transition function.
- Υ – stack alphabet.
- q_0 – initial state.
- I – initial stack symbol.
- F – set of accepting states.

PDA moves:-

(i) Element to be added to the stack.

(ii) Element may be deleted from the stack.

$(q_0, a, Z_0) = (q_0, \epsilon)$ → deleted.

(iii) They may (or) may not change their state.

ex: 1. $(q_0, a, Z_0) = (q_0, aZ_0)$

i/p symbol, initial stack symbol

2. $(q_0, a, Z_0) = (q_1, aZ_0)$

Graphical Representation of PDA:-

$\delta(q, a, \epsilon) = (q, a)$

$\delta(q, b, \epsilon) = (q, b)$

$\delta(q, c, \epsilon) = (p, \epsilon)$

$\delta(p, a, a) = (p, \epsilon)$

$\delta(p, b, b) = (p, \epsilon)$.

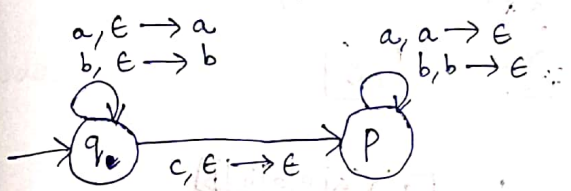
$\Rightarrow$, $Q = \{q, p\}$

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

$\Upsilon = (a, b)$

$q_0 = \{q\}$

$F = \{p\}$



Language Acceptance by PDA:-

(1) When PDA reaches final state.
(2) When PDA reaches empty stack (i.e if stack is empty).

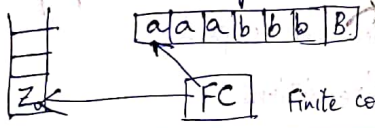
} conditions for acceptance (any 1 may be true).

ex: 1. Design a PDA which accepts $a^n b^n \mid n \geq 1$.

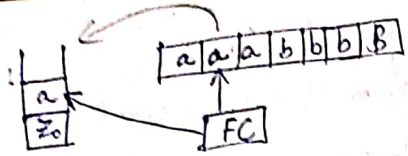
sol: $L = \{a^n b^n \mid n \geq 1\}$.

Let's consider the string aaabbb

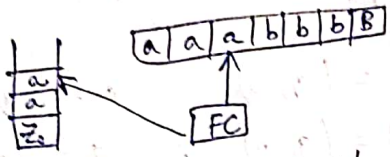
Step 1:



Step 2:

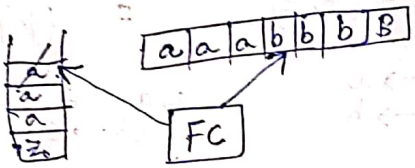


Step 3:

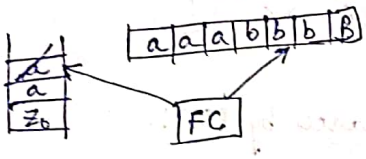


For every input 'a' we added 'a' into stack.
Now for every input 'b' pop 'a' from stack.

Step 4:



Step 5:



Step 6:

Step 7:

Reached final state & stack is empty. So stop.
Hence PDA accepted.

Now, write transitions:

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

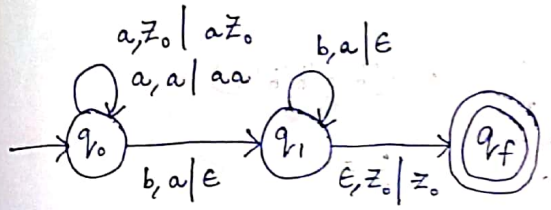
i/p $\delta(q_0, a, a) = (q_0, aa)$ — Top of stack

$\delta(q_0, b, a) = (q_1, \epsilon)$

$\delta(q_1, b, a) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, Z_0) = (q_f, Z_0)$.

∴

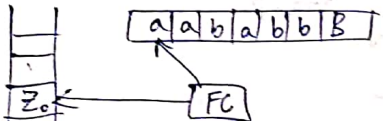


Consider aabb ✓.
abb ✗
ba ✗

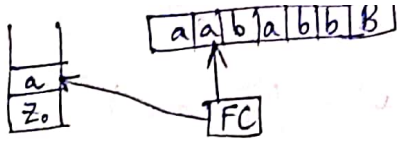
H.W:
① Design a PDA which accepts $a == b$ (i.e. no. of a's equal to no. of b's).
② Design a PDA which accepts $0^n 1^{2n} \mid n \geq 1$.

① Sol^n: Let's consider the string aababb.
First if 'a' or 'b' comes then push it into stack.
Next if same symbol comes then add to stack.
otherwise pop from stack.

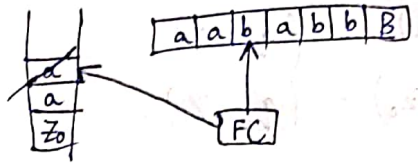
Step 1:



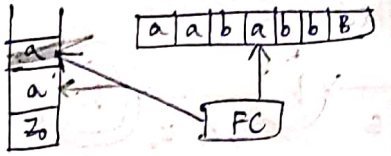
Step 2:



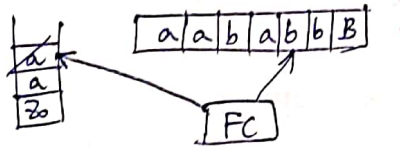
Step 3:



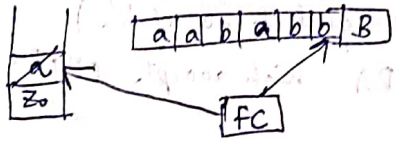
Step 4:



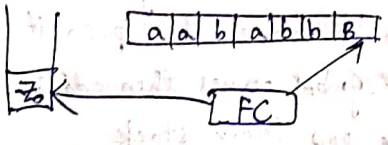
Step 5:



Step 6:



Step 7:



Empty Stack.

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, Z_0) = (q_0, bZ_0)$

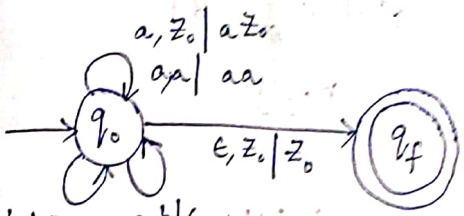
$\delta(q_0, b, b) = (q_0, bb)$

$\delta(q_0, a, b) = (q_0, \epsilon)$

$\delta(q_0, b, a) = (q_0, \epsilon)$

$\delta(q_0, \epsilon, Z_0) = (q_f, Z_0)$.

$\therefore$



b, Z0 | bZ0 a, b | ε

b, b | bb b, a | ε

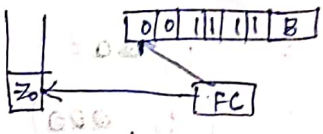
(2) Sol^n: $L = \{0^n 1^{2n} \mid n \geq 1\}$.

Consider the string 001111

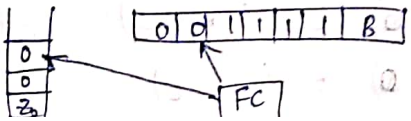
For every input '0', add '00' into stack.

For every input '1', pop '0' from stack.

Step 1:



Step 2:



(2) Sol^n: $L = \{0^n 1^{2n} \mid n \geq 1\}$.

011

001111

$\delta(q_0, 0, Z_0) = (q_0, 0Z_0)$

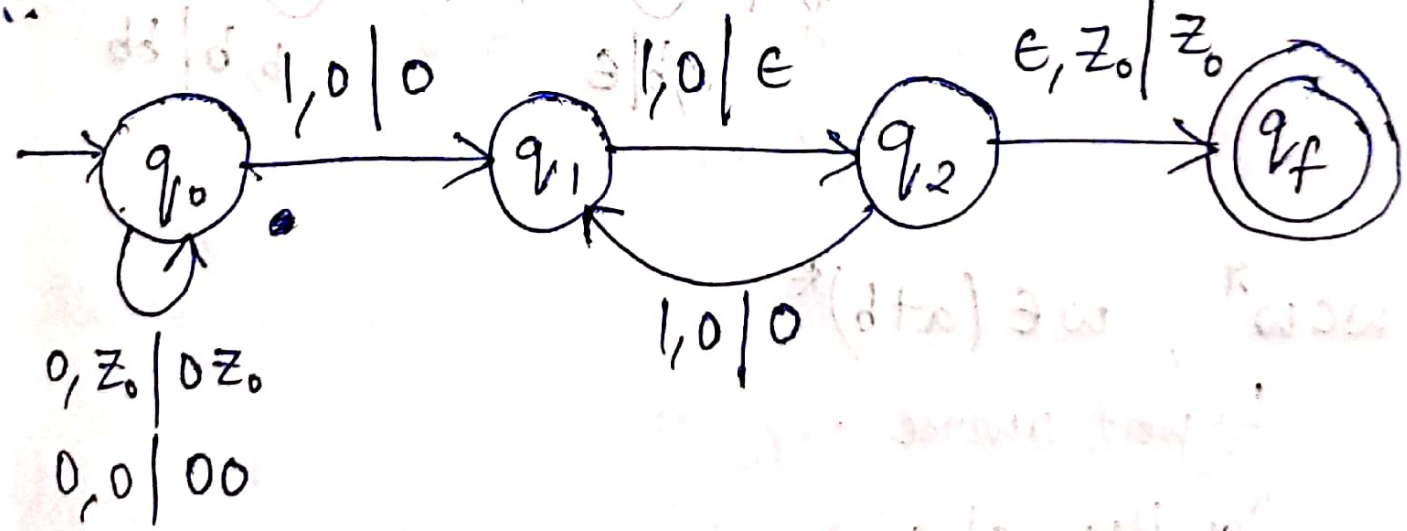
$\delta(q_0, 0, 0) = (q_0, 00)$

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

$\delta(q_1, 1, 0) = (q_2, \epsilon)$

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

$\delta(q_2, \epsilon, Z_0) = (q_f, Z_0)$.



This the PDA which accepts $0^n 1^{2n} / n \geq 1$.

Q. Design a PDA which accepts

$L = \{ a^3 b^n c^n \mid n \geq 0 \}$.

Soln:

$\delta(q_0, a, Z_0) = (q_1, Z_0)$

$\delta(q_1, a, Z_0) = (q_2, Z_0)$

$\delta(q_2, a, Z_0) = (q_3, Z_0)$ $\Rightarrow \delta(q_3, \epsilon, Z_0) = (q_f, Z_0)$

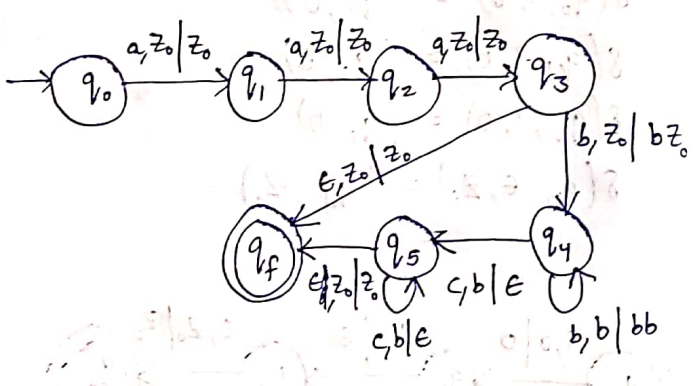
$\delta(q_3, b, Z_0) = (q_4, bZ_0)$

$\delta(q_4, b, b) = (q_4, bb)$

$\delta(q_4, c, b) = (q_5, \epsilon)$

$\delta(q_5, c, b) = (q_5, \epsilon)$

$\delta(q_5, \epsilon, Z_0) = (q_f, Z_0)$



Q. wcw^R, $w \in (a+b)^*$

↳ word reverse.

Soln: Consider abcba ✓

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, a, b) = (q_0, ab)$

$\delta(q_0, b, Z_0) = (q_0, bZ_0)$

$\delta(q_0, b, b) = (q_0, bb)$

$\delta(q_0, b, a) = (q_0, ba)$

$\delta(q_0, c, Z_0) = (q_1, Z_0)$

$\delta(q_0, c, a) = (q_1, a)$

$\delta(q_0, c, b) = (q_1, b)$

$\delta(q_1, a, a) = (q_1, \epsilon)$

$\delta(q_1, b, b) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, Z_0) = (q_f, Z_0)$

Equivalence of acceptance of final state and empty stack:-

$L = a^n b^n$

$\delta(q_0, a, Z_0) \rightarrow (q_0, aZ_0)$

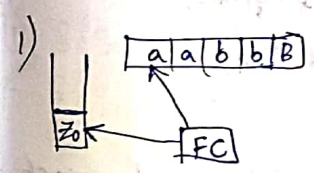
$\delta(q_0, a, a) \rightarrow (q_0, aa)$

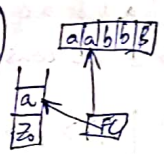
$\delta(q_0, b, a) \rightarrow (q_1, \epsilon)$

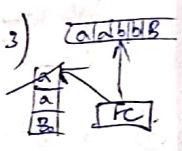
$\delta(q_1, b, a) \rightarrow (q_1, \epsilon)$

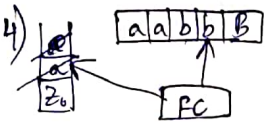
$\delta(q_1, \epsilon, Z_0) \rightarrow (q_f, Z_0)$

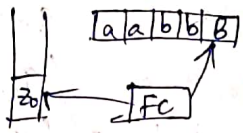
Assume aabb.

1) 

2) 

3) 

4) 

5) 

Empty Stack.

Final state :- ~ same as transition diagram. (can write in any manner).

State (s)	i/p a a b b	transition (t)	top	State after move
q_0	a	①	Z_0	q_0
q_0	a	②	a	q_0
q_0	b	③	a	q_1
q_1	b	④	a	q_1
q_1	ϵ		Z_0	q_f

Find whether given langs are R.L, CFL?

① $a^k \mid k$ is even.
R.L ✓ (we have finite automata).
CFL ✓ bcz it is R.L.

② $a^n b^n \mid n \geq 0$
Not R.L (a's & b's same & infinite).
CFL ✓ (bcz we constructed PDA).

③ $a^i b^j c^k \mid i > j > k$
Not Regular.L. because i, j, k depend on each other.

Not CFL (as there are 3 comparisons, we can't construct PDA).

④ $a^i b^j c^k \mid j = i+k$
Not R.L (i, j, k depend on each other).
$a^i b^{i+k} c^k \longrightarrow a^i b^i \; b^k c^k$
push a for a, pop b for b; push b for b, pop b for c

Hence it is CFL.

Constructing PDA for given CFG :-

① $S \longrightarrow aAA$
$A \longrightarrow aS \mid bS \mid a$

soln: $S \longrightarrow aAA$ ✓ $\delta(q, a, S) \longrightarrow (q, AA)$
$A \longrightarrow aS \mid bS \mid a$ ✓ $\delta(q, a, A) \longrightarrow (q, S)$
$\delta(q, b, A) \longrightarrow (q, S)$
$\delta(q, a, A) \longrightarrow (q, \epsilon)$

Conversion of PDA to CFG :-

① $\delta(q_0, b, Z_0) = (q_0, ZZ_0)$ — 1
$\delta(q_0, \epsilon, Z_0) = (q_0, \epsilon)$ — 2
$\delta(q_0, b, Z) = (q_0, ZZ)$ — 3
$\delta(q_0, a, Z) = (q_1, Z)$ — 4

$\delta(q_1, b, z) = (q_1, \epsilon)$ — 5

$\delta(q_1, a, z_0) = (q_0, z_0)$ — 6

Sol^n: First we add S transition for every problem.

$S \rightarrow [q_0, z_0, q_0] \mid [q_0, z_0, q_1]$

[∵ there there are q_0 & q_1 states overall and initially z_0 on top of stack].

Now consider ①,

$\delta(q_0, b, z_0) = (q_0, zz_0)$ — 2 to 4 we get so [ ] [ ]

$[q_0, z_0, q_0] \rightarrow b[q_0, z q_0] [q_0 z_0 q_0]$

$[q_0, z_0, q_0] \rightarrow b[q_0, z q_1] [q_1 z_0 q_0]$

$[q_0, z_0, q_1] \rightarrow b[q_0, z q_0] [q_0 z_0 q_1]$

$[q_0, z_0, q_1] \rightarrow b[q_0, z q_1] [q_1 z_0 q_1]$

commas not a problem (better remove commas).

② $\delta(q_0, \epsilon, z_0) = (q_0, \epsilon)$

$[q_0, z_0, q_0] \rightarrow \epsilon$

③ $\delta(q_0, b, z) = (q_0, zz)$

$[q_0, z, q_0] \rightarrow b[q_0 z q_0] [q_0 z q_0]$

$[q_0, z, q_0] \rightarrow b[q_0 z q_1] [q_1 z q_0]$

$[q_0, z, q_1] \rightarrow b[q_0 z q_0] [q_0 z q_1]$

$[q_0, z, q_1] \rightarrow b[q_0 z q_1] [q_1 z q_1]$

④ $\delta(q_0, a, z) = (q_1, z)$

$[q_0 z q_0] \rightarrow a[q_1 z q_0]$

$[q_0 z q_1] \rightarrow a[q_1 z q_1]$

⑤ $\delta(q_1, b, z) = (q_1, \epsilon)$

$[q_1, z, q_1] \rightarrow b$

⑥ $\delta(q_1, a, z_0) = (q_0, z_0)$

$[q_1 z_0 q_0] \rightarrow a[q_0, z_0, q_0]$

$[q_1, z_0, q_1] \rightarrow a[q_0, z_0, q_1]$

Now rename the variables.

$q_0 z_0 q_0$ to A, $q_0 z_0 q_1$ to B, $q_1 z_0 q_0$ to C,

$q_1 z_0 q_1$ to D, $q_0 z q_0$ to E, $q_0 z q_1$ to F

$q_1 z q_1$ to G, $q_1 z q_0$ to H,

Hence the productions are renamed as:

$S \rightarrow A \mid B$

$A \rightarrow bEA \mid bFC$

$B \rightarrow bEB \mid bFD$

$E \rightarrow bEE \mid bFH$

$F \rightarrow bEF \mid bFG$

$E \rightarrow \epsilon \mid aH$

$F \rightarrow aG$

Q2. Give the equivalent CFG for the following PDA $M = \{\{q_0, q_1\}, \{0,1\}, \{0,1,Z_0\}, \delta, q_0, Z_0, \phi\}$ where δ is defined by

$\delta(q_0, \epsilon, Z_0) = (q_1, \epsilon)$ — 1
$\delta(q_0, 0, Z_0) = (q_0, 0Z_0)$ — 2
$\delta(q_0, 0, 0) = (q_0, 00)$ — 3
$\delta(q_0, 1, 0) = (q_0, 10)$ — 4
$\delta(q_0, 1, 1) = (q_0, 11)$ — 5
$\delta(q_0, 0, 1) = (q_1, \epsilon)$ — 6
$\delta(q_1, 0, 1) = (q_1, \epsilon)$ — 7
$\delta(q_1, 0, 0) = (q_1, \epsilon)$ — 8
$\delta(q_1, \epsilon, Z_0) = (q_1, \epsilon)$ — 9

Soln: First add the S transition.

$S \rightarrow [q_0, Z_0, q_0] \mid [q_0, Z_0, q_1]$

Now consider ①

$\delta(q_0, \epsilon, Z_0) = (q_1, \epsilon)$

$[q_0, Z_0 q_0] \rightarrow \epsilon$
$[q_0 Z_0 q_1] \rightarrow \epsilon$

②, $\delta(q_0, 0, Z_0) = (q_0, 0Z_0)$

$[q_0 Z_0 q_0] \rightarrow 0[q_0 0 q_0][q_0 Z_0 q_0]$
$[q_0 Z_0 q_0] \rightarrow 0[q_0 0 q_1][q_1 Z_0 q_0]$
$[q_0 Z_0 q_1] \rightarrow 0[q_0 0 q_0][q_0 Z_0 q_1]$
$[q_0 Z_0 q_1] \rightarrow 0[q_0 0 q_1][q_1 Z_0 q_1]$

③ $\delta(q_0, 0, 0) = (q_0, 00)$

$[q_0 0 q_0] \rightarrow 0[q_0 0 q_0][q_0 0 q_0]$
$[q_0 0 q_0] \rightarrow 0[q_0 0 q_1][q_1 0 q_0]$
$[q_0 0 q_1] \rightarrow 0[q_0 0 q_0][q_0 0 q_1]$
$[q_0 0 q_1] \rightarrow 0[q_0 0 q_1][q_1 0 q_1]$

④ $\delta(q_0, 1, 0) = (q_0, 10)$.

$[q_0 0 q_0] \rightarrow 1[q_0 1 q_0][q_0 0 q_0]$
$[q_0 0 q_0] \rightarrow 1[q_0 1 q_1][q_1 0 q_0]$
$[q_0 0 q_1] \rightarrow 1[q_0 1 q_0][q_0 0 q_1]$
$[q_0 0 q_1] \rightarrow 1[q_0 1 q_1][q_1 0 q_1]$

⑤ $\delta(q_0, 1, 1) = (q_0, 11)$

$[q_0 1 q_0] \rightarrow 1[q_0 1 q_0][q_0 1 q_0]$
$[q_0 1 q_0] \rightarrow 1[q_0 1 q_1][q_1 1 q_0]$
$[q_0 1 q_1] \rightarrow 1[q_0 1 q_0][q_0 1 q_1]$
$[q_0 1 q_1] \rightarrow 1[q_0 1 q_1][q_1 1 q_1]$

⑥ $\delta(q_0, 0, 1) = (q_1, \epsilon)$

$[q_0 1 q_0] \rightarrow 0$
$[q_0 1 q_1] \rightarrow 0$

⑦ $\delta(q_1, 0, 1) = (q_1, \epsilon)$

$[q_1 1 q_1] \rightarrow 0$

⑧ $\delta(q_1, 0, 0) = (q_1, \epsilon)$

$[q_1 0 q_1] \rightarrow 0$

⑨ $\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$

$[q_1 z_0 q_1] \rightarrow \epsilon$

Now rename the variables.

$q_0 z_0 q_0$ to A, $q_0 z_0 q_1$ to B, $q_0 0 q_0$ to C, $q_0 0 q_1$ to D, $q_0 z_0 q_0$ to E, $q_1 z_0 q_1$ to F, $q_1 0 q_0$ to G, $q_1 0 q_1$ to H, $q_0 1 q_0$ to I, $q_0 1 q_1$ to J, $q_1 1 q_0$ to K, $q_1 1 q_1$ to L,

Hence the Productions are renamed as:

$S \rightarrow A \mid B$

$A \rightarrow \epsilon \mid 0CA \mid 0DE$

$B \rightarrow \epsilon \mid 0CB \mid 0DF$

$C \rightarrow 0CC \mid 0DG \mid 1IC \mid 1JG$

$D \rightarrow 0CD \mid 0DH \mid 1ID \mid 1JH$

$I \rightarrow 1II \mid 1JK \mid 0$

$J \rightarrow 1IJ \mid 1JL \mid 0$

$L \rightarrow 0$

$H \rightarrow 0$

$F \rightarrow \epsilon$

DPDA (Deterministic Push Down Automata)

NPDA (Non-deterministic " " ")

DPDA	NPDA
WCW^R	WW^R
We need central point because we need to know upto where to push & where to pop.	No need of central point

Accepts Deterministic CFL

DCFL (between regular & CFL)

7 tuples.

Instantaneous description of PDA :-

ex: ababcbab. For given string write instantaneous description of PDA.

$(q, ababcbab, \epsilon)$

Consider 'a' and push into stack.

$= (q, babcbab, a)$

push 'b' $= (q, abcbab, ba)$

push 'a' $= (q, bcbab, aba)$

push 'b' $= (q, cbab, baba)$

while reading 'c' don't push 'c' into the stack as it is in WCW^R form.

~~16/7/21~~ ~~Unit-5~~

$= (q, \bullet bab, baba)$

While reading 'b', the 'b' in stack will be popped.

$= (q, ab, aba)$

read 'a' & pop 'a' in the stack.

$= (q, b, ba)$

read 'b' & pop 'b' in the stack.

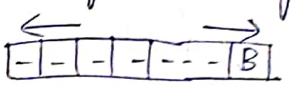
$= (q, \epsilon, a)$

'a' is still present in the stack so it is not considerable.

17/7/21. Unit-5

Turing Machine :-

* Discovered by Alan Turing in the year 1936.



B = Blank.

* If you are in a particular block we can go from left to right or from right to left.

7 tuples are,

$(Q, \Sigma, \Gamma, \delta, q_0, B, F)$.

Q = finite set of states

Σ = finite input alphabet (never includes blank)

Γ = tape alphabet (includes Blank)

δ = transition function.

$\delta = Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$.

Depending on its present state & present tape alphabet, it will move to new state, change the tape symbol and move head pointer to either left or right.

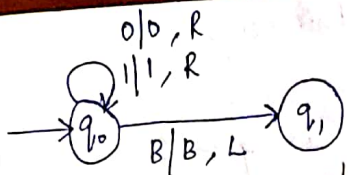
q_0 - Stack state

B - Blank symbol

F - final state

Q. Design a turing machine to accept the string belonging to the language $(0+1)^*$.

Soln:



$\delta(q_0, 0) = (q_0, 0, R)$ — 0/p right

$\delta(q_0, 1) = (q_0, 1, R)$

$\delta(q_0, B) = (q_1, B, L)$ — left

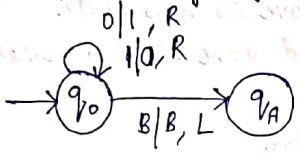
Tabular representation:

	0	1	B
q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	(q_1, B, L)
q_1	—	—	—

H.W.

Design a turing machine for finding 1's complement of a given binary number.

Soln:



q_A : Acceptance state i.e final state

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

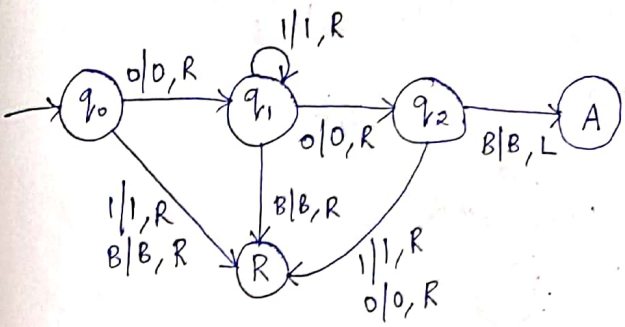
$\delta(q_0, 1) = (q_0, 0, R)$

$\delta(q_0, B) = (q_A, B, L)$.

	0	1	B
q_0	$(q_0, 1, R)$	$(q_0, 0, R)$	(q_A, B, L)
q_A	—	—	—

Q1. Design a TM that accepts the string 01^*0.

Soln: 00, 010, 0110, ...



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

$\delta(q_0, B) = (R, B, R)$

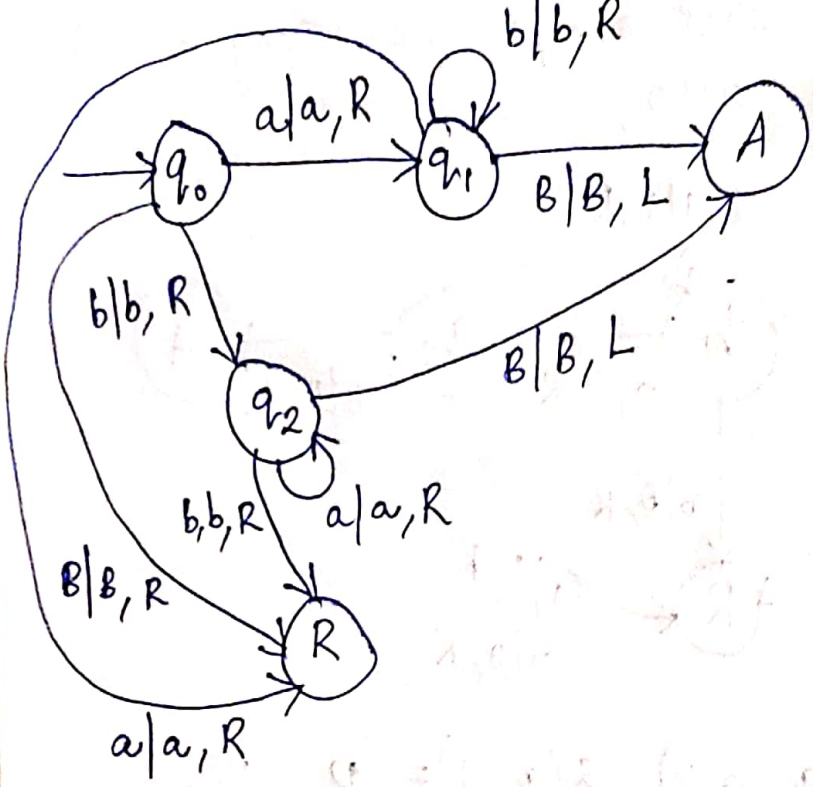
$\delta(q_1, 0) = (q_2, 0, R)$, $\delta(q_1, 1) = (q_1, 1, R)$, $\delta(q_1, B) = (R, B, R)$

$\delta(q_2, 0) = (R, 0, R)$, $\delta(q_2, 1) = (R, 1, R)$, $\delta(q_2, B) = (A, B, L)$

	0	1	B
q_0	$(q_1, 0, R)$	$(R, 1, R)$	(R, B, R)
q_1	$(q_2, 0, R)$	$(q_1, 1, R)$	(R, B, R)
q_2	$(R, 0, R)$	$(R, 1, R)$	(A, B, L)
A	—	—	—
R	—	—	—

Q2. Design a TM that accepts the strings ab^* or ba^*.

Soln: { a, b, ab, abb, abbb, ba, baa, baaa, ... }



$\delta(q_0, a) = (q_1, a, R)$, $\delta(q_0, b) = (q_2, b, R)$, $\delta(q_0, B) = (R, B, R)$

$\delta(q_1, a) = (R, a, R)$, $\delta(q_1, b) = (q_1, b, R)$, $\delta(q_1, B) = (A, B, L)$

$\delta(q_2, a) = (q_2, a, R)$, $\delta(q_2, b) = (R, b, R)$, $\delta(q_2, B) = (A, B, L)$

	a	b	B
q_0	(q_1, a, R)	(q_2, b, R)	(R, B, R)
q_1	(R, a, R)	(q_1, b, R)	(A, B, L)
q_2	(q_2, a, R)	(R, b, R)	(A, B, L)
A	—	—	—
R	—	—	—