UNIT-1. INTRODUCTION TO FINITA AUTOMATA

Topics to be covered :

Introduction to Finite Automata : *Structural Representation, The Central Concept of Automata Theory - Alphabet, String & langauges. Deterministic Finite Automata(DFA), Non-Deterministic Finite Automata(NFA), and Equivalance of NFA and DFA , Finite Automata with Epsilon (Ɛ) Transitions.*

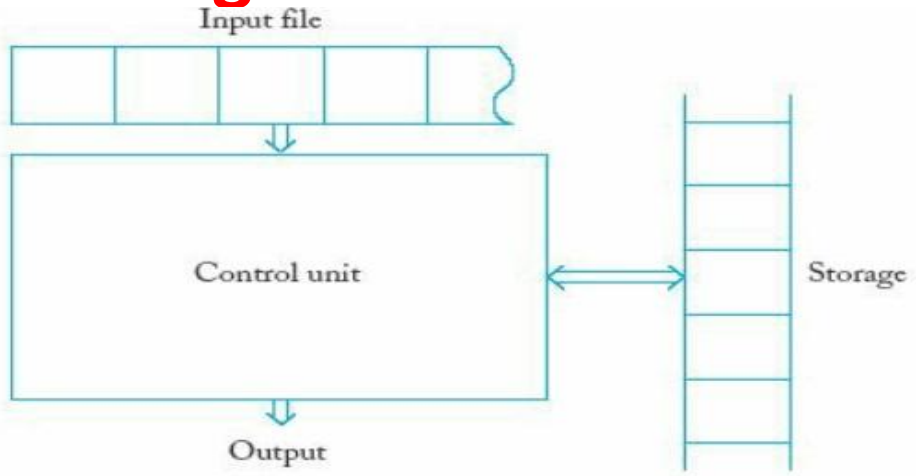
Upon completion you will be able to

- **Explain the concepts of Auotomata Theory and its Applications.**
- **Design DFA, NFA and *Ɛ-NFA for Regular languages***
- ***Demonstrate the equivalence of* DFA, NFA and *Ɛ-NFA*s.**

Deterministic Finite Automata(DFA)

- Structural /Schematic Representation and working of a Finite Automata.
- Types of Automata
  - Deterministic and Non Deterministic
- The cental concept of Automata theory
- Formal Definition of Deterministic Finite Automata-DFA
- Processing of a string by DFA
- Extended Transition Function δ^*
- Languages of an DFA
- Examples

Structural/Schematic Representation and working of a Finite Automata.



Automata has the follwing essential features :

1. It has a mechanism for reading the input that is assumed to be string over a given alphabet and is strored in a **input file or Input beffer** which the automata can read one character at a time but cannot write. The input file is divided into 'n' number of cell, where each cell is cable of holding the single character. The input mecahnism can read the input file from left to right one character at a time. It can also detect the end of input.
2. The Automata can produce an **Output** of some form.
3. The Automata may have a temporary **Storage** device **consisting of an unlimited number of cells, each Cell is capable of holding a single character from an alphabet. The Automata can read and change the contents of the storage cells**

4. Finally Automata has a **Control Unit,** which can be in any one of a finite number of internal states, and can change state in some defined manner.

Working of Automata :

An Automata is assumed to operate in discrete manner of timeframe. At any given instance of time, the **control unit** is in some internal state and the input mechanism is scanning the perticular input symbol on the **input file.** The internal state of the **control unit** at next time step is determined by the **transision function - δ.**

The **Transision function - δ** gives the next state interms of the **current state and current input symbol and the information currently in the temporary storage**. During the transition from one state to next state the **output** may be produced or the information in the temporary staorage changed.

The term **Configuaration** will be used to refer to perticular state of the **control unit, input file** and **temporary storage**. The transition of the automata from configuration to another will be called as **MOVE.**

NOTE : This genaral model covers all the automatra like PDA, Turing Machine. The finite state control will be common to all specific cases, however the diffrence will arise from the way the output can be produced and the nature of the temporary storage. The power of differnt types of automata is governed by nature of Temporary storage.

Central concept of automata theory

- **Alphabets - ∑**
  - An **alphabet** is a finite, non empty set of symbols. ∑- symbol is used denote the same.

    Ex-1. ∑ = {0, 1}, the binary alphabet

    Ex-2. ∑ ={a, b,.....z}, the set of all lower case letters

- **Strings**
  - A **string** (referred as word) is finite sequence of symbols choosen from some alphabet.

    Ex-1. if ∑ = {0, 1}, is the binary alphabet then 011011 is string from binary alphabet

  - The **Empty string** is a string with zero occurance of symbols from the alphabet. This is denoted by **'Ɛ'** which can be choosen from any alphabet-∑

- Power of an alphabet - $\sum k$ - If $\sum$ is an alphabet, then power of an alphabet is set of strings of certain length from alphabet by an exponential notation. i.e We define $\sum^k$ to be the set of strings of length k, each of whose symbol is in $\sum$.

Ex-1 if $\sum$ = {0,1}, then

1. **$\sum^1$= {0,1} is the set of strings of length 1.**
2. **$\sum^2$= {00,01,10,11} is the set of strings of length 2. and so on.... Note that $\sum^0$= Ɛ is string of length 0 no matter what the alphabet -$\sum$ is**
3. **The set of all strings of any length is conventionaly denoted by $\sum^*$ = {Ɛ, 0,1,00,01,10,11, 000,010,.....}. This can also be denoted as follows $\sum^* = \sum^0 \cup \sum^1 \cup \sum^2 \cup \sum^3 \cup$.....**
4. **$\sum^+ = \sum^1 \cup \sum^2 \cup \sum^3 \cup$..... This excludes the empty string - Ɛ**
5. **$\sum^*$ can also be written as - $\sum^* = \sum^+ \cup \{ Ɛ \}$**

– **Concatenation of string -** Let **x** and **y** be **strings**. Then **xy** denotes the **concatenation** of **string x** and **y**, that is **string formed** by making a **copy of x** and followed by **copy of y**.

Ex-1. Let x = 1101 and y = 0001.Then xy = 11010001

- **Languages**
  - A set of strings all of which are choosen from $\sum^*$, where $\sum$ is a perticluar alphabet is called a **Language**. If $\sum$ is an alphabet and $L \subseteq \sum^*$, then L is a **language** over $\sum$.
    - **Note that the language over $\sum$ need not include strings with all the symbols of $\sum$**

Types of Automata

- Depending on how the **Transision function - δ** is defined there are two types of Autromata.

1. **Determinstic Finite Automata :**

A deteministric automata is one in which each move is uniquely determinined by the current configuaration. i.e A **finite automata** is said to be **deterministic** iff, for **any state 's'** and for **any input symbol 'a'** the **Transision function - δ** is uniquely defined. i.e next state can be easily determined

Types of Automata

2. **Non Determinstic Finite Automata :**

A **non deteministric automata** is one in which each move is not uniquely determinined by the current configuaration. i.e A **finite automata** is said to be **non deterministic** iff, for **any state 's'** and for **any input symbol 'a'** the **Transision function - δ** is not uniquely defined. i.e next state cannot be easily determined.

- **DFA - Deterministic Finite Automata**

Prepared by Dr. Mallikarjun Math Professor
& Head CC

Formal Definition

- The Deterministic Automata (DFA) is defined as 5-tuple or quintuple indicating five components:

$$D = \{ Q, \sum, \delta, q_0, F\}$$

Where:

1. Q is a finite set of states.
2. $\sum$ is a finite set of input symbols.
3. δ is the transition function that takes a state in Q and an input symbol in $\sum$ as arguments and returns a state

$$\delta : Q \times \sum \rightarrow Q$$

4. q_0 is the start state that belongs to Q
5. F is a subset of Q, is the set of final (or accepting) states.

Processing of strings by DFA

- Suppose **w=a1,a2,….an** is the string to be processed then DFA starts with **start state q0** and **a1 the first character** and consults the **transition function δ** to find the **next state** that DFA stays after processing the **input symbol a1.**
- This is continued for all the subsequent **input characters from a2,a3,….an** and current state to find next state by consulting **transition function δ** and finally DFA stays in a **state q.**
- and if a **state q** is a **final state** then the **input w** is **accepted, otherwise rejected**.

Simpler Notations of DFA

- There are two types of notations to represent DFA
  1. **Transition Diagram**
  2. **Trantion Table**

1. Transition Diagram : A **transition diagram** for DFA A ={ Q, ∑, δ, q0, F} is a directed graph and defined as follows :

a. For each state in Q there is node and nodes are denoted by Circle

b. For each state ‘q’ in Q and for each input symbol ‘a’ in ∑, let δ(q, a) = p then transition diagram must contain an arc or edge from state ‘q’ to state ‘p’ with labelled ‘a’.

c. there is an arrow into the start state q0 labelled START

d. Node corresponding to accepting states (Those are in F) are marked by double circle other states have a single circle

Simpler Notations of DFA

2. A transition table : A transition table for DFA A ={ Q, ∑, δ, q0, F} is a conventional tabular representation of transition function δ, where

- **Rows** corresponds to the states in Q for the transition table.
- **Columns** corresponds to the **inputs from** ∑.
- The entry for the row corresponding to **state 'q'** and the column corresponding to **input 'a'** is the **state 'p' from δ(q, a)**

Extended Transition Function δ *

- The extended transition function δ* for DFA is an extension of Transition function δ of DFA and is defined as follows

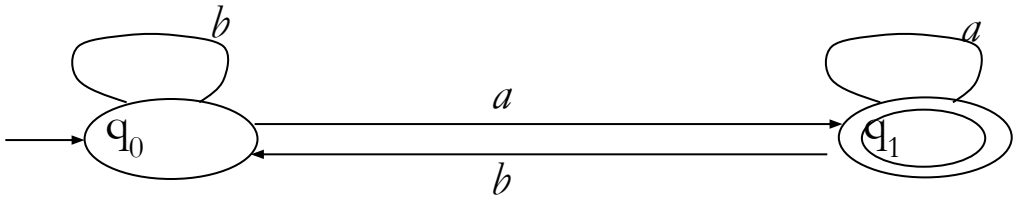
Basis : $\delta^*(q,\epsilon) = \{q\}$ that is without reading any input symbols DFA is still in same state q.

Induction: Suppose w is of the form w=xa where a is the final symbol of w and x is the rest of w.

Then $\delta^*(q, w)= \delta(\delta^*(q, x),a)$

If $\delta^*(q, x) =p$ then

$\delta^*(q, w)= \delta(\delta^*(q, x),a) = \delta(p, a)$



Use of Extended transition function δ* to validate the inputs

Example -1 - Computation of δ*(q0,bbaa) :

δ*(q0,€) = q0 → By Basis

By Induction :

δ*(q0,b) =δ(δ*(q0,€),b) = δ(q0, b) = q0

δ*(q0,bb) =δ(δ*(q0,b),b) =δ(q0,b) = q0

δ*(q0,bba) =δ(δ*(q0,bb),a) =δ(q0,a) = q1

δ*(q0,bbaa) =δ(δ*(q0,bba),a) =δ(q1,a) = q1

The string w = bbaa is **valid as DFA remains in q1 and is a final state**

Exercise problems

Example -2 - Computation of δ*(q0,abbab) :

δ*(q0,€) = q0 → By Basis

By Induction :

δ*(q0,a) =δ(δ*(q0,€),a) = δ(q0, a) = q1

δ*(q0,ab) =δ(δ*(q0,a),b) =δ(q1,b) = q0

δ*(q0,abb) =δ(δ*(q0,ab),b) =δ(q0,b) = q0

δ*(q0,abba) =δ(δ*(q0,abb),a) =δ(q0,a) = q1

δ*(q0,abbab) =δ(δ*(q0,abba),b) =δ(q1,b) = q0

The string w = abbab is **invalid as DFA remains in q0 and is not a final state**

Exercize problems : 1. Compute δ*(q0,aababb)

2. Compute δ*(q0,aaabaa)

Language of DFA

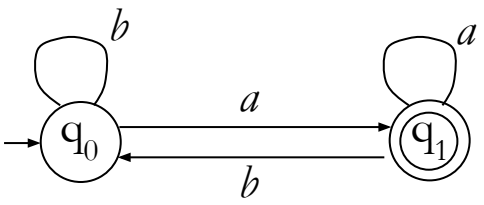
- If A={Q,∑,δ, q0, F} is an NFA then

  L(A) = {w | δ*(q0,w) = p Є F (i.e it is F) }

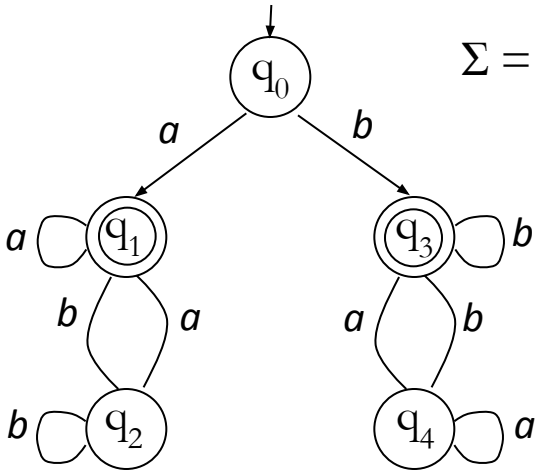
  that is L(A) is the set of strings w in ∑ * such that δ*(q0,w) contains an accepting state
- In other words - The language of a DFA $(Q, \Sigma, \delta, q_0, F)$ is the set of all strings over Σ that, starting from q_0 and following the transitions as the string is read left to right, will reach some accepting state.

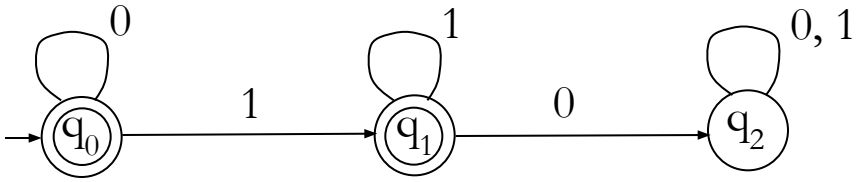
Examples

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



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





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

What are the languages of these automata?

Examples on Design Of DFA

Design the DFA for the following languages on the Alphabet $\sum$={a, b}

1. Set of all strings that starts with prefix ‘ab’.

2. Set of all strings that contains exactly one ‘a’.

3a. Set of all strings that contains at least one ‘a’.

3b. Set of all strings having substring ‘aa’.

4 Set of all strings that contains not more than three a’s

5. Set of all strings of a’s and b’s ending with substring ‘abb’.

6. Set of all strings of a's and b's not ending with substring 'abb'.
7. L={ w : |w| mod 3 = 0 }
8. L={ w : |w| mod 5 <> 0 }
9. L={ awa : w$\in$ {a, b}* }
10. L={ aw1aaw2a : w1,w2 $\in$ {a, b}* }
11. L={ ab^5wb^4 : w$\in$ {a, b}* }
12. Set of all strings of a's and b's having three consecutive a's.

13. Set of all strings of a's and b's ending with 'ab' or 'ba'.

14. L= { w : $n_a(w)$ =2, $n_b(w)$>=3, where w belongs to $\sum^*$ and $\sum$ ={a,b} }

15. L= { w : |w|mod 3 >= |w| mod 2, where w belongs to $\sum^*$ and $\sum$ ={a,b} } }

16. L= { w : $n_a(w)$ mod =2, $n_b(w)$ mod 2, where w belongs to $\sum^*$ and $\sum$ ={a,b} } }

Non Deterministic Finite Automata(NFA)

- Introduction and Why study NFA ?
- Formal Definition
- Processing of a string
- Extended Transition Function δ*
- Languages of an NFA
- Design of NFA for some languages
- Equivalence of Deterministic and Non Deterministic Finite Automata
- Conversion of NFA to DFA using subset construction Scheme by Lazy evaluation
  - Algorithm
  - Examples

Introduction

- Why NFA ?

  In order to construct a DFA for some languages we have the following difficulties.

  1. Construction is difficult.

  2. DFA cannot guess about its inputs

  3. At any point of time the DFA is in only one state and does not have the power to be in several states.

  4. Not an efficient mechanism to describe some complicated languages concisely.

These difficulties can be overcome by using NFA

Formal Definition

- The Non Deterministic Automata (NFA) is defined as 5-tuple or quintuple indicating five components:

$A = \{ Q, \sum, \delta, q_0, F\}$

Where:

1. Q is a finite set of states.
2. $\sum$ is a finite set of input symbols.
3. δ is the transition function that takes a state in Q and an input symbol in $\sum$ as arguments and returns a subset of Q i.e a set of states.

   $\delta : Q \times \sum \rightarrow 2^Q$
4. q_0 is the start state that belongs to Q
5. F is a subset of Q, is the set of final (or accepting) states.

Processing of string by an NFA

- Suppose **w=a1,a2,….an** is the string to be processed then NFA starts with **start state q0** and **a1 the first character** and consults the **transition function δ** and takes **all possible transitions** and **stays in multiple states.**
- This is continued for **all the inputs** for **each state** and finally the **set of states** in which NFA stays contains **at least on final state** then the **input is accepted, otherwise rejected.**

Extended Transition Function δ *

- The extended transition function δ* for NFA is an extension of Transition function δ of NFA and is defined as follows

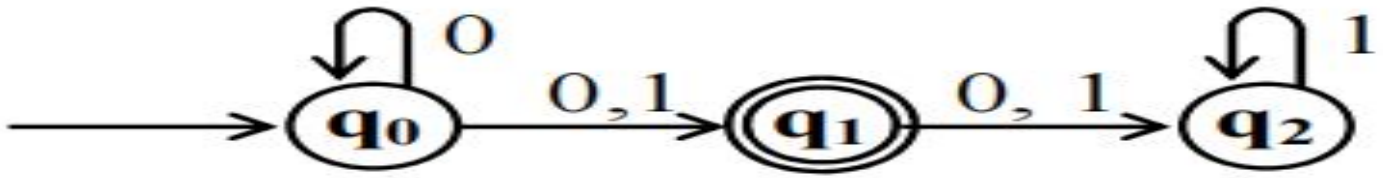
Basis : $\delta^*(q,\epsilon) = \{q\}$ that is without reading any input symbols NFA is still in same state q.

Induction: Suppose w is of the form w=xa where a is the final symbol of w and x is the rest of w.

Then $\delta^*(q, w)= \delta(\delta^*(q, x),a)$

If $\delta^*(q, x) =\{p1, p2,\ldots\ldots pk\}$ then

$$\delta^*(q, w)= \delta(\delta^*(q, x),a) = \bigcup_{i=1}^{k} \delta(pi, a) = \{r1,r2\ldots rm\}$$



Use of Extended transition function δ* to validate the inputs :

Example -1 - Computation of δ*(q0,001) :

δ*(q0,€) = q0 → By Basis

δ*(q0,0) =δ(δ*(q0,€),0) = δ(q0,0)= {q0,q1} ← By induction

δ*(q0,00) = δ(δ*(q0,0),0) = δ({q0,q1},0)

= δ(q0,0) Ụ δ(q1,0)

= {q0,q1} Ụ {q2}

= {q0,q1,q2}

δ*(q0,001) = δ(δ*(q0,00),1) = δ({q0,q1,q2},1)

= δ(q0,1) Ụ δ(q1,1) Ụ δ(q2,1)

= {q0,q1} Ụ {q2} Ụ { q2}

= {q0,q1,q2}

The string **w = 001 is valid as NFA** remains in {q0,q1,q2} and contains **q1, the final state**

Example -2 - Computation of δ*(q0,111) :

δ*(q0,€) = q0 → By Basis

δ*(q0,1) =δ(δ*(q0,€),1) = δ(q0,1)= {q1} ← By induction

δ*(q0,11) = δ(δ*(q0,1),1) = δ({q1},1) = {q2}

δ*(q0,111) = δ(δ*(q0,11),1) = δ({q2},1) = {q2]

The string **w = 111 is invalid as NFA** remains in {q2} and **does not contains q1, the final state.**

Exercize problems : 1. Compute δ*(q0,0010111)

2. Compute δ*(q0,00010)

Languages of an NFA

- If A={Q,∑,δ, q0, F} is an NFA then

  $L(A) = \{w \mid \delta^*(q0,w) \sqcap F <> \varphi \}$

  that is L(A) is the set of strings w in $\sum^*$ such that δ*(q0,w) contains at least one accepting state

Problems on the design of NFA

- Design the NFA's to recognize the following sets of strings

a. abc, abd, and aacd $\{\sum=\{a,b,c,d\}\}$

b. 0101, 101 and 011 $\{\sum=\{0,1\}\}$

c. ab,bc and ca $\{\sum=\{a,b,c\}\}$

d. Set of strings of a's and b's having prefix 'a'

e. Set of strings of a's and b's having exactly one 'a'

f. Set of strings a's and b's ending with substring 'abb'

g. $L= \{ w \mid w\varepsilon\ abab^n \text{ or } aba^n \text{ for } n>=0\}$

Equivalence of Deterministic and Non Deterministic Finite Automata

- There are many languages for which a NFA is easier to construct than DFA. It means that every language that can described by some NFA can also be described by DFA.
- We have a subset construction scheme using which any NFA is Converted to an equivalent DFA

Conversion of NFA to DFA using Subset Construction Scheme

The Subset Construction Scheme is a basic version that starts with a given NFA

$N=(Q_N,\sum,\delta_N, q_0, F_N)$ and

DFA $D=(Q_D,\sum,\delta_D, \{q_0\}, F_D)$ is to be constructed in such way that L(D) = L(N).

In this method if there are n states for NFA - N then the number of states for DFA-D is consider to be **2^N number of states** and **δ_D** is defined for the all **2^N tnumber of states.**

The following steps needs to be followed

1. Initialize DFA state set **QD to {q0}** and is the **start state of DFA D** which is the set containing only the **start state of NFA N.**
2. **While** (δ_D for all DFA state in Q_D is not defined) **Do**

   Begin

   For each state **'S' in Q_D** and for each **'a' in** $\sum$ find next state by applying transition function δ_N.

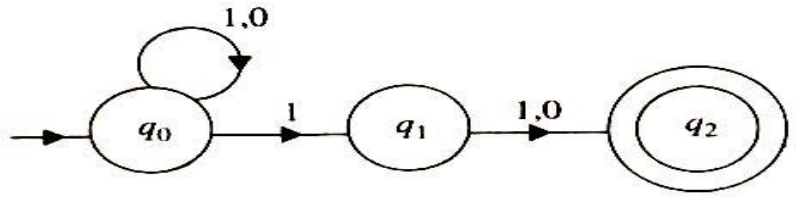
   i.e $\delta_D(S, a) = \bigcup_{p \text{ in } S} \delta_N(p, a)$

   END

3. FD is final states of DFA which contains all sets of NFA's states that include at least one final state of NFA.

   i.e if **State S** in FD then $S \cap F_N <> \emptyset$

Trace of Subset Construction Scheme Algorithm



1 NFA accepting the set of all strings whose second last symbol is 1

Following are NFA componets :

Q_N = { q0, q1, q2 }
$\sum$ = { 0, 1 }
δ_N = { Transition function Refer above Diagram }
q0 = q0
F_N = {q2}

if N is the number of states for NFA and DFA will have 2^N number of states
N=3 hence DFA will have 8 states.

Following are the DFA compoinets :

Q_D = { {Ø}, {q0}, {q1}, {q2}, {q0,q1}, {q01,q2}, {q1,q2}, {q0,q1,q2} }
$\sum$ = { 0, 1 }
δ_D = { Obtained by subset construction scheme }
q0 = {q0}
F_N = { Obtained by subset construction scheme }

δ_D	0	1	δ_D → Caclulations for inputs 0 and 1
{Ø}	{Ø}	{Ø}	$\delta_D(\{Ø\}, 0) = \delta_N(\{Ø\}, 0) = \{Ø\}$ and $\delta_D(\{Ø\}, 1) = \delta_N(\{Ø\}, 1) = \{Ø\}$
→**{q0}**	{q0}	{q0,q1}	$\delta_D(\{q0\}, 0) = \delta_N(q0, 0) = \{q0\}$ and $\delta_D(\{q0\}, 1) = \delta_N(q0, 1) = \{q0,q1\}$
{q1}	{q2}	{q2}	$\delta_D(\{q1\}, 0) = \delta_N(q1, 0) = \{q2\}$ and $\delta_D(\{q1\}, 1) = \delta_N(q1, 1) = \{q2\}$
*{q2}	{Ø}	{Ø}	$\delta_D(\{q2\}, 0) = \delta_N(q2, 0) = \{Ø\}$ and $\delta_D(\{q2\}, 0) = \delta_N(q2, 0) = \{Ø\}$
{q0,q1}	{q0,q2}	{q0,q1,q2}	$\delta_D(\{q0,q1\}, 0) = \delta_N(q0, 0) \cup \delta_N(q1, 0)$ $= \{q0\} \cup \{q2\} = \{q0,q2\}$ $\delta_D(\{q0,q1\}, 1) = \delta_N(q0, 1) \cup \delta_N(q1, 1)$ $= \{q0,q1\} \cup \{q2\} = \{q0, q1,q2\}$
*{q0,q2}	{q0}	{q0,q1}	$\delta_D(\{q0,q2\}, 0) = \delta_N(q0, 0) \cup \delta_N(q2, 0)$ $= \{q0\} \cup \{Ø\} = \{q0\}$ $\delta_D(\{q0,q2\}, 1) = \delta_N(q0, 1) \cup \delta_N(q2, 1)$ $= \{q0, q1\} \cup \{Ø\} = \{q0,q1\}$

δ_D	0	1	Calulations for inputs 0 and 1
*{q1,q2}	{q2}	{q2}	$\delta_D(\{q1,q2\}, 0) = \delta_N(q1, 0) \cup \delta_N(q2, 0)$ = {q2} ∪ {Ø} = {q2} $\delta_D(\{q1,q2\}, 1) = \delta_N(q1, 1) \cup \delta_N(q2, 1)$ = {q2} ∪ {Ø} = {q2}
*{q0,q1,q2}	{q0,q2}	{q0,q1,q2}	$\delta_D(\{q0,q1,q2\}, 0) = \delta_N(q0, 0) \cup \delta_N(q1, 0) \cup \delta_N(q2, 0)$ = {q0} ∪ { q2} ∪ {Ø} = { q0,q2) $\delta_D(\{q0,q1,q2\}, 1) = \delta_N(q0, 1) \cup \delta_N(q1, 1) \cup \delta_N(q2, 1)$ = {q0,q1} ∪ { q2} ∪ {Ø} = { q0, q1,q2 }

Conversion of NFA to DFA by using Subset Construction Scheme by Lasy Evaluation

In this method we will not consider 2^N **number of states** in finding δ_D as not all states are relevant.

i.e few states which are **not reachable** from start state are irrelavant and those states should not be considered while computing δ_D for DFA.

This method of computing δ_D only for relevant state (reachable state) is called **subset construction scheme by Lazy evaluation**

1. Initialize DFA state **set QD to {q0}** and is the **start state of DFA D** which is the set containing only the **start state of NFA N.**
2. **While** (δ_D for all DFA state in Q_D is not defined) **Do**

   Begin

   For each state **'S' in Q_D** and for each **'a' in** $\sum$ find next state T_D by applying transition function δ_N and add T_D to Q_D if it is not already there.

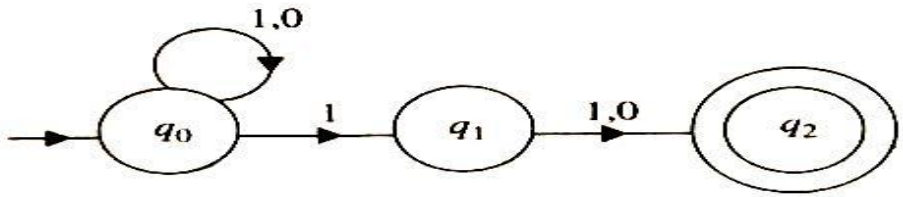
   i.e $\delta_D(S, a) = \bigcup_{p \text{ in } S} \delta_N(p, a) \rightarrow T_D$

   END

3. FD is final states of DFA which contains all sets of NFA's states that include at least one final state of NFA.

   i.e if **State S** in FD then $S \cap F_N <> \emptyset$

Example -1 NFA to DFA conversion by Subset Construction Schmeme by set LASY EVALUATION



1 NFA accepting the set of all strings whose second last symbol is 1

Let **Q_D initialized to be {q0} i.e Q_D= {q0}**

Q_D = { {q0}, {q0,q1}, {q0, q2}, {q0, q1, q2 } }

δ_D	0	1	δ_D → Calulations for inputs 0 and 1
→**{q0}**	{q0}	{q0,q1}	$\delta_D(\{q0\}, 0) = \delta_N(q0, 0)$ = **{q0}** → Old set **Not ADDED to Q_D** $\delta_D(\{q0\}, 1) = \delta_N(q0, 1)$ = **{q0,q1}** → New set for DFA added to Q_D
{q0,q1}	{q0,q2}	{q0,q1,q2}	$\delta_D(\{q0,q1\}, 0) = \delta_N(q0, 0) \cup \delta_N(q1, 0)$ = {q0} Ụ { q2} Ụ {Ø} **= { q0,q2 }** New set for DFA added to Q_D $\delta_D(\{q0,q1\}, 1) = \delta_N(q0, 1) \cup \delta_N(q1, 1)$ = {q0,q1} Ụ { q2} Ụ {Ø} **= { q0, q1,q2}** New set for DFA added to Q_D
⋇{q0,q2}	{q0}	{q0,q1}	$\delta_D(\{q0,q2\}, 0) = \delta_N(q0, 0) \cup \delta_N(q2, 0)$ = {q0} Ụ {Ø} = **{q0}** → Old set **Not ADDED to Q_D** $\delta_D(\{q0,q2\}, 1) = \delta_N(q0, 1) \cup \delta_N(q2, 1)$ = {q0, q1} Ụ {Ø} = **{q0,q1}** → Old set **Not ADDED to Q_D**

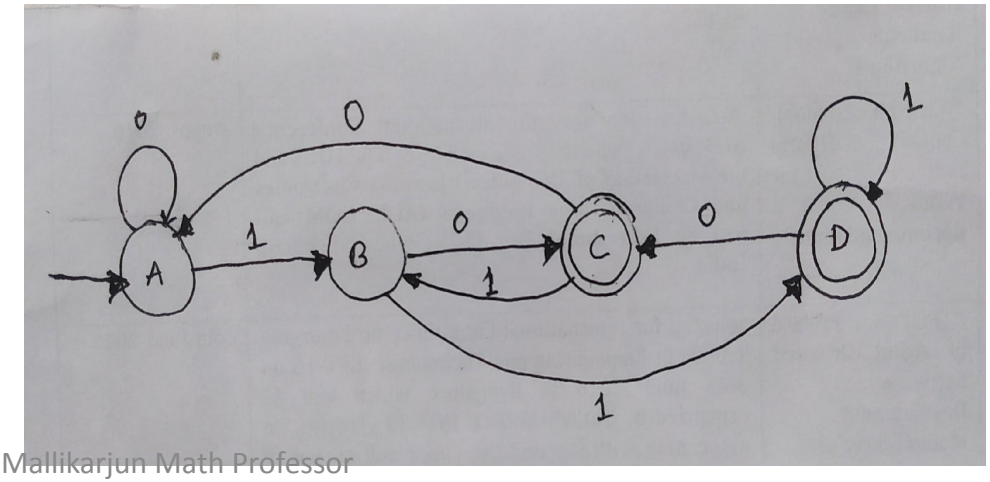
Q_D = { {q0}, {q0,q1}, {q0, q2}, {q0, q1, q2 } }

δ_D	0	1	δ_D → Calulations for inputs 0 and 1
⋇{q0,q1,q2}	{q0,q2}	{q0,q1,q2}	$\delta_D(\{q0,q1,q2\}, 0) = \delta_N(q0, 0) \cup \delta_N(q1, 0) \cup \delta_N(q2, 0)$ $= \{q0\} \cup \{ q2\} \cup \{\emptyset\}$ = { q0,q2) → Old set **Not ADDED to Q_D** $\delta_D(\{q0,q1,q2\}, 1) = \delta_N(q0, 1) \cup \delta_N(q1, 1) \cup \delta_N(q2, 1)$ $= \{q0,q1\} \cup \{ q2\} \cup \{\emptyset\}$ = { q0, q1,q2 } → Old set **Not ADDED to Q_D**

δ_D for all DFA state in Q_D have been defined

We can rename the states and write the Final DFA diagram.

Q_D = { {q0} → **A,**
{q0,q1} → **B,**
{q0, q2}→ **C,**
{q0, q1, q2 }→ **D**
}



- Exercise problems on NFA to DFA conversion

Ex-1

δ	0	1
$\rightarrow q_0$	$\{q_0, q1\}$	[q1]
q1	q2	q2
*q2	Ø	q2

Ex-2

δ	0	1
$\rightarrow q_0$	$\{q_0, q1\}$	[q0]
q1	Ø	q2
*q2	{q0, q2}	q1

Ex-3

δ	0	1
→p	{p, r}	q
q	{r, s}	p
*r	{p, s}	r
*s	{q, r}	Ø

Ex-4

δ	0	1
→p	{p, q}	p
q	{r, s}	t
r	{p, r}	t
*s	Ø	Ø
*t	Ø	Ø

Є-NFA

- Introduction
- Formal Definition
- Processing of string
- Extended Transition Function δ*
  - What is Є-Closure? And its definition
  - Definition of Extended Transition Function δ*
- Languages of Є-NFA
- Conversion of Є-NFA to DFA using subset construction Scheme by Lazy evaluation
  - Algorithm
  - Examples
- Theorems

Introduction

- Whenever a Transition takes place from one state to another State we say that NFA or DFA has taken a move and input is consumed and the input pointer is advanced to Next char.
- But some times it necessary to take a transition spontaneously without consuming or reading the input. NFA with this feature is called as Є-NFA. This gives an added programming convenience to finite automata.

Formal Definition

Formally we represent Є-NFA E by E={Q_E,∑,δ_E, q0, F_E} where all components have their same interpretation as for an NFA, except that δ_E is now a function that takes as arguments:

1. A state in $Q_E$and
2. A member of ∑ Ụ {Є }, that is either an input symbol or the symbol Є.

Processing of string and Extended Transition Function δ*

ε-NFA **uses** Є-closure() **function while processing the string 'w' which can be defined as follows :**

Є-closure of any **state q** contains a **state q** and **all the states from q** that are **reachable with only Є-transition.**

Formally we can define **Є-closure of state q - ECLOSE(q) is** recursively as follows :

Basis : **State q** is in **ECLOSE(q).**

INDUCTION : If **state p** is **ECLOSE(q)** and there is transition from **state p** to **state r** labeled **Є** then **r** is in **ECLOSE(q)**

Example to use Є-closure() function

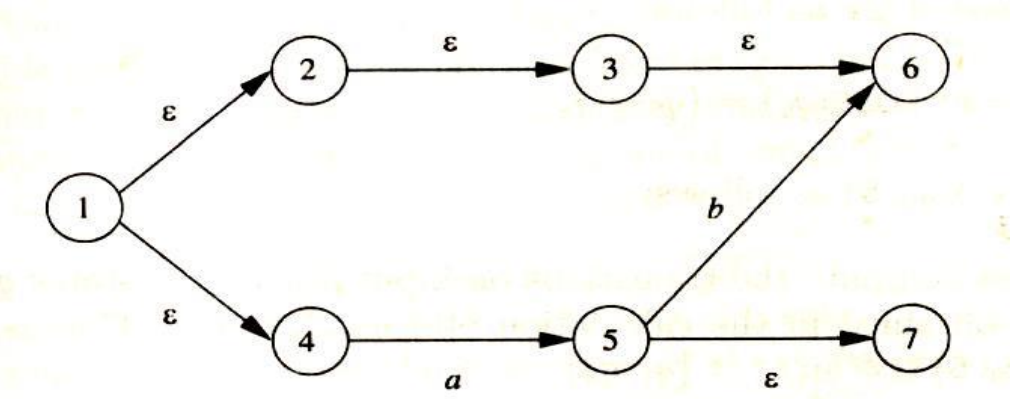


Figure 2.21: Some states and transitions

Є-closure(1) = { 1 } → By Basis **state 1** is included in the set

= { 1, **2, 4** } → By Induction as from **state 1**, **2 and 4** are reachable only by **Ɛ - Treansition**

= { 1, 2, 4, **3** } → By Induction as from **state 2**, **state 3** is reachable only by **Ɛ - Treansition**

= { 1, 2, 4 3, **6** } → By Induction as from **state 3**, **state 6** is reachable only by **Ɛ - Treansition**

= { 1, 2, 4 3, 6 } → Computaion stops as from states **1,2, 4,3,6** no other staes can reached with **Ɛ - Treansition**

Extended transition function δ* :

The extended transition function δ* for Є- NFA is an extension of Transition function δ of Є-NFA and is defined as follows

BASIS : δ*(q,Є) =ECLOSE(q) that is without reading any input symbols Є- NFA is in the set of states which is Є-closure of state q.

INDUCTION: Suppose w is of the form w=xa where a is the final symbol of w and x is the rest of w. Note that aЄ∑ and it cannot be Є.

Then δ*(q, w)= δ(δ*(q, x),a)

If δ*(q, x) ={p1, p2,……..pk} then

$$\delta^*(q, w)= \delta(\delta^*(q, x),a) = \bigcup_{i=1}^{k} \delta(pi, a) = \{r1,r2\ldots rm\}$$

$$= \bigcup_{j=1}^{m} ECLOSE(rj)$$

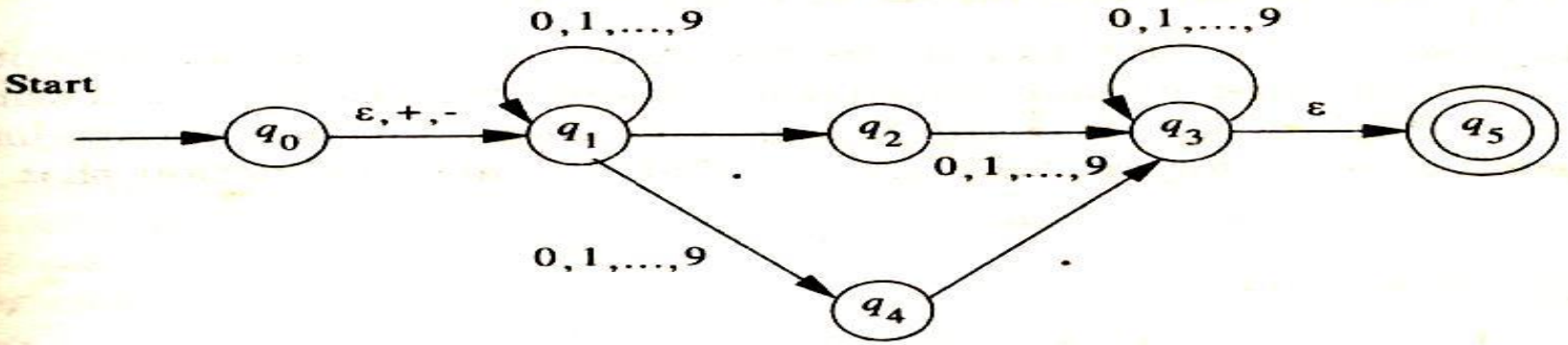


Figure 2.18: An NFA that searches for the words *state* and *regular*

Use of Extended transition function δ* to validate the inputs :

Example -1 - Computation of δ*(q0, -2•62) :

δ*(q0,Ɛ) = Ɛ-closure(q0) → By Basis

= {q0,q1}

δ*(q0,-) =δ(δ*(q0,Ɛ),-) = δ({q0,q1},-) ← By induction

= δ({q0,-) Ụ δ(q1,-)

= { q1} Ụ {Ø}

= {q1}

= Ɛ-closure(q1) ← By Induction

= {q1}

Use of Extended transition function δ* to validate the inputs :

Example -1 - Computation of δ*(q0, -2•62) :

δ*(q0,-2) =δ(δ*(q0,-), 2) = δ({q1}, 2) ← By induction

= δ({q1}, 2)

= { q1, q4}

= Є-closure(q1,q4) ← By Induction

= {q1,q4}

δ*(q0,-2•) =δ(δ*(q0,-2), •) = δ({q1,q4}, •) ← By induction

= δ({q1}, •) Ụ δ({q4}, •)

= { q2} Ụ {q3}

= Є-closure(q2,q3) ← By Induction

= {q2,q3,q5}

δ*(q0,-2•6) =δ(δ*(q0,-2•), 6) = δ({q2,q3,q5}, 6) ← By induction

= δ({q2}, 6) Ụ δ({q3}, 6) Ụ δ({q5}, 6)

= { q3 } Ụ {q3} Ụ {Ø}

= {q3}

= Є-closure(q3) ← By Induction

= {q3,q5}

$\delta^*(q0,-2\bullet62) = \delta(\delta^*(q0,-2\bullet6), 2)$ ← By induction

$= \delta(\{q3,q5\}, 2)$ =

$= \delta(\{q3\}, 2) \cup \delta(\{q5\}, 6)$

$= \{ q3 \} \cup \{\emptyset\}$

$= \{q3\}$

$= \epsilon$-closure(q3) ← By Induction

$= \{q3,q5\}$

The string **w = -2•62 is valid as Ɛ-NFA** remains in {q3,q5} and contains **q5, the final state**

Exercise problems :

Validate the following input Decimal Numbers :

{ +8.98, 56.92 and 23..45 }

Languages of an Є-NFA

- If E={Q,∑,δ, q0, F} is an Є-NFA then

  $L(E) = \{w \mid \delta^*(q0,w) \sqcap F <> \varphi \}$

  that is L(E) is the set of strings w in ∑ * such that δ*(q0,w) contains at least one accepting state

Conversion of ε-NFA to DFA by using Subset Construction Scheme by Lasy Evaluation

In this method we will not consider 2^N **number of states** in finding δ_D as not all states are relevant.

i.e few states which are **not reachable** from start state are irrelavant and those states should not be considered in computing δ_D .

This method of computing δ_D for only relevant state (reachable state) is called **subset construction scheme by Lazy evaluation**

1. Initialize DFA state **set QD to ECLOSE(q0)** and is the **start state of DFA D** which is the set containing only the **start state of Ɛ-NFA N.**
2. **While** (δ_D for all DFA state in Q_D is not defined) **Do**

   Begin

   For each state **'S' in Q_D** and for each **'a' in** $\sum$ find next state T_D by applying transition function δ_N and add T_D to Q_D if it is not already there.

   i.e $\delta_D(S, a) = \bigcup_{p \text{ in } S} \delta_N(p, a) \rightarrow T_D$

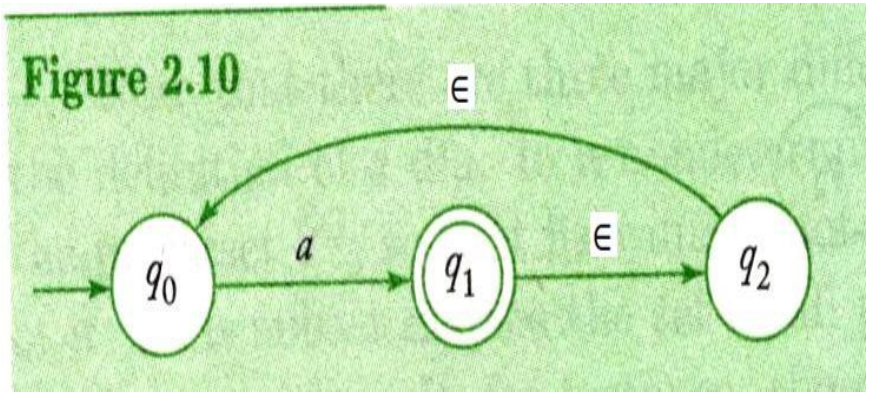
   and then Find ECLOSE(T_D) and add to Q_D if it is not already there

   END

3. **FD** is final states of DFA which contains all sets of **Ɛ-NFA's** states that include at least one **final state of Ɛ-NFA.**

   i.e if **State S** in FD then $S \cap F_N <> \emptyset$

Example -1 Ɛ-NFA to DFA conversion



Q_D = { {q0}, {q1,q2,q0} }

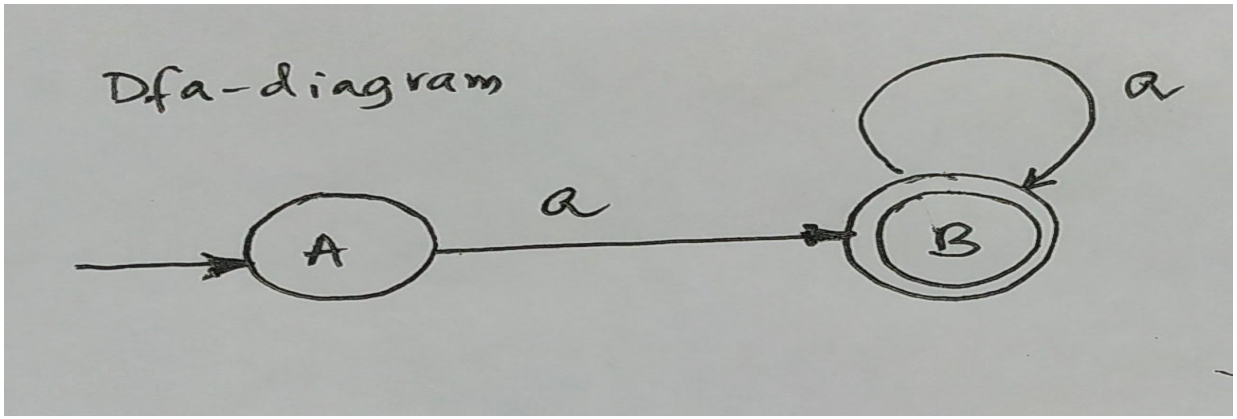
Ɛ-Closure(q0) = {q0} → Initial state for DFA and added to Q_D

δ_D	a	δ_D → Calulations for inputs a € ∑
→{q0}	{q1,q2, q0}	$\delta_D(\{q0\}, a) = \delta_N(q0, a) = \{q1\}$ and Ɛ-Closure(q1) = {q1,q2,,q0} → New set for DFA added to Q_D
✱{q1,q2,q0}	{q1,q2,q0}	$\delta_D(\{q1,q2,q0\}, a) = \delta_N(q1, a) \cup \delta_N(q2, a) \cup \delta_N(q0, a)$ $= \{\emptyset\} \cup \{\emptyset\} \cup \{q1\}$ $= \{q1\}$ **Ɛ-Closure(q1) = {q1,q2,q0} Not ADDED to Q_D**

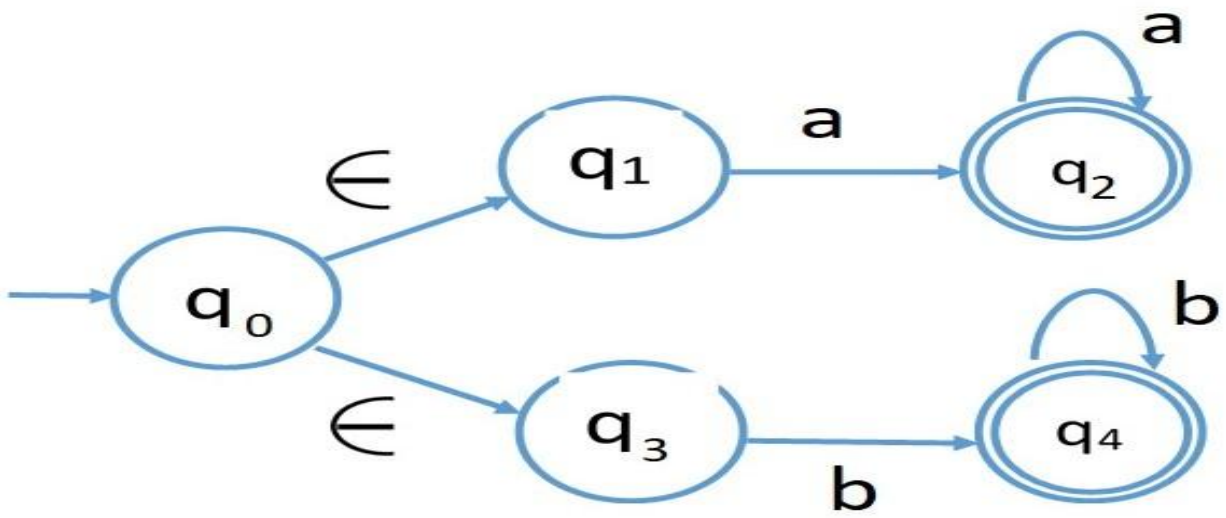
δ_D for all DFA state in Q_D have been defined

We can rename the states and write the Final DFA diagram.

Q_D = { {q0} → **A,**
{q1, q2, q0} → **B**,
}



Example -2 Ɛ-NFA to DFA conversion



Q_D = { { q0, q1, q3 } }

Ɛ-closure (q0) = { q0, q1, q3 } → Initial state for DFA and added to Q_D

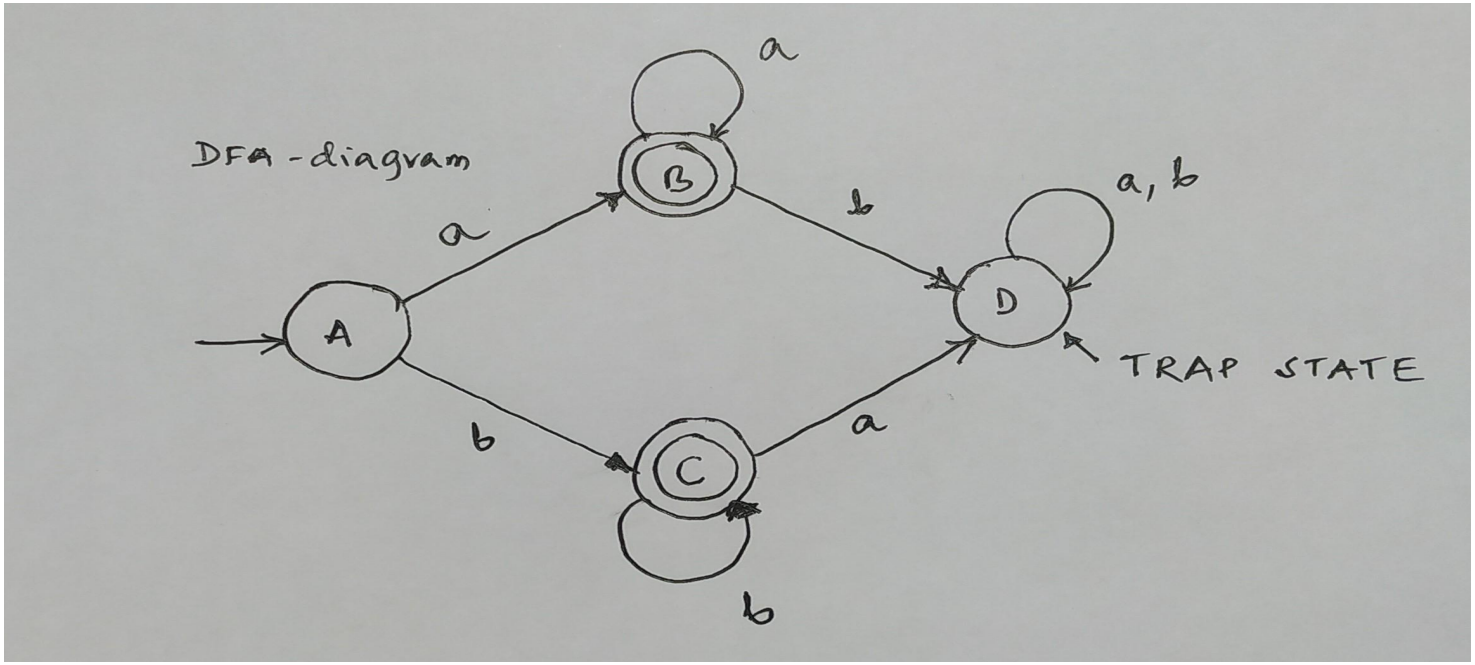
$Q_D = \{ \{ q0, q1, q3 \}, \{q2\}, \{q4\}, \{\emptyset\} \}$

δ_D	a	b	δ_D → Calulations for inputs a,b € ∑
→**{q0,q1,q3}**	{q2}	{q4}	$\delta_D(\{q0,q1,q3\}, a) = \delta_N(q0, a) \cup \delta_N(q1, a) \cup \delta_N(q3, a)$ = {Ø} Ụ {q2} Ụ {Ø} = {q2} **Ɛ-Closure(q2)** = {q2} → New set for DFA added to Q_D $\delta_D(\{q0,q1,q3\}, b) = \delta_N(q0, b) \cup \delta_N(q1, b) \cup \delta_N(q3, b)$ = {Ø} Ụ {Ø} Ụ {q4} = {q4} Ɛ-Closure(q4) = {q4} → New set for DFA added to Q_D
***{q2}**	{q2}	{Ø}	$\delta_D(\{q2\}, a) = \delta_N(q2, a)$ = {q2} **Ɛ-Closure(q2) = {q2} Not ADDED to Q_D** $\delta_D(\{q2\}, b) = \delta_N(q2, b)$ = {Ø} **Ɛ-Closure(Ø)** = { Ø} → New set for DFA added to Q_D
***{q4}**	{Ø}	{q4}	$\delta_D(\{q4\}, a) = \delta_N(q4, a)$ = {Ø} **Ɛ-Closure(Ø) = {Ø} Not ADDED to Q_D** $\delta_D(\{q4\}, b) = \delta_N(q4, b)$ = {q4} **Ɛ-Closure(q4)** = { q4} → New set for DFA added to Q_D
{Ø}	{Ø}	{Ø}	$\delta_D(\{\emptyset\}, a) = \delta_N(\{\emptyset\}, a)$ = {Ø} **Ɛ-Closure(Ø) = {Ø} Not ADDED to Q_D** $\delta_D(\{\emptyset\}, b) = \delta_N(\{\emptyset\}\ b)$ = {Ø} **Ɛ-Closure(q4) = {Ø} Not ADDED to Q_D**

δ_D for all DFA state in Q_D have been defined

We can rename the states and write the Final DFA diagram.

Q_D = { {q0,q1,q3} → **A,**
{q2} → **B**,
{q4}→ **C,**
{Ø }→ **D**
}



Excecise problems on Ɛ-NFA

Ex-1 Consider the following Ɛ-NFA

δ,	Ɛ	a	b
→p	{r}	{ q}	{p, r}
q	Ø	{p}	Ø
*r	{p,q}	{r}	{p}

a. Compute Ɛ-closure of each state.
b. Convert to an Equivalent DFA

Ex-1 Consider the following Ɛ-NFA

δ,	Ɛ	a	b	c
→p	{q,r}	Ø	{q }	{r}
q	Ø	{p}	{r}	{p,q}
*r	Ø	Ø	Ø	Ø

a. Compute Ɛ-closure of each state.
b. Convert to an Equivalent DFA

Ex-3. Convert the following Ɛ-NFA to and an equivalent DFA

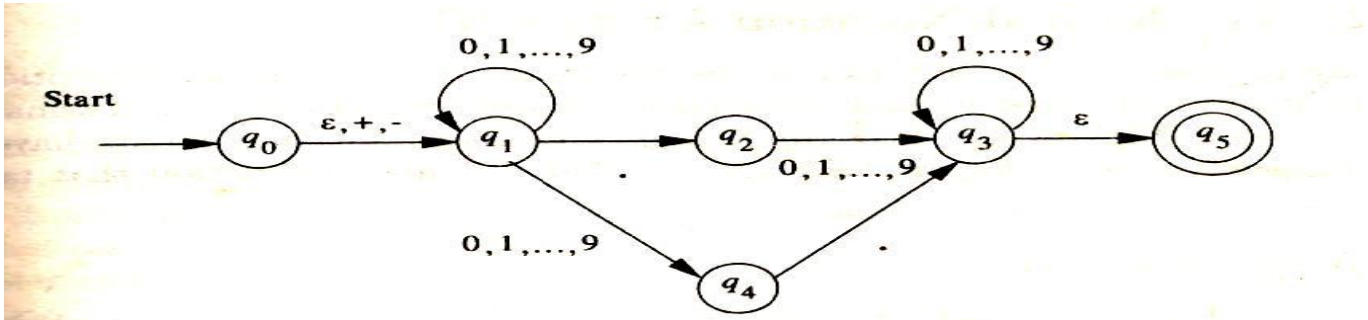


Figure 2.18: An NFA that searches for the words *state* and *regular*

Theorem-2

Definition : If $D=(Q_D,\sum,\delta_D, \{q_0\}, F_D)$ is the DFA constructed from $N =(Q_N,\sum,\delta_N, q_0, F_N)$ by the subset construction scheme then $L(D) = L(N)$.

Proof : By induction we have to first prove on $|w|$ is that

$$\delta_D^*(\{q_0\}, w)=\delta_N^*(q0,w)$$

NOTE : We know that δ^* function returns a set of state from NFA and δ_D^* for DFA interprets this as one state of Q_D.

BASIS : Let |w| = 0 it means w=Є. By the basis definitions of δ* for DFA and NFA

- we have $\delta_D^*(\{q0\},Є)$ and $\delta_N^*(q0,Є)$ are $\{q_0\}$.

INDUCTION : Let |w| = n, break up w as w=xa where a is the final symbol of w and x is the remaining characters.

- By the Induction hypothesis of δ_N^* on w, we know that

$$\delta_N^*(q0,w) = \delta_N(\delta_N^*(q0,x),a)$$

- In order to see the transition δ_N on 'a' we must have processed x from start state q0. Let $\delta_N^*(q0,x) = \{p1,p2\ldots pk\}$.
- Therefore the inductive part of the definition of δ* for NFA tells us that

$$\delta_N^*(q0,w) = \delta_N(\delta_N^*(q0,x)\ a) = \delta_N(\{p1,p2\ldots pk\},a)$$

$$= \bigcup_{i=1}^{k} \delta_N^*(pi,a) \quad \rightarrow 1$$

- By induction hypothesis of δ_D^* on 'w' we know that
  $\delta_D^*(\{q0\}, w) = \delta_D(\delta_D^*(\{q0\},x),a)$ where **w=xa** such that **a** is the **final symbol of w** and **x** is the **remaining characters**.
  Here in order to see the transition δ_D on **'a'** we must have processed **'x'**.
- By subset construction scheme, δ_D is expressed in term δ_N and set of NFA states is one DFA state in Q_D.
  i.e $\delta_D^*(\{q0\},x) = \delta_N^*(q0,x) = \{p1,p2\ldots pk\}$
- There for δ_D^* **on w** is expressed as :

$$\delta_D^*(\{q0\}, w) = \delta_D(\delta_D^*(\{q0\},x),a)$$
$$= \delta_D(\delta_N^*(\{q0\},x),a) \quad k$$
$$= \delta_D(\{p1,p2..pk\},a) = \bigcup_{i=1} \delta_N(pi, a) \rightarrow 2$$

- From 1 and 2 it demonstrates that
  $\delta_D^*(\{q0\}, w) = \delta_N^*(q0,w)$ and w is valid only when they contain state 'p' in $F_{N.}$
- This proves the argument of the theorem that **L(D) = L(N),** when DFA is constucted from NFA by a Subset construction Schme.

END of UNIT-1.