Module - 1

V Sem
Branch CSE

Finite State Machines (FSMs)

Deterministic Finite State Machines:

The DFSM M is a quintuple $(K, \Sigma, \delta, S, A)$ where

K is a finite set of states
Σ is the i/p alphabet
$S \in K$ is the start state
$A \subseteq K$ is the set of accepting states
δ is the transition fn. It maps

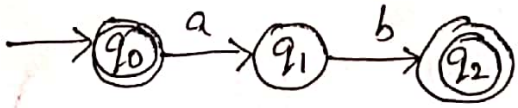
$K \times \Sigma$ to K
state i/p symbol state.

Problems on DFSM:

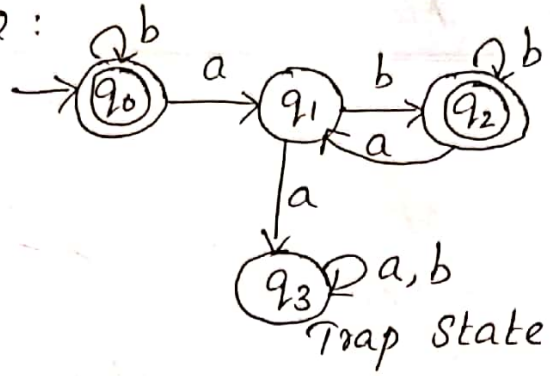
Obtain the DFSM for the following languages.

1. $L = \{ w \in \{a,b\}^* :$ every a is immediately followed by a $b\}$

Step1:



Step 2 :

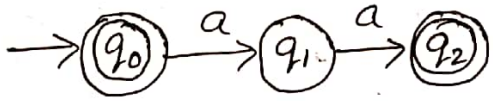


Transition Table.

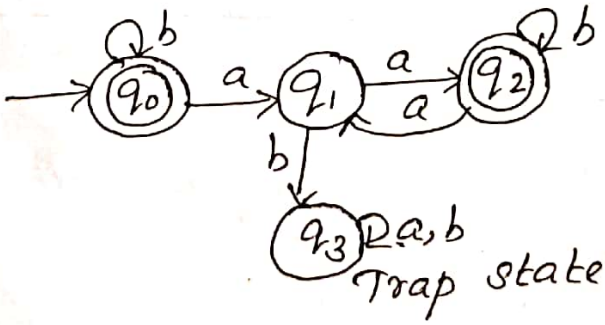
	δ	a	b
→	q_0	q_1	q_0
	q_1	Trap	q_2
*	q_2	q_1	q_2

2. $L = \{w \in \{a,b\}^* :$ every a region in w is of even length.$\}$

Step 1 :



Step 2 :

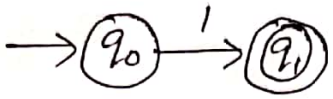


Transition Table

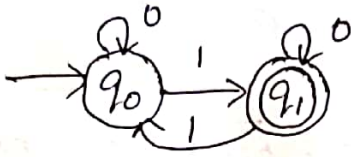
	δ	a	b
→	q_0	q_1	q_0
	q_1	q_2	Trap
*	q_2	q_1	q_2

3. $L = \{ w \in \{0,1\}^* : w \text{ has odd parity} \}$

Step 1:



Step 2:

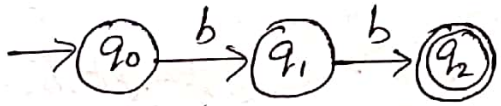


Transition Table

	δ	0	1
→	q_0	q_0	q_1
*	q_1	q_1	q_0

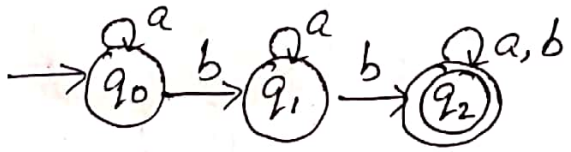
4. $L = \{ w \in \{a,b\}^* : w \text{ has no more than one } b \}$

Step 1:



Do for more than 1 b and then take complement.

Step 2:



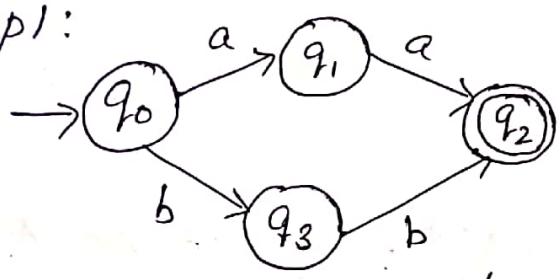
Step 3:

Transition Table.

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

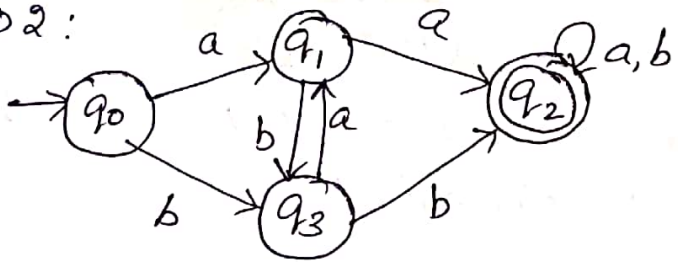
5. $L = \{w \in \{a,b\}^* :$ no two consecutive characters are same $\}$

Step1:

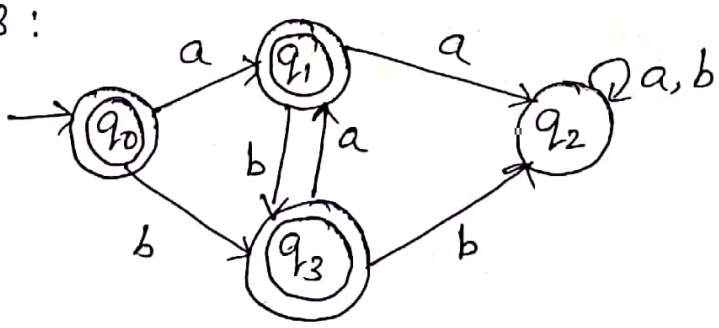


Do for consecutive characters are same and then take complement.

Step2:



Step3:



Transition Table

	δ	a	b
→	q_0	q_1	q_3
*	q_1	q_2	q_3
	q_2	q_2	q_2
*	q_3	q_1	q_2

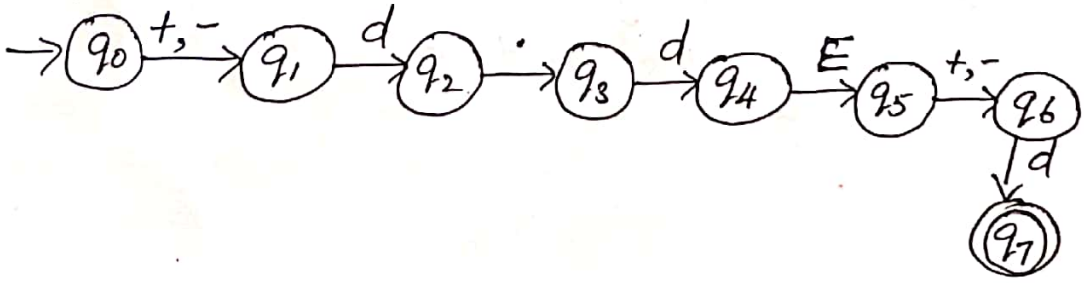
6. Let FLOAT = {w : w is the string representation of a floating point number}

Assume the following syntax :

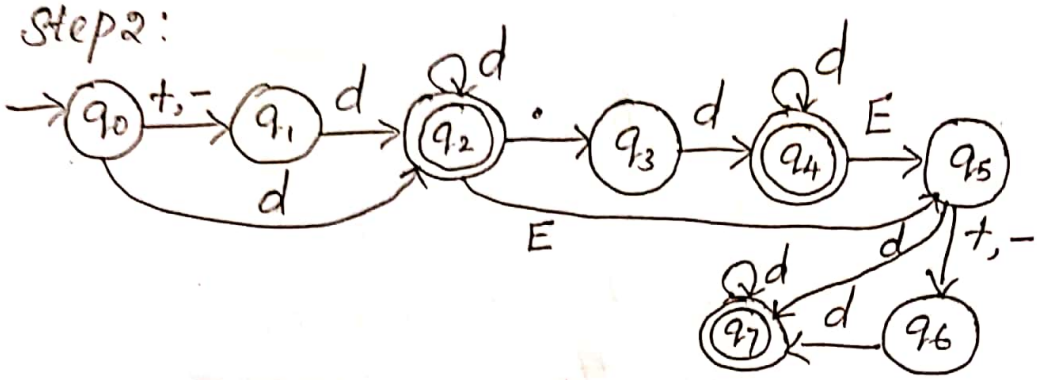
*) A floating point number is an optional sign, followed by a decimal number, followed by an optional exponent.

*) A decimal number may be of the form x or x.y, where x and y are non empty strings of decimal digits.

Step 1 :

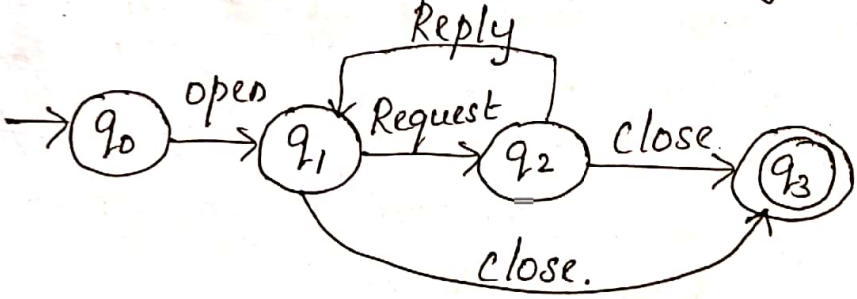


Step2:

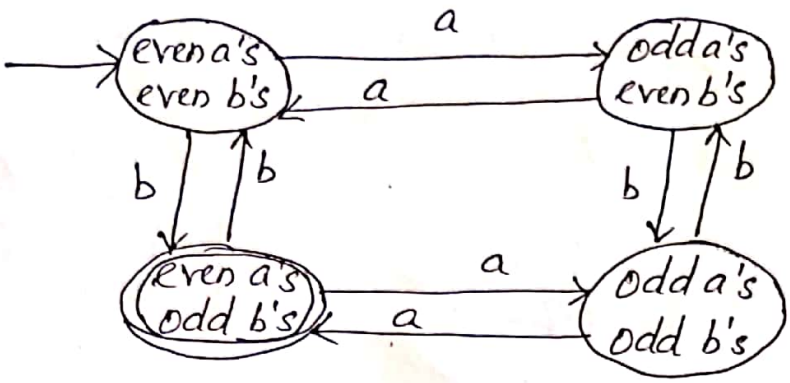


7. L is the language that contains all the legal sequences of messages exchanged b/w a client and a server using simple communication protocol.

Let Σ_L = {open, Request, Reply, Close, }

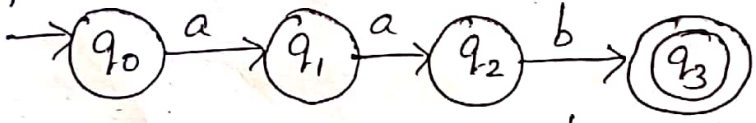


8. $L = \{w \in \{a,b\}^* : w$ contains an even number of a's and an odd number of b's $\}$.



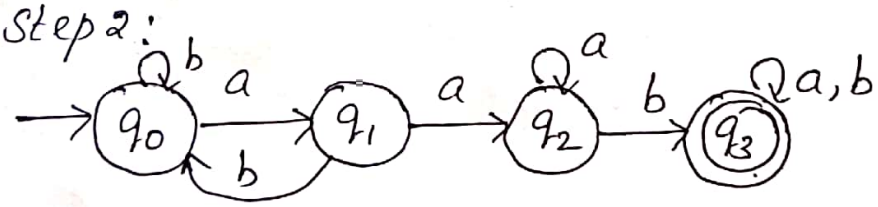
9. $L = \{w \in \{a,b\}^* : w$ doesnot contain the substring aab

Step 1:

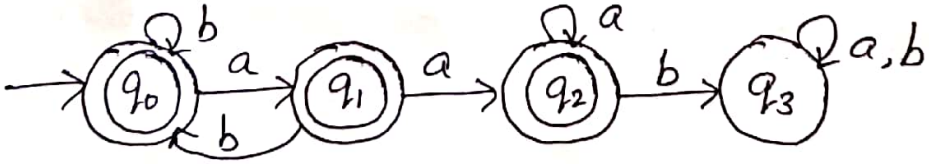


Do for substring aab and then take complement.

Step 2:



Step 3:



Transition Table:

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

Non Deterministic FSM.

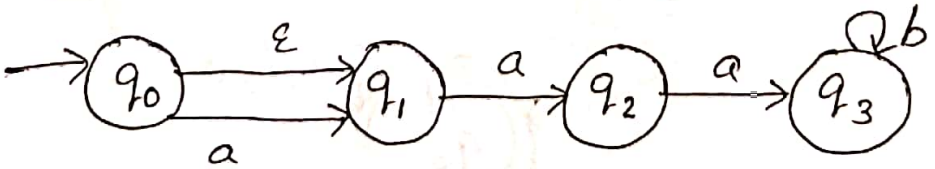
A NDFSM M is a quintuple machine where $M = K, \Sigma, \Delta, S, A$

K is a finite set of states,
Σ is an alphabet
$S \in K$ is the start state
$A \subseteq K$ is the set of final states
Δ is the transition relation. It is a finite subset of $K \times (\Sigma \cup \{\varepsilon\}) \times K$.

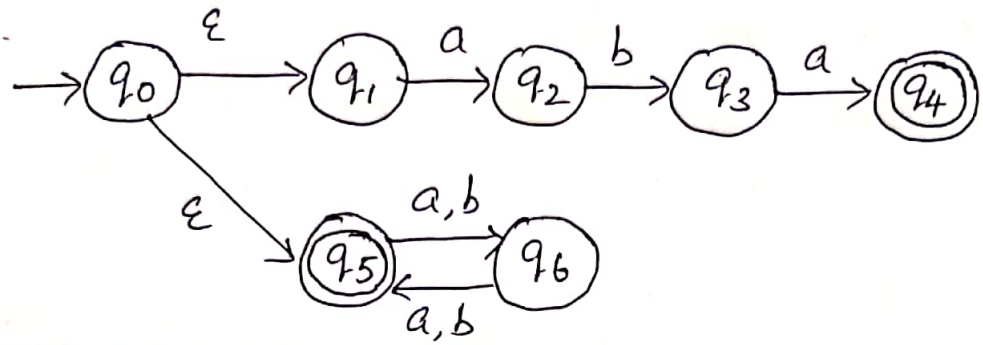
Problems on NDFSM

Obtain the NDFSM for the following languages.

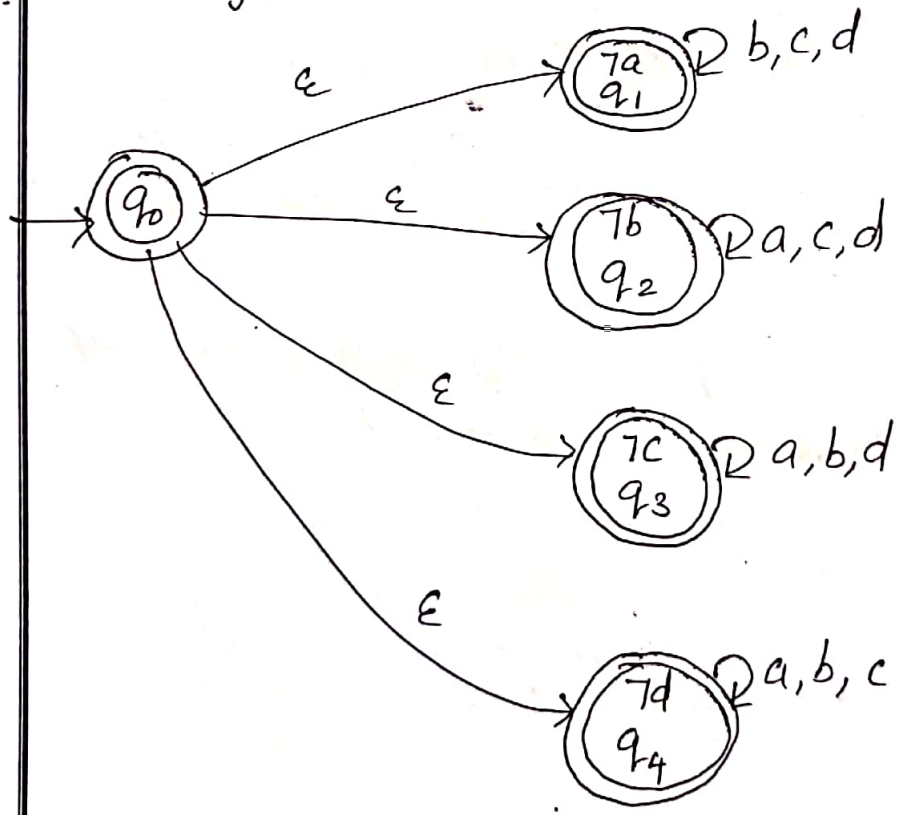
1) $L = \{w \in \{a,b\}^* :$ every w is made up of an optional a followed by aa followed by 0 or more b's $\}$



2) $L = \{ w \in \{a,b\}^* : w = aba \text{ or } |w| \text{ is even} \}$

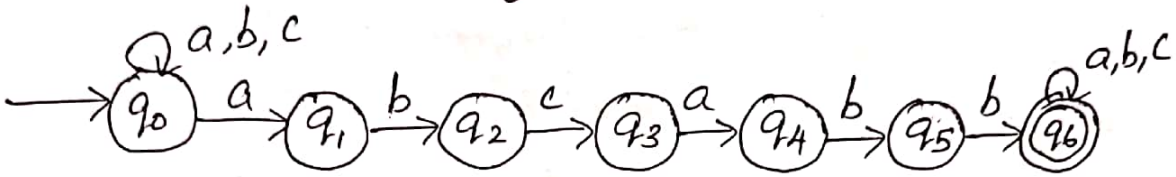


3) Let $\Sigma = \{a,b,c,d\}$, $L_{missing} = \{ w :$ there is a symbol $a_i \in \Sigma$ not appearing in $w \}$.

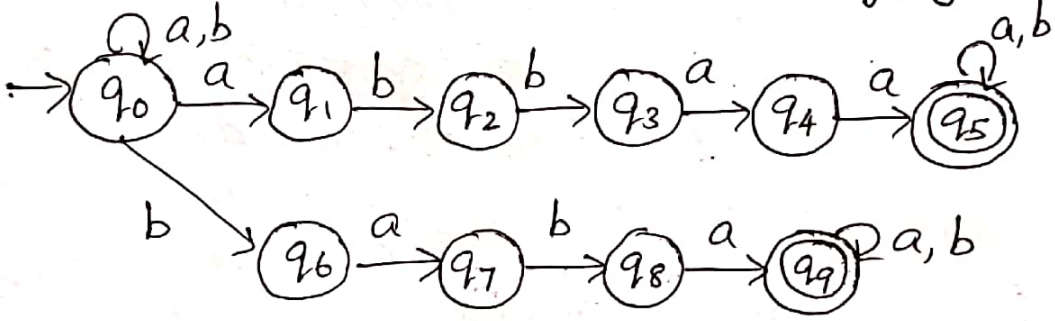


4) Let $L = \{w \in \{a,b,c\}^* : \exists x, y \in \{a,b,c\}^* (w = x\,abcabb\,y)\}$

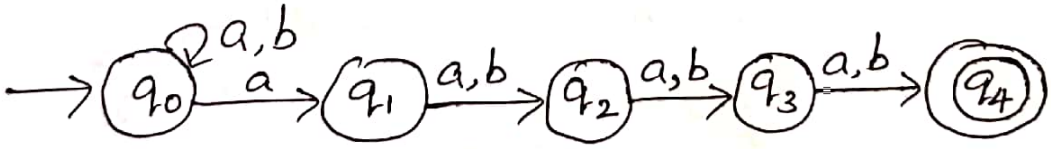
In other words w must contain atleast one occurrence of the substring abcabb.



5) Let $L = \{w \in \{a,b\}^* : \exists x, y \in \{a,b\}^* ((w = x\,abbaa\,y) \vee (w = x\,baba\,y))\}$



b) $L = \{w \in \{a,b\}^* :$ the fourth from the last character is a $\}$



Conversion of NDFSM to DFSM

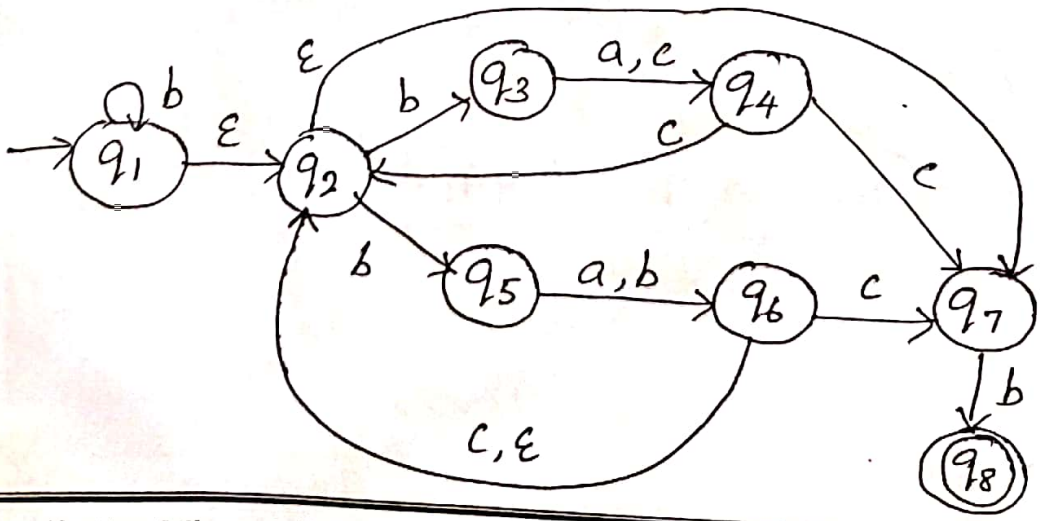
Algorithm used is ndfsmtodfsm

The algorithm ndfsmtodfsm is important for two reasons:

*) It proves the theorem that, for every NDFSM there exists an equivalent DFSM.

*) It lets us use nondeterminism as a design tool, even though we may ultimately need a deterministic m/c. If we have implementation of ndfsmtodfsm then if we can build an NDFSM to solve our problem, ndfsmtodfsm can easily construct an equivalent DFSM.

Consider the NDFSM. Using ndfsmtodfsm to build a DFSM.



$ECLOSE(q_1) = \{q_1, q_2, q_7\}$ —— (A)

Consider state (A)

when i/p is a

$\delta(A, a) = ECLOSE(\delta(A, a))$

$= ECLOSE(\phi)$

$= \phi$

when i/p is b

$\delta(A, b) = ECLOSE(\delta(A, b))$

$= ECLOSE(q_1, q_3, q_5, q_8)$

$= \{q_1, q_2, q_7, q_3, q_5, q_8\}$

$= \{q_1, q_2, q_3, q_5, q_7, q_8\}$ —— (B)

when i/p is c

$\delta(A, c) = ECLOSE(\delta(A, c))$

$= ECLOSE(\phi)$

$= \phi$

Consider state (B)

When i/p is a

$\delta(B, a) = ECLOSE(\delta(B, a))$

$= ECLOSE(q_4, q_6)$

$= \{q_4, q_6, q_2, q_7\}$

$= \{q_2, q_4, q_6, q_7\}$ —— (C)

When i/p is b

$\delta(B, b) = ECLOSE(\delta(B, b))$

$= ECLOSE(q_1, q_3, q_5, q_8, q_6)$

$= \{q_1, q_2, q_7, q_3, q_5, q_8, q_6, q_2\}$

$= \{q_1, q_2, q_3, q_5, q_6, q_7, q_8\}$ — (D)

When i/p is c

$\delta(B, c) = ECLOSE(\delta(B, c))$

$= ECLOSE(q_4)$

$= \{q_4\}$ — (E)

Consider state (C)

When i/p is a

$\delta(C, a) = ECLOSE(\delta(C, a))$

$= ECLOSE(\phi)$

$= \phi$

When i/p is b

$\delta(C, b) = ECLOSE(\delta(C, b))$

$= ECLOSE(q_3, q_5, q_8)$

$= ECLOSE(q_3, q_5, q_8\}$ — (F)

When i/p is c

$\delta(C, c) = ECLOSE(\delta(C, c)$

$= ECLOSE(q_7, q_2\}$

$= \{q_2, q_7\}$ — (G)

Consider the state (D)

$\delta(D,a)$ = ECLOSE $(\delta(D,a))$

= ECLOSE (q_4, q_6)

= $\{q_4, q_6, q_2, q_7\}$

= $\{q_2, q_4, q_6, q_7\}$ — (C)

when i/p is b

$\delta(D,b)$ = ECLOSE$(\delta(D,b)$

= ECLOSE $(q_1, q_3, q_5, q_6, q_8)$

= $\{q_1, q_2, q_7, q_3, q_5, q_6, q_8\}$

= $\{q_1, q_2, q_3, q_5, q_6, q_7, q_8\}$ — (B)

when i/p is c

$\delta(D,c)$ = ECLOSE $(\delta(D,c))$

= ECLOSE (q_4, q_2, q_7)

= $\{q_2, q_4, q_7\}$ — (H)

Consider state (E)

when i/p is a

$\delta(E,a)$ = ECLOSE $(\delta(E,a))$

= ECLOSE$(\phi) = \phi$

when i/p is b

$\delta(E,b)$ = ECLOSE$(\delta(E,b))$

= ECLOSE$(\phi) = \phi$

When i/P is c

$\delta(E, c) = ECLOSE(\delta(E, c))$

$= ECLOSE(q_2, q_7)$

$= \{q_2, q_7\}$ —— (G)

Consider state (F)

when i/P is a

$\delta(F, a) = ECLOSE(\delta(F, a))$

$= ECLOSE(q_4, q_6)$

$= \{q_2, q_4, q_6, q_7\}$ —— (C)

when i/P is b

$\delta(F, b) = ECLOSE(\delta(F, b))$

$= ECLOSE(q_6)$

$= \{q_6, q_2, q_7\}$

$= \{q_2, q_6, q_7\}$ —— (I)

when i/P is c

$\delta(F, c) = ECLOSE(\delta(F, c))$

$= ECLOSE(\phi) = \phi$

Consider state (G)

when i/P is a

$\delta(q,a)$ = ECLOSE($\delta(q,a)$)
= ECLOSE(ϕ) = ϕ

when i/p is b

$\delta(q,b)$ = ECLOSE($\delta(q,b)$)
= ECLOSE(q_3, q_5, q_8)
= $\{q_3, q_5, q_8\}$ —— (F)

when i/p is c

$\delta(q,c)$ = ECLOSE($\delta(q,c)$)
= ECLOSE(ϕ) = ϕ

Consider state (H)

when i/p is a

$\delta(H,a)$ = ECLOSE($\delta(H,a)$)
= ECLOSE(ϕ) = ϕ

when i/p is b

$\delta(H,b)$ = ECLOSE($\delta(H,b)$)
= ECLOSE(q_3, q_5, q_8)
= $\{q_3, q_5, q_8\}$ —— (F)

when i/p is c

$\delta(H,c)$ = ECLOSE($\delta(H,c)$)
= ECLOSE(q_2, q_7)
= $\{q_2, q_7\}$ —— (G)

Consider State (I)

when i/p is a

$\delta(I, a) = ECLOSE(\delta(I, a))$

$= ECLOSE(\phi) = \phi$

when i/p is b

$\delta(I, b) = ECLOSE(\delta(I, b))$

$= ECLOSE(q_3, q_5, q_8)$

$= \{q_3, q_5, q_8\}$ — (F)

when i/p is c

$\delta(I, c) = ECLOSE(\delta(I, c))$

$= ECLOSE(q_7, q_2)$

$= \{q_2, q_7\}$ — (G)

Transition Table:

δ	a	b	c
→ A	φ	B	φ
* B	C	D	E
C	φ	F	G
* D	C	D	H
E	φ	φ	G
* F	C	I	φ
G	φ	F	φ
H	G	F	G
I	φ	F	G

From FSM to Operational System:

FSM for real problems can be turned into operational systems.

*) An FSM can be translated into a circuit design and implemented directly in hlw.

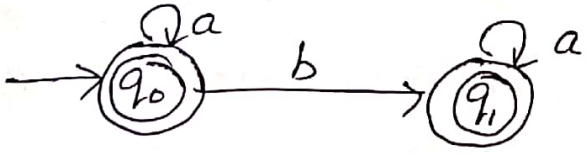
*) An FSM can be simulated by a general purpose interpreter.

*) An FSM can be used as a specification for some critical aspect of the behaviour of a complex system.

Simulators for FSMs

Simulating DFSMs.

Consider the DFSM. $L = \{w \in \{a,b\}^* :$ w contains no more than one b$\}$



q_0 : s = get_next_symbol
If s = end_of_file then accept
Else if s = a then go to q_0
Else if s = b then go to q_1

q_1 : s = get_next_symbol
If s = end_of_file then accept
Else if s = a then go to q_1
Else if s = b then reject
End.

Simple interpreter for DFSM.

dfsmsimulate (M: DFSM, w: string) =

1. st = s
2. Repeat
   2.1 c = get_next_symbol (w)
   2.2 if c ≠ end_of_file then
      2.2.1 st = δ'(st, c)
   until c = end_of_file
3. If st ∈ A then accept else reject.

ndfsmsimulate (M: NDFSM, w: string) =

1. Declare the set st
2. Declare the set st1
3. st = eps (s)
4. Repeat

```
c = get_next_symbol(w)
if c ≠ end_of_file then do
   st1 = ϕ
For all q ∈ st do:
   For all r : (q, c, r) ∈ Δ do:
      st1 = st1 ∪ eps(r)
   st = st1
   If st = ϕ then exit
until c = end_of_file
```

5. If $st \cap A \neq \phi$ then accept else reject.

Equivalence classes $\approx_L$

What are indistinguishable states?
or equivalent states.

Two states p and q of a DFSM are equivalent (indistinguishable) if and only if $\delta(p,w)$ and $\delta(q,w)$ are final states or both $\delta(p,w)$ and $\delta(q,w)$ or non final states.

$\delta(P,w) \in F$ and $\delta(q,w) \in F$

or $\delta(P,w) \notin F$ and $\delta(q,w) \notin F$

Distinguishable States :

$\delta(P,w) \in F$ and $\delta(q,w) \notin F$

or $\delta(P,w) \notin F$ and $\delta(q,w) \in F$

Determine the equivalence classes for the following languages.

1) Let $\Sigma = \{a,b\}$. Let $L = \{w \in \{a,b\}^* :$ no two adjacent characters are the same.

[1]	[ε]	[ε]
[2]	[a, aba, ababa...]	[all non empty strings that end in a and have no identical adjacent characters].
[3]	[b, ab, bab, abab]	[all nonempty strings that end in b and have no identical adjacent characters]

[4] [aa, abaa, ababb...] [all strings that contain atleast one pair of identical adjacent characters]

2) Let $L = \{w \in \Sigma^* :$ every a is immediately followed by a b$\}$

The equivalence classes of L are

[1] [ε, b, abb...] [all strings in L]

[2] [a, abbba....] [all strings that end in a and have no prior a that is not followed by a b]

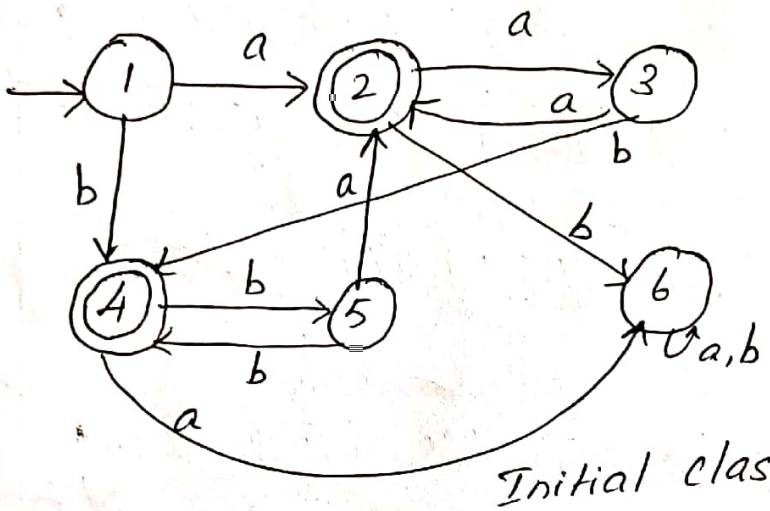
[3] [aa, abaa...] [all strings that contain atleast one instance of aa]

3) Let $\Sigma = \{a, b\}$. Let $L = \{w \in \{a, b\}^* :$ no two adjacent characters are the same.

[1]	[ε]	[ε]
[2]	[a, aba, ababa...]	[all nonempty strings that end in a and have no identical adjacent characters]
[3]	[b, ab, bab, abab...]	[all nonempty strings that end in b and have no identical adjacent characters]
[4]	[aa, abaa, ababb...]	[all strings that contain atleast one pair of identical characters; these strings are not in L, no matter what comes next].

Minimizing DFSM.

Using minDFSM, find a minimal m/c.



Initial classes = {[2,4], [1,3,5,6]}

At Step 1 :

((2,a), [1,3,5,6]) ((4,a), [1,3,5,6]) Don't split

((2,b), [1,3,5,6]) ((4,b), [1,3,5,6])

((1,a), [2,4]) ((3,a), [2,4]) ((5,a), [2,4])

((1,b), [2,4]) ((3,b), [2,4]) ((5,b), [2,4])

((6,a), [1,3,5,6] Split

((6,b), [1,3,5,6]

classes = {[2,4], [1,3,5], [6]}

At Step 2 :

((2,a), [1,3,5]) ((4,a), [6]) Split

((2,b), [6]) ((4,b), [1,3,5])

((1,a), [2,4] ((3,a), [2,4]) ((5,a), [2,4]) Don't split

((1,b), [2,4]) ((3,b), [2,4]) ((5,b), [2,4])

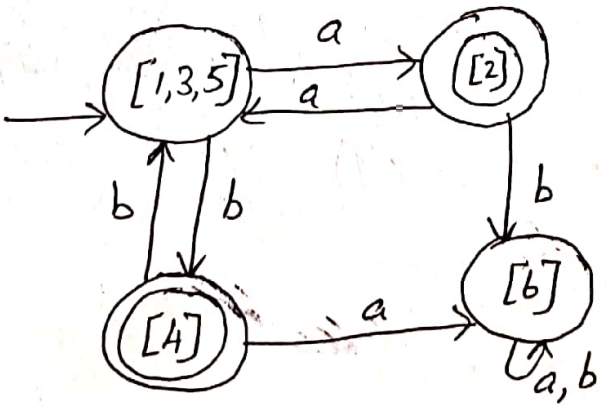
classes = {[2], [4], [1,3,5], [6]}

At Step 3 :

((1,a), [2]) ((3,a), [2]) ((5,a), [2]) Don't split

((1,b), [4] ((3,b), [4]) ((5,b), [4])

So minDFSM is

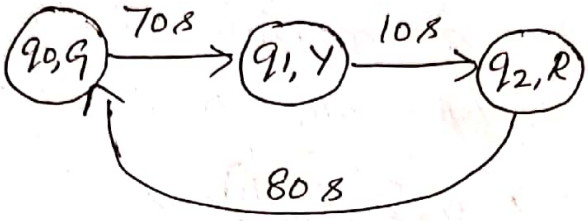


Finite State Transducers:

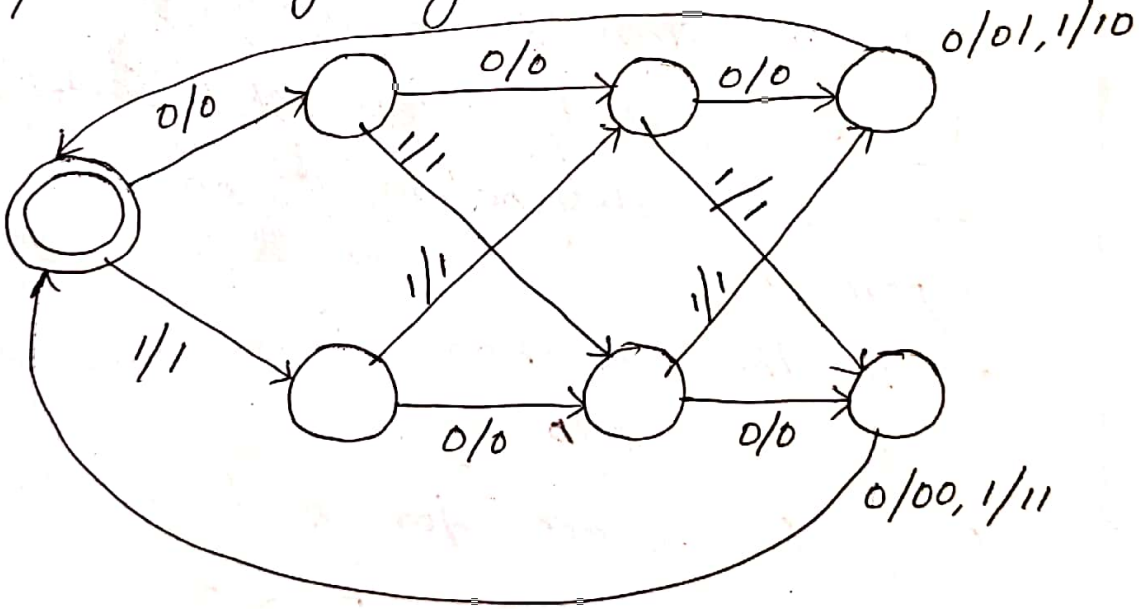
FST associates an o/p with each state of a machine. Both Moore m/c and Mealy m/c is a 7 tuple m/c. $(Q, \Sigma, O, \delta, D; S, A)$

Q is set of finite states.
Σ is i/p alphabet
O is o/p alphabet
$s \in k$ is start state
$A \subseteq k$ is set of accept state (although for some designation is not important).
δ is the transition fn.
D is the o/p fn.
Only D varies for both the m/cs.
Moore m/c : $D : Q \rightarrow O$
Mealy m/c : $D : Q \times \Sigma \rightarrow O$

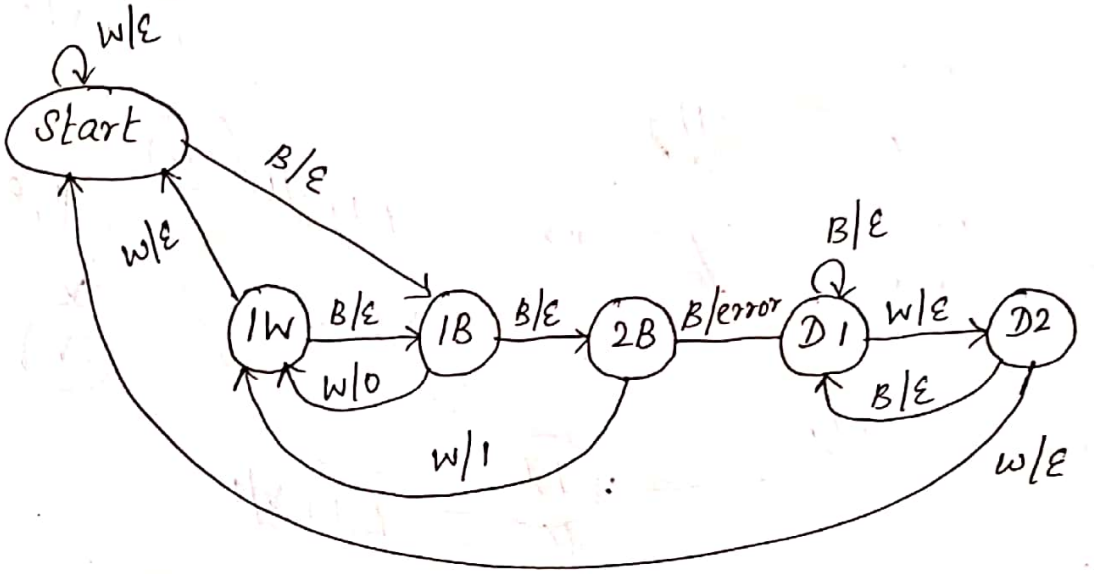
1) Use Moore m/c for designing a single direction traffic light.



2) Using Mealy m/c, construct an odd parity bit generator after every four binary digits that it reads.



3. Using Mealy M/c, construct Bar code reader



*) Correct bar code starts with black column,

*) So white space ahead of the first black column is ignored.

*) Every complete bar code there are atleast two white columns. So the reader should, at that point, reset to be ready to read the next code.

*) If the reader sees three or more black columns in a row, it must indicate an error and stay in its error state until it is reset by seeing two white columns.

Bidirectional Transducers : Soundex dfg.