Module 4

CHAPTER 4

Knowledge and Reasoning

Syllabus

Knowledge based Agents, Brief Overview of propositional logic, First Order Logic: Syntax and Semantic, Inference in FOL, Forward chaining, backward Chaining. Knowledge Engineering in First-Order Logic, Unification, Resolution

Uncertain Knowledge and Reasoning: Uncertainty, Representing knowledge in an uncertain domain, The semantics of belief network, Simple Inference in belief network

Scanned with CamScanner

Scanned with CamScanner

Module 4

Scanned with CamScanner

4.1 KNOWLEDGE AND REASONING

- Search-based problem-solving programs require some knowledge to be implemented. Knowledge can be a particular **state or path** toward solution, rules, etc.
- In order to use this knowledge, it must be represented in a particular way with a certain format. Knowledge Representation (KR) is an important issue in computer science, in general and in AI in particular. "The **dominant paradigm** for building intelligent systems since the early 1970s has been based on the premise that intelligence presupposes knowledge".
- Generally, knowledge is represented in the **system's knowledge base**, which consists of **data structures** and **programs**.

4.1.1 Knowledge Progression

GQ. What is knowledge ?

- **Definition :** Knowledge is a progression that starts with data which is of limited utility. By organizing or analyzing the data, we understand what the data means, and this becomes **information.**

The interpretation or evaluation of information yields knowledge. An understanding of the principles embodied within the knowledge is **wisdom.**

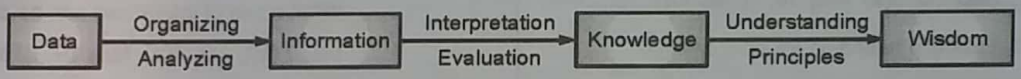


Fig. 4.1.1 : Knowledge Progression

1. **Data:** is viewed as collection of disconnected facts.
   Example: It is raining.
2. **Information** emerges when relationships among facts are established and understood; Provides answers to **"who", "what", "where"**, and **"when"**.
   Example: The temperature dropped 15 degrees and then it started raining.
3. **Knowledge** emerges when relationships among patterns are identified and understood; provides answers as **"how"**.
   Example: If the humidity is very high and the temperature drops substantially, then an atmosphere is unlikely to hold the moisture, so it rains.
4. **Wisdom:** is the pinnacle of understanding, uncovers the principles of relationships that describe patterns. Provides answers as **"why"**.
   Example: Encompasses understanding of all the interactions that happen between raining, evaporation, air currents, temperature gradients and changes.

4.1.2 Types of Knowledge

1. Procedural Knowledge
2. Declarative knowledge

1. Procedural knowledge : is a **compiled knowledge** related to the performance of some task. For example, the steps used to solve an algebraic equation are expressed as procedural knowledge.

2. Declarative knowledge : on the other hand, is passive knowledge expressed as statements of facts about the world. Personnel data in a datable is typical of declarative knowledge.

Heuristic knowledge

There is one more special type of knowledge frequently used by humans to solve complex problems, it is the heuristic knowledge. Heuristics are the knowledge used to make good judgments or strategies, tricks or rules of thumb used to simplify the solution of problems.

For example : In locating a fault in a TV set, an experienced technician will not start by making numerous but instead will immediately reason that the high voltage fly back transformer or related component is the culprit and then it leads to a quick solution.

4.1.3 Knowledge Agent

GQ. What is knowledge agent ?

In artificial intelligence, a knowledge agent is **autonomous entity**, which observes through sensors and acts upon an environment using actuators (i.e. it is an agent) and direct its activities towards achieving their goals. It may also learn or use knowledge to achieve their goals.

Scanned with CamScanner

4.1.4 Levels of Knowledge Representation

GQ. Write a note on "knowledge representation". Or what are the different levels of knowledge representations. **Or** What are the methods of knowledge representation, name them.

- Knowledge consists of facts, concepts, rules, and so on. It can be represented in different forms, as mental images in one's thoughts, as spoken or written words in some language, as graphical or other pictures, and as character strings or collections of magnetic spots stored in a computer.

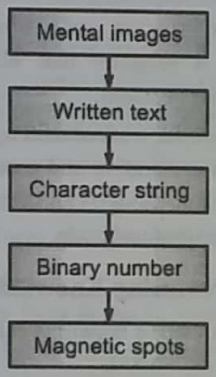


Fig. 4.1.2 : Different levels of knowledge representation

(1) Any choice of representation will depend on the type of problem to be solved and make use of inference methods available. For example, suppose we wish to write a program to play a simple card game using the standard deck of **52** playing cards.

(2) We will need some way to represent the cards dealt to each player and a way to express the rules. We can represent cards in different ways.

(3) The most straightforward way is to record the suit (**clubs, diamonds, hearts, spades**) and face values (ace, 2, 3,....., 10, jack, queen, king) as a symbolic pair. So the queen of hearts might be represented as **<queen, hearts>** Alternatively, we could assign abbreviated codes (c6 for the 6 of clubs), numeric values which ignore suit (1, 2, ..., 13), or some other scheme.

(4) Consider the problem of discovering a pattern in the sequence of numbers **1 1 2 3 4 7**. A change of base in the number from 10 to 2 transforms the number to 01 **101 101 101 101 101 1**.

4.1.5 Various Levels of Knowledge-based Agent

Knowledge-based agent can be described at three different levels : These are :

1. Knowledge level;
2. Logical level;
3. Implementation level.

4.1.6 Knowledge Level

(a) Knowledge-based agents are those agents who have the capability of maintaining an internal state of knowledge, reason over that knowledge, update their knowledge after observations and take actions. These agents can represent the world with some formal representation and act intelligently.

(b) Knowledge-based agents are composed of two main parts :

(I) Knowledge-base and

(II) Inference system.

A knowledge-based agent must be able to do the following :

(i) An agent should be able to do represent states, actions, etc.

(ii) An agent should be able to incorporate new percepts.

(iii) An agent can update the internal representation of the world.

(iv) An agent can deduce the internal representation of the world.

(v) An agent can deduce appropriate actions.

The architecture of knowledge-based agent is shown in Fig. 4.1.3.

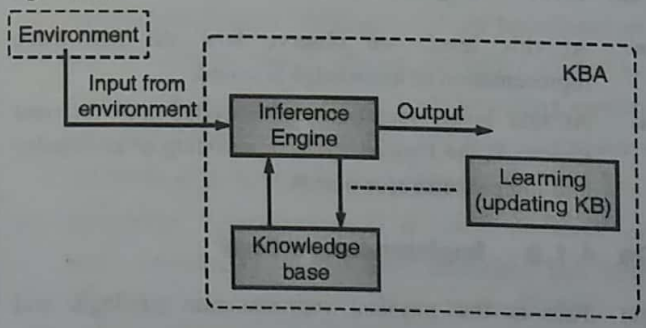


Fig. 4.1.3 : Architecture of knowledge-based agent

The Fig. 4.1.3 is representing a generalised architecture for a knowledge-based agent (KBA).

Module 4

Scanned with CamScanner

KBA takes input from the environment by perceiving the environment.

The input is taken by the inference engine of the agent and it communicates with KB. The learning element of KBA regularly updates the KB by learning new knowledge.

☞ **(I) Knowledge-base**

Knowledge-base is required for updating knowledge for an agent to learn with experiences and take action as per the knowledge.

☞ **(II) Inference system**

- Inference means deriving new sentences from old.
- Inference system allows us to add a new sentence to the knowledge base.
- Inference system applies logical rules to KB to deduce new information.
- Inference system generates new facts so that an agent can update KB.
- An inference system works mainly in two rules which are given as :
  (i) Forward chaining (ii) Backward chaining

☞ **Operations performed by KBA**

Following are **three operations** which are performed by KBA in order to show the intelligent behaviour :

1. **Tell :** This operation tells the knowledge base what it perceives from the environment.
2. **Ask :** This operation asks the knowledge base what action it should perform.
3. **Perform :** It performs the selected action.

4.1.7 Logical Level

- At this level, we observe how the knowledge representation of knowledge is stored.
- At this level, sentences are encoded into different logics. At the logical level, an encoding of knowledge into logical sentences occurs.

4.1.8 Implementation Level

(1) This is the physical representation of logic and knowledge. At the implementation level agent performs actions as per logical and knowledge level.

(2) At this level, an automated taxi agent actually implement his knowledge and logic so that he can reach to the destination.

Approaches to designing a knowledge-based agent are as follows

☞ **(I) Declarative approach**

(1) We can create a knowledge-based agent by initializing with an empty knowledge base and telling the agent all the sentences with which we want to start with.

(2) This approach is called Declarative approach.

☞ **(II) Procedural approach**

(1) In the procedural approach, we directly encode desired behaviour as a program code which means we just need to write a program that already encodes the desired behaviour or agent.

(2) But in the real world, a successful agent can be built by combining both declarative and procedural approaches, and declarative knowledge can often be compiled into more efficient procedural code.

4.1.9 Representing Knowledge Methods

1. Frames and associative networks (also called semantic and conceptual networks),
2. Fuzzy logic,
3. Model logics and
4. Object- oriented methods.

Clearly a representation in the proper base greatly simplifies finding the pattern solution. A typical statement in this logic might express the family relationship of fatherhood as **FATHER (john, Jim)** where the predicate father is used to express the fact that John is the father of Jim.

Other representation schemes include frames and associative networks (also called **semantic** and **conceptual networks**), **fuzzy logic**, **model logics** and **object- oriented methods**.

(1) **Frames** are flexible structures that permit the grouping closely related knowledge. For example, an object such as a ball and its properties (**size color, function**) and its relationship to other objects, (to the left of on top of, and so on) are grouped together into a single structure for easy access.

(2) **Networks** also permit easy access to groups of related items. They associate objects and linkages TO show their relationship to other objects.

Scanned with CamScanner

(3) **Fuzzy logic** is a generalization of predicate logic developed to permit varying degrees of some property such as tall. In classical two-valued logic, TALL (john) is either true of false, but in **fuzzy logic** this statement may be partially true.

(4) **Modal logic** is an extension of classical logic. It was also developed to better represent common sense reasoning by permitting condition such as **likely or possible.**

(5) **Object oriented representations** package an object together with its attributes and functions, therefore chiding these facts. Operations are performed by sending messages between the objects.

4.1.10 Acquisition of Knowledge

GQ. Write a short note on "knowledge acquisition".

(1) Decisions and actions in knowledge-based systems come from acquisition of the knowledge in specified ways. Some form of input will initiate a search for a goal or decision.

(2) For this known facts in the knowledge base be located, compared, and if necessary altered in some way. This process may set up other subgoals and require further inputs, till a final solution is found. The acquisitions are the computational equivalent of reasoning. This requires a form of inference or deduction, using the knowledge and inferring rules.

(3) All forms of reasoning require a certain amount of searching and matching. These two operations require a lot of computation time in AI systems. In a way, it gives a set-back to the acquisition of knowledge.

(4) One of the greatest bottlenecks in building knowledge-rich systems is the acquisition and validation of the knowledge. Knowledge can come from various sources, such as **experts, textbooks, reports, technical articles**, and the like. Reading research articles, taking college courses, consulting with expert colleagues, and using clinical databases are examples of knowledge acquisition.

(5) Each of these activities provides a way to acquire new knowledge through reading, observation and engaging in life-long learning activities.

(6) To be useful, the knowledge must be accurate, presented at the right level for encoding, in complete sense that all essential facts and rules are included, free of inconsistencies, and so on.

(7) Eliciting facts, heuristics, procedures, and rules from an expert is a slow, and tedious process. Experience in building dozens of expert systems and other knowledge- based systems over the past fifteen years has shown this to be the single most time-consuming and costly part of the building process.

(8) This has led to the development of some sophisticated acquisition tools, including a variety of intelligent editors; editors which provide much assistance to the knowledge engineers and system users.

(9) The acquisition problem has also stimulated much research in machine learning systems, that is; systems which can learn new knowledge autonomously without the aid of humans.

(10) Since knowledge-based systems depend on large quantities of high quality knowledge. for their success, it is essential that better methods of acquisition, refinement, and validation be developed.

(11) The ultimate goal is to develop techniques that permit systems to learn new knowledge autonomously and continually improve the quality of the knowledge they possess.

Module 4

4.1.11 Significance of Knowledge Representation

GQ. What is the significance of knowledge representation ?

(1) The Oxford English Dictionary defines knowledge as intellectual acquaintance with, or perception of, fact or truth. A representation is a way of describing certain fragments or information so that any reasoning system can easily adopt it for inference purposes.

(2) Knowledge representation is a study of ways of how knowledge is actually picturized and how effectively it resembles the representation of knowledge in human brain.

Scanned with CamScanner

4.1.12 Characteristics Of Knowledge Representation

GQ. What are the characteristics of knowledge representation.

A knowledge representation system should provide ways of representing complex knowledge and should possess the following **characteristics** :

1. The representation scheme should have a set of well-defined syntax and semantics. This will help in representing various kinds of knowledge.
2. The knowledge representation scheme should have a good expressive capacity, a good expressive capability will catalyze the inference mechanism in its reasoning process.
3. From computer system point of view, the representation must be efficient. By this we mean that it should use only limited resources without compromising on the expressive power.

4.1.12(A)Knowledge Representation Schemes

Various Knowledge Representation Schemes are as follows :

(i) Semantic networks
(ii) Frames
(iii) Conceptual dependency
(iv) Scripts

4.1.13 Properties of Knowledge Representation System

The following properties should be possessed by a knowledge representation system :

(1) Representational Adequacy : The ability to represent the required knowledge.

(2) Inferential Adequacy : The ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from the original.

(3) Inferential Efficiency : The ability to direct the inferential mechanisms into the most productive directions by storing appropriate guides.

(4) Acquisitional Efficiency : The ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

4.1.14 Procedural and Declarative Knowledge

(I) Declarative

A declarative representation is one in which knowledge is specified, but the use of that knowledge is not given. A declarative representation, we must augment it with a program that specifies what is to be done to the knowledge and how.

For example : a set of logical assertions can be combined with a resolution theorem prover to give a complete program for solving problems. There is a different way, though, in which logical assertions can be viewed, namely as a program, rather than data to a program. In this view, the implication statements define the legitimate reasoning paths and the atomic assertions provide the starting points (or, if we reason backward, the ending points) of those paths.

(II) Procedural

- A procedural representation is one in which the control information that is necessary to use the knowledge is considered to be embedded in the knowledge itself. To use procedural representation, we need to augment it with an interpreter that follows the instructions given in the knowledge.
- Screening logical assertions as code is not a very essential idea, given that a programs are really data to other programs that interpret (or compile) and execute them. The real difference between the declarative and the procedural views of knowledge lies in where control information resides. For example, consider the knowledge base :
- man (Marcus) ; man (Caesar) ; person(Cleopatra)

  $\forall a$: man $(a) \rightarrow$ person(a)

  Now consider trying to extract from this knowledge base the answer to the question $\exists$: person(z)
- We want to bind y to a particular value for which person is true. Our knowledge base justifies any of the following answers :

  z = Marcus , z = Casear, z = Cleopatra
- For the reason that there is more than one value that satisfies the predicate, but only one value is needed, the answer to the question will depend on the order in which the assertions are examined during the search for response.

Scanned with CamScanner

4.1.15 Difference between Procedural and Declarative Knowledge

Table 4.1.1

Sr. No.	Procedural Knowledge	Declarative Knowledge
1.	Follow black box.	Follow the white box.
2.	Based on process orientation.	Based on data orientation.
3.	Possible to faster usage.	Based on knowledge in the process of system design.
4.	When we have to achieve in a particular result.	According knowledge format that may be manipulated and analyzed.
5.	Knowing how to do something.	It is knowledge about something.
6.	Simple data type can be used.	Large data type can be used.
7.	Followed in C++ and Cobol.	Followed by SQL.
8.	According and programming any simple task.	According SQL any simple task in one line code i.e. one line SQL statement.
9.	Programmer should understand execution.	Programmer should not interact with SQL.
10.	Initially faster but later it can be slow.	Initially slower but possibly faster later .
11.	Work on interpreter of language.	Work on data engine with the DBMS.
12.	Example : If Manmohan or Monu is older.	Example : Manmohan is older than monu.

4.2 PROPOSITIONAL LOGIC (FIRST ORDER LOGIC)

4.2.1 Introduction to Logic

Logic is the discipline that deals with the **methods of reasoning**. On an elementary level, logic provides rules and techniques for determining whether a given argument is valid.

Commonsense logic

It is deriving conclusions from personal experience or knowledge. A conclusion that something makes sense, so it is right or something doesn't make sense, so it is wrong.

Let us consider one classic example: A bear walked one k.m. due south, then turned to the left and walked one k.m. due East. Then it turned to the left again and walked one k.m. due North and arrived back at its starting point.

What was the colour of the bear? Now, actually the bear walked **3 sides** of a **square,** Like this

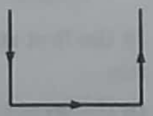


Fig. 4.2.1

But since it ended up where it started from, its actual path must have been a triangle :

Fig. 4.2.2

The only two places in the world where this could happen is either the **North-pole or South-pole**. The south-pole is not possible, since it is impossible to travel south from south (pole). So the bear was at North pole, i.e. it was a polar bear. So the bear was **white.**

This problem is solved by knowledge, i.e., it requires a logical type of mind to apply that knowledge to a particular problem. It can also be called as **inductive logic,** which does not permit arguments **outside** the facts available.

Logical reasoning

It is used in mathematics to prove theorems; in computer science to verify the correctness of programs and to prove theorems. And in our everyday lives to solve a multitude of problems.

Module 4

Scanned with CamScanner

4.2.2 Logic Language

- One of the basic difficulties in developing an approach to logic is the limitation of ordinary language when it comes to presenting statements and conclusions. [Exactly the same problem arises with computers. You cannot instruct computers in ordinary language-it has to be consistent with the input/output capabilities of the computer].
- Our aim is now very simple- to give each statement an exact meaning and manipulate such statement in a logical manner, determined by the rules and theorems. Here, we discuss a few of the basic ideas; i.e. rules and theorem.

4.2.3 Propositions and Logical Operations

❑ **Definition :** A statement or proposition is a declarative sentence that is either true or false, **but not both.**

Illustrative Ex. 4.2.1 :

(i) The earth is round

(ii) $3 + 4 = 7$

(iii) $4 + x = 9$

(iv) Do you speak Gujarathi ?

(v) Take two aspirins.

(vi) The temperature on the surface of the planet mars is 500°F.

(vii) The sun will come out tomorrow

☑ **Soln. :**

(i) and (ii) are statements which are true.

(iii) it is not a statement, since it is true or false depends on the value of x. But we can say that it is a declarative sentence.

If we put $x = 5$, it becomes a true statement, if we take value of $x \neq 5$, it becomes a false. Such statements are open statements.

Thus if a mathematical statement is **neither true nor false**, it is called **open statement.**

(iv) It is a question, not a statement.

(v) It is a command, but not a statement

(vi) It is a statement, because in principle we can determine if it is true or false.

(vii) It is a statement, since it is true or false but not both.

4.2.4 Compound Propositions

Propositions composed of **sub propositions** are called **compound propositions.** A proposition is said to be **primitive** if it cannot be broken down into simpler propositions; that is, if it is not composite.

Compound propositions or statements are composed of various **logical connectives**.

☞ **Examples of compound propositions**

(i) 'Roses are red and violets are blue' is a compound statement with sub statements: " Roses are red" and "violets are blue".

(ii) "John is intelligent and studies every night" is a compound statement with sub propositions :

"John is intelligent" and" John studies every night.

4.3 BASIC OPERATIONS

(I) Conjunction ($p \wedge q$)

(II) Disjunction ($p \vee q$)

(III) Negation ($\neg$)

(I) $p \wedge q$ conjunction of p and q, read "p and q"

Two propositions p and q can be combined by the word **'and'** to form a compound proposition and called as **conjunction** of the original propositions.

Symbolically, $p \wedge q$ is a compound proposition. And it has a TRUE value.

❑ **Definition :** If p and q are true, then $p \wedge q$ is also true, otherwise $p \wedge q$ is false.

We prepare the table for the truth-value of $p \wedge q$.

➡ **Note :** T stands for true and F stands for false.

Table 4.3.1 : Truth table for Conjunction

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

(i) **In the first row**; if p is true and q is true then $p \wedge q$ is true.

(ii) **In the second row** : if p is true and q is false then $p \wedge q$ is false. And so on.

Scanned with CamScanner

Remark : $p \wedge q$ is true **only when** p and q both are true.

☞ **Examples based on conjunction**

Form the conjunction of p and q for the following

(i) $p : 3 < 5$ and $q : -2 > -4$

$\therefore p \wedge q : 3 < 5$ and $-2 > -4$

(ii) p : it is hot, q : $2 < 4$

$\therefore p \wedge q$ it is hot and $2 < 4$.

(iii) p : it is snowing, q : I am cold

$\therefore p \wedge q$: it is snowing and I am cold.

☞ **Remarks on the above examples**

Example : (ii) Shows that we may join two totally unrelated statements by the connective 'and ($\wedge$)'

(II) Disjunction

(i) If p and q are statements, the **disjunction** of p and q is the compound statement "**p or q**" denoted by $p \vee q$.

(ii) The connective **'or'** is denoted by the symbol $\vee$. The compound statement $p \vee q$ is true if **at least one of p or q is true**; it is **false** when both **p and q are false**.

The truth table of $p \vee q$:

Table 4.3.2 : Truth table of $p \vee q$

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

(iii) Negation : ($\neg$)

If p be any statement then negation of p is denoted by $\neg p$ and is read as 'not p'.

If p is true then $\neg p$ is false. If p is false then $\neg p$ is true.

☞ **Truth table for Negation**

Table 4.3.3 : Truth table for Negation

p	$\neg p$
T	F
F	T

☞ **Example based on Negation**

If **p** : Gopal is good at sports

then $\neg$ **p** : Gopal is not good at sports.

Remark : Negation is also denoted by $\sim$. Thus if p is true, then $\sim p$ is false

4.4 PROPOSITIONS AND TRUTH-TABLES

Proposition : A proposition is also called a well-formed formula of logical variables **p, q, r,** .... and logical connectives ($\wedge, \vee, \neg$) We denote such a proposition by **P (p, q, r, ...)**.

Truth-table : The truth-value of a proposition depends upon the truth values of its variables. (Thus the truth value of a proposition is known once the truth values of its variables are known). This relationship can be shown through a **truth-table.**

4.4.1 Method of constructing Truth-table of the Proposition

- **Step (i) :** The first columns of the table are for the variables **p, q,**
- **Step (ii) :** Allow the rows for all possible combination of T and F for these variables. (For 2 variables, $2^2 = 4$ rows are necessary; for 3 variables, $2^3 = 8$ rows are necessary, and, in general, for n variables, 2^n rows are required.
- **Step (iii) :** There is a column for each "**elementary**" stage of the constructions for the truth-value of the proposition
- **Step (iv) :** The truth value of each step being determined from the previous stages by the definitions of the connectives $\wedge, \vee, \neg$
- **Step (v) :** Finally, in the last column, we obtain the truth value of the proposition.

Module 4

Scanned with CamScanner

4.4.2 Examples Based on the Proposition

Ex. 4.4.1

(i) Find the truth-table of the proposition $\neg (p \wedge \neg q)$

p	q	¬ q	p ∧ ¬ q	¬ (p ∧ ¬ q)
T	T	F	F	T
T	F	T	T	F
F	T	F	F	T
F	F	T	F	T

(a)

=

p	q	¬ (p ∧ ¬q)
T	T	T
T	F	F
F	T	T
F	F	T

(b)

Remark

The truth table of the preposition consists of the columns under the variables and the column under the proposition, as shown in (b).

Ex. 4.4.2 : Find the truth-table of $\neg p \wedge q$.

☑ **Soln. :** We construct the table :

p	q	¬ p	¬ p ∧ q
T	T	F	F
T	F	F	F
F	T	T	T
F	F	T	F

(i) For two variables p, q we choose the truth-values in the first two columns as shown.

(ii) We find the truth value of ¬ p using the negation ¬.

(iii) We find the truth value of ¬ p ∧ q using the conjunction ∧.

UEx. 4.4.3 (MU - Q. 5(b), May 19, 10 Marks)

1. John likes all kinds of food.
   ∀x : food(x) → likes (john, x)
2. Apples are food.
   food(apple)
3. Chicken is food
   food(chicken)
4. Anything anyone eats are isn't kiled by is food
   ∀x: (∃y:eats(y, x) ∧ ¬ killed by (y, x)) → food(x)
5. Bill eats peanuts and is still alive.
   A. eats (Bill, peanuts) B. alive (Bill)
6. Sue eats everything Bill eats
   ∀x: eats(Bill, x) → eats (Sue, x)
7. ∀x: ∀y :alive(x) → ¬ killedby(x, y)

☑ **Soln. :**

1. John likes all kind of food.
2. Apples and chicken are food.
3. Anything anyone eats is not killed by food.
4. Bill eats peanuts and is still alive.
5. Sue eats everything that Bill eats.

▶ **Step I :** Converting the given statements into FOPL :

1. ∀x : food (x) → likes (John, x)
2. food (apple) ∧ food (chicken)
3. ∀a, ∀b : eats (a, b) ∧ ¬ killed (a) → food (b)
4. eats (Bill, peanuts) ∧ ¬ killed (Bill)
5. ∀y : eats (Bill, y) → eats (Sue, y)

▶ **Step II :** Converting FOPL statements into causal form

1. ¬ food (x) ∨ likes (John, x)
2. food (apple),
3. food (chicken)
4. ¬ eats (a, b) ∨ killed (a) ∨ food (b)
5. eats (Bill, peanuts)
6. ¬ e killed (Bill)
7. eats (Bill, y) ∨ eats (Sue, y)

Conclusion : Likes (John, Peanuts)

▶ **Step III : Resolution performance**

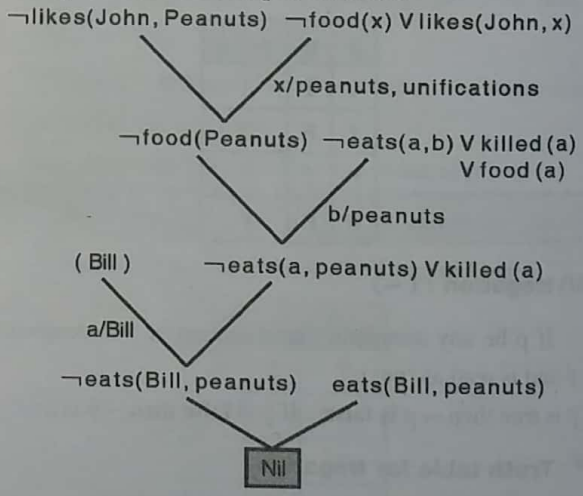


Fig. Ex. 4.4.3

Scanned with CamScanner

4.5 TAUTOLOGIES AND CONTRADICTIONS

- **Definition 1 :** A proposition P (p, q, ...) is a tautology if it contains only T in the last column of its truth table, i.e. if P is true for any truth values of its variables.
- **Definition 2 :** A proposition P (p, q, ...) is a **contradiction** if it contains only 'F' in the last column of its truth table, i.e., it P is false for any truth values of its variables.

For example (i) "p or not p", i.e., $p \vee \sim p$ is tautology

Table 4.5.1 : Truth table of "p or not p is tautology"

P	~ p	p ∨ ~ p
T	F	T
F	T	T

(ii) '$p \wedge \sim p$' is a contradiction By truth table,

'$p \wedge \sim p$' is a contradiction.

Table 4.5.2 : Truth table of "p or not p is contradiction "

P	~ p	p ∧ ~ p
T	F	F
F	T	F

4.5.1 Examples on Tautology

Ex. 4.5.1 : Show that $p \vee \neg (p \wedge q)$ is a tautology.

Soln. :

We prepare the truth-table for $p \vee \neg (p \wedge q)$:

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

(i) We choose the truth-values for p, q in first two columns.

(ii) Then truth value for $p \wedge q$, (conjunction)

(iii) Negation of $(p \wedge q)$

(iv) Truth value of $p \vee \neg (p \wedge q)$; which is a tautology.

4.5.2 Theorems

Theorem (1)

If **P (p, q, ...)** is a tautology, then $\neg$ **P (p, q, ...)** is a contradiction, and conversely.

Proof

Since a tautology is always true, (i.e., it contains only T in the last column); the negation of a tautology is always false, (i.e. it contains only F in the last column)

$\therefore$ $\neg$ **P (p, q, ...)** is a contradiction, and conversely.

Theorem (2) : (Principle of Substitution)

Suppose **P (p, q, ...)** is a tautology, then P (p_1, p_2, ...) is a tautology for any propositions p_1, p_2.

Proof :

Since **P (p, q, ...)** is a tautology. And it does not depend on truth values of its variables p, q, ..., we can substitute P_1, for p, P_2 for q, ... in the given tautology P (p, q, ...) and it becomes a tautology.

Ex. 4.5.2 : Verify that $(p \wedge \neg q) \vee \neg (p \wedge \neg q)$ is a tautology.

Soln. :

Let $P = p \wedge \neg q$, then the proposition becomes $p \vee \neg p$. We have seen that proposition $p \vee \neg p$ is a tautology.

$\therefore$ $(p \wedge \neg q) \vee \neg (p \wedge \neg q)$ is a tautology.

Module 4

4.6 CONDITIONAL CONNECTIVES OR IMPLICATION

In mathematics, we come across statements like "**If p then q**". Such statements are called **conditional statements** and are denoted by $\mathbf{p \rightarrow q}$.

Remark

1. The conditional statement $p \rightarrow q$ is sometimes read as (i) p implies q, (ii) p is **sufficient** for q (iii) q is necessary for p (iv) p only if q.
2. The conditional $p \rightarrow q$ is false when the first **p is true** and the second part **q is false** ; i.e., when **p is false**, the conditional $\mathbf{p \rightarrow q}$ is **true** regardless of the truth **value of q.**
3. (i) If p is true, q is true then $p \rightarrow q$ is true.
   (ii) If p is true, q is false then $p \rightarrow q$ is false.
   (iii) If p is false, q is true, then $p \rightarrow q$ is true.
   (iv) If p is false, q is false, the $p \rightarrow q$ is true.

Scanned with CamScanner

☞ **Truth table for $p \rightarrow q$**

Table 4.6.1 : Truth table for $p \rightarrow q$

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

4. We observe that the truth tables of $\neg p \vee q$ and $p \rightarrow q$ are identical. (check!); i.e., $p \rightarrow q$ is logically equivalent to $\neg p \vee q$.

   i.e., $p \rightarrow q \equiv \neg p \vee q$.

4.6.1 Examples

Ex. 4.6.1 : Determine the truth value of the following statements :

(i) If Bombay is in India, then $4 + 5 = 9$.

(ii) If Bombay is in India, then $4 + 5 = 3$.

☑ **Soln. :**

(i) Let p = Bombay is in India, $q = 4 + 5 = 9$

$\because$ p is true, q is true, $\therefore$ $p \rightarrow q$ is true.

(ii) Let p = Bombay is in India,

$q = 4 + 5 = 3$

$\because$ p is true, but q is false $\therefore$ $p \rightarrow q$ is false.

Ex. 4.6.2 : Rewrite the following statements without using the conditional.

(i) If it is hot, he wears a hat

(ii) If F is field, it is integral domain.

☑ **Soln. :**

(i) Recall that $p \rightarrow q = \sim p \vee q$; i.e.

if 'p then q, is equivalent to 'not p or q'.

$\therefore$ It is not hot or he wears a hat.

(ii) F is not field or F is integral domain.

4.6.2 Conditional Statements and Variations

Now we consider other simple **conditional** propositions containing p and q.

1. Let $\mathbf{p \rightarrow q}$ be conditional proposition. Then **'$\mathbf{q \rightarrow p}$'** is called **converse** conditional proposition.
2. **'$\mathbf{\neg p \rightarrow q}$'** is called **inverse** of the original conditional proposition $\mathbf{p \rightarrow q}$.
3. **'$\mathbf{\neg q \rightarrow \neg p}$'** is called **contrapositive** of the original conditional proposition $\mathbf{p \rightarrow q}$.

☞ **Example of conditional proposition**

Which, if any, of the above propositions are logically equivalent to $p \rightarrow q$.

We construct the truth-table for the above conditional proposition.

Table 4.6.2 : Truth table for conditional proposition

p	q	$\neg p$	$\neg q$	Conditional $p \rightarrow q$	Converse $q \rightarrow p$	Inverse $\neg p \rightarrow \neg q$	Contra positive $\neg q \rightarrow \neg p$
T	T	F	F	T	T	T	T
T	F	F	T	F	T	T	F
F	T	T	F	T	F	F	T
F	F	T	T	T	T	T	T

Only the contra positive '$\neg q \rightarrow \neg p$' is logically equivalent to the original conditional probability $p \rightarrow q$.

4.6.3 Advantage and Disadvantages of Propositional Logic

☞ **Advantages of Propositional Logic**

1. It is used in artificial intelligence for planning, problem-solving, intelligent control and most importantly for decision-making.
2. It is about Boolean functions and the statements where there are more than just true and false values, includes the certainty as well as uncertainty.
3. It led to the foundation of machine learning models.
4. It is a useful tool for reasoning.

☞ **Disadvantages of Propositional Logic**

1. We cannot represent relations like All, some, or none with propositional logic.

   Example : All the boys are intelligent.
2. Propositional logic has limited expressive power.
3. In propositional logic, we cannot describe statements in terms of their properties or logical relationships.
4. Propositional logic is lacking of syntax for representing objects in the domain of internet.

Scanned with CamScanner

4.6.4 Theorem of Contra-Positive of the Statements

Contra-positive is a proposition or theorem formed by contradicting both the subject and predicate or both the hypothesis and conclusion of a given proposition or theorem and interchanging them "if not-B then not-A" is the contra-positive of "if A then B".

To form the contra-positive of the conditional statement, we interchange the hypothesis and the conclusion of the inverse statement. The contra-positive of "if it rains, then they cancel school" is " If they do not cancel school, then it does not rain". If the converse is true, then the inverse is also logically true.

A conditional statement $p \rightarrow q$ and its contra positive $\neg q \rightarrow \neg p$ are logically equivalent.

If the statement is true, then the contra-positive is also logically true. If the converse is true, then the inverse is also logically true.

If two angles are congruent, then they have the same measure.

Converse, Inverse, Contrapositive

Statement	If p, then q
Inverse	If not p, then not q
Contrapositive	If not q, then not p.

The contrapositive of a conditional statement of the form "if p then q" is "If $\neg$ q then $\neg$ p".

Symbolically the contrapositive of p q is $\neg q \neg p$.

(i) **Conditional :** The conditional of q by p is "If p then q" or "p implies q" and is denoted by '$p \rightarrow q$'.

(ii) **Biconditional : (iff) :** The biconditional of p and q is "p, if and only if, q" and is denoted by $p \leftrightarrow q$.

(iii) **Only if :** p only if q means "If not q then not p" or equivalently, "if p then q".

(iv) **Sufficient condition :** p is a sufficient condition for q means "if p then q".

Ex. 4.6.3 : Determine the contra positive of the statements.

(i) If Bhide is a teacher, then he is poor.

(ii) If Thombare studies, he will pass the test.

☑ **Soln. :**

(i) The contra positive of $p \rightarrow q$ is $\neg q \rightarrow \neg p$.

∴ The contra positive of the given statement is

If Bhide is not poor, then he is not a teacher.

(ii) The contra positive is :

If Thombare does not study, then he will not pass the test.

4.6.5 Biconditional : $p \leftrightarrow q$

Another common statement is of the form "p if and only if q". Such statements are called biconditional statements and are denoted by $p \leftrightarrow q$.

☞ **Truth-Table for $p \leftrightarrow q$**

Table 4.6.3 : Truth table for $p \leftrightarrow q$

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

The biconditional $p \leftrightarrow q$ is true whenever p and q have the same truth values and false otherwise.

☞ **Theorem**

The propositions **P (p, q, ...)** and **Q (p, q, ...)** are logically equivalent if and only if the proposition.

P(p, q, ...) ↔ Q (p, q, ...) is a tautology.

Proof

▶ **Step (I) :** Let P (p, q, ...) = Q (p, q, ...). Then they have the same truth table.

∴ P(p, q, ...) ↔ Q (p, q, ...) is true for any values of the variables p, q, ..., It means that the proposition is tautology.

▶ **Step (II) :** Since each step is reversible,

∴ Converse is also true.

Module 4

4.7 ARGUMENTS

❑ **Definition :** An **argument** is an assertion such that for a given set of propositions P_1, P_2, ..., P_n called **premises**, gives rise to another proposition Q (or a consequence Q), called the **conclusion.** And such an argument is denoted by $P_1, P_2, ..., P_n \, Q$

❑ **Definition :** Valid Argument or Logical Argument :

1. An argument $P_1, P_2, ..., P_n$ Q is said to be logical or valid if Q is **true whenever** all the premises $P_1, P_2, ..., P_n$ are **true.**
2. An argument which is not valid is called a **fallacy**.

Scanned with CamScanner

Ex. 4.7.1 : Show that the following argument is valid

$p, p \rightarrow q \vdash q$

(This is called as Law of detachment).

☑ **Soln. :** We prepare the truth-table for $p \rightarrow q$

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

(i) p is true in 1 and 2 lines; $p \rightarrow q$ is true in 1, 3, 4 lines.

(ii) p and $p \rightarrow q$ are true in line 1.

Since in this case q is also true.

∴ The argument is valid.

☞ **Remark**

For the argument to be valid, it is not necessary **that 'p', 'p → q' and 'q' are true in all the rows.**

Ex. 4.7.2 : Show that the following argument is a fallacy :

$p \rightarrow q, q \vdash p$

☑ **Soln. :**

We observe that $p \rightarrow q$ and q are true in case (3) in the **above** table, but in this case p is false.

Hence the argument is fallacy.

4.7.1 Theorem on Tautology

The argument $P_1, P_2, ..., P_n \vdash Q$ is valid if and only if the proposition $(P_1 \wedge P_2 \wedge .... \wedge P_n) \rightarrow Q$ is a tautology.

Proof

We recall that $P_1, P_2, ...,P_n$ are true if and only if $P_1 \wedge P_2 \wedge .... \wedge P_n$ is true. .... (i)

Thus the argument $P_1, P_2, ...P_n \vdash Q$ is valid if Q is true whenever $P_1, P_2, ... P_n$ is true or equivalently $P_1 \wedge P_2 \wedge ... \wedge P_n$ is true.

That is the proposition $(P_1 \wedge P_2 \wedge ... \wedge P_n) \rightarrow Q$ is a tautology.

4.7.2 Fundamental Principle of Logical Reasoning

The principle states that :

"If p implies q and q implies r, then p implies r."

that is if the following argument is valid :

$p \rightarrow q, q \rightarrow r \vdash p \rightarrow r$. (Law of syllogism).

4.7.3 Verification of Law of Syllogism

Ex. 4.7.3 : Show that the following argument is a fallacy :

$p \rightarrow q, \neg p \vdash \neg q$.

☑ **Soln. :**

We construct the truth table.

(i) $(p \rightarrow q) \wedge (\neg p)$; and $\neg q$ and then

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

and check whether it is a tautology or not.

☑ **Soln. :**

p	q	$p \rightarrow q$	$\neg p$	$(p \rightarrow q) \wedge (\neg p)$	$\neg q$	$[(p \rightarrow q) \wedge \neg p] \rightarrow \neg q$
T	T	T	F	F	F	T
T	F	F	F	F	T	T
F	T	T	T	T	F	F
F	F	T	T	T	T	T

Since the proposition $[(p \rightarrow q) \wedge \neg p] \rightarrow \neg q$ is not a tautology (last column), hence it is a fallacy.

Also observe that in line (3) of the truth table $p \rightarrow q$ and $\neg p$ are true but $\neg q$ is false.

Ex. 4.7.4 : Prove that the argument is valid.

$P \leftrightarrow q, q \vdash p$.

☑ **Soln. :**

We can establish the validity of the argument in two different methods :

Method 1

We construct truth-table for $p \leftrightarrow q$,

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

(i) Now, $p \leftrightarrow q$ is true in lines 1 and 4, and q in lines1 and 3.

∴ $p \leftrightarrow q$ and q are both true in line 1, where p is also true.

∴ The argument $p \leftrightarrow q, q \vdash p$ is valid

Scanned with CamScanner

Method 2

Now, we construct the truth-table of $[(p \leftrightarrow q) \wedge q] \rightarrow p$ as follows :

p	q	$p \leftrightarrow q$	$(p \leftrightarrow q) \wedge q$	$[(p \leftrightarrow q) \wedge q] \rightarrow p$
T	T	T	T	T
T	F	F	F	T
F	T	F	F	T
F	F	T	F	T

Since $[(p \leftrightarrow q) \wedge q] \rightarrow p$ is a tautology, the argument is valid.

Ex. 4.7.5 : Determine the validity of the argument

$p \rightarrow q, \neg q \vdash \neg p$.

☑ **Soln.:**

We construct the truth table for

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

p	q	$p \rightarrow q$	$\neg q$	$(p \rightarrow q) \wedge (\neg q)]$	$\neg p$	$[(p \rightarrow q) \wedge (\neg q)] \rightarrow \neg p$
T	T	T	F	F	F	T
T	F	F	T	F	F	T
F	T	T	F	F	T	T
F	F	T	T	T	T	T

Since the proposition $[(p \rightarrow q) \wedge \neg q] \rightarrow \neg p$ is a tautology, the given argument is valid.

Ex. 4.7.6 : Show that $(p \wedge q) \rightarrow (p \vee q)$ is a tautology.

☑ **Soln. :**

We prepare truth table :

Since the truth value of $(p \wedge q) \rightarrow (p \vee q)$ is T for all values of p and q, the proposition is a tautology.

p	q	$p \wedge q$	$p \vee q$	$p \wedge q \rightarrow p \vee q$
T	T	T	T	T
T	F	F	T	T
F	T	F	T	T
F	F	F	F	T

Ex. 4.7.7 : State the converse of each of the following implications:

(i) If $4 + 4 = 8$, then I am not the PM of India.

(ii) If I am late, then I did not take the train to work.

(iii) If I have enough money, then I will buy a car and I will buy a home.

☑ **Soln. :**

(i) If I am not the PM of India, then $4 + 4 = 8$.

(ii) If I do not take the train to work, then I am late.

(iii) If I buy a car and I buy a house, then I have enough money.

Ex. 4.7.8 : Determine the truth value for each of the following statements :

(i) If 8 is even, then Bombay has a large population.

(ii) If 8 is even, then Bombay has a small population.

(iii) If 8 is odd, then Bombay has a large population.

(iv) if 8 is odd, then Bombay has a small population.

☑ **Soln. :**

Let p = 8 be even, q = Bombay has large population.

We prepare the truth-value table.

Sr. No.	p	q	$p \rightarrow q$
(i)	T	T	T
(ii)	T	F	F
(iii)	F	T	T
(iv)	F	F	T

Ex. 4.7.9 : Construct truth tables to determine whether the given statement is a tautology, a contingency or an absurdity :

(i) $p \rightarrow (q \rightarrow p)$ (ii) $q \rightarrow (q \rightarrow p)$.

☑ **Soln. :**

(i) We prepare the truth-table :

p	q	$q \rightarrow p$	$p \rightarrow (q \rightarrow p)$
T	T	T	T
T	F	T	T
F	T	F	T
F	F	T	T

Since the truth-value of $p \rightarrow (q \rightarrow p)$ is T for all values of p and q, the proposition is tautology.

(ii) The truth-table is :

p	q	$q \rightarrow p$	$q \rightarrow (q \rightarrow p)$
T	T	T	T
T	F	T	T
F	T	F	F
F	F	T	T

Since the truth-value of $q \rightarrow (q \rightarrow p)$ is not T for all values of p and q, the proposition is **contingency**.

Module 4

Scanned with CamScanner

4.8 PRECEDENCE RULE

- We can always use a truth table to show that an **argument-form** is valid. We do this by showing that whenever premises are true, the conclusion must also be true.
- However, this can be a tedious approach. For example, when an argument form involves 10 different propositional variables, it requires $2^{10} = 1024$ different rows to show the argument form is valid, or not.
- Fortunately, we do not have to resort to truth tables. Instead we can first establish, Rule of precedence, i.e., the order of preference in which the connectives are applied in a formula of propositions that has no brackets is :

(i) $\neg$ (ii) $\wedge$ (iii) $\vee$ and $\neg$ (iv) $\rightarrow$ and $\leftrightarrow$

4.9 DUALITY LAW

In this section, we shall consider formulae which contain the connectives $\wedge$, $\vee$, $\neg$. Well shall see later that, any formula containing any other connective can be replaced by an equivalent formula containing only these three connectives.

❑ **Definition :** Two formulae, A and A*, are said to be duals of each other if either one can be obtained from the other by replacing $\wedge$ by $\vee$ and $\vee$ by $\wedge$.

The connectives $\wedge$ and $\vee$ are called duals of each other. If the formula A contains the special variable T or F, then A*, its dual, is obtained by replacing T by F and F by T in addition to the above-mentioned interchanges.

4.9.1 Illustrative Example based on Duals

Ex. 4.9.1 : Write the duals of :

(i) $(p \vee q) \wedge r$; (ii) $(p \wedge q) \vee r)$

(iii) $(p \vee T)$, (iv) $\neg (p \vee q) \wedge (p \vee \neg (q \wedge \neg r)$

☑ **Soln. :** Duals are

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

(iii) $(p \wedge F)$, (iv) $\neg (p \wedge q) \vee (p \wedge \neg (q \vee \neg r))$

4.9.2 Logical Identities

1. De Morgan's Laws

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

Dual is : (ii) $\neg (p \wedge q) = (\neg p \vee \neg q)$

2. Associative Laws

(i) $p \vee (q \vee r) \equiv (p \vee q) \vee r$

Dual is : (ii) $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$

3. Commutative Laws

(i) $p \vee q \equiv q \vee p$; **Dual is :** $p \wedge q \equiv q \wedge p$

4. Idempotent Laws

(i) $p \vee p \equiv p$; **Dual is :** $p \wedge p \equiv p$

5. Double Negation : (Involution Laws)

$\neg (\neg p) = p.$

6. Distributive Laws

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

Dual is : $p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$

7. Absorption Laws

(i) $p \vee (p \wedge q) \equiv p$ Dual : (ii) $p \wedge (p \vee q) = p$

8. Complement Laws

$p \vee \neg p \equiv T$; Dual : $p \wedge \neg p \equiv F$

9. $\neg T \equiv F$, $\neg F \equiv T$

4.9.3 Examples on Logical Equivalency

Ex. 4.9.2 : Show that the propositions $\neg (p \wedge q)$ and $\neg p \vee q$ are logically equivalent.

☑ **Soln. :**

Note : There is an error in the problem we have to show that $\neg (p \wedge q)$ and $(\neg p \vee \neg q)]$ are logically equivalent.

We prepare truth-tables for both expressions.

(i) Truth tables for $\neg (p \wedge q)$ Truth table for $(\neg p \vee \neg q)$

p	q	$p \wedge q$	$\neg (p \wedge q)$
T	T	T	F
T	F	F	T
F	T	F	T
F	F	F	T

p	q	$\neg p$	$\neg q$	$\neg p \vee \neg q$
T	T	F	F	F
T	F	F	T	T
F	T	T	F	T
F	F	T	T	T

Scanned with CamScanner

Since the truth tables are the same, i.e., both propositions are false in the first case and true in the other three cases, ∴ the propositions, ¬ (p ∧ q) and (¬ p) ∨ (¬ q) are logically equivalent ;

i.e., ¬ (p ∧ q) ≡ (¬ p) ∨ (¬ q).

4.10 NORMAL FORMS

(i) Let **P** (P_1, P_2, ..., P_n) be a given statement, where P_1, P_2, ..., P_n are variables. To obtain truth-value of the formula P, we develop truth-table for P. Such a truth-table has 2^n rows.

- If P has **true** value for all possible assignments then P is said to be Tautology.
- If P has **false** value for all possible assignments, then P is said to be **contradiction.**
- If P has **truth** value, T for at least one combination of truth-values of P_1, P_2, ..., P_n then P is said to be **satisfiable.**

(ii) The problem of finding whether a given statement is tautology or contradiction or satisfiable in a finite number of steps is called as **decision problem.**

(iii) For decision problems, the construction of truth-tables may not be a practical solution. Therefore we consider alternate procedure known as reduction to **Normal Forms.**

There are two such forms :

1. Disjunctive Normal Form (DNF).
2. Conjunctive Normal Form (CNF).

These forms are also called as **Canonical forms.**

4.10.1 Disjunctive Normal Form

Disjunctive normal form is a disjunction of fundamental conjunctions (∧). Now fundamental conjunctions (∧) are conjunction of simple statements, i.e. joining two statements by ∧.

i.e. **p, ¬ p, ¬ p ∧ q. p ∧ q, p ∧ ¬ p ∧ q** are fundamental conjunctions.

Some examples of DNF are as follows :

(i) (p ∧ q ∧ r) ∨ (p ∧ r) ∨ (q ∧ r)

(ii) (p ∧ ¬ q) ∨ (p ∧ r)

(iii) (p ∧ q ∧ r) ∨ ¬ q

(iv) (¬ p ∧ q) ∨ (p ∧ q) ∨ q.

(v) **Remark :** q ∧ ¬ q, p ∧ ¬ p are always false, (F).

4.10.2 Examples of DNF

Ex. 4.10.1 : Obtain the d.n.f. of the form

(p → q) ∧ (¬ p ∧ q)

Soln. :

(p → q) ∧ (¬ p ∧ q)

≡ (¬ p ∨ q) ∧ (¬ p ∧ q)

Using distributive law, ∵ [(p → q) = ¬ p ∨ q]

≡ [¬ p ∧ (¬ p ∧ q)] ∨ [q ∧ ¬ p ∧ q]

Using associative and commutative laws ;

≡ [(¬ p ∧ ¬ p) ∧ q] ∨ [q ∧ q ∧ ¬ p]

Using idempotent law ; i.e. ¬ p ∧ ¬ p = ¬ p and q ∧ q = q

= [¬ p ∧ q] ∨ [q ∧ ¬ p] is dnf.

Ex. 4.10.2 : Obtain the d.n.f. of the form: p ∧ (p → q)

Soln. : We have

p ∧ (p → q)

= p ∧ (¬ p ∨ q) (∵ p → q ≡ ¬ p ∨ q)

Using distributire law,

= (p ∧ ¬ p) ∨ (p ∧ q) = F ∨ (p ∧ q) (∵ p ∧ ¬ p = F)

= (p ∧ q) is d.n.f.

4.10.3 Conjunctive Normal Form (cnf)

A statement form, which consists of a **conjunction of fundamental disjunction** is called a conjunctive normal form (abbreviated as cnf). Now fundamental disjunctions (∨) are disjunction of simple statements, i.e. joining two statements by '∨'.

Remark

1. We know that p ∨ ¬ p is always true.
   i.e., p ∨ ¬ p ∨ r is logically equivalent to a tautology.
2. cnf is a tautology if and only if every fundamental disjunction contained in it is a tautology.

Examples

(i) p ∧ r (ii) ¬ p ∧ (p ∨ r)

(iii) (p ∨ q ∨ r) ∧ (¬ p ∨ r)

Module 4

Scanned with CamScanner

4.10.3(A) Conversion from PL to CNF

UQ. Explain the st25eps involved in converting the propositional logic statement into CNF with a suitable example

(MU - Q. 6(a), May 2016, 10 Marks)

To convert propositional formula to CNF, we perform the following steps.

- **Step 1 :** Push **negations** into the formula, repeatedly applying **De Morgan's Law,** until all negations only apply to atoms. We obtain a formula in **negation normal form** :
  - (i) $\neg (p \vee q)$ to $(\neg p) \wedge (\neg q)$
  - (ii) $\neg (p \vee q)$ to $(\neg p) \vee (\neg q)$
- **Step 2 :** We apply distributive law repeatedly where a disjunction occurs over a conjunction. When it is completed, the formula is in CNF.
  - (i) $p \vee (q \wedge r)$ to $(p \vee q) \wedge (p \vee r)$

☞ **Remark**

To obtain a formula in disjunctive normal form, simply apply the distribution of $\wedge$ over $\vee$ in step (2)

Example : $A \rightarrow ((B \wedge C))$

$\equiv \neg A \vee (B \wedge C)$

$\equiv (\neg A \vee B) \wedge (\neg A \vee C)$

UQ. Convert the following propositional logic statement into CNF.

$A \rightarrow (B \leftrightarrow C)$ (MU- Q. 1(e), Dec. 17, 5 Marks)

- **Step (I) :** FOL : $A \rightarrow (B \leftrightarrow C)$
- **Step (II) :** Normalise the given statements

$A \rightarrow (B \rightarrow C \wedge C \rightarrow B)$

$A \rightarrow (B \rightarrow C) \wedge A \rightarrow (C \rightarrow B)$

- **Step (III) :** Converting to CNF :

We apply the rule : $P \rightarrow Q \equiv \neg P \vee Q$

$\therefore \neg A \vee (\neg B \vee C) \wedge \neg A \vee (\neg C \vee B)$

$\neg A \vee \big((\neg B \vee C) \wedge (\neg C \vee B)\big)$

4.10.4 Examples on Conjunctive Normal Form

Ex. 4.10.3 : Obtain conjunctive normal form of

(i) $(\neg p \rightarrow r) \wedge (p \rightarrow q)$ (ii) $(p \wedge q) \vee (\neg p \wedge q \wedge r)$.

☑ **Soln. :**

(I) $(\neg p \rightarrow r) \wedge (p \rightarrow q)$

$= [\neg (\neg p \vee r)] \wedge [p \rightarrow q]$

$= [(\neg\neg p) \vee r] \wedge [\neg p \vee q]$ $(\because p \rightarrow r = \neg p \vee r)$

$= [p \vee r] \wedge [\neg p \vee q]$ is cnf.

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

Using distributive (left) law ;

$= [p \vee (\neg p \wedge q \wedge r)] \wedge [q \vee (\neg p \wedge q \wedge r)]$

$= [(p \vee \neg p) \wedge (p \vee q) \wedge (p \vee r)] \wedge [(q \vee \neg p) \wedge (q \vee q) \wedge (q \vee r)]$

$= [T \wedge (p \vee q) \wedge (p \vee r)] \wedge [(q \vee \neg p) \wedge (q) \wedge (q \vee r)]$

$= [(p \vee q) \wedge (p \vee r)] \wedge [(q \vee \neg p) \wedge (q) \wedge (q \vee r)]$ is cnf.

Ex. 4.10.4 : Obtain cnf of the following :

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

☑ **Soln. :**

$(p \rightarrow q) \wedge (q \rightarrow p)$ $(\because (p \rightarrow q) = \neg p \vee q)$

$= (\neg p \vee q) \wedge (\neg q \vee p)$ is cnf.

Ex. 4.10.5 : Obtain the Conjunctive normal form of :

(i) $p \wedge (p \rightarrow q)$ (ii) $\neg (p \vee q) \leftrightarrow (p \wedge q)$.

☑ **Soln. :**

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

$= p \wedge (\neg p \vee q)$ is cnf. $(\because p \rightarrow q = \neg p \vee q)$

(ii) $\neg (p \vee q) \leftrightarrow (p \wedge q)$.

$= [\neg (\neg (p \vee q) \vee (p \wedge q)] \wedge [\neg (p \wedge q) \vee \neg (p \vee q)]$

$[\because p \leftrightarrow q = (\neg p \vee q) \wedge (\neg q \vee p)]$

$= [(p \vee q) \vee (p \wedge q)] \wedge [(\neg p \vee \neg q) \vee (\neg p \wedge \neg q)]$

$[\because \neg (p \wedge q) = \neg p \vee \neg q; \neg (p \vee q) = \neg p \wedge \neg q]$.

$= [(p \vee q \vee p) \wedge (p \vee q \vee q)] \wedge [(\neg p \vee \neg q \vee \neg p)]$

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

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

$[\because p \vee p = p$ and $\neg p \vee \neg p = \neg p]$

$[\because p \wedge p = p ; \neg p \wedge \neg p = p]$

$= (p \vee q) \wedge (\neg p \vee \neg q)$ is cnf

Scanned with CamScanner

4.11 TRUTH TABLE METHOD TO FIND DNF

Let p be a statement form containing n variables $P_1, P_2, ..., P_n$. Its dnf can be obtained from the truth-table, as follows :

- **Step (I) :** Consider the truth value (T) from P.
- **Step (II) :** Form the conjunction ('$\wedge$') :

  $(P_1 \wedge P_2 \wedge ... \wedge P_i \wedge ... \wedge P_j \wedge ... \wedge P_n)$,

  Where if P_i is true consider P_i and

  if P_j is false consider $\neg P_j$.

  Such a term is called **minterm.**
- **Step (III) :** The disjunction of minterms is the dnf of the given form.

4.11.1 Examples on dnf

Ex. 4.11.1 : Find the dnf of $(\neg p \rightarrow r) \wedge (p \leftrightarrow q)$.

☑ **Soln. :**

- **Step (I) :** First we prepare the truth-table.

p	q	r	¬ p	(¬ p → r)	(p ↔ q)	(¬ p → r) ∧ (p ↔ q)
T	T	T	F	T	T	T
T	T	F	F	T	T	T
T	F	T	F	T	F	F
T	F	F	F	T	F	F
F	T	T	T	T	F	F
F	T	F	T	F	F	F
F	F	T	T	T	T	T
F	F	F	T	F	T	F

- **Step (II) :** (i) Consider only 'T' from last column and choose corresponding values (T) from p, q, r. E.g. for first row, corresponding p, q and r are true, so we consider $(p \wedge q \wedge r)$.

  (ii) Similarly for 2^{nd} row, corresponding p and q are true, but r is false; so we consider $(p \wedge q \wedge \neg r)$.

  (iii) Again for 7^{th} row, corresponding p, q are false and r is true, so we consider, $(\neg p \wedge \neg q \wedge r)$.
- **Step (III) :** Hence the dnf of $(\neg p \rightarrow r) \wedge (p \leftrightarrow q)$ is equal to :

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

4.12 PREDICATE LOGIC (FIRST ORDER LOGIC)

Propositional logic, cannot adequately express the meaning of statements in mathematics and in computer programs.

For example

Statements involving variables such as :

(i) "x > 3" (ii) "x = y + 3" (iii) "x = y + z";

(iv) "Computer x is under attack by an intruder"

(v) "Computer y is functioning properly"

These statements are neither true nor false, if the values of the variables are not specified. Here we discuss the ways that propositions can be produced from such statements.

4.12.1 Predicates

- The statement "**x is greater than 3**" has two parts. The first part, the variable x, is the subject of the statement.

  The second part – called as Predicate "**is greater than 3**" – refers to the property that the subject of the statement can have.
- We denote the statement "x is greater than 3" by P(x), where "P" denotes the **predicate** "is greater than 3" and 'x' is the variable. The statement **P(x) is also said to be the value of P at x.**
- Once a value has been assigned to the variable x, the statement P(x) becomes a **proposition** and has a truth value, and P is called as **propositional function.**

Ex. 4.12.1 : Let P(x) denote the statement "x > 3". What are the truth values of P(4) and P(2).

☑ **Soln. :**

To obtain P(4), we put x = 4 in the statement 'x > 3'. ∴ P(4) is "4 > 3", which is **true.** But P(2), which is the statement "2 > 3" is false.

Ex. 4.12.2 : Consider the statement "x = y + 3". We can denote the statement by Q(x, y), where x and y are variables and Q is the predicate. What are the truth values of the propositions Q(1, 2) and Q(3, 0) ?

☑ **Soln. :**

To obtain Q(1, 2); we put x = 1, y = 2 in the statement Q(x, y). Hence Q (1, 2) : "1 = 2 + 3" which is false.

The statement Q (3, 0) is the proposition "3 = 0 + 3", which is true.

Module 4

Scanned with CamScanner

Ex. 4.12.3 : Let R(x, y, z) denote the statement "x + y = z". What are the truth values of the propositions R (1, 2, 3) and R (0, 0, 1) ?

☑ **Soln. :**

The proposition R (1, 2, 3) is obtained by putting x = 1, y = 2, z = 3 in the statement R (x, y, z). We see that R (1, 2, 3) is "1 + 2 = 3" which is true.

Also, note that R(0, 0, 1) which is the statement "0 + 0 = 1", is false.

☞ **Remark**

1. In general, a statement involving n variables x_1, x_2,...,x_n can be denoted by $P(x_1, x_2,..,x_n)$, and is the value of the propositional function P at the n-tuple $(x_1, x_2, ..., x_n)$

   P is also called an **n-place predicate** or a **n-ary predicate**.
2. A predicate is generally not a proposition but every proposition is a propositional function or a predicate.

⏭ 4.13 THE UNIVERSAL QUANTIFIER

- Many mathematical statements assert that a property is true for all values of a variable in a particular domain. The 'domain' of the values of a variable is also called as 'universe of discourse', or 'domain of discourse', (often just referred to as the 'domain').
- The universal quantification of P(x) for a particular domain is the proposition that tells that P(x) is true for all values of x in the domain.

❑ **Definition :** The universal quantification of **P(x)** is the statement.

- "P(x) for all values of x in the domain." The notation ∀ x P(x) as **"for all x P(x)".**
- An element for which P(x) is false is called a counterexample of **∀ x P(x).**

☞ **Remark (i)**

(1) The symbol '∀' is known as universal quantifier.

(2) The proposition "for all x, P(x)", which is interpreted as "for all values of x, P(x) is true." is a proposition in which the variable x is said to be 'universally quantified' [and '∀' is known as **universal quantifier**]

4.13.1 Existential Quantifiers

- Suppose for the predicate **P(x), ∀ x [P(x)]** is false, but there exists at least one value of x (or some values of x) for which, P(x) is true, then we say that, x is bounded by **existential quantification.**
- The symbol used for 'there exists' is denoted by '∃'. Thus **∃x (P(x))** means 'there exists a value of **x** in the domain for which **P(x)** is true'.

☞ **Remark**

(1) Negation of ∀ x(P(x)) is "∀ x, P(x) is not true", i.e. 'there exists' at least one x for which P(x) is not true, or in other words, there exists an x for which '¬ P(x)' is true.

(2) Observe that the statement ∃ x P(x) is false if and only if there is no element x in the domain for which P(x) is true.

We **summarise the meaning of universal quantification of P(x) :**

Table 4.13.1 : Universal quantification

Statement	When true ?	When false ?
∀ x P(x)	P(x) is true for every x	There is an x for which P(x) is false
∃ x P(x)	There is an x for which P(x) is true	P(x) is false for every x.

Specifying the domain is **mandatory** when **quantifiers** are used. The truth value of a quantified statement often depends on which elements are in the domain.

⏭ 4.14 EXAMPLES OF QUANTIFICATION

Ex. 4.14.1 : Let P(x) be the statement "$x^2 > 0$", where the universe of discourse consists of all integers. What is the truth-value of the quantification ∀ x P(x) ?

☑ **Soln. :**

[**Note :** 'Universe of discourse' is domain of the statement "$x^2 > 0$" consisting of all integers]

We see that x = 0 is a counter-example because $x^2 = 0$ when x = 0, so that x^2 is not greater than 0 when x = 0.

Scanned with CamScanner

Remark

When all the elements in the domain can be listed-say, $x_1, x_2,...x_n$ – it follows that the universal quantification $\forall x\ P(x)$ is same as the conjunction $P(x_1) \wedge P(x_2) \wedge ... \wedge P(x_n)$, because this conjunction is true if and only if $P(x_1), P(x_2),..., P(x_n)$ are all true.

Ex. 4.14.2 : What is the truth-value of $\forall x\ P(x)$, where $P(x)$ is the statement "$x^2 < 10$" and the domain consists of positive integers not exceeding 4 ?

Soln. :

The domain consists of integers 1, 2, 3, 4. We observe that (from the above remark) the statement $\forall x\ P(x)$ is the same as the conjunction.

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

The statement P(4) "$4^2 < 10$" is false

$\therefore \forall x \quad P(x)$ is false.

Ex. 4.14.3 : Let Q(x) denote the statement "x = x + 1". What is the truth value of the quantification $\exists x\ Q(x)$, where the domain consists of all real numbers.

Soln. :

We note that Q(x) is false for every real number x,

$\therefore$ The existential qualification of Q(x), which is $\exists x\ Q(x)$, is **false.**

Remark

- When all elements in the domain can be listed – say, $x_1, x_2, ..., x_n$ the existential quantification $\exists x\ P(x)$ is the same as the disjunction.

  $P(x_1) \vee P(x_2) \vee ... \vee P(x_n)$
- Because this disjunction is true if and only if at least one of $p(x_1), p(x_2), ..., p(x_n)$ is true.

Ex. 4.14.4 : What is the truth value of $\exists x\ P(x)$, where $P(x)$ is the statement "$x^2 > 10$" and the universe of discourse consists of positive integers not exceeding 4 ?

Soln. :

- The given domain is {1, 2, 3, 4}. From the above problem, we say that the proposition $\exists x\ P(x)$ is the same as the disjunction.

  $P(1) \vee P(2) \vee P(3) \vee P(4)$
- Because P(4), which is the statement "$4^2 > 10$", is true, it follows that

  $[P(1) \vee P(2) \vee P(3) \vee P(4)]$ is true

  $\therefore \exists x\ P(x)$ is true

Ex. 4.14.5 : What do the statements $\forall x < 0\ (x^2 > 0)$, $\forall y \neq 0\ (y^3 \neq 0)$ and $\exists z > 0\ (z^2 = 2)$ mean, where the domain in each case consists of the real numbers.

Soln. :

(i) The statement $\forall x < 0\ (x^2 > 0)$ states that for every real number x with $x < 0$, $x^2 > 0$. That is, it states "The square of a real number is positive"

This statement is same as $\forall x\ (x < 0 \rightarrow x^2 > 0)$.

(ii) The statement $\forall y \neq 0\ (y^3 \neq 0)$ states that for every real number y with $y \neq 0$, $y^3 \neq 0$.

That is, it states that "the cube of every non-zero real number is non-zero".

This statement is equivalent to $\forall y\ (y \neq 0 \rightarrow y^3 \neq 0)$

(iii) The statement $\exists z > 0\ (z^2 = 2)$ states that there exists a real number z' with $z > 0$ such that $z^2 = 2$. That is, it states "There is a positive square root of 2". The statement is equivalent to $\exists z\ (z > 0 \wedge z^2 = 2)$.

4.14.1 Free and Bound Variables

- **Bound Variables :** When a quantifier is used on the variable x, we say that this occurrence of the variable is **bound.**
- **Free Variables:** An occurrence of a variable that is not bound by a quantifier is said to be **free.** [A variable is free if it is outside the scope of all quantifiers in the formula that specifies this variable, or set equal to a particular value]
- **For example:** in the statement $\exists x\ (x + y = 1)$, the variable **x** is **bound** by the existential quantification $\exists x$, but the variable **y** is **free** because it is not bound by a quantifier and no value is assigned to this variable.

Module 4

4.14.2 Logical Equivalences Involving Quantifiers

We have introduced the notion **of logical equivalences of compound propositions.** Now, we extend this notion to expressions involving predicates and quantifiers.

- **Definition :** Statements involving predicates and quantifiers are **logically equivalent** if and only if they have the same truth value no matter which predicates are substituted into these statements and which domain of discourse is used for the variables in these propositional functions.

Scanned with CamScanner

We use the notation **S = T** to indicate that two statements S and T involving predicates and quantifiers are logically equivalent.

Ex. 4.14.6 : Show that $\forall x [P(x) \wedge Q(x)]$ and $\forall x P(x) \wedge \forall x Q(x)$ are logically equivalent, where the same domain is used throughout.

Soln. :

Step (I)

- Suppose that $\forall x [P(x) \wedge Q(x)]$ is true. This means that for any 'a' in the domain $p(a) \wedge Q(a)$ is true. Hence, p(a) is true and Q(a) is true.
- Because p(a) is true and Q(a) is true **for every element** in the domain, we conclude that $\forall x P(x)$ and $\forall x Q(x)$ are both true.

$\therefore \forall x P(x) \wedge \forall x Q(x)$ be true

Step II

- Now let $\forall x p(x) \wedge \forall x Q(x)$ be true. It follows that
- $\forall x P(x)$ is true and $\forall x Q(x)$ is true. Hence, if 'a' is in the domain, then p(a) is true and Q(a) is true. [Because P(x) and Q(x) are both true for all elements in the domain].

If follows that **for all** a, $p(a) \wedge Q(a)$ is true. Hence $\forall x [P(x) \wedge Q(x)]$ is true.

$\therefore \forall x [P(x) \wedge Q(x)] = \forall x P(x) \wedge \forall x Q(x)$.

4.14.3 Negating Quantified Expressions

Here we consider the **negation** of a quantified expression:

First we consider **'universal quantifier '∀''**,

We consider an example :

"Every student has taken a course in calculus."

- This statement is a universal quantification, namely, $\forall$ **x P(x),** where **P(x)** is the statement : "x has taken a course in calculus" and **domain** consists of the students in your class.
- **The negation of this statement is** : "It is not the case that every student in your class has taken a course in calculus".

  This statement is equivalent to: "There is a student in your class who has not taken a course in calculus".
- This is simply the existential quantification of the negation of the original propositional function, namely, $\exists x \neg P(x)$.
- The above two statements illustrate the following **logical equivalence** : $\neg \forall x P(x) = \exists x \neg P(x)$.

4.14.4 Negating an Existential Qualification

We take an example to negate an **existential qualification** : Let us consider an example :

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

This is an existential qualification : $\exists x$ **Q(x),** where Q(x) is the statement "x has taken a course in calculus."

The negation of this statement :

(1) "It is not the case that there is a student in this class, who has taken a course in calculus."

This proposition is equivalent to :

(2) "Every student in this class has not taken calculus."

We phrase the above two statements in the language of quantifiers :

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

This is logical equivalence,

4.14.5 De Morgan;s Laws for Negations for Quantifiers

Table 4.14.1 : De Morgan's laws for quantifiers

Negation	Equivalent statement	When is negative true ?	When false
$\neg \exists x P(x)$	$\forall x \neg P(x)$	For every x, P(x) is false.	There is an x for which P(x) is true
$\neg \forall x P(x)$	$\exists x \neg P(x)$	There is an x for which P(x) is false	P(x) is true for every x

4.14.6 Different inference Rules for FOPL.

UQ. Explain different inference Rules for FOPL.

(MU - Q. 4(b), May 18, 10 Marks)

- Inference in First-Order Logic is used to deduce new facts or sentences from existing sentences.
- Let us first see some basic terminologies used in FOL.

Scanned with CamScanner

Substitution : Substitution is a fundamental operation performed on terms and formulas. It occurs in all inference systems in first-order logic. The substitution is complex in the presence of quantifiers in FOL.

If we write F [a/x], so it refers to substitute a constant 'a' in place of variable 'x'.

Equality : First-order logic also uses what is called as Equality in FOL. For this, we can use **equality symbols** which specify that the two terms refer to the same subject.

Example : Brother (Ramesh) = Ashok

Here, the object referred by the Brother (Ramesh) is similar to the object referred by Ashok.

The equality symbol can also be used with negation to represent that two terms are not the same objects.

Example : $\neg (x = y)$ which is equivalent to $x \neq y$.

☞ FOL Inference rules for quantifier

As propositional logic we also have inference rules in first-order logic, the following are some basic inference rules in FOL :

(i) Universal Generalisation
(ii) Universal Instantiation
(iii) Existential Instantiation
(iv) Existential Introduction

▶ (I) Universal Generalisation

Universal generalisation is a valid inference rule which states that if premise P(c) is true for any arbitrary element c, then we can have a conclusion as $\forall x\ P(x)$. It can be represented as :

$$\frac{P(c)}{\forall x\ P(x)}$$

▶ (ii) Universal Instantiation (UI)

Universal Instantiation is also called as universal elimination or UI is a valid inference rule. It can be applied multiple times to add new sentences.

As per UI, **we can infer any sentence obtained by substituting a ground term for the variable.**

The UI rule states that we can infer any sentence P(c) by substituting a ground term c (a constant within domain x) from $\forall x\ P(x)$ for any object in the universe of discourse.

It can be represented as $\frac{\forall x\ P(x)}{P(c)}$

Example (i) : If "Every person like ice-cream" $\rightarrow \forall x\ P(x)$ so we can infer that " John likes ice-cream" $\rightarrow P(c)$.

Example (ii) : "All kings who are greedy are Evil".

In FOL form :

$\forall x$ king (x) $\wedge$ greedy (x) $\rightarrow$ Evil (x)

▶ (iii) Existential Instantiation

Existential Instantiation is also called as Existential Elimination, which is a valid inference rule in first-order logic. It can be applied only to replace the existential sentence.

This rule states that one can infer P(c) from the formula given in the form of $\exists x\ P(x)$ for a new constant symbol c.

It is represented as $\frac{\exists x P(x)}{P(c)}$,

▶ (iv) Existential Introduction

- An existential introduction is also known as an existential generalisation, which is a valid inference rule in first-order logic.
- This rule states that if there is some element c in the universe of discourse which has a property P, then we can infer that there exists something in the universe which has the property P.

It can be expresses as :

$$\frac{P(c)}{\exists x\ P(x)}$$

Example :

'Priyanka got good marks in English'

"Therefore, someone got good marks in English."

☞ Generalised Modus Ponens Rule:

For the inference process in FOL, we have a single inference rule which is called 'Generalised Modus Ponens'.

Generalised modus ponens can be summarised as, " P implies Q and P is asserted to be true, therefore Q must be true".

According to modus ponens, for atomic sentences P_i, P_i, q, where there is a substitution θ such that subst (θ, p_i') = subset (θ, p_i)

It can be written as :

$$\frac{P_1', P_2'}{\ldots, P_n'\ (P_1 \wedge P_2 \wedge \ldots \wedge P_n) \rightarrow q,\ \text{subst} \cdot (\theta, q)}$$

Example : We will use this rule for kings are evil, so we will find some x such that x is king, and x is greedy so we can infer that x is evil.

Here, let p_1' be king (John)	P_1 is king (x)
P_2' is greedy (y)	P_2 is greedy (x)
θ is {x/ John, y/ John}	q is evil (x)
Subst (θ,q)	

Module 4

Scanned with CamScanner

4.14.7 Examples

Ex. 4.14.7 : Consider the statements :

"All humming birds are richly coloured"

"No large birds live on honey"

"Birds that do not live on honey are dull in colour."

"Humming birds are small"

Let P(x), Q(x), R(x) and S(x) be the statements "x is a humming bird", "x is large", "x lives on honey," and "x is richly coloured" respectively. Assuming that the domain consists of all birds, express the statements in the argument using qualifiers and P(x), Q(x), R(x) and S(x).

Soln. :

We express the statements using qualifiers

$\forall x [P(x) \rightarrow S(x)]$

$\neg \exists x [Q(x) \wedge R(x)]$

$\forall x [\neg R(x) \rightarrow \neg S(x)]$

$\forall x [P(x) \neg Q(x)]$

Remarks

(i) Here we assume "small" as "not large" and

(ii) "dull in colour" as "not richly coloured"

Ex. 4.14.8 : For the universe of integers, let p(x), q(x), r(x) and t(x) be the following open statements :

p(x) : x > 0, q(x) : x is even, r(x) : x a perfect square,

t(x) : x is divisible by s.

Write the following statements in symbolic form

(i) At least one integer is even

(ii) There exists a positive integer that is even.

(iii) If x is even then x is not divisible by 5,

(iv) No even integer is divisible by 5,

(v) There exists an even integer divisible by 5.

Soln. :

(i) $\exists x\, q(x)$

(ii) $\exists x [p(x) \wedge q(x)]$

(iii) $\forall x [q(x) \rightarrow \neg t(x)]$

(iv) $\forall x \neg [q(x) \wedge t(x)]$

(v) $\exists x [q(x) \wedge t(x)]$

UEx. 4.14.9 : (MU- Q. 2(b), 2016, 5 Marks)

Write first order logic statements for following statements

Soln. :

(1) If a perfect square is divisible by a prime P then it is also divisible by square of P.

$\exists x$: (x divisible by p) $\rightarrow$ x (divisible by p^2)

(2) x $\rightarrow$ likes ($\neg$ chemistry $\wedge$ $\neg$ History)

$\neg$ x $\vee$ ($\neg$ chemistry $\wedge$ $\neg$ History)

(3) If it is Saturday and warm. Then sam is in park

$\forall x,\ y\ (x \wedge y) \rightarrow$ park (sam)

(4) Anything anyone eats and is not killed by is food.

$\forall x\ \forall y$: eat (x, y) $\wedge$ $\neg$ killed (x) $\rightarrow$ food (y)

$\forall x, \forall y$

$\neg$ (eats (x, y) $\wedge$ $\neg$ killed (x)) $\vee$ food (y)

4.14.8 FOPL

UEx. 4.14.10 : (MU- Q. 1(c), May 2018, 6 Marks)

What is FOPL ? Represent the following sentences using FOPL.

(i) John has atleast two friends

(ii) If two people are friends then they are not enemies.

Soln. :

The First Order Predicate Logic (FOPL) is a method of formal representation of Natural Language (NL) text. The prolog language for AI programming has its foundations in FOPL. If demonstrates how to translate NL to FOPL in the form of facts and rules, use of quantifiers and variables, syntax and semantics of FOPL, and conversion of predicate expression to clause forms. This is followed with unification of predicate expressions using instantiations and substitutions, compositions of substitutions, unification algorithm and its analysis.

(i) x : x (John) $\rightarrow$ at least friends (y $\wedge$z)

(ii) x $\wedge$ y (friends) $\rightarrow$ ($\neg$ x) $\wedge$ ($\neg$ y) (enemies)

Scanned with CamScanner

4.14.9 Comparison between Propositional Logic or First Order Logic (Predicate Logic)

UQ. Distinguish between Propositional Logic (PL) and first order predicate logic (FOPL) knowledge representation mechanisms. Take suitable example for each point of differentiation. **(MU - Q. 2(b), May 19, 10 Marks)**

Sr. No.	Parameters	Propositional Logic (PL)	Predicate Logic (FOL)
1.	Definition	Propositional logic deals with simple declarative propositions.	First order logic additionally covers predicates and quantification.
2.	Entities	A proposition is a collection of declarative statements that has either a truth value "true" or a truth value "false".	Predicate logic is an expression of one or more variables defined on some specific domain.
3.	Boolean values	Propositional logic is a simple form of logic which is also known as Boolean logic.	Predicate logic is a collection of formal systems which uses quantified variables over non-logical objects and allows the use of sentences which contain variables.
4.	Truth values	A proposition has truth values (0, 1) which means it can have one of the two values i.e. True or False.	Predicate logic is an expression consisting of variables with a specific domain. It is also known as Boolean logic.
5.	Usefulness	It is the most basic and widely used logic.	Predicate logic is an extension of propositional logic.
		This logic is used for the development of powerful search algorithms including implementation methods.	Predicate logic deals with infinite structures as well. The quantifiers are the linguistic marks that permit one to treat with such infinite structures.
		Propositional logic is used in AI for planning, problem-solving intelligent control and for decision-making.	A predicate with variables can be made a proposition by either authorising a value to the variable or by quantifying the variable.
6.	Nature	It also includes certainty as well as uncertainty.	It consists of objects, functions, relations between the objects.
7.	Representations	It led to the foundation for machine learning models.	Predicate logic helps analyse the scope of the subject over the predicate.
8.	Language	It is a useful tool for reasoning.	It is different from propositional logic which lacks quantifiers.
9.	Level logic	It has limitation because it cannot see inside prepositions and take advantage of relationships among them.	Predicate logic is undecidable, since universal and existential quantifiers treat with infinite structures.

Module 4

Scanned with CamScanner

4.15 FORWARD CHAINING AND BACKWARD CHAINING

UQ. Explain Forward-chaining and Backward-Chaining algorithm with the help of example.
(MU – Q. 1(a), May 17 4 Marks, Q. 6(C), Dec. 17, 5 Marks, Q. 6(a), Dec. 18, Q. 6(a), May 19, Q. 5(a), Dec. 19, 10 Marks)

Rule-based system architecture consists of a set of rules, a set of facts, and an **inference engine**. The need is to find what new facts can be derived.

Given a set of rules, there are essentially two ways to generate new knowledge: one, forward chaining and the other, backward chaining.

4.15.1 Forward Chaining

UQ. Illustrate Forward chaining in propositional logic with example. (MU - Q. 3(b), May 17, 10 Marks)

Forward chaining employs the system starts from that a set of facts, and a set of rules, and tries to find a way of using those rules and facts to deduce a conclusion or come up with a suitable course of action.

This is known as **data-driven reasoning** because the reasoning starts from a set of data and ends up at the goal, which is the conclusion.

Steps followed in forward chaining

(i) When applying forward chaining, the first step is to take the facts in the **fact database** and see if any combination of these matches all the antecedents of one of the rules in the rule database.

(ii) When all the **antecedents** of a rule are matched by facts in the database, then this rule is **triggered**.

(iii) Usually, when a rule is triggered, it is then **fired**, which means its conclusion is added to the facts database. If the conclusion of the rule that has been fired is an action or a recommendation, then the system may cause that action to take place or the recommendation to be made.

(iv) **For example**, consider the following set of rules used to control an elevator in a three-story building :

Rule 1: IF on first floor and button is pressed on first floor
THEN open door

Rule 2: IF on first floor
AND buttons is pressed on second floor
THEN go to second floor.

Rule 3: IF on first floor
AND buttons is pressed on third floor THEN go to third floor.

Rule 4 : IF on second floor AND button is pressed on first floor AND already going to third floor THEN remember to go to first floor later.

4.15.2 Backward Chaining

UQ. Illustrate Forward chaining and backward chaining in propositional logic with example.
(MU - Q. 3(b), May 17, 10 Marks)

In backward chaining, we start from a conclusion, which is the **hypothesis** we wish to prove, and we aim to show how that conclusion can be reached from the rules and facts in the database.

The conclusion we are aiming to prove is called a **goal**, and so reasoning in this way is known as **goal-driven reasoning**.

Backward chaining used in formulating plans

- A plan is a sequence of actions that a program decides to take to solve a particular problem. Backward chaining can make the process of formulating a plan more efficient than forward chaining.
- Backward chaining in this way starts with the goal state, which is the set of conditions the agent wishes to achieve in carrying out its plan. It now examines this state and sees what actions could lead to it.
- **For example**, if the goal state involves a block being on a table, then one possible action would be to place that block on the table.
- This action might not be possible from the start state, and so further actions need to be added before this action in order to reach it from the start state.
- In this way, a plan can be formulated starting from the goal and working back toward the start state. The benefit in this method is particularly clear in situations where the first state allows a very large number of possible actions.
- In this kind of situation, it can be very inefficient to attempt to formulate a plan using forward chaining

Scanned with CamScanner

because it involves examining every possible action, without paying any attention to which action might be the best one to lead to the goal state.

- Backward chaining ensures that each action that is taken is one that will definitely lead to the goal, and in many cases this will make the planning process far more efficient.

4.15.3 Forward Reasoning

GQ. Explain forward and backward reasoning with examples.

GQ Explain reasoning with example. Compare forward and backward reasoning with example.

GQ Differentiate between forward and backward reasoning.

- **Forward reasoning is also called as forward chaining** in the field of artificial intelligence. It is one of the methods that is used as a reasoning engine with working with inference driven entities.
- Forward reasoning is one of the most population implementation strategies in the concepts of **expert systems and production rule-based systems**.
- With forward chaining, it makes **use of the existing data** alongside the inference rules to extract move data from the user until a certain goal is reached.
- An inference engine **will iterate through the process** of obtaining new data to eventually satisfy the goal.
- The working is based on the **real-world implementation** of the if-then clauses.
- Forward chaining is **a down-up process** in which it works from the bottom to the top in the order of the occurrence of data.

4.15.4 Modus Ponens

UQ. Explain modus ponen with suitable example.

(MU - Q. 1(C), Dec. 17, Q. 1(b), May 16, 4 Marks)

- Forward reasoning (chaining) can be described as repeated application of **'modus ponens'.**
- Forward chaining is a popular implementation strategy for expert systems, business and production rule systems.
- We consider the following famous example which we will use in both forward and backward reasoning.

Remark : 'Modus ponens' is a rule of inference and it states that if P and P $\rightarrow$ Q are true, then Q is also true.

Example : "As per the law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missiles, and all the missiles were sold to it by Robert, who is an American citizen".

Prove that "Robert is criminal".

☑ **Soln. :**

To solve the problem, we convert all the facts into first-order definite clauses, and then we will use a forward-chaining algorithm to reach the goal.

(I) Facts converting into First Order Logic (FOL) : (a) It is a crime for an American to sell weapons to hostile nations. (Let p, q, r be the variables)

American (p) $\wedge$ weapon (q) $\wedge$ sells (p, q, r) $\wedge$ hostile (r) $\rightarrow$ criminal (p) ...(1)

(b) Country A has some missiles

? P Owns (A, p) $\wedge$ Missile (p)

It can be written in two definite clauses by using **Existential Instantiation,** introducing new constant T_1.

Owns (A, T_1) ...(2)

Missile (T_1) ...(3)

(c) All of the missiles were sold to country A by Robert.

? P Missile (p) $\wedge$ Owns (A, p) $\rightarrow$ Sells (Robert, p, A) ...(4)

(d) Missiles are weapons.

Missile (p) $\rightarrow$ Weapons (p) ...(5)

(e) Enemy of America is known as hostile.

Enemy (p, America) $\rightarrow$ Hostile (p) ...(6)

(f) Country A is an enemy of America

Enemy (A, America) ...(7)

(g) Robert is American

American (Robert) ...(8)

Remark : Existential instantiation is also called as (existential elimination) is a rule of inference which says that, given a formula of the form $(\exists x)\ \phi(x)$, one may infer $\phi(c)$ for a new constant symbol $\subset$.

4.15.5 Forward Chaining Proof

▶ **Step 1 :** We begin with the known facts and will choose the sentences which do not have implications, such as : American (Robert), Enemy (A, America), Owns (A, T_1), and Missile (T_1). All these facts are represented as :

American (Robert)		Missile (T_1)		Owns (A, T_1)		Enemy (A, America)

Module 4

Scanned with CamScanner

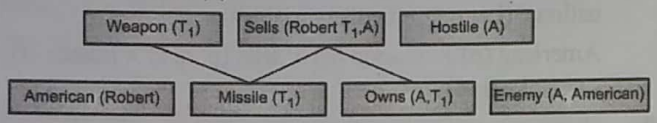
- **Step 2 :** We see the facts which infer from available facts and with satisfied premises.

Rule (1) : does not satisfy premises, so it will not be added in the first iteration.

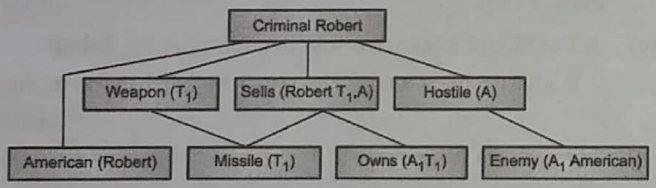
Rule (2) and (3) are already added.

Rule (4) : Satisfy with the substitution $\{p/T_1\}$, so sells (Robert, T_1, A) is added, which infers from the conjunction of Rule (2) and (3)

Rule (5) : is satisfied with the substitution (P/A), so hostile (A) is added and which infers from Rule (6).



- **Step 3 :** As we can check Rule (1) is satisfied with the substitution {p/Robert, q/T_1, r/A}, so we can add criminal (Robert) which infers all the available facts. And hence we reached our goal statement.



Hence it is proved that Robert is criminal using forward reasoning (chaining) approach.

4.15.6 Backward Chaining

Backward chaining is also known as a backward deduction or backward reasoning method when using an inference engine. A backward chaining algorithm is a form of reasoning, which starts with the goal and works backward, chaining through rules to find known facts that support the goal.

4.15.7 Properties of Backward Reasoning (Chaining)

(a) It is known as a top-down approach.

(b) Backward-chaining is based on modus ponens inference rule.

(c) In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true.

(d) It is called a goal-driven approach, as a list of goals decides which rules are selected and used.

(e) Backward-chaining algorithm is used in game theory, automated theorem proving tools, inference engines, proof assistants, and various AI applications.

(f) The backward-chaining method mostly used a depth-first search strategy for proof.

Example :

We use the same above problem in backward-chaining.

We rewrite all the rules :

(a) American (p) $\wedge$ weapon (a) $\wedge$ sells (p, q, r) $\wedge$ hostile (r) $\rightarrow$ criminal (D) ...(1)

(b) Owns (A, T_1) ...(2)

(c) ? p Missiles (p) $\wedge$ Owns (A, p) $\rightarrow$ Sells (Robert, p, A) ...(3)

(d) Missile (p) $\rightarrow$ Weapons (p) ...(4)

(e) Enemy (p, America) $\rightarrow$ Hostile (p) ...(5)

(f) Enemy (A, America) ...(6)

(g) American (Robert) ...(7)

4.15.8 Backward Chaining Proof

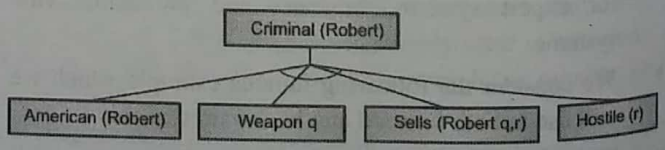
In Backward chaining, we begin with our goal predicate, which is criminal (Robert) and then infer further rules.

- **Step 1 :** We assume the goal fact. And from the goal-fact, we infer other facts, and at last, we prove those facts true.

So our goal fact is "Rober is Criminal", so following is the predicate of it.

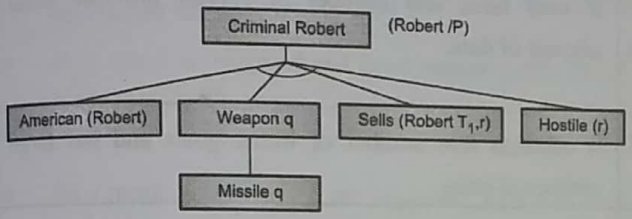
- **Step 2 :** Here we infer other facts from goal fact which satisfies the rules. So as we can see in Rule 1, the goal predicate criminal (Robert) is present with substitution (Robert/p). So we will add all the conjunctive facts below the first level and we replace p with Robert.

Here we can see American (Robert) is a fact, so it is proved here.

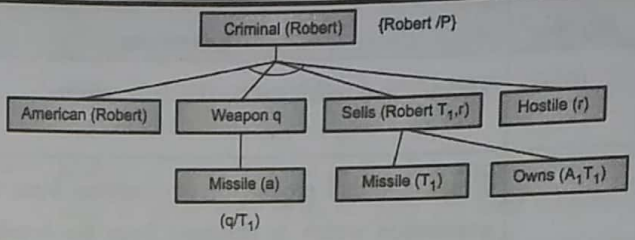


Scanned with CamScanner

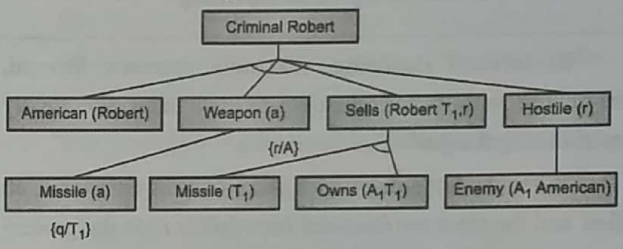
- **Step 3 :** At step 3, we extract further fact Missile (q) which infer from weapon (a) , as it satisfies Rule (5). Weapon (q) is also true with the substitution of a constant T_1 at q.



- **Step 4 :** Now we can infer facts Missile (T_1) and Owns (A, T_1) from sells (Robert, T_1, r) which satisfies the Rule (4) with the substitution of A in place of r.
  So these two statements are proved here.



- **Step 5 :** Now, we can infer the fact Enemy (A, America) from Hostile (A) which satisfies Rule (6). And hence all the statements are proved true using backward chaining.



4.15.9 Comparison between Forward and Backward Reasoning

Sr. No.	Forward Reasoning	Backward reasoning
1	It is a data-driven task.	It is goal driven task.
2	It begins with new data.	It begins with conclusions that are uncertain.
3.	The objective is to find a conclusion	The objective is to find the facts that support the conclusions.
4.	It uses an opportunistic type of approach	It uses a conservative type of approach.
5.	It flows from incipient to the consequence.	It flows from consequence to the incipient.
6.	The precedence of these constraints have to match the current state.	It is based on the decision fetched by the initial state. The system helps choose a goal state, and reasons in a backward direction. It is also known as a decision-driven or goal-driven inference technique.
7.	The inference engine searches the knowledge base with the given information depending on the constraints.	First step is that the goal state and rules are selected.
8.	The first step is that the system is given one or more constraints.	Sub-goals are made from the selected rule, which need to be satisfied for the goal state to be true.
9.	The rules are searched for in the knowledge base for every constraint.	The initial conditions are set such that they satisfy all the sub-goals. The established states are matched to the initial state provided.

Module 4

Scanned with CamScanner

Sr. No.	Forward Reasoning	Backward reasoning
10.	The rule that fulfils the condition is selected.	If the condition is fulfilled, then the goal is the solution. Otherwise the goal is rejected.
11.	Every rule can produce new condition from the conclusion which is obtained from the invoked one.	If tests have less number of rules,it provides small amount of data.
12.	New conditions can be added and are processed again. The step ends if no new conditions exist.	It contains less number of initial goals and has large number of rules
13.	It follows top-down reasoning.	It follows bottom-up reasoning technique.

In forward reasoning, reasoning proceeds forward, beginning with factor, chaining through rules and finally establishing the goal.

When the left side of a sequence of rules is instantiated first and the rules are executed from left to right the process is called forward chaining/reasoning.

This is also known as data-driven search, since, input data are used to guide the direction of the inference process. For example, we can chain forward to show that when a student is encouraged, is healthy, and has goals, the student will succeed.

ENCOURAGED (student) MOTIVATED (students)

MOTIVATED (student) & HEALTHY (student)

WORKHARD (student) WORKHARD (student) & HASGOALS (student)

EXCELL (student) EXCELL (student) → SUCCEED (student)

On the other hand, when the right side of the rules is instantiated first, the left-hand conditions become subgoals. These subgoals may in turn cause sub- subgoals to be established, and so on until facts are found to match the lowest subgoal conditions. When this form of inference takes place, we say that backward chaining is performed. This form of inference is also known as goal-driven inference since an initial goal establishes the backward direction of the inferring.

For example, in MYCIN the initial goal in a consultation is "Does the patient have a certain disease ?" This causes subgoals to be established such as "are certain bacteria present in the patient?" Determining if certain bacteria are present may require such things as tests on cultures taken from the patient. This process of setting up subgoals to confirm a goal continues until all the subgoals are eventually satisfied or fail. If satisfied, the backward chain is established thereby confirming the main goal.

Some systems use both forward and backward chaining/ reasoning, depending on the type of problem and the information available. Likewise rules may be tested exhaustively or selectively, depending on the control structure.

4.16 EXAMPLE TO COMPARE FORWARD AND BACKWARD CHAINING

In this case, we will revert to our use of symbols for logical statements, in order to clarify the explanation, but we could equally well be using rules about elevators or the weather.

Rules :

Rule 1 : $A \wedge B \rightarrow C$ Rule 2 : $A \rightarrow D$

Rule 3 : $C \wedge D \rightarrow E$ Rule 4 : $B \wedge E \wedge F \rightarrow G$

Rule 5 : $A \wedge E \rightarrow H$ Rule 6 : $D \wedge E \wedge H \rightarrow I$

Facts :

Fact 1 : A Fact 2 : B Fact 3 : F

Goal :

Our goal is to prove H :

- First let us use forward chaining. As our conflict resolution strategy, we will fire rules in the order they appear in the database, starting from rule 1.
- In the initial state, rules 1 and 2 are both triggered. We

Scanned with CamScanner

will start by firing rule 1, which means we add C to our fact database. Next, rule 2 is fired, meaning we add D to our fact database.

- We now have the facts A, B, C, D, F, but we have not yet reached our goal, which is G.
- Now rule 3 is triggered and fired, meaning that fact E is added to the database.
- As a result, rules 4 and 5 are triggered. Rule 4 is fired first, resulting in fact G being added to the database, and then rule 5 is fired, and Fact H is added to the database.
- We have now proved our goal and do not need to go on any further. This deduction is presented in the following table :

Table 4.16.1 : Deduction representation

Facts	Rules triggered	Rule fired
A,B,F	1,2	1
A,B,C,F	2	2
A,B,C,D,F,	3	3
A,B,C,D,E,F	4, 5	4
A,B,C,D,E,F,G	5	5
A,B,C,D,E,F,G,H	6	STOP

- Now we will consider the same problem using backward chaining. To do so, we will use a goals database in addition to the rule and fact databases.
- In this case, the goals database starts with just the conclusion, H, which we want to prove. We will now see which rules would need to fire to lead to this conclusion. Rule 5 is the only one that has H as a conclusion, so to prove H, we must prove the antecedents of rule 5, which are A and E.
- Fact A is already in the database, so we only need to prove the other antecedent, E. Therefore, E is added to the goal database. Once we have proved E, we now know that this is sufficient to prove H, so we can remove H from the goals database.
- So now we attempt to prove Fact E. Rule 3 has E as its conclusion, so to prove E, we must prove the antecedents of rule 3, which are C and D.
- Neither of these facts is in the fact database, so we need to prove both of them. They are both therefore added to the goals database. D is the conclusion of rule 2 and rule 2's antecedent, A, is already in the fact database, so we can conclude D and add it to the fact database.
- Similarly, C is the conclusion of rule 1, and rule 1's antecedents, A and B, are both in the fact database. So, we have now proved all the goals in the goal database and have therefore proved H and can stop.
- **This process is represented in the table below :**

Table 4.16.2 : Process representation

Facts	Goals	Matching rules
A,B,F	H	5
A,B,F	E	3
A,B,F	C,D	1
A,B,C,F	D	2
A,B,C,D,F	D	STOP

- In this case, backward chaining needed to use one fewer rule. If the rule database had a large number of other rules that had A, B and F as their antecedents, then forward chaining might well have been even more inefficient.
- In general, backward chaining is appropriate in cases where there are few possible conclusions (or even just one) and many possible facts, not very many of which are necessarily relevant to the conclusion.
- Forward chaining is more appropriate when there are many possible conclusions. The way in which forward or backward chaining is usually chosen is to consider which way an expert would solve the problem. This is particularly appropriate because rule-based reasoning is often used in expert systems.

Module 4

Scanned with CamScanner

4.17 DIFFERENCE BETWEEN FORWARD AND BACKWARD CHAINING

S. No	Forward chaining	Backward chaining
1.	In this a problem solving technique used by expert system when they are faced with a scenario and have to give a solution or conclusion to this scenario.	It is a reasoning technique employed by expert system to take a goal and prove that this goal founded legitimately according to the rule base it possesses.
2.	The system will work its way through the roles, finding which ones fit and which leads to which goals using **deductive reasoning**.	It is a form of reverse engineering, which is very applicable in situations where there are so many roles that could be applied to a single problem. The system could be shifting through roles before it gets anywhere.
3.	Forward chaining is used when a conclusion is not known before hand and it has to reason its way through to one.	Backward chaining is more appropriate when the conclusion is already known.
4.	If matches conditions and then Generates inferences from those conditions. And then generates inferences from those conditions. These conditions can in turn match other roles. Basically, this takes a set of initial conditions and then draws all inferences from those conditions.	Backward chaining is used for **interrogative applications** (finding items that fulfil certain criteria) one commercial example of a backward chaining application might be finding which insurance policies are covered by a particular reinsurance contract.
5.	Starts with initial facts.	Starts with some hypothesis or goal.
6.	Asks many questions.	Asks a few questions.
7.	Tests all the rules.	Tests some rules.
8.	Slow, because it tests all the rules.	Fast, because it tests fewer rules.
9.	Provides a huge amount of information from just a small amount of data.	Provides a small amount of information from just a small amount of data.
10.	Attempts to infer everything possible from the available information.	Searches only that part of the knowledge base that is relevant to the current problem.
11.	Primarily data-driven.	Goal-driven.
12.	Uses input; searches rules for answer.	Begins with a hypothesis; seeks information until the hypothesis is accepted or rejected.
13.	Top-down reasoning.	Bottom-up reasoning.
14.	Works forward to find conclusions from the facts.	Works backward to find facts that support the hypothesis.
15.	Tends to be breadth-first.	Tends to be depth-first.
16.	Suitable for problems that start from data collection, **e.g.** planning, monitoring, control.	Suitable for problems that start from a hypothesis, **e.g.** diagnosis.
17.	Non-focused because it infers all conclusions, and may answer unrelated questions.	Focused; questions all focused to prove the goal and search as only the part of KB that is related to the problem.
18.	Explanation not facilitated.	Explanation is facilitated.
19.	All data is available.	Data must be acquired interactively (i.e. on demand).
20.	A small number of initial states but a high number of conclusions.	A small number of initial goals and a large number of rules that matches the facts.
21.	Forming a goal is difficult.	Easy to form a goal.

Scanned with CamScanner

4.18 SEMANTIC NETWORKS

GQ. What are semantic networks (or semantic nets) and its classification?

- Semantic networks are an **alternative to predicate logic** as a form of knowledge representation.
- The idea is that we can store our knowledge in the form of a **graph**, with **nodes** representing objects in the world, and **arcs** representing relationships between those objects.
- The physical attributes of a person can be represented as in Fig. 4.18.1

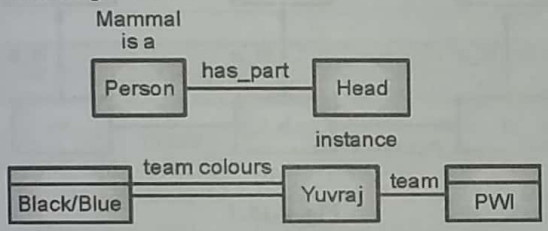


Fig. 4.18.1 : A semantic network

- These values can also be represented in logic as : Is a (person, mammal), instance (Yuvraj, person) team (Yuvraj, PWI). We have already seen how conventional predicates such as lecturer (Poonam) can be written as instance (Poonam, lecturer).
- But, we have a problem: how we can predicate have more than 2 place predicates in semantic nets ? For example score (PWI, India, 20).

  Solution is that, create new nodes to represent new objects either contained or alluded to in the knowledge, game and fixture in the current example. Relate information to nodes and fill up slots.

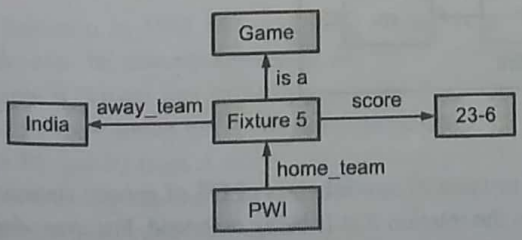


Fig. 4.18.2 : A sematic network for n-place predicate

4.18.1 Advantages, Disadvantages of Semantic Nets

Advantages of semantic nets

1. Semantic network can represent **default values of different categories**.
2. Semantic networks are **simple and easy** to understand.
3. Semantic networks are **easy to translate** in to prologue.
4. Semantic network arcs **represent relationship** between nodes.
5. In semantic networks the **relationships are handled by pointers**.
6. Semantic networks provide good visualization. Being **diagrammatic representation** they are **easy to view**.

Limitations of semantic networks

1. Lack of link names standards; make it difficult to understand the net meaning.
2. Even the nodes naming is not standard. If a node is labelled "**car**" this may mean
   - The class of a car
   - A specific car
   - The concept of a car
3. Answering negative query like "**is XYZ a car**" takes a very long time.
4. Semantic nets are logically inadequate because they cannot define knowledge in a way logic can.
5. Semantic nets can be used better in representing binary relations, but not all types of relations.
6. Logic enhancements have been made and heuristic enhancements have been tried by attaching procedures to the nodes in the semantic nets. The procedures will be executed when the node is activated.

4.18.2 Examples for Semantic Representation

GQ. Give semantic representation for following facts : Ram is taller than Hari.

Ram is taller than hari :

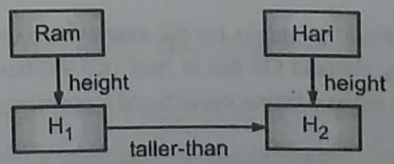


Fig. 4.18.3

Module 4

Scanned with CamScanner

GQ. Construct semantic net representation for the following :

(i) Pompeian (Marcus), Blacksmith (Marcus)

(ii) Rani gave the green flowered vase to her favorite cousin.

(i) Semantic net representation of "Pompeian (Marcus), Blacksmith (Marcus)" will be,

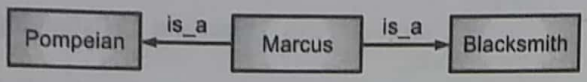


Fig. 4.18.4

(ii) Semantic net representation of "Marry gave the green flowered vase to her favorite cousin" will be,

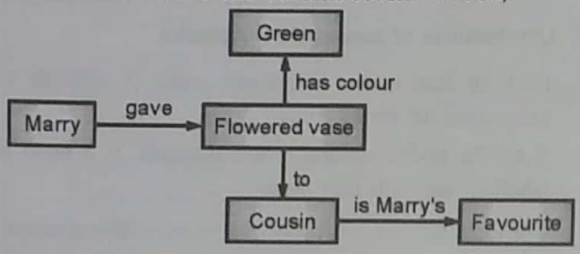


Fig. 4.18.5

GQ.

(i) Represent the following sentence in semantic net :

(a) John gave the book to marry.

(b) Every dog has bitten a postman.

(ii) Bharathiar University Computer Centre, the mini-computer system is a generic node because many mini-computer systems exist and that node has to eater to all of them. On the contrary, Individual of instance nodes explicitly state that they are specific instances of a generic node.

(i) (a) Semantic net representation of sentence "**John gave book to marry**" will be :

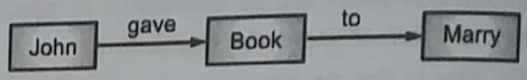


Fig. 4.18.6

(b) Semantic net representation of sentence "**Every dog has bitten a postman**" will be :

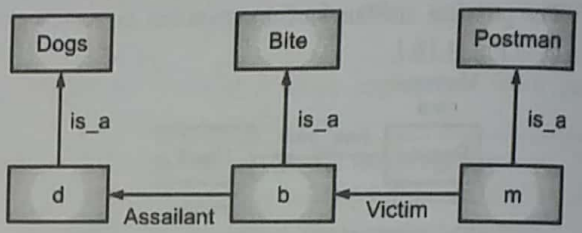


Fig. 4.18.7

The nodes-dogs, bite and hitings and mail earners nodes **d**, b and m represent a particular dog. a particular biting and a particular mail earner This fact can easily be represented by a single net with no partitioning If we represent the fact "Every dog has bitten a postman" or the logic

$\forall(x)$ **Dog(x)** $\rightarrow \exists$ **y Postman-carrier (y)** $\wedge$ **Bite (x, y)**

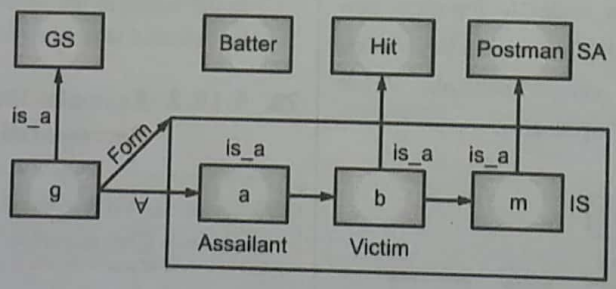


Fig. 4.18.8

(ii) The node '**g**' stands for the **assertion** given above. Node g is an instance of **special class** of **OS** of general statements. Every element GS has at least two attributes, a form which states the relation that is being asserated. For even dog d, there exists a biting event b and a postman m such that d is the assailant of b and m is the victim.

Scanned with CamScanner

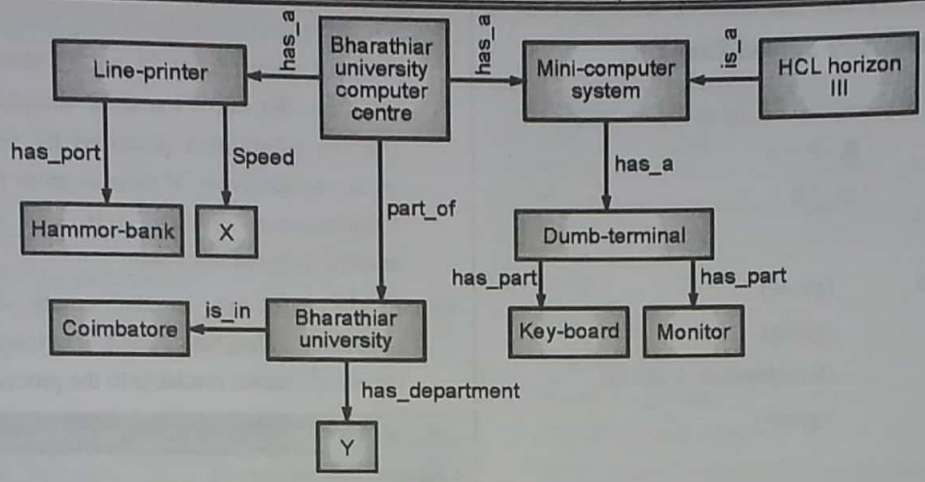


Fig. 4.18.9

4.19 RESOLUTION

GQ. What do you mean by resolution and unification? Explain with example.

Or Write short note on resolution algorithm.

4.19.1 Resolution and Unification

If various statements are given, and we are required to state a conclusion of those statements, then this process is called **Resolution.**

Resolution is a single inference rule which can efficiently operate on the conjunctive normal form or clausal form.

Unification is a 'key concept in proofs' by resolutions.

4.19.2 Resolution Algorithm

Robinson in 1965 introduced the resolution principle which can be directly applied to any set of clauses. The principle is "Given any two clauses A and B, if there is a literal P1 in A which has a complementary literal P2 in B, delete P1 and P2 from A and B and construct a disjunction of the remaining clauses. The clause so constructed is called the resolvent of A and B".

For example, consider the following clauses :

A: $P \vee Q \vee R$ B: $\neg P \vee Q \vee R$

C: $\neg Q \vee R$, D = $Q \vee R$

Clause A has the **literal P** which is complementary to **–P** in **B**. Hence, both of them are deleted and a resolvent (disjunction of A and B after the complementary clauses are removed is generated).

That resolvent again has a **literal Q** whose negation is available in C. Hence resolving those two, one has the final resolvent.

A : $P \vee Q \vee R$	(given in the problem)
B : $-P \vee Q \vee R$	(given in the problem)
D: $Q \vee R$	(resolvent of A and B)
C: $-Q \vee R$	(given in the problem)
E: R	(resolvent of C and D)

Module 4

It is possible to picturize the path of the problem using a **deduction tree**. In fact, it is easier for one to grasp the flow of the problem using the deduction tree. The deduction tree is,

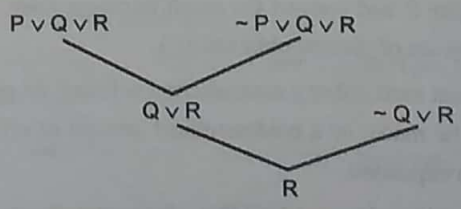


Fig. 4.19.1 : Deduction tree

Sometimes, the resolution might ultimately lead to an empty set or **NIL**. The Following is such an example.

Scanned with CamScanner

4.19.3 University Solved Examples

Ex. 4.19.1 : Perform resolution on the set of clauses

A : P ∨ Q ∨ R B : P ∨ R

C : ¬Q D : ¬R

☑ **Soln. :**

A : P ∨ Q ∨ R (given)

B ¬ P ∨ R (given)

X : Q ∨ R (Resolvent of A and B)

C : ¬Q (given)

Y

: R (Resolvent of X and C)

D : ¬R (given)

Z : NIL (Resolvent of Y and D)

The deduction tree is,

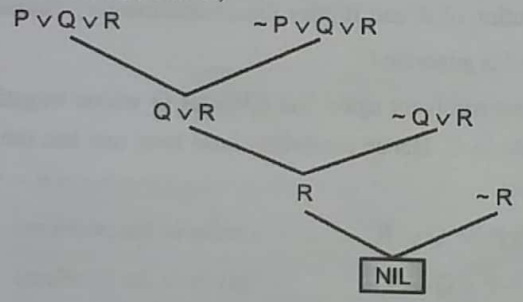


Fig. Ex. 4.19.1

If a A is a formula of predicate calculus, then (x | t). A denotes the formula that results when every occurrence of x in A is substituted by t.

Algorithm Steps

1. Convert all the statements of F to clause form.
2. Negate P and convert the result to clause form. Add it to the set of clauses obtained in 1.
3. Repeat until either a contradiction is found, no progress can be made, or a predetermined amount of effort has been expended.

(i) **Select two clauses :** Call these the parent clause.

(ii) **Resolve them together :** The resolvent will be the disjunction of all literal of both parent clauses with appropriate substitution performed and with the following exception. If there is one pair of literals T1 and ¬ T2 such that one of the parent clauses contains T1 and the other contains T2 and if T1and T2 are unifiable, then neither T1 nor T2 should appear in the resolvent. We call T1 and T2 complementary literals. Use the substitution produced by the unification to create the resolvent. If there is more than one, pair of complementary literals, only one pair should be omitted from the resolvent.

(iii) If the resolvent is the empty clause, then a contradiction has been found. If it is not, then add it to the set of clauses available to the procedure.

UEx. 4.19.2 (MU - Q. 1(c), Dec. 19, 5 Marks)

Consider following facts :

1. It is Humid
2. If it is Humid then it is hot
3. If it is hot and humid then it will rain. Prove that "It will Rain"

☑ **Soln. :**

▶ **Step I : Propositional symbols :**

It is humid : H

It is hot : O

It will rain : R

▶ **Step II : Propositional logic**

(i) H (ii) $H \rightarrow O$ (iii) $H \wedge O \rightarrow R$

▶ **Step III : In CNF form**

(i) H (ii) $\neg H \vee O$ (iii) $\neg H \vee \neg O \vee R$

▶ **Step IV :** We assume negation

It is not raining, i.e. $\neg R$

▶ **Step V :** We form resolution tree.

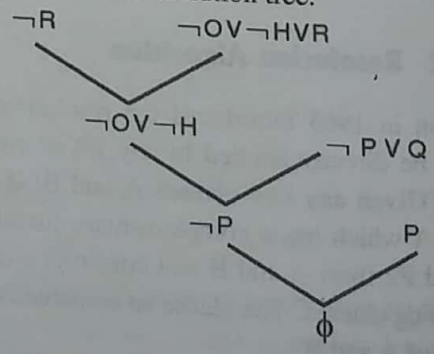


Fig. Ex. 4.19.2 : Resolution Tree

Since the end is empty.

∴ We conclude that it will rain.

Scanned with CamScanner

UEx. 4.19.3 (MU - Q. 3(b), Dec. 15, 10 Marks)

Consider the following axioms:

All people who are graduating are happy.

All happy people smile.

Someone is graduating.

Explain the following :

i) Represent these axioms in first order predicate logic.

ii) Convert each formula to clause form.

iii) Prove that "Is someone smiling?" using resolution technique. Draw the resolution tree.

☑ **Soln. :**

▶ **Step I :** Symbolic logic :

x = people

G = people graduating,

H = happy people,

S = smiling people

▶ **Step II : First order propositional logic**

(i) $\forall x\ G\ \forall x\ H$

(ii) $\forall x\ H,\ \forall x\ S$

(iii) $\exists x\ G \vee \forall x\ G(f(x)) \simeq G(y)$

▶ **Step III : In clause form :**

(i) $\forall x\ G\ \forall x\ H$;

CNF : $\neg G \vee H$

(ii) $\forall x\ H,\ \forall S(x)$;

CNF : $\neg H \vee S(x)$

(iii) Clause form :

$\exists x \neg G \vee H$

(iv) $\forall x \neg H \vee S$

(v) $\exists x\ G$

▶ **Step IV :** We negate the conclusion :

$\neg \exists x\ S(x) \simeq \forall x \neg S$

Resolution tree

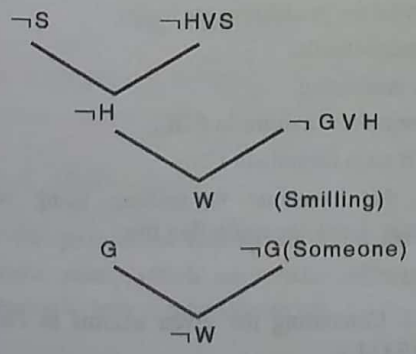


Fig. Ex. 4.19.3

∴ Someone is smiling.

UEx. 4.19.4 (MU - Q. 4(a), Dec. 18, 10 Marks, Q. 4(A), Dec. 17, 10 Marks)

Consider the statements : mammals drink milk, man is mortal, man is a mammal, Tom is a man and we are supposed to prove these.

☑ **Soln. :**

▶ **Step I :** We have

Tom is a man,

Man is a mammal,

Mammals drink milk.

So we have to establish that Tom drinks milk.

First we write down the implication proposition.

M : Mammals drink milk.

Mammals (Tom) → drink (Tom, Milk)

A : Man is mortal.

Man (Tom) → mortal (Tom)

Man is a mammal.

N : Man (Tom) → mammal (Tom)

Tom is a man.

S : Man (Tom)

Goal : Tom drinks (milk)

▶ **Step II :** We note that

(i) Mammal (Tom) → drink (Tom, milk)

(ii) Man (Tom) → mortal (Tom)

(iii) Man (Tom) → mammal (Tom) are propositions and S = man (Tom) is an assertion.

▶ **Step III :** Now in disjunction form,

(i) Not mammal (Tom) OR drink (Tom, Milk)

(ii) Not man (Tom) OR mortal (Tom).

(iii) Not man (Tom) OR mammal (Tom).

▶ **Step III : Resolution Tree :**

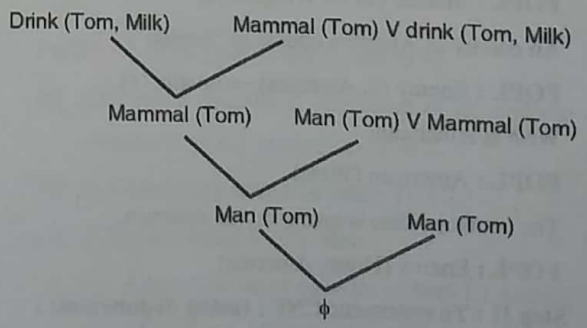


Fig. Ex. 4.19.4

Module 4

Scanned with CamScanner

Thus man (Tom) is not a man but

Tom is a man (∵ We have arrived at empty clause)

∴ Tom does not drink milk is contradiction.

∴ Tom drinks milk.

UEx. 4.19.5 (MU - Q. 4(a), May 16, 10 Marks, Q. 4(a), May 19, 10 Marks)

The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles and all of its missiles were sold to it by Colonel West, who is American.

i. Represent the above sentences in first order predicate logic (FOPL).

ii. Convert them to clause form.

iii. Prove that "West is Criminal" using resolution technique.

☑ **Soln. :**

▶ **Step I : To represent sentences in FOPL :**

1. It is a crime for an American to sell weapons to hostile nations.

   Let x : American, y : Weapon

   z : Hostile, M : Missile.

   FOPL : American (x) ∧ weapon (y) ∧ sells (x, y, z) ∧ Hostile (z) → criminal (x).

2. Nono has some missiles :

   ∃x Owns (Nono, x) ∧ Missile (x).

   ∴ FOPL : Owns (Nono, M) and missile (M).

3. All of its missiles were sold by Colonel West.

   FOPL : Missile (x) ∧ Owns (Nono, x) → sells (West, x, Nono).

4. Missiles are weapons.

   FOPL : Missile (x) → Weapon (x)

5. An enemy of America counts as "hostile".

   FOPL : Enemy (x, America) → hostile (x)

6. West is American.

   FOPL : American (West)

7. The country Nono is an enemy of America.

   FOPL : Enemy (Nono, America)

▶ **Step II : To represent CNF : (using disjunction) :**

1. ¬ American (x) ∨ ¬ weapon (y) ∨ ¬ sells (x, y, z) ∨ ¬ Hostile (z) ∨ criminal (x).
2. Owns (Nono, M), Missile (M)
3. ¬ Missile (x) ∨ ¬ owns (None, x) ∨ sells (West, x, Nono)
4. ¬ Missile (x) ∨ weapon (x)
5. (An enemy of America counts as "hostile")

   ¬ Enemy (x, America) ∨ Hostile (x)
6. (West, Who is American)

   American (West).
7. (The country Nono, an enemy of America)

   Enemy (Nono, America).

▶ **Step III : 'Resolution Technique' (using CNF) :**

1. ¬ American (x) ∨ ¬ weapon (y) ∨ ¬ sells (x, y, z) ∨ ¬ Hostile (z) ∨ criminal (x).
2. ¬ Missile (x) ∨ ¬ owns (Nono, x) ∨ sells (West, x, Nono)
3. ¬ enemy (x, America) ∨ Hostile (x)
4. ¬ Missile (x) ∨ weapon (x)
5. Owns (Nono, M)
6. Missile (M)
7. American (West)
8. Enemy (Nano, America)
9. ¬ Criminal (West)

▶ **Step IV : Conclusion**

We discard that West is not criminal. Hence, we conclude that 'West is criminal'.

UEx. 4.19.6 (MU : Dec. 15, 12 Marks)

Consider following axioms :

All people who are graduating are happy.

All happy people smile.

Someone is graduating

(i) Represent these axioms in FOL.

(ii) Convert each formula in CNF

(iii) Prove that someone is smiling using resolution technique. Draw the resolution tree.

☑ **Soln. :**

▶ **Step I :** Converting the given axioms in First Order Logic (F.O.L)

Scanned with CamScanner

(i) Let x stand for people :

(a) $\forall x$: graduating (x) $\rightarrow$ happy (x)

(b) $\forall x$: happy (x) $\rightarrow$ smile (x)

(c) Someone is graduating : $\exists x$: graduating (x)

▶ **Step II :**

Converting First Order Logic (F.O.L.) to conjuctive normal form (C. N. F.)

Note that (x $\rightarrow$ y) is equivalent to ($\neg$x $\vee$y)

$\therefore$ (a) $\neg$ graduating (x) $\vee$ happy (x)

(b) $\neg$ happy (x_1) $\vee$ smile (x_1)

(c) graduating (x_2)

▶ **Step III :**

We use Resolution Technique
To show that x_3 smile (x_3).
We negate the statement.
i.e. $\neg$ smile (x_3)

Resolution tree

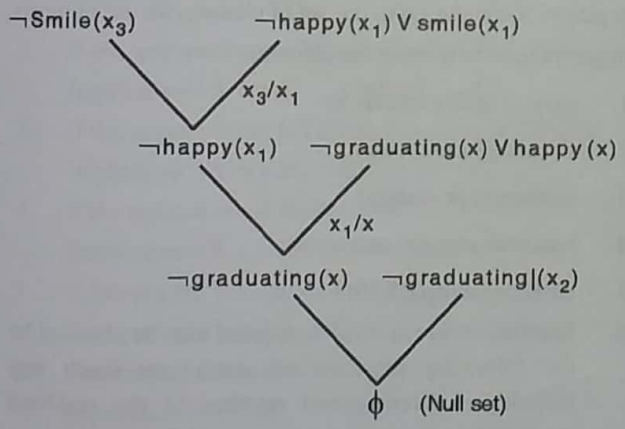


Fig. Ex. 4.19.6

$\therefore$ Our assumption is wrong.

$\therefore$ Someone is smiling.

4.19.4 Unification

1. It is the process of finding substitutions for lifted inference rules, which can make different logical expression to look similar (identical)
2. Unification is a procedure for **determining substitutions** needed to make two first order logic expressions match.
3. Unification is important component of all **first order logic inference algorithms**.
4. The unification algorithm takes two sentences and returns a unifier for them, if one exists.

❑ **Unifier :** A substitution that make two clauses resolvable is called a unifier and the process of identifying such unifiers is carried out by the unification algorithm.

The unification algorithm tries to find out the **most General Unifier (MGU)** between a given set of atomic formulae. Any substitution that makes 2 and more expression equal is called as verifying lineal.

☞ **Algorithm : Unify (L1, L2)**

1. If L1 or L2 are both variables or constants, then:
   (a) If L1 and L2 are identical, then return NIL.
   (b) Else if L1 is a variable, then if L1 occurs in L2 then return {FAIL}, else return (L2/L1).
   (c) Else if L2 is a variable then if L2 occurs in L1 then return {FAIL}, else (L1/L2).
   (d) Else return {FAIL}.
2. If the initial predicate symbols in L1 and L2 are not identical, then return {FAIL}.
3. If L1 and L2 have a different number of arguments, then return {FAIL}.
4. Set SUBST to NIL. (AT the end of this procedure, SUBST will contain all the substitutions unify L1 and L2).
5. For **i** $\leftarrow$ **1 to** number of arguments in L1:
   (a) Call unify with the i^{th} argument of L1 and the i^{th} argument of L2, putting result in S.
   (b) If S contains FAIL then return {FAIL}.
   (c) If S is not equal to NIL then :
      (i) Apply S to the remainder of both L1 and L2.
      (ii) SUBST : = APPENDS (S, SUBST).
6. Return SUBST.

Module 4

Scanned with CamScanner

4.19.5 Conflict Resolution

GQ. Explain conflict resolution. Explain the term refutation. And let's take an example to explain this :

The following statements are assumed to be true :

1. Steve only likes easy courses.
2. Science courses are hard.
3. All the courses in the basket weaving department are easy.
4. BK30I is a basket weaving course. We ask : What course would steve like?

OR

GQ. What is conflict resolution? Illustrate with an example in any production system.

❑ **Definition :** Conflict set is the set of rules that have their conditions satisfied by **working memory elements**. Conflict resolution normally selects a single rule to fire.

☞ **The popular conflict resolution mechanisms are :**

1. Refractory 2. Recency 3. Specificity

1. **Refractory :** A rule should not be allowed to fire more than once on the same data. Discards executed rules from the conflict set. Prevents undesired loops.
2. **Recency :** Rank instantiations in terms of the recency of the elements in the premise of the rule. Rules which use more recent data are preferred. Working memory elements are time-tagged indicating at what cycle each fact was added to working memory.
3. **Specificity :** Rules which have a greater number of conditions and are therefore more difficult to satisfy, are preferred to more general rules with fewer conditions. More specific rules are 'better' because they take more of the data into account.

4.19.6 Refutation

UQ. Explain Resolution by Refutation with suitable example. (MU - Q. 3(a), May 18, 10 Marks)

Refutation is nothing but a technique that a resolution procedure used to prove a statement, i.e., an attempt to show that negation of the statement produces a contradiction with known statements.

We consider an example :

The following statements are assumed to be true :

1. Steve only likes easy courses.
2. Science courses are hard.
3. All the courses in the basket-weaving department are easy.
4. BK 101 is a basket-weaving course.

We ask : What course would Steve like ?

The predicate logic encoding of the premises of the previous problem is as follows:

1. $\forall(x)$ easy(x) $\rightarrow$ likes(steve, x)
2. $\forall$ (x) science(x) $\rightarrow$ easy(x)
3. $\forall$ (x) basket weaving(x) $\rightarrow$ easy(x)
4. basket weaving (BK30l)

The conclusion is encoded as, likes(steve, x).

First we put our premises in the clause form and the negation of conclusion to our set of clauses (we use numbers in parentheses to number the clauses):

1. easy(x) or likes(steve, x)
2. science(x) or -easy(x)
3. science(x) or -easy(x)
4. basketweaving(x) or easy(x)
5. basketweaving(BK301)
6. **likes(steve, x) :** A resolution proof may be obtained by the following sequence of resolutions (each step includes a parenthesized number of the resolvent generated in the current step; 1 and 5 means that we resolve clauses (1) and (5).
7. 1 and 6 yields resolvent-easy(x).
8. 4 and 7 yields resolvent ~basketweaving(x).
9. 5 and 8 yields empty clause; the substitution X/BK30I is produced by the unification algorithm which says that the only wff of the form likes (steve, x) which follows from the premises is likes(steve,BK30l).

Thus, resolution gives us a way to find additional assumptions (in this case **x = BK301**) which make our theorem true.

Scanned with CamScanner

4.19.7 Example based on Propositional Variables

GQ.

(i) If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.

Use these propositional variables :

Y = unicorn is mYthical R = unicorn is mortal

M = unicorn is a maMmal H = unicorn is Horned

G = unicorn is magical

Convert the English into propositional logic implicative form and conjunctive normal form (CNF). The first one is done for you as an example. (Note : "immortal" means "not mortal.")

(ii) Perform Resolute on the set of clauses

A : P ∨ Q ∨ R , B : ~ P ∨ R , C : ~ Q D : ~ R

(i)

1. If the unicorn is mYthical, then it is not moRtal.
   Implicative Y = > ¬ R. CNF (¬ Y ∨ ¬ R).
2. If the unicorn is not mYthical, then it is mortal
   Implicative ¬ Y = > R. CNF (Y ∨ R).
3. If the unicorn is not mYthical, then it is a maMmal.
   Implicative ¬ Y = > M. CNF (Y ∨ M).
4. If the unicorn is not moRtal, then it is Horned.
   Implicative ¬ R = > H. CNF (R ∨ H).
5. If the unicorn is a maMmal, then it is Horned.
   Implicative M = > H. CNF (¬ M ∨ H).
6. The unicorn is maGical if it is Horned.
   Implicative H = > G. CNF (¬ H ∨ G).

(ii)

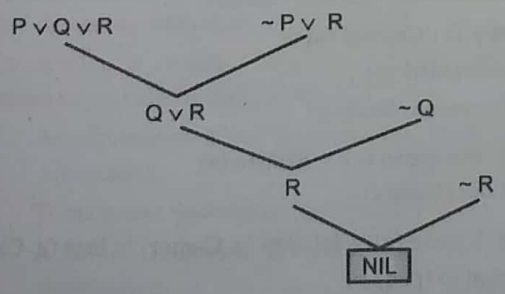


Fig. 4.19.2

Ex. 4.19.6 : Convert the following statement into clause form : (∀x) (∃y) (∀z) [p (x, y) ∧ q (x, z) → r (z, x)]

☑ **Soln. :**

Applying rule (1), we have

(∀x) (∃y) (∀z) [¬ p (x, y) ∧ q (x, z) ∨ r (z, x)]

Applying rule (2), we get

(∀x) (∃y) (∀z) [¬ p (x, y) ∨ ¬ q (x, z) ∨ r (z, x)]

Applying rule (5); we get

(∀x) (∀z) [¬ p (x, A (x)) ∨ ¬ q (x, z) ∨ r (z, x)]

By applying rule (6), we get

[¬ p (x, A (x)) ∨ ¬ q (x, z) ∨ r (z, x)]

By applying rule (7)

¬ p (x, A (x)) ∨ ¬ q (x, z) ∨ r (z, x)

The above expression is a clausal form.

Ex. 4.19.7 : Convert the following sentences into propositional logic and prove by resolution that :

"Schweta will go to college"

1. Schweta likes maths and she likes stories.
2. If she likes maths then she likes algebra.
3. If she likes algebra and likes physics then she will go to college.
4. She does not like stories or she likes physics.
5. She does not like chemistry and history.

Module 4

☑ **Soln. : Propositional logic :**

Let Schweta like maths = M

Schweta like stories = S

Schweta like algebra = A

Schweta like physics = P

Schweta will to go to college = C

Schweta likes history = H

Schweta likes Chemisty = Z

(1) M ∧ S (2) M → A (3) A ∧ P → C

(4) ¬ S ∨ P (5) ¬ H ∧ ¬ Z

Clausal form

1. MA 2. S
3. ¬ M ∨ A 4. ¬ (A ∧ P) ∨ C = ¬ ∨ ¬ P ∨ C
5. ¬ S ∨ P 6. ¬ H 7. ¬ Z

Scanned with CamScanner

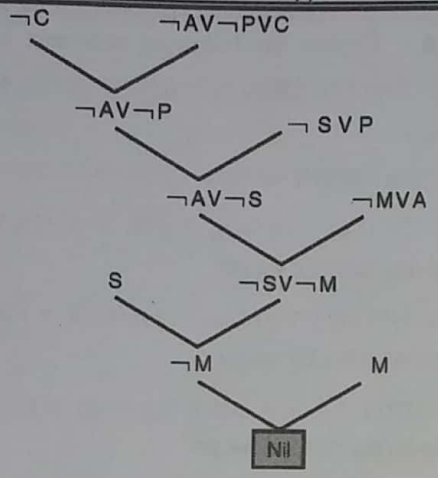


Fig. Ex. 4.19.7

Resolution proof for 'Alice will go to college'

Ex. 4.19.8 : Convert the following sentences into propositional logic and prove by resolution that, "What course would Steeve like?"

1. Steeve only likes easy course
2. Science courses are hard.
3. All the courses in CSE department are easy.
4. DBMS is a CSE course.

Use Resolution to prove that "What course would Steeve like ?"

☑ **Soln. :**

▶ **Step I : First order propositional logic**

1. ∀x course (x) ∧ easy (x) → likes (Steeve, x)
2. Course (science) → ¬ as y (science)
3. ∀a : course (a) ∧ CSE (a) → easy (a)
4. CSE (DBMS)

▶ **Step II : Clausal form**

1. ¬ course (x) ∨ ¬ easy (x) ∨ like (Steeve, x)
2. ¬ course (science) ¬ ∨ ¬ easy (science).
3. ¬ course (a) ∨ ¬ CSE (a) ∨ easy (a)
4. CSE (DBMS).

▶ **Step III :** Conclusion

What course would Steeve like ?

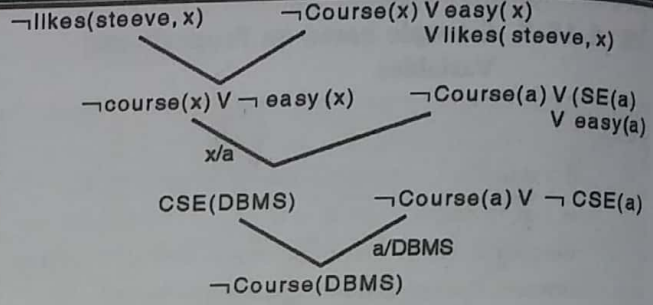


Fig. Ex. 4.19.8

∴ Steeve likes DBMS.

Ex. 4.19.9

1. Markus was a man.
2. Markus was a pompiean.
3. All pompieans were Romans.
4. Caesor ws a ruler.
5. All Romans were either loyal to Caesor or hated him.
6. Everyone is loyal to someone.
7. People only try to assasinate rulers that they are not loyal to.
8. Markus tried to assasinate Caesor.

Proof by resolution that Markus hates Caesor.

☑ **Soln. :**

▶ **Step I : First Order Predicate Logic (FOPL)**

1. Man (Markus)
2. Pompiean (Markus)
3. ∀x : Pompiean (x) → Roman (x)
4. Ruler (Caesor)
5. ∀a : Roman (a) → Loyal to (a, Caesor) ∨ hate (a, Caesor)
6. ∀p, ∃q : Loyal to (p, q)
7. ∀y, ∀z : People (y) ∧ ruler (z) ∧ assassinate (y, z). → ¬ loyal to (y, z).
8. Assassinate (Markus, Caesor).

▶ **Step II : Clausal form**

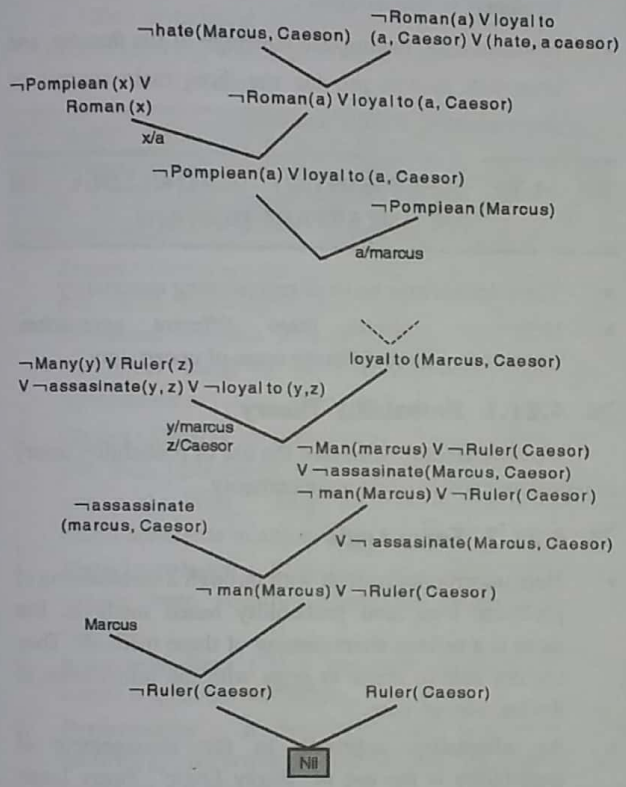
1. Man (Markus)
2. Pompiean (Markus)
3. ¬ Pompiean (x) ∨ Roman (x)
4. Ruler (Caesor)
5. ¬ Roman (a) ∨ loyal to (a, Caesor) ∨ hate (a, Caesor)
6. loyal to (p, q)
7. ¬ (y) ∨ ¬ Ruler (z) ∨ ¬ assasinate (y, z) V → loyal to (y, z)
8. Assasinate (Markus, Caesor)

Scanned with CamScanner

Step III : Conclusion

hate (Markus, Caesor)

Resolution Tree



A resolution proof
Fig. Ex. 4.19.9

4.20 UNCERTAIN KNOWLEDGE AND REASONING

- In a reasoning system, there are several types of uncertainty. Reasoning under uncertainty research in AI is 'focused on uncertainty of truth value' in order to find the values other than True and False.
- To develop a system that reasons with uncertainty means to provide the following :
  1. An explanation about the origin and nature of the uncertainty.
  2. To represent uncertainty in a formal language.
  3. A set of inference rules that derive uncertain conclusions.
  4. An efficient memory-control mechanism for uncertainty management.

4.20.1 Non-monotonic Logics

- A reasoning system is monotonic if the truthfulness of a conclusion does not change when new information is added to the system.
- In contrast, in a system doing non-monotonic reasoning the set of conclusions may either grow or shrink when new information is obtained.
- Simply speaking, the truth value of propositions in a non-monotonic logic can be classified into the following types :
  1. Facts that are definitely true, such as 'Crow is a bird'.
  2. Default rules that are normally true, such as 'Birds fly'.
  3. Tentative conclusions that are true, such as 'Crow flies'.
- **Remark :** When an inconsistency is recognised, only the truth value of the last type is changed.

4.20.2 Probabilistic Reasoning

- The basic idea of probabilistic reasoning is to use probability theory to represent and process uncertainty.
- Though no conclusion from probability theory is absolutely true, the one with the highest probability is preferred. Under certain conditions (assumptions) probability theory gives the optimum solutions.
- "It is remarkable that a science which began with the consideration of games of chance should become the most important object of human knowledge. The most important questions of life are, for the most part, really only problems of probability.The theory of probabilities is at bottom nothing but common sense reduced to Calculus."

  – **Pierre Simon de Laplace(1812).**

We extend the basic Boolean connectives to probability functions :

1. **Negation :** $P(\neg A) = 1 - P(A)$
2. **Conjunction :** $P(A \wedge B) = P(A) \cdot P(B)$, if A and B are independent of each other.
3. **Disjunction :** $P(A \vee B) = P(A) + P(B)$, if A and B never happen at the same time.
4. **Conditional :** Probability of B given A is

$$P(B / A) = \frac{P(B \wedge A)}{P(A)};$$

From which Bayes' theorem is derived

Module 4

Scanned with CamScanner

5. Bayesian reasoning determines the probability of an event with uncertain knowledge.

   It is a way to calculate the value of P (B/A) with the knowledge of P (A/B).

4.20.3 Bayesian Networks

- Bayesian Networks are directed graphs in which the nodes represent a variables of interest and the links represent informational or causal dependencies **among** the variables.
- The strength of dependency is represented by conditional probabilities.

4.20.4 Challenges to Probabilistic Approaches

1. Unknown probability values.
2. Inconsistent probability assignments.
3. Computational expense.

Considering the uncertainty in probability judgements, we go further to study 'imprecise probability'.

4.20.5 Fuzzy Logic

- 'Fuzzy logic' is the reflection of the impression of human language and reasoning.
- Examples of fuzzy concepts : 'young', 'furniture', 'most', 'cloudy' and so on.
- According to Fuzzy Logic, whether an instance belongs to a concept is usually a matter of 'degree'. Fuzzy logic uses a 'degree of membership' which is a real number in [0, 1].

4.20.6 Basic Fuzzy Operators

1. **Negation :** $M(\neg A) = 1 - M(A)$.
2. **Conjunction :** $M(A \wedge B) = \text{Min } \{M(A), M(B)\}$.
3. **Disjunction :** $M(A \vee B) = \text{Max } \{M(A), M(B)\}$.

In building a Fuzzy system, the designer needs to provide all membership functions included in it, by considering how the concepts are used by average people.

4.20.7 Challenges to Fuzzy Approaches

1. Degree of membership is often dependent on content.
2. General purpose fuzzy rules are hard to obtain.

4.20.8 Causes of Uncertainty in AI

- In machine learning the biggest source of difficulty for beginners is 'Uncertainty'.
- Noise in data, incomplete coverage of the domain, and imperfect models provide the three main sources of uncertainty in machine learning.

4.21 REPRESENTING KNOWLEDGE IN AN UNCERTAIN DOMAIN

- There are various ways of representing uncertainty.
- Here we consider three different approaches, representing three different areas of uncertainty :

4.21.1 Probability Theory

We have already discussed the use of probability theory to represent knowledge in an uncertainty.

4.21.2 Fuzzy Logic

- Here uncertainty is dealt with through a combination of predicate logic and probability based methods. But there is a serious short coming of these methods. They are not able to come to grips with the information in the knowledge-base.
- An alternative approach to the management of uncertainty is the use of 'Fuzzy Logic'. Fuzzy Logic provides a systematic framework for dealing with fuzzy quantifiers, e.g., most, many, about 0.7, etc.
- Fuzzy Logic considers both predicate logic and probability theory, and deals with different types of uncertainty within a single conceptual framework.
- In Fuzzy Logic, the deduction of a conclusion from a set of premises is reduced. And from a non linear program through the application of projection and extension principles, a solution is obtained.

4.21.3 Truth Maintenance System : (TMS)

- To choose the actions, reasoning programs must be able to make assumptions and then revise their beliefs when discoveries contradict these assumptions.
- The Truth Maintenance System (TMS) is a problem solver subsystem for performing these functions by recording and maintaining the reasons for program beliefs. Such recorded reasons are useful in constructing explanations of program actions and in guiding the course of action of a problem solver.

Scanned with CamScanner

- TMS can be best visualised in terms of graphs.
- It stores the latest truth value of any predicate. The system is developed with the idea that truthfulness of a predicate can change with the time, as new knowledge is added or existing knowledge is updated. It keeps a record showing which items of knowledge is currently believed or disbelieved.

4.22 REPRESENTATION OF KNOWLEDGE

In AI systems, knowledge is represented in the following manner.

1. **Events :** Events are the occurrence of things in the real world. Anything which happens in real time are considered as events. It is an important element as it is the initial thing to be considered in knowledge representation.
2. **Object :** Objects are nothing but facts that are actually true. Such facts can be universal truth such as "The sun sets in the West." "Dogs are faithful", or any facts which holds true in any events.
3. **Meta-knowledge :** It is those knowledge which are already been acquired either by human brain or machine.
4. **Knowledge-base :** It is the core component of the agents acquiring knowledge.
5. **Performance :** It describes about how good the knowledge is acquired and it can be applied to machines.

4.22.1 Categories of Knowledge

There are two main categories of knowledge :

1. Tacit knowledge
2. Explicit knowledge

1. Tacit knowledge

- Tacit knowledge is the knowledge which exists within a human being. It does correspond to informal or implicit type of knowledge.
- It is quite difficult to articulate formally and is also difficult to communicate and share.

2. Explicit knowledge

- Explicit knowledge is the knowledge which exist outside a human being.
- It corresponds to formal type of knowledge, it is easier to articulate compared to tacit knowledge and is easier to share, store or even process.

4.22.2 Types of Knowledge Representation

Types of knowledge representations are as following :

A. Declarative knowledge
B. Procedural knowledge
C. Matter knowledge
D. Heuristic knowledge
E. Structural knowledge

A. Declarative Knowledge

- It is a segment of knowledge which stores factual information in a memory and it seems to be static in nature.
- These can be things or events or processes and the domain of such knowledge finds the relation between events or things.

B. Procedural Knowledge

- This knowledge is less general than declarative knowledge and is also known as imperative knowledge.
- It can have the potential to declare the accomplishment of a particular thing. It is generally used by modern mobile robots where they can be planned to attack into a building or perform navigation in a room.
- If we consider declarative knowledge implanted into a modern robot, it will be assigned just a map instead of detailed plan of attack into a building.

C. Meta Knowledge

- In the field of AI, the knowledge of pre-defined knowledge is known as meta-knowledge. A study of planning, tagging and learning are some of the examples of meta knowledge.
- This model tends to change with time and utilise a different specification. A knowledge engineer may utilise different forms of meta-knowledge given below :
- Accuracy, Applicability, Assessment, Consistency, Completeness, Disambiguation, Justification life span, Purpose, Source, Reliability.

D. Heuristic Knowledge

- This knowledge is also known as shallow knowledge and it follows the principle of thumb rule.
- It is very efficient in reasoning process as it solves the problems based on the records of past problems or the problems which are compiled by experts.
- It provides knowledge based on the experiences, it gathered during the past problems

Module 4

Scanned with CamScanner

E. Structural Knowledge

- This is the most basic knowledge used and applied in problem solving.
- It tries to find out a relationship between concepts and objects. It describes relationship between various concepts such as kind of, part of, and grouping of something.
- Let us describe a relationship of the knowledge along with a flowchart.

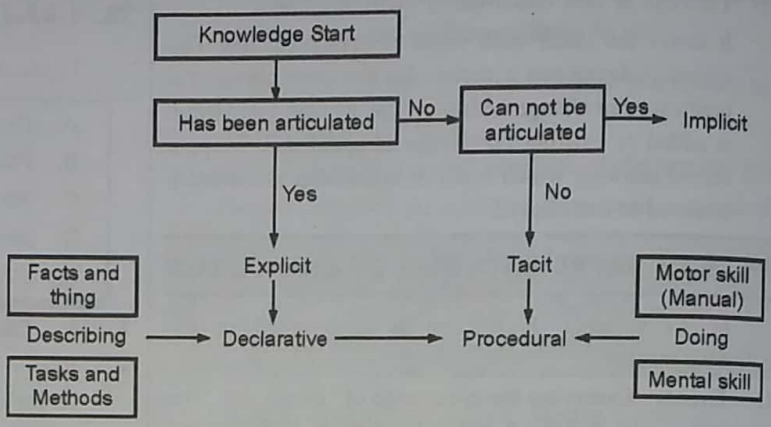


Fig. 4.22.1 : Relationship among types of knowledge

- Here, declarative knowledge is represented as **describing one** and procedural knowledge is represented as **doing one**.
- One more inference is, declarative knowledge is termed as explicit while procedural knowledge is termed as tacit.
- If the knowledge can be articulated, it is a declarative knowledge and if cannot be articulated, it is known as procedural knowledge.

4.22.3 Causes of Uncertainty

- 1. uncertainties relate to situations where it is impossible to exactly describe state of future outcomes.
- 2. In climate change adaptation, uncertainties arise from different sources, e.g. natural climate variability, future emissions, modelling, behavioural, socio-economic and technological responses and ecological dynamics.
- 3. Uncertainty can arise from multiple causes and situations; the lack of information or on the contrary the abundance of information with conflicting different pieces of information, measurement errors, linguistic ambiguity, or the subjectivity of opinions.
- 4. Reducing uncertainties is a major challenge in climate change adaptation. Some uncertainty can be reduced by acquiring knowledge while intrinsic uncertainty cannot be reduced.

4.23 THE SEMANTICS OF BELIEF NETWORK

UQ. Define Belief Network. Describe the steps of constructing belief network with an example. What types of inferences can be drawn from that ?

(MU - Q. 4(b), May 19, 10 Marks)

- A semantic network, or frame network is a knowledge base that represents semantic relations between concepts in a network. This is often used as an form of knowledge representation.
- It is a directed or undirected graph consisting of vertices, which represent concepts, and edges, which represent semantic relations between concepts, mapping or connecting semantic fields.
- A semantic network may be initiated as, for example a graph database or a concept map.

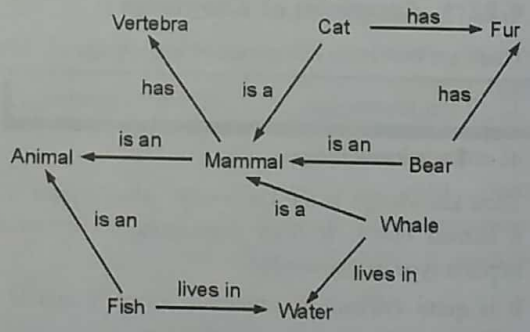


Fig. 4.23.1: Example of semantic network

Semantic networks are used in natural language processing applications such as, semantic parsing and word-sense disambiguation.

Scanned with CamScanner

4.23.1 Basics of Semantic Networks

- A semantic network is used when one has knowledge that is best understood as a set of concepts that are related to one another.
- Most semantic networks are cognitively based. They also consists of arcs and nodes which can be organised into a taxonomic hierarchy.
- Semantic networks contributed ideas of spreading activation, inheritance, nodes as proto-objects.

4.23.2 Examples of Semantic Network

1. WordNet

- An example of semantic network is WordNet. It groups English words into sets of synonyms called synsets, provides short, general definitions and records various semantic relations between these synonym sets.
- Some of the most common semantic relations defined are Meronymy (A is a meronym of B if A is a part of B), holonymy (B is a holonym of A if B contains A), hyponymy(Or troponymy) (A is subordinate of B, A is a kind of B), hypernymy (A is a subordinate of B), synonymy (A denotes the same as B), and antonymy (A denotes the opposite of B).
- In the social sciences people sometimes use the term semantic network to refer to co-occurrence networks.
- The basic idea is that words that occur in a unit of text, e.g. a sentence, are semantically related to one another. Ties based on co-occurrence can then be used to construct semantic networks.

4.23.3 Purpose of Semantics

- The purpose of semantics is to propose exact meanings of words and phrases, and remove confusion, which might lead the readers to believe a word has many possible meanings.
- It makes a relationship between a word and sentence through their meanings.

4.23.4 Focus on Semantics

- The study of semantics looks at how meaning works in language, and because of this it often uses native speaker intuitions about the meaning of words and phrases to base research on.
- We all understand semantics already on a subconscious level, it is how we understand each other when we speak.

4.24 INFERENCE IN BELIEF NETWORK

- An inference is an idea or conclusion that is drawn from evidence and reasoning. An inference is an educated guess.
- We **learn about** some things by experiencing them first hand, but we gain other knowledge by inference- the process of inferring things based on what is already known.
- Belief networks are popular tools for encoding uncertainty in expert systems.
- These networks rely on inference algorithms to compute beliefs in the context of observed evidence, network evidence, expert systems, join tree, probabilistic inference, probability propagation, reasoning under uncertainty.

Module 4

4.24.1 Purpose of Belief Network

- A belief network specifies a joint probability distribution from which arbitrary conditional probabilities can be derived.
- A network can be queried by asking for the conditional probability of any variables conditioned on the values of any other variables.

Scanned with CamScanner

4.24.2 Method of Belief Network

- A belief network is a directed model of conditional dependence among a set of random variables.
  - The precise statement of a conditional independence in a belief network takes into account the directionality.
  - To define a belief network, we begin with a set of random variables that represent all of the features of the model. Let the variables be $\{x_1, x_2,\ldots,x_n\}$ with the total ordering as $x_1,x_2,\ldots..,x_n$.
- We define the **parents** of random variable x_i, written parents (x_i), to be a minimal set of predecessors of x_i in the total ordering such that the other predecessors of x_i are conditionally independent of x_i.
- That is, parents $(x_i) \subseteq \{x_1, x_2, ..., x_{i-1}\}$ such that

  $P(x_i / x_{i-1}, ..., x_1) = P(x_i / \text{parents } (x_i))$.
- The probability over all of the variables, $P(x_1, x_2,\ldots..,x_n)$ is called the 'joint probability distribution'.
- A belief network defines a **factorization** of the joint probability distribution, where the conditional probabilities from factors that are multiplied together.

4.24.3 PPTC Algorithm

- Belief networks are popular tools for encoding uncertainty in expert systems.
- These networks rely on inference algorithms to compute beliefs in the context of observed evidence.
- One established method for exact inference on belief networks is the probability propagation in trees of clusters (PPTC) algorithm.
- PPTC converts belief network into a secondary structure, then computer probabilities by manipulating the secondary structure.

Chapter Ends...

❑❑❑

Scanned with CamScanner