A complete note on

Introduction to Automata Theory, Formal Language and Computability Theory

BSc CSIT 4th Semester

texas College

Chapter 1

Mathematical Preliminaries: Set Functions and Relations etc.

Sets

A set is a collection of well defined objects.Usually the element of a set has common properties.e.g. all the student who enroll for a course “theory of computation” make up a set.

Examples

The set of even positive integer less than 20 can be expressed by

E = {2,4,6,8,10,12,14,16,18}

Or E = {x|x is even and 0<x<20}

Finite and Infinite Sets

A set is finite if it contains finite number of elements. And infinite otherwise.The empty set has no element and is denoted by ϕ.

Cardinality of set:

It is a number of element in a set. The cardinality of set E is

|E|=9.

Subset :

A set A is subset of a set B if each element of A is also element of B and is denoted by $A \subseteq B$.

Set operations

Union:

The union of two set has elements, the elements of one of the two sets and possibly both.Union isdenoted by U.

Intersection:

The intersection of two sets is the collection of all elements of the two sets which are common and **is** denoted by ∩.

Differences:

The difference of two sets A and B, denoted by A-B, is the set of all elements that are in the set A but not in the set B.

Sequences and Tuples

A sequence of objects is a list of objects in some order. For example, the sequence 7,4,17 would be written as (7,4,17).In set the order does not matter but in sequence it does. Also,repetition is not permitted in a set but is allowed in a sequence. Likeset , sequence may be finite or infinite.

Relations And Functions

A binary relation on two sets A and B is a subset of A×B.for example, if A={1,3,9}, B={x,y}, then {(1,x),(3,y),(9,x)} is a binary relation on 2- sets. Binary relations on K-sets $A_1,A_2,\ldots\ldots.A_k$ can be similarly defined.

A function is an object that setup an input- output relationship i.e. a function takes an input and produces the required output. For a function f , with input x, the output y, we write f(x)=y.We also say that f maps x to y.

A binary relation r is an equivalence relation if R satisfies :

R is reflexive.i.e. for every x,(x,x)ϵR.

R is symmetric i.e. for every x and y , (x,y)ϵRimlies (y,x) ϵR.

R is transitive i..e. for every x,y, and z, (x,y) ϵR and (y,z) ϵRimples (x,z) ϵR.

Closures

Closures is an important relationship among sets and is a general tool for dealing with sets and relationship of many kinds. Let R be a binary relation on a set A. Then the reflexive closure of R is arelation R' such that :

1. R' is reflexive (symmetric, transitive)
2. R'$\supseteq$ R.
3. If R'' is a reflexive relation containing R then R$\supseteq$ R

Method of proofs:

Mathematical Induction

Let A be a set of natural numbers such that :

i. 0ϵA

ii. For each natural number *n*, if{0,1,2,3,…….n}ϵA. Then A=N. In particular, induction is used to prove assertions of the form " for all nϵN, the property is valid". i.e.

In the basis step, one has to show that P(0) us true. i.e. the property is true for 0.

P holds for n will be the assumption.

Then one has to prove the validity of P for n+1.

Strong mathematical Inductions

Another form of proof by induction over natural numbers is called strong induction. Suppose we want to prove that P(n) is true for all n≥t. Then in the induction step, we assume that P(j) us true for all j, t≤j≤k. Then using this, we prove P(k). in ordinary induction in the induction step, we assume P(k-1) to prove P(k). There are some instances, where the result can be proved easily using strong induction. In some cases, it will not be possible to use weak induction and one use strong induction.

Computation:

If it involves a computer, a program running on a computer and numbers going in and out then computation is likely happening.

Theory of computation:

- It is a Study of power and limits of computing. It has three interacting components:
    - Automata Theory
    - Computability Theory
    - Complexity Theory

Computability Theory: -
- What can be computed?
- Are there problems that no program can solve?

Complexity Theory: -

- What can be computed efficiently?
- Are there problems that no program can solve in a limited amount of time or space?

Automata Theory: -

- Study of abstract machine and their properties, providing a mathematical notion of "computer"
- Automata are abstract mathematical models of machines that perform computations on an input by moving through a series of states or configurations. If the computation of an automaton reaches an accepting configuration it accepts that input.

Study of Automata

- For software designing and checking behavior of digital circuits.
- For designing software for checking large body of text as a collection ofweb pages, to find occurrence of words, phrases, patters (i.e. patternrecognition, string matching, …)
- Designing “lexical analyzer” of a compiler, that breaks input text intological units called “tokens

Abstract Model

An abstract model is a model of computer system (considered either as hardware or software) constructed to allow a detailed and precise analysis of how the computer system works. Such a model usually consists of input, output and operations that can be performed and so can be thought of as a processor. E.g. an abstract machine that models a banking system can have operations like “deposit”, “withdraw”, “transfer”, etc.

Brief History:

Before 1930’s, no any computer were there and Alen Turing introduced an abstract machine that had all the capabilities of today’s computers. This conclusion applies to today’s real machines.

Later in 1940’s and 1950’s, simple kinds of machines called finite automata were introduced by a number of researchers.

In late 1950’s the linguist N. Chomsky begun the study of formal grammar which are closely related to abstract automata.

In 1969 S. Cook extended Turing’s study of what could and what couldn’t be computed and classified the problem as:

-Decidable
-Tractable/intractable

The basic concepts of Languages

The basic terms that pervade the theory of automata include “alphabets”, “strings”, “languages”, etc.

Alphabets: - (Represented by ‘Σ’)

Alphabet is a finite non-empty set of symbols. The symbols can be the letters such as {a, b, c}, bits {0, 1}, digits {0, 1, 2, 3… 9}. Common characters like $, #, etc.

{0,1} – Binary alphabets
{+, −, *} – Special symbols

Strings: - (Strings are denoted by lower case letters)

String is a finite sequence of symbols taken from some alphabet. E.g. 0110 is a string from binary alphabet, "automata" is a string over alphabet {a, b, c … z}.

Empty String: -
It is a string with zero occurrences of symbols. It is denoted by 'ε' (epsilon).

Length of String

The length of a string w, denoted by | w |, is the number of positions for symbols in w. we have for every string s, length (s) ≥ 0.
| ε | = 0 as empty string have no symbols.
| 0110 | = 4

Power of alphabet

The set of all strings of certain length k from an alphabet is the k_{th}power of that alphabet.
i.e. Σ_k= {w / |w| = k}
If Σ = {0, 1} then,

Σ^0= {ε}
Σ^1= {0, 1}
Σ^2= {00, 01, 10, 11}
Σ^3= {000, 001, 010, 011, 100, 101, 110, 111}

Kleen Closure

The set of all the strings over an alphabet Σ is called kleen closure of Σ & is denoted by Σ_*. Thus, kleen closure is set of all the strings over alphabet Σ with length 0 or more.

$\therefore \Sigma_* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup$ ……………
E.g. A = {0}
$A_* = \{0^n / n = 0, 1, 2, \ldots\}$.

Positive Closure: -
The set of all the strings over an alphabet Σ, except the empty string is called positive closure and is denoted by Σ_+.

$\therefore \Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup$……………

Language:

A language L over an alphabet Σ is subset of all the strings that can be formed out of Σ; i.e. a language is subset of kleen closure over an alphabet Σ; $L \subseteq \Sigma^*$. (Set of strings chosen from Σ^*defines language). For example;

- Set of all strings over Σ = {0, 1} with equal number of 0's & 1's.
  L = {ε, 01, 0011, 000111, ………}
- φis an empty language & is a language over any alphabet.

- {ε} is a language consisting of only empty string.
- Set of binary numbers whose value is a prime:
  L = {10, 11, 101, 111, 1011, ……}

Concatenation of Strings

Let *x* &*y* be strings then *xy* denotes concatenation of *x* &*y*, i.e. the string formed by making a copy of *x* & following it by a copy of *y*.

More precisely, if *x* is the string of *i* symbols as $x = a_1a_2a_3\ldots a_i$&*y* is the string of *j* symbols as $y = b_1b_2b_3\ldots b_j$ then *xy* is the string of *i* + *j* symbols as $xy = a_1a_2a_3\ldots a_ib_1b_2b_3\ldots b_j$.

For example;

x = 000
y = 111
xy= 000111 &
yx= 111000

Note: 'ε' is identity for concatenation; i.e. for any *w*, ε*w*= *w*ε = *w*.

Suffix of a string

A string *s* is called a suffix of a string *w* if it is obtained by removing 0 or more leading symbols in *w*. For example;

w = abcd
s = bcd is suffix of *w*.

*here s*is proper suffix if *s* ≠ *w*.

Prefix of a string

A string *s* is called a prefix of a string *w* if it is obtained by removing 0 or more trailing symbols of *w*. For example;

w = abcd
s = abc is prefix of *w*,

Here, *s* is proper suffix i.e. *s* is proper suffix if *s* ≠ *w*.

Substring

A string *s* is called substring of a string *w* if it is obtained by removing 0 or more leading or trailing symbols in *w*. It is proper substring of *w* if $s \neq w$.
If *s* is a string then *Substr (s, i, j)* is substring of *s* beginning at i_{th}position & ending at j_{th} position both inclusive.

Problem

A problem is the question of deciding whether a given string is a member of some particular language.
In other words, if Σ is an alphabet & L is a language over Σ, then problem is;
- Given a string w in Σ^*, decide whether or not *w* is in L.

Exercises:

1. Let A be a set with n distinct elements. How many different binary relations on A are there?

2. If ∑= {a,b,c} then find the followings

    a. $\sum^1, \sum^2, \sum^3$.

3. If ∑={0,1}. Then find the following languages
    a. The language of string of length zero.
    b. The language of strings of 0's and 1's with equal number of each.
    c. The language $\{0^n 1^n | n \geq 1\}$
    d. The language $\{0^i 0^j \mid 0 \leq i \leq j\}$.
    e. The language of strings with odd number of 0's and even number of 1's.

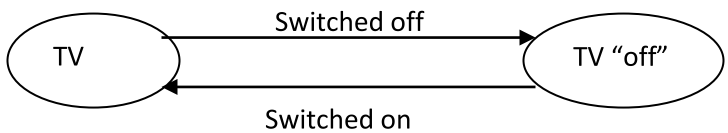
4. Define the Kleen closure and power of alphabets.

Chapter 2

Finite Automata (DFA and NFA, epsilon NFA)

Intuitive example

Consider a man watching a TV in his room. The TV is in ”on” state. When it is switched off, the TV goes to “off” state. When it is switched on, it again goes to “on” state. This can be represented by following picture.



The above figure is called state diagram.

A language is a subset of the set of strings over an alphabet. A language can be generated by grammar. A language can also be recognized by a machine. Such machine is called recognition device. The simplest machine is the finite state automaton.

2.1 Finite Automata

A finite automaton is a mathematical (model) abstract machine that has a set of “states” and its “control” moves from state to state in response to external “inputs”. The control may be either “deterministic” meaning that the automation can’t be in more than one state at any one time, or “non deterministic”, meaning that it may be in several states at once. This distinguishes the class of automata as DFA or NFA.

- The DFA, i.e. Deterministic Finite Automata can’t be in more than one state at any time.
- The NFA, i.e. Non-Deterministic Finite Automata can be in more than one state at a time.

2.1.1Applications:

The finite state machines are used in applications in computer science and data networking. For example, finite-state machines are basis for programs for spell checking, indexing, grammar checking, searching large bodies of text, recognizing speech, transforming text using markup languages such as XML & HTML, and network protocols that specify how computers communicate.

2.2. Deterministic Finite Automata

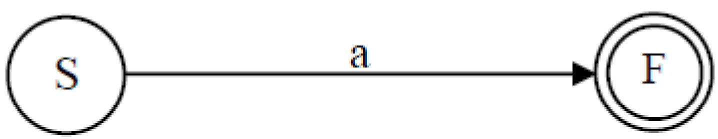
Definition

A deterministic finite automaton is defined by a quintuple (5-tuple) as (Q, Σ, δ, q_0, F).

Where,
Q = Finite set of states,
Σ = Finite set of input symbols,
δ = A transition function that maps Q × Σ -> Q
q_0 = A start state; $q_0 \in Q$
F = Set of final states; F ⊆ Q.

A transistion function δ that takes as arguments a state and an input symbol and returns a state. In our diagram, δ is represented by arcs between states and the labels on the arcs.

For example



If s is a state and a is an input symbol then δ(p,a) is that state q such that there are arcs labled 'a' from p to q.

2.2.1General Notations of DFA

There are two preferred notations for describing this class of automata;
- Transition Table
- Transition Diagram

a) Transition Table: -

Transition table is a conventional, tabular representation of the transition function δ that takes the arguments from Q × Σ & returns a value which is one of the states of the automation. The row of the table corresponds to the states while column corresponds to the input symbol. The starting state in the table is represented by -> followed by the state i.e. ->q, for q being start state, whereas final state as *q, for q being final state.
The entry for a row corresponding to state q and the column corresponding to input *a*, is the state δ (q, a).

For example:

I. Consider a DFA;
Q = {q_0, q_1, q_2, q_3}

$\Sigma = \{0, 1\}$
$q_0 = q_0$
$F = \{q_0\}$
$\delta = Q \times \Sigma \rightarrow Q$

Then the transition table for above DFA is as follows:

δ	0	1
* -> q_0	q2	q1
q1	q3	q0
q2	q0	q3
q3	q1	q2

This DFA accepts strings having both an even number of 0’s & even number of 1’s.

b) Transition Diagram:

A transition diagram of a DFA is a graphical representation where; (or is a graph)

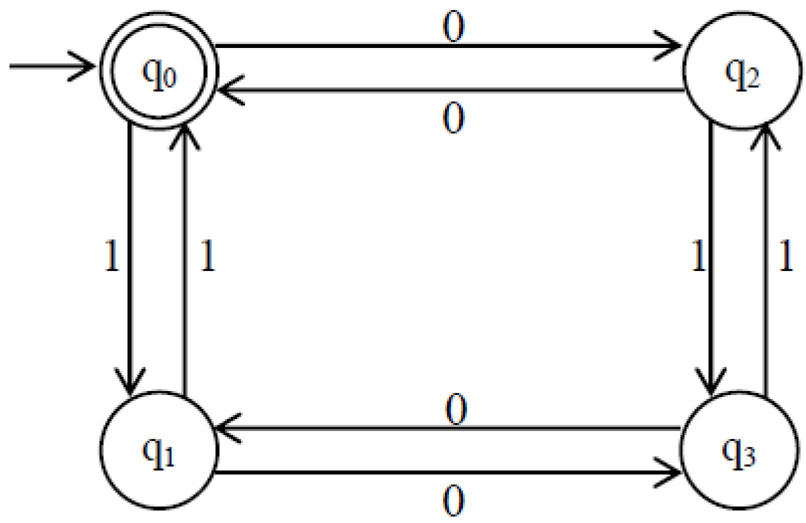
- For each state in Q, there is a node represented by circle,

- For each state q in Q and each input a in Σ, if δ (q, a) = p then there is an arc from node q to p labeled a in the transition diagram. If more than one input symbol cause the transition from state q to p then arc from q to p is labeled by a list of those symbols.

- The start state is labeled by an arrow written with “start” on the node.

- The final or accepting state is marked by double circle.

- For the example I considered previously, the corresponding transition diagram is:



2.2.2. How a DFA process strings?

The first thing we need to understand about a DFA is how DFA decides whether or not to "accept" a sequence of input symbols. The "language" of the DFA is the set of all symbols that the DFA accepts. Suppose $a_1, a_2, \ldots\ldots a_n$is a sequence of input symbols. We start out with the DFA in its start state, q_0. We consult the transition function δ also for this purpose. Say $\delta(q_0, a_1) = q_1$ to find the state that the DFA enters after processing the first input symbol a_1. We then process the next input symbol a_2, by evaluating $\delta(q_1, a_2)$; suppose this state be q_2. We continue in this manner, finding states q_3, q_4, …, q_n. such that $\delta(q_{i-1}, a_i) = q_i$ for each i. if $q_n$is a member of F, then input a_1, a_2, --- a_n is accepted & if not then it is rejected.

Extended Transition Function of DFA($\hat{\delta}$): -

The extended transition function of DFA, denoted by $\hat{\delta}$is a transition function that takes two arguments as input, one is the state q of Q and another is a string $w \in \Sigma^*$, and generates a state $p \in Q$. This state p is that the automaton reaches when starting in state q & processing the sequence of inputs w.

i.e. $\hat{\delta}(q, w) = p$

Let us define by induction on length of input string as follows:

Basis step:$\hat{\delta}(q,\epsilon) = q$.i.e. from state q, reading no input symbol stays at the same state.

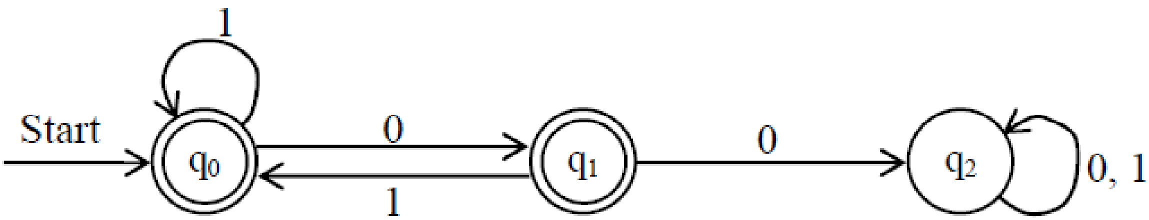
Induction:Let w be a string from Σ^* such that $w = xa$, where x is substring of w withoutlast symbol and a is the last symbol of w, then $\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$.

Thus, to compute $\hat{\delta}(q, w)$, we first compute $\hat{\delta}$ (q, x), the state the automaton is in afterprocessing all but last symbol of w. let this state is p, i.e. $\hat{\delta}$ (q, x) = p.

Then, $\hat{\delta}$ (q, w) is what we get by making a transition from state p on input a, the lastsymbol of w.

i.e. $\hat{\delta}(q, w) = \delta (p, a)$

For Example



Now compute $\hat{\delta}$ (q_0,1001)

$= \delta (\hat{\delta} (q_0, 100), 1)$

$= \delta (\delta (\hat{\delta} (q_0, 10), 0), 1)$

$= \delta (\delta (\delta (\hat{\delta} (q_0, 1), 0), 0), 1)$

$= \delta (\delta (\delta (\delta (\hat{\delta} (q_0, \varepsilon), 1), 0), 0), 1)$

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

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

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

$= \delta (q_2, 1)$

$= q_2$, *so accepted.*

c) **Compute** $\hat{\delta}$ (q_0,101) yourself.(ans : Not accepted by above DFA)

String accepted by a DFA

A string x is accepted by a DFA (Q, Σ, δ, q_0, F) if; $\hat{\delta}$ (q, x) = p ∈ F.

Language of DFA

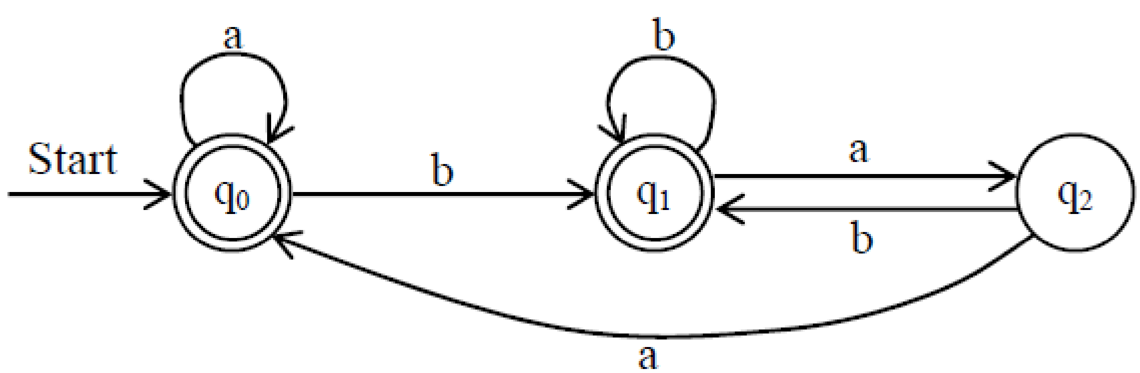
The language of DFA M = (Q, Σ, δ, q_0, F) denoted by L(M) is a set of strings over Σ*that are accepted by M.

i.e; L(M) = {w/ $\hat{\delta}$ (q_0, w) = p ∈ F}

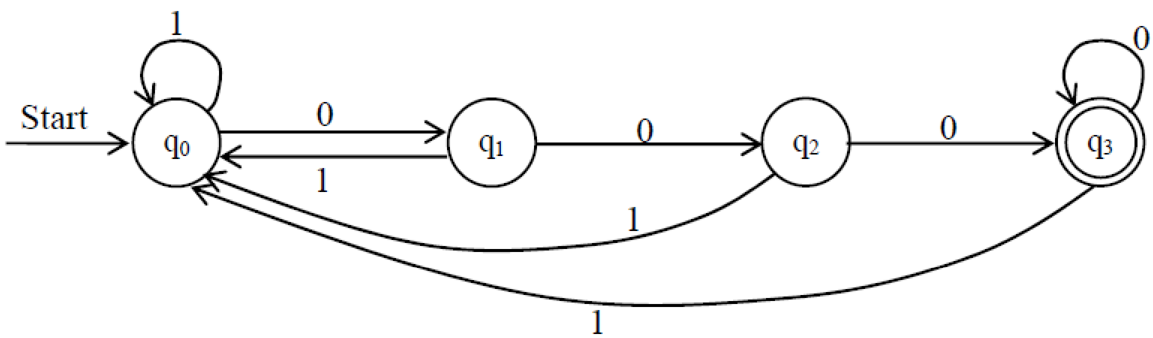
That is; the language of a DFA is the set of all strings w that take DFA starting from start state to one of the accepting states. The language of DFA is called regular language.

Examples (DFA Design for recognition of a given language)

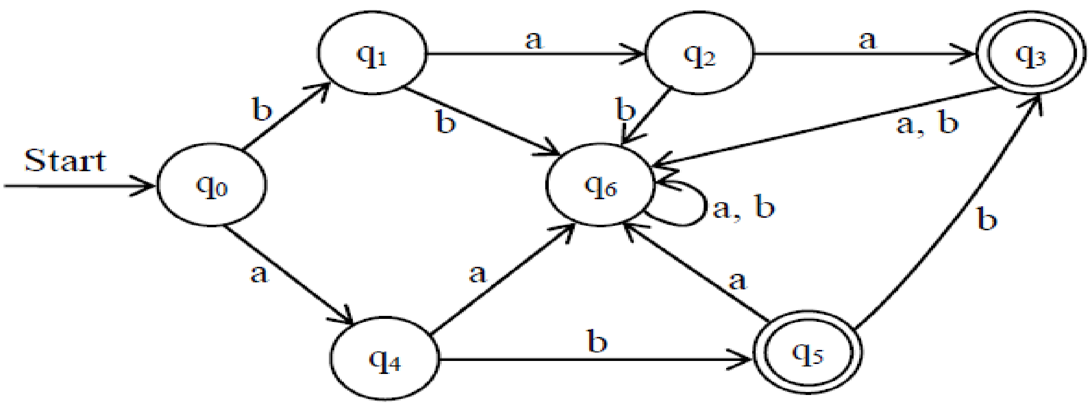
1. Construct a DFA, that accepts all the strings over Σ = {a, b} that do not end with ba.



2. DFA accepting all string over Σ = {0, 1} ending with 3 consecutive 0's.

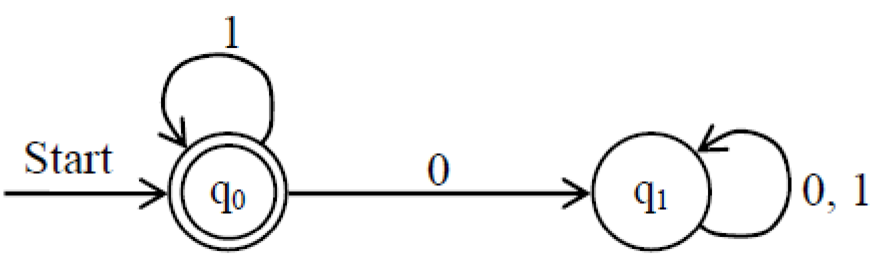


3. DFA over {a, b} accepting {baa, ab, abb}

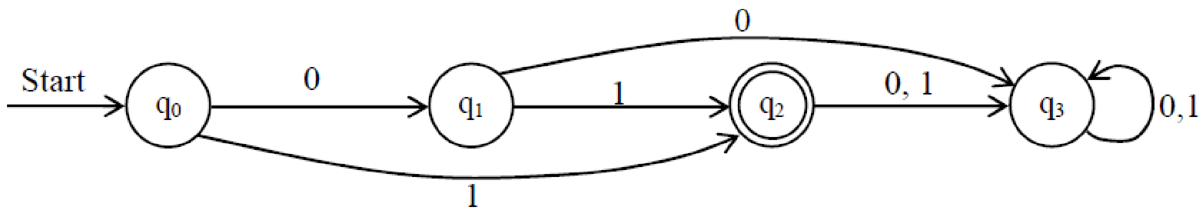


4. DFA accepting zero or more consecutive 1's.

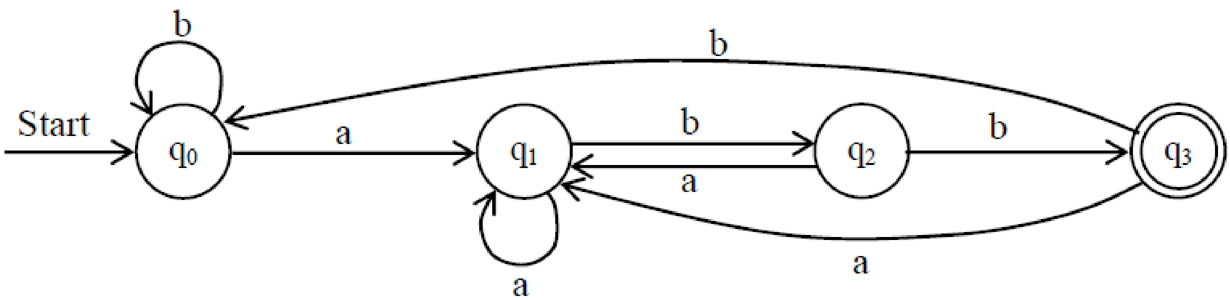
i.e. L (M) = {1^n / n = 0, 1, 2, ……}



5. DFA over {0, 1} accepting {1, 01}



6. DFA over {a, b} that accepts the strings ending with abb.



Exercises: (Please do this exercise as homework problems and the question in final exam will be of this patterns)

1. **Give the DFA for the language of string over {0.1} in which each string end with 11. [2067,TU BSc CSIT]**

2. **Give the DFA accepting the string over {a,b} such that each string does not end with ab.[2067, TU B.Sc CSIT]**

3. Give the DFA for the language of string over {a,b} such that each string contain aba as substring.
4. Give the DFA for the langague of string over {0,1} such that each string start with 01.
5. The question from book: 2.2.4, 2.2.5 of chapter 2.

2.3. Non-Deterministic Finite Automata (NFA)

A non-deterministic finite automaton is a mathematical model that consists of:

- A set of states Q, (finite)
- A finite set of input symbols Σ, (alphabets)
- A transition function that maps state symbol pair to sets of states.
- A state $q_0 \in Q$, that is distinguished as a start (initial) state.
- A set of final states F distinguished as accepting (final) state. $F \subseteq Q$.

Thus, NFA can also be interpreted by a quintuple; $(Q, \Sigma, \delta, q_0, F)$ where δ is $Q \times \Sigma = 2^Q$.Unlike DFA, a transition function in NFA takes the NFA from one state to several states just with a single input.

For example;

1.

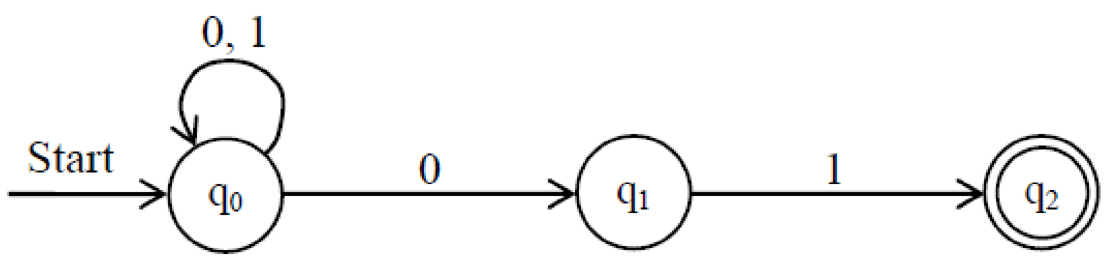
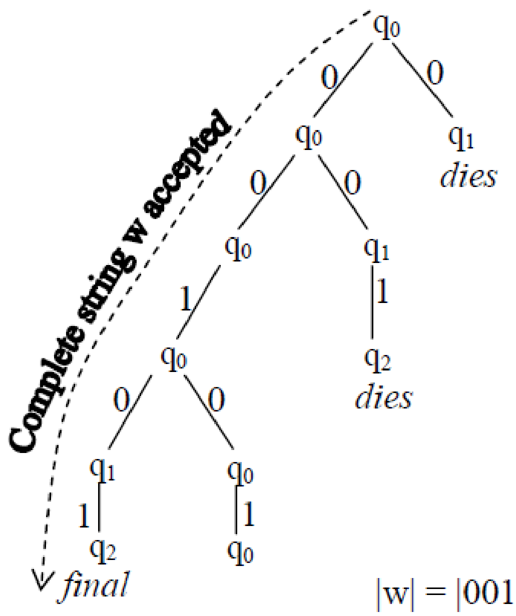


Fig: - NFA accepting all strings that end in 01.

Here, from state q_1, there is no any arc for input symbol 0 & no any arc out of q_2 for 0 & 1. So, we can conclude in a NFA, there may be zero no. of arcs out of each state for each input symbol. While in DFA, it has exactly one arc out of each state for each input symbol.

δ, the transition function is a function that takes a state in Q and an input symbol in ∑ as arguments and returns a subset of Q The only difference between an NFA and DFA is in type of value that δ returns.In NFA, δ returns a set of states and in case of DFA it returns a single state.

For input sequence w = 00101, the NFA can be in the states during the processing of the input are as:



$|w| = |00101| = 5$

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

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

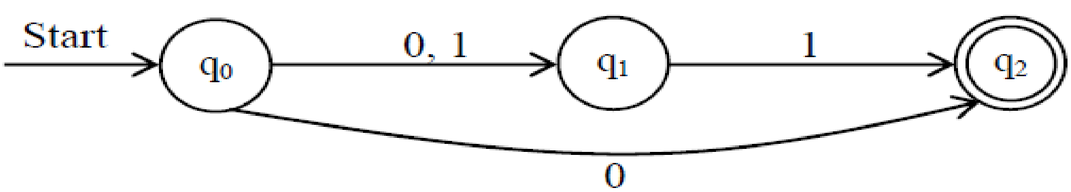
$q_0 = \{q_0\}$

$F = \{q_2\}$

Transition table:

δ:	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{\phi\}$	$\{q_2\}$
$*q_2$	$\{\phi\}$	$\{\phi\}$

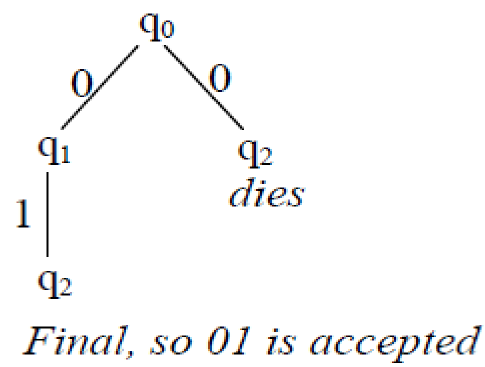
2. NFA over {0, 1} accepting strings {0, 01, 11}.



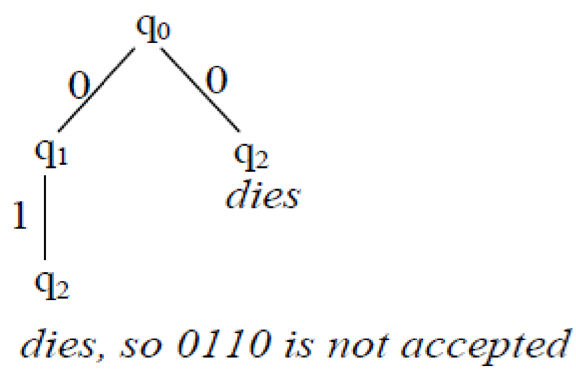
Transition table:

δ:	0	1
$\rightarrow q_0$	$\{q_0, q_2\}$	$\{q_1\}$
q_1	$\{\phi\}$	$\{q_2\}$
$*q_2$	$\{\phi\}$	$\{\phi\}$

Computation tree for 01;



Computation tree for 0110



The Extended transition function of NFA

As for DFA's, we need to define the extended transition function $\hat{\delta}$ that takes a state q and a string of input symbol w and returns the set of states that is in if it starts in state q and processes the string w.

Definition by Induction:

Basis Step:$\hat{\delta}(q, \varepsilon) = \{q\}$ i.e. reading no input symbol remains into the same state.

Induction: Let w be a string from Σ^* such that w = xa, where x is a substring of without last symbol a.

Also let,

$\hat{\delta}(q, x) = \{p_1, p_2, p_3, \ldots p_k\}$

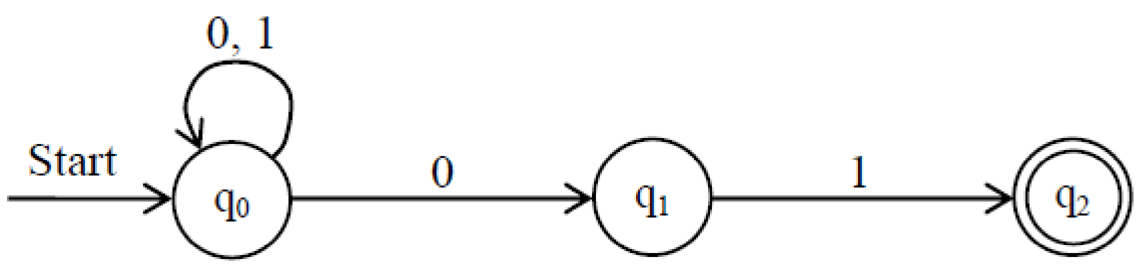
and

$$\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, r_3, \dots r_m\}$$

Then, $\hat{\delta}$ (q, w) = $\{r_1, r_2, r_3, \dots r_m\}$

Thus, to compute $\hat{\delta}$ (q, w) we first compute $\hat{\delta}$ (q, x) & then following any transition from each of these states with input a.

Consider, a NFA,



Now, computing for $\hat{\delta}$ (q_0, 01101)

Solution:

$\hat{\delta}$ (q_0, 01101)

$\hat{\delta}$ (q_0, ε) = $\{q_0\}$

$\hat{\delta}$ (q_0, 0) = $\{q_0, q_1\}$

$\hat{\delta}$ (q_0, 01) = δ (q_0, 1) ∪ δ (q_1, 1) = $\{q_0\}$ ∪ $\{q_2\}$ = $\{q_0, q_2\}$

$\hat{\delta}$ (q_0, 011) = δ (q_0, 1) ∪ δ (q_2, 1) = $\{q_0\}$ ∪ $\{\phi\}$ = $\{q_0\}$

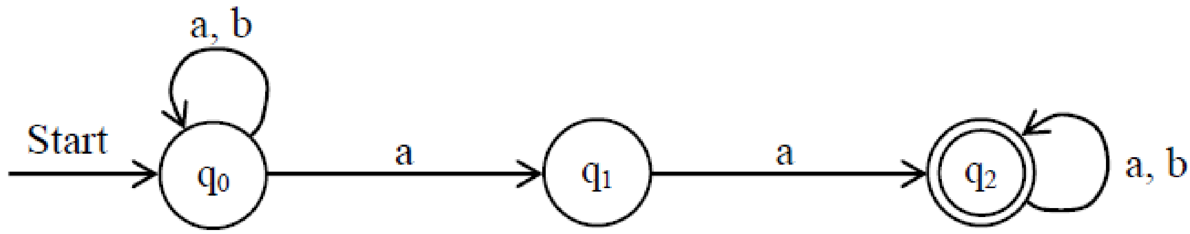
$\hat{\delta}$ (q_0, 0110) = δ (q_0, 0) = $\{q_0\}$ ∪ $\{q_1\}$ = $\{q_0, q_1\}$

$\hat{\delta}$ (q_0, 01101) = δ (q_0, 1) ∪ δ (q_1, 1) = $\{q_0\}$ ∪ $\{q_2\}$ = $\{q_0, q_2\}$

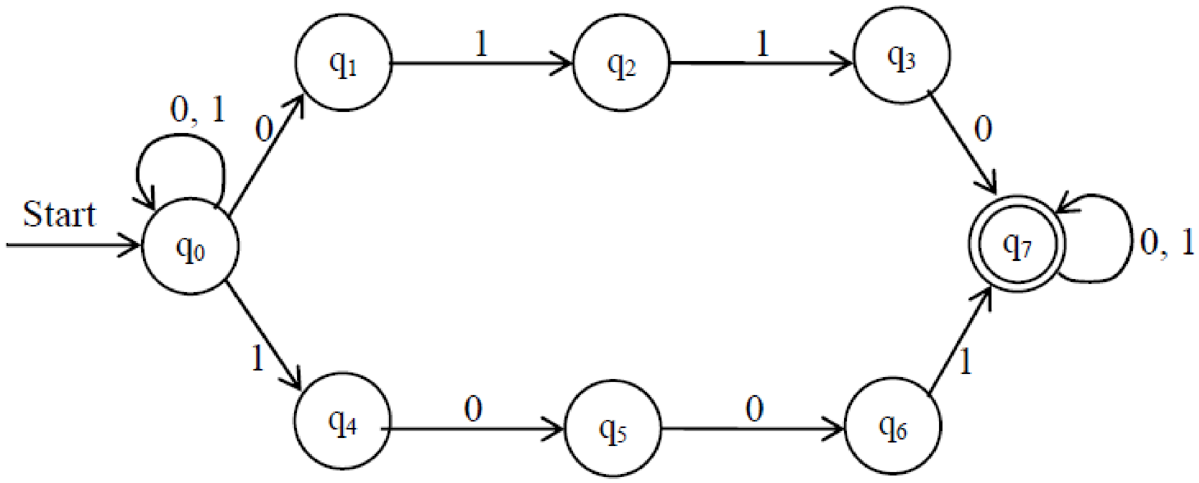
Since the result of above computation returns the set of state $\{q_0, q_2\}$) which include the accepting state q2 of NFA so the string 01101 is accepted by above NFA.

Examples (Design NFA to recognize the given language)

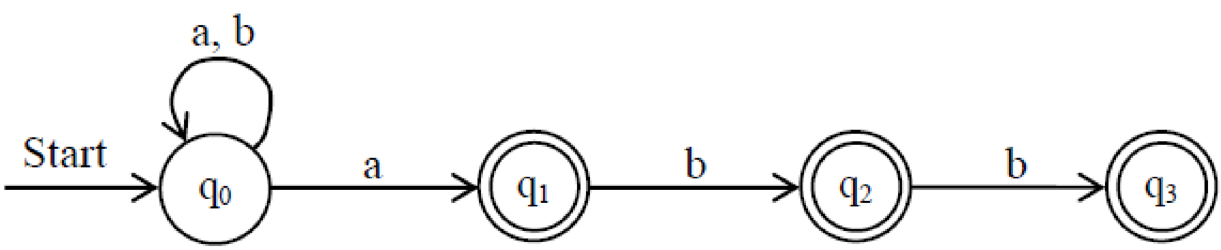
1. **Construct a NFA over {a, b} that accepts strings having *aa*as substring.**



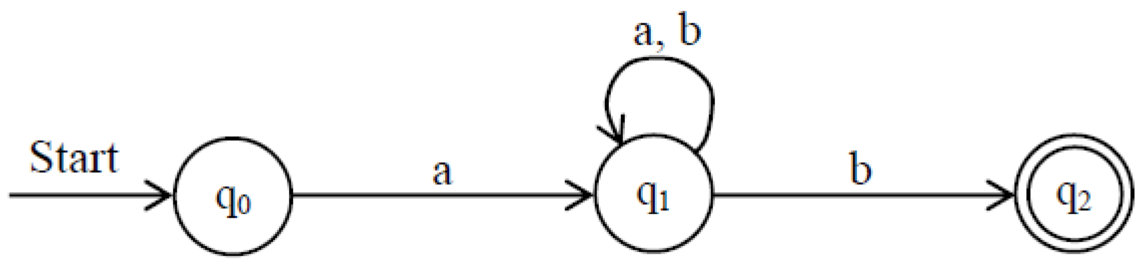
2. **NFA for strings over {0, 1} that contain substring 0110 or 1001**



3. **NFA over {a, b} that have “a” as one of the last 3 characters.**



4. **NFA over {a, b} that accepts strings starting with *a* and ending with *b*.**



Language of NFA

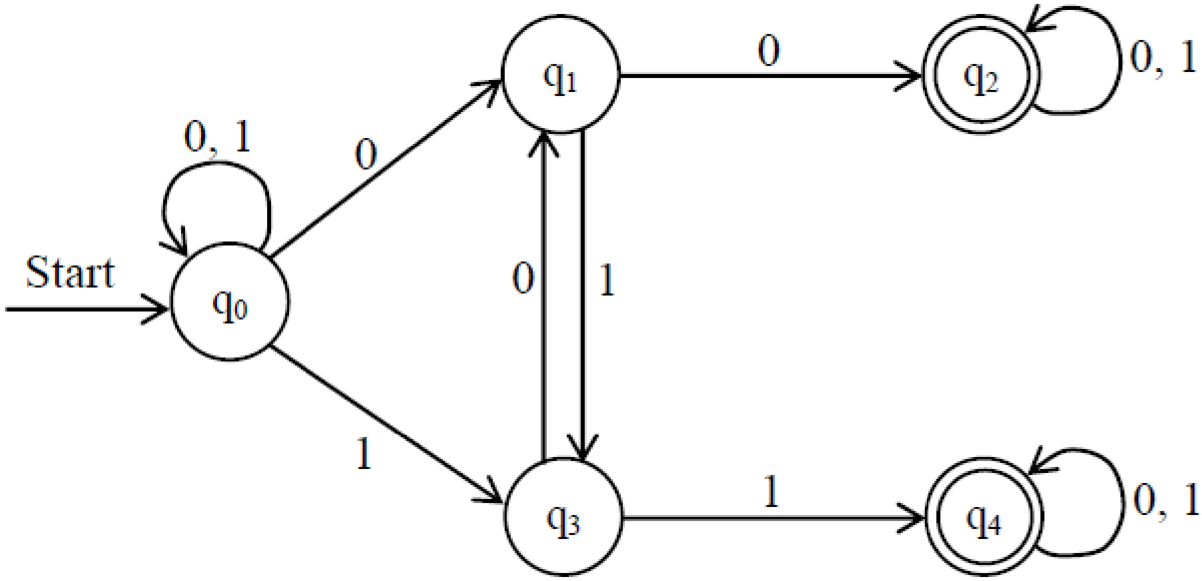
The language of NFA, M = (Q, Σ, δ, q0, F), denoted by L (M) is;

L (M) = {w/ $\hat{\delta}$(q, w) ∩ F ≠ ϕ }

i.e. L(M) is the set of strings w in ∑* such that $\hat{\delta}(q_0,w)$ contains at least one state accepting state.

Examples

1. **Design a NFA for the language over {0, 1} that have at least two consecutive 0's or1's**



Now, compute for acceptance of string 10110;

Solution

$\hat{\delta}(q_0, 10110)$

Start with starting state as

$\hat{\delta}(q_0, \varepsilon) = \{q_0\}$

$\hat{\delta}(q_0, 1) = \{q_0, q_3\}$
$(q0, 10) = \delta(q_0, 0) \cup \delta(q_3, 0) = \{q_1, q_0\} \cup \{q1\} = \{q1, q_0\}$
$(q0, 101) = \delta(q_1, 1) \cup \delta(q_0, 1) = \{q3\} \cup \{q_3\} = \{q3\}$
$(q0, 1011) = \delta(q3, 1) = \{q4\}$
$(q0, 10110) = \delta(q4, 0) = \{q4\} = \{q4\}$

So accepted (since the result in final state)

Exercise

1. **Question from book: 2.3.4 of chapter 2**
2. **Give a NFA to accept the language of string over{a.b} in which each string contain abb as substring.**
3. **Give a NFA which accepts binary strings which have at least one pair of '00' or one pair of '11'.**

2.4. Equivalence of NFA & DFA

Although there are many languages for which NFA is easier to construct than DFA, it can be proved that every language that can be described by some NFA can also be described by some DFA.

The DFA has more transition than NFA and in worst case the smallest DFA can have 2^n state while the smallest NFA for the same language has only n states.

We now show that DFAs & NFAs accept exactly the same set of languages. That isnon-determinism does not make a finite automaton more powerful.

To show that NFAs and DFAs accept the same class of language, we show;

Any language accepted by a NFA can also be accepted by some DFA. For thiswe describe an algorithm that takes any NFA and converts it into a DFA thataccepts the same language. The algorithm is called "subset constructionalgorithm".

The key idea behind the algorithm is that; the equivalent DFA simulates theNFA by keeping track of the possible states it could be in. Each state of DFAcorresponds to a subset of the set of states of the NFA, hence the name of thealgorithm. If NFA has n-states, the DFA can have 2^n states (at most), althoughit usually has many less.

The steps are:

To convert a NFA, N = (Q_N, Σ, δ_N, q_0, F_N) into an equivalentDFA D = (Q_D, Σ, δ_D, q_0, F_D), we have following steps.

1.The start state of D is the set of start states of N i.e. if q0 is start state of Nthen D has start state as {q0}.

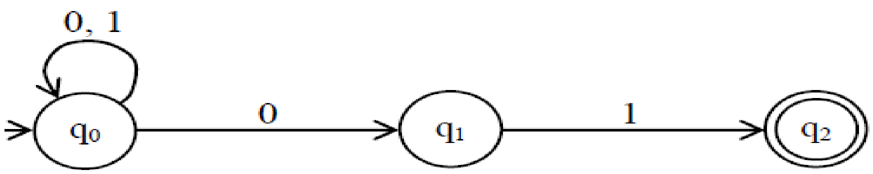
2.Q_D is set of subsets of Q_N i.e. $Q_D = 2^{QN}$. So, Q_D is power set of Q_N. So if $Q_N$has n states then Q_D will have 2^n states. However, all of these states may notbe accessible from start state of Q_D so they can be eliminated. So Q_D willhave less than 2^n states.

3.F_D is set of subsets S of Q_N such that $S \cap F_N \neq \phi$ i.e. F_D is all sets of N'sstates that include at least one final state of N.

For each set $S \subseteq Q_N$ & each input $a \in \Sigma$, $\delta_D (S, a)= \bigcup_{p \, in \, s} \delta N(p,a)$

i.e. for any state {q0, q1, q2, … qk} of the DFA & any input a, the next stateof the DFA is the set of all states of the NFA that can result as next states ifthe NFA is in any of the state's q0, q1, q2, … qk when it reads a.

For Example

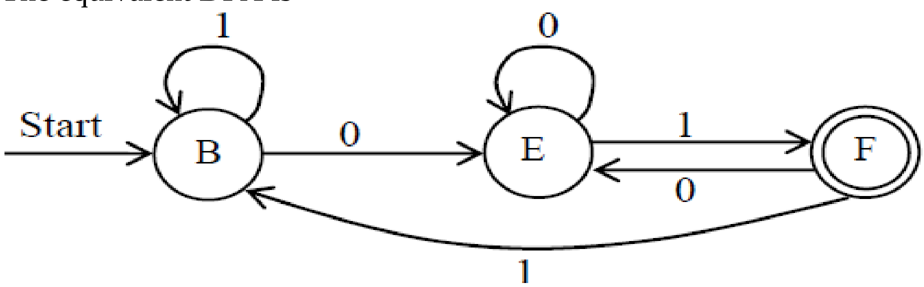


	δ:	0	1
A	ϕ	ϕ	ϕ
B	→ $\{q_0\}$	$\{q_0,q_1\}$	$\{q_0\}$
C	$\{q_1\}$	ϕ	$\{q_2\}$
D	$*\{q_2\}$	ϕ	ϕ
E	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
F	$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
G	$*\{q_1, q_2\}$	ϕ	$\{q_2\}$
H	$*\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

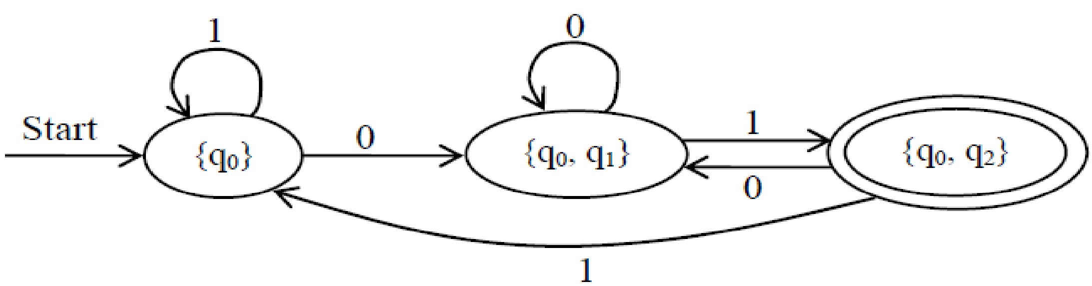
The same table can be represented with renaming the state on table entry as

δ:	0	1
A	A	A
→B	E	B
C	A	D
*D	A	A
E	E	F
*F	E	B
*G	A	D
*H	E	F

The equivalent DFA is



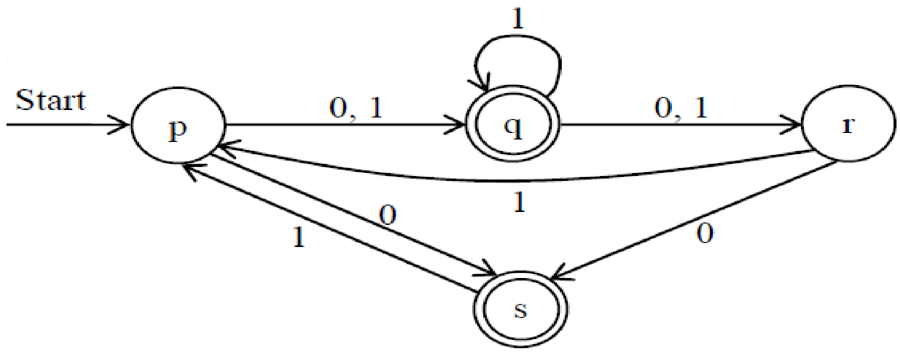
Or



The other state are removed because they are not reachable from start state.

Example2

Convert the NFA to DFA



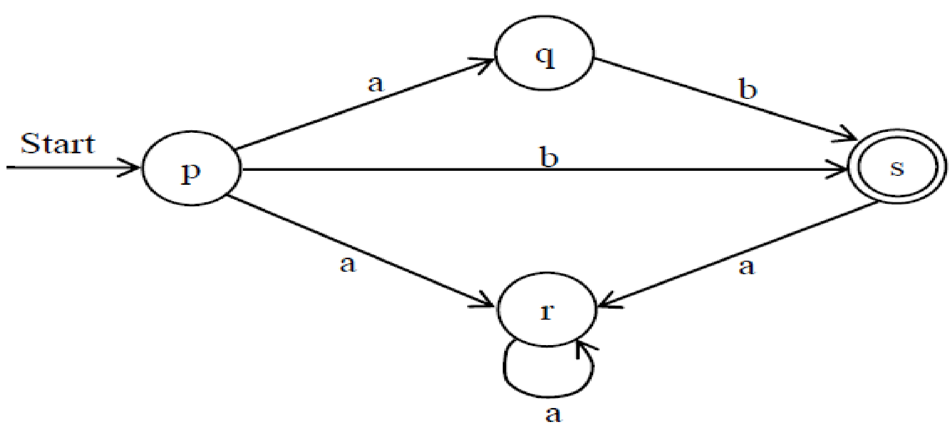
Solution: using the subset construction we have the DFA as

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

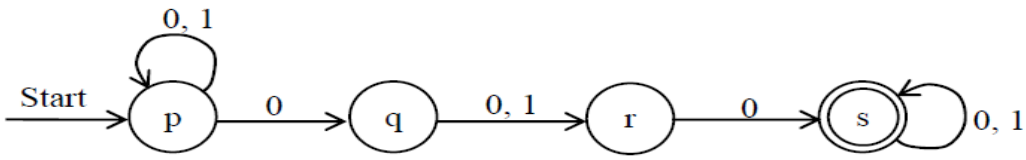
Draw the DFA diagram from above transition table yourself.

Exercises

1. Convert the following NFA to DFA



2.



3.Question from text book: 2.3.1, 2.3.2

Theorem 1:

For any NFA, N = (Q_N, Σ, δ $_N$, q0, FN) accepting language L ⊆Σ there is a DFA D =(Q_D, Σ, δ $_D$, q0',F_D) that also accepts L i.e. L (N) = L (D).*

Proof: -

The DFA D, say can be defined as;
$Q_D = 2^{QN}$, q0 = {q0}

Let S = { p1, p2, p3, … pk} ∈ Q_D. Then for S ∈ Q_D& a ∈ Σ,

$$\delta_D(s, a) = \bigcup_{p_i \in S} \delta_N(p_i, a)$$

F_D = {S / S ∈ Q_D& S ∩ F_N ≠ ϕ }

The fact that D accepts the same language as N is as;
for any string w ∈ Σ*;
$\hat{\delta}_N$ (q0, w) = $\hat{\delta}_D$ (q0, w)

Thus, we prove this fact by induction on length of w.

Basis Step:

Let |w| = 0, then w = ε,

$\hat{\delta}_N$(q0, ε)={q0)=q0=$\hat{\delta}_D$(q0, ε)

Induction step;

Let |w| = n + 1 is a string such that w = xa& |x| = n, |a| = 1; a being last symbol.
Let the inductive hypothesis is that x satisfies.
Thus,
$\hat{\delta}_D$ (q0', x) =$\hat{\delta}_N$ (q0, x), let these states be {p1, p2, p3, …pk}

Now,

$\hat{\delta}_N$ (q0, w) $= \hat{\delta}_N$ (q0, xa)

$= \delta_N (\hat{\delta}_N$ (q0, x), a)

$= \delta_N$ ({p1, p2, p3, …pk}, a) [Since, from inductive step]

$= \cup\delta_N(p_i,a)$……………………(1)

Also

$\hat{\delta}_D$ (q0', w) $= \hat{\delta}_D$ (q0', xa)

$= \delta_D (\hat{\delta}_D$ (q0', x), a)

$= \delta_D (\hat{\delta}_N$ (q0, x), a) [Since, by the inductive step as it is true for x]

$= \delta_D$ ({p1, p2, p3, …pk}, a) [Since, from inductive step]

Now, from subset construction, we can write,

δ_D ({p1, p2, p3, … pk}, a) $= \cup\delta_N(p_i,a)$

so, we have

$\hat{\delta}_D$ (q0', w) $= \cup\delta_N(p_i,a)$………………………..(2)

Now we conclude from 1 and 2 that

$\hat{\delta}_N$ (q0, w) $= \hat{\delta}_D$ (q0', w).

Hence, if this relation is true for |x| = n, then it is also true for |w| = n + 1.

$\therefore$ DFA D & NFA N accepts the same language.

i.e. L (D) = L (N)**Proved.**

Theorem 2:

A language L is accepted by some DFA if and only if L is accepted by some NFA.

Proof:

'if' part (A language is accepted by some DFA if L is accepted by some NFA):

It is the subset construction and is proved in previous theorem. In exam, you should write the proof of previous theorem here.

Only if part (a language is accepted by some NFA if L is accepted by some DFA):

Here we have to convert the DFA into an identical NFA.

Consider we have a DFA D = (Q_D, Σ, δ_D, q0, F_D).

This DFA can be interpreted as a NFA having the transition diagram with exactly one choice oftransition for any input.

Let NFA N = (Q_N, Σ, δ_N, q0', F_N) to be equivalent to D.
Where $Q_N = Q_D$, $F_N = F_D$, q0' = q0 and $\delta_N$is defined by the rule

If $\delta_D(p,a)=q$ then $\delta_N(p,a)= \{q\}$.

.
Then to show if L is accepted by D then it is also accepted by N, it is sufficient to show, for anystring $w \in \Sigma^*$,$\hat{\delta}_D$ (q0, w) = $\hat{\delta}_N$ (q0, w)

We can proof this fact using induction on length of the string.
Basis step: -

Let |w| = 0 i.e. w = ε
$\therefore \hat{\delta}_D$ (q0, w) =$\hat{\delta}_D$ (q0, ε) = q0
$\hat{\delta}_N$ (q0, w) = $\hat{\delta}_N$ (q0, ε) = {q0}
$\therefore \hat{\delta}_D$ (q0, w) = $\hat{\delta}_N$ (q0, w) for |w| = 0 is true.
Induction: -
Let |w| = n + 1 & w = xa. Where |x| = n & |a| = 1; a being the last symbol.
Let the inductive hypothesis is that it is true for x.
$\therefore$if $\hat{\delta}_D$ (q0, x) = p, then $\hat{\delta}_N$ (q0, x) = {p}

i.e. $\hat{\delta}_D$ (q0, x) = $\hat{\delta}_N$ (q0, x)

Now,

$\hat{\delta}_D$ (q0, w) = $\hat{\delta}_D$ (q0, xa)
= δ_D ($\hat{\delta}_D$ (q0, x), a)
= δ_D (p, a) [from inductive step$\hat{\delta}_D$ (q0, x,=p]
= r, say

Now,

$\hat{\delta}_N$ (q0, w) = $\hat{\delta}_N$ (q0, xa)
= δ_N ($\hat{\delta}_N$ (q0, x), a) [from inductive steps]
= δ_N({p}, a)
=r [from the rule that define δ_N)

Hence proved.i.e.$\hat{\delta}_D$ (q0, w)=$\hat{\delta}_N$ (q0, w)

NFA with ε-transition (ε-NFA)

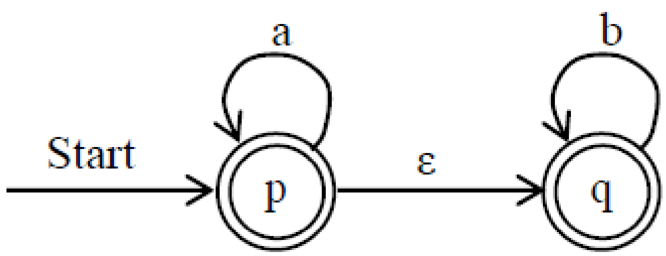
This is another extension of finite automation. The new feature that it incorporates is, itallows a transition on ε, the empty string, so that a NFA could make a transitionspontaneously without receiving an input symbol.

Like Non-determinism added to DFA, this feature does not expand the class of language that can be accepted by finite automata, but it does give some added programming connivance. This are very helpful when we study regular expression (RE) and prove the equivalence between class of language accepted by RE and finite automata.

A NFA with ε-transition is defined by five tuples (Q, Σ, δ, q0, F), where;

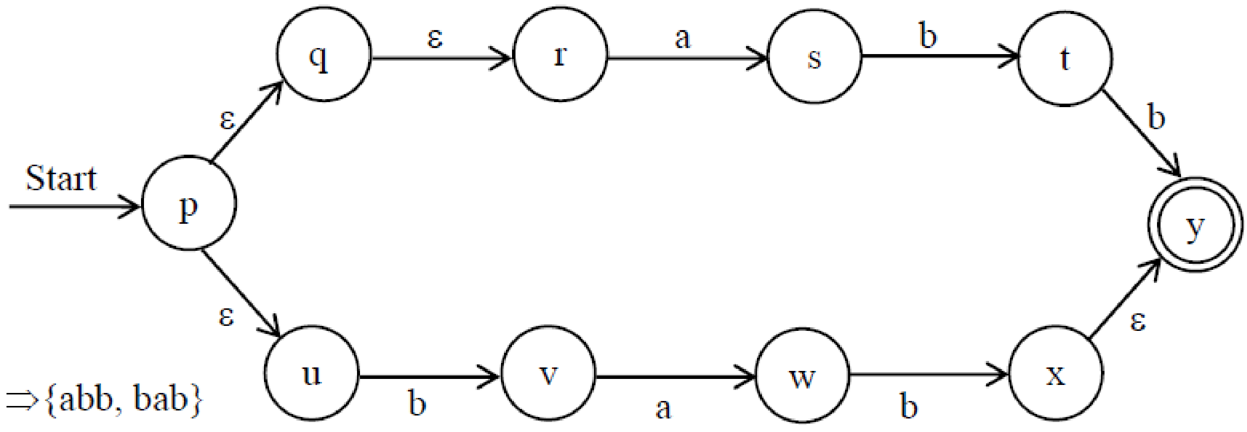
Q = set of finite states
Σ = set of finite input symbols
q0 = Initial state, q0 $\in$ Q
F = set of final states; F $\subseteq$ Q
δ = a transition function that maps;
$Q \times \Sigma \cup \{\varepsilon\} \dashrightarrow 2^Q$

For examples:
1.



This accepted the language {a,aa,ab,abb,b,bbb,…………………}

2.



ε-closure of a state:

ε-closure of a state 'q' can be obtained by following all transitions out of q that arelabeled ε. After we get to another state by following ε, we follow the ε-transitions outof those states & so on, eventually finding every state that can be reached from q alongany path whose arcs are all labeled ε.

Formally, we can define ε-closure of the state q as;

Basis: state q is in ε-closure (q).

Induction: If state q is reached with ε-transition from state q, p is in ε-closure (q). And ifthere is an arc from p to r labeled ε, then r is in ε-closure (q) and so on.

Extended Transition Function of ε-NFA: -

The extended transition function of ε-NFA denoted by $\hat{\delta}$,is defined as;

i) BASIS STEP: - $\hat{\delta}$ (q, ε) = ε-closure (q)

ii) INDUCTION STEP: -

Let w = xa be a string, where x is substring of w without last symbol a and a $\in \Sigma$ but a $\neq$ ε.

Let$\hat{\delta}$ (q, x) = {p1, p2, … pk} i.e. pi's are the states that can be reached from qfollowing path labeled x which can end with many ε & can have many ε.

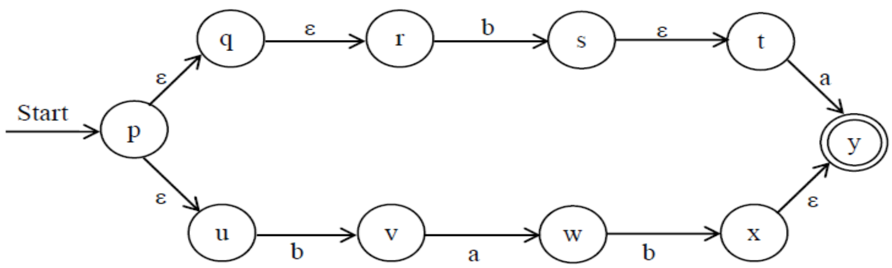
Also let,

$$\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, \ldots r_m\}$$

then

$$\delta(q, x) = \bigcup_{j=1}^{m} \varepsilon - closure\ (r_j)$$

Example



Now compute for string ba

-$\hat{\delta}$ (p,ε)=ε-closure(p)={p,q,r,u}

Compute for b i.e.
-δ(p,b)Uδ(q,b)Uδ(r,b)Uδ(u,b)={s,v}
-ε-colsure(s)Uε-closure(v)={s,t,v}

Computer for next input ‘a’
-δ(s,a)Uδ(t,a)Uδ(v,a)={y,w}
-ε-closure(y)Uε-closure(w)={y,w}

The final result set contains the one of the final state so the string is accepted.

Assignment 1(30 marks)

Short questions

1. Define set, function and relation. 3 marks

2. Define mathematical induction. 2 marks

3. Define alphabets, strings and languages 3 marks

4. Define three components of theory of computations 3 marks

Longs questions

1. Define DFA and NFA with example. (Give Mathematical Definition) 6 marks

1. Design the DFA for the language of all string that start with 0 and end with 1 over the symbols {0,1}.

Represent it with transition table and transition diagram. 6 marks

2. Design NFA for the language of all string that contain 001 as substring over the alphabet {0,1}. Represent it with transition table and transition diagram. 6 marks

Note: 1 marksis reserved for the fair and clean handwriting.