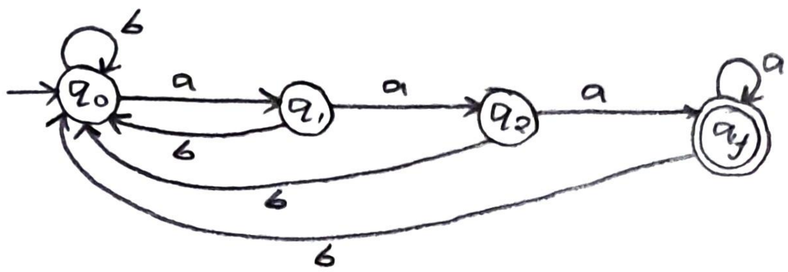
Theory of Computation

Mirza Yasser Ali Baig
21WJ1A6642
CSM(AI&ML)

Assignment - 01

1. Design a DFA that accepts all the strings end with aaa from $\Sigma = \{a, b\}$, draw the corresponding transition table and check whether string w = bbababaaa is accepted or not

Design of a DFA that accepts all the strings end with aaa from $\Sigma = \{a, b\}$



Transition Table

Q	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
q_2	q_f	q_0
q_f^*	q_f	q_0

Given,
String w = bbababaaa

$\delta(q_0, \underline{b}bababaaa) \vdash \delta(q_0, b\underline{b}ababaaa) \vdash \delta(q_0, bb\underline{a}babaaa)$

$\vdash \delta(q_1, bba\underline{b}abaaa) \vdash \delta(q_0, bbab\underline{a}baaa) \vdash \delta(q_1, bbaba\underline{b}aaa)$

$\vdash \delta(q_0, bbabab\underline{a}aa) \vdash \delta(q_1, bbababa\underline{a}a) \vdash \delta(q_2, bbababaa\underline{a})$

$\vdash \delta(q_f, bbababaaa) = q_f$

$\therefore$ The string w = bbababaaa is accepted

2) Describe DFA and write rules for designing DFA

DFA stands for Deterministic Finite Automata it is a type of finite state machine or finite state automata.

DFA is described by $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q \rightarrow$ Set of finite non-empty states

$\Sigma \rightarrow$ Set of finite non-empty input alphabets

$q_0 \rightarrow$ Initial state & $q_0 \in Q$

$F \rightarrow$ Set of final states or accepting states & $F \subseteq Q$

$\delta \rightarrow$ Mapping function / Transition function in the form

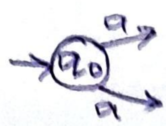
$$\delta : Q \times \Sigma \rightarrow Q$$

As the name suggest the automata is deterministic because with single input it can move exactly to one state.

Rules of designing a DFA

(i) Apply all the input symbols exactly once to each state of the machine.

ex: $\{a, b\}$ 

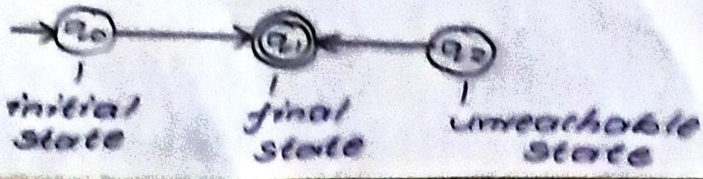
(allowed) 

(not allowed)

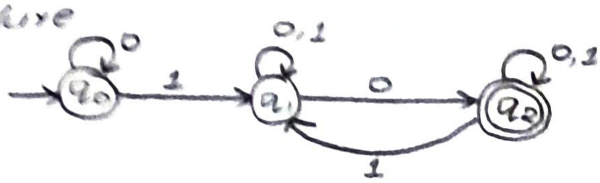
(ii) To satisfy rule (i) if required construct a dead state in the machine

Dead State: A state which is reachable from the initial state but doesn't participate in the acceptance of the string is known as Dead State.

Unreachable State: A state which can not be reached from the initial state is called unreachable state.



3. Convert the following NFA to DFA and explain the procedure



Conversion of NFA to DFA:

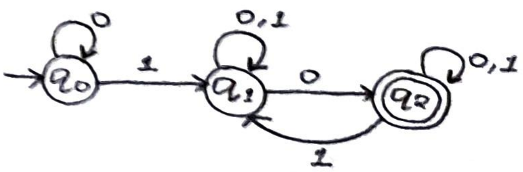
Step(1): Write all transitions from initial state on every input symbol in Σ.

Step(2): Repeat step(1) for every new state, if a transition on some input symbol resulted in a set of states then it is also considered as a new single state.

Step(3): Repeat step(2) until no more new state is found.

Step(4): The final states of equivalent DFA are all those states which consists one accepting state / final state of the given NFA.

Given, NFA

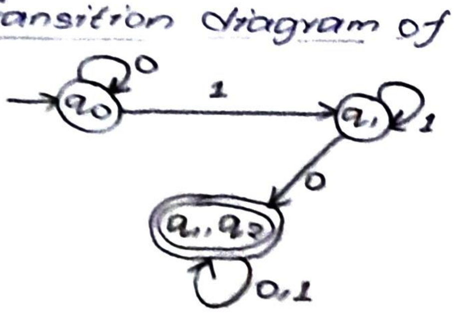


Transition table of DFA

Q	0	1
$\rightarrow q_0$	$\{q_0\}$	$\{q_1\}$
q_1	$\{q_1, q_2\}$	$\{q_1\}$
$\{q_1, q_2\}^*$	$\{q_1, q_2\}$	$\{q_1, q_2\}$

- $\delta(\{q_1, q_2\}, 0) \rightarrow \delta(q_1, 0) \cup \delta(q_2, 0)$
  $\rightarrow \{q_1, q_2\} \cup \{q_2\}$
  $\rightarrow \{q_1, q_2\}$
- $\delta(\{q_1, q_2\}, 1) \rightarrow \delta(q_1, 1) \cup \delta(q_2, 1)$
  $\rightarrow \{q_1\} \cup \{q_1, q_2\}$
  $\rightarrow \{q_1, q_2\}$

Transition diagram of DFA:



4. Explain NFA - ϵ with an example

NFA - ϵ stands for Non-deterministic finite automata with ϵ- moves is a type of finite state machine or finite state automata.

NFA - ϵ is defined by $M = (Q, \Sigma, \delta', q_0, F)$ where,

$Q \rightarrow$ Set of finite non-empty states

$\Sigma \rightarrow$ Set of finite non-empty input alphabets

$q_0 \rightarrow$ Initial state & $q_0 \in Q$

$F \rightarrow$ Set of final states on accepting states & $F \subseteq Q$

$\delta' \rightarrow$ Mapping function / Transition function in the form

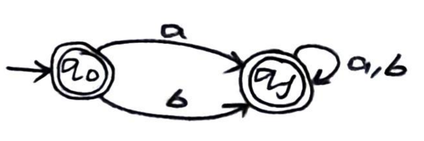
$$\delta' : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$$

$$\delta' : Q \times \Sigma \rightarrow 2^Q$$

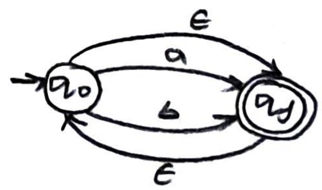
example:

(i) $\Sigma = \{a, b\}$

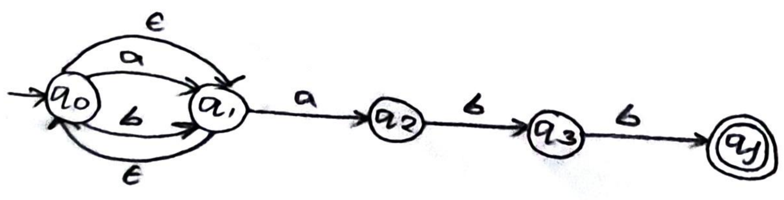
$\Sigma^* = (a+b)^* = \{\epsilon, a, b, aa, bb, ab, ba, aaa, aba, bbb, bab, \ldots\}$



$\simeq$



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



5. Explain the procedure of minimization of DFA with example.

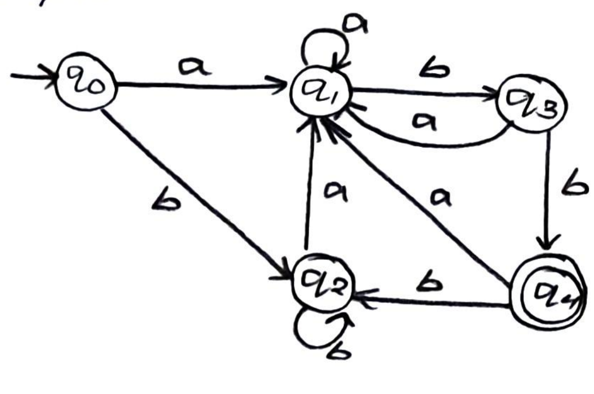
It means reducing the number of states from the given finite automata. Reducing number of states means we have to reduce or remove unreachable state, dead state & redundant state from the given DFA.

Procedure:

1. Remove all the unreachable states from the given DFA.
2. Divide the rest of the states into two groups
   (i) Non-final states (ii) final states
3. Apply every input symbol to each state and findout the resultant state.
4. Now identify the similarity between any two states of the same group and merge them into one.
5. Otherwise divide them to create another group.
6. Repeat step 4 & 5 until no change occur
7. Now draw the transition diagram of the minimized DFA.

example:

(i)



G_1:

$\boxed{q_0 \begin{cases} a & \delta(q_0,a) = q_1 \\ b & \delta(q_0,b) = q_2 \end{cases}}$

$q_1 \begin{cases} a & \delta(q_1,a) = q_1 \\ b & \delta(q_1,b) = q_3 \end{cases}$

$\boxed{q_2 \begin{cases} a & \delta(q_2,a) = q_1 \\ b & \delta(q_2,b) = q_2 \end{cases}}$

(q_0,q_2)

$q_3 \begin{cases} a & \delta(q_3,a) = q_1 \\ b & \delta(q_3,b) = q_4 \end{cases}$

G_2:

$q_4 \begin{cases} a & \delta(q_4,a) = q_1 \\ b & \delta(q_4,b) = q_2 \end{cases}$

Minimized DFA: