1. MATHEMATICAL PRELIMINARIES

1.1 Alphabets, Strings and Languages
- 1.1.1 Alphabet
- 1.1.2 String
- 1.1.3 Languages

1.2 Graphs and Trees
- 1.2.1 Graphs
- 1.2.2 Trees

1.3 Introduction to Formal Proof
- 1.3.1 Deductive proof
- 1.3.2 Reduction to definitions
- 1.3.3 Other theorem forms

1.4 Additional Forms of Proof
- 1.4.1 Proofs about sets
- 1.4.2 Contrapositive
- 1.4.3 Proof by contradiction
- 1.4.4 Counter examples

1.5 Inductive Proofs
- 1.5.1 Inductions on integers
- 1.5.2 Structural inductions
- 1.5.3 Mutual inductions

1.6 Sets, Relations and Functions
- 1.6.1 Sets
- 1.6.2 Relation
- 1.6.3 Function

CHAPTER - 1
MATHEMATICAL PRELIMINARIES

1.1 ALPHABETS, STRINGS AND LANGUAGES

1.1.1 Alphabet

An alphabet is a finite, non-empty set of symbols. It is denoted by Σ.

Examples 1.1

(i) $\Sigma = \{a,b\}$, an alphabet of 2 symbols a and b

(ii) $\Sigma = \{0,1,2\}$ an alphabet of 3 symbols 0, 1 and 2

1.1.2 String

A string (or) word over an alphabet set Σ is a finite sequence of symbols from Σ.

Examples 1.2

(i) If $\Sigma = \{a, b\}$ then *abab*, *aabba*, *aaabba* are all strings over the alphabet $\Sigma = \{a,b\}$

(ii) If $\Sigma = \{a\}$ then *a*, *aa*, *aaa* are all strings over the alphabet $\Sigma = \{a\}$

We use lower case letters *a*,*b*,*c* for elements of Σ and *u*, *v*, *w* for string names.

Operations on strings

1. Length of a string

Let *w* be the string then the "length of the string |*w*|", is the number of symbols composing the string.

Examples 1.3

(i) If $w = abcd$ then $|w| = 4$

(ii) If $x = 01010110$ then $|x| = 8$

(iii) If $y = 0110$ then $|y| = 4$

(iv) $|\varepsilon| = 0$

2. Empty (or) Null string

(i) The string consisting of zero symbols or the length of a string is zero.

(ii) It is denoted by ε or λ.

(iii) $|\varepsilon| = 0 = |\lambda|$

3. Concatenation of string

The concatenation of two strings *w* and *v* is the string obtained by appending the symbols of *v* to the right end of *w*.

ie. $w = a_1 a_2 a_3 \ldots\ldots\ldots\ldots a_m$

$v = b_1 b_2 b_3 \ldots\ldots\ldots\ldots b_n$

then $wv = a_1 a_2 a_3 \ldots\ldots\ldots\ldots a_m b_1 b_2 b_3 \ldots\ldots\ldots\ldots b_n$

Examples 1.4

(i) $x = 010$ $y = 1$

$xy = 0101$ $yx = 1010$.

(ii) x = AL y = GOL

xy = ALGOL

(iii) Empty string is the identity element for concatenation operator ie. $w\varepsilon = \varepsilon w = w$

4. Reverse of a string

(i) The reverse of a string is obtained by writing the symbols in reverse order.

(ii) Let *w* be a string. Then its reverse is w^R

ie. $w = a_1 a_2 \ldots\ldots\ldots\ldots a_m$

$w^R = a_m \ldots\ldots\ldots\ldots a_2 a_1$

Example 1.5

Let $u = 0101011$

$u^R = 1101010$

5. Powers of an alphabet

Let Σ be an alphabet.

Σ^* denotes the set of all strings over the alphabet Σ.

Σ^m denotes the set of all strings over the alphabet Σ of length *m*.

Example 1.6

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

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

$\Sigma^1 = \{0, 1\}$ set of all strings of length one over $\Sigma = \{0, 1\}$

$\Sigma^2 = \{00, 01, 10, 11\}$ set of all strings of length two over $\Sigma = \{0, 1\}$

Σ^3 = {000, 001, 010, 011, 100, 101, 110, 111} set of all strings of length three over Σ = {0,1}

6. Kleene closure

Let Σ be an alphabet. Then "Kleene Closure Σ^*" denotes the set of all strings (including ε, empty string) over the alphabet Σ.

Examples 1.7

(i) If Σ = {*a*} then Σ^* = {ε, *a*, *aa*, *aaa*, ...} i.e.

Σ^0 = {ε}

Σ^1 = {*a*}

Σ^2 = {*aa*}

(ii) If Σ = {0, 1} then Σ^* = {ε, 0, 1, 00, 01, 10, ...}

(iii) Σ= {0} then Σ^* = {ε, 0, 00, 000,...)

$\therefore \Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup ....$

7. Substring

A string *v* appears within another string *w* (w=*uv*) is called "substring of *w*." If *w*=*uv*, then substrings *u* & *v* are said to be prefix and suffix of *w* respectively.

Examples 1.8

(i) *w*=*abbab* be a string over an alphabet Σ = {*a*, *b*} then *a*, *ab*, *abb*, *ba*, *bab*,... are all substrings of *w*.

(ii) *w* = 123

Prefixes = {ε, 1, 12, 123}

Suffixes = {ε, 3, 23, 123}

(iii) *w* = *abbab*

Prefixes = {ε, *a*, *ab*, *abb*, *abba*, *abbab*}

Suffixes = {ε, *b*, *ab*, *bab*, *bbab*, *abbab*}

8. Palindrome

(i) A palindrome is a string, which is same whether written forward (or) backward.

Example 1.9 : madam, malayalam, noon, nun, 121.

(ii) If the length of a palindrome is even, then it can be obtained by concatenation of a string and its reverse.

Example 1.10 : If *u* = 01 u^R = 10.

then even palindrome = 0110

9. Properties of string operations

(i) Concatenation is associative ; that is for all strings u,v and w,

$(uv)\ w = u\ (vw)$

(ii) If u and v are strings, then the length of their concatenation is the sum of the individual lengths, i.e.,

$|uv| = |u| + |v|$.

Example 1.11

$x = abc \qquad y = 123$

$xy = abc123$

$|xy| = 6$ ie $|x| = 3 \quad |y| = 3$

hence $|xy| = |x| + |y|$

1.1.3 Languages

For any alphabet Σ, any subset L of Σ^* is a language. A set of strings from an alphabet is called a "language."

Examples 1.12

(i) *English* : It is a language over $\Sigma=\{a, b, c, ...z\}$

(ii) *Binary strings* : {0, 1, 01, 10, 0101,.....} is a language over $\Sigma=\{0,1\}$

(iii) If $\Sigma = \{a, b\}$ is an alphabet then $\Sigma^* = \{\varepsilon, a, b, aa, ab, ...\}$ is a language.

Notations :

(i) $\{\lambda\}$ (or) $\{\varepsilon\}$: Empty string (or) Null string language.

It is a language over every alphabet and it contains exactly one string ε (or) λ.

(ii) ϕ : Empty language

It contains no strings.

(iii) Σ^* : Universal language

It contains all (finite) string over the alphabet Σ.

Note :

$\phi \neq \{\lambda\}$ ie ϕ has no string where as $\{\varepsilon\}$ (or) $\{\lambda\}$ has one string ε (or) λ.

Operations on language

1. Product (or) concatenation

$L_1 . L_2 = L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$

ie., the concatenation of two languages L_1 and L_2 are set of all strings contained by concatenating any element of L_1 with any element of L_2.

2. Reversal

The reverse of a language is the set of all string reversals. ie., $L^R = \{w^R : w \in L\}$

3. Power

For a given language L, $L^0 = \{\lambda\}$

We define L^n as L concatenated itself *n* times

ie $L^0 = \{x\}$

$L^1 = L$

$L^K = L \,.\, L^{K-1}$

(or)

$L^K = \{x_1 .... x_K : x_i \in L\}$ where *i* ranges from 1 to K.

4. Kleene star (or) star closure

For a language L,

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

(or)

$L^* = L^0 \cup L^1 \cup L^2 ....$

5. Kleene plus (or) positive closure

For a language L,

$$L^+ = \bigcup_{i=1}^{\infty} L^i$$

(or)

$L^+ = L^1 \cup L^2 \cup L^3 \cup ...$

$= L^* - \{\lambda\}$

6. Union

The union of L_1 and L_2 denoted by $L_1 \cup L_2$ is

$L_1 \cup L_2 = \{w: w \in L_1 \text{ or } w \in L_2\}$

7. Intersection

The intersection of L_1 and L_2 denoed by $L_1 \cap L_2$ is

$L_1 \cap L_2 = \{w : w \in L_1 \text{ and } w \in L_2\}$

1.2 GRAPHS AND TREES

The theory of graphs and trees are widely applied in formal languages, compiler writing and artificial intelligence. Hence we focus about their properties as short description.

1.2.1 Graphs

Definition : A graph, denoted by G = (V,E) consists of a finite set of *vertices* (or) *nodes* V and a set E, a pair of vertices called *edges*.

Example 1.13

$V = \{ 1, 2, 3, 4\}$

$e_1 = \{1,2\}$

$e_2 = \{2,3\}$

$e_3 = \{3.4\}$

A *path* in a graph is a sequence of vertices $v_1, v_2, v_3, ... v_k$, $k \geq 1$ such that there is an edge (v_i, v_{i+1}) for each i, $1 \leq i < k$.

The length of the path is k-1.

If $v_1 = v_k$, then the path is said to be *cycle* (because starting and ending at same vertex).

Directed graph (or digraph)

It is also denoted by G = (V, E). Here edges are represented by directed arcs.

1.2.2 Trees

A *tree* (strictly speaking ordered, directed tree) is a digraph satisfying following properties:

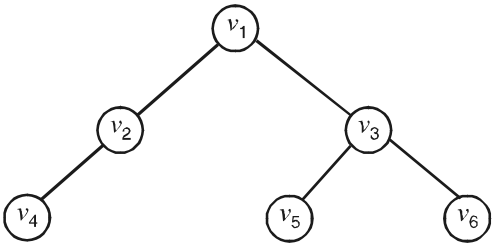
(i) There is one vertex called the root, of the tree which is distinguished from all other vertices and the root has no predecessors.

(ii) There is a directed path from the root to every other vertex.

(iii) Every vertex except the root has exactly one predecessor.

(iv) The successors of each vertex are ordered from left to right.

Example 1.14



v_1 = root

successors of $v_1 = \{v_2, v_3\}$

successors of $v_3 = \{v_5 , v_6\}$

1.3 INTRODUCTION TO FORMAL PROOF

(i) Deductive proofs - the truth of a statement is shown by a detailed sequence of steps and reasons.

(ii) A formal proof or derivation is a finite sequence of sentences (called well-formed formulas in the case of a formal language) each of which is an axiom or follows from the preceding sentences in the sequence by a rule of inference.

(iii) Some computer scientists take the extreme view that a formal proof of the correctness of a program should go hand-in-hand with the writing of the program itself. We doubt that doing so is productive. Some also say that proof has no place in the discipline of programming. The slogan "if you are not sure your program is correct, run it and see" is commonly offered by them.

(iv) Testing programs is surely essential. However, testing goes only so far, since you cannot try your program on every input.

(v) To make your iteration or recursion correct, you need to set up an inductive hypothesis, and it is helpful to reason, formally or informally, that the hypothesis is consistent with the iteration or recursion. This process of understanding the workings of a correct program is essentially the same as the process of proving theorems by induction.

(vi) Automata theory covers methodologies of formal proof. It can be of Either

a) Inductive kind (recursive proofs of a parameterized statement that use the statement itself with lower values of the parameter).

b) Deductive kind (a sequence of justified steps)

1.3.1 Deductive Proofs

A deductive proof consists of a sequence of statements whose truth leads us from some initial statement, called the hypothesis or the given statement(s), to a conclusion statement.

Each step in the proof must follow, by some accepted logical principle, from either the given facts, or some of the previous statements in the deductive proof, or a combination of these.

The hypothesis may be true or false, typically depending on values of its parameters. Often, the hypothesis consists of several independent statements connected by a logical AND.

The theorem that is proved when we go from a hypothesis H to a conclusion C is the statement “if H then C”. We say that C is deduced from H.

An example theorem of the form “if H then C” will illustrate these points.

Theorem : If $x \geq 4$, then $2^x \geq x^2$.

Proof

The hypothesis H is “$x \geq 4$”. This hypothesis has a parameter, x and thus is neither true nor false. Rather, its truth depends on the value of x;

e.g., H is true for $x = 6$ and false for $x = 2$.

The conclusion C is "$2^x \geq x^2$" This statement also uses parameter x and is true for certain values of x and not others.

For example, C is false for $x = 3$, since $2^3=8$, which is not as large as $3^2=9$.

On the other hand, C is true for $x=4$, since $2^4=4^2=16$.

For $x = 5$, the statement is also true, since $2^5=32$ is at least as large as $5^2 = 25$.

Perhaps you can see the intuitive argument that tells us the conclusion $2^x \geq x^2$ will be true whenever $x \geq 4$. We already saw that it is true for $x = 4$.

As x grows larger than 4, the left side, 2^x doubles each time x increases by 1. However, the right side, x^2, grows by the ratio $\left(\frac{x+1}{x}\right)^2$. If $x \geq 4$, then $(x + 1)/x$ cannot be greater than 1.25, and therefore $= (1.25)^2 = 1.5625$. Since $1.5625 < 2$, each time x increases above 4 the left side 2^x grows more than the right side x^2. Thus, as long as we start from a value like $x = 4$ the inequality $2^x \geq x^2$ is already satisfied.

Theorem: If x is the sum of the squares of four positive integers, then $2^x \geq x^2$.

Proof

In deductive proof, we go from a hypothesis H to a conclusion C,

i.e., if H then C.

Step 1 : x is the sum of the squares of four integers. Let a, b, c, d be four integers.

$\therefore x = a^2 + b^2 + c^2 + d^2$

Step 2 : The integers being squared are atleast 1.

$\therefore a \geq 1; b \geq 1; c \geq 1; d \geq 1.$

Step 3 : Since the integers is atleast 1, then its squares is also atleast 1.

$\therefore a^2 \geq 1; \quad b^2 \geq 1; \quad c^2 \geq 1; \quad d^2 \geq 1.$

Step 4 : From Step 1 and Step 3, we can inter that x is sum of four squares and each squares is at least 1.

$\therefore x$ is atleast 1+1+1+1

$\therefore x \geq 4.$

Step 5 : Step 4 is the hypothesis of the previous problem ($2^x \geq x^2$ if $x \geq 4$)

$\therefore$ We can conclude that

$2^x \geq x^2$ or $a^2 + b^2 + c^2 + d^2$

1.3.2 Reduction to Definitions

If the hypothesis does not use familiar terms like integer, multiplication, addition etc., then we can convert all terms in the hypothesis to their definitions.

Theorem: A set S is finite. If there exists an integer *n* such that S has exacty *n* elements. $|S| = n$, where *n* denotes the number of elements in the set S. If S and T are both subsets of some infinite set U, then T is the complement of S (with respect to U) if $S \cup T = U$ and $S \cap T = \phi$. Then T is infinite.

Proof

We can use proof by contradiction. It is a technique where we assume that the conclusion is false. Then use that assumption together with bypothesis, prove the opposite of one of the given statements of the hypothesis. So the only possibility that remains is that the conclusion is true whenever the hypothesis is true.

Here T is finite (because we assume the conclusion is false), but T is infinite = Given.

Let us assume T is finite, along with the statement of the hypothesis, S is finite.

i.e., $|S| = n$ for some integer *n*.

$|T| = m$ for some integer *m*.

Now given statement tells us that

$S \cup T = |S| \cup |T| = n+m$, $n+m$ is a integer it follows U is finite. But it contradicts the given statement U is infinite.

So the conclusion is true whenever the hypothesis is true. Therefore T is infinite.

1.3.3 Other Theorem Forms

1. If - then

The most common forms of if - then statements are if H then C can be rewritten as

(i) H implies C

(ii) H only if C

(iii) C if H

(iv) Whenever H holds, C follows.

So the theorem if $x \geq 4$, $2^x \geq x^2$ can be rewritten as

(a) $x \geq 4$, implies $2^x \geq x^2$

(b) $x \geq 4$ only if $2^x \geq x^2$

(c) $2^x \geq x^2$ if $x \geq 4$

(d) Whenever $x \geq 4$ holds, $2^x \geq x^2$ follows.

2. If - and - only - if statements

The statements of the form “A if and only if B” or “A iff B” has two if - then statements, is “if A then B” and “if B then A”.

Note :

$\lfloor x \rfloor$, floor of real number x, is the greatest integer equal to or less than x.

$\lceil x \rceil$, ceiling of real number x, is the least integer equal to or greater than x.

Theorem : Let x be a real number. Then $\lfloor x \rfloor = \lceil x \rceil$ if and only if x is an integer.

Proof

$\lfloor x \rfloor \leq x$ by definition of floor ..... (1)

$\lceil x \rceil \geq x$ by definition of ceiling ..... (2)

we are given with

$$\lfloor x \rfloor = \lceil x \rceil$$

Substituting (1) in (2), we get.

$$\lceil x \rceil \leq x.$$

Since $\lceil x \rceil \geq x$, by arithmetic inequality we get $\lceil x \rceil = x$.

3. Theorems that appear 'Not' to be 'if-then' statements.

Theorem : $\text{Sin}^2\theta + \text{Cos}^2\theta = 1$.

It does not have any hypothesis.

This theorem can be written in if-then is "if θ is an angle, then $\sin^2\theta + \cos^2\theta = 1$".

1.4 ADDITIONAL FORMS OF PROOF

The following are the additional forms of proofs.

1. Proofs about sets.
2. Proofs by contradiction.
3. Proofs by counter example.

1.4.1 Proofs about Sets

(i) Sets contains symbols to form character strings.

(ii) Sets in automata theory are called as languages.

If E and F are two expressions representing sets, the statement E = F means that two sets represented are same. i.e., every element in the set represented by E is in the set represented by F and viceversa.

Example 1.15

Union of sets obey commutative law. Let S and R be sets. Then $R \cup S = S \cup R$.

If E is the expression in $R \cup S$ and F is the expresion is $S \cup R$, then

E = F, an element x is in E iff x is in F.

Theorem: Distributive law of union over intersection

Let R, S, T be sets. Then prove that

$R \cup (S \cap T) = (R \cup S) \cap (R \cup T)$

Proof

A iff B has two parts

(i) ***if part*** : "if B then A" and

(ii) ***only if part*** : if A then B, which is equivalent form "A only if B".

Let E is the expression in $R \cup (S \cap T)$ and

F is the expression in $(R \cup S) \cap (R \cup T)$

(i) If part : if *x* is in E, *x* is in F.

	Statement	***Justification***
(a)	*x* is in $R \cup (S \cap T)$	Given
(b)	*x* is in R or *x* is in $(S \cap T)$	(a) and definition of union
(c)	*x* is in R or *x* is in both S and T	(b) and definition of intersection
(d)	*x* is in $R \cup S$	(c) and definition of union.
(e)	*x* is in $R \cup T$	(c) and definition of union.
(f)	*x* is in $(R \cup S) \cap (R \cup T)$	(d), (c) and definitin of intersection

(ii) We can also prove it by "only - if" statement.

	Statement	***Justification***
(a)	*x* is in $(R \cup S) \cap (R \cup T)$	Given
(b)	*x* is in $R \cup S$	(a) and definition of intersection
(c)	*x* is in $R \cup T$	(a) and definition of intersection
(d)	*x* is inR or *x* is in both S and T	(b), (c) and reasoning about unions
(e)	*x* is in R or *x* is in $S \cap T$	(d), and definitin of intersections
(f)	*x* is in $R \cup (S \cap T)$	(e) and definition of union.

1.4.2 Contrapositive

It is a logical relationship between two statements.

For example :

A proposition or statement P="Socrates is a man" and proposition Q="Socrates is human" than "if P then Q" implies "Socrates is a man then Socrates is human".

The contrapositive is the reversal of negation of both statements. i.e., if not C, then not H.

"if P then Q" and "if not Q then not P" are logically equivalent. This logical equivalence is shown in Figure 1.1

P	Q	⌉P	⌉Q	P→Q	⌉Q→P
T	T	F	F	T	T
T	F	F	T	F	F
F	F	F	F	T	T
F	F	T	T	T	T

Figure 1.1. Truth table for contrapositive

Example 1.16

if $x \geq 4$, then $2^x \geq x^2$. Then the contrapositive of this statement is "if not $2^x \geq x^2$, then not $x \geq 4$" It is otherwise "not $a \geq b$" is equal to $a < b$.

$\therefore$ not $x \geq 4$ is $x < 4$

not $2^x \geq x^2$ is $2^x < x^2$.

i.e., "if $2^x < x^2$, then $x < 4$".

1.4.3 Proof by Contradiction

(i) It is derived from Latin meaning reduction to the "absurd".

(ii) In proof by contradiction, we assume that the theorem is false and then show that this assumption leads to an obviously false consequence, called a contradiction.

(iii) Another way to prove a statement of the form "if H then C" is to prove the statement "H and not C implies falsehood".

Step a : State by assuming both the hypothesis H and the negation of the conclusion C.

Step b : Compute the proof by showing that something known to be false follows logically from H and C. This form of proof is called proof by contradiction.

Example 1.17

Jack Sees Sill, who has just come in from outdoors. On observing that she is completely dry, he knows that it is not raining.

Proof

His proof = that it is not raining

Assume the negation of conclusion, ie, it is raining, then Jill would be wet. But she is not wet, so it must not be raining.

Theorem: Let S be a finite subset of some infinite set U. Let T be completment of S with respect to U . Then T is infinite.

Proof

H = "S is finite set of U,

U is an infinite set,

T is the complement of S with respect to U"

Conclusion C = "T is infinite"

We proceed to prove the theorem by proof by contrediction.

We assume not C is true is.

T is finite.

from the assuption S is finise, if T is also finite, then U = S∪T is also finite. But hypothesis says U is infinite. Therefore the logical statement is false.

Theorem : Show that is irrational.

$\sqrt{2} = \frac{m}{n}$

Proof : A number is rational if it is a fraction m/n where m, n are integers. i.e it is the ratio of m and n.

A number is irrational if it is not ratinal.

We assume the conclusion is false is is rational.

, where m, n are integers. Both m and n canot be even numbers.

Multiply both sides by n, we obtain

n = m

Square both sides

$2n^2 = m^2$

m^2 is 2 times the integer n^2, so m^2 is even. So m is even. (because square of edd number is odd)

m = 2K.

$2n^2 = (2K)^2 = 4K^2$

$n^2 = 2K^2$

So n^2 is even and hence n is even. Thus we have established both m, n are even, which is contradiction to our assumption.

Hence is irrational.

1.4.4 Counter Examples

(i) It is an exception to a proposition general rule. i.e., Specific instance of the falsity of a universal quantification.

Example 1.18: The statement "all students are lazy".

Proof

Counter example, a hardworking diligent student counters the statement.

Theorem: All primes are odd.

Proof

The integer 2 is prime, but 2 is even.

Theorem: There is no pair of integers a and b such that a mod $b = b$ mod a.

Proof

Let us assume $a < b$.

a mod $b = a$ $\qquad$ $a = qb + r$

$a = 0 \times b + a$. $\qquad$ q = quotient

$\qquad\qquad\qquad\quad$ r = remainder.

But b mod $a < a$, is between 0 and a–1. Thus when $a < b$, b mod $a < a$ mod b, so

a mod $b = b$ mod a is impossible.

It is same argument for $a > b$ also.

Consider $a = b$.

a mod $b = b$ mod $a = 0$ (x mod $x = 0$)

by counter example, let us take $a = b = 2$,

a mod $b = b$ mod $a = 0$

i.e., 2 mod 2 = 2 mod 2 = 0.

1.5 INDUCTIVE PROOFS

1.5.1 Induction on Integers

Proof by Induction is a technique by which the truth of a number of statements can be inferred from the truth of a few specific instances.

Suppose, let P(n) be a statement about a non-negative integer n.

The principle of mathematical induction is that P(n) follows from :

(a) P(1)

(b) P (n-1) implies P(n) for all $n \geq 1$.

Condition(a) is called *basis* and the condition(b) is called *inductive step*, because it connects P_n with P_{n+1}

Example 1.19

Prove by mathematical induction :

$$1+2+3+..... + n =$$

Solution:

Let P(n) = 1+2+3+.....+n =

Basis step :

For n = 1

L.H.S = 1 and R.H.S = = 1

Inductive step :

Assume the result for $n = k$

$$1+2+3+..... +k = \frac{k(k+1)}{2}$$

We have to prove the result for $n = k+1$.

$$1+2+3 + ... + k+k+1 =$$

$$= (k+1)\left[\frac{k}{2}+1\right]$$

$$= (k+1)\left[\frac{(k+2)}{2}\right]$$

$$= \frac{(k+1)(k+1+1)}{2}$$

=

Thus by induction, it is true for all n.

Example 1.20

Prove by induction :

Solution :

Let P(n) : $1^2+2^2+3^2 + ... + n^2 = \frac{n(n+1)(2n+1)}{6}$

Basis step:

For $n = 1$

L.H.S. = 1

R.H.S =

Inductive step:

Assume the result for $n = k$

$1^2+2^2 +..... + k^2 = \frac{k(k+1)(2k+1)}{6}$

We have to prove the result for $n = k + 1$

$1^2+2^2+ ..... + k^2+(k+1)^2 =$

$$= (k+1)\left[\frac{k(2k+1)}{6}+(k+1)\right]$$

$$= (k+1)\left[\frac{2k^2+k+6k+6}{6}\right]$$

$$= \frac{(k+1)(2k^2+7k+6)}{6}$$

$$= (k+1)\frac{(2k^2+4k+3k+6)}{6}$$

$$= (k+1)\frac{(2k(k+2)+3(k+2)}{6}$$

$$= \frac{(k+1)(k+2)\ .\ (2k+3)}{6}$$

Hence it is proved for all values of *n*.

Example 1.21 : If $x \leq 4$, then $2^x \leq x^2$.

Solution :

Basis step :

If $x = 4$, then 2^x and x^2 are both 16.

thus $2^4 \geq 4^2$ holds.

Inductive step :

$\frac{1}{x}$

We need to prove for $x + 1$

$2^{|x+1|} \geq 1x + 11^2$

Let $s(x)$ tells $2^x \geq x^2$.

We need to show $s(x+1)$ is:

$2^{(x+1)} \geq (x+1)^2$.

$2^{x+1} = 2 \text{ x } 2^x. \geq 2x^2$ (because $2^x \geq x^2$)

we need to prove

$2x^2 \geq (x+1)^2$

$2x^2 \geq x^2+2x+1$

$\therefore\ x^2 \geq 2x+1$ ..... (1)

dividing (1) by x, we get

$x \geq 2+$ ..... (2)

since $x \geq 4$, we know $1/x \leq 1/4$

LHS is $x \geq 4$, where as the

RHS is 2+0.25 = 2.25.

$\therefore 2x^2 \geq (x+1)^2$ for x ≥ 4.

$\therefore s(x+1)$ is true.

Thus $2^{x+1} \geq (x+1)^2$.

1.5.2 Structural Inductions

(i) In automata theory, there are several recursively defined structures about which we need to prove statements.

(ii) The examples are trees and expressions.

(iii) Like inductions, all recursive definitions have a basis case, where one or more elementary structures are defined, and an inductive step, where more complex structures are defined in terms of previously defined structures.

(iv) Structural induction is a proof method that is used in mathematical logic, computer science, graph theory, and some other mathematical fields. It is a generalization of mathematical induction.

(v) A recursive definition or inductive definition is one that defines something in terms of itself (that is, recursively), in a useful way.

Example 1.22

Let us take the recursive definition of a tree:

Basis: A single node is a tree, and that node is the root of the tree.

Induction:

If $T_1, T_2, \ldots, T_k$ are trees, then we can form a new tree as follows:

(i) Begin with a new node N, which is the root of the tree.

(ii) Add copies of all the trees $T_1, T_2, \ldots, T_k$.

(iii) Add edges from node N to the roots of each of the trees $T_1, T_2, \ldots, T_k$.

Figure 1.2 Inductive construction of a tree

Figure shows the inductive construction of a tree with root N from k smaller trees.

Example 1.23

Let us take expressions using the arithmetic operators + and *, with both numbers and variables allowed as operands.

Basis : Any number or letter (i.e., a variable) is an expression.

Induction : If E and F are expressions, then so are E + F, E*F, and (E).

For example, both 2 and *x* are expressions by the basis. The inductive step tells us *x*+2, (*x* + 2) and 2*(*x* + 2) are all expressions. Notice how each of these expressions depends on the previous ones being expressions.

When we have a recursive definition, we can prove theorems about it using the following proof form, which is called structural induction. Let S(X) be a statement about the structures X that are defined by some particular recursive definition.

(i) As a basis, prove S(X) for the basis structure(s) X.

(ii) For the inductive step, take a. structure X that the recursive definition says is formed from $Y_1, Y_2, \ldots\ldots\ldots, Y_k$. Assume that the statements $S(Y_1)$, $S(Y_2)$, ……., $S(Y_k)$, and use these to prove S(X).

Our conclusion is that S(X) is true for all X.

The following Theorem gives the facts and proof for trees and expressions.

Theorem : Every tree has one more node than it has edges.

Proof

The formal statement S(T) we need to prove by structural induction is: “if T is a tree, and T has *n* nodes and *e* edges, then $n = e + 1$”.

Basis : The basis case is when T is a single node. Then $n = 1$ and $e = 0$, so the relationship $n = e + 1$ holds.

Induction : Let T be a tree built by the inductive step of the definition, from root node N and *k* smaller trees $T_1, T_2, \ldots, T_k$. We may assume that the statements $S(T_i)$ hold for $i = 1, 2, \ldots, k$. That is, let T_i have n_i nodes and e_i edges; then $n_i = e_i + 1$.

The nodes of T are node N and all the nodes of the T_i's. There are thus $1 + n_1 + n_2 + \ldots + n_k$ nodes in T. The edges of T are the *k* edges we added explicitly in the inductive definition step, plus the edges of the T_i's. Hence, T has

$$k + e_1 + e_2 + \ldots\ldots + e_k \qquad \ldots\ldots\ldots\ldots\ldots \quad (1)$$

edges. If we substitute $e_i + 1$ for n_i in the count of the number of nodes of T we find that T has

$$1 + [e_1 + 1] + [e_2 + 1] + \ldots\ldots\ldots\ldots + [e_k + 1] \qquad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

nodes. Since there are *k* of the "+1" terms in (1), we can regroup (2) as

$$k + 1 + e_1 + e_2 + \ldots\ldots + e_k \qquad \ldots\ldots\ldots\ldots\ldots (3)$$

This expression is exactly 1 more than the expression (1) that was given for the number of edges of T. Thus, T has one more node than it has edges.

Theorem : Every expression has an equal number of left and right parentheses.

Proof

Formally, we prove the statement S(G) about any expression G that is defined by the recursion Example described earlier: the numbers of left and right parentheses in G are the same.

***Basis* :** If G is defined by the basis, then G is a number or variable. These expressions have 0 left parentheses and 0 right parentheses, so the numbers are equal.

Induction : There are three rules whereby expression G may have been constructed according to the inductive step in the definition:

(i) G = E + F.

(ii) G = E * F.

(iii) G = (E)

We may assume that S(E) and S(F) are true; that is, E has the same number of left and right parentheses, say n of each, and F likewise has the same number of left and right parentheses, say m of each. Then we can compute the numbers of left and right parentheses in G for each of the three cases, as follows:

(i) If G = E + F, then G has $n + m$ left parentheses and $n + m$ right parentheses; n of each come from E and m of each come from F.

(ii) If G = E * F, the count of parentheses for G is again $n + m$ of each, for the same reason as in case (i).

(iii) If G = (E), then there are $n + 1$ left parentheses in G -- one left parenthesis is explicitly shown, and the other n are present in E. Likewise, there are $n + 1$ right parentheses in G; one is explicit and the other n are in E.

In each of the three cases, we see that the numbers of left and right parentheses in G are the same. This observation completes the inductive step and completes the proof.

1.5.3 Mutual Inductions

Sometimes, we cannot prove a single statement by induction, but rather need to prove a group of statements $S_1(n)$, $S_2(n)$,……., $S_k(n)$ together by induction on n. Automata theory provides many such situations. In the following example we sample the common situation

where we need to explain what an automaton does by proving a group of statements, one for each state. These statements tell under what sequences of inputs the automaton gets into each of the states.

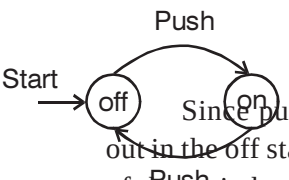
Strictly speaking, proving a group of statements is no different from proving the conjunction (logical AND) of all the statements. For instance, the group of statements $S_1(n)$, $S_2(n)$,……., $S_k(n)$ could be replaced by the single statement

$S_1(n)$ AND $S_2(n)$ AND ……. AND $S_k(n)$

However, when there are really several independent statements to prove, it is generally less confusing to keep the statements separate and to prove them all in their own parts of the basis and inductive steps. This sort of proof is called mutual induction. We will illustrate the necessary steps for a mutual recursion.

Example 1.24

Let us take the on/off switch, which can be represented as an automaton. The automaton itself is reproduced as given below.



Since pushing the button switches the state between on and off, and the switch starts out in the off state, we expect that the following statements will together explain the operation of the switch:

S_1 (n): The automaton is in state off after n pushes if and only if n is even.

S_2 (n): The automaton is in state on after n pushes if and only if n is odd.

We might suppose that S_1 implies S_2 and vice-versa, since we know that a number n cannot be both even and odd. However, what is not always true about an automaton is that it is in one and only one state. It happens that the automaton is always in exactly one state, but that fact must be proved as part of the mutual induction.

We give the basis and inductive parts of the proofs of statements $S_1(n)$ and $S_2(n)$ below. The proofs depend on several facts about odd and even integers: if we add or subtract 1 from an even integer, we get an odd integer, and if we add or subtract 1 from an odd integer we get an even integer.

Basis : For the basis, we choose $n = 0$. Since there are two statements, each of which must be proved in both directions (because S_1 and S_2 are each "if-and-only-if" statements), there are actually four cases to the basis, and four cases to the induction as well.

(i) [S_1; If] Since 0 is in fact even, we must show that after 0 pushes, the automaton is in state *off*. Since that is the start state, the automaton is indeed in state *off* after 0 pushes.

(ii) [S_1; Only-if] The automaton is in state *off* after 0 pushes, so we must show that 0 is even. But 0 is even by definition of "even", so there is nothing more to prove.

(iii) [S_2; If] The hypothesis of the "if" part of S_2 is that 0 is odd. Since this hypothesis H is false, any statement of the form "if H then C" is true, which has discussed earlier. Thus, this part of the basis also holds.

(iv) [S_2; Only-if] The hypothesis, that the automaton is in state on after 0 pushes, is also false, since the only way to get to state on is by following an arc labeled Push, which requires that the button be pushed at least once. Since the hypothesis is false, we can again conclude that the if-then statement is true.

Induction : Now, we assume that $S_1(n)$ and $S_2(n)$ are true, and try to prove $S_1(n + 1)$ and $S_2(n + 1)$. Again, the proof separates into four parts.

(i) [$S_1 (n + 1)$; If] The hypothesis for this part is that $n + 1$ is even. Thus, n is odd. The "if" part of statement $S(n)$ says that after n pushes, the automaton is in state on. The arc from on to off labeled Push tells us that the $(n + 1)$st push will cause the automaton to enter state off. That completes the proof of the "if" part of $S_1(n + 1)$.

(ii) [$S_1(n + 1)$; Only-if] The hypothesis is that the automaton is in state off after $n + 1$ pushes. Inspecting the automaton tells us that the only way to get to state off after one or more moves is to be in state on and receive an input push. Thus, if we are in state off, after n+1 pushes, we must have been in state on after n pushes. Then, we may use the "only-if" part of statement $S_2(n)$ to conclude that n is odd. Consequently, $n + 1$ is even, which is the desired conclusion for the only-if portion of $S_1(n + 1)$.

(iii) [$S_2(n$+1); If] This part is essentially the same as part (i), with the roles of statements S_1 and S_2 exchanged, and with the roles of "odd" and : "even" exchanged.

(iv) [$S_2(n + 1)$; Only-if] This part is essentially the same as part (ii), with the roles of Statements S_1 and S_2 exchanged, and with the roles of "odd" and "even" Exchanged.

Each of the statements must be proved separately in the basis and in the inductive step.

If the statements are "if-and-only-if," then both directions of each statement must be proved, both in the basis and in the induction.

1.6 SETS, RELATIONS AND FUNCTIONS

1.6.1 Sets

(i) A *set* is a well defined collection of elements, without repetition (e.g.) L = {*a*, *b*, *c*, *d*} is a set of four elements.

(ii) We denote the set by $\{x \in A : P(x)\}$. That is the set of elements x such that $P(x)$ is true.

(iii) The *empty* (or) *null set* contains zero elements and denoted by ϕ.

(iv) A set is *finite* if it has finite number of elements. otherwise a set is *infinite*.

L = {0,1} - finite set

L = {1, 2, 3.....} = set of all natural numbers-infinite set

(v) A is a *subset* of B, $A \subseteq B$, if every element of A is an element of B.

(vi) Two sets A and B are *equal* (A=B) if and only if $A \subseteq B$ and $B \subseteq A$.

(vii) If $A \subseteq B$ and $A \neq B$, then A is *proper subset* of B.

Set operations

1. ***Union*** **$A \cup B$**

   $\{x : x \in A \text{ or } x \in B\}$ - set of all elements that are in A (or) in B.

2. ***Intersection*** **$A \cap B$**

   $\{x : x \in A \text{ and } x \in B\}$ - set of all elements that are in both A and B.

   If $A \cap B = \phi$, A and B are disjoint.

3. ***Difference*** **A–B**

   $\{x : x \in A \text{ and } x \in B\}$ - set of all elements that are in A but are not in B.

4. ***Cartesian product*** **A x B**

   $\{(x,y) : x \in A \text{ and } y \in B\}$ - set of all possible ordered pairs (x,y) with $x \in A$ and $y \in B$.

5. ***Power set***

   The set 2^A of all subsets of a set A is the power set of A.

Example 1.25

Let A = {1, 2}

B = {2, 3}

(a) $A \cup B = \{1, 2\} \cup \{2, 3\}$

$= \{1, 2, 3\}$

(b) $A \cap B = \{1, 2\} \cap \{2, 3\}$

$= \{2\}$

(c) $A - B = \{1, 2\} - \{2, 3\}$

$= \{1\}$

(d) $A \times B = \{1, 2\} \times \{2, 3\}$

$= \{(1, 2), (1, 3), (2, 2), (2, 3)\}$

(e) $2^A = 2^{\{1, 2\}}$

$= \{(\phi), (1), (2), (1, 2)\}$

1.6.2 Relation

A binary relation R from set A into another set B, is a subset of A x B.

ie., R is said is be a relation if and only if $R \subseteq A \times B$.

Example 1.26

$f = R = \{(1, 3), (2, 4)\} \subseteq A \times B$.

then $f(1) = 3$

$f(2) = 4$.

Properties of relation

Let $R \subseteq A \times A$ be a relation.

1. R is *"reflexive"* if $(a, a) \in R$ for each $a \in A$
2. R is *"symmetric"* if $(b,a) \in R$ whenever $(a, b) \in R$.
3. R is *"antisymmetric"* if whenever (a,b) and (b,a) are both in R, then $a = b$,
4. R is *"transitive"* if $(a,b) \in R$ and $(b,c) \in R$ then $(a,c) \in R$.
5. R is an *"equivalence relation"* if it is
   a) reflexive
   b) symmetric
   c) transitive
6. R is *"partial order"* if it is
   a) reflexive
   b) antisymmetric
   c) transitive
7. R is *"total order"* if it is a partial order and if, for all distinct a and b in A either (a, b) (or) (b, a) is in R.

1.6.3 Function

A function or a map from a set X into a set Y is a rule which assigns to every element x in X a unique element in Y which is denoted by $f(x)$ i.e. $f : X \rightarrow Y$

* $f : A \rightarrow B$ is *"one to one (injective)"* if distinct $a, a^1 \in A$,

$f(a) \neq f(a^1)$.

* $f : A \rightarrow B$ is *"onto (surjective)"* if, for all $b \in B$, there is some $a \in A$ such that $f(a) = b$
* A *"total function f"* is *"bijective"* if it is both *"one to one"* and *"onto"*

Example 1.27

Let A= {1, 2} B = {3, 4}

f_1 = {(1, 3), (2, 4)} is bijective (because it is both *"one to one"* and *"onto"*)

f_2 = {(1,4), (2,4)} is neither *"one to one"* nor *"onto"*