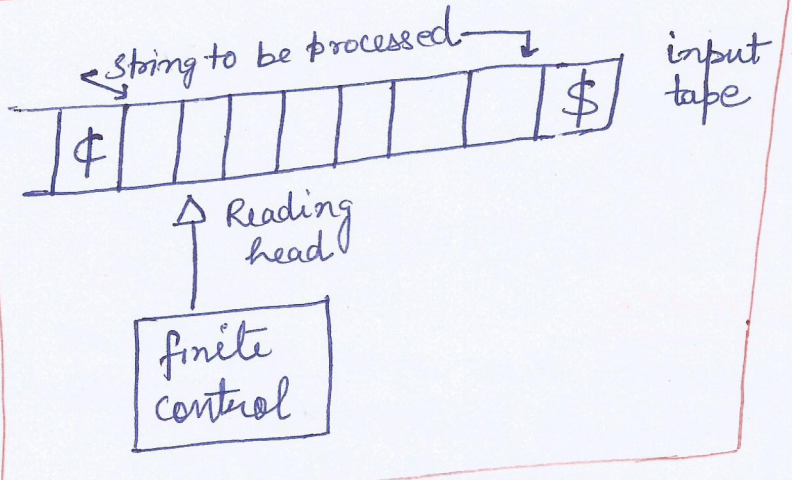
Prof. Minakshi

FINITE AUTOMATA

- An automata that accepts/recognizes a regular language is called a finite automata.

Block diagram of finite automata



Input tape:

- input tape is divided into cells, each cell containing a single symbol from input alphabet Σ.
- The end cells contain end-markers.
- absence of endmarkers indicate infinite length of input tape.
- The left-to-right sequence of symbols b/w to end-markers is the input string to be processed

Reading Head: The head examines one symbol at a time and can move one square/cell either to left or to right.

- Here ~~for~~ the movement of reading-head is restricted to right side.

Finite Control: The input to finite control is the input symbol under the reading head, say a, and the present state q of the automata say q, to give the following outputs:

(a) A motion of Readinghead along the tape to the next square (in some a null move).

(b) The next state of the finite automata given by a transition function $\delta(q, a)$.

* **Finite automata:** Finite automata involve states and transitions among states in response to inputs.
Finite automata are useful for building several different kind of softwares, for example, lexical analyzes component of compiler and system for verifying the correctness

Types of finite automata

(a) Deterministic Finite Automata (DFA)
(b) Non-deterministic Finite Automata (NFA or NDFA)
(c) NFA with ϵ moves (ϵ-NFA)

All finite automata are having same expressive powers.

$$L(DFA) \cong L(NFA) \cong L(\epsilon NFA)$$

where $L(DFA)$ is the class of languages accepted by DFA.

DETERMINISTIC FINITE AUTOMATA

For every input symbol of an alphabet, there is an exactly one transition from every state of the finite automata then such FA is called deterministic finite automata (DFA).

Representation or specifications of DFA:

A finite automata can be represented by 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

Q is a finite non-empty set of states
Σ is a finite non-empty set of input symbols.
δ is a transition function that maps ($Q \times \Sigma$ into Q) and usually called a direct transition function
$q_0 \in Q$ is the initial state
$F \subseteq Q$ is the set of final states or accepting states

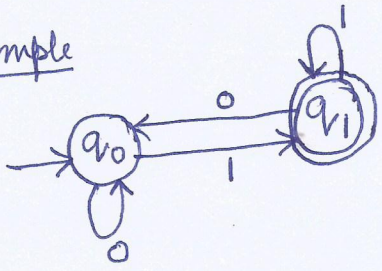
Notations for automata

- Two preferred notations for describing automata
  - Transition Diagram
  - Transition Table
- A transition diagram is a labelled directed graph in which each vertex (or node) represents a state and edges are labelled with input / output.
- A transition table, which is a tabular listing of δ function, which by implication tells us the set of states and input alphabet.

Transition Diagram

- Every state represented by circle
- For an initial state, a circle is associated with an arrow
- Final state represented by double circle.
- Every state is connected with transitions associated with an input

Example



Transition Table

- Table contains rows and columns.
- Each row corresponds to the state of FA.
- Each column corresponds to the input symbols of automata.
- Initial state is associated with an arrow
- Final is associated with a circle or star.
- Example

	0	1
$\rightarrow q_0$	q_0	q_1
(q_1)	q_0	q_1

→ or Extended Transition function.

Direct Transition Function vs. Indirect Transition function

Transition function describes the change of states during the transition.

Direct Transition Function: The transition function δ which maps $Q \times \Sigma$ into Q is called direct transition function δ

maps a state and an input symbol →

Indirect Transition Function: The transition function δ which maps $Q \times \Sigma^*$ into Q (i.e maps a state and a string of input symbols including the empty string into a state) is called the indirect transition function. δ^*

maps a state and a string including epsilon ϵ →

Properties of transition functions

P1 $\delta(q, \wedge) = q$ is a finite automata ⟹ the state of automata can be changed only by an input symbol.

P2 For all string w and input symbols a.

$$\delta(q, aw) = \delta(\delta(q,a), w)$$
$$\delta(q, wa) = \delta(\delta(q,w), a)$$

P3 For any transition function and for any two input strings x and y.

$$\delta(q, xy) = \delta(\delta(q,x), y)$$

P4 if $\delta(q,x) = \delta(q,y)$ then $\delta(q, xz) = \delta(q, yz)$ for all strings z in Σ^+

ACCEPTABILITY OF A STRING BY A FINITE AUTOMATA

A string x is accepted by a finite automata M

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

if $\delta(q_0, x) = q$ for some $q \in F$

This is basically the acceptability of a string by a final state or an accepting state.

Example $M = (Q = \{q_0, q_1\}, \Sigma = \{0, 1\}, q_0, \delta, F = \{q_1\})$

Transition function table:

	0	1
$\rightarrow q_0$	q_0	q_1
(q_1)	q_0	q_1

String to be processed:

$x = 101$

Ques: For the given machine, give the entire sequence of states for input string $x = 101$.

Ques: whether the string is accepted or not by the given automata.

↓ symbol indicates the current input symbol to be processed

$$\delta^*(q_0, \underset{\downarrow}{101})$$

$$= \delta^*(q_1, \underset{\downarrow}{01})$$

$$= \delta^*(q_0, \underset{\downarrow}{1})$$

$$= q_1 \text{ (Final state)}$$

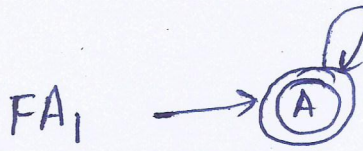
string x (101) accepted

Hence $q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_0 \xrightarrow{1} q_1$

Language Acceptance:

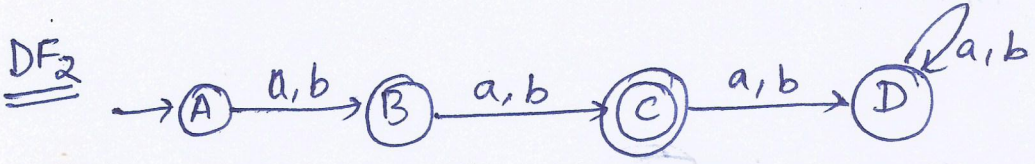
A FA is said to accept a language if all strings in the language are accepted and all strings not in the language are rejected.

For example: A $L = \{aa, ab, ba, bb\}$.



This FA_1 accepts all the strings which are in L.

But this FA_1 does not accept language L because it does not reject all the strings which are not present in L. (means it also accepts all the strings which are not in L).



DF_2 accepts a language L because it accepts all the strings which are in L and also rejects all the strings which are not in L.

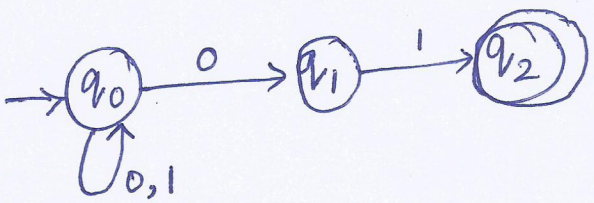
Non-deterministic Finite Automata or Non-deterministic Finite Machines

For every valid string there exists atleast one path from initial state that reaches to final state. (Every path may not reach to final state but atleast one path should exist).

NFA cover transitions for all valid strings, NFA need not cover transitions for invalid combinations.

For every valid string, NFA reaches to one of its final states with atleast one path. For invalid strings it may or may not contain path that reaches to non-final state.

Example



An NFA accepting all strings that end in 01

Representation of NDFA

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

An NFA is represented like a DFA. where

Q is a finite nonempty set of states

Σ is a finite nonempty set of input symbols.

δ is a transition function that maps

$Q \times \Sigma$ into 2^Q

2^Q ← power set of Q the set of all subsets of Q

$q_0 \in Q$ is the initial state

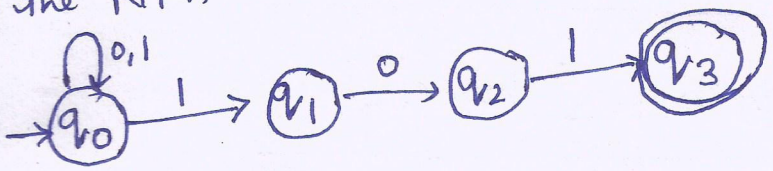
$F \subseteq Q$ is the set of final states.

DFA construction from NFA using subset construction method.

ques: construct a minimal DFA to accept the string that always ends with 101.

Ans: Language $L = \{ 0101, 1101, 00101, 01101, 10101, 11101, \ldots \}$

→ First draw the NFA



Transition Table

	0	1
→ q_0	q_0	q_0, q_1
q_1	q_2	ϕ
q_2	ϕ	q_3
(q_3)	ϕ	ϕ

$M = NFA \Rightarrow (Q, \Sigma, q_0, \delta, F)$

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

q_0 (starting state

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

$\delta \Rightarrow$ Transition Table shows.

$F \Rightarrow \{q_3\}$

Now convert this NFA into DFA using Subset construction method.

$M' = DFA \Rightarrow (Q', \Sigma, q_0', \delta', F')$

(i) $Q' = 2^Q$ (any state in Q' is denoted by $[q_1, q_2, \ldots q_j]$ where $q_1, q_2, q_3 \ldots q_j \in Q$.

(ii) $q_0' = [q_0]$

(iii) $F' \Rightarrow$ is the set of all subsets of Q containing some final state of M

(iv) $\Rightarrow \delta'([q_1, q_2, q_3 \ldots q_i], a) = \delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) \ldots \cup \delta(q_i, a)$

equivalently $\delta'([q_1, q_2, q_3 \ldots q_i], a) = [p_1, p_2, \ldots p_j]$ if and only if

$\delta(\{q_1, q_2, q_3 \ldots q_i\}, a) = [p_1, p_2, \ldots p_j]$

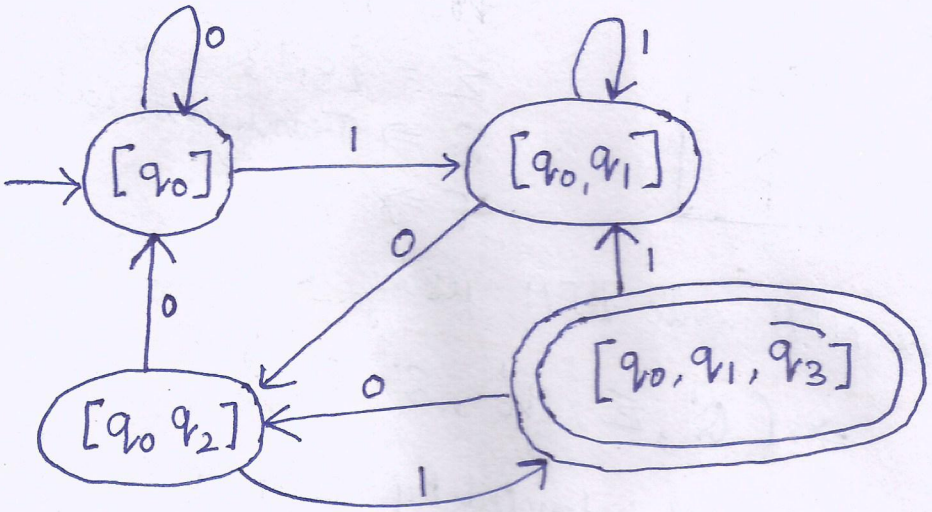
subset construction method

Transition Table for M'

	0	1
→ $[q_0]$	$[q_0]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_2]$	$[q_0, q_1]$
$[q_0, q_2]$	$[q_0]$	$[q_0, q_1, q_3]$
$[q_0, q_1, q_3]$	$[q_0, q_2]$	$[q_0, q_1]$

Explore only those states which has already occurred at right side of transition table.

DFA M' Transition Diagram:



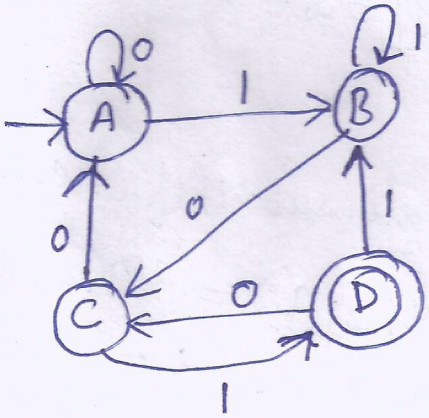
Another way to draw

$[q_0] \Rightarrow A$

$[q_0 q_1] \Rightarrow B$

$[q_0 q_2] \Rightarrow C$

$[q_0 q_1 q_3] \Rightarrow D$

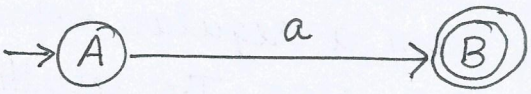


Ques: construct a minimal DFA which accepts set of all strings over {a, b} where each string starts with an 'a'.

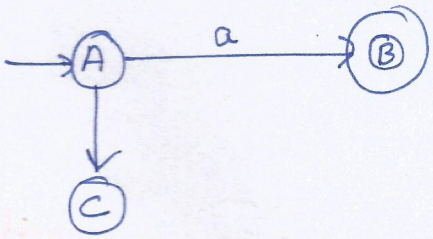
Ans First step is to find the language

$L = \{ a, aa, ab, aaa, aab, aba, abb - - - \}$

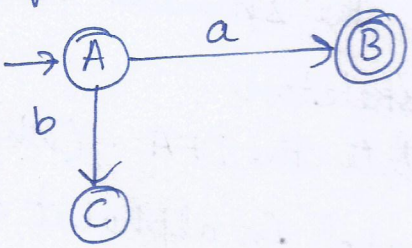
The above language is infinite, let's start with the smallest possible string and give the skeleton. This skeleton has to accept the smallest possible string.



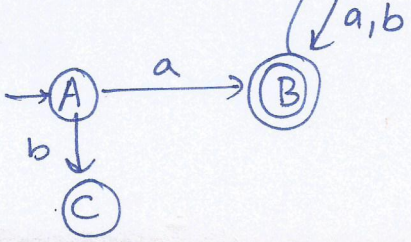
This is not DFA. Now complete it. Check state A for symbol a. We have transition. But what happens on b.
if b is in the beginning, it means a strings starts with b. then the string must not be accepted.



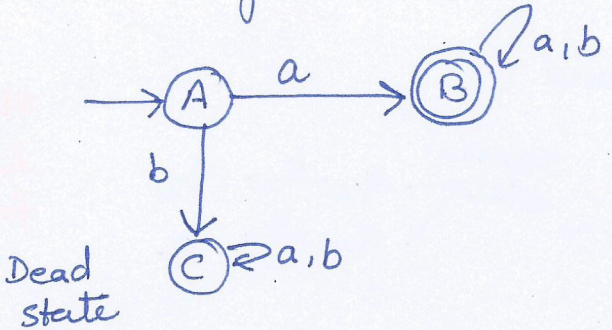
Now at b, we go to some other state C and we will not accept anything from there



Check state B. for symbols a and b.



Now state C for symbols a, b.



REGULAR EXPRESSIONS: The language accepted by finite automata are easily described by simple expressions called regular expressions.

Regular expressions has 3 operations

(i) + (union)

(ii) . concatenation

(iii) * Kleene closure

Let Σ be an alphabet. The regular expressions over Σ and the sets that they denote are defined recursively as follows:

(1) ϕ is a regular expression and denote the empty set ϕ.

(2) ϵ is a regular expression and denote the set $\{\epsilon\}$

(3) for each a in Σ, a is a regular expression and denote the set $\{a\}$.

(4) if r and s are regular expressions denoting the language R and S, respectively then

$(r+s)$ regular expression denotes the set $R \cup S$

(rs) regular expression denotes the set RS

r^* regular expression denotes the set R^*

Question: write down the regular expression for the language that accepts all strings of length 2 over $\{a, b\}$

Ans $L = \{aa, ab, ba, bb\}$

step (1) In this case first step is to write language

$L = \{aa, ab, ba, bb\}$

step (2) Then apply union on all strings i.e

$$aa + ab + ba + bb$$

Since the language is finite, hence regular expression will be the union of all strings.

$$a(a+b) + b(a+b)$$

$$\boxed{(a+b)(a+b)}$$ ← regular expression for all strings of length 2 over $\Sigma = \{a, b\}$.

Similarly RE for strings of length 3 over $\Sigma = \{a, b\}$

$$(a+b)(a+b)(a+b)$$

Question: How many number of strings of length n can be formed over an alphabet $\Sigma = \{a, b\}$

Ans: length of $\Sigma = |\Sigma| = 2$

To fill n places we have two options

$$\underset{\{a,b\}}{\_} \quad \underset{\{a,b\}}{\_} \quad \underset{\{a,b\}}{\_} \quad \_ \quad \_ \quad \_ \quad \_ \quad \cdots \quad \underset{\{a,b\}_n}{\text{upto } n}$$

So number of strings of length $n = 2^n$

or we can write it as

No. of string of length n over an $\Sigma = |\Sigma|^n$

Ques: write a regular expression for the language that contains all the string containing (every zero is followed by two ones)

$$\Rightarrow \quad 1^* + (011)^*$$

$$\Rightarrow \quad (1 + 011)^*$$

Epsilon-NFA or ∈-NFA

- ∈-NFA is same as NFA but it can include ∈ (epsilon) transitions
- For every valid string there exists a path from initial state to one of its final state.
- For invalid strings, it may or may not contain a path that reaches to non-final state.

specification of ∈-NFA

∈-NFA is a 5-tuple notation where,

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

$Q \Rightarrow$ is the set of states (finite and non-empty)

$\Sigma =$ a finite non-empty set of input symbols.

$q_0 =$ initial state or starting state $q_0 \in Q$

$F =$ a set of final states.

$\delta =$ a transition function

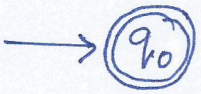
$$\delta: Q \times \Sigma \cup \{\in\} \rightarrow 2^Q$$

ϵ-NFA or NFA with Epsilon:

Formally, we define a NFA with epsilon to be a quintuple $(Q, \Sigma, q_0, \delta, F)$ with all components as before but δ, the transition function, maps $Q \times (\Sigma \cup \{\epsilon\}) \longrightarrow 2^Q$.

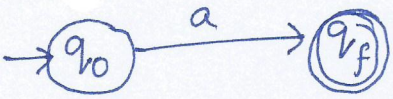
ϵ NFA from regular expression r:-

(i) $r = \epsilon$

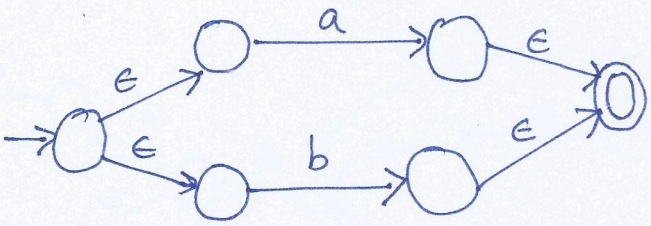


(ii) $r = \phi$

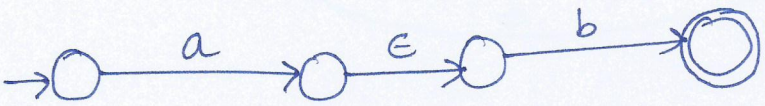
(iii) $r = a$



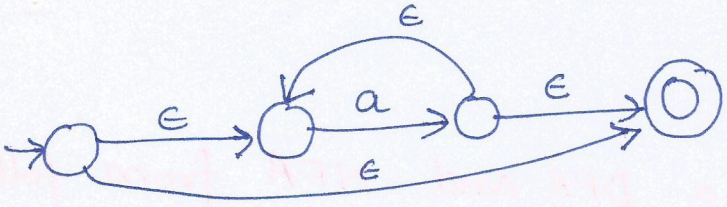
(iv) $r = (a+b)$



(v) $r = ab$



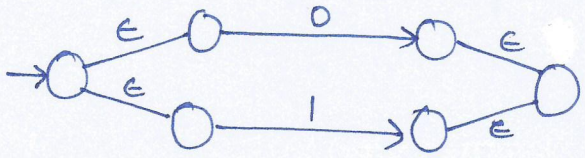
(vi) $r = a^*$



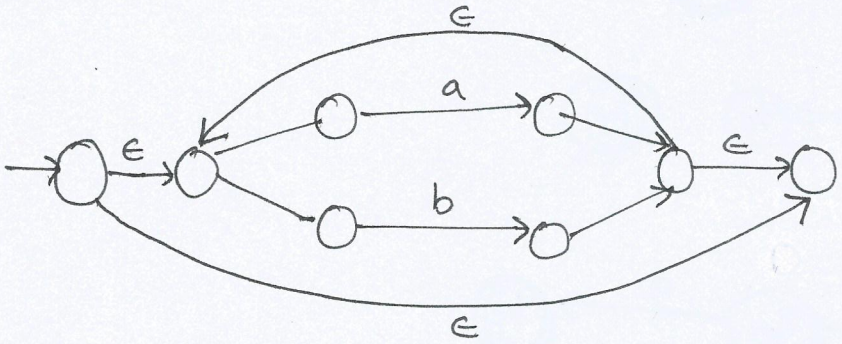
Ques: construct ∈NFA from the following regular expression.

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

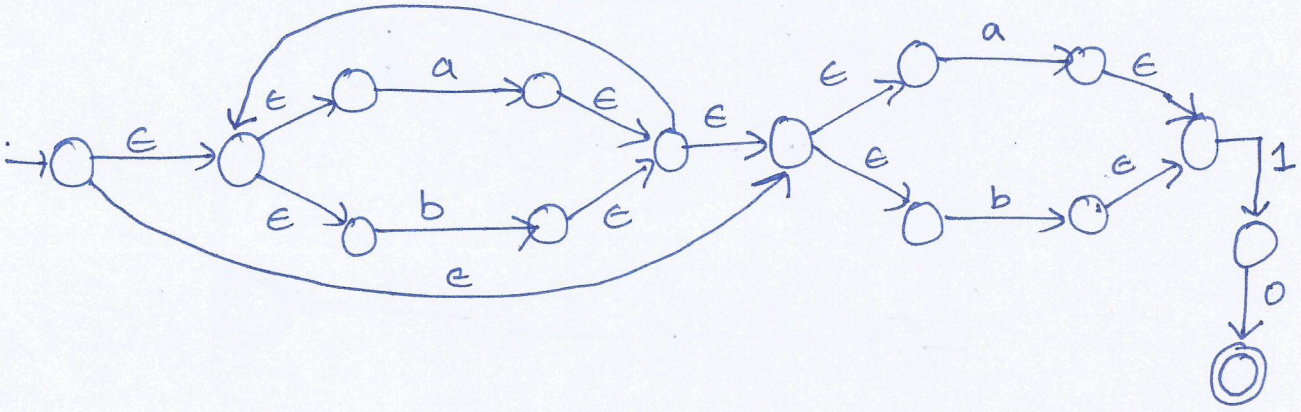
$r = (0+1)$



$r = (0+1)^*$



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



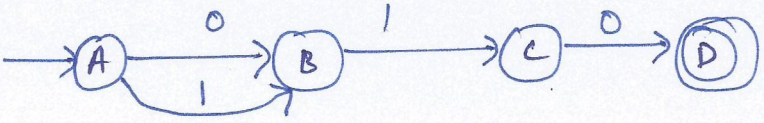
Ques: create a DFA and NFA from following regular expression

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

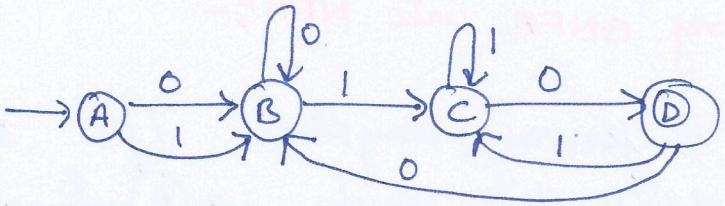
create DFA → ① First identify the language and find smallest possible string

$$L = \{010, 110, 0010, 1010 - - - -\}$$

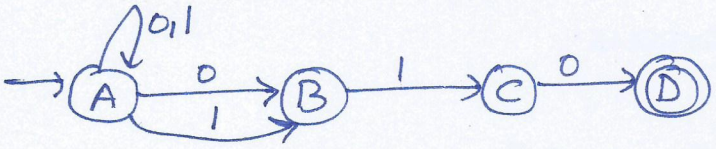
② Draw skeleton



③ Now complete DFA.



Create NFA from regular expression



Conversion of NFA to DFA : [Subset Construction Method]

convert following NFA into DFA

	0	1
→ A	A, B	A, B
B	ϕ	C
C	D	ϕ
(D)	ϕ	ϕ

NFA Transition Table

states/Σ	0	1
→ A	[AB]	[AB]
[AB]	[AB]	[ABC]
[ABC]	[ABD]	[ABC]
([ABD])	[AB]	[ABC]

DFA Transition Table

Algo : Conversion from NFA to DFA

① start with starting state

② Combine multiple states into a single state

③ Explore states only getting on RHS.

④ For transition from a combined state do the union of states.
For example state [AB]
transition on 0 from [AB] we do the union of transition from A on zero and transition from B on zero 0.

Applications and Limitations of FA

Applications:

- FA used for designing lexical analyzer of a compiler.
- FA used for recognizing pattern using regular expressions.
- FA used in text editors.
- FA used for implementing spell checker
- FA used for designing sequential circuits (Transducers using mealy and Moore-machines)

Limitations

- The input tape is read only.
- The memory it has, only in terms of states.
- It is most restrictive model.
- The FA can not recognize following languages:
  - a set of strings over balanced parenthesis
  - $L = \{a^n b^n \mid n \geq 1\}$
  - a set of strings of binary having equal number of 1's and 0's.

Equivalence of NFA and DFA

- **Theorem:** For every NFA N there exists an equivalent DFA M or vice-versa.

- **Proof:** Two ways to prove this theorem.

  - **Proof by construction**
    - Given an arbitrary NFA N, construct an equivalent DFA M.
    - Subset Construction method of NFA to DFA conversion.
  - **Proof by Induction** NFA N accepts a string w if and only if DFA M accepts w.

Proof by construction:

subset construction method already discussed in Notes.

Proof by Induction:

- we are going to prove by induction that the DFA M obtained from NFA N by the conversion algorithm accepts the same language

Let $N = (Q_1, \Sigma, q_{1,0}, \delta_1, A_1)$

$M = (Q_2, \Sigma, q_{2,0}, \delta_2, A_2)$

- To prove for any string w, $\delta_1^*(q_{1,0}, w) = \delta_2^*(q_{2,0}, w)$
- **Prove by induction**
  - **Basic step** $w = \epsilon$

$\delta_2^*(q_{2,0}, \epsilon) = q_{2,0}$ $\Rightarrow$ starting state of M, DFA

$= \{q_{1,0}\}$ $\Rightarrow$ by construction method, starting state of DFA will same as in NFA.

$= \delta_1^*(q_{1,0}, \epsilon)$ $\Rightarrow$ by definition of δ_1^*

  - **inductive step:** Assume that $\delta_1^*(q_{1,0}, w) = \delta_2^*(q_{2,0}, w)$ for any arbitrary string w. ——— Induction hypothesis

- **For any string w and any symbol a in Σ.**

$$\delta_1^*(q_{1,0}, wa) = \bigcup_{p \in \delta_1^*(q_{1,0}, w)} \delta_1(p, a)$$

union: There exists atleast one path for w from starting state $q_{1,0}$

$$= \delta_2(\delta_1^*(q_{1,0}, w), a)$$

$$= \delta_2(\delta_2^*(q_{2,0}, w), a)$$

$$= \delta_2^*(q_{2,0}, wa)$$

Thus for any string w, $\delta_1^*(q_{1,0}, w) = \delta_2^*(q_{2,0}, w)$ holds.

Conversion of ϵ-NFA into NFA

ϵ-closure (state)

- ϵ-closure or ϵ^* of any state S is the set of all states which can be reachable from state S only on ϵ transitions.
- It is a recursive process.

Initial state q_0'

ϵ-closure of initial state q_0 in NFA.

$$q_0' = \epsilon\text{-closure}(q_0)$$

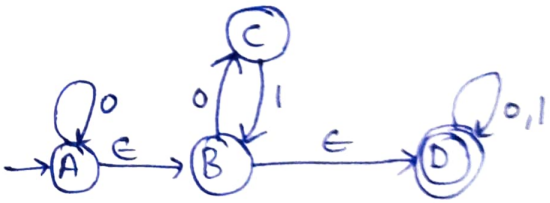
Transition δ'

Transition on from state q_i' on symbol a can be defined as

$$\epsilon\text{-closure}(\epsilon\text{-closure}(S), a)$$

Example

Convert the following ϵNF into NFA



How find out the epsilon-closure of each state

ϵ-closure(A) = {A, B, D}

$A \xrightarrow{\epsilon} A$ → Always reach A from A.

$A \xrightarrow{\epsilon} B$

$A \xrightarrow{\epsilon} B \xrightarrow{\epsilon} D$

ϵ-closure(B) = {B, D}

$B \xrightarrow{\epsilon} B$

$B \xrightarrow{\epsilon} D$

ϵ-closure(C) = {C}

$C \xrightarrow{\epsilon} C$

ϵ-closure(D) = {D}

$D \xrightarrow{\epsilon} D$

Now find out transitions from each state on each symbol

- For this first take the ϵ-closure of state, then find the transition on symbol from each state in ϵ-closure, then again find the ϵ-closure of each state

state A: ϵ^* → A; 0 → A; ϵ^* → A, B, D

B: 0 → C; ϵ^* → C

D: 0 → D; ϵ^* → D

union of states ⟹ {A, B, C, D}

state A: ϵ^* → A; 1 (symbol) → ϕ; ϵ^* → ϕ

B: 1 → ϕ; ϵ^* → ϕ

D: 1 → D; ϵ^* → D

union ⟹ {D}

state B: ϵ^* → B; 0 → C; ϵ^* → C

D: 0 → D; ϵ^* → D

union ⟹ {C, D}

state B: ϵ^* → B; 1 → ϕ; ϵ^* → ϕ

D: 1 → D; ϵ^* → D

⟹ {D}

state C: ϵ^* → C; 0 → ϕ; ϵ^* → ϕ ⟹ ϕ

state C: ϵ^* → C; 1 → B; ϵ^* → {B, D} ⟹ {B, D}

state D: ϵ^* → D; 0 → D; ϵ^* → {D}

state D: ϵ^* → D; 1 → D; ϵ^* → {D} ⟹ {D}

Transition Table for NFA

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

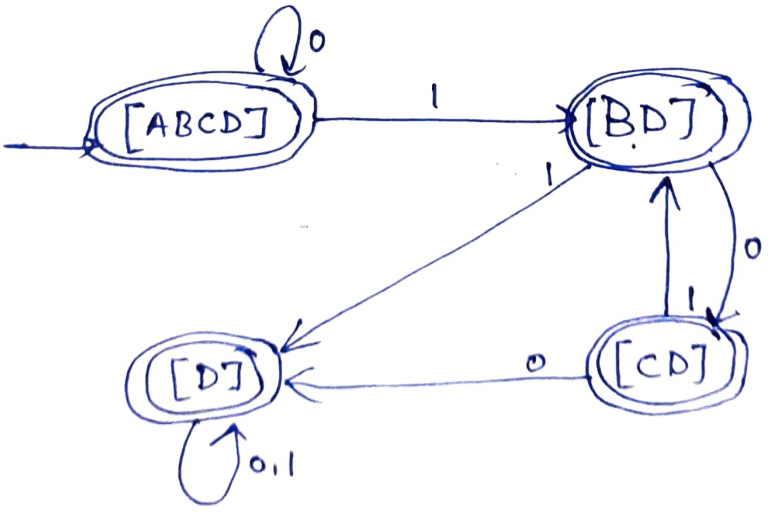
initial state
ϵ-closure (A)
= {A, B, C, D}

Conversion NFA to DFA

Transition Table for DFA

	0	1
→[ABCD]	[ABCD]	[BD]
[BD]	[CD]	[D]
[CD]	[D]	[BD]
[D]	[D]	[D]

DFA



Prof. Minakshi

Prof. Minakshi

Minimization of DFA

Example ①

Transition Table

PS	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_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

Step 1 To find π_0 (0-Equivalent classes)

separate final and non-final states.

$\pi_0 = \{ \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}, \{q_2\} \}$

Non final, Final

- Let $S_1 = \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}$

  $S_2 = \{q_2\}$

Step 2 To find π_1 (1-equivalent classes)

~~Transition~~ Table

	0	set belong	1	set belong
q_0	q_1	S1	q_5	S1
q_1	q_6	S1	q_2	S2
q_2	q_0	S1	q_2	S2
q_3	q_2	S2	q_6	S1
q_4	q_7	S1	q_5	S1
q_5	q_2	S2	q_6	S1
q_6	q_6	S1	q_4	S1
q_7	q_6	S1	q_2	S2

Group all the state having same row for set.

$\{q_0, q_4, q_6\} \Rightarrow$ S1 S1

$\{q_1, q_7\} \Rightarrow$ S1 S2

$\{q_3, q_5\} \Rightarrow$ S2 S1

No need to check for $\{q_2\}$

~~$\pi_1 = \{ \{q_2\}, \{q_1, q_7\}, \{q_3, q_5\} \{q_0, q_4, q_6\} \}$~~

$\pi_1 = \{ \{q_0, q_4, q_6\}, \{q_1, q_7\} \{q_3, q_5\}, \{q_2\} \}$

Let $G_{11} = \{q_0, q_4, q_6\}$ $G_{13} = \{q_3 q_5\}$,

$G_{12} = \{q_1, q_7\}$ $G_{14} = \{q_2\}$

Step3: To find π_2 (2-Equivalent classes)

Table

	0		1	
→ q_0	q_1	G_{12}	q_5	G_{13}
q_1	q_6	G_{11}	q_2	G_{14}
q_2	q_0	G_{11}	q_2	G_{14}
q_3	q_2	G_{14}	q_6	G_{11}
q_4	q_7	G_{12}	q_5	G_{13}
q_5	q_2	G_{14}	q_6	G_{11}
q_6	q_6	G_{11}	q_4	G_{11}
q_7	q_6	G_{11}	q_2	G_{14}

Group all states for same row for sets.

check for $\{q_0 q_4 q_6\}$

$\{q_0 q_4\} \Rightarrow G_{12} G_{13}$

$\{q_6\} \Rightarrow G_{11} G_{11}$

check for $\{q_1, q_7\} \Rightarrow G_{11} G_{12}$

$\{q_3 q_5\} \Rightarrow G_{14} G_{11}$

$\pi_2 = \{ \{q_0, q_4\}, \{q_1, q_7\}, \{q_3, q_5\}, \{q_6\} \{q_2\} \}$

Step4: To find π_3 (3-Equivalent class) Let $H_1 = \{q_0, q_4\}$, $H_2 = \{q_1, q_7\}$

$H_3 = \{q_3, q_5\}$, $H_4 = \{q_6\}$

$H_5 = \{q_2\}$

Table

	0		1	
q_0	q_1	H_2	q_5	H_3
q_1	q_6	H_4	q_2	H_5
q_2	q_0	H_1	q_2	H_5
q_3	q_2	H_5	q_6	H_4
q_4	q_7	H_2	q_5	H_3
q_5	q_2	H_5	q_6	H_4
q_6	q_6	H_4	q_4	H_1
q_7	q_6	H_4	q_2	H_5

$\{q_0 q_4\} \Rightarrow H_2 H_3$

$\{q_1 q_7\} \Rightarrow H_4 H_5$

$\{q_3 q_5\} \Rightarrow H_5 H_4$

$\{q_6\}$

$\{q_2\}$

$\pi_3 = \{ \{q_0, q_4\}, \{q_1, q_7\}, \{q_3, q_5\}, \{q_6\}, \{q_2\} \}$

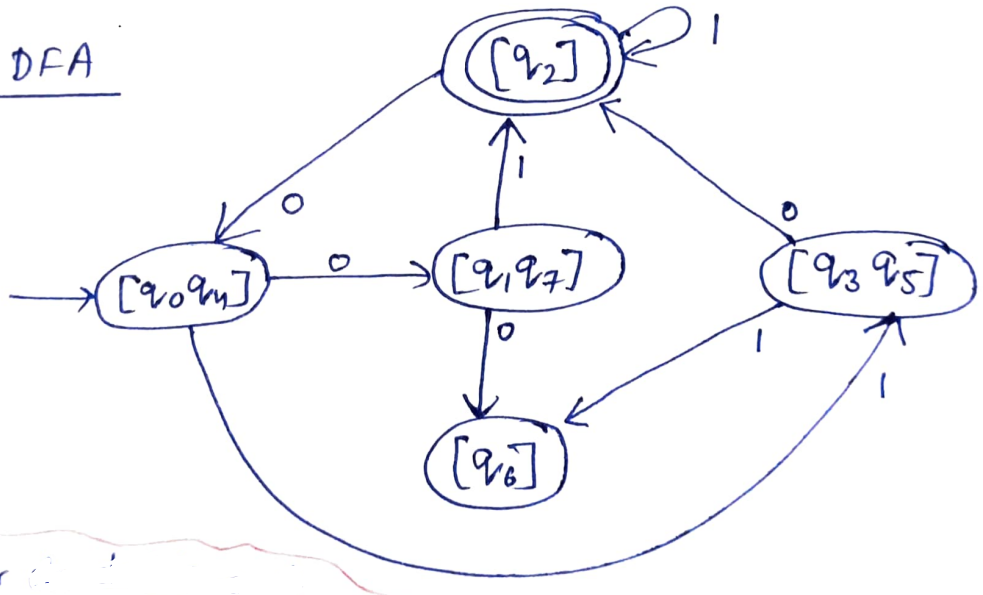
$\pi_3 = \pi_2$ stop the process

Minimized DFA $M = (Q', \Sigma, q_0', \delta', F')$

δ' Transition function defined as

	0	1
$\rightarrow [q_0 q_4]$	$[q_1 q_7]$	$[q_3 q_5]$
$[q_1 q_7]$	$[q_6]$	$[q_2]$
$[q_3 q_5]$	$[q_2]$	$[q_6]$
$[q_6]$	$[q_6]$	$[q_0 q_4]$
$(\![q_2]\!)$	$[q_0 q_4]$	$[q_2]$

Minimized DFA

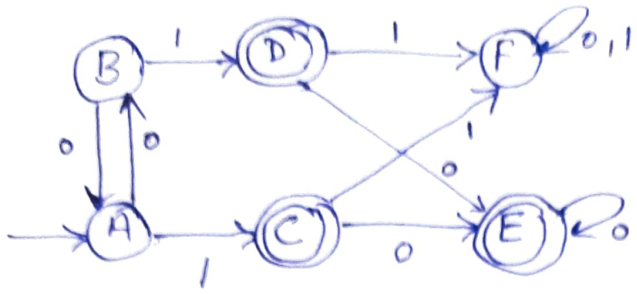


Note check for unreachable state in DFA

if unreachable state is present, remove it from the minimized DFA.

Prof. Menakshi

Example 2 Minimization of DFA



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

Step 1 To find π_0

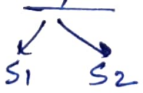
To find 0-equivalent classes:

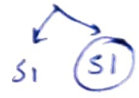
{A, B, F}, {C, D, E}
non-final final

Step 2 Let S_1 = {A, B, F}
S_2 = {C, D, E}

	0	1
A	B S_1	C S_2
B	A S_1	D S_2
C	E S_2	F S_1
D	E S_2	F S_1
E	E S_2	F S_1
F	F S_1	F S_1

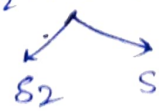
A, B ⇒ similar.





{A, B, F} ⟹ {A, B} {F}.

{C, D, E} ⟹ {C, D, E}.



π_1 = {{A, B}, {F}, {C, D, E}}
S_1 S_2 S_3

To find π_2

	0	1
A	B (S1)	C (S3)
B	A (S1)	D (S3)
C	E (S3)	F (S2)
D	E (S3)	F (S2)
E	E (S3	F (S2)
F	F (S2)	F (S2)

π_2 = {{A, B}, {F}, {C, D, F}}

$\pi_1 = \pi_2$.

Minimized DFA M' $(Q', \Sigma, q_0', \delta', F')$

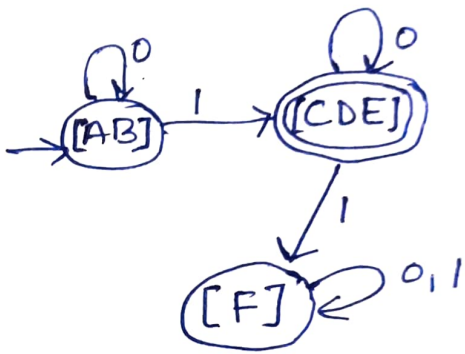
- $Q' = \{ [A,B], [F], [C,D,E] \}$

$q_0' \Rightarrow [AB]$

$F' \Rightarrow [CDE]$

δ' is defined by the table

	0	1
→ [AB]	[AB]	[CDE]
[CDE]	[CDE]	[F]
[F]	[F]	[F]



Prof. Minakshi

Myhill Nerode Theorem:

Indistinguishable strings

- Let L be any language over Σ^*.
- strings x and y in Σ^* are indistinguishable by L if for every string $z \in \Sigma^*$

  either both xz and yz are in L (both reaching final states)

  or both xz and yz are not in L (both reaching non-final states)

we write $x \equiv_L y$ — x is equivalent to y.

↑ equivalence relation

Distinguishable strings

- Let L be any language over Σ^*
- two strings x and y are distinguishable in L if there exists z such that exactly one of xz and yz in L.

one is reaching to final state — another is reaching to non-final state.

Myhill Nerode Theorem:

Language L is regular if and only if $\equiv_L$ has a finite number of equivalent classes.
Furthermore, there is a DFA $M = (Q, \Sigma, q_0, \delta, F)$ with $L(M) = L$ having precisely one state for each equivalence class of $\equiv_L$.

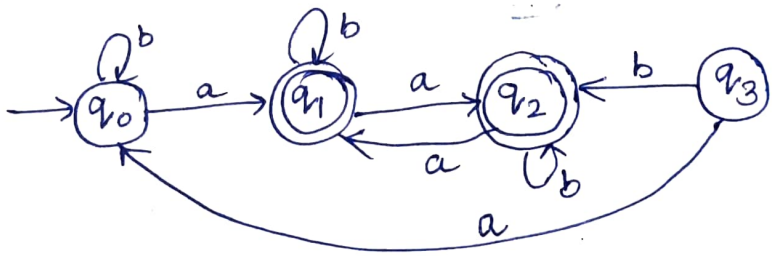
- Myhill Nerode theorem says ⇒ A regular language always has finite number of equivalent classes.
  ⇒ There always exists a DFA that accepts L.

DFA Minimization using Myhill Narode Theorem

Algorithm:

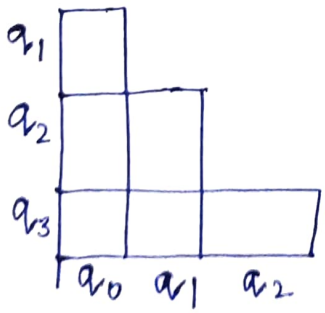
1. Draw a table for all pair of states (Q_i, Q_j)
2. All cells are initially unmarked
3. For each state pair (Q_i, Q_j):
   - for each input symbol 'a':
     - if both states Q_i and Q_j go to either final states or non-final states.
     - then put '✓' in the cell.
     - otherwise put 'X' in the cell.
4. Repeat until all the pairs got marked.
5. Now pair of states can be combined if they have common states.

Example



How to draw table

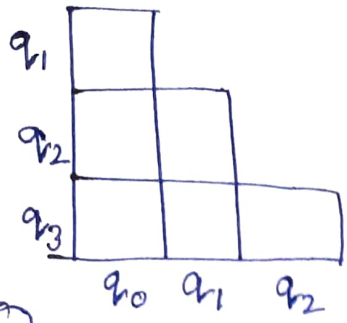
step1



step2

All pairs of state are unmarked.

step3 start marking of pair of states for each input symbol.



(1) check pair q_0, q_1

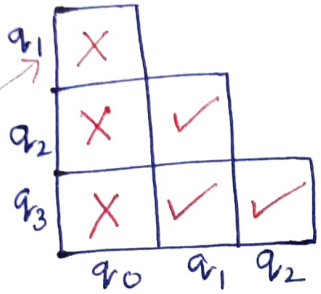
$q_0 \xrightarrow{a} q_1 \in F$

$q_1 \xrightarrow{a} q_2 \in F$

$q_0 \xrightarrow{b} q_0 \notin F$

$q_1 \xrightarrow{b} q_1 \in F$

put X in (q_0, q_1) cell.



(2) check pair $q_0\ q_2$

$q_0 \xrightarrow{a} q_1 \in F$

$q_2 \xrightarrow{a} q_1 \in F$

$q_0 \xrightarrow{b} q_0 \notin F$

$q_2 \xrightarrow{b} q_2 \in F$

put X in $q_0\ q_2$ cell.

(3) check $q_0\ q_3$

$q_0 \xrightarrow{a} q_1 \in F$

$q_3 \xrightarrow{a} q_1 \in F$

$q_0 \xrightarrow{b} q_0 \notin F$

$q_3 \xrightarrow{b} q_2 \in F$

put X in $q_0\ q_3$

(4) check $q_1\ q_2$

$q_1 \xrightarrow{a} q_2 \in F$

$q_2 \xrightarrow{a} q_1 \in F$

$q_1 \xrightarrow{b} q_1 \in F$

$q_2 \xrightarrow{b} q_2 \in F$

put ✓ in $q_1\ q_2$

(5) check $q_1\ q_3$

$q_1 \xrightarrow{a} q_2 \in F$

$q_3 \xrightarrow{a} q_1 \in F$

$q_1 \xrightarrow{b} q_1 \in F$

$q_3 \xrightarrow{b} q_2 \in F$

put ✓ $q_1\ q_3$

(6) check $q_2\ q_3$

$q_2 \xrightarrow{a} q_1 \in F$

$q_3 \xrightarrow{a} q_1 \in F$

$q_2 \xrightarrow{b} q_2 \in F$

$q_3 \xrightarrow{b} q_2 \in F$

put ✓ in $q_2\ q_3$

pairs (q_1q_2) (q_1q_3) (q_2q_3) have ✓ in their cells.

Step combine states pair if they have common states.

all pairs have common states:-

$(q_1,q_2), (q_1,q_3); (q_2,q_3) \Rightarrow \{q_1, q_2, q_3\}$

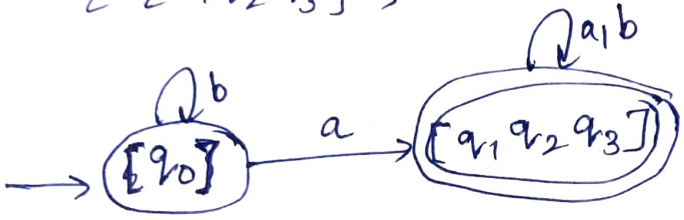
Minimized DFA

$M' = (Q', \Sigma, q_0', \delta', F')$ $\quad q_0' = \{q_0\}$

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

$\Sigma = \{a,b\}$ $\quad q_0' \Rightarrow \{q_0\}$.

$F' = \{\{q_1q_2q_3\}\}$



δ' defined by:

	a	b
$\rightarrow[q_0]$	$[q_1q_2q_3]$	$[q_0]$
$[q_1q_2q_3]$	$[q_1q_2q_3]$	$[q_1q_2q_3]$

Applications of Myhill Nerode Theorem:

Myhill Nerode theorem is used for

- To prove that a certain language is regular or not.
- To find the minimum number of states in DFA (minimization of DFA).

Prof. Minakshi

FINITE AUTOMATA WITH OUTPUTS or MEALY and MOORE MACHIN

- Mealy and Moore machines are output generatorss.
- Moore and mealy machines are deterministic finite automata with output. But there is no final state concept in moore and mealy machines.

Moore machine

- 6-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

where

Q is a finite set of states

Σ is the input alphabet

Δ is the output alphabet

δ is the transition function which maps $Q \times \Sigma$ into Q

λ is the output function mapping Q into Δ

q_0 is the initial state

Mealy machine

6-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0$

where

Q is a finite set of states

Σ is the input alphabet

Δ is the output alphabet

δ is the transition function which maps $Q \times \Sigma$ into Q

λ is the output function mapping $Q \times \Sigma$ into Δ

q_0 is the initial state.

Moore Machine: - if output symbol is associated with each state of automata, then such automata is called as moore machine.

- For every input string, it generates an equivalent output string.

if length of input string = n

then length of output string = $n+1$

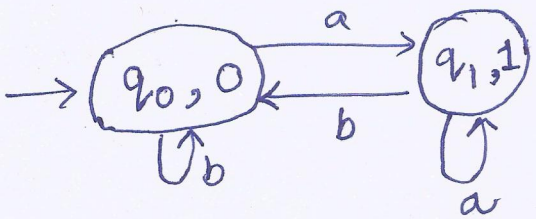
Because moore machine generates the output associated with initial state without reading an input symbol.

Example

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

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

$Q = \{q_0, q_1\}$



Moore machine that generates output '1' for every occurrence of 'a' in the input string otherwise it generates output '0'

Present state	next state		output
	a	b	λ
→q_0	q_1	q_0	0
q_1	q_1	q_0	1

Ques: For string aabb show the output string

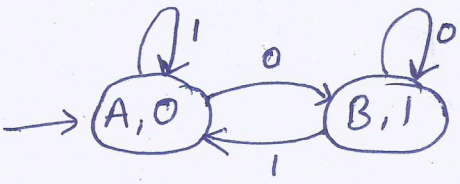
Transition : $q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_0 \xrightarrow{b} q_0$

Output : 0 1 1 0 0

Length of input string $|aabb| = 4 \Longrightarrow n$

Length of output string $|01100| = 5 \Longrightarrow n+1$

Ques: construct a moore machine that produces one's complement of a binary number.



For complement

input	output
0	1
1	0

PS	Next state		O/P
	0	1	
→A	B	A	0
B	B	A	1

input string 0011

Transition : $A \xrightarrow{0} B \xrightarrow{0} B \xrightarrow{1} A \xrightarrow{1} A$

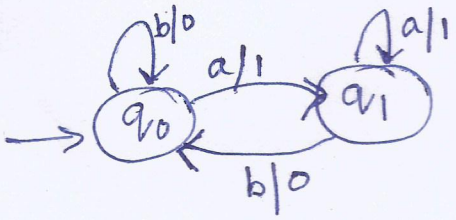
Output : 0 1 1 0 0

Note: Read the input left to right
Read the output right to left.

Mealy Machine:

- if output symbol is associated with the transition (state and input) of finite state machine, then such finite automata is called mealy machine.
- it covers the transitions for all strings over given alphabet
- For every input string, it generates the equivalent output string.
- if length of input string = n
  then length of output string = n

construct a mealy machine that generates output 1 for every occurrence of a in the input string otherwise it generates output 0.



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

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

$Q = \{q_0, q_1\}$

Table Representation

Present state	a=		b.	
	Next state	output	Next state	output
$\rightarrow q_0$	q_1	1	q_0	0
q_1	q_1	1	q_0	0

Ques: For string aabb show the output string

Transition $q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_0 \xrightarrow{b} q_0$

Output 1 1 0 0

length of string = |aabb| = 4

length of output string |1100| = 4

EQUIVALENCE OF MOORE AND MEALY MACHINES

- Every moore machine can be converted into equivalent mealy machine
- Every mealy machine can be converted into equivalent moore machine
- Moore and mealy machines are equivalent

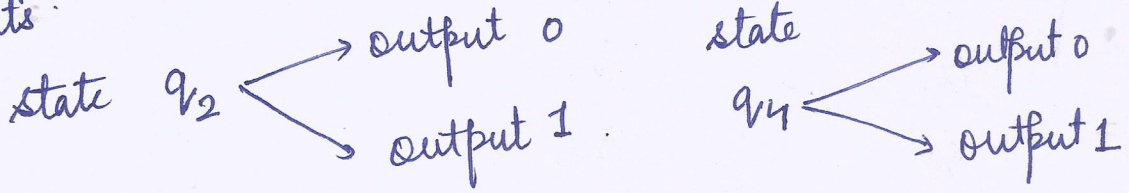
Transforming a mealy machine into equivalent moore machine

- For any state q_i in next state column, determine the number of different outputs associated with q_i
- Split q_i into several different states, the number of such states being equal to number of different outputs

Example Given mealy machine

PS	input $a=0$ NS	output	input $a=1$ NS	output
$\rightarrow q_1$	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

- First identify the states under next state column with different outputs



- split the state into different states ⇒ based on number of different outputs

New state table

Present State	Next state: input a=0 state	Next state: input a=0 output	Next state: input a=1 state	Next state: input a=1 output
→ q_1	q_3	0	q_{20}	0
q_{20}	q_1	1	q_{40}	0
q_{21}	q_1	1	q_{40}	0
q_3	q_{21}	1	q_1	1
q_{40}	q_{41}	1	q_3	0
q_{41}	q_{41}	1	q_3	0

The pair of states and outputs can be rearranged as following to construct moore machine.

Present state	Next state: a=0 state	Next state: a=1 state	Output
→ q_1	q_3	q_{20}	1
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_3	q_{21}	q_1	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

write here the associated output of present state

Converting a moore machine into its equivalent mealy machine

- we have to define the output function λ' for mealy machine as a function of the present state and the input symbol.

$$\lambda'(q,a) = \lambda(\delta(q,a)) \text{ for all state } q \text{ and input } a.$$

(λ: output of; $\delta(q,a)$: next state)

Example construct a mealy machine equivalent to the given moore machine.

present state	Next state a=0	Next state a=1	output
→ q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

mealy machine

Present state	Next state: state (a=0)	output (a=0)	state (a=1)	output (a=1)
q_0	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_0	0

Note we can reduce the number of states in any model by considering states with identical transitions. if two states have identical transition (rows) then delete one of them and do the proper replacement