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 is denoted by .

Intersection:

The intersection of two sets os 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. Like set , sequence may be finite or infinite.

Relations A nd 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)ϵR imlies (y,x) ϵR.

R is transitive i..e. for every x,y, and z, (x,y) ϵR and (y,z) ϵR imples (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 $n \geq t$. Then in the induction step, we assume that P(j) us true for all j, $t \leq j \leq k$. 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 of web pages, to find occurrence of words, phrases, patters (i.e. pattern recognition, string matching, …)
- Designing "lexical analyzer" of a compiler, that breaks input text into logical 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. $\Sigma_k = \{w / |w| = k\}$
If $\Sigma = \{0, 1\}$ then,

$\Sigma^0 = \{\varepsilon\}$
$\Sigma^1 = \{0, 1\}$
$\Sigma^2 = \{00, 01, 10, 11\}$
$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

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, ……}
- φ is 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*, $\varepsilon w = w\varepsilon = w$.

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 $s \neq w$.

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 $s \neq w$.

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 $\sum = \{a,b,c\}$ then find the followings

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

3. If $\sum = \{0,1\}$. 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 $\{0^n1^n \mid n \geq 1\}$
    d. The language $\{0^i0^j \mid 0 \leq i \leq j\}$.
    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.

Readings: Plz, read the *chapter 1 sections* 1.1, 1.1.3, 1.5 of Text book. And solve some numerical example given in Text book.