Basic Foundations

Unit 1

Set Theory

A set is a collection of well defined objects

The element of a set have common properties

eg: All the student who enroll for a course "Robotics and Artificial Intelligence" make up a set

Set Theory

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

$S = \{2,4,6,8,10,12,14,16,18\}$

Or $S = \{x \mid x \text{ is even and } 0<x<20\}$

A set is **finite** if it contains finite number of elements; **infinite** otherwise

An **empty set** has no elements and is denoted by $\varnothing$

Set Theory

The **cardinality** of a set is the number of elements in it

The cardinality of set S is $|S|=9$

A set A is **subset** of a set B if each element of A is also element of B

It is denoted by $A \subseteq B$

Set Theory

The **power set** is a set which includes all the subsets including the empty set and the original set itself

If set A = {x, y, z} , power set of A, P(A) = { {x}, {y}, {z}, {x, y}, {y, z}, {x, z}, {x, y, z}, {} }

A **unit set** is a set containing only one element

Set Theory

The **union** of two sets A and B is a set containing all elements that are in A or in B (possibly both)

$A \cup B = \{x : x \in A \text{ or } x \in B\}$

The **intersection** of sets A and B is the set of all elements which are common to both A and B

$A \cap B = \{x : x \in A \text{ and } x \in B\}$

Set Theory

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

If A = {1, 2, 3, 4, 5, 6} and B = {3, 4, 5, 6, 7, 8},

A – B = {1, 2}

B – A = {7, 8}

Logic

Logic is the study of correct reasoning

A valid inference is one where there is a specific relation of logic support between the assumptions of the inference and its conclusion

Logic

Propositional logic studies the logical relationships between propositions (or statements, sentences, assertions) taken as a whole, and connected via logical connectives

A **proposition** is a declarative sentence that is either True or False, but not both

eg: $1 + 1 = 2$ $\rightarrow$ TRUE

‘b’ is a vowel $\rightarrow$ FALSE

Logic

Predicate logic deals with **predicates**, which are propositions containing variables

A **predicate** is an expression of one or more variables defined on some specific domain

eg: G(x, y) denotes "x is greater than y"

Logic

The variable of predicates is quantified by **quantifiers**

Construction of propositions from predicates using quantifiers is called quantification

There are two types of quantifier in predicate logic – **Universal Quantifier** and **Existential Quantifier**

Logic

Universal quantifier states that the statements within its scope are true for every value of the specific variable

It is denoted by the symbol $\forall$

$\forall xP(x)$ is read as for every value of x, P(x) is true

eg: If P(x) denotes x is mortal and universe of discourse is all humans, then $\forall xP(x)$ denotes “All humans are mortal”

Logic

Existential quantifier states that the statements within its scope are true for some values of the specific variable

It is denoted by the symbol $\exists$

$\exists xP(x)$ is read as for some values of x, P(x) is true

eg: If P(x) denotes x is dishonest and universe of discourse is some people, then $\exists xP(x)$ denotes “Some people are dishonest”

Functions

If A and B are two sets, a **function** f from A to B is a rule that assigns each element x of A an element f(x) of B

For a function f, with input x, and output y, we write f(x) = y

Functions

A **binary relation** on two sets A and B is a subset of A × B

eg: if A = {1,3,9}, B = {x,y}, then {(1,x), (3,y), (9,x)} is a binary relation on 2- sets

k-ary relations on k-sets A_1, A_2, …., A_k can be similarly defined

Functions

A binary relation R is an equivalence relation if:

R is reflexive i.e. for every x, $(x, x) \in R$

R is symmetric i.e. for every x and y , $(x, y) \in R$ implies $(y, x) \in R$

R is transitive i.e. for every x, y, and z, $(x, y) \in R$ and $(y, z) \in R$ implies $(x, z) \in R$

Proofs

Direct Proof

Proof by Contradiction

Mathematical Induction

etc

Theory of Computation

It is the study of power and limits of computing

It has three interacting components:

- Automata Theory
- Computability Theory
- Complexity Theory

Computability Theory

The computability theory defines whether a problem is “solvable” by any abstract machine

Some problems are computable while others are not

Computation is done by various computation models depending on the nature of the problem at hand

Eg machines: the turing machine, finite state machines

Complexity Theory

This theoretical computer science branch is all about studying the cost of solving problems while focusing on resources (usually time & space) needed as the metric

Are there problems that no program can solve in a limited amount of time or space?

The running time of an algorithm varies with the inputs and usually grows with the size of the inputs

Complexity Theory

Common measurements when analyzing algorithms:

- $O(1) \rightarrow$ Constant time/space

- $O(n) \rightarrow$ Linear time/space

- $O(\log n) \rightarrow$ Logarithmic time

- $O(n^2) \rightarrow$ Quadratic time

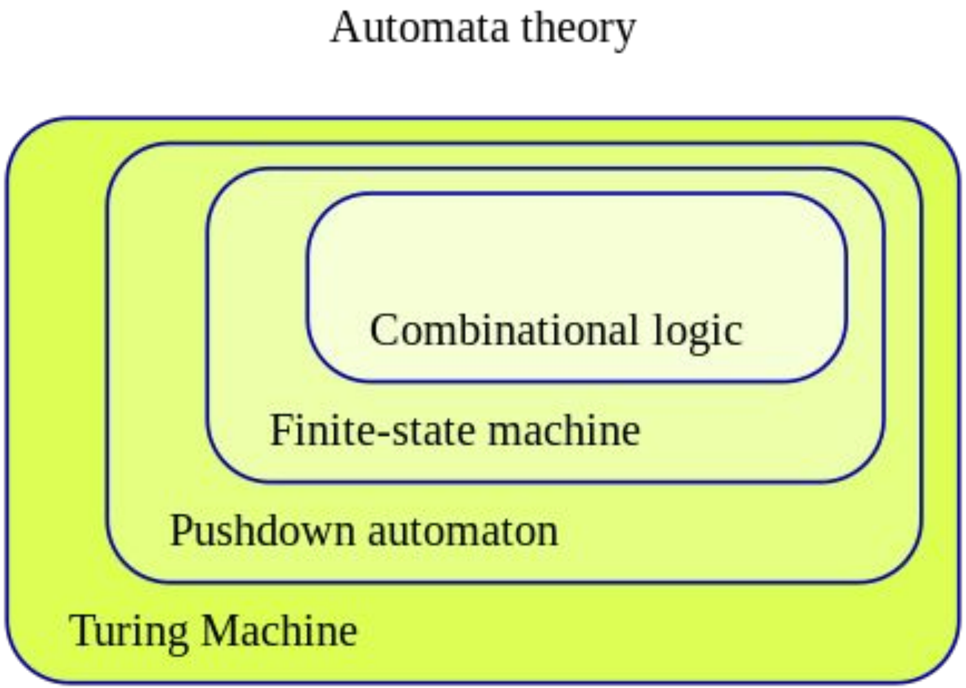
Automata Theory

It is the study of abstract machines and their properties, providing a mathematical notion of a “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

Automata Theory



Automata Theory

Automata is studied for:

- checking behavior of digital circuits

- checking large body of text like a collection of web pages, to find occurrence of words, phrases, patterns

- designing lexical analyzer of a compiler, that breaks input text into logical units called tokens

Abstract Model

An abstract model is a model of a computer system constructed to allow 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

eg: an abstract machine that models a banking system can have operations like “deposit”, “withdraw”, “transfer”, etc

Basic Concepts of Automata Theory

An **alphabet** is a finite, nonempty set of symbols

Conventionally, we use the symbol Σ for an alphabet

eg: $\Sigma = \{0, 1\}$ is the binary alphabet

$\Sigma = \{a, b, c, \ldots., z\}$ is the set of all lower-case letters

Basic Concepts of Automata Theory

A **string** is a finite sequence of symbols chosen from some alphabet

For eg, 01001 is a string from the binary alphabet $\Sigma = \{0, 1\}$

The string 101 is another string chosen from this alphabet

Basic Concepts of Automata Theory

The **empty string** is the string with zero occurrences of symbols

It is denoted by ε (epsilon)

This string may be chosen from any alphabet whatsoever

Basic Concepts of Automata Theory

The **length of a string** w, denoted by |w|, is the number of positions for symbols in w

For every string s, $|s| \geq 0$

$|\varepsilon| = 0$

$|0110| = 4$

Basic Concepts of Automata Theory

The set of all strings of certain length k from an alphabet is the kth **power of that alphabet**

That is, $\Sigma^k = \{w \mid |w| = k\}$

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

$\Sigma^0 = \{\varepsilon\}$

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

$\Sigma^2 = \{00, 01, 10, 11\}$

$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

Basic Concepts of Automata Theory

The set of all the strings over an alphabet Σ is called **kleene closure** of Σ and is denoted by Σ^*

Kleene closure is a set of all the strings over alphabet Σ with length 0 or more

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots\ldots\ldots\ldots$

eg: $\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, \ldots.\}$

Basic Concepts of Automata Theory

The set of all the strings over an alphabet Σ except the empty string is called **positive closure** and is denoted by Σ^+

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots\ldots\ldots\ldots\ldots$

$\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$

Basic Concepts of Automata Theory

A **language** L over an alphabet Σ is the subset of all strings that can be formed out of Σ

That is, a language is a subset of kleene closure over an alphabet Σ

Set of strings chosen from Σ^* defines a language; $L \subseteq \Sigma^*$

Basic Concepts of Automata Theory

eg: Set of all strings over $\Sigma = \{0, 1\}$ with equal number of 0's & 1's:

$$L = \{\varepsilon, 01, 0011, 000111, \ldots\ldots\ldots\}$$

eg: Set of binary numbers whose value is a prime:

$$L = \{10, 11, 101, 111, 1011, \ldots\ldots\}$$

Basic Concepts of Automata Theory

Σ^* is a language for any alphabet Σ

$\emptyset$ is the **empty language** and it is a language over any alphabet

$\{\varepsilon\}$ is a language consisting only of empty string

Basic Concepts of Automata Theory

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

If x is the string of i symbols as $x = a_1a_2a_3...a_i$ & y is the string of j symbols as $y = b_1b_2b_3...b_j$ then xy is the string of i + j symbols as $xy = a_1a_2a_3...a_ib_1b_2b_3...b_j$

Basic Concepts of Automata Theory

eg: If $x = 000, y = 111$

then $xy = 000111, yx = 111000$

ε is identity for concatenation; i.e. for any string w, $\varepsilon w = w\varepsilon = w$

Basic Concepts of Automata Theory

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

eg: If w = abcd

then s = bcd is a suffix of w

s is a **proper suffix** if s ≠ w

Basic Concepts of Automata Theory

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

eg: If w = abcd

then s = abc is a prefix of w

s is a **proper prefix** if s ≠ w

Basic Concepts of Automata Theory

A string s is called a **substring of a string** w if it is obtained by removing 0 or more leading or trailing symbols in w

s is a **proper substring** of w if $s \neq w$

Basic Concepts of Automata Theory

A **problem** is the question of deciding whether a given string is a member of some particular language

That is, 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.”

Practice Questions

(Q1) If $\Sigma = \{a,b,c\}$, find: Σ^1, Σ^2, Σ^3.

(Q2) If $\Sigma=\{0,1\}$, find:

(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 $\{0^i1^j \mid 0 \leq i \leq j\}$.

(e) The language of strings with odd number of 0's and even number of 1's.