Puspanjali
Date / /
Page

Name :- Sabita Dandi
Semester :- Second (2nd)
Roll No :- 28
Subject :- Discrete Structure (CSC-152)

Puspanjali

Teacher's Signature

Course Structure

Unit - 1 : Logic, Induction and Reasoning [10 hr.]

1.1 : Logic
1.2 : Induction and Reasoning

Unit - 2 : Finite State Automata (FSA) [10 hr.]

2.1 : Deterministic Finite State Automata (DFA)
2.2 : Non-Deterministic Finite State Automata (NFA)
2.3 : Regular expression
2.4 : Language and grammar

Unit - 3 : Recurrence relation [8 hr.]

3.1 : Recurrence relation
3.2 : Solution of Recurrence relation
3.3 : Some Theorem

Unit - 4 : Graph Theory [15 hr.]

4.1 : Introduction
4.2 : Graph connectivity
4.3 : Euler and Hamilton Path
4.4 : Shortest Path Algorithm
4.5 : Planner graph
4.6 : Graph colouring
4.7 : Tree and spanning tree
4.8 : Network Flow Problem

Question Pattern in Exam:

Full Marks - 80

Pass Marks - 32

Group	A	[10x2 = 20]
Group	B	[5x4 = 20]
Group	C	[5x8 = 40]

"No choice is available."

Text book: Kenneth H. Rosen, "Discrete Mathematical Structure with Application to Computer Science". [6th edition]

- Reference - Joe. L. Molt, "Discrete Mathematics for Computer Scientist & Mathematicians". [2nd edition] [Indian book]

Unit-1

Introduction:

Discrete Mathematics deals with discrete objects. Discrete objects are those objects that can be counted and are not connected. For example: houses, tree, desk, integer, etc. So dealing with these object requires different concepts like counting techniques, knowledge of different discrete structure that are needed to understand what exactly the discrete structure is like set, relation, graph, etc.

Discrete Mathematics is also called finite mathematics, is the study of mathematical structure that are fundamentally discrete.

Logic:

The term Logic came from Greek word 'Logus' which is sometimes translated as discourse or reason. In the field of Artificial Intelligence (AI), logic is defined as the representation language of knowledge. Hence, logic is called as the language of reasoning. Logic needs some rules so that we can apply those rules for specifying the meaning of mathematical statement. There are lots of application of logic in the field of computer science. For example; designing the circuit, programming, program verification, etc.

Proposition:

- Proposition is the fundamental concept in Logic.
- A proposition is the declarative sentence (sentence that declares the fact) that is either true or false but not both.

For example:-

$2+2=5$ (false)

$7-1=6$ (True)

Kathmandu is capital of Nepal (True)

Today is friday (True if today is friday, otherwise false)

Following sentence are not propositions:

i) What time is it?

ii) Who are you?

iii) $x>5$

iv) $x+1=2$

Here, sentence (i) and (ii) are not propositions because they are not declarative sentence. Sentence (iii) & (iv) are not propositions because they are neither true nor false.

- Propositions are denoted conventionally by using small letters like p, q, r, s, ... These letters are called propositional variables.
- The truth value of propositions are denoted by T if it is true propositions and by F for false propositions.

- The logic that deals with proposition are called propositional calculus and propositional logic.

Logical operators/connectives and compound propositions:

Logical operators are used to construct mathematical statement having one or more propositions. The propositions obtained after combining the one or more propositions is called compound propositions. The truth table is used to get the relationship between truth values of propositions.

Negation (Not):-

- Let P be proposition. The negation of P is denoted by ¬P (also denoted by $\bar{P}$), is the statement:
  "It is not the case that P."
- The proposition ¬P is read as "not P."
- The truth value of negation of P, is the opposite of the truth value of P.

Example 1:

P: Today is friday.

Negation is ¬P: It is not the case that today is friday.

More simply ¬P: Today is not friday.

Truth table

P	¬P
T	F
F	T

Fig: Truth Table for negation of P.

Teacher's Signature

Conjunction (and):-

- Let p and q be two propositions. The conjunction of p and q is denoted by $p \wedge q$ is the proposition "p and q."
- The conjunction $p \wedge q$ is true when both p and q are true and is false otherwise.

Example 1:

If we have the propositions:

p: Ram is intelligent.

q: Ram is diligent.

The conjunction of these propositions p and q is:

$p \wedge q$: Ram is intelligent and diligent.

This proposition $p \wedge q$ is true when Ram is intelligent and he is diligent also, false otherwise.

Example 2:

p: Today is friday.

q: It is raining today.

$p \wedge q$: Today is friday and it is raining today.

This proposition $p \wedge q$ is true on rainy friday and is false on any day that is not friday and on friday when it doesnot rain.

Truth table:

P	q	P∧q
T	T	T
T	F	F
F	T	F
F	F	F

Fig: TT for conjunction

Disjunction (OR):-

- Let P and q be two propositions. The disjunction of P and q is denoted by PVq, is the proposition "P or q".
- The disjunction pvq is false when both p and q are false and is true otherwise.

Example 1:

If we have two propositions;

P: Ram is intelligent

q: Ram is diligent.

The disjunction of these propositions P and q is:

PVq: Ram is intelligent or diligent.

This proposition is false only when Ram is not intelligent and not diligent, true otherwise.

Truth table:

p	q	pvq
T	T	T
T	F	T
F	T	T
F	F	F

Fig: TT for disjunction

Exclusive OR:-

- Let p and q be propositions. The exclusive or of p and q is denoted by $p \oplus q$, is the proposition that is true when exactly one of p and q is true and false otherwise.
- As opposed to disjunction which is inclusive, the general meaning of the english sentence can be used to know whether 'or' is used inclusive or exclusive.

Example:

If we have the propositions;

p: Ram drink coffee in the morning.

q: Ram drink tea in the morning.

The exclusive or of these proposition is:

$p \oplus q$: Ram drink coffee or tea in the morning.

Truth table:

P	q	P⊕q
T	T	F
T	F	T
F	T	T
F	F	F

Fig: ⊕ for exclusive or.

Conditional statement / implication (→):-

- Let P and q be propositions. The conditional statement P→q is the propositions "if P then q."
- The conditional statement P→q is false when P is true and q is false and true otherwise. In conditional statement P→q, P is called the hypothesis or anticedent or premise and q is called conclusion or consequences.
- Conditional statement is also called implication.
- A variety of terminology is used to express P→q like:

i) "if P, then q"
ii) "P implies q"
iii) "P only if q"
iv) "P is sufficient for q"
v) "q when P"
vi) "q whenever P"
vii) "q follow from P"

Example:

P: "Today is sunday."
q: "It is hot"

Then implication or conditional statement can be
$p \to q$: if today is sunday, then it is hot.
or, Today is sunday only if it is hot today.

Truth table:

P	q	$P \to q$
T	T	T
T	F	F
F	T	T
F	F	T

Fig: TT for conditional statement.

Converse, contrapositive and inverse:

We can form some new conditional statements starting with a conditional statement $p \to q$.

Some of the related conditional statements formed from $p \to q$ are:

Converse: $q \to p$
Contraposition: $\neg q \to \neg p$
Inverse: $\neg p \to \neg q$

Converse:

The proposition $q \to p$ is called the converse of $p \to q$.

Contrapositive:

The proposition $\neg q \to \neg p$ is contrapositive of $p \to q$.

Inverse:

The proposition $\neg p \to \neg q$ is inverse of $p \to q$.

Note: When two compound propositions always have the same truth value we call them equivalent.

Conditional statement and its contrapositive are equivalent:

Proof:

Now using truth table:

p	q	$p \to q$	$\neg p$	$\neg q$	$\neg q \to \neg p$
T	T	T	F	F	T
T	F	F	F	T	F
F	T	T	T	F	T
F	F	T	T	T	T

→ equivalent statement ←

Fig: Showing conditional statement and its contrapositive are logically equivalent

Converse and Inverse of conditional statement are equivalent but neither is equivalent with original conditional statement:

P	q	$P \to q$	$q \to P$	$\neg P$	$\neg q$	$\neg P \to \neg q$
T	T	T	T	F	F	T
T	F	F	T	F	T	T
F	T	T	F	T	F	F
F	F	T	T	T	T	T

equivalent

Example:

Q. What are the contrapositive, the converse and inverse of conditional statement:
"if it is raining, then home team wins."
P q

⇒ Contrapositive: $(\neg q \to \neg P)$: if home team doesnot win, then it is not raining.

Converse: $(q \to P)$: "if home team wins, then it is raining."

Inverse: $(\neg P \to \neg q)$: "if it is not raining, then home team does not wins."

Biconditionals (↔):-

- Let p and q be propositions. The biconditional statement $p \leftrightarrow q$ is the proposition "p if and only if q".
- The biconditional statement $p \leftrightarrow q$ is true when p and q have same truth values and is false otherwise.
- Biconditional statements are also called bi-implication.
- Note that the statement $p \leftrightarrow q$ is true when both conditional statements $p \leftrightarrow q$ and $q \leftrightarrow p$ are true and false otherwise.
- Some of the terminologies used for biconditionals are:-
  - "p if and only if q"
  - "if p then q, and conversely"
  - "p is necessary and sufficient for q."

Example:

1) p: You can take the flight
   q: You buy the ticket
   $p \leftrightarrow q$: You can take the flight if and only if you buy the ticket.

2) p: "today is sunday"
   q: "It is hot today"
   $p \leftrightarrow q$: "Today is sunday if and only if it is hot today."

Truth table:

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Fig: TT for biconditional $p \leftrightarrow q$

Teacher's Signature

Prove that $p \leftrightarrow q$ has same truth value as $(p \rightarrow q) \wedge (q \rightarrow p)$ using truth table.

P	q	$p \rightarrow q$	$q \rightarrow p$	$p \leftrightarrow q$	$(p \rightarrow q) \wedge (q \rightarrow p)$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	T	F	F	F
F	F	T	T	T	T

equivalent

Truth table for compound proposition:-

- We can use the connectives to build up complicated compound proposition involving any number of propositional variable.
- We can use the truth tables to determine the truth value of these compound proposition.

For example: Given a compound proposition $(P \vee \neg q) \rightarrow (p \wedge q)$

The truth table for this compound proposition is:

P	q	$\neg q$	$P \vee \neg q$	$p \wedge q$	$(P \vee \neg q) \rightarrow (p \wedge q)$
T	T	F	T	T	T
T	F	T	T	F	F
F	T	F	F	F	T
F	F	T	T	F	F

Puspanjali
Date / /
Page

Precedence of logical operator:-

- We generally use parenthesis to specify the order in which logical operator in compound proposition are to be applied. For example: $(p \vee q) \wedge (\neg r)$ is the conjunction of $p \vee q$ and $\neg r$.
- However to reduce the number of parenthesis, we specify that negation operator is applied before all the operator. This means that $\neg p \wedge q$ is conjunction of $\neg p$ and q i.e. $(\neg p) \wedge q$ but not the negation of the conjunction p and q i.e. $\neg(p \wedge q)$.
- Another general rule of precedence is that the conjunction operator takes precedence over disjunction operator.
  For example:- $p \wedge q \vee r$ means that $(p \wedge q) \vee r$ rather than $p \wedge (q \vee r)$.
- Finally it is an accepted rule that conditional and biconditional operator $\rightarrow$, $\leftrightarrow$ have lower precedence than conjunction and disjunction operator.
  For example:- $p \wedge q \rightarrow r$ is same as $(p \wedge q) \rightarrow r$
- The following table displays precedence level of logical operator:

Operator	Precedence
$\neg$	1
$\wedge$	2
$\vee$	3
$\rightarrow$	4
$\leftrightarrow$	5

Teacher's Signature

Translating of english sentence into logical oper expression:

- There are many reasons to translate english sentence into expression involving propositional variable and logical connectives.

i) Translating the english sentence into logical expression removes ambiguity.

ii) By translating english statement into logical expression, we can use these logical expression to analyze, determine their truth values, manipulate them and use the rule of inference to reason about them.

- The process of translating english sentence into logical expression consist of following steps:

a) First we restate the given sentence into building block sentence.

b) Represent each sentence by propositional variable.

c) Connect each propositional variable with the appropriate connectives.

Example 1:

If it is raining, then home team wins.

p : It is raining

q : home team wins

Logical expression: $p \rightarrow q$

Puspanjali
Date / /
Page

Example 2:
You can access the internet from campus only if you are CSIT student or staff of campus.

⇒ p: "you can access the internet from campus".
q: "you are CSIT student"
s: "you are staff of campus".

Logical expression:

$$p \rightarrow (q \vee s)$$

Example 3:
You can join the army when your are not under 5feet tall or you are older than 21.

⇒ p: you can join the army
q: you are not under 5 feet tall
r: you are older than 21

Logical expression: $(q \vee r) \rightarrow p$

Just for practice:
Translate given logical expression into english sentence.

a) $\neg p$ b) $r \wedge \neg p$ c) $(\neg r \vee p) \vee q$

When,
p = "It rained last night"
q = "The sprinkles came on last night"
r = "The lawn was wet this morning."

Teacher's Signature

a) $\neg p$: It didn't rain last night.

b) $(r \wedge \neg p)$: The lawn was wet this morning and it didn't rain last night.

c) $(\neg r \vee p) \vee q$: Either the lawn was not wet this morning or it rained last night or the sprinkles came on last night.

Example:

Let p, q and r be the propositions with truth value T, F, T respectively. Evaluate the following.

i) $\neg r \vee \neg(p \wedge q)$

ii) $\neg(p \vee q) \wedge (\neg r \vee q)$

i) $\neg r \vee \neg(p \wedge q)$
$= \neg r \vee \neg(T \wedge F)$
$= \neg r \vee \neg F$
$= \neg T \vee \neg F$
$= F \vee T$
$= T$

ii) $\neg(p \vee q) \wedge (\neg r \vee q)$
$= \neg(T \vee F) \wedge (\neg T \vee F)$
$= \neg T \wedge (F \vee F)$
$= F \wedge F$
$= F$

Bit operation:

- **Bitwise AND (∧) operation:-**

The bitwise AND perform logical ANDing between two operand in the form of string. The result of ANDing operation is 1 if both bit have value 1, otherwise it is zero.

Let : A = 1001
B = 1100
A∧B = 1000

- **Bitwise OR (∨) operation:-**

The bitwise OR perform logical bitwise ORing between two bit string. The result of ORing operation is 1 if either of the bit have a value 1, otherwise zero.

Let : A = 1001
B = 1101
A∨B = 1101

- **Bitwise XOR (⊕):-**

The bitwise XOR becomes performs logical XORing between two bit string. The result of X-ORing is 1 only if one of the bit have the value 1 otherwise it is zero.

Let A = 1001
B = 1101
A⊕B = 0100

Tautology:

A compound proposition that is always true, no matter what the truth value of proposition that occur in it is called a tautology.

Example:

$P \vee \neg P$ is always true so it is a tautology.

P	¬P	P ∨ ¬P
T	F	T
F	T	T

Contradiction:

A compound proposition that is always false, no matter what the truth value of the proposition that occur in it is called contradiction.

Example:

$P \wedge (\neg P)$ is always false so it is contradiction.

P	¬P	P ∧ (¬P)
T	F	F
F	T	F

Contingency:

A compound proposition that is neither a tautology nor a contradiction is called a contingency.

Example:

$P \rightarrow q$

Logical equivalences:

- Compound proposition that have the same truth value in all possible cases are called logically equivalent.
- In another word, the compound proposition P and q are logically equivalent if $p \leftrightarrow q$ is a tautology. The notation $\equiv$ denotes P and q are logically equivalent.
- The symbol $\Leftrightarrow$ is sometimes also used instead of $\equiv$ to denote logical equivalence.
- One way to determine whether two compound proposition are equivalent is to use a truth table.

Some important logical equivalences:

1.	Identity laws	$P \wedge T \equiv P$ $P \vee F \equiv P$
2.	Domination laws	$P \vee T \equiv T$ $P \wedge F \equiv F$
3.	Idempotent laws	$P \vee P \equiv P$ $P \wedge P \equiv P$
4.	Double negation laws	$\neg(\neg P) \equiv P$
5.	Commutative laws	$P \vee q \equiv q \vee P$ $P \wedge q \equiv q \wedge P$

6. Associative laws $(p \vee q) \vee r \equiv p \vee (q \vee r)$
$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$

7. Distributive laws $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$

8. De-morgan laws $\neg(p \wedge q) \equiv \neg p \vee \neg q$
$\neg(p \vee q) \equiv \neg p \wedge \neg q$

9. Absorption laws $p \vee (p \wedge q) \equiv p$
$p \wedge (p \vee q) \equiv p$

10. Negation laws $p \vee \neg p \equiv T$
$p \wedge \neg p \equiv F$

Rules associated with conditional statement:

$p \rightarrow q \equiv \neg p \vee q$
$p \rightarrow q \equiv \neg q \rightarrow \neg p$
$p \vee q \equiv \neg p \rightarrow q$
$p \wedge q \equiv \neg(q \rightarrow \neg p)$
$(p \rightarrow q) \wedge (p \rightarrow r) \equiv p \rightarrow (q \wedge r)$
$(p \rightarrow r) \wedge (q \rightarrow r) \equiv (p \wedge q) \rightarrow r$
$(p \rightarrow q) \vee (p \rightarrow r) \equiv p \rightarrow (q \vee r)$
$(p \rightarrow r) \vee (q \rightarrow r) \equiv (p \wedge q) \rightarrow r$

Puspanjali
Date / /
Page

Rules associated with biconditionals:

$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$

$(p \leftrightarrow q) \equiv \neg p \leftrightarrow \neg q$

$(p \leftrightarrow q) \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$

$\neg (p \leftrightarrow q) \equiv p \leftrightarrow \neg q$

1. Identity laws

a) $p \wedge T \equiv p$ b) $p \vee F \equiv p$

Verification:

p	$p \wedge T$	$p \vee F$
T	T	T
F	F	F

$p \wedge T \equiv p$ $p \vee F \equiv p$

2. Domination laws

a) $p \wedge F \equiv F$ b) $p \vee T \equiv T$

Verification:

p	$p \wedge F$	$p \vee T$
T	F	T
F	F	T

$p \wedge F \equiv F$ $p \vee T \equiv T$

Puspanjali

Teacher's Signature

3. Idempotent law:

a) $P \vee P \equiv P$

b) $P \wedge P \equiv P$

Puspanjali
Date / /
Page

6. Associative law:

a) $(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$ b) $(P \vee Q) \vee R \equiv P \vee (Q \vee R)$

a) Verification:

P	Q	R	$(P \wedge Q)$	$(P \wedge Q) \wedge R$	$(Q \wedge R)$	$P \wedge (Q \wedge R)$
T	T	T	T	T	T	T
T	T	F	T	F	F	F
T	F	T	F	F	F	F
T	F	F	F	F	F	F
F	T	T	F	F	T	F
F	T	F	F	F	F	F
F	F	T	F	F	F	F
F	F	F	F	F	F	F

equivalent.

b) Verification:

P	Q	R	$(P \vee Q)$	$(P \vee Q) \vee R$	$(Q \vee R)$	$P \vee (Q \vee R)$
T	T	T	T	T	T	T
T	T	F	T	T	T	T
T	F	T	T	T	T	T
T	F	F	T	T	F	T
F	T	T	T	T	T	T
F	T	F	T	T	T	T
F	F	T	F	T	T	T
F	F	F	F	F	F	F

equivalent

Puspanjali
Date / /
Page

Just for practice:
Use the De-morgan's law to express the negation of "Ram has a cellphone and he has a laptop computer."

Sol:

Let p: Ram has a cellphone.
q: Ram has a laptop computer.

The given sentence can be expressed in logical expression as:
$p \wedge q$
Now, to negate a given sentence, we need $\neg(p \wedge q)$.
but we know,
$\neg(p \wedge q) \equiv \neg p \vee \neg q$ (according to De-morgan's law).
So,
$\neg(p \wedge q)$: "Ram does not have a cellphone or he doesn't have a laptop computer.

Proving a logical equivalence:
We can prove a logical equivalence by
i) Truth table
ii) by using the law of logical equivalence.

Example: $\neg(p \vee (\neg p \wedge q)) \equiv \neg p \wedge \neg q$
L.H.S. $\neg(p \vee (\neg p \wedge q))$
$\equiv \neg p \wedge \neg(\neg p \wedge q)$ by using De-morgan's law.
$\equiv \neg p \wedge \neg(\neg p) \vee \neg q$ by using Demorgan's law.
$\equiv \neg p \wedge (p \vee \neg q)$ by using double negation law.
$\equiv (\neg p \wedge p) \vee (\neg p \wedge \neg q)$ by using distributive law.

Teacher's Signature

$\equiv F \vee (\neg P \wedge \neg q)$ since $\neg P \wedge P \equiv F$
$\equiv (\neg P \wedge \neg q) \vee F$ commutative law
$\equiv (\neg P \wedge \neg q)$ identity law

proved.

By truth table;

P	q	¬P	¬q	(¬P∧q)	P∨(¬P∧q)	¬(P∨(¬P∧q))	¬P∧¬q
T	T	F	F	F	T	F	F
T	F	F	T	F	T	F	F
F	T	T	F	T	T	T	F
F	F	T	T	F	F	T	T

equivalent

Showing the compound statement is tautology:-

- To show the statement is tautology, we either use law of logical equivalences to demonstrate that the statement is equivalent to true or we use the truth table.

For example: Show that $(p \wedge q) \rightarrow (p \vee q)$ is a tautology

a) by using truth table:

- We can verify this statement with the help of truth table as follows:

P	q	P∧q	P∨q	(P∧q)→(P∨q)
T	T	T	T	T
T	F	F	T	T
F	T	F	T	T
F	F	F	F	T

From above, the truth value of $(p \wedge q) \rightarrow (p \vee q)$ is always true no matter what the value of p and q.

So, given statement is tautology.

proved.

Teacher's Signature

Q. $(P \wedge Q) \rightarrow (P \vee Q)$

$\equiv \neg(P \wedge Q) \vee (P \vee Q)$ → (law of implication)

$\equiv (\neg P \vee \neg Q) \vee (P \vee Q)$

$\equiv (\neg P \vee P) \vee (\neg Q \vee Q)$ → (distributive law)

$\equiv T \vee T$ → (Negation law)

$\equiv T$

Hence proved.

Q. $(P \wedge Q) \rightarrow P$

$\equiv \neg(P \wedge Q) \vee P$ (law of implication)

$\equiv (\neg P \vee \neg Q) \vee P$ (De-morgan's law)

$\equiv (\neg P \vee P) \vee (\neg Q \vee P)$

$\equiv (P \vee \neg P) \vee \neg Q$

$\equiv T \vee \neg Q$

$\equiv T$

Hence proved!!!

Q. $P \rightarrow (P \vee Q)$

$\equiv \neg P \vee (P \vee Q)$

$\equiv (\neg P \vee P) \vee Q$ (commutative law)

$\equiv T \vee Q$

$\equiv T$

Hence proved!!!

Q. $(P \wedge Q) \rightarrow (P \vee Q)$

P	Q	P∧Q	P∨Q	(P∧Q)→(P∨Q)
T	T	T	T	T
T	F	F	T	T
F	T	F	T	T
F	F	F	F	T

proved!!!

Q. $(P \wedge Q) \rightarrow P$

P	Q	P∧Q	(P∧Q)→P
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	T

proved!!!

Q. $P \rightarrow (P \vee Q)$

P	Q	P∨Q	P→(P∨Q)
T	T	T	T
T	F	T	T
F	T	T	T
F	F	F	T

proved!!!

More examples of tautology:

1) $p \wedge q \rightarrow (p \rightarrow q)$

$\equiv \neg(p \wedge q) \vee (p \rightarrow q)$

$\equiv \neg(p \wedge q) \vee (\neg p \vee q)$

$\equiv (\neg p \vee \neg q) \vee (\neg p \vee q)$ Demorgan's law

$\equiv (\neg p \vee \neg q) \vee (q \vee \neg p)$ Commutative law

$\equiv [(\neg p \vee \neg q) \vee q] \vee \neg p$ Associative law

$\equiv [\neg p \vee (\neg q \vee q)] \vee \neg p$ Associative law

$\equiv (\neg p \vee T) \vee \neg p$ Negation law

$\equiv T \vee \neg p$ Domination law

$\equiv T$ Domination law

Hence given statement is tautology.

2) $[p \wedge (p \rightarrow q)] \rightarrow q$

$\equiv \neg[p \wedge (p \rightarrow q)] \vee q$ Law of implication

$\equiv \neg[p \wedge (\neg p \vee q)] \vee q$ Law of implication

$\equiv [\neg p \vee \neg(\neg p \vee q)] \vee q$ De-morgan's law

$\equiv [\neg p \vee (p \wedge \neg q)] \vee q$ De-morgan's law

$\equiv [(\neg p \vee p) \wedge (\neg p \vee \neg q)] \vee q$ Distributive law

$\equiv [T \wedge (\neg p \vee \neg q)] \vee q$ Negation law

$\equiv (\neg p \vee \neg q) \vee q$ Identity law.

$\equiv \neg p \vee (\neg q \vee q)$

$\equiv \neg p \vee T$

$\equiv T$

Hence given statement is tautology.

Predicates and quantifier:-

- Propositional logic can not adequately express the meaning of statement in Mathematics and in natural language.

For Example:

"Some student in your class room has television in home."

"The person x was born on the city "y"."

These statement can not be represented by using propositional logic to conclude the truth of the statement.

For such statement, we need generalized and powerful type of logic called predicate logic.

Predicates:-

Any declarative statement involving variable often found in Mathematical assertion and in computer program which are neither true nor false when the values of the variables are not specified is called predicates.

For example:

$x > 3$; "$x = y + 3$"

Computer x is under attack by intruder.

- The statement "x is greater than 3" has two part. The first part the variable x, is the subject of statement.
- Second part "is greater than 3" called predicate that refers to property that the subject of statement have.
- We denote the statement "x is greater than 3" by P(x) where

Teacher's Signature

P denotes predicates "is greater than 3" and x is variable.

- We also call P as propositional function where P(x) gives value of P at x.
- Once the value has been assigned to variable x, the statement P(x) becomes a proposition and has a truth value.

For example:

Let P(x) denotes the statement "x>3". If we put x as 2 then we conclude that P(2) is false since 2 is not greater than 3.

- We can also have statements that involves more than one variables.

For example:

$x = y + 3$

We can denote this statement Q(x,y) where x and y are the variables and Q is the predicates. When values are assigned to x and y, the statement Q(x,y) has the truth value.

Predicate Logic:-

The logic involving predicates is called predicate logic or predicate calculus. Predicate logic is generally used to represent more generalized sentence than propositional logic.

More simply,
The area of logic that deals with predicate and quantifier is called predicate logic or predicate calculus.

Example:
Let $Q(x,y)$ denotes the statement "$x = y+3$." What are the truth value of proposition $Q(1,2)$ and $Q(3,0)$.

Sol:
Given, $Q(x,y) : x = y+3$
For $Q(1,2)$, set $x=1$ and $y=2$
$Q(1,2) : 1 = 2+3$ (False)
Similarly,
For $Q(3,0)$, set $x=3$ and $y=0$
$Q(3,0) : 3 = 0+3$ (True)

- In general statement involving the n variable $x_1, x_2 \ldots x_n$ can be denoted by $P(x_1, x_2, \ldots x_n)$.
- The statement of the form $P(x_1, x_2, \ldots, x_n)$ is the value of proposition function P at n-tuple $(x_1, x_2, \ldots, x_n)$ and is called n-place or n-ary predicates.

Definition type ko question

Quantifier :-
Quantifiers are the tools to make the propositional function in to proposition.
- The construction of proposition from the predicate using quantifie is called quantification.
- Quantification express the extent to which predicate is true

Puspanjali
Teacher's Signature

over a range of elements. In english the word all, some, many, none, and few are used in quantification.

- The variable that appear in the statement can take different possible values and all the possible values that the variable can take from a domain called "universe of discourse" or "universal set."

- There are mainly two types of quantifier:
  i) Universal quantifier
  ii) Existential quantifier

1) Universal quantifier:-

- The phrase "for all" denoted by $\forall$ is called universal quantifier.
- The process of converting predicates into proposition using universal quantifier is called universal quantification.
- The universal quantification of $P(x)$, denoted by $\forall x\, P(x)$ is a proposition where $P(x)$ is true for all values of x in the universe of discourse.
- We can represent the universal quantification by using english language like;
  for all x $P(x)$ holds or for every x $P(x)$ holds or for each x $P(x)$ holds.

Example:

Let universe of discourse = {all student of CDCSIT}.
P(x) represents "x takes graphics class."

Here, universal quantification of given predicate is:
$\forall x\, P(x)$: all student of CDCSIT takes graphics class.
: for all student x of CDCSIT, x takes graphics class.

- The universal quantification is conjunction of all proposition that are obtained by assigning the value of the variable in predicate.
i.e. $\forall x\, P(x) = P(x_1) \wedge P(x_2) \wedge P(x_3) \wedge \ldots P(x_n)$
where $(x_1, x_2, \ldots x_n)$ are the values in domain of discourse.

- Let universe of discourse is {Ram, Hari, Sita} then the value of universal quantification is given by,
$P(Ram) \wedge P(Hari) \wedge P(sita)$

- An element for which $\forall x\, P(x)$ is false is called counter example of $\forall x\, P(x)$.

- A statement $\forall x\, P(x)$ is false if and only if $P(x)$ is not always true for every x in the domain. We need a single counter example to show $\forall x\, P(x)$ is false.

Teacher's Signature

Example:

Let $Q(x)$ be the statement "$x < 2$". What is the truth value of quantification $\forall x\, Q(x)$, where domain course of all real number.

Sol:

- $Q(x)$ is not true for every real number x.
- For instance $Q(3)$ is false, that is $x = 3$ is the counter example of statement $\forall x\, Q(x)$: for every x, $x < 2$ where domain of discourse is all real number.

Thus,

$\forall x\, Q(x)$ is false.

Example:

What is the truth value of $\forall x\, P(x)$, where $P(x)$ is the statement $x^2 < 10$ and domain consist of the positive integer not exceeding 4.

Sol:

The statement $\forall x\, P(x)$ is same as conjunction.

$P(1) \wedge P(2) \wedge P(3) \wedge P(4)$

Because $P(4)$ which is the statement $4^2 < 10$ is false so, $\forall x\, P(x)$ is false.

ii) Existential quantifier:-

- The phrase "there exist" denoted by $\exists$ is called existential quantifier.
- The process of converting predicate to proposition during existential quantifier is called existential quantification.
- The existential quantification of $P(x)$, denoted by $\exists x\, P(x)$ is a proposition "there exist an element x in the domain such that $P(x)$ hold i.e. $P(x)$ is true for some values of x in the domain.
- Besides the word "there exists" the other form of representation include:
  "There is an x such that $P(x)$.
  "There is at least one x such that $P(x)$.
  "For some x, $P(x)$.

- The statement $\exists x\, P(x)$ is false if and only if there is no element x in the domain for which $P(x)$ is true i.e. $\exists x\, P(x)$ is false if and only if $P(x)$ is false for every element of the domain.

- The existential quantification $\exists x\, P(x)$ is true when $P(x)$ is true for some x in the domain. So existential quantification $\exists x P(x)$ is same as:
  $\exists x\, P(x) = P(x_1) \vee P(x_2) \vee \ldots P(x_n)$ because this disjunction is true if and only if at least one of $P(x_1), P(x_2), \ldots P(x_n)$ is true.

Teacher's Signature

Example:

Let $P(x): x > 3$

Find the truth value of $\exists x\, P(x)$ where domain consist of all real number.

Soln:

Because $x > 3$ is sometimes true i.e. when $x = 4$.

Hence, $\exists x\, P(x)$ is true.

Other quantifier:

Uniqueness quantifier:-

- It is denoted by $\exists!$ or $\exists_1$. The notation $\exists! x\, P(x)$ states "there exist a unique x such that $P(x)$ is true."
- Other phrases for uniqueness quantification include, "there is exactly one" and there is one and only one."

For example:

"There is a unique student in the class who has studied CPL".

This statement is same as:

"There is a unique student x in the class, x has studied CPL.

Hence, this statement can be represented by uniqueness quantifier as

$\exists_1 x\, C(x)$ where domain = {all student in the class}

Precedence of quantifier:

- The quantifiers $\forall$ and $\exists$ have higher precedence than all logical operators from propositional calculus.

For example:

$\forall x P(x) \vee Q(x)$ is disjunction of $\forall x P(x)$ and $Q(x)$.

- In other word, $(\forall x P(x)) \vee Q(x)$ rather than $\forall x (P(x) \vee Q(x))$.

Bounded and free variable:

- When a variable is assigned a value or associated with quantifier, it is known as bounded variable.
- If the variable is not bounded then it is known as free variable.
- For example;
  - i) $P(x,y)$ – x and y are free variable.
  - ii) $P(3,y)$ – y is free variable
  - iii) $\forall x P(x)$ – x is bounded variable.
  - iv) $\forall x P(x,y)$ – here x is bounded and y is free variable.
- The part of logical expression to which quantifier is applied is called scope of this quantifier.
- Consequently, a variable is free if it is outside the scope of all quantifier.

Logical equivalence involving quantifiers:

- Statement involving predicates and quantifiers are logically equivalent if and only if they have the same truth value no matter which predicates are substituted into those statement and which domain of discourse is used for the variable in these propositional function.

- We use $S \equiv T$ to indicate that two statement S and T involving predicate and quantifier are logically equivalent. For example:

$$\forall x(P(x) \wedge Q(x)) \equiv \forall x P(x) \wedge \forall x Q(x).$$

L.H.S. $\forall x(P(x) \wedge Q(x))$.

domain $= \{x_1, x_2, \ldots, x_n\}$

$$\equiv (P(x_1) \wedge Q(x_1)) \wedge (P(x_2) \wedge Q(x_2)) \wedge (P(x_n) \wedge Q(x_n) \ldots$$
$$\equiv (P(x_1) \wedge P(x_2) \ldots P(x_n)) \wedge (Q(x_1) \wedge Q(x_2) \wedge \ldots Q(x_n)$$
$$\equiv \forall x P(x) \wedge \forall x Q(x).$$

Example:

1. For every computer in lab, computer contains Linux and Window.

$\equiv$ For every computer in lab, computer contains Linux and for every computer in lab, computer contains Windows.

2. There exist.

$\exists x (P(x) \vee Q(x)) \equiv \exists x\, P(x) \vee \exists x\, Q(x)$

There exist some computer in the lab which contains Linux or Windows.

There exist some computer in the lab which contains Linux or there exist some computer in the lab which contains Windows.

L.H.S. $\exists x (P(x) \vee Q(x))$

Let domain $\{x_1, x_2, \ldots, x_n\}$

$\equiv (P(x_1) \vee Q(x_1)) \vee (P(x_2) \vee Q(x_2)) \ldots (P(x_n) \vee Q(x_n))$

$\equiv (P(x_1) \vee P(x_2) \ldots P(x_n)) \vee (Q(x_1) \vee Q(x_2) \vee \ldots Q(x_n))$

$\equiv \exists x\, P(x) \vee \exists x\, Q(x)$

Translating the sentences into logical expressions:

Example:

"Every student in this class has studied calculus."

Let domain = {student in this class}

If we introduce a variable in a sentence.

"For every student x in this class, x has studied calculus."

Let, $C(x)$: x has studied calculus.

$\forall x\, C(x)$.

If we change the domain = {all people}

Then we express our original statement as:

"For every people x, if people x is student in this class then x has studied calculus." $\rightarrow C(x)$.

$\therefore \forall x (P(x) \rightarrow C(x))$

Teacher's Signature

2. Express the statement "some student in this class has visited kathmandu" using predicate and quantifier.
Let domain = {student in this class}

Sol:

⇒ If we introduce variable, original statement can be expressed as:

- "There exist some student x in this class such that x has visited kathmandu."
- "There is student x in this class such that x has visited kathmandu."

Let, M(x) : x has visited kathmandu.
∴ ∃x M(x).

⇒ Let domain = {all student}

- "There is some student x such that x is student of this class and x has visited kathmandu.
Let; C(x) : x is student of this class.
M(x) : x has visited kathmandu.
∴ ∃x (C(x) ∧ M(x))

3. "Some student of this college passed CSIT entrance examination"
domain = {all student}

Sol:

⇒ This statement can be expressed as:
There is some student x such that x is student of this

college and x has passed CSIT entrance examination.
Let;
P(x) : x is student of this college.
Q(x): x has passed CSIT entrance examination.
∴ ∃x (P(x) ∧ Q(x))

4. "Every student in this class has studied discrete Mathematics."
domain : {set of all person}

Sol:

This statement can be expressed as;
"For every person x, if x is student in this class then x has studied discrete Mathematics."
let P(x) : x is student in this class
Q(x) : x has studied discrete Mathematics.
∴ ∀x (P(x) → Q(x))

5. S(x) : x is student
C(x) : x is clever
M(x) : x is successful.

a) Some student are clever (domain : all people)
- There is some people x such that x is student and x is clever.
∴ ∃x (S(x) ∧ C(x))

b) Some student are not successful (domain: all people)
- There is some people x such that x is student and x is not successful.
∴ ∃x (S(x) ∧ ¬M(x))

Teacher's Signature

Translate the following statements into logical expression in three different ways by varying a domain and by using predicates with one and with two variables.

1. "There is a student in your school who can not speak hindi."

Let, domain = {student in your school}

- "There is a student 'x' in your school such that x can't speak hindi."

Let, H(x) : x can't speak hindi.

$\therefore \exists x\, H(x)$

domain = {all people}

- There is a person x such that x is student in your school and x cannot speak hindi.

Let y(x): x is student in your school.

$\therefore \exists x\, (y(x) \wedge H(x))$

domain = {all people}.

if we Let S(x,y) : Person x can speak language y.
where domain y = {set of all language}.

- Then above original statement can be represented as:

"There is a person x such that x is student in your school and x can not speak hindi."

$\therefore \exists x\, (y(x) \wedge \neg S(x, \text{hindi}))$

Teacher's Signature

Translating logical expression into english sentences:

Example 1:

Let $p(x)$: "x spends more than five hour every weekday in class."

domain = {all student}

Express each of these quantification in english.

a) $\exists x\, P(x)$

- There is student x such that x spends more than five hour every weekday in class.
- There is student who spends more than five hour every weekday in class.

Example 2:

$C(x)$: x is comedian

$f(x)$: x is funny

domain = {all people}

a) $\forall x\, (C(x) \rightarrow f(x))$

- For every people x, if x is comedian then x is funny.
- Every comedian is funny.

or, All comedian is funny.

b) $\forall x\, (C(x) \wedge f(x))$

- Every person is commedian and funny.

Teacher's Signature

c) $\exists x (C(x) \rightarrow f(x))$

soln:

- There exist a person x such that if x is comedian then x is funny.
- Some comedian are funny.

Negating a quantified expression:

Let us consider,

"Every student in your class has taken a course in Calculus."

Let domain = { all student in your class }

→ $\forall x\, P(x)$ where $P(x)$: x has a course in calculus.

The negation of given statement is:

- "It is not the case that every student in your class has taken a course in calculus."
- There is student in your class who has not taken a course in calculus which is represented as;

$$\exists x \neg P(x)$$

From above:

$$\boxed{\neg \forall x P(x) \equiv \exists x \neg P(x)}$$

Consider another example:

"There is a student in this class who has taken a course in calculus."

where domain = { all student in this class }

→ $\exists x\, P(x)$

where $P(x)$: x has taken a course in calculus.

Negation of this statement:

It is not the case that there is a student in this class who has taken a course in calculus.

This is equivalent to;

"Every student in this class has not taken a course in calculus."

$\forall x \neg P(x)$

From this;

$$\boxed{\neg \exists x P(x) \equiv \forall x \neg P(x)}$$

Example 1:

Express each of these statement using quantifier, then form the negation of quantifier and express negation in simple english.

a) Some old dogs can learn new trick.

domain = {all old dog}

→ $\exists x P(x)$

where, $P(x)$: x can learn new trick

We know,

$\neg \exists x P(x) \equiv \forall x \neg P(x)$

Every old dog cannot learn new trick.

No old dog can learn new trick.

b) There is no dog that can talk.

domain = {all dog}

→ Then original statement is : $\neg \exists x P(x)$.

Negation is : $\neg\neg \exists x P(x) \equiv \exists x P(x)$

i.e. There is dog that can talk.

Teacher's Signature

Example 2:

Find the negation of statement: $\forall x (x^2 > x)$

→ Negation is:

$$\equiv \neg \forall x (x^2 > x)$$
$$\equiv \exists x \neg (x^2 > x)$$
$$\equiv \exists x (x^2 \leq x)$$

Nested quantifier:

- Two quantifier are nested if one is within the scope of other.

For example:

$$\forall x \exists y (x+y=0)$$
$$\forall x \forall y (x+y=y+x)$$

Here, $\forall x \exists y$ is same thing as $\forall x Q(x)$ where $Q(x)$ is $\exists y P(x,y)$ and $P(x,y)$ is $x+y=0$.

- The following table summarizes the meaning of different possible quantification involving two variables.

Statement	When true?	When false?
$\forall x \forall y P(x,y)$	$P(x,y)$ is true for every pair of x,y.	There is a pair x,y for which $P(x,y)$ is false.
$\forall x \exists y P(x,y)$	For every x there is y for which $P(x,y)$ is true.	There is x such that $P(x,y)$ is false for every y.

Teacher's Signature

Statement	When true?	When false?
$\exists x \forall y\ P(x,y)$	There is an x for which $P(x,y)$ is true for every y	For every x there is y for which $P(x,y)$ is false.
$\exists x \exists y\ P(x,y)$	There is a pair x,y for which $P(x,y)$ is true.	For (x,y) is false for every pair of x and y.

Note:

$\forall x \forall y\ P(x,y) \equiv \forall y \forall x\ P(x,y)$

$\exists x \exists y\ P(x,y) \equiv \exists y \exists x\ P(x,y)$

$\forall x \exists y\ P(x,y) \not\equiv \exists y \forall x\ P(x,y)$

Example:

Let $Q(x,y) = x+y=0$

Find the truth value of:

i) $\forall x \exists y\ Q(x,y)$

ii) $\exists y \forall x\ Q(x,y)$

where domain for all variable = {all real number}

Sol:

i) $\forall x \exists y\ Q(x,y)$

- It denotes the proposition:

"For every real number x there exists real number y such that $Q(x,y)$.

Teacher's Signature

→ Given a real number x, there is a real number y such that $x+y=0$ namely $y=-x$. Hence statement $\forall x \exists y Q(x,y)$ is true.

ii) $\exists y \forall x Q(x,y)$

It denotes the proposition;

"There is real number y such that for every real number x $Q(x,y)$.

⇒ Justification: No matter what value of y is choosen, there is only one value of x for which $x+y=0$ but not for every x so the statement $\exists y \forall x Q(x,y)$ is false.

Hence from above;

$$\forall x \exists y P(x,y) \not\equiv \exists y \forall x P(x,y)$$

Translating Mathematical statement into statement involving Nested quantifier:

Example 1:

"The product of two negative real number is positive."

Let domain = {set of real number}

⇒

$$\forall x \forall y \left((x<0) \wedge (y<0) \rightarrow (xy>0) \right)$$

$$\forall x \forall y \left((x<0) \wedge (y<0) \rightarrow P(x,y) \right)$$

Example 2:
"The difference of two positive integer is not necessarily positive."
where domain = {set of all integer}.

Sol:

$\equiv \neg\forall x \forall y\ ((x>0) \wedge (y>0) \rightarrow ((x-y)>0))$
$\equiv \exists x \neg\forall y\ ((x>0) \wedge (y>0) \rightarrow (x-y)>0)$
$\equiv \exists x \exists y\ \neg((x>0) \wedge (y>0) \rightarrow \neg(x-y)>0)$

Translating english sentence into logical expression:

Example 1:
"If a person is female and is parent, then this person is someone's mother."
where domain = {all people}.

Sol:

Given statement can be expressed as;
"For every person x, if person x is female and x is parent then there exists a person y such that person x is mother of y."

Let, $f(x)$: x is female.
$P(x)$: x is parent
$M(x,y)$: x is mother of y.

The original statement can be expressed using quantifier as;

$\forall x\ ((f(x) \wedge P(x)) \rightarrow \exists y\ M(x,y))$
$\forall x \exists y\ [(f(x) \wedge P(x)) \rightarrow M(x,y)]$

Teacher's Signature

Example 2:

Let L(x,y) be the statement "x and loves y."
domain for x and y = {all people in the world}.
Use the quantifier to express following statement.

a) "Everybody loves Jerry"
$\forall x\, L(x, \text{Jerry})$

b) There is somebody whom everybody loves."
$\exists y \forall x\, L(x,y)$.

c) Everybody loves everybody.
$\forall x \forall y\, L(x,y)$

d) There is someone who loves no one besides himself.
$\exists x \forall y\, (L(x,y) \rightarrow (y = x))$

Example 3:

"Every student in this class has taken at least one computer science course."

Sol^n.

$\forall x \exists y\, M(x,y)$

where, M(x,y) = x has taken y course.
x = {all student in this class}
y = {all computer science course}.

Example 4:

Let $L(x,y)$ = x loves y

a) everyone loves someone

$\forall x \exists y\, L(x,y)$

b) There is a people who is loved by everyone.

$\exists y \forall x\, L(x,y)$

Negating Nested quantifier:-

Example 1:

a) Express negation of the statement $\forall x \exists y (xy = 1)$

sol:

Negation is $\neg\forall x \exists y (xy = 1)$

$\equiv \exists x \neg\exists y (xy = 1)$

$\equiv \exists x \forall y \neg(xy = 1)$

$\equiv \exists x \forall y (xy \neq 1)$

There exists x such that for all y, $xy \neq 1$.

b) Someone has visited every country in world except Libiya.

Let domain = {all people}

sol: Some people x has visited every country y in world except Libiya.

$\exists x (V(x,y) \wedge (y \neq \text{Libiya}))$

Negation is; $\neg\exists x (V(x,y) \wedge (y \neq \text{Libiya}))$

$\equiv \forall x \neg(V(x,y) \wedge (y \neq \text{Libiya}))$
$\equiv \forall x (\neg V(x,y) \vee \neg(y \neq \text{Libiya}))$
$\equiv \forall x (\neg V(x,y) \vee (y = \text{Libiya}))$

i.e. Every person has visited Libiya or has not visited a country other than Libiya.

i.e. Every person has not visited a country other than Libiya or visited Libiya.

1.2 : Induction and Reasoning :

Argument :-

An argument in propositional logic is sequence of propositions. All proposition in the sequence except the last is called premises and the last is called conclusion. An argument is valid if the truth of all its premises implies that conclusion is true.

For example:

"If you have current password, then you can log on the network."

"you have a current password"

Therefore,

"you can log on to the network. } - conclusion

Argument form:

To analyze an argument, we replace propositions by propositional variable. This changed an argument to argument form. Thus, an argument form in propositional logic is sequence of compound propositions involving the propositional variable. An argument form is valid if the conclusion is true when all the premises are true.

In the above example;

Let, p: you have current password

q: you can log on the Network.

Then argument has argument form:

$$\begin{array}{l} p \rightarrow q \\ \underline{p} \\ \therefore \quad q \end{array}$$

We say that this form of argument is valid because whenever all its premises are true, the conclusion must also be true.

An argument form with premises $p_1, p_2, \ldots, p_n$ and conclusion q is valid when $(p_1 \wedge p_2 \wedge \ldots \wedge p_n)$ is a tautology.

Rule of inference:

To draw a conclusion from the given premises, we must be able to apply some well defined steps that helps reaching the conclusion. These steps of reaching the conclusion are provided by rules of inference.

In other word, to deduce the new statement from statement we already have, we use the rule of inference which are templates for constructing valid arguments.

Some of the rule of inference are:-

1) Modus Ponens (Law of detachment):

Whenever two propositions p and $p \rightarrow q$ are true, then we confirm that q is true. We write this rule as;

$$\begin{array}{l} p \\ p \rightarrow q \\ \hline \therefore q \end{array}$$

This rule is valid rule of inference because the implication, $[p \wedge (p \rightarrow q)] \rightarrow q$ is a tautology.

Example:

"It snows today."

"If it snows today, then we will stay at home."

Then by, Modus Ponens, it logically infers:

We will stay at home.

$$\begin{array}{l} p \\ p \rightarrow q \\ \hline \therefore q \end{array}$$

2) Hypothetical syllogism (Transitive Rule):

Whenever two propositions $p \to q$ and $q \to r$ are both true then we confirm that implication $p \to r$ is true. We write this rule as;

$$\begin{array}{l} p \to q \\ q \to r \\ \hline \therefore p \to r \end{array}$$

This rule is valid rule of inference because the implication $[(p \to q) \wedge (q \to r)] \to (p \to r)$ is a tautology.

This rule can be extended to a larger number of implication that;

$$\begin{array}{l} p \to q \\ q \to r \\ r \to s \\ \hline p \to s \end{array}$$

Example:

"if today is sunday, then today is rainy day."

"if today is rainy day, then it is wet today."

Therefore;

"If today is sunday, then it is wet today."

Rule 3: Addition

$$\frac{P}{\therefore P \vee Q}$$

This rule is valid rule of inference because the implication $[p \rightarrow (p \vee q)]$ is a tautology.

Example:

Ram is a Mathematics major, therefore Ram is either Mathematic major or computer science major.

Rule 4: Simplication

$$\frac{P \wedge Q}{\therefore P. \quad Q.}$$

Example:

Ram is a Mathematics major and computer science major therefore Ram is a Mathematics major.

Rule 5: Conjunction

$$\frac{P \quad Q}{\therefore P \wedge Q}$$

This rule of inference is valid rule of inference because $[(p) \wedge (q)] \rightarrow p \wedge q$ is a tautology.

Example:
Ram is Mathematics major, Ram is computer science major therefore Ram is Mathematics major and computer science major.

Rule 6: Modus Tollens

$$\begin{array}{l} \neg q \\ p \rightarrow q \\ \hline \therefore \neg p \end{array}$$

This is valid rule of inference because $[\neg q \wedge (p \rightarrow q)] \rightarrow \neg p$ is a tautology.

Example:
If it snows today, then university will be closed. The university is not closed therefore it does not snow today.

Rule 7: Disjunctive syllogism

$$\begin{array}{l} p \vee q \\ \neg p \\ \hline \therefore q \end{array}$$

This rule of inference is valid rule of inference because $[(p \vee q) \wedge \neg p] \rightarrow q$ is a tautology.

Example:
It is snow today or cold today. It is not snow today therefore it is cold today.

Teacher's Signature

Rule 8: Resolution

$$\frac{p \vee q \quad \neg p \vee r}{\therefore q \vee r}$$

This rule of inference is valid rule of inference because $[(p \vee q) \wedge (\neg p \vee r)] \rightarrow q \vee r$ is a tautology.

Example:

It is snowy today or cold today. It is not snowy today or it is sunny. Therefore, It is cold today or sunny today.

In summary

Rule of inference	Tautology	Name
1) $\frac{p \quad p \rightarrow q}{\therefore q}$	$[p \wedge (p \rightarrow q)] \rightarrow q$	Modus ponens
2) $\frac{\neg q \quad p \rightarrow q}{\therefore \neg p}$	$[\neg q \wedge (p \rightarrow q)] \rightarrow \neg q$	Modus tollens
3) $\frac{p \rightarrow q \quad q \rightarrow r}{\therefore p \rightarrow r}$	$[(p \rightarrow q) \wedge (q \rightarrow r)] \rightarrow (p \rightarrow r)$	Hypothetical syllogism

	Rule of inference	Tautology	Name
4)	$p \vee q$ $\neg p$ $\therefore q$	$[(p \vee q) \wedge \neg p] \rightarrow q$	Disjunctive syllogism
5)	p $\therefore p \vee q$	$p \rightarrow (p \vee q)$	Addition
6)	$p \wedge q$ $\therefore p$	$(p \wedge q) \rightarrow p$	simplification
7)	p q $\therefore p \wedge q$	$[(p) \wedge (q)] \rightarrow (p \wedge q)$	Conjunction
8)	$p \vee q$ $\neg p \vee r$ $\therefore q \vee r$	$[(p \vee q) \wedge (\neg p \vee r)] \rightarrow (q \vee r)$	Resolution

068:- State which rule of inference is basis of the following argument.

1) It is below freezing now. Therefore, it is either freezing or raining now.

Sol:-

Let p: It is below freezing now.
q: It is raining now.

Then the given argument has the form.

$$\frac{P}{\therefore P \vee q}$$

Hence, this argument uses the addition rule.

2069: **Valid argument and valid argument form:**

An argument in propositional logic is valid if the truth of all its premises implies that the conclusion is true. An argument form is valid if no matter which particular proposition are substituted for propositional variable in its premises, the conclusion is true if the premises are all true. i.e. the argument form with premises $P_1, P_2, \ldots, P_n$ and conclusion q is valid when $(P_1 \wedge P_2 \wedge \ldots \wedge P_n) \rightarrow q$ is a tautology. The key to show an argument in propositional logic is valid is to show that its argument form is valid.

Some times, valid argument can lead to incorrect conclusion if one or more of the false premises are used in the argument.

Proving the validity of argument:

Example:- Determine whether the following argument is valid or not.

If it is rainy, then pool will be closed. It is rainy therefore the pool is closed.

Sol:

Let p: It is rainy.
q: The pool will be closed.
Hence symbolic form of given argument is:

$$\begin{array}{l} p \to q \\ p \\ \hline \therefore q \end{array}$$

To show that given argument is valid we need to show:
$[(p \to q) \wedge q] \to q$ is tautology.

p	q	$(p \to q)$	$(p \to q) \wedge q$	$[(p \to q) \wedge q] \to q$
T	T	T	T	T
T	F	T	F	T
F	T	F	F	T
F	F	F	F	T

Hence, the given argument is valid.

Example: 2

Show that the following argument is valid using rule of inference.

If today is tuesday, I have test in Mathematics or economics. If my economics professor is sick, I will not have a test in economics. Today is tuesday and my economics professor is sick. Therefore I have a test in Mathematics.

Sol:

⇒ Let,

T: Today is tuesday
M: I have a test in Mathematics
E: I have a test in Economics
S: My economics professor is sick.

Now given argument have following form.

1. $T \rightarrow (M \vee E)$
2. $S \rightarrow \neg E$
3. $T \wedge S$

$\therefore M$

By using simplification on (3)

4) T
5) S

By using Modus Ponens on (1) and (4)

6) $M \vee E$

By using Modus Ponens on (2) and (5)

7) $\neg E$

By using disjunctive syllogism on (6) and (7)

8) M

Hence, given argument is valid with conclusion:
I have a test in Mathematics.

Example 3:

Use the rules of inference to show that the hypothesis: "Ram works hard." "If Ram works hard, then he is dull boy" and "if Ram is a dull boy, then he will not get the job" imply the conclusion "Ram will not get the job."

Sol:

Let,

W: "Ram works hard"
d: "Ram is dull boy"
J: "Ram will get the job."

The symbolic form of above hypothesis is:

1) W
2) $W \rightarrow d$
3) $d \rightarrow \neg J$

$\therefore \neg J$

Steps:	reason:
i) W	hypothesis
ii) $W \rightarrow d$	hypothesis
iii) d	by Modus Ponens on (i) and (ii)
iv) $d \rightarrow \neg J$	hypothesis
v) $\neg J$	Modus Ponens on (iii) and (iv)

Hence, given argument conclude:
Ram will not get the job.

Example 4:
For each of given set of premises, what relevant conclusion can be drawn state the rule of inference to obtain each conclusion.

a) "If I play hockey, then I will score the next day." I will use the whirlpool if I score." I did not use whirlpool.

Sol:

Let,
p: I play hockey
q: I will score the next day
r: I use the whirlpool.

Then above premises are:
i) $p \rightarrow q$
ii) $q \rightarrow r$
iii) $\neg r$

From (i) and (ii);
$p \rightarrow r$ ——(iv)

Conclusion: If I play hockey then I use the whirlpool.

From (iv) and (ii)
I did not play hockey.

Fallacies :-

The fallacies are argument that are convincing but not correct. So, fallacies produce faulty inferences. So fallacies are contengency rather than tautology. There are different types of fallacies that may encounter during the construction of arguments.

1. Fallacy of affirming the conclusion

$$\frac{p \to q \quad q}{\therefore p}$$

Many incorrect argument treat the argument with premises '$p \to q$' and 'q' and conclusion p as a valid argument form which is not. These type of incorrect reasoning is called fallacy of affirming conclusion. This kind of fallacy has the form:

$$\frac{p \to q \quad q}{\therefore p}$$

i.e. $[(p \to q) \wedge q] \to p$ is not a tautology hence it is fallacy.

Example:

If the price of the gold is rising, then inflation is surely coming. Inflation is surely coming, therefore the price of gold is rising.

2. Fallacy of denying hypothesis:

Many incorrect argument treat the argument with premises '$p \to q$' and '$\neg p$' and the conclusion '$\neg q$' as a valid argument form which is not. This type of incorrect reasoning is called fallacy of denying hypothesis. This kinds of fallacy has the form:

$$\begin{array}{l} p \to q \\ \neg p \\ \hline \therefore \neg q \end{array}$$

ie. $[(p \to q) \wedge \neg p] \to \neg q$ which is not a tautology because it is false when p is false, q is true. Hence it is fallacy.

Example:

If today is sunday, then it rains today. Today is not sunday therefore it doesn't rain today.

Rule of inference for quantified statement:

Some rule of inference for quantified statement are given below:

1. Universal instantiation:

It is the rule of inference used to conclude that P(c) is true where c is the particular member of domain, given that premise $\forall x\, P(x)$ is supposed to be true.
This rule can be written as;

$$\frac{\forall x\, P(x)}{\therefore P(c) \text{ for any arbitrary } c \text{ in domain.}}$$

2. Universal generalization:

It is the rule of inference used to conclude that $\forall x P(x)$ is true given that the premise that P(c) is true for all element c in domain.

This rule can be written as;

P(c) for all c in domain.

3. Existential instantiation:

It is the rule of inference that allows us to conclude that there is an element c in the domain for which P(c) is true if we know that $\exists x\, P(x)$ is true.

This rule can be written as:

$$\frac{\exists x\, P(x)}{\therefore P(c) \text{ for some 'c' in the domain}}$$

Teacher's Signature

4. Existential generalization:

This is the rule of inference that is used to conclude that $\exists x\, P(x)$ is true when a particular element c in the domain makes $P(c)$ true.

This rule can be written as;

$P(c)$ for some c in the domain.

Example: Symbolize the following argument and check for its validity.

all lions are dangerous animal.

There are lions

Therefore there are dangerous animal.

Let $L(x)$: x is lion

$D(x)$: x is dangerous.

i) $\forall x\, (L(x) \rightarrow D(x))$

ii) $\exists x\, L(x)$

$\therefore \exists x\, D(x)$

Step	reason
1. $\forall x (L(x) \rightarrow D(x))$	premises
2. $L(c) \rightarrow D(c)$	universal instantiation
3. $\exists x\, L(x)$	premise
4. $L(c)$	existential instantiation
5. $D(c)$	Modus ponens on (2) & (4)
6. $\exists x\, D(x)$	Existential generalization.

Hence, it is valid argument.

Teacher's Signature

Methods of Proof:

Some terminology:-

1. **Proof:-**

A proof is a valid argument that establishes the truth of mathematical statement. A proof can use hypothesis of theorem, previously proven theorems and rule of inferences to establish the truth of mathematical statement. The final step in the proof establish the truth of statement be improved.

2. **Theorem:-**

Formally, theorem is a statement that can be shown to be true. Theorem can also be referred to as a fact or result and theorem becomes true with proof.

Methods of Proving implication ($p \rightarrow q$)

To prove the theorem of the form '$p \rightarrow q$', we need to show that conditional statement is true. Different method of proving the implication are:

1) **Direct proof:**

The implication $p \rightarrow q$ can be proved by showing that if p is true, then q must also be true. The construction of direct proof of $p \rightarrow q$ begins by assuming p is true and using the already available information. The conclusion

q is shown to be true i.e. steps in direct proof are :-

i) Assume p is true
ii) Using definition, previously proven theorem together with rules of inference to show q must also be true.

Hence, direct proof lead from the hypothesis of theorem to the conclusion. They begins with the premise, continue with the sequence of deduction and end with the conclusion.

Example:

if n is an odd integer, then n^2 is odd.

$p \rightarrow q$

Proof:

Assume n is an odd.

Then,

$n = 2k+1$ (by the definition of odd integer)

Squaring on both sides;

$n^2 = 4k^2+4k+1$

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

Since, $(2k^2+2k)$ is integer.

$\therefore n^2$ is odd

Hence, we proved that if n is an odd integer then n^2 is an odd.

Note:

The integer n is even if there exists an integer k such that $n = 2k$. and n is odd if there exists an integer k such that $n = 2k+1$.

2) Indirect Proof (Proof by contraposition):

The proof method on which we do not start with hypothesis and end with the conclusion are called indirect proofs.

The proof by contraposition makes the use of fact that conditional statement '$p \to q$' is equivalent to its contrapositive i.e. $\neg q \to \neg p$ so in indirect proof the implication '$p \to q$' can be proved by showing that its contrapositive '$\neg q \to \neg p$' is true. These contrapositive is shown to be true by direct proof from the assumption that $\neg q$ is true and we show $\neg p$ is true.

Thus indirect proof proceeds as follows:-

i) Assume q is false
ii) Prove on the basis of assumption and other available information p is false.

Example:

Prove that if n is an integer and $3n+2$ is odd, then n is odd.

Proof:

First step in indirect proof is to assume that conclusion

of conditional statement is false.
i.e. assume n is even
$n = 2k$ for some integer k.
Substituting value of n on $3n+2$.

$$3n+2 = 3\times 2k+2$$
$$= 6k+2$$
$$= 2(3k+1)$$

Since $3k+1$ is integer, let it be r.
$3n+2 = 2r$ where r is integer.

This concludes $3n+2$ is even hence proved.

3) Trivial Proof:

In the implication '$p \to q$', if we can show that the consequence q is true then regardless of truth value of p, the implication '$p \to q$' is true. Such type of proof technique is called trivial proof.
Thus, construction of the trivial proof of '$p \to q$' requires showing that truth value of q is true.

Example:

1. For real number x if $x \geq 0$ then $x^2+2x+1 \geq 0$.

Trivial proof:

We need to show that q is true.

x^2+2x+1

$\Rightarrow (x+1)^2 \geq 0$ (by definition of real number)

Hence proved..

2. If x is integer, then 3 is odd.

Proof:

$3 = 2\times 1+1$ (by the definition of odd)

Hence proved...

4) Vacous proof:

In the implication 'p→q', if we can show that the hypothesis p is false then regardless of truth value of q, the implication 'p→q' is true. Such kind of proof of an implication p→q is called vacous proof.

Thus, the construction of vacous proof of p→q requires showing that p is false.

Example:

For every real number x if $x^2 < -1$ then $x^2+1<0$.

Proof:

For every real number x, $x^2 \geq 0$ (by the def^n of Real no.)

$\Rightarrow x^2 \not< -1$

∴ For each real number x if $x^2 < -1$ then $x^2+1<0$.

proved.

Teacher's Signature

5) Proof by contradiction:

We can prove that p is true if we can show that $\neg p \rightarrow (r \wedge \neg r)$ is true for some proposition 'r'.

Proof of this type is called proof by contradiction.

For proving the implication 'p→q' the step of proof are as follows:

i) Assume $(p \wedge \neg q)$ is true

ii) Try to show that above assumption $\neg(p \wedge \neg q)$ is false.

iii) When this assumption is found to be false then implication p→q is true.

iv) Since p→q is equivalent to $(\neg p \vee q)$ and $\neg(\neg p \vee q)$ is $(p \wedge \neg q)$. So, if our assumption is false then its negation i.e. original statement is true.

Example:

if a^2 is even number then a is even number.

Proof:

Let p be "a^2 is even" and q be "a is even".

So, assume a^2 is even and a is odd.

Since a is odd number

$\Rightarrow a = 2k+1$ for some integer k.

$\Rightarrow a^2 = 4k^2 + 4k + 1$

$\Rightarrow a^2 = 2(2k^2 + 2k) + 1$

Let $2k^2 + 2k = m$ where m is integer.

$\Rightarrow a^2 = 2m + 1$

$\therefore a^2$ is odd.

Hence, it contradicts our assumption hence original statement is true.

Proved...

Teacher's Signature

6) Proof by cases:

The implication of the form $(P_1 \vee P_2 \vee P_3 \vee \ldots \vee P_n) \to q$ can be proved by proving n conditional statement $P_i \to q$ where $i = 0, 1, 2, \ldots, n$ individually, such type of proof is called proof by cases.

A proof by cases covers all possible cases that arises in the theorem.

Example:

Prove that if n is an integer, then $n^2 \geq n$.

Proof:

We can prove that $n^2 \geq n$ for every integer by considering three cases.

Case 1: When $n = 0$

because $0^2 \geq 0$ is true, hence case 1 is true.

Case 2: When $n \geq 1$

$\Rightarrow n \cdot n \geq n$ (multiplying both sides by n)

$\Rightarrow n^2 \geq n$ is true hence case 2 is true.

Case 3: When $n \leq -1$

We already know,

$n^2 \geq 0$ for any integer

$\Rightarrow n^2 \geq n$, is true hence Case 3 is true.

Finally the inequality $n^2 \geq n$ holds in all three cases hence proved...

7) **Proof of equivalence:**

To prove a theorem that is a biconditional statement i.e. statement of the form $p \leftrightarrow q$, we show that $(p \rightarrow q)$ and $(q \rightarrow p)$ are true.

Example:

Prove that if n is a positive integer, then n is even if and only if $7n+4$ is even.

$\Rightarrow$ (n is even if and only if $7n+4$ is even where n is positive integer.)

Proof:

This theorem has the form p if and only if q. (i.e. $p \leftrightarrow q$)

Proving $p \rightarrow q$: if n is even then $7n+4$ is even

Assume n is even

$\Rightarrow n = 2k$ for some integer k

So, $7n+4 = 7 \times 2k + 4$

$\Rightarrow 7n+4 = 2(7k+2)$

Let $7k+2 = m$ (an integer)

$\Rightarrow 7n+4 = 2m$ for integer m.

Hence by direct proof if n is even then $7n+4$ is even.

Proving $q \rightarrow p$: if $7n+4$ is even then n is even.

Assume n is odd

$\Rightarrow n = 2k+1$ for some integer k

Then, $7n+4 = 7(2k+1)+4$

$\Rightarrow 7n+4 = 14k+7+4$

Teacher's Signature

$\Rightarrow 7n+4 = 2(7k+5)+1$

let $7k+5 = m$, an integer

$7n+4 = 2m+1$ for integer m

Hence by indirect proof it is proved that if $7n+4$ is even then n is even.

So, original statement is true.

Proved...

8) Existence proof:

A proof of a proposition of the form $\exists x P(x)$ is called an existence proof. There are several ways to prove a theorem of this type. The first method of existence proof of $\exists x P(x)$ can be given by finding an element A such that $P(A)$ is true. Such existence proof is called constructive existence proof. Other method do not find an element A such that $P(A)$ is true but rather proof $\exists x P(x)$ in some other ways, such proof is called non-constructive existence proof.

Example:

Prove that there is natural number that is even and prime.

Proof:

lets choose $x = 2$ (where x have the domain, all natural number)

$\Rightarrow$ 2 is even and prime

Hence proved...

Teacher's Signature

Mathematical induction:

Mathematical induction is extremely important proof technique that can be used to prove the result about large variety of discrete objects. For example: It if used to prove the result about complexity of algorithm, theorem about graph and trees as well as wide range of identities and inequalities.

Principle of Mathematical induction (Weak induction):

To prove that P(n) is true for all positive integer n where P(n) is a proposition function, we need to follow the following steps:

1. **Basis step: (Base of induction)**
   In this step, we need to verify that P(n) is true for $n=1$ i.e. P(1) is true.

2. **Inductive hypothesis:**
   In this step, we assume that P(k) is true for $n=k$ i.e. P(k) is true for an arbitrary k.

3. **Inductive step:**
   In this step, we prove that P(n) is true for $n=k+1$ i.e. P(k+1) is true on the basis of inductive hypothesis.
   i.e. Show that, $P(k) \rightarrow P(k+1)$ is true.

This proof technique can be expressed as the rule of inference as:

$$[P(1) \wedge (P(k) \rightarrow P(k+1))] \rightarrow \forall n\, P(n)$$

Example:

Show that

$1+2+3+\ldots+n = \frac{n(n+1)}{2}$ for all positive integer n.

Proof:

Let P(n) be the proposition "the sum of first n positive integer is $\frac{n(n+1)}{2}$"

1. Basis step:

P(1) is true because $1 = \frac{1(1+1)}{2}$

2. Inductive hypothesis:

We assume that P(k) is true for an arbitrary k.

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

3. Inductive step:

Under the inductive hypothesis, It must be shown that P(k+1) is true,

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

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

Teacher's Signature

For this;

$$1+2+3+\ldots+k+k+1 = \frac{k(k+1)+(k+1)}{2}$$ { by adding k+1 on both sides of inductive hypothesis.

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

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

This completes the inductive step.

Hence, the sum of first n positive integer is $\frac{n(n+1)}{2}$

i.e. $\forall n\, P(n)$ is true where 'n' is positive integer. Proved

Strong induction (Second Principle of Mathematical induction) or Complete induction:

The strong induction uses the same concept as in the mathematical induction however the inductive step in these two proof methods are different. In mathematical induction, the inductive step shows that if inductive hypothesis P(k) is true then P(k+1) is also true but in a proof by strong induction the inductive step shows that if P(j) is true for all positive integer not exceeding k, then P(k+1) is true.
i.e. In inductive hypothesis, we assume that P(j) is true for $j = 1, 2, \ldots, k$.

Formal definition of strong induction:

To prove that P(n) is true for all positive integer n, where P(n) is the propositional function, we need to follow the following step:-

1) **Basis step:**

In this step, we verify that P(1) is true.

2) **Inductive step:**

In this step, we show that the conditional statement $[P(1) \wedge P(2) \wedge P(3) \ldots P(k)] \rightarrow P(k+1)$ is true for all positive integer k.

To show the inductive step, our inductive hypothesis is the assumption that P(j) is true for $j=1,2,\ldots,k$.

Example:

Prove that n can be factorized as a prime for all $n>1$.

Proof:

Let P(n) be the proposition "n can be written as factor of primes."

Basis step:

P(2) is true, because 2 can be written as factor of prime i.e. itself.

Inductive step:

Assume P(j) is true for all positive integer $2 \leq j \leq k$. Then we have to show that P(k+1) is true.

Now, there are two cases;

When (k+1) is prime

if (k+1) is prime then P(k+1) is true because any prime can be written as factor of prime.

When (k+1) is composite

if (k+1) is composite then it can be written as:

$k+1 = a*b$ where, $2 \leq a \leq b \leq k$

By inductive hypothesis, both 'a' and 'b' are written as prime factor. Hence (k+1) is written as factor of prime.

proved.

Recursive definition:

Sometimes it is difficult to define an object explicitly. However, it may be easy to define this object in terms of itself, this process is called recursion. Recursion can be used to generate sequence, recursive function and recursive algorithm.

Recursively defined function:

We use two steps to define the function with the set of non-negative integer as its domain.

Teacher's Signature

Basis step:

Specify the value of function at 0 or at base of the recursion.

Recursive step:

Give the rule for finding its value at an integer from its value at smaller integer. Such a definition is called recursive or inductive definition.

Example:

1. Let the given sequence is 2, 5, 8, 11, ...

Recursive definition:

Basis step $s(0) = 2$

Recursive step $s(n+1) = s(n) + 3$

2. Let the sequence 1, 3, 9, 27....

Recursive definition:

Basis step $s(0) = 1$

Recursive step $s(n+1) = 3 \cdot s(n)$

Recursive algorithm:

An algorithm is called recursive if it solve the problem by reducing it into instances of same problem with smaller input.

Example: An algorithm to find the factorial of given number.

```
fact(n)
{
  if n==0
  then return 1
else
    return n*fact(n-1)
}
```

Puspanjali
Date / /
Page

Unit:2

Finite State Automata

Automata theory:

Automata theory is the study of abstract computing devices or machine. Automata are abstract mathematical model of machine that perform computation on an input by moving thraugh series of state or configuration. If the computation of automaton reaches an accepting configuration then it accept that input.

Basic concept of Automata theory:

- **Alphabets:**

Alphabet is a finite non-empty set of symbol. Conventionally, we use the symbol Σ for an alphabets.

Example:

$\Sigma = \{0,1\}$, the binary alphabet
$\Sigma = \{0,1,2,3,\ldots,9\}$ decimal alphabet
$\Sigma = \{a,b,c,\ldots,z\}$ set of all lowercase letter.

String:

String is the finite sequence of symbol taken from some alphabet Σ.

For example: Let $\Sigma = \{a,b\}$

then, abab is a string over $\Sigma = \{a,b\}$
01100 is a string over $\Sigma = \{0,1\}$.

Teacher's Signature

String is generally denoted by lowercase letter w, x, y, z.

Empty string:

It is the string with zero occurrence of symbol. It is denoted by $\epsilon, \lambda, \gamma$. This string may be choosen from any alphabet.

Length of string:

The length of string w is denoted by $|w|$ is the number of positions for symbol in w.

For example:

Let $w = 0110$

$|w| = 5$

$|\epsilon| = 0$ as ϵ is empty string.

Power of alphabets:

The set of all string of a certain length k from an alphabet is the k^{th} power of that alphabet.

i.e.

$$\Sigma^k = \{ w \mid |w| = k \}$$ (such that)

Example:

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

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

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

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

$\Sigma^3 = \{000, 111, \ldots\}$

Teacher's Signature

Kleen closure of alphabet:

The set of all string over an alphabet Σ is called kleen closure of Σ and is denoted by Σ^*. Thus, kleen closure is set of all string over alphabet Σ with length 0 or more i.e.

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

For example:-

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

$$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 111, \ldots\}$$

Question:

Determine the kleen closure of $A = \{0\}$, $B = \{0,1\}$, $C = \{11\}$

⇒

$A^* = \{\epsilon, 0, 00, 000, \ldots\}$

i.e. set of all string over $\{0\}$.

$B^* = \{\epsilon, 0, 1, 10, 01, 11, \ldots\}$

i.e. set of all string over $\{0,1\}$.

$C^* = \{\epsilon, 11, 1111, \ldots\}$

i.e. set of all string over 11.

Positive closure:

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

i.e. $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots\ldots$

For example:

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

$\Sigma^+ = \{a, b, ab, ba, bb, aa \ldots\}$

Hence, $\Sigma^+ = \Sigma^* - \{\epsilon\}$

Language:-

A language L is a set of string all of which are choosen from some Σ^* where Σ is a particular alphabet A language is subset of kleen closure of alphabet Σ i.e. $L \subseteq \Sigma^*$.

A language over Σ need not include string with all the symbol of Σ, so once we have established that L is language over Σ, we also know it is a language over any alphabet which is superset of Σ.

Example:-

i) set of all string over $\Sigma = \{0, 1\}$ with equal no zero's and one's.

$L = \{\epsilon, 01, 0011, 1010, \ldots\}$

ii) Σ^* is a language over any alphabet E that contains all string over E.

iii) ϕ is an empty language which is a language over any alphabet.

iv) $\{\epsilon\}$ is a language consisting only the empty string.

$L = \{\epsilon\}$

Teacher's Signature

v) Set of all string representing binary number whose decimal value is prime.

$L = \{10, 11, 101 \_ \_ \_ \}$

Concatenation of string:

Let x and y be a string. then xy denotes the concatenation of x and y which is the string formed by making copy of x and following it by a copy of y.

For example:-

Let $x = 000$

$y = 111$

$xy = 000111$

$yx = 111000$

Note:

$\epsilon w = w\epsilon = w$

Suppose that A and B are subset of Σ^*, where Σ is an alphabet. The concatenation of A and B is denoted by AB. is the set of all string of the form xy where x is a string in A and y is a string in B.

Example:

let, $A = \{0, 00\}$

$B = \{10, 11\}$

$AB = \{010, 011, 0010, 0011\}$

$BA = \{100, 1000, 110, 1100\}$

From the definition of concatenation, we can define A^n for $n = 0, 1, 2, \ldots$ as

$$A^0 = \{\epsilon\}$$

$$A^{n+1} = A^n \cdot A$$ where A is the set of string taken from Σ^*.

For example:-

Let $A = \{1, 00\}$

$A^0 = \{\epsilon\}$

$A^1 = \{1, 00\}$

$A^2 = \{11, 001, 100, 0000\}$

Suffix of string:

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

For example:-

baab

Here aab is the suffix of given string by removing one leading symbol.

Prefix of string:

A string s is a prefix of string w if it is obtained by removing 0 or more trailing symbol of w.

Example:-

$W = baab$

Hence, baa is the prefix of W by removing one trailing symbol.

Substring:

A string S is called substring of a string W if it is obtained by removing 0 or more leading and trailing symbols in W. If W is a string then substring from i to j is the string obtained from W beginning at i^{th} position and ending at j^{th} position both inclusive.

Example:-

$W = abaab \qquad \Sigma = \{a, b\}$

The substring $\{w, 2, 4\} = baa$

Finite Automata:

A finite automaton is a mathematical 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 automaton cannot be in more than one state at one time or non-deterministic meaning that it may be in several states at once. Depending upon the control, finite automata are of two types i.e. DFA (Deterministic Finite Automata) and NFA (Non-deterministic Finite Automata). The DFA can't be in more than one state at any time but the NFA can be in more than one state at a time.

The finite automata may generate output or it may not be.

Representation of Finite State Machine or Finite Automata:

The finite state machine can be represented with:

i) State Transition Table (State Table):

A state transition table is a table showing what state finite state machine move to, based on current state and inputs. The row in a table indicates current states and column indicates input symbol. Each cell in the table indicates next state if the event is happened.

ii) State Transition Diagram (State Diagram):

Another way to represent finite state is to use a state diagram which is a directed graph with labelled edges. In this diagram each state is represented by a circle. Arrow labelled input and output or simply input are shown for each transition between the state represented by a circle. For example:-

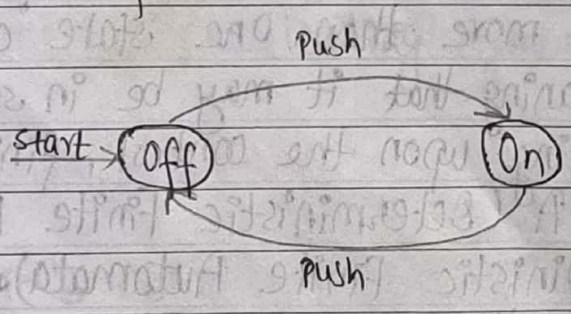


Fig: Transition diagram

S/Σ	Push
Off	On
On	Off

Fig: State Table

Finite State Machine in Sequential Circuit:-

Finite State Machine has proven to be very efficient means for modeling sequential circuit. For example:- We can represent the binary adder using finite state machine as below:-

Consider a binary adder which adds two binary strings, the addition is performed bit by bit starting from the least significant bit and carry must be taken into account during the addition.

A finite state machine to represent a binary adder can be constructed using two states. The start state S_0 is used to remember that previous carry is 0 and other state S_1 is used to remember that previous carry is 1. The input to the machine are pair of bits, there are four possible inputs i.e. 00, 01, 10, 11. The state diagram for this machine is represented as:

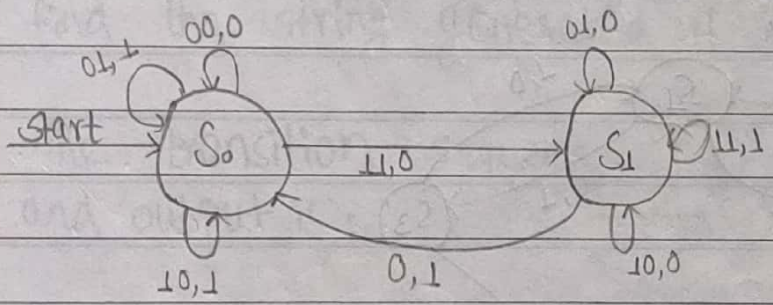


Fig: Transition diagram of binary adder

Finite State Machine With Output:

A finite state machine with output is defined by six tuples as:

$$M = (Q, \Sigma, O, \delta, f, q_0)$$

where,

M = Name of finite state machine

Q = finite set of states

Σ = finite set of input alphabet.

O = finite set of output alphabet

δ = A transition function that assign to each state and input pair a New state

i.e. $Q \times \Sigma \rightarrow Q$

f = An output function that assign to each state and input pair an output

i.e. $Q \times \Sigma \rightarrow O$

q_0 = A start state i.e. $q_0 \in Q$

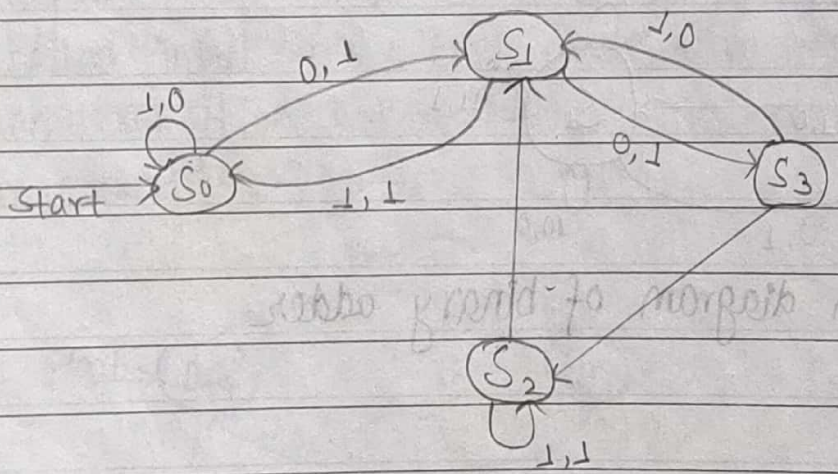


Fig: State transition diagram of finite state Machine with output

Teacher's Signature

State	δ input		f input	
	0	1	0	1
S_0	S_1	S_0	1	0
S_1	S_3	S_0	1	1
S_2	S_1	S_2	0	1
S_3	S_2	S_1	0	0

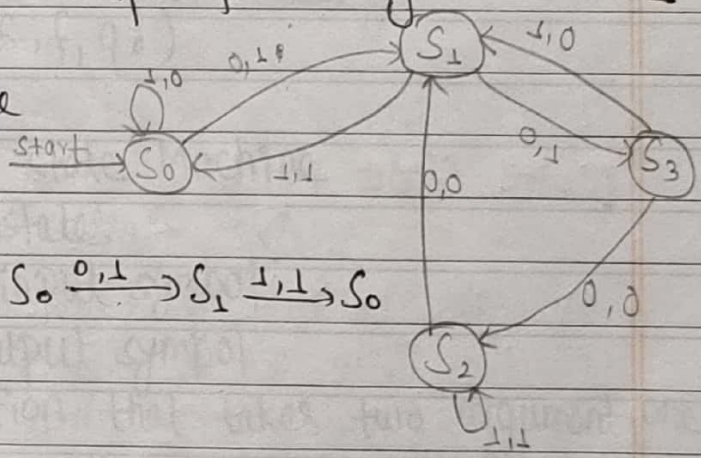
Fig: State transition table for finite state machine with output.

Question:

From the above finite state machine with output find the string generated if input string is 101011

Sol:

The transition sequence and output is:



$S_0 \xrightarrow{1,0} S_0 \xrightarrow{0,1} S_1 \xrightarrow{1,1} S_0 \xrightarrow{0,1} S_1 \xrightarrow{1,1} S_0 \xrightarrow{1,0} S_0$

Total output: 011110.

Q. Input: 101101

Sol:

The transition sequence and output is:

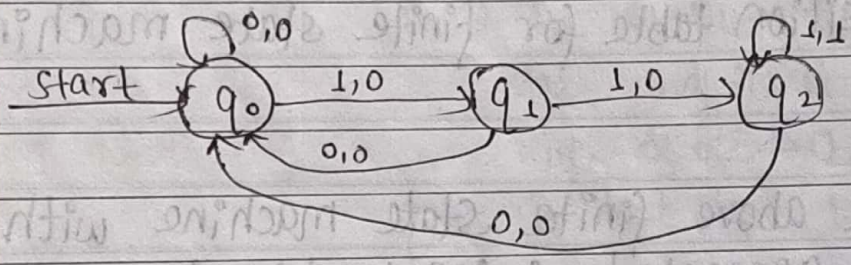
$S_0 \xrightarrow{1,0} S_0 \xrightarrow{0,1} S_1 \xrightarrow{1,1} S_0 \xrightarrow{1,0} S_0 \xrightarrow{0,1} S_1 \xrightarrow{1,1} S_0$.

Hence, output of given input string is 011011.

Example:

Construct a finite state machine with output 1 if and only if input string read so far end with 11, given that $\Sigma = \{0,1\}$ $O = \{0,1\}$.

Soln:



Types of finite state Machine with output:

There are two types of finite state machine with output;

i) Mealy machine
ii) Moore machine

i) Mealy Machine:-

Mealy machine is the finite state machine whose output values are determined by both its current state and current input. All the example discussed above are the Mealy machine.

ii) Moore Machine:-

Moore machine is the finite state machine in which output values are determined by its current state i.e. in these machine, output is associated with each state. Formally, more machine is defined as;

$$M = (Q, \Sigma, O, \delta, f, q_0)$$

where,

M = Name of finite state Machine
Q = finite set of states
Σ = finite set of input symbol
O = finite set of output symbol
δ = transition function that takes two argument; one is finite state and another is input symbol and output : a single state i.e. $Q \times \Sigma \to Q$
f = An outputfunction which maps state to output.
q_0 = start state i.e. $q_0 \in Q$.

Teacher's Signature

Example:

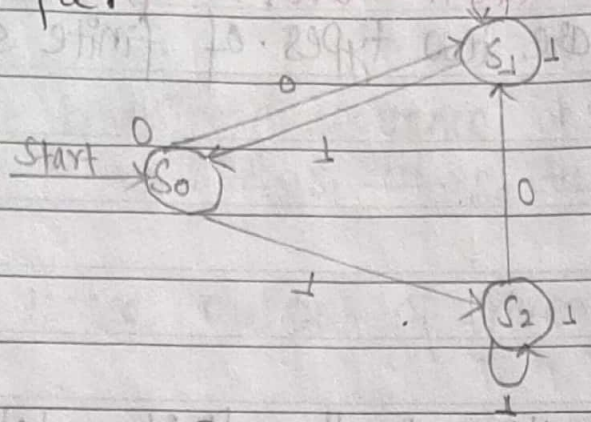


Fig: Transition diagram.

Hence,

$Q = \{S_0, S_1, S_2\}$

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

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

$q_0 = S_0$

State	δ		f
	0	1	
S_0	S_1	S_2	0
S_1	S_1	S_0	1
S_2	S_1	S_2	1

Fig: Transition Table.

Finite State Machine with no output:

The finite state machine with no output are also known as finite state automata. It consists of the finite set of states, finite input alphabet, a transition function that assigns to each state and input pair a new state. It also contains initial or start state and subset of final or accepting state. These machine differs from the finite state machine with output in that

they donot produce output but they do have a set of final states. They recognize the string if and only if it takes the start state to final state.

The finite automata with no output can be either deterministic or non-deterministic in nature.

Deterministic Finite Automata (DFA):-

A deterministic finite automaton is defined by 5-tuple as:

$$M = (Q, \Sigma, \delta, q_0, F)$$

where,

M = Name of DFA.
Q = finite set of states.
Σ = finite set of input symbols.
δ = transition function that takes an argument a state and an input symbol and returns the state
i.e. $Q \times \Sigma \rightarrow Q$.
q_0 = start state $q_0 \in Q$.
F = A set of final or accepting state $F \subseteq Q$.

General Notation of DFA:

There are two preferred notations for describing the DFAs:

i) Transition table
ii) Transition diagram

Teacher's Signature

1) Transition diagram:

A transition diagram for DFA $M = (Q, \Sigma, \delta, q_0, F)$ is a graphical representation where;

a) For each state q in Q, there is a node represented by circle.

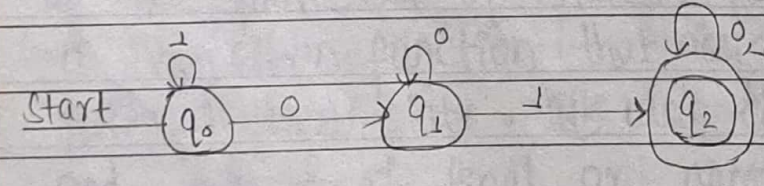
b) For each state q in Q and each input symbol a in Σ, if $\delta(q, a) = p$, then the transition diagram has an arc from node q to node p labelled with a. If more than one input symbol causes the transition from state q to p then arc from q to p is labelled by the list of these symbols.

c) There is an arrow into a start state q_0 labelled start. This arrow doesnot originate from any node

d) The final or accepting state is marked by double circle.

For example:

DFA accepting all string over {0,1} having the substring 01 is given in following transition diagram.



Here,
$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
$q = q_0$
$F = q_2$

Fig:- Transition diagram of given DFA.

Teacher's Signature

ii) Transition table:

Transition table is a conventional, tabular representation of the transition function δ that takes two arguments from $Q \times \Sigma$ and returns a value which is one of the state of the automaton. The row of the table correspond to the state while column correspond to the input symbol. The entry for the row corresponding to the state q and column corresponding to input a is the state $\delta(q, a)$.

The starting state in the table is represented by '$\rightarrow$' followed by the state and final state is represented by '$*$' followed by state.

For example:

The transition table for above DFA is;

δ / Σ	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
$*q_2$	q_2	q_2

Fig: Transition table for DFA

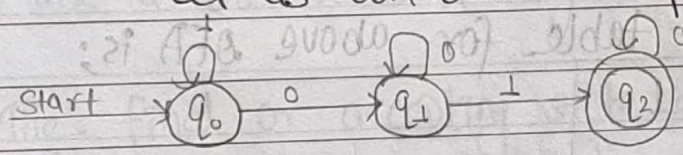
How DFA process the string:-

Suppose, $a_1, a_2, \ldots, a_n$ is a sequence of input symbols. We start the DFA in its start state q_0, we consult the transition function δ say $\delta(q_0, a_1) = q_1$ to find the state that DFA enters after processing the input symbol a_1. We process the next input symbol a_2 by evaluating $\delta(q_1, a_2) = q_2$. Let us suppose this state is q_2. We continue in this manner, finding the states $q_3, q_4, \ldots q_n$ such that $\delta(q_i, a_i) = q_i$ for each i. If q_n is the member of final state then input $a_1, 2, \ldots a_n$ is accepted, if not then it is rejected.

Hence, language of DFA is the set of all string that the DFA accept.

Example:

Let us consider the following DFA.



*. Process the string. 1101

$$q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_2$$

Hence, we reach from start state to final state after reading the input string. Hence, string is accepted.

Puspanjali
Date / /
Page

* Process the string 1100.

$q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_1$

Hence q_1 is not final state hence given DFA doesnot accept the input string.

Extended transition function:

The extended transition function of DFA is denoted by $\hat{\delta}$ is a transition function that takes two argument as a input a q state Q and string w and returns a state $P \in Q$. This state P is a state that automaton reaches when starting in state and processing the sequence of input w. We can define $\hat{\delta}$ by induction on the length of input string as

Basis step:

$\hat{\delta}(q, \epsilon) = q$ i.e. if we are in state q and read no inputs then we are still in q.

Induction step:

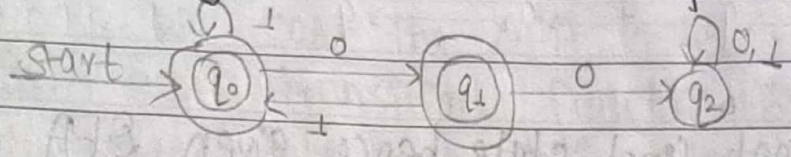
Suppose w is a string from Σ^* of the form $w = xa$, where x is a substring of w without the last symbol and 'a' is the last symbol of w, then

$$\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$$

Teacher's Signature

Example:

From given DFA below, compute $\hat{\delta}(q_0, 1001)$



Sol:

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

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

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

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

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

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

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

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

$= \delta(q_2, 1)$

$= q_2$

Hence, string is rejected.

String accepted by DFA:-

A string x is accepted by DFA $M = \{Q, \Sigma, \delta, q_0, f\}$ if $\hat{\delta}(q_0, x) = P \in F$.

Language of DFA:-

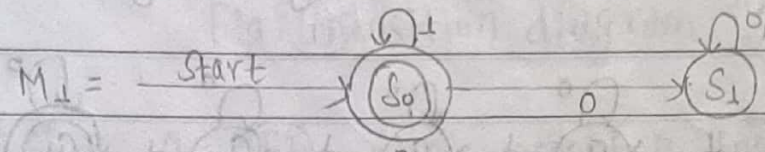
The language of DFA $M = \{Q, \Sigma, \delta, q_0, f\}$ is denoted by $L(M)$ is a set of string over Σ^* that are accepted by DFA

i.e. $L(M) = \{ w \mid \hat{\delta}(q_0, w) = P \in F \}$

In short, language of DFA is a set of all string w that take a DFA from start state to one of the final state. The language of DFA is called regular language.

Example:-

1) Find the $L(M_1)$:

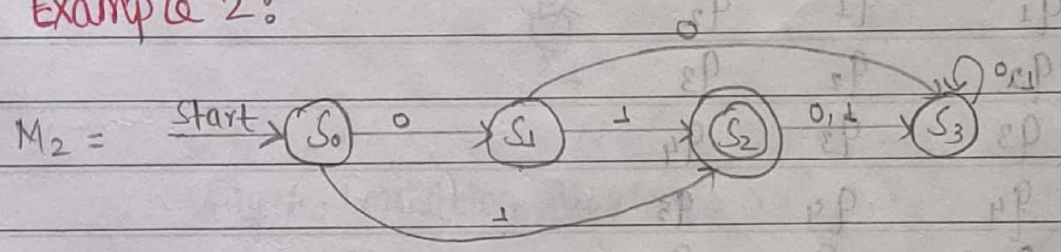


$\{\epsilon, 1, 11, 111, \ldots\}$

$L(M_1)$ = { string w such that each string w contain zero or more 1's. }

$= \{ w \mid w \text{ contains zero or more one's.} \quad w = \epsilon \cup \{1^n\} \quad n = 1,2,3. \}$

Example 2:



$L(M_2) = \{01, 1\}$

Examples of DFA:

Example 1:

Construct a DFA which accept the set of string containing exactly four 1's in every string over $\Sigma = \{0,1\}$

Sol?

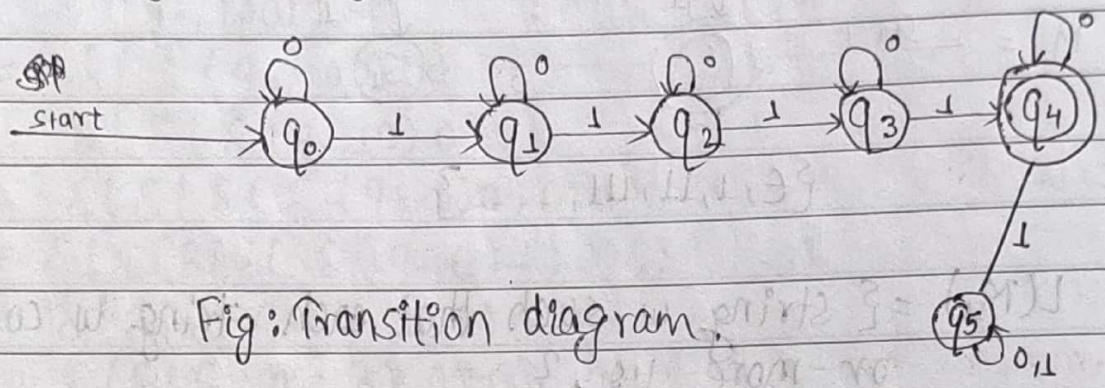


Fig: Transition diagram.

δ / Σ	0	1
→ q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_2	q_3
q_3	q_3	q_4
* q_4	q_4	q_5
q_5	q_5	q_5

Fig: Transition table.

Puspanjali
Date / /
Page

2. Example 2:
Construct a DFA that accepts string containing exactly one 1's over alphabet {0,1}.

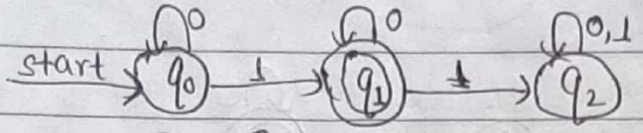


Fig: Transition diagram

3. Design a DFA that accept a language containing the string whose second symbol is '0' and fourth symbol is 1.

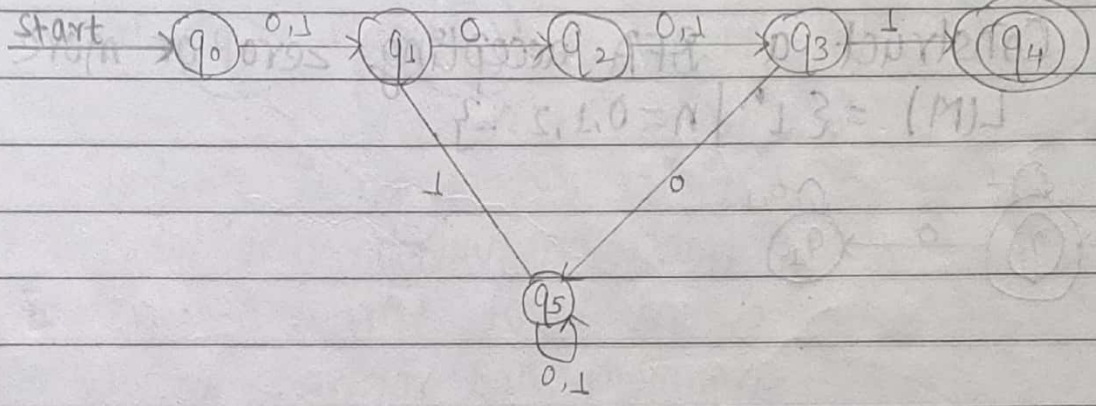


Fig: Transition diagram

Teacher's Signature

Example:
Construct a DFA accepting all string over $\Sigma = \{0,1\}$ ending with 3 consecutive 0's.

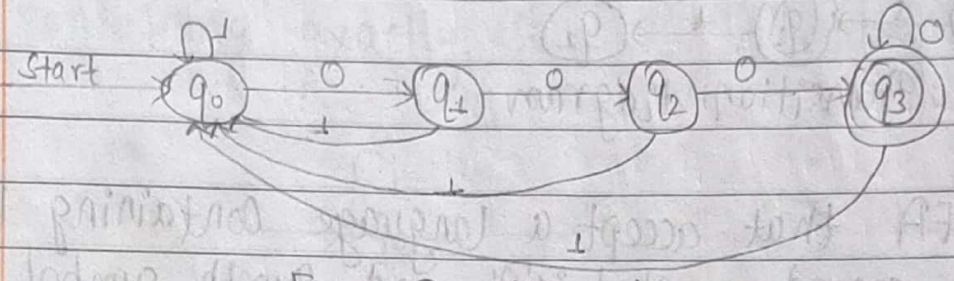
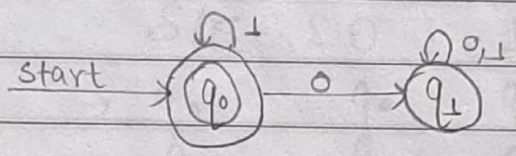


Fig: Transition diagram

Example:
Construct a DFA accepting zero or more consecutive one. $L(M) = \{1^n \mid n = 0,1,2...\}$.



Example: Construct a DFA over $\{a,b\}$ that accept a string ending with abb and not ending with abb.

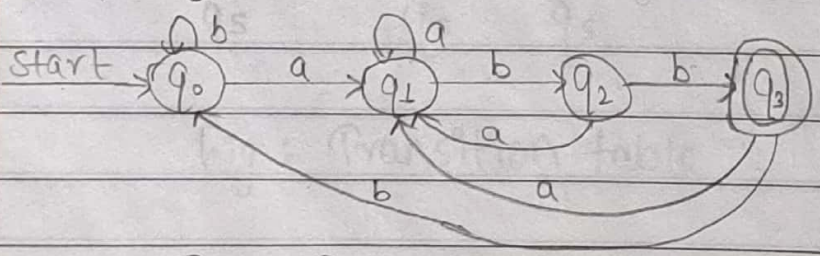


Fig: Transition diagram ending with abb.

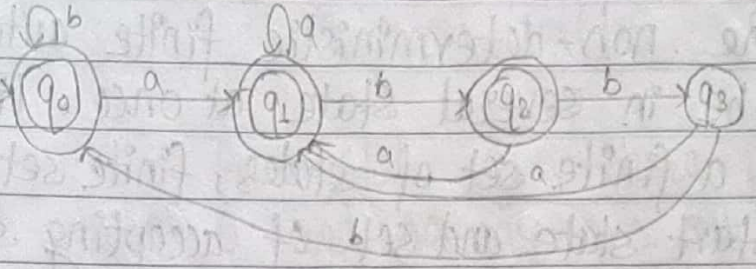
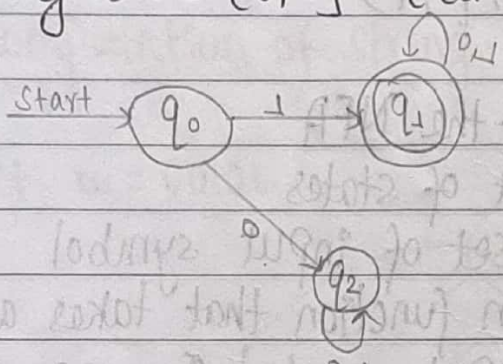


Fig: Transition diagram not ending with abb.

Example:

Construct a DFA that accepts a language of all string over {0,1} that always begins with 1.



Teacher's Signature

Non-deterministic finite automata:

The non-deterministic finite automata has the power to be in several state at once. Like the DFA, an NFA has a finite set of states, finite set of input symbol, one start state and set of accepting state. It also has the transition function as in DFA.

Formally, a non-deterministic finite automata (NFA) is a mathematical model which is defined by the five tuples as:

$$M = (Q, \Sigma, \delta, q_0, F)$$

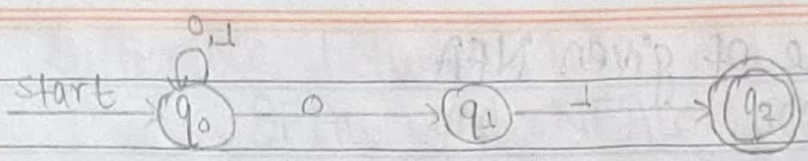
where,

M = Name of the NFA
Q = finite set of states
Σ = finite set of input symbol
δ = transition function that takes a state in Q and input symbol $\in \Sigma$ as an argument and returns the set of states.
i.e. $Q \times \Sigma \longrightarrow 2^Q$
q_0 = start state i.e. $q_0 \in Q$
F = set of final states $F \subseteq Q$.

Unlike the DFA, the transition function in NFA takes the NFA from one state to several state just with a single input.

For example:-

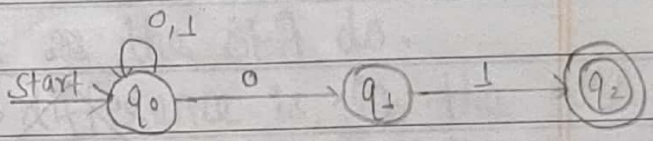
NFA that accept all string that end in 01 is presented below:



Here from state q_1, there is no any arc for input symbol 0 and no any arc out of q_2 for 0 and 1. From this, it is clear that in NFA, there may be 0 no. of arc or transition out of each state for each input symbol. While in DFA, it has exactly one arc out of each state for each input symbol.

Computation of string:

Let w = 00101.



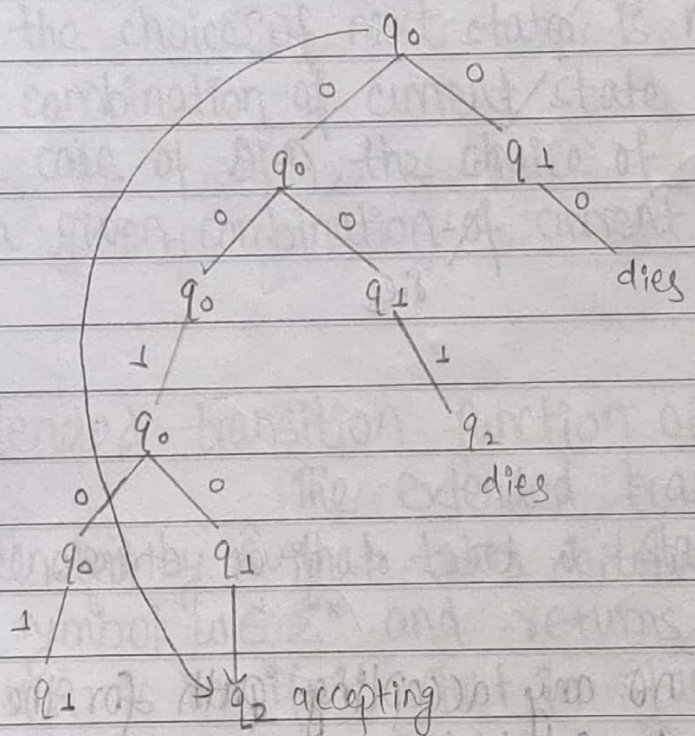


Fig: Computation tree of given string.

Teacher's Signature

Transition table of given NFA

δ/Σ	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{\phi\}$	$\{q_2\}$
* q_2	$\{\phi\}$	$\{\phi\}$

Computation tree

let w = 0110

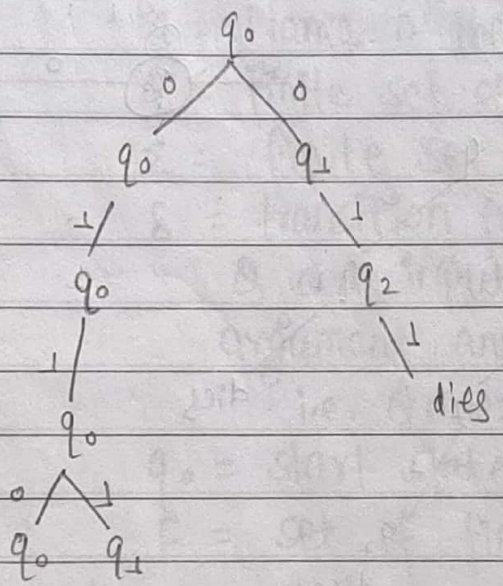


Fig: Computation tree of given string.

Hence, there is no any accepting path for the given string from start state to final state. Hence, string is rejected.

Imp:: **Difference between NFA and DFA**

DFA's and NFA's accept exactly the same set of language i.e. regular language. This means that non-determinism doesnot make a finite automaton more powerful. Any language accepted by NFA can also be accepted by some DFA and vice-versa. Thus DFA and NFA have the same capability.

The only difference between the DFA and the NFA is in the type of δ. For NFA, δ is a function that takes a state and input symbol as a argument and returns a set of 0, 1 or more states rather than returning exactly one states as the DFA do.

The another difference is, in the case of NFA the choice of next state is not deterministic for a given combination of current state and input symbol but in the case of DFA, the choice of next state is deterministic for a given combination of current state and input symbol.

Extended transition function of NFA:

The extended transition function of NFA is also denoted by $\hat{\delta}$ that takes a state $q \in Q$ and string of input symbol $w \in \Sigma^*$ and returns the set of states that the NFA is in, if it start in state q and process the string w. Formally we define $\hat{\delta}$ for an NFA by induction as follows:

Basis step:

$\hat{\delta}(q, \epsilon) = \{q\}$ i.e. Reading no input symbol remains into same state.

Induction step:

Suppose w is the string from Σ^* of the form $w = xa$, where x is the substring of w without last symbol 'a'.

Also suppose that,

$$\hat{\delta}(q, x) = \{P_1, P_2, P_3, \ldots, P_k\}$$

and,

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

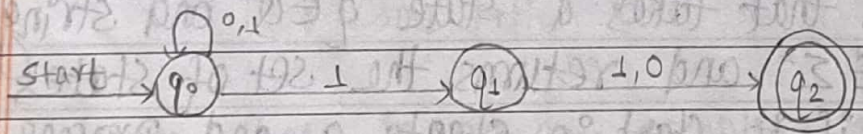
Then,

$$\hat{\delta}(q, w) = \{r_1, r_2, r_3, \ldots, r_k\}$$

Thus, to compute $\hat{\delta}(q, w)$ we first compute $\hat{\delta}(q, x)$ and then following any transition from any of these states i.e. labelled a.

Example:

Consider the following NFA:



Computing $\hat{\delta}(q_0, 01010)$

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

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

$\hat{\delta}(q_0, 1) = \{q_0, q_1\}$

$\hat{\delta}(q_0, 010) = \delta(q_0, 0) \cup \delta(q_1, 0)$
$= \{q_0\} \cup q_2$
$= \{q_0, q_2\}$

$\hat{\delta}(q_0, 0101) = \delta(q_0, 1) \cup \delta(q_2, 1)$
$= \{q_0, q_1\} \cup \phi$
$= \{q_0, q_1\}$

$\hat{\delta}(q_0, 01010) = \delta(q_0, 0) \cup \delta(q_1, 0)$
$= \{q_0\} \cup \{q_2\}$
$= \{q_0, q_2\}$

Since the result of above computation returns the set of state $\{q_0, q_2\}$ which include the accepting state q_2 of NFA show the string 01010 is accepted by above NFA.

Language of NFA:

An NFA accept a string w if it is possible to make any sequence of choices of next state while reading the symbol of w and go from start state to any accepting state. Formally, the language of NFA $M = \{Q, \Sigma, \delta, q_0, F\}$ is denoted by $L(M)$ and defined as: $L(M) = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \phi\}$

Teacher's Signature

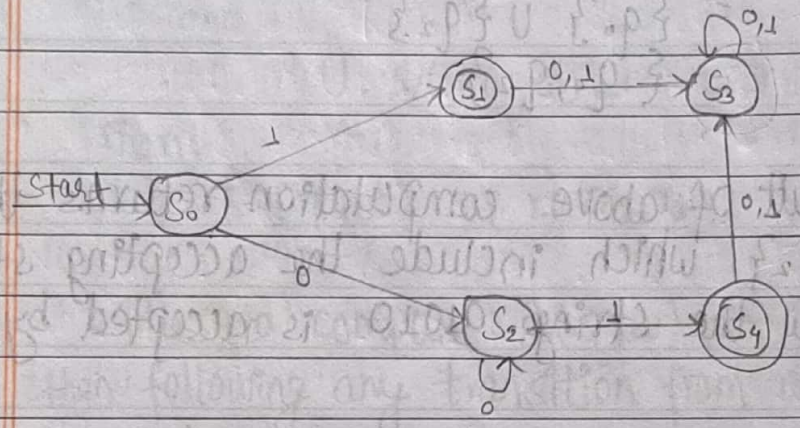
i.e. L(M) is the set of string w in Σ^* such that $\hat{\delta}(q_0, w)$ contains at least one accepting state.

Imp... **Equivalence of finite Automata:**

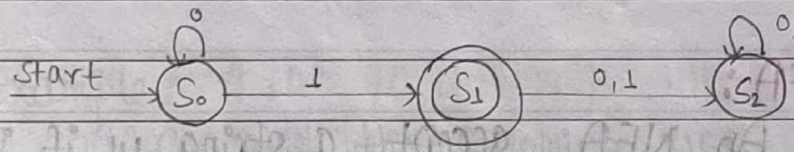
Two finite state automata are called equivalent if they recognize the same language. More formally, any two finite automata say M_1 and M_2 are equivalent if $L(M_1) = L(M_2)$.

For example:- following two finite automata are equivalent,

M_1 =



M_2 =



Here,

$L(M_1) = \{1, 01, 001, 0001 \ldots\}$

$= \{w | w$ consist of zero or more number of 0 is followed by 1$\}$.

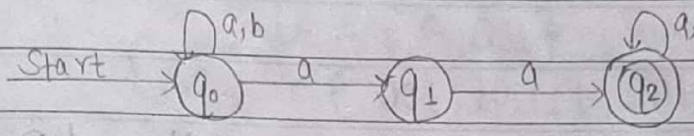
$L(M_2) = \{1, 01, 001, \ldots\}$

$\therefore L(M_1) = L(M_2)$

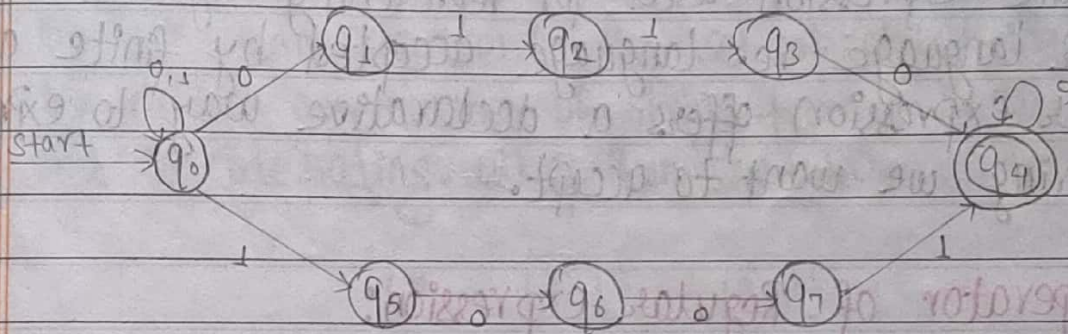
Hence, M_1 and M_2 are equivalent.

Examples of NFA:

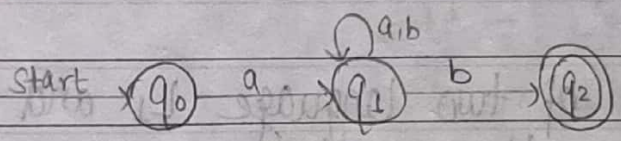
1. Construct a NFA over $\{a, b\}$ that accept the string having aa as substring.



2. Construct a NFA over $\{0, 1\}$ that contains substring 0110 or 1001.

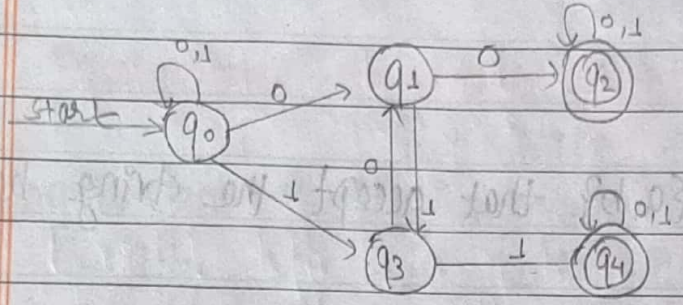


3. Construct the NFA accept string with a ending with b.



Teacher's Signature

4. Construct a NFA for the language over {0,1} that have at least two consecutive 0's or 1's.



Regular expression:

Regular expression are another type of language defining notation. Regular expression are algebraic expression used for representing regular set or regular language i.e. language accepted by finite automata. Regular expression offers a declarative way to express the string we want to accept.

The operator of Regular expression:

Basically, there are three operators that are used to generate the language that are regular.

1) Union (U, +, V):-

The union of two language L_1 and L_2 is denoted by $L_1 \cup L_2$ is the set of string that are either in L_1 or L_2 or both.

For example:-

$L_1 = \{001, 10, 111\}$, $L_2 = \{\epsilon, 001\}$

Teacher's Signature

$L_1 \cup L_2 = \{\epsilon, 001, 10, 111\}$

ii) Concatenation (·):-

The concatenation of two language L_1 and L_2 is the set of string that can be formed by taking any string in L_1 and concate it with any string in L_2.

For example:-

$L_1 = \{001, 10, 111\}$

$L_2 = \{\epsilon, 001\}$

$L_1 \cdot L_2 = \{001, 10, 111, 001001, 10001, 111001\}$

iii) Kleen closure (*) and positive closure (+):-

The closure (*) of a language L is denoted by L^* and represents the set of string that can be formed by taking any number of string from L possibly with repetition and concatenating all of them i.e.

$$L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L^1 \cup L^2 \ldots\ldots$$

where,

$L^0 = \{\epsilon\}$

$L^1 = \{L\}$

$L^i = LLL\ldots\ldots$ (i.e. the concatenation of i copies of L)

For example:-

$L = \{0, 11\}$

$L^* = \{\epsilon, 0, 11, 011, 00, 110, 1111 \ldots\ldots\}$

Teacher's Signature

And, positive closure of any language L is denoted by L^+ and defined as;

$$L^+ = \bigcup_{i=1}^{\infty} L^i = L^1 \cup L^2 \cup \ldots = L^* - \{\epsilon\}$$

Formal definition of Regular expression:

The regular expression over alphabet or set Σ are defined inductively as:

Basis step:

1. The symbol ϕ and ϵ are regular expression representing the language ϕ and $\{\epsilon\}$ respectively i.e. $L(\epsilon) = \epsilon$ and $L(\phi) = \phi$.
2. If a is a symbol in Σ then a is regular expression representing the language $\{a\}$ i.e. $L(a) = \{a\}$.

Induction step:

1. If r and s are regular expression.
   - r+s is a regular expression denoting the union of L(r) and L(s) i.e. $L(r+s) = L(r) \cup L(s)$
   - rs is a regular expression denoting the concatenation of L(r) and L(s) i.e. $L(rs) = L(r) \cdot L(s)$.
   - r^* is a regular expression denoting the closure of L(r) i.e. $L(r^*) = (L(r))^*$
   - (r) is a regular expression denoting the same language

as r. i.e. $L((r)) = L(r)$

Precedence of regular expression operation:-

Precedence means that operator are associated with their operands in particular order. For the regular expression, the following is the order of precedence for the operator.

1) The '*' operator is the highest precedence i.e. it applies only to smallest sequence of symbol to its left that is well formed regular expression

2) Next precedence comes the concatenation of dot (•) operator.

3) Next precedence comes the union operator (∪).

Regular language or regular set:

The regular language or regular set are those that can be formed using the operation of concatenation, union and kleen closure in arbitrary order starting with empty set, empty string and single term set. The regular language are those language that can be recognized by using finite state automaton. The set represented by regular expression are called regular set or regular language.

Teacher's Signature

Formally,

Let Σ be an alphabet, the class of regular language over Σ is defined inductively as:

- ϕ is a regular language representing empty language.
- $\{\epsilon\}$ is a regular language representing the language of empty string.
- For each $a \in \Sigma$, $\{a\}$ is regular language.
- If $L_1, L_2, \ldots, L_n$ is a regular language then so is $L_1 \cup L_2 \cup \ldots \cup L_n$.
- If $L_1, L_2, \ldots, L_n$ is a regular language then so is $L_1 \cdot L_2 \cdot L_3 \cdots L_n$
- If L is regular language then so is L^*.

Similarities in regular expression and automata:

The regular expression approach for describing language is fundamentally different from finite automaton approach. However, those two notation turn out to represent exactly the same set of language which we call regular language. A regular expression is one way of describing finite state automata. Both regular expression and finite automata are used to describe regular language so both have the equivalent expressive power i.e.

For every regular expression R, there is a corresponding finite automaton that accept the set of string generated by R and for every finite automaton A, there is the corresponding regular expression that generates the

set of string generated by A.

Algebric rules for regular expression:

1) **Commutativity :-**

If r and s are regular expression representing the language L(r) and L(s) then comutativity

$$r+s = s+r$$
$$r \cdot s \neq s \cdot r$$

2) **Associativity :-**

The union as well as concatenation of regular expression are associative. If r,s,t are regular expression representing the language L(r), L(s), L(t) then;

$$t+(r+s) = (t+r)+s$$
$$t \cdot (r \cdot s) = (t \cdot r) \cdot s$$

3) **Distributive :-**

A distributive law involves two operators and assists that one operator can be pushed down to be applied to each argument of other operator individually i.e. for any regular expression r,s,t representing the regular language L(r), L(s) and L(t) then

$$r.(s+t) = rs+rt \quad ; \text{left distribution}$$
$$(s+t).r = sr+tr \quad ; \text{right distribution}$$

Teacher's Signature

iv) Identity law:-

An identity for a operator is a value such that when the operator is applied to the identity and some other value, the result is the other value.

ϕ is an identity for union i.e.

$r + \phi = r$.

ϵ is the identity for concatenation.

i.e. $r.\epsilon = \epsilon.r = r$

v) Idempotent law:-

An operator is said to be idempotent if the result of applying it to two of the same values as a argument then the result is also the same value.

For any regular expression r, representing the regular language L(r) then,

$r + r = r$.

vi) Law of closure:-

For any regular expression r, representing the regular language L(r), the following rule holds

$(r^*)^* = r^*$

$\phi^* = \epsilon$

$\epsilon^* = \epsilon$

$r^* = r^+ + \epsilon$

$r^+ = r^* - \{\epsilon\}$

Teacher's Signature

Example:-

1. What are the string in a regular set specified by following regular expression over set & $\Sigma = \{0,1\}$

i) 0^*

i.e. $0^* = \{\epsilon, 0, 00, 000, 0000, \ldots\}$

ii) 1^*

i.e. $1^* = \{\epsilon, 1, 11, 111, 1111, \ldots\}$

iii) 01^*

i.e. $01^* = \{0, 01, 011, 0111, 01111, \ldots\}$

i.e. string start with 0 and followed by any number 1.

iv) 0^*10^*

i.e. $0^*10^* = \{1, 01, 010, \ldots\}$

i.e. The string having exactly 1 1s.

v) $(0+1)$

i.e. $(0+1) = \{0, 1\}$

vi) $(0+1)^*$

i.e. $(0+1)^* = \{0,1\}^* = \{\epsilon, 0, 1, 00, 11, 10, 000, \ldots\}$

vii) $(10)^*$

i.e. $(10)^* = \{\epsilon, 10, 1010, 101010, \ldots\}$

i.e. The set of string having equal number of 0 and 1.

2. Find the language represented by regular expression.

a) $01^* + 1$

Here,

$$L(01) = L(0) \cdot L(1)$$
$$= 0 \cdot 1$$
$$= 01$$

$$L(01^*) = L(0) \cdot L(1^*)$$
$$= 0 \cdot \{\epsilon, 1, 11, 111, \ldots\}$$
$$= \{0, 01, 011, 0111, \ldots\}$$

$$L(01^* + 1) = L(01^*) + L(1)$$
$$= \{0, 01, 011, 0111\} + \{1\}$$
$$= \{0, 1, 01, 011, 0111 \ldots\}$$

b) $(0^*1)^*$

Here,

$$L(0^*1) = L(0^*) \cdot L(1)$$
$$= \{\epsilon, 0, 00, 000, \ldots\} \cdot 1$$
$$= \{1, 01, 001, 0001, \ldots\}$$

$$L((0^*1)^*) = \{\epsilon, 1, 01, 001, 0001, \ldots\}$$

More examples of Regular expression:

Consider $\Sigma = \{0,1\}$ then some regular expression over Σ are:

i) 0^*10^* $\{w|w$ contains a single $1\}$
$= \{1, 010, 00100 \ldots\}$

ii) $\Sigma^*1\Sigma^*$ $\{w|w$ contain at least one 1s$\}$
$= \{1, 010, 111 \ldots\}$

iii) $\Sigma^*001\Sigma^*$ $\{w|w$ contain 001 as a substring$\}$
$= \{001, 0001, 00010 \ldots\}$

iv) $(\Sigma\Sigma)^* = (00+01+10+11)^*$ $\{w|w$ has the even length$\}$
$= (00, 01, 10, 11)$

v) $0^*10^*10^*10^*$ $= \{w|w$ contain exactly 3 1s.$\}$

vi) $1^*(01^*01^*)^*$ $= \{w|w$ contain even number zero$\}$

Find the regular expression for the following:

i) for string that have substring either 001 or 100.

Sol: First regular expression for the set of string having 001 as substring.

$$(0+1)^*001(0+1)^*$$

Secondly regular expression for the set of string having 100 as substring.

$$(0+1)^* 100 (0+1)^*$$

Now,

Regular expression for string having either 001 or 100 as substring.

$$(0+1)^* 001 (0+1)^* + (0+1)^* 100 (0+1)^*.$$

2) For string ending with 11.

→ $(0+1)^* 11$.

3) Regular expression that denotes C identifier.

Sol: Let us assume that C identifier start with alphabet or _ and contains any combination of alphabet, digit and underscore then required regular expression:

(alphabet + _) (digit + alphabet + _)*

4) Even number of a's is followed by odd number of b. $\Sigma = \{a, b\}$

Sol: Let first write the regular expression for even number of aa.

$$(aa)^*$$

Secondly the regular expression for odd number of b's.

$$(bb)^* b$$

Now, after concatenating these we obtain the regular expression for given problem.

i.e. $(aa)^* . (bb)^* b$

Puspanjali
Date / /
Page

Exam: Let $S = \{0,1\}$. give the regular expression corresponding to the regular set given below:

a) $\{00, 010, 0110, 01110 \ldots\}$

- Here, the required regular expression for given set is:

01^*0

b) $\{0, 001, 000, 00001, 00000, 000001 \ldots\}$

$0^+(\epsilon + 1)$

Exam: Determine whether 0101 belongs to each of these regular set.

a) 01^*0^*

→ NO.

Since the given string contains 1 at end but in the regular expression given above doesnot represent the language of string containing 1 at end.

b) $0(11)^*(01)^*$

→ NO.

c) $0^*(10)^*1^*$

→ Yes.

d) $(01)^*(11)^*$

→ Yes

Teacher's Signature

Language and grammar

Natural Language and Formal Language:

Natural language are the languages that people speak such as english, nepali, etc. The syntax of natural language is extremely complicated. Infact, it doesnot seems possible to specify all the rules of syntax for natural languages.

Formal language unlike the natural language is specified by well-defined set of rules of syntax. These are the language that are designed by the people for specific application. Formal language is the set of string of symbol that may constraint by rules that are specific to it.

A formal language L over an alphabet Σ is a subset of Σ^*. The sentences of formal language are described by using the grammar.

Some terminology:

1) **Vocabulary (or alphabet):-**

A vocabulary V is a finite non-empty set of elements called symbol.

ii) Sentence (or word):-

A word over the vocabulary V is the string of finite length of element V.

iii) The empty string or null string:-

The empty string or null string is denoted by ϵ or λ, is the string containing no symbol.

iv) Language:-

A language over vocabulary V is the subset of V^* where V^* represents all words over V.

Grammar or Syntax:-

The grammar are used to generate the word of language and to determine whether the word is in the language.

Formally, grammar for the language L is the set of rules through which string of L generated or determined whether the string is in the language L. The grammar of the language determines the form of the sentence in the language.

Grammar has the vocabulary V, which is the set of symbols used to derive the member of the language. Some element of the vocabulary cannot be replaced by other symbol, those are called terminals and other member of vocabulary which can be replaced by other symbols called

non-terminals. There is a special member of vocabulary called start symbol denoted by S. There are also some rules called production of the grammar.

Phrase structure grammar:

The phrase structure grammar is a type of generative grammar, consisting of rules determining the structure and interpretation of the sentences. Phrase structure grammar are used to generate the word of the language and to determine whether a word is in a language.

Formally, a phrase structure grammar is defined by four tuples:

$$G = (V, T, P, S)$$

where,

V = a vocabulary
T = set of terminal symbols $T \subseteq V$
P = set of production rules
S = start symbol $S \in V$

Thus, phrase structure grammar consists of collection of rules also called production. The symbol involved in the production may be non-terminal or terminal symbol. Non-terminal symbol are represented by capital letters and terminal symbol are represented by lowercase letters. One of the non-terminal is designated as start symbol.

Puspanjali
Date / /
Page

It usually occurs on the left-hand side of the top-most rule. Every production in P must contain at least one non-terminal on its left-side in phrase structure grammar.

For example:-

$S \rightarrow ABa$
$A \rightarrow BB$
$B \rightarrow ab$
$AB \rightarrow b$

This is the example of phrase structure grammar.
where, $V = \{a, b, A, B, S\}$
$T = \{a, b\}$
$S = S$ is the start symbol
P = Production given above

Derivation:-

This is the process of deriving string by applying production rule of the grammar. If the grammar G has a production $\alpha \rightarrow \beta$, we can say that $x\alpha y$ derives $x\beta y$ in G. This derivation is written as

$$x\alpha y \Rightarrow x\beta y$$

Derivation are under taken by beginning with starting symbol S of the grammar G and applying the production from P. The derivation terminates when sentencial form is obtained consists of only terminal.

Teacher's Signature

Let $G = (V, T, P, S)$ be phrase structure grammar and let, $w_0 = lz_0r$ and $w_1 = lz_1r$ are be a string then

i) if $z_0 \to z_1$ is the production of G, we say that w_1 is directly derivable from w_0 and $w_0 \Rightarrow w_1$.

ii) if $w_0, w_1, w_2 \ldots w_n$ are the string over V such that $w_0 \Rightarrow w_1 \Rightarrow w_2 \Rightarrow w_3 \Rightarrow \ldots \Rightarrow w_n$ then we say that w_n is derivable from w_0 and denoted as $w_0 \Rightarrow w_n$

From this, it is clear that the sequence of steps used to obtain some string from start symbol is called derivation.

Example of derivation:-

1. $S \to aAS \mid a$
   $A \to SbA \mid SS \mid ba$

Show the derivation of string aaabaaa.

$\Rightarrow$ $S \Rightarrow aAS$
$S \to aSSS$ ($\because A \to SS$)
$S \Rightarrow aaSS$ ($\because S \to a$)
$S \Rightarrow aaaASS$ ($\because S \to aAS$)
$S \Rightarrow aaabaSS$ ($\because A \to ba$)
$S \Rightarrow aaabaaS$ ($\because S \to a$)
$S \Rightarrow aaabaaa$ ($\because S \to a$)

Hence,
$S \overset{*}{\Rightarrow} aaabaaa$ i.e. $aaabaaa \in L(G)$

2. Consider a grammar

$G = (V, T, P, S)$

where, $V = \{a, b, A, B, S\}$

$T = \{a, b\}$

S is start symbol

$P = \{S \to Aba \mid AB \mid ab, A \to BB, B \to ab\}$

Show the derivation of string ab, ababab.

Sol:

Derivation for ab.

$S \Rightarrow ab \quad (S \to ab)$

Derivation for ababab.

$S \Rightarrow AB \quad (\because S \to AB)$

$S \Rightarrow BBB \quad (\because A \to BB)$

$S \Rightarrow abBB \quad (\because B \to ab)$

$S \Rightarrow ababB \quad (\because B \to ab)$

$S \Rightarrow ababab \quad (\because B \to ab)$

Hence,

$S \overset{*}{\Rightarrow} ababab$

So, $ababab \in L(G)$.

Language of grammar:-

Let $G = (V, T, P, S)$ is a grammar then language of grammar G is denoted by $L(G)$ is the set of terminal string that have the derivation from the start symbol in G i.e. $L(G) = \{w \in T^* \mid S \overset{*}{\Rightarrow} w\}$

Teacher's Signature

Thus, the set of all string that can be derived from the grammar is said to be the language generated from that grammar.

Imp:: **Example 1:**

Let G be the grammar with $V = \{S, A, a, b\}$, $T = \{a, b\}$, S is the start symbol, $P = \{S \to aA, S \to b, A \to aa\}$ What is $L(G)$?

Sol:

Here,

$S \Rightarrow b \quad (\because S \to b)$

So, $b \in L(G)$

$S \Rightarrow aA \quad (S \to aA)$

$S \Rightarrow aaa \quad (A \to aa)$

$aaa \in L(G)$

Hence,

$L(G) = \{b, aaa\}$

Example 2:

Let G be the grammar with

$V = \{S, 0, 1\}$

$T = \{0, 1\}$

S is start symbol

$P = \{S \to 1S, S \to 0\}$

What is $L(G)$.

Scanned with CamScanner

Sol:

Here,

$S \Rightarrow 0$

$0 \in L(G)$

$S \Rightarrow 1S$

$S \Rightarrow 110$

$110 \in L(G)$

$S \Rightarrow 1S$

$S \Rightarrow 11S$

$S \Rightarrow 1110$

$1110 \in L(G)$

From above it is clear that;

$L(G) = \{0, 110, 1110, 111110, \ldots\}$

i.e. set of all string beginning with even number of 1's and end with 0.

Example 3:

What is language of following grammar.

$S \rightarrow \epsilon$

$S \rightarrow 0S1$

Sol:

Here,

$S \Rightarrow \epsilon$

$\epsilon \in L(G)$

$S \Rightarrow 0S1$

$S \Rightarrow 01$
$S \Rightarrow 0S1$
$S \Rightarrow 00S11$
$S \Rightarrow 0011$

From above it is clear that:
$L(G) = \{\in, 01, 0011, 000111, \ldots\}$
i.e. string with equal no of zero is followed by equal no of 1.

Exam:
Let $G = (V, S, V_0, \rightarrow)$ where $V = \{V_0, x, y, z\}$
$S = \{x, y, z\}$
and V_0 is start symbol.

$V_0 \rightarrow xV_0$
$V_0 \rightarrow yV_0$
$V_0 \rightarrow z$

What is $L(G)$?

Sol:

Here,

$V_0 \Rightarrow xV_0$ $\quad$ $V_0 \Rightarrow yV_0$
$V_0 \Rightarrow xz$ $\quad$ $V_0 \Rightarrow yz$
$V_0 \Rightarrow z$

i.e. $xz \in L(G)$ $\quad$ $yz \in L(G)$ $\quad$ $z \in L(G)$

$V_0 \Rightarrow xV_0$
$V_0 \Rightarrow xxV_0$
$V_0 \Rightarrow xxz$

From above;

$L(G) = (xz, yz, z, xxyz, xxz, yyz, xyz \ldots)$

i.e. string start with either x or y and always end with z.

Q. Construct a grammar to generate the set $\{0^m 1^n \mid m$ and n are non-negative integer$\}$.

Soln:

$A \to \epsilon$

$A \to 0A$

$A \to A1$

where, $V = \{A, 0, 1\}$

$T = \{0, 1\}$

P = given above

$S = A$

Example:

Construct a grammar which generates a palindrom for binary number.

Soln:

$S \to 0$

$S \to 1$

$S \to \epsilon$

$S \to 0S0$

$S \to 1S1$

where, S is start symbol

$V = \{S, 0, 1\}$

$T = \{0, 1\}$

Teacher's Signature

Construct a grammar of the language wcw^R where w is a string over $\{a, b\}$

Sol^n:

$S \rightarrow aSa$

$S \rightarrow bSb$

$S \rightarrow c$

where, $V = \{a, b, c\}$

$T = \{a, b\}$

P = given above

S = start symbol

Types of Phrase structure grammar:-

Phrase structure grammar can be classified according to the types of production that are allowed. Noum Chomsky, a founder of formal language theory provide an initial classification of the phrase structure grammar with the following hierarchy:

i) Type 0 (unrestricted grammar)
ii) Type 1 (Context sensitive grammar)
iii) Type 2 (Context free grammar)
iv) Type 3 (Regular grammar)

1) Type 0 (Unrestricted grammar):

An unrestricted grammar is a formal grammar on which no restrictions are made on the left and right side of the grammar's production. The type 0 grammar generates type 0 language called recursively

enumerable language that can be recognized by the turing machine.

The unrestricted grammar is defined as:

$G = (V, T, P, S)$

where, V = finite set of non-terminal
T = finite set of terminals
S = start symbol
P is the set of production of the form $\alpha \rightarrow \beta$
where,
α and β are string of terminals and non-terminals with $\alpha \neq \epsilon$

Example:-

$S \rightarrow AbC$

$Ab \rightarrow c$

$cC \rightarrow Sc \mid \epsilon$

This above grammar accept empty string or string containing c.

II) Type 1 (Context sensitive grammar):

Type 1 grammar generates the context sensitive language. The context sensitive grammar is defined as

$G = (V, T, P, S)$

where,
V = finite set of non-terminals
T = finite set of terminals
S = starting symbol that is non terminals.

P: production rule of the form;

$\alpha A \beta \rightarrow \alpha \gamma \beta$.

where, α and β are string of zero or more terminals and non-terminals symbol with γ must be non empty.

The rule $S \rightarrow \epsilon$ is allowed if S does not appear on right side of the rule i.e. when there is the production in the grammar of the form $\alpha A \beta \rightarrow \alpha \gamma \beta$ the grammar is called context sensitive or Type 1 because A can be replaced by γ only when it is surrounded by string α and β.

The language described by those grammar are exactly recognized by a linear bounded automaton.

Example:-

$S \rightarrow aSBC \mid aBC$

$BC \rightarrow HB$

$HB \rightarrow HC$

$aB \rightarrow ab$

$bB \rightarrow bb$

$bC \rightarrow bc$

$cC \rightarrow cc$

Teacher's Signature

III) Type 2 (Context free grammar):

Type 2 grammar are the context free grammars and generates the context free language.

Context free grammar is defined as;

$G = (V, T, P, S)$

where, V = finite set of non-terminal symbols

T = set of terminal symbols

S = start symbol that is non terminal

P is the set of production rule of the form $A \rightarrow B$; where A is a single symbol that is non terminal, B is the string of terminal or non-terminals.

The language generated by context free grammar is context free language and can be recognized by push-down automata.

Example:-

$S \rightarrow SS$

$S \rightarrow (S)$

$S \rightarrow ()$

IV) Type 3 (Regular grammar):

Type 3 grammar generates the regular language. A regular grammar may be left linear or right linear. If all the production are of the form

$A \rightarrow wB$ or, $A \rightarrow w$

Teacher's Signature

where, A and B are non-terminals and $w \in T^*$ Then we say that grammar is right linear.

If all the production of context free grammar (CFG) are of the form

$A \rightarrow Bw$

or, $A \rightarrow w$

Then we called it as left linear. Thus if

Thus, if the grammar is left linear or right linear then that grammar is called regular grammar.

Example:-

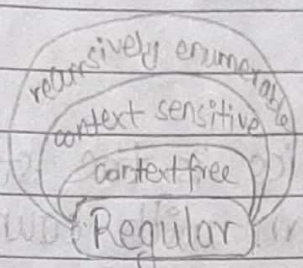
$S \rightarrow abS \mid a$ } Right linear

$S \rightarrow S_1 a$
$S_1 \rightarrow S_1 ab \mid \epsilon$ } Left linear

Chomsky hierarchy:-

The Noum-Chomsky, a founder of formal language theory provide an initial classification of language into four types. Type 0 language are those generated by unrestricted grammar i.e. recursively enumerable language. Type 1 consists of context sensitive language, Type 2 consists of context free language and Type 3 consists of regular language as in figure.

Teacher's Signature



Unit 3:
Recurrence relation

Combinatorics:-

Combinatorics is the study of arrangement of object which concern with counting problems. Technique for counting are important in Mathematics and Computer science. We must count the object to solve many different types of problems. For example: counting is used to determine the complexity of algorithms, counting is also required to determine whether there are enough telephone number or Internet Protocol addresses to need the demand.

Basic Counting principle:

There are two basic counting principles that can be used to solve the counting problem, they are:

1) Sum rule principle
2) Product rule principle

1) Sum rule principle:

If a task can be done either in one of n_1 ways or in one of n_2 ways, where none of the set of n_1 ways is same as any of the set of

n_2 ways then there are $n_1 + n_2$ ways to do the task.

We can extend the sum rule as:

Suppose that a task can be done in one of n_1 ways or in one of n_2 ways, ..., or in one of n_m ways where none of the set of n_i ways of doing the task is same as any of the set of n_j ways for all pair i and j. The number of ways to do the task is $n_1 + n_2 + \ldots + n_m$

Example:-

1) If there are 24 boys and 18 girls in a class find the number of ways of selecting one student as a class representative.

Sol:

The one student is selected either from boys or from the girls. So, by using the sum rule, there are

$24 + 18 = 42$ ways of selecting one student as a class representative.

2) How many ways can we draw a heart or diamond from an arbitrary desk of playing card.

Sol:

There are total 13 card of heart and 13 card of diamond. So there are 13 ways of selecting a heart and 13 ways of selecting a diamond. So by

So by sum rule total no of ways of picking heart or diamond is $13 + 13 = 26$.

3. A student can choose a computer project from one of 3 list. The 3 list contain 23, 25, 19 project respectively. How many ways a student can choose his project?

Sol:

The student can choose a project from the first list in 23 ways or from the 2nd list in 25 ways or from the 3rd list in 19 ways. Hence, by the sum rule $23+25+19 = 67$ ways of choosing his project.

4. How many ways can we get sum 4 or sum 8 when two distinguishable dice are rolled?

Sol:

Since dice are distinguishable, outcome (1,3) is different from (3,1). So to get 4 as a sum we have the pair (1,3), (2,2) and (3,1). so total of 3 ways.

And similarly, getting 8 from pair (2,6) (6,2) (3,5) (5,3) (4,4). So total of 5 ways.

Hence, by sum rule getting sum of 4 or 8 is $3+5 = 8$ ways.

Puspanjali
Date / /
Page

11) Product rule principle:

Suppose that a procedure can be broken down into sequence of two tasks. If there are n_1 ways to do the first task and for each of these ways of doing first task there are n_2 ways to do the second task then there are $n_1 \cdot n_2$ ways to do the procedure.

We can extend the product rule as:

Suppose that procedure is carried out by performing the task $T_1, T_2, T_3, \ldots, T_m$ in sequence. If each task T_i can be done in n_i ways independent of previous task done then there are $n_1 \cdot n_2 \cdot n_3 \ldots n_m$ ways to carried out the procedure.

Example:

1) An office building contains 27 floors and has 37 offices on each floor. How many offices are there in the building?

Sol:

No. of floors in a office building = 27
No. of offices in each floor = 37
Now,
By using product rule total no. of offices are: 27×37
$= 999$

11) The char of auditorium are to be labelled with a letter and a positive integer not exceeding 100. What is the largest number of chair that can be labelled differently?

Puspanjali
Teacher's Signature

Soln:

The procedure of labelling a chair consists of two task namely assigning one of the 26 letter and then assigning one of the 100 possible integer. Hence by the product rule, there are $(26 \times 100) = 2600$ ways of labelling the chair

iii) How many different licence plates are available if each plate contains a sequence of 3 letters followed by 3 digits?

Soln:

There are 26 choices for each of the 3 letters and 10 choices for each of the 3 digits. Hence, by product rule there are total of $(26 \times 26 \times 26 \times 10 \times 10 \times 10) = 1,75,76,000$ ways

iv) How many 3 digit number can be formed using the digit. 1, 3, 4, 5, 6, 8 and 9 if no digit can be repeated?

Soln:

There are seven ways of choosing a digit in 100 position, but once one digit is used it is not available for 10^{th} position. So there are 6 ways of choosing digit for tenth position and there are 5 ways of choosing digit for unit position. Hence by product rule there are total of $(7 \times 6 \times 5) = 210$ ways.

Counting problem that uses both sum rule and product rule:

Example:

How many different license plate are there that involve 1, 2 or 3 letter followed by 4 digits.

Sol?

We can form plates with 1 letters followed by 4 digits in:
$(26 \times 10 \times 10 \times 10 \times 10) = 260000$ ways.

We can form plates with 2 letters followed by 4 digits in:
$(26 \times 26 \times 10 \times 10 \times 10 \times 10) = 6760000$ ways

We can form plates with 3 letters followed by 4 digits in:
$(26 \times 26 \times 26 \times 10 \times 10 \times 10 \times 10) = 175760000$ ways

So, by sum rule, total no. of license plates are:
$(260000 + 6760000 + 175760000) = 182780000.$

Inclusion-exclusion principle:

Suppose a task can be done in n_1 or n_2 ways. But sum of the set of n_1 ways to do the task are same as sum of the n_2 other ways to do the task. In this situation we cannot use the sum rule to count the no. of ways to do the task.

To correctly count the no. of ways to do the task we add the no. of ways to do it in one way and

no. of ways to do it in another ways and then subtract the no. of ways that is both among the set of n_1 ways and set of n_2 ways. This principle is called principle of inclusion-exclusion. Sometimes, it is also called subtraction principle of counting.

Set representation of inclusion:

Let A_1 and A_2 be a sets and there are $|A_1|$ elements on a set A_1 and $|A_2|$ elements on a set A_2 then total number of elements in the set $A_1 \cup A_2$ is the sum of elements in the set A_1 and A_2 minus number of elements on the both set A_1 and A_2.

i.e. $|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2|$

This principle hold true for any no. of set. For three finite set A_1, A_2 and A_3 then,

$$|A_1 \cup A_2 \cup A_3| = |A_1| + |A_2| + |A_3| + |A_1 \cap A_2 \cap A_3| - |A_1 \cap A_2| - |A_1 \cap A_3| - |A_2 \cap A_3|$$

Example:

How many bit string of length eight either start with a 1 or end with two bits 00.

Soln:

The number of bit string of length eight begins with 1 is $2^7 = 128$.

The number of bit string of length eight end with 00 is $2^6 = 64$.
The number of bit string of length eight and end with 00 is $2^5 = 32$

So, by inclusion-exclusion principle;
Total bit string of length eight either start with 1 bit or end with 00 is

$$128 + 64 - 32 = 160$$

1 ..... | ..... 00 | ... 00
2^7 choice | 2^6 | 1 | 2^5 | 1

V.Imp: **The pigeonhole principle:**

The pigeonhole principle states that, if there are more pigeons than pigeonholes then there must be at least one pigeonhole with at two pigeons on it. The concept of pigeon can be extended to any objects.

Theorem 1:

If k is a positive integer and k+1 or more object are placed into k boxes, then there is at least one box containing two or more of the object.

Proof:

We will prove the pigeonhole principle using the proof by contradiction.

Suppose that k+1 or more objects are placed into k boxes and no boxes contain more than one object

in it. There are k boxes then there must be k objects such that there are no two objects in the box. This contradicts our assumption. So there is at least one box containing two or more of the object.

proved.

Example:

Show that if there are 27 english word then at least two word begins with same letters.

Proof:

There are 27 english word but we have only 26 letter in english alphabet that can be used in the beginning of the word. So by Pigeonhole principle there must be at least two word that begins with the same letters.

Generalized Pigeonhole principle:

If N objects are placed into k boxes then there is at least one box containing at least $\lceil N/k \rceil$ objects.

Proof:

Suppose N objects are placed into k boxes and there is no box containing more than one $\lceil N/k \rceil - 1$ objects.

So total number of objects is at most

$$k\left(\left\lceil \frac{N}{k} \right\rceil - 1\right) < k\cdot\left(\left(\frac{N}{k}+1\right) - 1\right) = N$$

$$\therefore k\left(\left\lceil \frac{N}{k} \right\rceil - 1\right) < N$$

This is contradiction that N objects are placed into k boxes. Hence, theorem is proved.

Note:

The formula for finding the minimum number of objects N such that one of the k boxes contains r object is

$$N = [\, k(r-1) + 1 \,]$$

Example:

1) Given a group of 100 people. At least how many people were born in the same month.

Sol:

Number of month (box) = 12
Number of people (object) = 100
So by pigeonhole principle;
At least $\left\lceil \frac{100}{12} \right\rceil = 9$ people have same birth month.

2) If a class has 24 student, what is the maximum number of possible grading that must be alone to ensure that at least two student with the same grade.

Sol:

Total number of student (N) = 24
Total number of grade be k (let)
i.e.

$$\left\lceil \frac{24}{k} \right\rceil = 2$$

We know that for satisfying this

$$N = \lceil k(r-1)+1 \rceil$$

or, $24 = k \cdot 1 + 1$

$\therefore k = 23$

Hence, to ensure at least two student with the same grade we need maximum no. of 23 grades for 24 students.

3) Find the minimum number of student required in a class to be sure that 3 of them are born in same month.

Sol:

$k = 12$

$r = 3$

By,

$$N = \lceil k(r-1)+1 \rceil$$
$$= \lceil 12(3-1)+1 \rceil$$
$$= 12 \times 2 + 1$$
$$= 24 + 1$$
$$= 25 \text{ Ans}$$

Permutation and Combination:

Permutation:

Permutation of a set of distinct object is an ordered arrangement of these object. An ordered arrangement of r object from a set of n object is called r-permutation of n object and is denoted by $P(n,r)$ or ${}^nP_r$.

For example;

Let $S = \{a, b, c\}$ then 2-permutation of S are ordered arrangement ab, ac, bc, ba, ca, cb.

Hence, there are six 2-permutation of this set with 3 elements.

Theorem:

The total number of permutation of set of n object taking r objects at a time is given by:

$$P(n,r) = n.(n-1)(n-2)\ldots(n-r+1) = \frac{n!}{(n-r)!} \quad (r \leq n)$$

Proof:

The no. of permutation of a set of n object taken r at a time is equivalent to the no. of ways in which r position can be filled by those n object. So there are n choices to fill up the first position. Now, there are only n-1 objects left to fill up the second position i.e. there are n-1 choices to fill up 2nd position, n-2 choices to fill up 3rd position and so on. At last, to fill the rth position there are n-(r-1) choices. Thus by product rule,

$$\begin{aligned} P(n,r) &= n.(n-1).(n-2)\ldots n-(r-1) \\ &= n.(n-1).(n-2)\ldots n-r+1 \\ &= \frac{n.(n-1).(n-2)\ldots(n-(r-1)).(n-r)\ldots 2.1}{(n-r)\ldots 2.1} \\ &= \frac{n!}{(n-r)!} \end{aligned}$$

Hence proved...

Example:

1) How many license plates consisting of 3 different digit can be made out of given integer 3,4,5,6,7.

Sol:

Total no. of integer $(n) = 5$

$r = 3$

We know,

$$\frac{n!}{(n-r)!}$$

$$= \frac{5!}{(5-3)!} = \frac{5!}{2!} = \frac{5 \times 4 \times 3 \times 2!}{2!} = 60$$

2) How many ways are there to select a 1st prize winner, a 2nd prize winner and 3rd prize winner from 100 different people who have entered a context.?

Sol:

$N = 100$

$r = 3$

$P(100, 3) = ?$

Here,

The way to select 1st prize winner is 100, 2nd prize winner is 99 and 3rd prize winner is 98.

So,

By permutation rule:

$P(100,3) = 100 \times 99 \times 98$

$=$

3) How many permutation of letter ABCDEFGH contains the string ABC?

Sol:

Because the letter ABC must occur as a block.
So, total no. of object (n) = 6.
We can find the answer by finding the no. of permutation of 3 objects.

i.e. $P(6,6) = 6!$
$= 720$.

Permutation with repetition:

Theorem:

The number of r permutation of a set of n object with repetition allowed is n^r.

Example:

How many string of length 3 can be formed with the english alphabet?

Sol:

This is the problem of arranging 26 english alphabet in 3 position i.e.

$P(26,3) = n^r$
$= 26^3$
$= 17576$

Teacher's Signature

Permutation with indistinguishable object:

Theorem:

The number of different permutation of n objects taken all at a time where there are p object of one kind, q object of second kind and r object of 3rd kind is

$$\frac{n!}{p! \times q! \times r!}$$

Example:

How many different string can be made by reordering the word "BENZENE"?

Sol:

Total number of alphabet (n) = 7
Number of alphabet of type 'E' = 3
Number of alphabet of type 'N' = 2

So,

Total number of permutation of different string $= \frac{7!}{3! \times 2!}$

$= 420$ #

Theorem:

The number of different permutation in circular order of n object taken all at a time is $(n-1)!$

Combination:

Combination of object means just their collection without regard to order or arrangement. An r combination of elements of a set is an unordered selection of r elements from the set. Thus, r combination is simply a subset of set with r element.

The r combination of a set with n distinct element is denoted by $C(n,r)$ or ${}^nC_r$ or $\binom{n}{r}$ and is called binomial coefficient and is given by the expression

$$C(n,r) = \frac{n!}{r!(n-r)!}$$

For example:

$S = \{a, b, c, d\}$

2-combination are : $\{a,b\}$ $\{a,c\}$ $\{a,d\}$ $\{b,c\}$ $\{b,d\}$ $\{c,d\}$

Theorem:

The number of r combination of a set with n elements where n is non-negative integer and r is an integer with $0 \le r \le n$ is $C(n,r)$

$$C(n,r) = \frac{n!}{r!(n-r)!}$$

Corollary:

Let n and r be non-negative integer with $r \leq n$ then,

$$C(n,r) = C(n,n-r)$$

Example:

1) How many ways are there to select 5 player from a 10 member Tennis team to make a trip to a match at another school?

Sol:

$$C(10,5) = \frac{10!}{5! \times 5!} = 252 \#$$

2) A cricket team is to be formed consisting of two wicket keepers, four bowlers and five batsman from a group of player consisting four wicket-keepers, 8 bowlers and 11 batsman. How many ways are there to form the cricket team?

Sol:

2 wicket keeper are selected from 4 wicket keeper in C(4,2) ways
4 bowlers are selected from 8 bowler in C(8,4) ways.
5 batsman are selected from 11 batsman in C(11,5) ways.

Hence by product rule,
Cricket team can be formed in $C(4,2) * C(8,2) * C(11,5)$

Binomial coefficient:

The number of r combination from the set with n element is denoted by $\binom{n}{r}$ or $C(n,r)$ is called binomial coefficient because these number occurs as a coefficient in the expansion of power of binomial expansion expression such as $(a+b)^n$.

A binomial expression is simply the sum of two terms such as $(x+y)$, $(a+b)$, $(1+2)$, etc.

Binomial Theorem:

Let x and y be two variables and n is a positive integer. Then

$$(x+y)^n = \sum_{J=0}^{n} \binom{n}{J} x^{n-J} y^J$$

$$= \binom{n}{0} x^{n-0} y^0 + \binom{n}{1} x^{n-1} y^1 + \binom{n}{2} x^{n-2} y^2 + \dots + \binom{n}{n} y^n$$

The coefficient $\binom{n}{0}, \binom{n}{1}, \dots, \binom{n}{n}$ in a binomial expression are called binomial coefficients. which are also denoted by $C_0, C_1, C_2, \dots$.

Teacher's Signature

General term:

The $(r+1)^{th}$ term in the expression of $(x+y)^n$ is usually called its general term which is denoted by t_{r+1}.

In the expansion of $(x+y)^n$.

t_1 = 1st term = $C(n,0)x^n$

t_2 = 2nd term = $C(n,1)x^{n-1}y^1$

t_3 = 3rd term = $C(n,2)x^{n-2}y^2$

⋮

t_{r+1} = $(r+1)^{th}$ term = $C(n,r)x^{n-r}y^r$

Thus, the general term in the expansion of $(x+y)^n$ is t_{r+1} which is;

$$\boxed{t_{r+1} = (r+1)^{th} \text{ term} = C(n,r)x^{n-r}y^r}$$

Corollary 1:

Let n be a non-negative integer then,

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

i.e. sum of binomial coefficient is 2^n.

Proof:

We have,

$$(x+y)^n = \binom{n}{0}x^{n-0}y^0 + \binom{n}{1}x^{n-1}y^1 + \ldots + \binom{n}{n}y^n$$

When $x=1$, $y=1$

$$(1+1)^n = \binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{n}$$

or, $$2^n = \sum_{k=0}^{n} \binom{n}{k}$$

Hence proved...

Corollary 2:

Let n be a positive integer, then

$$\sum_{k=0}^{n} (-1)^k \binom{n}{k} = 0$$

i.e. $$\binom{n}{0} - \binom{n}{1} + \binom{n}{2} - \dots\dots (-1)^n \binom{n}{n} = 0$$

Proof:

We have,

$$(x+y)^n = \binom{n}{0} x^n + \binom{n}{1} x^{n-1} y^1 + \binom{n}{2} x^{n-2} y^2 + \dots + \binom{n}{n} y^n$$

When $x=1$ and $y=-1$

$$(1+(-1))^n = \binom{n}{0} - \binom{n}{1} + \binom{n}{2} - \dots\dots (-1)^n \binom{n}{n}$$

$$0 = \sum_{k=0}^{n} (-1)^k \binom{n}{k}$$

Hence proved....

n=0 1
n=1 1 1
n=2 1 2 1
n=3 1 3 3 1
n=4 1 4 6 4 1
n=5 1 5 10 10 5 1

Fig: Pascal triangle.

This corollary implies that

$$\binom{n}{0} + \binom{n}{2} + \binom{n}{4} + \dots = \binom{n}{1} + \binom{n}{3} + \dots$$

Sum of even coefficient = sum of odd coefficient.

Example:

1) Expand $(2a+5)^5$ by binomial theorem.

Sol:

Comparing $(2a+5)^5$ with $(x+y)^n$

$x = 2a$, $y = 5$, $n = 5$

Now by using binomial theorem.

$$(2a+5)^5 = \sum_{r=0}^{n} \binom{n}{r} (2a)^{n-r} 5^r$$

$$= \binom{5}{0}(2a)^5 5^0 + \binom{5}{1}(2a)^{5-1} 5^1 + \binom{5}{2}(2a)^{5-2} 5^2 + \binom{5}{3}(2a)^{5-3} 5^3$$
$$+ \binom{5}{4}(2a)^{5-4} 5^4 + \binom{5}{5}(2a)^{5-5} 5^5$$

2) Find the 7^{th} term of $\left(x + \frac{1}{x}\right)^{10}$.

Sol:

Comparing $\left(x + \frac{1}{x}\right)^{10}$ with $(x+y)^n$ we get,

$x = x$, $y = \frac{1}{x}$, $n = 10$

The general term t_{r+1} of the expansion $\left(x+\frac{1}{x}\right)^{10}$ is:

$$t_{r+1} = C(n,r)\, x^{n-r} y^r$$

or, $$t_{r+1} = C(10,r)\, x^{10-r} \left(\frac{1}{x}\right)^r$$

or, $$t_{r+1} = C(10,r)\, x^{10-r} x^{-r}$$

$$= C(10,r)\, x^{10-r-r}$$

$$= C(10,r)\, x^{10-2r}$$

Now,

$$t_{6+1} = t_7 = C(10,6)\, x^{10-2\times 6}$$

$$= C(10,6)\, x^{-2}$$

$$= \frac{10!}{6!(10-6)!} x^{-2}$$

$$= \frac{10\times 9\times 8\times 7\times 6!}{6!\, 4!} x^{-2}$$

or, $$t_7 = 210 \cdot \left(\frac{1}{x}\right)^2$$

3) Find the coefficient of x^{16} in the expression of $\left(2x^2-\frac{x}{2}\right)^{12}$.

Sol:

Comparing $\left(2x^2-\frac{x}{2}\right)^{12}$ with $(x+y)^n$, we get

$$x = 2x^2,\ y = \left(\frac{-x}{2}\right),\ n = 12$$

The general term of the expansion of $\left(2x^2-\frac{x}{2}\right)^{12}$ is:

$$t_{r+1} = C(12,r)\,(2x^2)^{12-r}\left(\frac{-x}{2}\right)^r$$

Teacher's Signature

$$= C(12,r)\, 2^{12-r} \left(\frac{-1}{2}\right)^r \cdot x^{24-2r} \cdot x^r$$

$$= C(12,r)\, 2^{12-r} \left(\frac{-1}{2}\right)^r \cdot x^{24-r}$$

To find the coefficient of x^{16} we need;

$$24 - r = 16$$

$$r = 8$$

Then coefficient of x^{16} is:

$$C(12,8)\, 2^4 \cdot \left(\frac{-1}{2}\right)^8$$

$$= \frac{12!}{8! \times 4!} \cdot 2^4 \cdot \frac{1}{2^8}$$

$$= \frac{12!}{8! \times 4!} \times \frac{1}{2^4}$$

$$= \frac{495}{2^4}$$

$$= \frac{495}{16}$$

Finding the middle term:

To find the middle term in the expansion of $(x+y)^n$, we have to consider the cases when n is even and when n is odd number.

Case I: When n is even.

The number of term in the expansion of $(x+y)^n$ is $n+1$. So, when n is even there are odd number of term

and hence there is exactly one middle term which is;

$t_{\frac{n}{2}+1}$.

Case II: When n is odd.

When n is odd, the number of term in the expansion of $(x+y)^n$ is $n+1$ which is even. So there are two middle terms which are;

$t_{\frac{n+1}{2}}$ and $t_{\frac{n+1}{2}+1}$

Pascal's triangle:

The geometrical arrangement of binomial coefficient in the expansion of $(x+y)^n$ in a triangular form is called Pascal's triangle. This can be shown as:

n=0			1			
n=1		1		1		
n=2		1	2	1		
n=3	1	3	3	1		
n=4	1	4	6	4	1	
n=5	1	5	10	10	5	1

Teacher's Signature

Example:

Write down the expansion of $(1+y)^6$ using Pascal's triangle.

Sol:

Here, $n=6$ so we draw Pascal's triangle upto 6.

$n=0$				1			
$n=1$			1		1		
$n=2$		1		2		1	
$n=3$	1		3		3		1
$n=4$	1	4		6		4	1
$n=5$	1	5	10		10	5	1
$n=6$	1	6	15	20	15	6	1

$$(1+y)^6 = 1\cdot 1^6 + 6\cdot 1^5 y^1 + 15\cdot 1^4 y^2 + 20\cdot 1^3 y^3 + 15\cdot 1^2 y^4 + 6\cdot 1^1 y^5 + 1\cdot 1^0 y^6$$
$$= 1 + 6y + 15y^2 + 20y^3 + 15y^4 + 6y^5 + y^6.$$

(5 marks)

Recurrence relation:

A recurrence relation for the sequence $\{a_n\}$ is a equation that express a_n in terms of one or more of the previous terms of the sequence namely $a_0, a_1, a_2, \ldots, a_{n-1}$ for all integer n ~~greater than~~ ~~or equal to~~ where n_0 is non-negative integer.

A sequence is called a solution of the recurrence relation if its term satisfy the recurrence relation.

To find the complete sequence, few initial value are needed and these initial value are called initial condition. When recurrence relation with initial condition is given, then we can write down as many terms of sequence as needed.

For example:-

Let recurrence relation $f_n = f_{n-1} + f_{n-2}$ for $n \geq 2$ with initial condition $f_0 = 1$ and $f_1 = 1$

Then,

The term of sequence $\{f_n\} = \{1, 1, 2, 3, 5, \ldots\}$

Example 1:

Given a recurrence relation $a_n = a_{n-1} - a_{n-2}$ with initial condition $a_0 = 3$ and $a_1 = 5$. Find the term a_2 and a_3.

Sol:

Here,

$a_n = a_{n-1} - a_{n-2}$

$a_2 = a_{2-1} - a_{2-2}$

$= a_1 - a_0$

$= 5 - 3$

$= 2$ //

$a_3 = a_{3-1} - a_{3-2}$

$= a_2 - a_1$

$= 2 - 5$

$= -3$ //

Example 2:

Find the recurrence relation of the sequence 2, 5, 11, 23, 47, ...

Sol:

$a_1 = 2$

Teacher's Signature

$a_2 = 5 = 2 \times a_1 + 1$

$a_3 = 11 = 2 \times a_2 + 1$

$a_4 = 23 = 2 \times a_3 + 1$

⋮

$a_n = 2 \cdot a_{n-1} + 1$

Hence, recurrence relation for given sequence is

$a_n = 2\,a_{n-1} + 1$, $n \geq 2$ with initial condition $a_1 = 2$.

Example 3:

Find the recurrence relation of sequence 1, 1, 2, 3, 5, 8, 13.

Soln:

$a_1 = 1$

$a_2 = 1$

$a_3 = 2 = a_1 + a_2$

$a_4 = 3 = a_2 + a_3$

⋮

$a_n = a_{n-2} + a_{n-1}$

Hence, the recurrence relation is:

$a_n = a_{n-2} + a_{n-1}$; $n \geq 3$ with initial condition $a_1 = 1$, $a_2 = 1$.

Solving Recurrence relation:

If a recurrence relation involving sequence $a_0, a_1, a_2, \ldots, a_n$ then solution of such recurrence relation is to find explicit formula for general term a_n.

Example 1:

Solve the recurrence relation,

$a_n = 2a_{n-1}$; $n \geq 1$ and $a_0 = 3$.

Sol:

$a_0 = 3$

$a_1 = 2a_{1-1} = 2a_0$

$a_2 = 2a_{2-1} = 2a_1 = 2 \times 2a_0 = 2^2 a_0$

$a_3 = 2a_{3-1} = 2a_2 = 2 \times 2a_1 = 2 \times 2 \times 2a_0 = 2^3 a_0$

$\vdots$

$a_n = 2^n a_0$

$a_n = 2^n (3)$

Hence, general solution for given recurrence relation is

$a_n = 2^n \cdot 3$.

Example 2:

Solve the recurrence relation $a_n = a_{n-1} + 2$ with initial condition $a_1 = 3$. (iterative expansion)

Sol:

$a_n = a_{n-1} + 2$

$a_n = (a_{n-2} + 2) + 2$

$a_n = (a_{n-3} + 2) + 4$

$a_n = a_1 + 2(n+1)$

Example 3:

Determine whether the sequence $\{a_n\}$ where $a_n = 3n$ for every non-negative integer n is the solution of the recurrence relation $a_n = 2a_{n-1} - a_{n-2}$ for $n \geq 2$

Sol^n:

Given recurrence relation;

$a_n = 2a_{n-1} - a_{n-2}$

$3n = 2[3(n-1)] - 3(n-2)$

$3n = 2(3n-3) - 3n + 6$

$3n = 6n - 6 - 3n + 6$

$3n = 3n$

Hence, $a_n = 3n$ is the solution of given recurrence relation.

Modeling problem with recurrence relation:

We can use the recurrence relation to model the wide variety of problems such as finding compound interest, determining the no. of noops/steps in Tower of Hanoi puzzle, etc.

1) Compound interest modeling by recurrence relation:

Example:-

Suppose that a person deposit Rs. 10,000 in a saving account at a bank yielding 11% per year interest compounded annually. How much will be in the account after 30 year?

Sol:

Let, P_n denotes amount in account after n year.

Now, recurrence relation for the sequence $\{P_n\}$.

$$P_n = P_{n-1} + 0.11 \cdot P_{n-1}$$
$$= 1.11\, P_{n-1}$$

(∵ since, after n year amount equals to amount after n-1 year plus interest for n^{th} year.

Here, the initial condition;

$P_0 = 10,000$

Solving the recurrence relation;

$P_0 = 10,000$

$P_1 = P_0 + (0.11)P_0$

$= 1.11\,(P_0)$

$P_2 = 1.11(P_1) = 1.11 * 1.11(P_0) = (1.11)^2 P_0$

⋮

$P_n = (1.11)^n P_0$

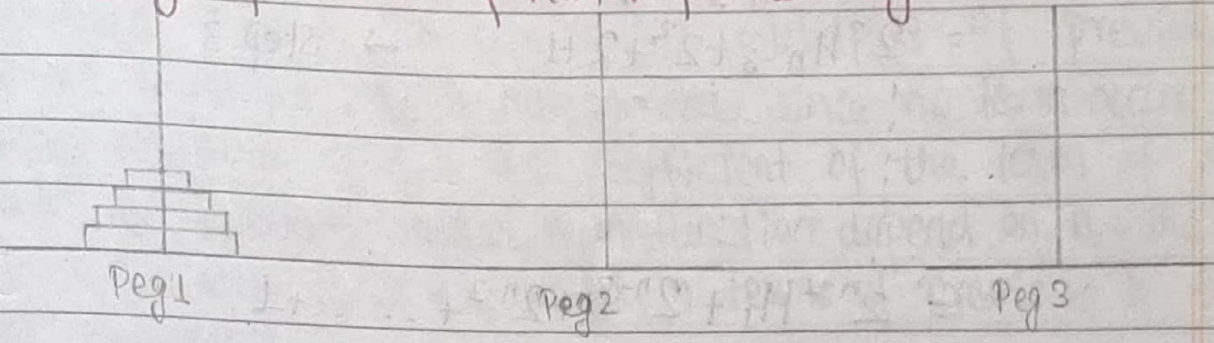
Now, Putting $n = 30$;

$P_{30} = (1.11)^{30} \times 10,000$

$= 2,28,922.9675$.

Teacher's Signature

2) Modeling of Tower of Hanoi problem by recurrence relation:



Let H_n denotes no. of moves needed to solve Tower of Hanoi problem with n-disk

- To set up the recurrence relation for the sequence $\{H_n\}$, we can transfer top $n-1$ disk to peg 3 using H_{n-1} moves.
- Then we can transfer largest disk to the peg 2 using one moves.
- Then we can transfer the $n-1$ disk on peg 3 to peg 2 using H_{n-1} additional moves. These shows that recurrence relation for Tower of Hanoi problem is

$$H_n = H_{n-1} + 1 + H_{n-1}$$
$$= 2H_{n-1} + 1$$

And,

Initial condition for tower of Hanoi problem is $H_1 = 1$.

Now,

Solving this recurrence relation;

$$H_n = 2H_{n-1} + 1 \quad \text{step 1}$$
$$= 2 \cdot (2H_{n-1-1} + 1) + 1$$
$$= 2(2H_{n-2} + 1) + 1$$
$$= 2^2 H_{n-2} + 2 + 1 \quad \text{step 2}$$

$= 2^3(2H_{n-3}+1)+2+1$

$= 2^3 H_{n-3} + 2^2 + 2 + 1$ → Step 3

⋮

$= 2^{n-1} H_1 + 2^{n-2} + 2^{n-3} + \ldots + 1$

$= 2^{n-1} + 2^{n-2} + 2^{n-3} + \ldots + 2^0$

$H_n = 2^n - 1$ [Since, $\sum_{k=0}^{n} x^k = \frac{x^{n+1}-1}{x-1}$]

Solving recurrence relation:

There are different types of recurrence relation. There is no specific technique to solve all the recurrence relation. However, here we solve the recurrence relation with some particular form by using the systematic method.

1. Linear homogeneous recurrence relation with constant coefficient:

A linear homogeneous recurrence relation of degree k with constant coefficient is the recurrence relation of the form

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k}$$

where; $c_1, c_2, \ldots, c_k$ are real numbers and $c_k \neq 0$.

The above recurrence relation is linear since righthand side is a sum of the multiples of previous term of the sequence. It is homogeneous since no term occurs without being multiple of a_j. The coefficient of the term of sequence are all constant rather than function depend on n. The degree is k because it is expressed in terms of previous k term of sequence.

Example:

$a_n = a_{n-1} + 2a_{n-1}^2$ (it is not linear)

$H_n = 2H_{n-1} + 1$ (it is not homogeneous)

$P_n = n \cdot P_{n-1}$ (it is not linear homogeneous recurrence relation with constant coefficient)

$P_n = (0.11) P_{n-1}$ (is a LHRRCC of degree 1)

$f_n = f_{n-1} + f_{n-2}$ (is a LHRRCC of degree 2)

Solving linear homogeneous recurrence relation with constant coefficient:

The basic approach for solving linear homogeneous recurrance relation is to look for the solution of the form $a_n = r^n$ where r is a constant.

Teacher's Signature

Let $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k}$ —— (I)

$r^n = c_1 r^{n-1} + c_2 r^{n-2} + \ldots + c_k r^{n-k}$ —— (II)

When both sides of equation (II) is divided by r^{n-k}.

$r^k = c_1 r^{k-1} + c_2 r^{k-2} + \ldots + c_k$

$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \ldots - c_k = 0$ —— (III)

This eqn (III) is called characteristic equation of the recurrance relation and solution of this eqn are called characteristic root of the recurrance relation.