UNIT IV

CHAPTER 4

Knowledge

Logical Agents, Knowledge-Based Agents, The Wumpus World, Logic, Propositional Logic: A Very Simple Logic, Propositional Theorem Proving, Effective Propositional Model Checking, Agents Based on Propositional Logic, First-Order Logic, Representation Revisited, Syntax and Semantics of First-Order Logic, Using First-Order Logic, Knowledge Engineering in First-Order Logic.

4.1 LOGICAL AGENTS

- The idea is that an agent can represent knowledge of its world, its goals and current situation by sentences in logic.
- It decides what to do by inferring that a certain action is appropriate to achieve its goals.
- There are **five logical connectives** used in Artificial Intelligence (A.I.) and they are : Conjunction, Negotiation, Implication, Disjunction and Biconditional.

4.1.1 Conjunction

In logic and mathematics, **And** is the truth-functional operator of logical conjunction; the **and** of a set of operands is true if and only if all of its operands are true.

The **logical connective** that represents this operator is typically written as $\wedge$.

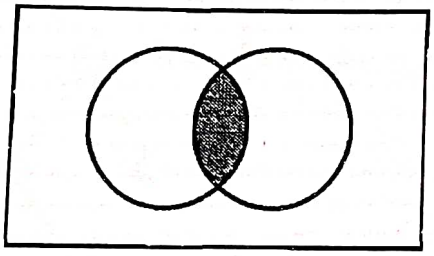


Fig. 4.1.1 : Logical conjuction ND

Logical conjunction **AND**

Venn diagaram : $A \wedge B$

$A \wedge B$ is true if and only if A is true and B is true.

An **operand** of a conjunction is **conjunct.**

Remarks

In other fields also the term 'conjunction' refers to similar concepts.

The truth table of $A \wedge B$.

A	B	$A \wedge B$
T	T	T
T	F	F
F	T	F
F	F	F

4.1.2 Disjunction

- Disjunctive conjunctions are **conjunctions** used to separate two or more mutually exclusive options presented in a sentence.
- When the connector between two statements is "or", you have a **disjunction.** In this case, only one statement in the compound statement needs to be true for the entire compound statement to be true.
- The disjunctive conjunctions most commonly used are **but, either, else, neither, nor,** or, **other,** and **otherwise.**
- Some disjunctive conjunctions are coordinating conjunctions - for example, the **either** and the **or** in **this sentence :**
- Poetry is usually either cheap or free.
- Here, either-or means it is cheap or free **but not both.**

Remark

- **Disjunctive conjunction** is also called as **disjunction.**
- The logical connective that represents disjunction is '$\vee$'.
- The truth table of $A \vee B$.

A	B	$A \vee B$
T	T	T
T	F	T
F	T	T
F	F	F

4.1.3 Negation

- In logic, 'negation' is also called as logical complement. It is an operation that takes a proposition **P** to another proposition **'not P'** written as $\neg P$, $\sim P$ or $\bar{P}$.

- It is true when P is False and False when P is true.

Truth Table

P	¬ P
T	F
F	T

4.1.4 Negotiation

- Successful negotiators prepare by determining their position along five dimensions.
- Legitimacy, Options, Goals, Independence and Commitment (LOGIC).
- Four most important elements of negotiation are :
- Strategy, Process, Tools and Tactics.
- Three basic styles of 'negotiation' are :

(i) Assertive (aggressive)

(ii) Accommodator (relationship oriented), and

(iii) Analyst (conflict avoidant)

4.1.4(A) Definition of Negotiation

- Negotiation is a dialogue between two parties to resolve conflicts or issues so that both parties find acceptable solution. Usually, it is a compromise involving give and take.
- Negotiation results when each party compromises to resolve a conflict for everyone's benefits.
- In the workplace, negotiations may take place between a team member and a manager.

Negotiation skills

- Negotiation skills are inherent qualities. They help two or more parties agree to a common logical solution.
- In the workplace, you may have to display your negotiating skills in various situation such as :
- Negotiating contract terms with a potential customer.
- Negotiation skills are soft skills and essential to become a negotiator and resolve workplace conflicts.
- But this skill set depends on the work environment, the parties involved and outcome desired.

4.1.4(B) Benefits of Negotiation Skills

(i) **Builds a relationship :** Despite the difference in opinion, negotiation skills help create goodwill and value. This builds a long-term relationship.

(ii) It helps to avoid future problems and conflicts.

(iii) It delivers lasting, quality solutions, rather than short-term solutions that do not satisfy the needs of either party.

4.1.5 Implication

Implication in logic is a relationship between two propositions in which the second is a logical consequence of the first.

4.2 WUMPUS WORLD

UQ. Explain WUMPUS world environment giving its PEAS description. Explain how percept sequence is generated ? (Q. 6(d), Dec. 15, 5 Marks, Q. 6(c), May 16, Q. 5(a), May 17, 10 Marks, Q. 6(A), Dec. 17, 5 Marks, Q. 4(a), May 18, 10 Marks)

Unit IV End Se

- The Wumpus world is an example to illustrate the worth of a knowledge-based agent and to represent knowledge representation. It was inspired by a video game 'Hunt the Wumpus' by Gregory Yob in 1973.
- The Wumpus world is a cave which has 4/4 rooms connected with passageways. So there are total 16 rooms which are connected with each other. We have a knowledge-based agent who will go forward in this world. The cave has a room with a beast which is called Wumpus who eats anyone who enters the room.

- The Wumpus can be shot by the agent, but the agent has a single arrow. In the Wumpns world, there are some pitts rooms which are bottomless and if agent falls in pits, then he will be stuck there forever.
- The exciting thing with this cave is that in one room there is a possibility of finding a heap of gold.
- The agent goal is to find the gold and climb out the cave without fallen into pits or eaten by wumpus.
- The agent will get a reward if he comes out with gold and he will get a penalty if eaten by Wumpus or falls in the pit.
- 'Wumpus is static and cannot move'.
- A sample diagram shows some rooms with pits, one room with wumpus and one agent at (1, 1) square location of the world.

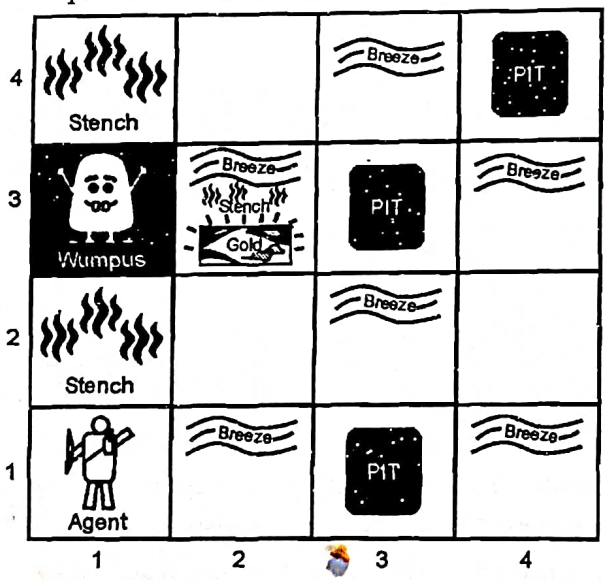


Fig. 4.2.1

- The following are some components which can help the agent to navigate the cave :

(a) The rooms adjacent to the Wumpus room are smelly, so that it would have some stench.

(b) The room adjacent to pits has a breeze, so if the agent reaches near to PIT, then he will perceive the breeze.

(c) There will be glitter in the room if and only if the room has gold.

(d) The Wumpus can be killed by the agent if the agent is facing to it, and Wumpus will emit a horrible scream which can be heard anywhere in the cave.

4.2.1 PEAS Description of Wumpus World

1. Performance Measure

(i) + 1000 reward points if the agent comes out of the cave with the gold.

(ii) – 1000 points penalty for being eaten by the Wumpus or falling into the pit.

(iii) – 1 for each action, and – 10 for using an arrow.

(iv) the game ends if either agent dies or came out of the cave.

2. Environment

(i) A 4×4 grid of rooms.

(ii) The agent initially in room square [1, 1], facing toward the right.

(iii) Location of Wumpus and gold are chosen randomly except the first square [1, 1].

(iv) Each square of the cave can be a pit with a probability 0.2 except the first square.

3. Actuators

(i) Left turn (ii) Right turn

(iii) Move forward (iv) Grab

(v) Release (vi) Shoot

4. Sensors

(i) The agent will perceive the 'stench' if he is in the room adjacent to the Wumpus. (Not diagonally).

(ii) The agent will perceive 'breeze' if he is in the room directly adjacent to the pit.

(iii) The agent will perceive the 'glitter' in the room where the gold is present.

(iv) The agent will perceive the 'bump' if he walks into a wall.

(v) When the Wumpus is shot, it emits a horrible 'scream' which can be perceived anywhere in the cave.

(vi) These percepts can be represented as five element list, in which we will have different indicators for each sensor.

(vii) If agent perceives stench, breeze, but no glitter, no bump, and no scream then it can be represented as [Stench, Breeze, None, None, None]

4.2.2 Properties of the Wumpus World

(i) **Partially Observable** : The agent can only perceive the close environment such as an adjacent room. Thus the Wumpus world is **partially observable.**

(ii) **Deterministic :** It is deterministic, as the result and outcome of the world are already known.

(iii) **Sequential :** The order is important, so it is sequential.

(iv) **Static :** It is static as Wumpus and Pits are not moving.

(v) **Discrete :** The environment is discrete.

(vi) **One agent :** The environment is a single agent as we have one agent only and Wumpus is not considered as an agent.

4.2.3 Exploring the Wumpus World

We explore the Wumpus world and we shall find how the agent finds its goal by applying logical reasoning.

Agent's first step

Initially, the agent is in the first room or as the square [1, 1] and we already know that this room is safe for the agent, so we add the symbol OK.

Symbol A is used to represent agent, symbol B for the breeze, G for Glitter or gold, V for the visited room, P for pits, W for Wumpus.

At room [1, 1] agent does not feel any breeze or any stench which means the adjacent squares are also O.K.

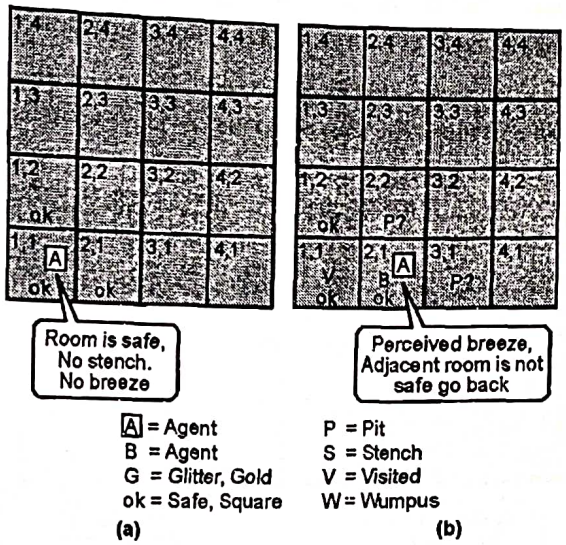


Fig. 4.2.2

Agent's second step

Now agent needs to move forward, so it will either move to [1, 2] or [2, 1]. Let us suppose agent moves to the room [2, 1], at this room agent perceives some breeze which means pit is around this room. The pit can be in [3, 1] or [2, 2], so we will add symbol (P ?) to say that, is this pit room ?

Now agent will stop and think and will not make any harmful move. The agent will go back to [1, 1] room. The rooms [1, 1] and [2, 1] are visited by the agent, so we use symbol V to represent the visited squares.

Agent's third step

At the third step, agent will move to the room [1, 2] which is O.K.

Unit IV End Sem

In the room [1, 2] agent perceives a stench which means there must be a Wumpus nearby. But Wumpus cannot be in the room [1, 1] as by rules of the game, and also not in [2, 2] (Agent had not detected any stench when he was at [2, 1]). Therefore agent infers that Wumpus is in the room [1, 3], and in the current state, there is no breeze which means in [2, 2] there is not pit and no Wumpus. So it is safe, and we will mark it OK, and agent moves further in [2, 2].

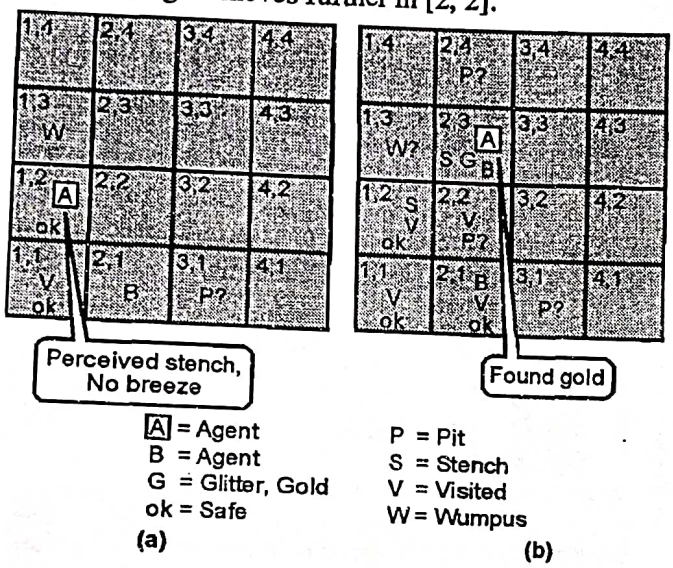


Fig. 4.2.3

☞ Agent's fourth step

At room [2, 2] no stench and no breeze present so let us suppose that agent decides to move to [2, 3]. At room [2, 3] agent perceives glitter, so it should grab the gold and climb out of the cave.

4.2.4 Knowledge-base for Wumpus World

Here we create a knowledge base for Wumpus-world; and derive some results for the Wumpus-world using propositional logic. We need symbol [i, j] for each location in the wumpus world.

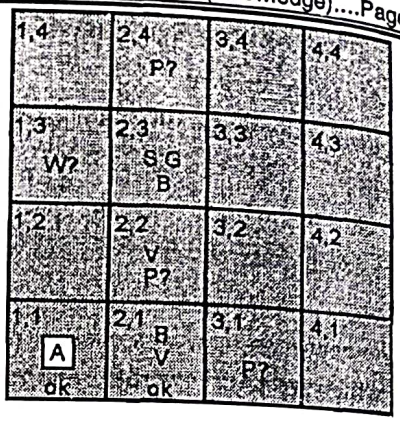


Fig. 4.2.4

Atomic proposition variable for Wumpus world :

(i) Let P_{ij} be true if there is a pit in the room [i, j].

(ii) Let B_{ij} be true if agent perceives breeze in [i, j], [dead or alive].

(iii) Let W_{ij} be true if there is Wumpus in the square [i, j].

(iv) Let S_{ij} be true if agent perceives stench in the square [i, j].

(v) Let V_{ij} be true if that square [i, j] is visited.

(vi) Let G_{ij} be true if there is gold (and glitter) in the square [i, j].

(vii) Let OK_{ij} be true if the room is safe.

4.2.5 Some Propositional Rules for the Wumpus World

$[R_1] \neg S_{11} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$

$[R_2] \neg S_{21} \rightarrow \neg W_{11} \wedge \neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31}$

$[R_3] \neg S_{12} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{22} \wedge \neg W_{13}$

$[R_4] S_{12} \rightarrow W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$

4.2.6 Representation of Knowledge base for Wumpus World

Here is the simple KB for Wumpus-world when an agent moves from room [1, 1] to room [2, 1].

$\neg W_{11}$	$\neg S_{11}$	$\neg P_{11}$	$\neg B_{11}$	$\neg G_{11}$	V_{11}	OK_{11}
$\neg W_{12}$	----	$\neg P_{12}$	-----	-----	$\neg V_{12}$	OK_{12}
$\neg W_{21}$	$\neg S_{21}$	$\neg P_{21}$	B_{21}	$\neg G_{21}$	V_{21}	OK_{21}

Here in the last row we have mentioned propositional variables for room [1, 1], which is showing that room does not have Wumpus ($\neg W_{11}$), no stench ($\neg S_{11}$), no pit ($\neg P_{11}$), no breeze ($\neg B_{11}$), no Gold ($\neg G_{11}$), visited (V_{11}), and the room is safe (OK_{11}).

4.3 PROPOSITIONAL LOGIC (FIRST ORDER LOGIC)

4.3.1 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

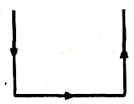


Fig. 4.3.1

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

Fig. 4.3.2

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.

Unit IV End Sem

4.3.2 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.

4.3.3 Propositions and Logical Operations

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

Illustrative Ex. 4.3.1 :

(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.

4.3.4 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.

4.4 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 4.4.1 : 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.

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 4.4.2 : 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 4.4.3 : 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

4.5 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.**

Unit IV End Sem

4.5.1 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.

4.5.2 Examples Based on the Proposition

Ex. 4.5.1

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

☑ **Soln. :**

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

(a)

=

p	q	$\neg (p \wedge \neg q)$
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. 4.5.2 : Find the truth-table of $\neg p \wedge q$.

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

p	q	$\neg p$	$\neg p \wedge q$
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 $\neg p$ using the negation $\neg$.

(iii) We find the truth value of $\neg p \wedge q$ using the conjunction $\wedge$.

Ex. 4.5.3

1. John likes all kinds of food.
   $\forall x$: food(x) $\rightarrow$ 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
   $\forall x$: ($\exists y$:eats(y, x) $\wedge \neg$ killed by (y, x)) $\rightarrow$ food(x)
5. Bill eats peanuts and is still alive.
   A. eats (Bill, peanuts) B. alive (Bill)
6. Sue eats everything Bill eats
   $\forall x$: eats(Bill, x) $\rightarrow$ eats (Sue, x)
7. $\forall x$: $\forall y$:alive(x) $\rightarrow \neg$ 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. $\forall x$: food (x) $\rightarrow$ likes (John, x)
2. food (apple) $\wedge$ food (chicken)
3. $\forall a, \forall b$: eats (a, b) $\wedge \neg$ killed (a) $\rightarrow$ food (b)
4. eats (Bill, peanuts) $\wedge \neg$ killed (Bill)
5. $\forall y$: eats (Bill, y) $\rightarrow$ eats (Sue, y)

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

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

Conclusion : Likes (John, Peanuts)

Step III : Resolution performance

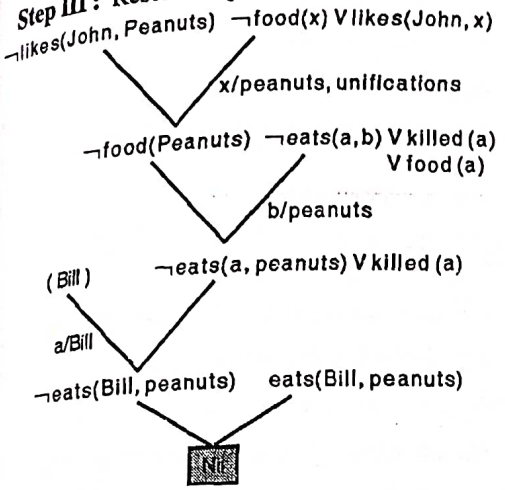


Fig. Ex. 4.5.3

4.6 EFFICIENT PROPOSITIONAL MODEL CHECKING

- Efficient model-checking inference algorithms for propositional logic includes backtracking and local search methods and can often solve large problems quickly.
- Efficient model checking inference algorithms for propositional logic includes backtracking and local search methods and can often solve large problems quickly.
- Families of algorithms for the SAT problem based on model checking :
  (a) based on backtracking.
  (b) based on local-hill-climbing search.

4.7 PROPOSITIONAL LOGIC (FIRST ORDER LOGIC)

4.7.1 Propositions and Logical Operations

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

Illustrative Ex. 4.7.1 :

(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.

Unit IV End Sem

4.7.2 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.

4.8 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 4.8.1 : 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 4.8.2 : Truth table of "p or not p is contradiction "

P	¬ p	p ∧ ¬ p
T	F	F
F	T	F

4.8.1 Examples on Tautology

Ex. 4.8.1 : 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.

4.8.2 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. 4.8.2 : 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.

4.9 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 $p \rightarrow q$.

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 $p \rightarrow q$ 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.

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

Table 4.9.1 : 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., $p \rightarrow q \equiv \neg p \vee q$.

4.9.1 Examples

Ex. 4.9.1 : 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

∵ p is true, q is true, ∴ $p \rightarrow q$ is true.

(ii) Let p = Bombay is in India,

$q = 4 + 5 = 3$

∵ p is true, but q is false ∴ $p \rightarrow q$ is false.

Ex. 4.9.2 : 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'.

∴ It is not hot or he wears a hat.

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

4.9.2 Conditional Statements and Variations

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

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

☞ **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.

Unit IV End

Table 4.9.2 : Truth table for conditional proposition

p	q	¬p	¬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 $p \rightarrow q$.

4.9.3 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.

4.9.4 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. 4.9.3 : 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.

4.9.5 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 4.9.3 : 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.

4.10 SYNTAX AND SEMANTICS OF FIRST ORDER LOGIC

There are two key parts of first order logic :

(i) The syntax determines which finite sequences of symbols are well formed expressions in the first order logic, and

(ii) the semantics determine the meaning behind these expressions.

(iii) The syntax of first order logic is defined relative to a signature. A signature σ consists of a set of constant symbols and a set of constant symbols and a set o predicate symbols.

We often refer to predicates as relation :

(iv) Basic elements of first order logic :

Constant : 1, 2, A, Ramesh, Pune, Cat

Connectives: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$

Equality : = =

Quantifier : $\forall, \exists$

(v) A predicate is an expression of one or more variables determined on some specific domain.

A predicate with variables can be made a proposition by either authorising a value to the variable or quantifying the variable.

☞ **Knowledge Engineering in First Order Logic (FOL)**

- The process of constructing a knowledge base in first-order logic is called as knowledge engineering.
- In **knowledge engineering**, someone investigates a particular domain, learns important concept of that domain, and then develop a formal representation of the objects. He is known as **knowledge engineer**.
- Here, we learn the knowledge engineering process in an electronic circuit domain.
- Following are some main steps of the knowledge-engineering process. Using these steps, we develop a knowledge-base and that helps us to reason about digital circuit (one-bit full adder) as shown in Fig. 4.10.1.

Unit IV End Sem

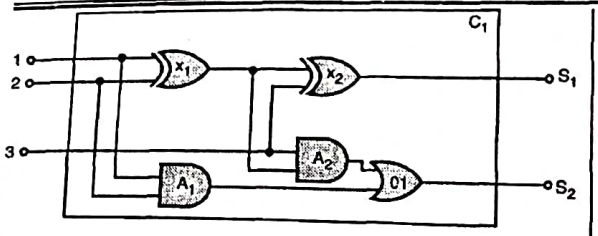


Fig. 4.10.1

- **Step 1 :** To identify the task :

We examine the functionality of the circuit.

(i) Does the circuit add properly ?

(ii) What will be the output of gate A_2, if all the inputs are high ?

At the second level, we will examine the circuit structure as :

(a) Which gate is connected to the first input terminal ?

(b) Does the circuit have feedback loops ?

- **Step 2 :** To assemble the relevant knowledge

For digital circuits, we require :

(i) Logic circuits are made up of wires and gates.

(ii) Signal flows through wires to the input terminal of the gate, and each gate produces the corresponding output which flows further.

(iii) Four types of gates are used in this logic circuit : AND, OR, XOR, and NOT.

(iv) All these gates have one output terminal and two input terminals (except NOT gate, it has one input terminal).

- **Step 3 :** The next step is to select functions, predicate, and constants to represent the circuits, terminals, signals and gates

First we distinguish the gates from each other and from other objects. Each gate is represented as an object and is named by a constant, such as, Gate (X_1).

The functionality of each gate is determined by its type, which is taken as constants such as AND, OR, XOR, or NOT. Circuits will be identified by a predicate : Circuit (C_1).

For the terminal, we use predicate : Terminal (X).

- **Step 4 :** Encode general knowledge about the domain

To encode the general knowledge about the logic circuit, we need the following rules :

(i) If two terminals are connected then they have the same input signal, it can be represented as :

$$\text{Signal}(t_1) = \text{Signal}(t_2)$$

(ii) Signal at every terminal will have either value 0 or 1.

(iii) Output of AND gate will be zero if and only if any of its input is zero.

(iv) Output of OR gate is 1 if and only if any of its input is 1 :

(v) Output of XOR gate is 1 if and only if its inputs are different.

(vi) Output of NOT gate is invert of its input :

(vii) All the gates in the above circuit have two inputs and one output (except NOT gate).

- **Step 5 :** Encode a description of the problem instance

For the given circuit C_1, we encode the problem instance in atomic sentences as below :

Since in the circuit there are two XOR, two AND, and one OR gate so atomic sentences for these gates will be :

For XOR gate : Type (X_1) = XOR, Type (X_2) = XOR

For AND gate : Type (A_1) = AND, Type (A_2) = AND

For OR gate : Type (O1) = OR

Then we represent the connections between all the gates.

Chapter Ends...

□□□

5.1 INFERENCE IN FIRST-ORDER LOGIC

Inference in first-order logic

- Inference means "carry forward". Inferences are steps in reasoning. These steps move from premises to logical consequences.
- Inference is divided into deduction and induction.
- Inductive inferences start with an observation and expand into a general conclusion (or theory).
- Inference in logic is derivation of conclusions form given information (called as premises) by any acceptable form of reasoning. Inferences are commonly drawn
  (1) by deduction. It analyses valid argument forms, and draws conclusions from the given premises,
  (2) by induction. It gives a general statement from many instances.
  (3) By using probability, which uses frequencies within a known domain to the conclusions (of given likelihood); and
  (4) using statistical reasoning, it concludes that a certain percentage of a set of entities will satisfy the stated conditions. Statistical inference uses quantitative or qualitative data which may be subject to random variables.

5.1.1 Different Inference Rules for FOPL

UQ. Explain different Inference Rules for FOPL. **(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.

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], it means we substitute a constant 'a' in place of variable 'x'.

Note : First-order logic is capable of expressing facts about some or all objects in the universe.

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 : It 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 :

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

This rule can be used if we want to show that element has a **similar** property.

Unit V End

Note that here x is not to be treated as a free variable.

Example : Let us write

p(c) : "A byte contains 8 bits", so for ∀ x p(x) "All bytes contain 8 bits." is also true.

(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 and we can 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 ∀x P(x) for any object under discussion.

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

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

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

In FOL form :

∀x king (x) ∧ greedy (x) → Evil (x)

King (Aurangjeb) ∧ Greedy (Aurangjab) → Evil (Aurangjab) King (Father (Taimur)) ∧ Greedy (Feather (Taimur)) → Evil (Father (Taimur))

(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 ∃ x P(x) for a new constant symbol c.

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

Consider the sentence

∃ x Crown (x) ∧ on head (x_1 John),

So, we can inter : Croun (k) ∧ on head (k, John) so far as k does not appear in the knowledge base.

The symbol 'k' is a constant symbol, and is called as **Skolem constant.**

The existential instantiation is a special case of skolemization process.

(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 $(\theta, p_i') = \text{subst}(\theta, p_i)$

It can be written as :

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

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) is king John is evil.

5.1.2 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. **(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.

Unit V Expl

Sr. No.	Parameters	Propositional Logic (PL)	Predicate Logic (FOL)
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.

5.1.3 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 ← 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.

5.1.4 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. BK301 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.

5.1.5 Refutation

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 (BK301)

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).

Unit V End Sem

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,BK30I).

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

5.1.6 Examples 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 maMmalH = 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)

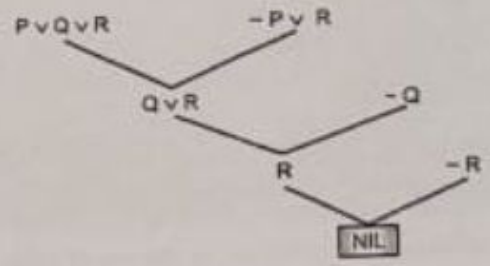


Fig. 5.1.1

Ex. 5.1.1 : 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. 5.1.2 : 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.

☑ **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) $M \wedge S$ (2) $M \rightarrow A$ (3) $A \wedge P \rightarrow C$

(4) $\neg S \vee P$ (5) $\neg H \wedge \neg Z$

Clausal form

1. MA
2. S
3. $\neg M \vee A$
4. $\neg (A \wedge P) \vee C = \neg \vee \neg P \vee C$
5. $\neg S \vee P$ 6. $\neg H$ 7. $\neg Z$

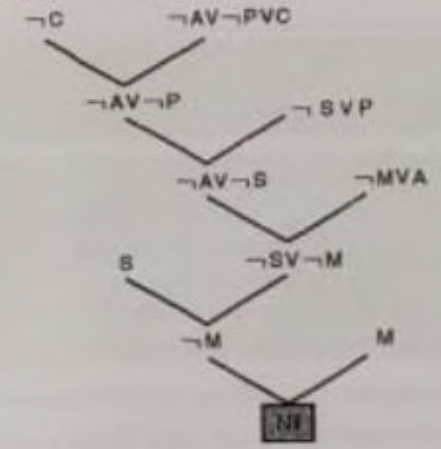


Fig. Ex. 5.1.1

Resolution proof for 'Alice will go to college'

Ex. 5.1.3 : 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. $\forall x$ course (x) $\wedge$ easy (x) $\rightarrow$ likes (Steeve, x)
2. Course (science) $\rightarrow \neg$ as y (science)
3. $\forall a$: course (a) $\wedge$ CSE (a) $\rightarrow$ easy (a)
4. CSE (DBMS)

▶ **Step II : Clausal form**

1. $\neg$ course (x) $\vee \neg$ easy (x) $\vee$ like (Steeve, x)
2. $\neg$ course (science) $\neg \vee \neg$ easy (science).
3. $\neg$ course (a) $\vee \neg$ CSE (a) $\vee$ easy (a)
4. CSE (DBMS).

▶ **Step III : Conclusion**

What course would Steeve like ?

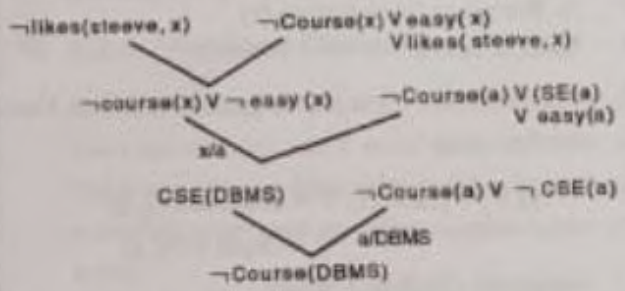


Fig. Ex. 5.1.3

∴ Steeve likes DBMS.

Ex. 5.1.4

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. $\forall x$: Pompiean (x) $\rightarrow$ Roman (x)
4. Ruler (Caesor)
5. $\forall a$: Roman (a) $\rightarrow$ Loyal to (a, Caesor) $\vee$ hate (a, Caesor)
6. $\forall p, \exists q$: Loyal to (p, q)
7. $\forall y, \forall z$: People (y) $\wedge$ ruler (z) $\wedge$ assassinate (y, z)
   $\rightarrow \neg$ loyal to (y, z).
8. Assassinate (Markus, Caesor).

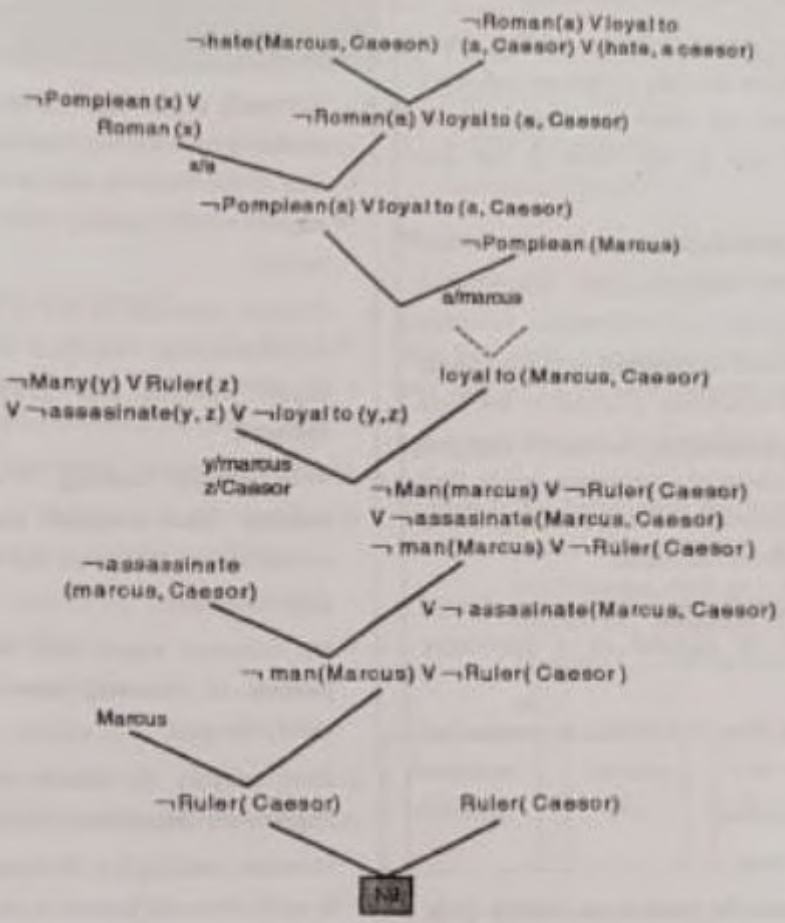
▸ **Step II : Clausal form**

1. Man (Markus)
2. Pompiean (Markus)
3. $\neg$ Pompiean (x) $\vee$ Roman (x)
4. Ruler (Caesor)
5. $\neg$ Roman (a) $\vee$ loyal to (a, Caesor) $\vee$ hate (a, Caesor)
6. loyal to (p, q)
7. $\neg$ (y) $\vee \neg$ Ruler (z) $\vee \neg$ assasinate (y, z)
   $\vee \rightarrow$ loyal to (y, z)
8. Assasinate (Markus, Caesor)

▸ **Step III : Conclusion**

hate (Markus, Caesor)

Resolution Tree



A resolution proof
Fig. Ex. 5.1.4

5.2 FORWARD CHAINING AND BACKWARD CHAINING

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

5.2.1 Forward Chaining

- 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.

Unit V End S

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.

5.2.2 Forward Reasoning

GQ. Explain forward and backward reasoning with examples

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

Or 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 more 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.

5.2.3 Modus Ponens

UQ. Explain modus ponens with suitable example. **(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 → 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".

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) ∧ weapon (q) ∧ sells (p, q, r) ∧ hostile (r) → criminal (p) ...(1)

(b) Country A has some missiles

? P Owns (A, p) ∧ 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) ∧ Owns (A, p) → Sells (Robert, p, A) ...(4)

(d) Missiles are weapons.

Missile (p) → Weapons (p) ...(5)

(e) Enemy of America is known as hostile.

Enemy (p, America) → 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 (∃x) ϕ(x), one may infer ϕ(c) for a new constant symbol c.

5.2.4 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)

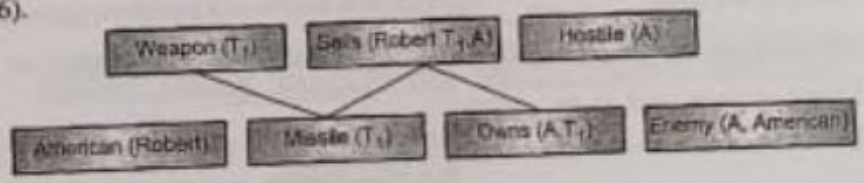
- **Step 2:** We see the facts which infer from available facts and with satisfied premises.

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.

5.2.5 Backward Chaining

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 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.
- 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.

5.2.6 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) ∧ weapon (a) ∧ sells (p, q, r) ∧ hostile (r) → criminal (D) ...(1)

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

(c) ? p Missiles (p) ∧ Owns (A, p) → Sells (Robert, p, A) ...(3)

(d) Missile (p) → Weapons (p) ...(4)

(e) Enemy (p, America) → Hostile (p) ...(5)

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

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

5.2.7 Backward Chaining Proof

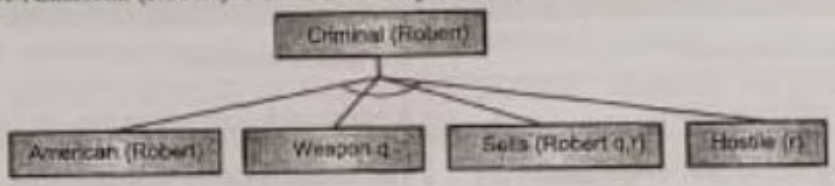
In Backward chaining, we begin with our goal predicate, which is criminal (Robert) and then infer further rules.

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

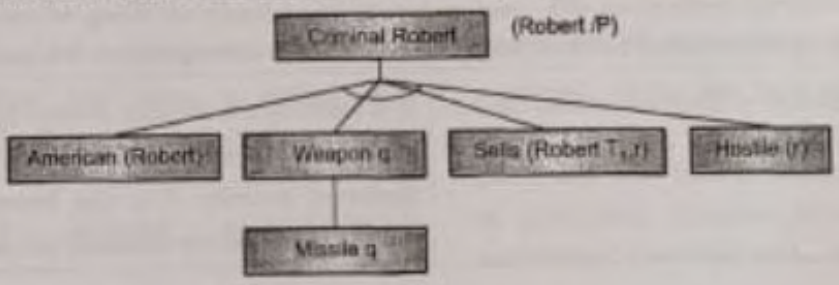
  So our goal fact is "Rober is Criminal", so following is the predicate of it.

- **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.

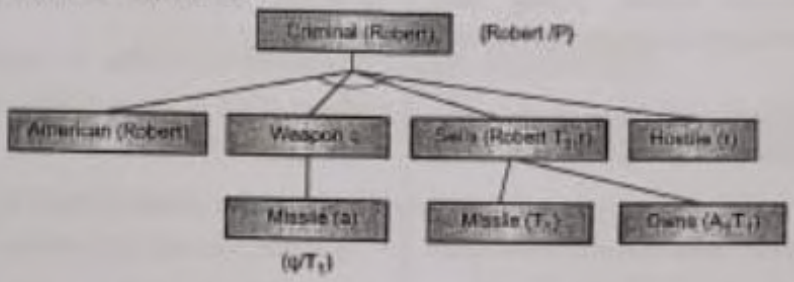


- **Step 3 :** At step 3, we extract further fact Missile (q) which infer from weapon (a) , as it satisfies Rule (5). Weapon (q) is also true with the substitution of a constant T_1 at q.



- **Step 4 :** Now we can infer facts Missile (T_2) 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.

5.2.8 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.

Sr. No.	Forward Reasoning	Backward reasoning
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.
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.

5.3 RESOLUTION

GQ. What do you mean by resolution and unification? Explain with example.
Or Write short note on resolution algorithm.

5.3.1 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.

5.3.2 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)

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,

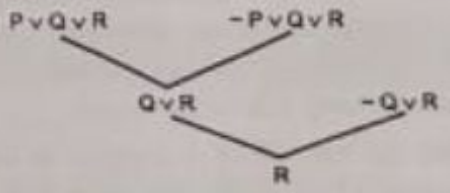


Fig. 5.3.1 : Deduction tree

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

5.3.3 Solved Examples

Ex. 5.3.1 : 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.

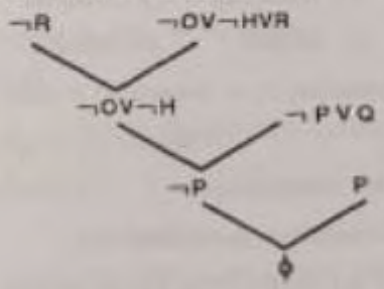


Fig. Ex. 5.3.1 : Resolution Tree

Since the end is empty.

∴ We conclude that it will rain.

Ex. 5.3.2 : 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

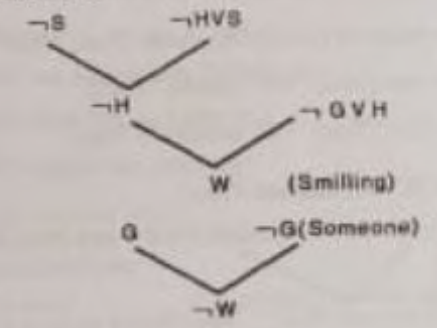


Fig. Ex. 5.3.2

∴ Someone is smiling.

Ex. 5.3.3 : 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 IV : Resolution Tree :**

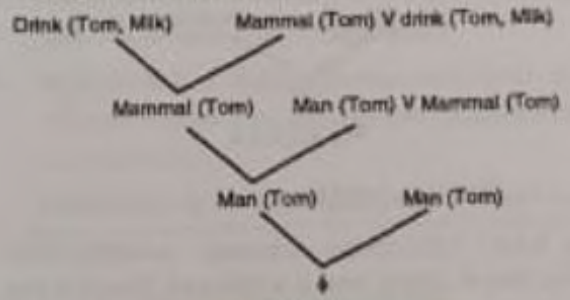


Fig. Ex. 5.3.3

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.

Ex. 5.3.4 : 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'.

Ex. 5.3.5 : 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)

(i) Let x stand for people :
(a) ∀x : graduating (x) → happy (x)
(b) ∀x : happy (x) → smile (x)
(c) Someone is graduating : ∃x : graduating (x)

▶ **Step II :**

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

Note that (x → y) is equivalent to (¬x ∨y)

∴ (a) ¬ graduating (x) ∨ happy (x)
(b) ¬ happy (x_1) ∨ 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. ¬ smile (x_3)

Resolution tree

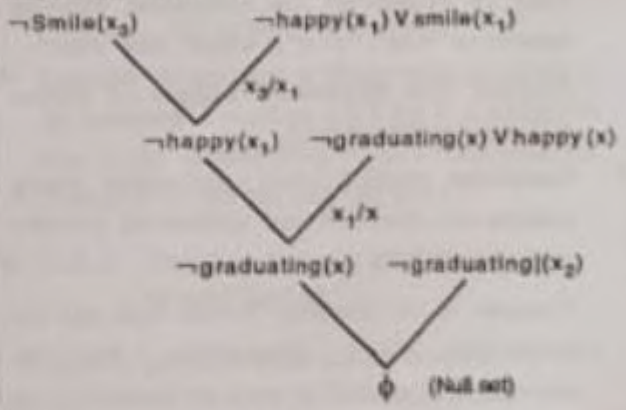


Fig. Ex. 5.3.5

∴ Our assumption is wrong.
∴ Someone is smiling

5.4 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**.

5.4.1 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.**

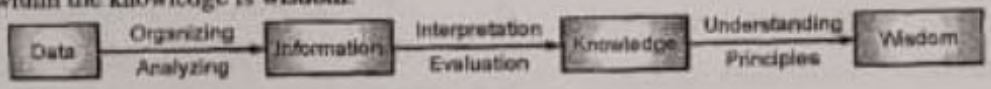


Fig. 5.4.1 : Knowledge Progression

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.

5.4.2 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.

5.4.3 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.

5.4.4 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.

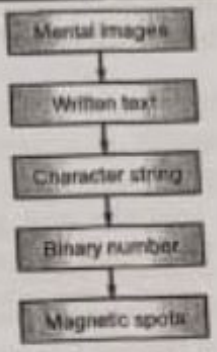


Fig. 5.4.2 : Different levels of knowledge representation

(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**.

5.4.5 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.

5.4.6 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. 5.4.3.

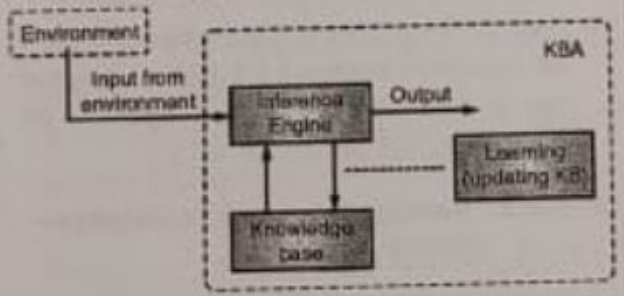


Fig. 5.4.3 : Architecture of knowledge-based agent

The Fig. 5.4.3 is representing a generalised architecture for a knowledge-based agent (KBA).

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.

5.4.7 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.

5.4.8 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.

5.4.9 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.

(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.

5.4.10 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.

5.4.11 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.

5.4.12 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.

Knowledge Representation Schemes

Various Knowledge Representation Schemes are as follows :

(i) Semantic networks

(ii) Frames

(iii) Conceptual dependency

(iv) Scripts

5.4.13 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.

5.4.14 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 ∃ : 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.

5.4.15 Difference between Procedural and Declarative Knowledge

Table 5.4.1

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.

5.5 PROPOSITIONAL LOGIC (FIRST ORDER LOGIC)

5.5.1 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

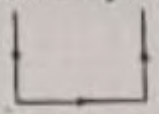


Fig. 5.5.1

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

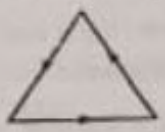


Fig. 5.5.2

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.

5.5.2 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.

5.5.3 Propositions and Logical Operations

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

Illustrative Ex. 5.5.1 :

(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.

5.5.4 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.

5.6 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 5.6.1 : 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.

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 ∨ q :

Table 5.6.2 : Truth table of p ∨ q

p	q	p ∨ q
T	T	T
T	F	T
F	T	T
F	F	F

(iii) Negation : (¬)

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

If p is true then ¬ p is false. If p is false then ¬ p is true.

☞ **Truth table for Negation**

Table 5.6.3 : Truth table for Negation

p	¬ p
T	F
F	T

☞ **Example based on Negation**

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

then ¬ **p** : Gopal is not good at sports.

☞ **Remark :** Negation is also denoted by ~. Thus if p is true, then ~ p is false

5.7 PROPOSITIONS AND TRUTH-TABLES

Proposition : A proposition is also called a well-formed formula of logical variables **p, q, r,** .... and logical connectives (∧,∨,¬) 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.**

5.7.1 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 ∧, ∨, ¬
- **Step (v) :** Finally, in the last column, we obtain the truth value of the proposition.

5.7.2 Examples Based on the Proposition

Ex. 5.7.1

(i) Find the truth-table of the proposition ¬ (p ∧ ¬ 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. 5.7.2 : Find the truth-table of $\neg p \wedge q$.

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

p	q	$\neg p$	$\neg p \wedge q$
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 $\neg p$ using the negation $\neg$

(iii) We find the truth value of $\neg p \wedge q$ using the conjunction $\wedge$.

Ex. 5.7.3

1. John likes all kinds of food.
   $\forall x$: food(x) → likes (john, x)
2. Apples are food.
   food(apple)
3. Chicken is food
   food(chicken)
4. Anything anyone eats are isn't killed by is food
   $\forall x$: ($\exists y$:eats(y, x) $\wedge \neg$ 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
   $\forall x$: eats(Bill, x) → eats (Sue, x)
7. $\forall x$: $\forall y$:alive(x) → $\neg$ killed by(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. $\forall x$: food (x) → likes (John, x)
2. food (apple) ∧ food (chicken)
3. $\forall a, \forall b$: eats (a, b) ∧ ¬ killed (a) → food (b)
4. eats (Bill, peanuts) ∧ ¬ killed (Bill)
5. $\forall y$: 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**

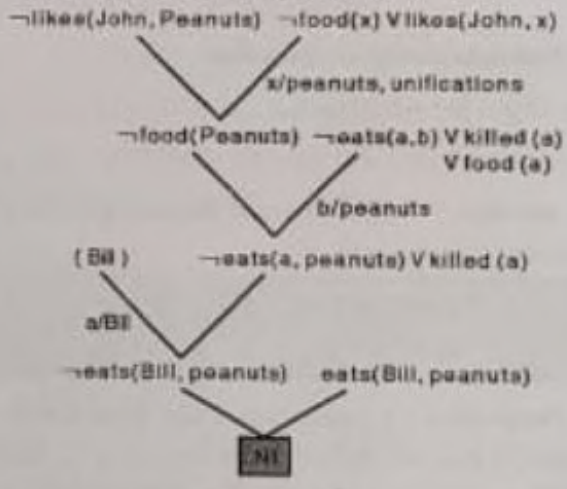


Fig. Ex. 5.7.3

5.8 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., $p \vee - p$ is tautology

Table 5.8.1 : Truth table of "p or not p is tautology"

P	$- p$	$p \vee - p$
T	F	T
F	T	T

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

Table 5.8.2 : Truth table of "p or not p is contradiction"

P	$- p$	$p \wedge - p$
T	F	F
F	T	F

5.8.1 Examples on Tautology

Ex. 5.8.1 : 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	$p \wedge q$	$\neg (p \wedge q)$	$p \vee \neg (p \wedge q)$
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.

5.8.2 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. 5.8.2 : 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.

5.9 ONTOLOGICAL ENGINEERING

- Ontology is a set of concepts and categories in a subject area or domain that shows their properties and the relations between them.
- Ontology is created automatically from large datasets.

- In brief, ontology is the science of the kinds and structures of objects. In simple terms, ontology seeks the classification and explanation of entities. Ontology concerns claims about the nature of being and existence.
- An example of ontology is 'when an engineer establishes different categories to divide existing things and how they fit together in the broader world.
- Another words for ontology are : cosmology, creation, perspective, position, view, viewpoint and underpinning.

5.9.1 Ontological Engineering

- In computer science, information science and systems engineering, ontology engineering is a field which studies the methods and methologies for building ontologies : It encompasses a representation, formal naming and definition of the categories, properties and relations between the concept, data and entities.
- In a broader sense, this field also includes a knowledge construction of the domain using formal ontology representations. A large scale representation of abstract concepts such as actions, time, physical objects and beliefs would be an example of ontological engineering.
- Ontology engineering is one of the areas of applied ontology. Core ideas and objectives of ontology engineering are also central in conceptual modelling.

Wikipedia : Ontology engineering

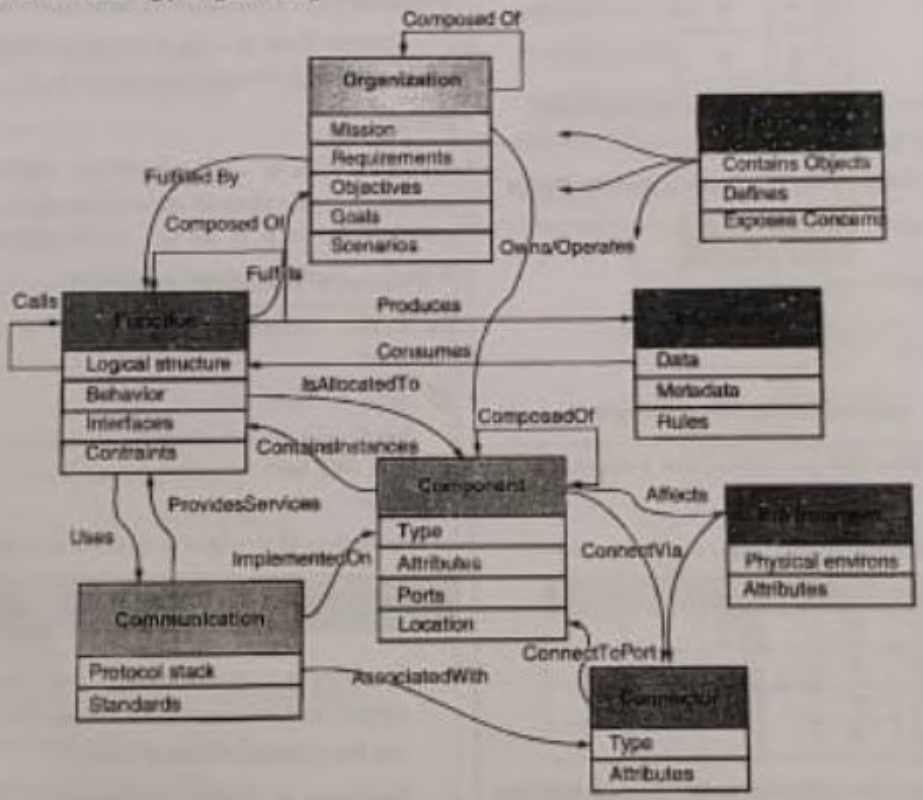


Fig. 5.9.1

Example of a constructed MBED top level ontology based on the normal set of views.

5.9.2 Use of Ontological Engineering

- Ontology's are now widely used in knowledge engineering, Artificial Intelligence and computer science, in applications related to knowledge management, natural language processing, e-commerce, intelligent integration information, information retrieval, integration of databases, b-informatics, and education.
- In computer and information science, ontology is a technical term denoting an artefact that is designed for a purpose, which is to enable the modelling of knowledge about some domain, real or imagined.
- In the context of computer and information sciences, an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse.
- The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). The definitions of the representational primitives include information about their meaning and constraints on their logically consistent applications.
- In the context of database systems, ontology can be viewed as a level of abstraction of data models, analogous to hierarchical and relational models, but intended for modelling knowledge about individuals, their attributes, and their relationships to other individuals.
- Ontologies are typically specified in languages that allows abstraction away from data structures and implementation strategies; in practice, the languages of ontologies are closer in expressive power to first-order logic than languages used to model databases.
- For this reason, ontologies are said to be at the "**semantic**" level, whereas database schema are models of data at the '**logical**' or '**physical**' level.
- Due to their independence from lower level data models, ontologies are used for integrating heterogeneous databases, enabling interoperability among disparate systems, and specifying interfaces to independent, knowledge-based services.
- In the technology stack of the semantic web standards, ontologies are called out as an explicit layer.
- There are now standard languages and a variety of commercial and open source tools for creating and working with ontologies.

5.9.3 Scientific Fundamentals

- Here we discuss ontology in the applied context of software and database engineering. An ontology specifies a vocabulary with which to make assertions, which may be inputs or outputs of knowledge agents (such as a software program). As an 'interface specification', the ontology provides a language for communicating with the agent. An agent supporting this interface is not required to use the terms of the ontology as an 'internal encoding' of its knowledge. Even then the definitions and formal constraints of the ontology do put
- Restrictions on what can be 'meaningfully' stated in this language. In essence, committing to an ontology (e.g. supporting an interface using the ontology's vocabulary) requires that statements that are asserted on inputs and output be 'logically consistent' with the definitions and constraints of the ontology.
- This is equivalent to the requirement that rows of a data base table (or insert statements in SQL) must be consistent with integrity constraints,

Unit V Unit 5

5.11 REASONING SYSTEMS FOR CATEGORIES : (XI)

Reasoning

Reasoning is the mental process of deriving logical conclusion and making predictions from available knowledge, facts and beliefs. Or we can say, " Reasoning is a way to infer facts from existing data." It is a general process of thinking rationally, to find valid conclusions.

In artificial intelligence, the reasoning is essential so that the machine can also think rationally as a human brain, and can perform like a human.

5.12 TYPES OF REASONING

In artificial intelligence, reasoning can be divided into the following categories :

1. Deductive reasoning.
2. Inductive reasoning.
3. Abductive reasoning.
4. Common sense Reasoning
5. Monotonic Reasoning
6. Non-monotonic reasoning

Remark

Inductive and deductive reasoning are the forms of propositional logic.

5.12.1 Deductive Reasoning

- Deductive reasoning is deducing new information from logically related known information. It is the form of valid reasoning, which means the argument's conclusion must be true when the premises are true.
- It is sometimes referred to as top-down reasoning, and contradictory to inductive reasoning.
- In deductive reasoning, the truth of the premises guarantees the truth of the conclusion.
- Deductive reasoning mostly starts from the general premises to the specific conclusion.

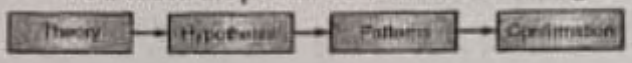
For example

Premises – 1 : All the human eats potatoes.

Premises – 2 : Ashok is human.

Conclusion : Ashok eats potatoes

We mention the process of deductive reasoning :

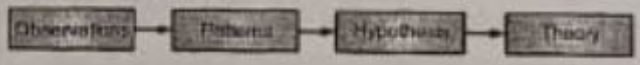


5.12.2 Inductive Reasoning

- Inductive reasoning is a form of reasoning to arrive at a conclusion using limited sets of facts by the process of generalisation, It starts with the series of Specific facts or data and reaches to a general statement or conclusion.
- It is also known as cause-effect reasoning or bottom-up reasoning
- In inductive reasoning, we use historical data or various premises to generate a generic rule, for which premises support the conclusion.
- In inductive reasoning, premises provide probable supports to the conclusion, so the truth of premises does not guarantee the truth of the conclusion.

Example : All of the pigeons we have seen in the zoo are white.

Conclusion : Therefore, we can expect all the pigeons to be white.



5.12.3 Abductive reasoning

- Abductive reasoning is a form of logical reasoning which starts with single or multiple observations then seeks to find the most likely explanation or conclusion for the observation.

- Abductive reasoning is an extension of deductive reasoning, but in abductive reasoning, premises do not guarantee the conclusion.

☞ **Example :**

Implication : Cricket ground is wet, if it is raining

Axiom : Cricket ground is wet.

Conclusion : It is raining.

5.12.4 Common Sense Reasoning

- Common sense reasoning is an informal form of reasoning, which can be gained through experiences.
- Common sense reasoning simulates the human ability to make presumptions about event which occurs on every day.
- It relies on good judgement rather than exact logic and operates on 'heuristic knowledge' and heuristic rules',

☞ **Example**

1. One person can be at one place at a time.
2. If I put my hand in a fire, then it will burn.

The above two statements are the examples of common sense reasoning which a

human mind can easily understand and assume.

5.12.5 Monotonic Reasoning

- In monotonic reasoning, once the conclusion is taken, then it will remain the same even if we add some other information to existing information in our knowledge base.
- In monotonic reasoning, adding knowledge does not decrease the set of prepositions that can be derived.
- To solve monotonic problems, we can derive the valid conclusion from the available facts only, and it will not be affected by new facts.
- Monotonic reasoning is used in conventional reasoning systems, and a logic –based system is monotonic.

☞ **Example**

- Earth revolves around the sun.
- It is a true fact, and it cannot be changed even if we add another sentence in knowledge base like,
- "The moon revolves around the earth" or "Earth is not round", etc.

☞ **Advantages of Monotonic Reasoning**

1. In monotonic reasoning, each old proof will always remain valid.
2. If we deduce some facts from available facts, then it will remain valid for always.

☞ **Disadvantages of monotonic Reasoning**

1. We cannot represent the real world problems using monotonic reasoning.
2. Hypothesis knowledge cannot be expressed with monotonic reasoning, which means facts should be ture.
3. Since we can only derive conclusions from the old proofs, so new knowledge from the real world cannot be added.

5.12.6 Non-monotonic Reasoning

In Non-monotonic reasoning, some conclusions may be invalidated if we add some more information to our knowledge base.

Non-monotonic reasoning deals with incomplete and uncertain models.

"Human perceptions for various things in daily life", is a general example of non-monotonic reasoning.

Example

Let us assume that the knowledge base contains the following knowledge :

Birds can fly.

Penguins cannot fly.

Pitty is a bird.

So from the above sentences, we conclude that "Pitty can fly". But if we add one another sentence into knowledge base "Pitty is a penguin," which concludes "Pitty cannot fly", so it invalidates the above conclusion.

Advantages of Non-monotonic reasoning

1. For real world systems such as Robot navigation, we can use non-monotonic reasoning.
2. In non-monotonic reasoning, we can choose Probabilistic facts or can make assumption.

Disadvantages of Non-monotonic Reasoning

1. In non-monotonic reasoning, the old facts may be invalidated by adding new sentences.
2. It cannot be used for theorem proving.

5.13 REASONING WITH DEFAULT INFORMATION

- Default reasoning is a 'form of defensible Reasoning' used to express facts like "by default, something is true".
- Default logic is a Non-monotonic logic proposed to formalise reasoning with default assumptions. Here 'Plausible conclusions are inferred based on general rules' which may have exceptions.
- Default logic can express facts like "by default, something is true", but by contrast, standard logic can only express that something is true or that something is false.
- A classical example is "birds typically fly". This rule can be expressed in standard logic either by "all bird fly", which is inconsistent because penguins do not fly, or by "all bird that are not penguins and .... fly", which requires exceptions to the rule that we specify.
- Default logic aims at formalising inference rules like this one without mentioning all their exceptions.

5.13.1 Syntax of Default Logic

A default theory is a pair (W, D), w is a set of logical formulas, called the background theory, that formalise the facts that are known and D is a set of default rules, each one being of the form :

Prerequisite : Justification 1, ... Justificat

Conclusion

According to this default, it we believe that prerequisite is true, and each of Justification, is consistent with our current beliefs, we believe that conclusion is true.

Example : the default rule "birds typically fly" is formalised as :

$$D = \left\{ \frac{\text{Bird (x) : Flies (x)}}{\text{Flies (x)}} \right\}$$

This rule means that, "If X is a bird, and it can be assumed that it flies, then we conclude that it flies.

Chapter Ends...

❑❑❑

UNIT VI

CHAPTER 6

Planning

Syllabus

Automated Planning, Classical Planning, Algorithms for Classical Planning, Heuristics for Planning, Hierarchical Planning, Planning and Acting in Nondeterministic Domains, Time, Schedules, and Resources, Analysis of Planning Approaches, Limits of AI, Ethics of AI, Future of AI, AI Components, AI Architectures.

6.1 AUTOMATED PLANNING

- Automated planning (or AI planning) is branch of artificial intelligence that concerns the realisation of strategies or action sequences, typically for execution by intelligent agents, autonomous robots and unmanned vehicles. Unlike classical control can classification problems, the solutions are complex and must be discovered and optimised in multidimensional space. Planning is also related to 'decision theory'.
- In known environments with available models, planning can be done **offline.** Solutions can be found and evaluated prior to execution. In dynamically unknown environments, the strategy often needs to be revised **online.** Models and policies must be adapted. Solutions usually resort to iterative trial and error processes commonly seen in artificial intelligence.
- These include dynamic programming, reinforcement learning and combinatorial optimisation. Languages used to describe planning and scheduling are often called 'action languages.'
- Given a description of the possible initial States of the world, a description of the desired goals, and a description of a set of possible actions, the planning problem is to synthesise a plan that is guaranteed (when applied to any of the initial states) to generate a state which contains the desired goals (such a state is called a goal state).
- The difficulty of planning depends on simplifying the employed assumptions. Several classes of planning problems can be identified depending on the properties the problems have in several dimensions.

(i) Are the actions deterministic or non-deterministic Are the probabilities for non-deterministic actions available.

(ii) For discrete state variables, do they have finite number of possible values ?

(iii) The number of initial states, finite or several.

(iv) Do actions have duration.

(v) Can several actions possible at a time ?

(vi) Is the objective of a plan to reach the desired goal state or to maximise the reward function.

(vii) The number of agents, one or many ?

(viii) Are the agents co-operative or selfish ?

Do all of the agents construct their own plans separately, or are the plans constructed centrally for al agents ?

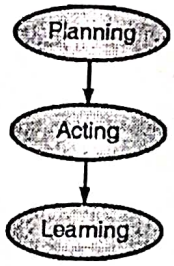
6.1.1 Classical Planning

Classical planning is the planning where an agent takes advantage of the problem structure to construct complex plans of an action. The agent performs three tasks in classical planning :

(i) **Planning :** The agent plans after knowing the problem in detail.

(ii) **Acting :** It decides the action to be taken.

(iii) **Leaning :** The agent learns new things from the actions taken by him.



A language known as PDDL (planning domain definition Language) which is used to represent all actions into one action schema.

PDLL describes the four basic things needed in a search problem :

(i) **Initial state :** It is the representation of each state as the conjunction of the ground and functionless atoms.

(ii) **Actins :** It is defined by a set of action schemas which define the 'Action' and 'Result' functions.

(iii) **Result :** it is obtained by the set of actions used by the agent.

(iv) **Goal :** It is a conjunction of literals, whose value is either positive or negative.

It is same as a precondition.

☞ **Examples**

(i) Air cargo transport
(ii) The spare tyre problem
(iii) The blocks world and many more.

▸ (i) **Air cargo transport**

We illustrate this problem with the help of the following actions :

(a) **Load :** This action is taken to load cargo.

(b) **Unload :** This action is taken to unload the cargo when it reaches its destination.

(c) **Fly :** This action is taken to fly from one place to another.

Hence, the air-cargo transport problem is based on loading and unloading the cargo and flying it from one place to another.

We present here an action schema for flying a plane from one location to another :

A set of ground (variable-free) actions can be written as a single 'action-schema'.

The schema is a 'litted' representation – it lifts the level of reasoning from propositional logic to a restricted subset of first-order logic.

For flying plane for one place to another : action schema is :

Action (fly (p. From, to)),

Precond : At (p, from) ∧ plave (P) ∧ Airport (from) ∧ Airport (to)

Effect : ¬ At (p, from) ∧ At (p, to).

▸ (ii) **The spare tyre problem**

The problem is that the agent needs to change the flat tyre. The aim is to place a good spare tyre over the cars axle. The actions follow.

(a) Remove the spare tyre from the trunk

(b) Remove the flat spare from the axle.

(c) Putting the spare on the axle.

(d) If the car is parked at an unsafe neighbourhood, leave the car unattended.

☞ **Advantage of classical planning**

(i) If has provided the facility to develop accurate domain- independent heuristics.

(ii) The systems are easy to understand and work efficiently.

6.1.2 Algorithms for Classical Planning

- Before we begin with planning classical algorithm, we discuss the complexity of classical planning.
- In theoretical complexity of problem, we distinguish two decision problems:
  1. Plan SAT is the question of whether there exists any plan that solves a planning problem
  2. Bounded plan SAT is whether there is a solution of length k or less that can be used to find an optimal plan.
- We observe that both decision problems are decidable for classical planning.
- Now we turn our attention to planning algorithms.

We see that the description of a planning problem defines a search problem :

UNIT VI End

i.e. We can also search **backward** from the goal, looking for the initial state.

From the figure below, we compare **forward** and **backward** searches.

(a) Forward (progression) search through the space of states, starting in the initial state and member of the set of goal states.

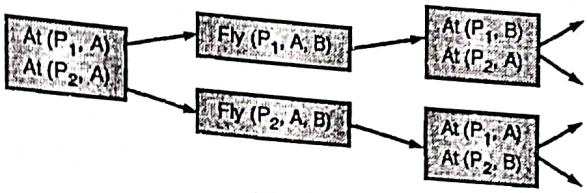


Fig. 6.1.1

(b) Backward (regression) search through sets of relevant states, starting from states representing the goal to search backward for the initial state using inverse of the actions.

Fig. 6.1.2

☞ Forward state-space search

Planning with forward state-space search is similar to the problem-solving approach. It is sometimes called progression planning, because it moves in the forward direction. We start with the problems initial state, considering sequences of actions until we reach a goal-state.

(i) The initial state of the search is set of positive ground literals, (literals not appearing are considered as false).

(ii) The actions which are applicable to a state are all those whose preconditions are satisfied.

The successor state is generated by adding the position effect literals and deleting the negative effect literals.

(iii) The goal test checks whether the state satisfies the goal of the planning problem.

(iv) The step cost of each action is generally 1.

(v) If function symbols are not present, the state space of a planning problem is finite then any graph search algorithm such as A* will be a complete planning algorithm.

☞ (b) Back-ward state-space search

(i) The main advantage of backward search is that it allows us to consider only relevant actions.

An action is relevant to a conjunctive goal if it achieves one of the conjuncts of the goal.

(ii) Consider for example, an air cargo problem with 10 airports, where each airport has 5 planes and 20 pieces of cargo.

The goal is to move all the cargo at airport A airport B. More precisely,

$At(C_1 B) \wedge At(C_2 B) ..... At(C_{20}, B)$

(iii) Let us consider the conjunct $At(C_1, B)$ working backwards, we can seek those actions which have this as an effect.

The relevant action UNLOAD $(C_1 \wedge P, B)$ achieves the first conjunct.

The action will work only if its preconditions are satisfied. Therefore, any predecessor state must include these preconditions :

$In(C_1, P) \wedge At(P_1 B)$ as subgoals.

Hence the predecessor description is :

$In(C_1, P) \wedge At(P, B) \wedge At(C_2, B) ....\wedge At(C_{20}, B).$

Termination occurs when a predecessor description is generated which is satisfied by the initial state of the planning problem.

For example :

$In(C_1, P_{12}) \wedge At(P_{12}, B) \wedge At(C_2, B) \wedge .....\wedge At(C_{20}, b)$

With substitution (P/P_{12}); We apply this substitution to the action leading from the state to the goal, producing the solution :

[Unload (C_1, P_{12}, B)]

6.2 HEURISTICS FOR PLANNING

Q. Write short notes on : (i) Planning agent (ii) State goal and action representation

6.2.1 Planning Agent

- The actual branching factor would be in the thousands or millions. The heuristic evaluation function can only choose states to determine which one is closer to the goal. It cannot eliminate actions from consideration.
- The agent makes guesses by considering actions and the evaluation function ranks those guesses. The agent picks the best guess, but then has no idea what to try next and therefore starts guessing again.
- It considers sequences of actions beginning from the initial state. The agent is forced to decide what to do in the initial state first, where possible choices are to go to any of the next places. Until the agent decides how to acquire the objects, it can't decide where to go. Planning emphasizes what is in operator and goal representations.

6.2.2 Three Key Ideas Behind Planning

There are Three Key Ideas behind Planning

- To "open up" the representations of state, goals, and operators so that a reasoner can more intelligently select actions when they are needed.
- The planner is free to add actions to the plan wherever they are needed, rather than in an incremental sequence starting at the initial state.
- Most parts of the world are independent of most other parts which makes it feasible to take a conjunctive goal and solve it with a divide-and-conquer strategy.

- An intelligent agent can act independently and has well-defined goals. It can adapt its behavior to its environment "a general-purpose system that, like a human, can perform a variety of different tasks under conditions that may not be known a priori." An agent must be aware of the user's goals and may have to behave in a changing world.

6.2.3 Two Types of Agents

1. Physical agents (usually known as robots)
2. Software agents (sometimes known as softbots)

1. Physical agents

These are physical artefacts and act in a physical environment, e.g. send a physical agent into a dangerous building. It must be able to see, it must know where it is, it must be able to move and search, plan its goals, execute its goals (physically), re-plan if necessary, communicate with other (possibly human) agents.

☞ **Consist of some or all of the following :**

(a) Computers

Top-level controller

Low-level controller, e.g. to manipulate a hand

(b) Sensors

Establish contact / non-contact with objects in the environment

To "see"

(c) Effectors

"arms", "hands", "feet"

(d) Auxiliary equipment

Tools, containers to put things in, tables to put things on

UNIT VI End S

Stages of development of physical agents :

- Slave manipulator operated by human master
- Limited sequence manipulator (hard to adjust)
- Teach-replay robot
- Computer-controlled robot
- Intelligent robot

Incentives for development of physical agents

- Job undesirability
- Lethal (radiation)
- Harmful (paint spraying, chemical handling)
- Risky (fire-fighting, combat)
- Strenuous (heavy loads, visual inspection)
- Noisy (riveting, hammering, forging)
- Boring (sorting, assembling)

2. Software agents

These are software programs and act in computers or computer networks.

Simple examples are programs that sort your incoming mail according to a given priority, that store documents in the correct folder (previously done by a human agent, a secretary).

☞ Three Laws of Robotics

- A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- A robot must obey orders given it by human beings except where such orders would conflict with the First Law.
- A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.
- A robot may not harm humanity, or, by inaction, allow humanity to come to harm.

☞ Agents and goals

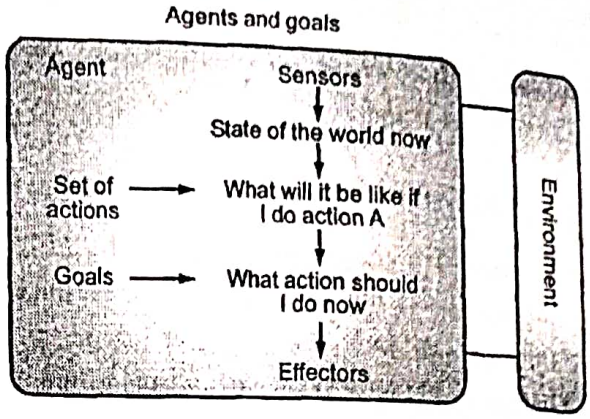


Fig. 6.2.1 : Software agents

(i) **Machine translation (MT)** is the task of automatically converting one natural language into another, preserving the meaning of the input text, and producing fluent text in the output language.

(ii) **State, goal and action representation :** Describe state, goal and action represents to plan, one should be able to represent the problem properly. Representation of the planning problem is **mapping of the states, actions and the goals.** For the representation, the language used should be concrete, understandable and expressive. In a broader perspective, there are different representation methods or ways that are followed like propositional, first order, state variable. Fig. 6.2.2 depicts the high-level diagram for planning.

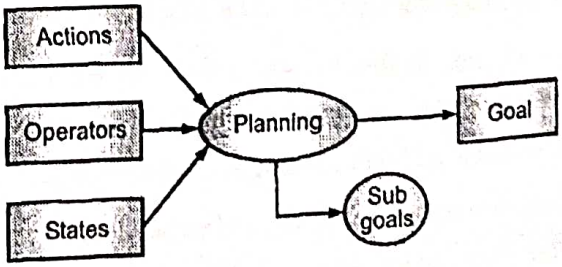


Fig. 6.2.2 : Planning

Planning essentially needs the representation in terms of **time**. This is required so that we are able to reason regarding the actions that are to be taken along with the reactions that we get back.

(iii) **State Representation** : States are the representation of the facts. States are represented as conjunction. It comprises the positive literals that specify the state.

A state is represented with a conjunction of positive literals using :

Logical Propositions: Poor ∧ Unknown

FOL literals: At (Plan1,OMA) ∧ At (Plan2,JFK)

FOL literals must be ground & function-free

Not allowed: At (x, y) or At (Father(Fred),Sydney)

Closed World Assumption :

What is not stated are assumed false.

6.3 HIERARCHICAL PLANNING

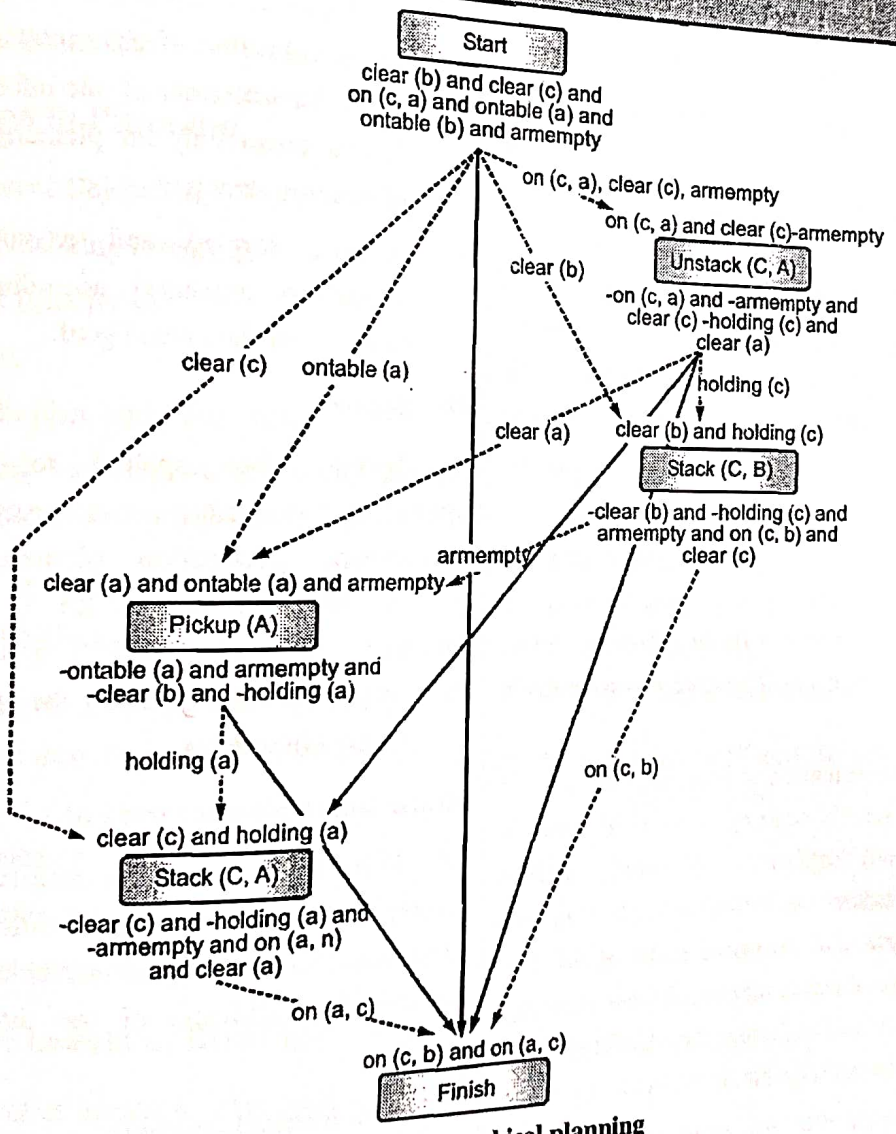


Fig. 6.3.1 : Hierarchical planning

- Take an unachieved precondition from the plan; achieve it. Resolve any threats using promotion or demotion. But, what the above fails to show is that planning involves search.
- At certain points in the algorithm, the planner will be faced with choices (alternative ways of refining the current unfinished plan). POP must try one of them but have the option of returning to explore the others.

☞ **There are basically two main 'choice points' in the algorithm**

- In **goal achievement**, a condition c might be achievable by any one of a number of new steps and/or existing steps. For each way of achieving c, a new version of the plan must be created and placed on the agenda.
- **Question :** A condition c might be achievable by new steps or existing steps. When placing these alternatives on the agenda, might be we arrange for the latter to come off the agenda before the former?
- When resolving threats, POP must choose between demotion and promotion. All preconditions must eventually be achieved, and so these aren't alternatives. The choice can be made irrevocably.
- Provided implementation of POP uses a complete and optimal search strategy, then POP itself is complete and optimal. However, **POP's branching factor** can still be high and the unfinished plans that we store on the agenda can be quite large data structures, so we typically abandon completeness/optimality to keep time and space more manageable.
- Search strategies that are more like depth-first search might be preferable. And we might use heuristics to order alternatives or even to prune the agenda.

6.4 HIERARCHICAL TASK NETWORK PLANNING

- In an Artificial Intelligence hierarchical task network planning is an approach to **automated planning** in which the dependency among actions can be given in the form of hierarchically structured networks.
- A hierarchical planning represents the **organisational levels** and units in the company for which one wants to plan. A planning hierarchy is a **combination of characteristic values** based on the characteristics of one information structure. You can create only one planning hierarchy for an information structure.
- Planning formed and executed by several subsystems organised according to a layered structure for the system goal.

☞ **System**

System when applied to production and operations management, the process is called a **hierarchical production planning and control** process, which is the essence of a manufacturing planning and control (**MPC**) system. Also, these decisions are usually made by the upper levels of the management hierarchy.

Hierarchical plan consists of :

There are 3 types of hierarchical plans. They are strategic, administrative and operating (technical core) .The three hierarchical plans are independent, and they support the fulfilment of the three organisational needs.

☞ **Hierarchical strategy**

The hierarchy of strategies describes a **layout and relations of corporate strategy** and **sub strategies** of the organisation. Individual strategies are arranged hierarchically and logically

consistent at the level of vision, mission, goals and metrics.

6.4.1 Steps to Create Planning Hierarchy

1. From the flexible planning menu, choose master data.
2. Leave all fields blank and choose 'confirm'
3. Choose 'save'
4. Choose: Change planning Hierarchy to create the hierarchy.
5. To continue, choose Enter

6.4.2 Steps in Planning

1. Determination of Objectives,
2. Constructing Planning Premises,
3. Evaluation of Alternatives.
4. Selecting plan,
5. Controlling the plan, and a few others.

Every business has its own problems, and so planning details differ from business to business.

6.4.3 Key Steps in Hierarchy of Strategies

Strategic objectives and Analysis. The first step is to define the vision, mission and values-statements of the organisation :

1. Strategic formulation.
2. Strategic implementation
3. Strategic Evaluation and control.

6.4.4 Three Levels of Strategy are

1. **Corporate level of strategy**. This level answers the foundational question of what you want to achieve.
2. **Business unit level strategy**. This level focuses on how you are going to compete.
3. **Market level strategy**. This strategy focuses on how you are going to grow.

6.4.5 Strategy Intent

- Strategy intent refers to the **purpose** for which the organisation strives for.
- It is philosophical framework of strategic management process. The hierarchy of strategic intent covers the **vision and mission**, business definition and the goals and objectives.

6.4.6 Hierarchical Goals in Strategic Management

- The hierarchy of strategic intent covers the vision and mission, business definition and the goals and objectives.
- **Stretch** is misfit between resources and aspirations. Leverage stretches the meagre resource base to meet the aspirations.
- All three steps in strategic planning occur within three hierarchical levels; upper management, middle management and operational levels. Thus, it is imperative to foster communication and interaction among employees and managers at all levels. So as to help the firm to operate as a more functional and effective team.

6.4.7 Hierarchical Arrangement of Objective

The hierarchy of objectives indicates that **managers at different levels** in the hierarchy of the organisation are concerned with different kinds of objectives according to the authority they are delegated with **Middle level managers** are involved in the setting of key result area objective and division objective.

Hierarchy of goals

The hierarchy of goals is a **goal-oriented iterative method** that can be used to define the scope of the given project. Thus, by setting goals that can be brought up to date as knowledge expands. The method consists of a hierarchy with a main goal on top, followed by sub-goals, project goals, deliveries and success criterion.

Hierarchy of objectives

The hierarchy of objectives is a **tool** that helps analyse and communicate a **project's objectives**. The hierarchy of objective organises the objectives of a project into different levels of hierarchy or tree.

6.4.8 Planning and Acting in Nondeterministic Domains

- It presents 'agent architectures' that can handle uncertain environments and disconnect deliberation with execution, and gives some example of real-world systems.
- Here, we extend planning to handle partially observable, nondeterministic, and unknown environments.
- Methods of search are as follows :
  - (I) Sensor less planning or conformant planning for environments with no observations.
  - (II) Contingence planning for partially observable and nondeterministic environments.
  - (III) Online planning and replanning for unknown environments.

▶ **(I) Sensor less planning :** Sensor less planning handles domains where the state of the world is not fully known.

It comes up with plans that work in all possible cases. While executing the plan, before performing each action, environment must be monitored.

▶ **(II) contingency planning :** We mention below five basic steps of contingency planning for epidemic, or other emergency situations.

1. Program Management

Here the team members identify the objectives of the contingency plan for each department and then the team plans out the risk factor for responding every potential threat. Some Teams are formed to deal with particular issues during an emergence such as information technology, vendors and supplies.

2. Planning

- The aim of the planning team is to conduct a thorough, realistic risk assessment and its impact on business policies.
- The risk assessment is the basis of the business policies, planning team has to consider prevention of risk hazards.

3. Implementation

- The contingency plan works out who gets notified, both internally and externally and in what order.
- Contingency plans dire its the organisation through each of the natural phases of the event and its goals. E.g. response, recovery and restoration.

4. Testing and Exercise

There must be enough storage of raw materials and resources in case of emergence outbreak. The cost of such contingency planning is comparatively less as compared to loss incurred due to delay or inability to react in time.

5. Program Improvement

- Companies will have to have good business sense to develop long-term support plans in case of epidemics, (just for example recent corona epidemics), Such epidemics may last for months, or even years.
- Outbreaks may last for months or more, so managing stock components is crucial.
- It is better for the good health and good reputation for the company to practice a disaster plan at least once a year. Each practice will give opportunities to learn and help to refine the contingency plans.

(III) Online planning and replanning for unknown environments

- Online planning is useful in major applications, like field monitoring and search and rescue. Planning informative paths online is necessary for robot autonomy.
- Sampling based approaches are capable of using informative formulation and hence they are used in online planning.

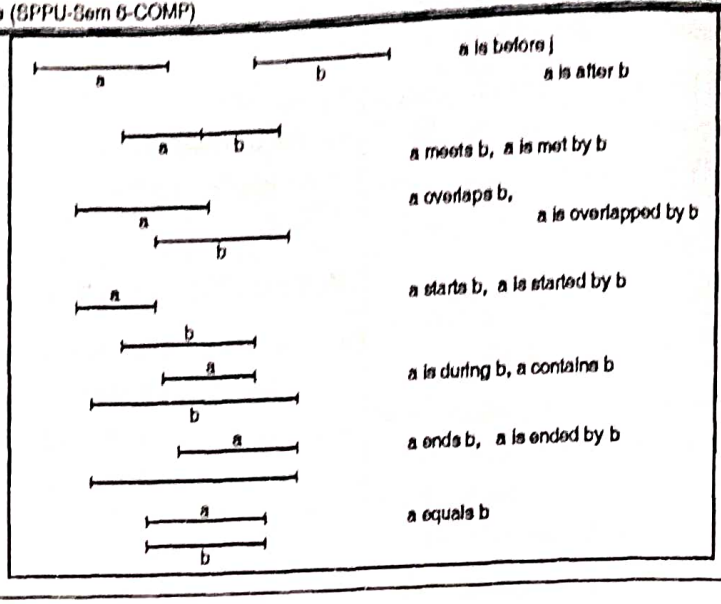
(IV) Sensorless planning

- Sensorless planning is also called as conformant planning.
- It gives out plans which work in all possible cases.
- It can handle the domain where the state of the world is not fully known.
- Sensorless planning is based on any perception. The algorithm gives that plan which reaches its goal.

6.4.9 Time, Schedules and Resources

- Here we discuss planning and talk about actions with durations and resource constraints.
- Classical planning representation talks about the order in which given task is to be carried out. But is cannot talk about time. how long an action takes and when it occurs.
- The most basic notion of **time** is that it is occupied by **events**. These events occur during intervals, continuous spaces of time.
- An interval has a starting point and an ending point, and a duration defined by these points. Intervals can be related to other intervals.
- It turns out that there are exactly thirteen ways in which two non-empty time intervals can related to one another.
- We exhibit the relationships. We note that there are actually only seven distinct relationships, the relationship of **equality** plus size other relationships that have their inverses :

NOTES



Now, we state rules for drawing inferences about time intervals.

For example, 'IS-BEFORE' relation is transitive. That is, it event 'a' occurred before event 'b' and if event 'b' occurred before C, then event a must have occurred before C.

We can reduce all the relations in the Fig. (i) to the single relation 'MEETS'.

1. The definition of the transitive relation 'ISBEFORE' is given as :

   a IS-BEFORE b' = $\exists K : (a \text{ meets } k) \wedge (k \text{ meets } b)$

   That is, it a IS-BEFORE b, then there must be k in between that meets a and b.

2. The first relation states that the points where intervals 'meet' are unique.

   We write.

   $\forall a, b : (\exists K (a \text{ meets } K) \wedge (b \text{ meets } k)) \rightarrow$

   $(\forall p : (a \text{ meets } p) \leftrightarrow (b \text{ meets } p))$.

   This implies that a and b cannot meet k at different point in time, i.e. every event has a unique starting time. or every event has unique ending time.

3. Given two places where intervals meet, exactly one of the following three conditions must hold;

   $\forall a, b, k, p : (a \text{ meets } b) \wedge (k \text{ meets } p) \rightarrow (a \text{ meets } p) \oplus$

   $\exists m : (a \text{ meets } m) \wedge (m \text{ meets } p) \oplus$

   $\exists m : (k \text{ meets } m) \wedge (m \text{ meets } b)$

Remarks

The notation '$\oplus$' stands for "exclusive - or". $a \oplus b \oplus c$ is logically equivalent to

$(a \wedge \neg b \wedge \neg c) \vee (\neg a \wedge b \wedge \neg c)$.

4. If states that there are always intervals surrounding any given interval.

   $\forall a : \exists b, k : (b \text{ meets } a) \wedge (a \text{ meets } k)$.

5. For two intervals that meet :

   $\forall a, b : (a \text{ meets } b) \rightarrow \exists m, n, (a + b) :$

   $(m \text{ meets } a) \wedge (b \text{ meets } n)$

   $\wedge (m \text{ meets } (a + b)) \wedge ((m + n) \text{ meets } n)$.

Scheduling is sometimes called as AI planning. It is a branch of artificial intelligence and is concerned with the realisation of action sequences for execution by intelligent agents and also by autonomous robots.

In known environments with available models, planning can be done offline. Solutions can be evaluated prior to its execution

In unknown environments, the strategy often needs to be revised online. Models and schedules must be adapted. Solutions usually resort to **iterative trial and error** processes. These are commonly seen in artificial intelligence.

Explicit time and resources into planning focuses on causal relations, between actions of the problems. That is an important step toward forwards modelling real-life problems.

Given a description of the possible initial states of the world. A description of the desired goals, and a description of a set of possible actions, the scheduling problem is to develop a plan, that will generate a desired goal.

Such a state is called a goal state.

The difficulty of scheduling is dependent on the properties of the problems in several dimension. We not them.

1. Are the actions deterministic or non-deterministic? For nondeterministic actions, are the associated probabilities available ?
2. Are the state variable discrete or continuous ?

   If discrete, what are the possible values ?
3. Number of initial states present?
4. What is the duration of actions ?
5. What is the objective of a plan?

Is it to reach a goal or to maximise a reward function ?

The approach we take here in scheduling is "plan first, schedule later", first we divide the whole problem into a planning phase in which actions are selected, and ordering constraints of the plan are imposed. Then we consider a 'scheduling phase'. In **scheduling phase** we add **temporal information** to the plan to check whether it meets resource and deadline constraints.

Let us consider and example.

To produce a schedule for an airline that says which planes are assigned to which flights.

At the same time we must know **departure** and **arrival times** also. This is two subject matter of scheduling.

There are also **resource constraints :**

For example, an airline has a limited number of staff-and staff who are on one flight cannot be on another at the same time.

☞ Representing resource constraints

A typical job-shop scheduling problem consists of a set of jobs and each job consist of a collection of actions with ordering constraints among them.

Each action has a **duration** and a set of resource constraints required by the action.

Each constraint specifies a type of resource (e.g. bolts, wrenches, or pilots), and the number of that resource required and whether that resource is consumable or reusable.

A solution to a job-shop scheduling problem must mention the **start times** for each action and should satisfy resource constraints.

The representation of resource as numerical quantities is a general technique called 'aggregation'. In aggregation individual objects are grouped into quantities when the objects are all indistinguishable with respect to the purpose at hand. Aggregation is essential for reducing complexity.

Consider e.g. when a proposed schedule has 10 concurrent **Inspect actions** but only 9 inspectors are available. When inspectors are represented as quantities, a failure is detected immediately.

But when inspectors are represented as individuals, the algorithm has to try all (101)

UNIT VII End S

Ways of assigning inspectors to actions.

☞ **Solving scheduling problems**

To minimise plan duration, we find the start times for all actions. These **'times'** must be consistent with the ordering constraints along with the problems. These ordering constraints can be checked with the **directed graph** relating the actions. We apply **Critical Path Method** (CPM) to this graph to determine the possible start and end times of each action.

The critical path determines the duration of the entire plan.

Shortening other paths does not shorten the plan as a whole. But if the start of any action on the critical path is delayed, it slows down the whole plan.

Actions that are off the critical path have a window of time (initial start) in which they can be executed.

The window is specified in terms of **earliest start time (ES)** and latest start time (LS).

The quantity (LS-ES) is known as the slack of an action.

'ES and LS times' together for all the actions constitute a schedule for the problem.

To compute ES and LS, mention the 'dynamic-programming algorithm'.

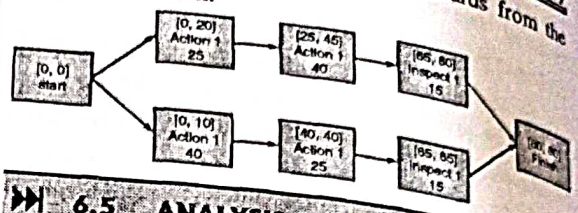
If X and Y are actions, and then X < Y means X comes before is.

(i) We begin with ES (start) = 0

(ii) All the actions from 'start' to action 'Y', the 'ES' value are assigned.

(iii) Process is repeated till every action has been assigned ES valve.

(iv) 'LS' valves are calculated, backwards from the **finish** action.



6.5 ANALYSIS OF PLANNING APPROACHES

GQ. Explain the different components of planning system. **Or** Given the components of planning system and briefly explain them. **Or** Explain the various components of planning system. How can you represent a planning action?

Components of planning system

1. Choose best rule to apply
2. Apply the chosen rule
3. Detecting when a solution has been found
4. Detecting dead ends
5. Repairing an almost correct solution

1. **Choose best rule :** Isolate set of differences between the desired goal state and the current state Identify those rules that are relevant to reducing those differences (means-end analysis). If several rules found then choose best using heuristic information.

2. **Apply rules :** In simple systems, applying rules is easy. Each rule simply specified the problem state that would result from its application. In complex systems, we must be able to deal with rules that specify only a small part of the complete problem state. One way is to describe, for each action, each of the changes it makes to the state description.

3. **Detecting a solution :** A planning system has succeeded in finding a solution to a problem when it has found a sequence of operators that

transform the initial problem state into the goal state. In simple problem-solving systems we know the solution by a straightforward match of the state description. But in complex problem different reasoning mechanisms can be used to describe the problem states, that reasoning mechanisms could be used to discover when a solution had been found.

4. **Detecting dead ends :** A planning system must be able to detect when it is exploring a path that can never lead to a solution. Above same reasoning mechanism can be used to detect dead ends. In search process reasoning forward from initial state, it can prune any path that lead to a state from which goal state cannot be reached. Similarly backward reasoning, some states can be pruned from the search space.

5. **Repairing an almost correct solution :** In completely decomposable problems can be solve the sub problems and combine the sub solutions yield a solution to the original problem. But try to solving nearly decomposable problems one way is use means-ends analysis technique to minimize the difference between initial states to goal state. One of the better way to represent knowledge about what went wrong and then apply a direct patch.

GQ. What is planning?

Planning : Planning can be viewed as a type of problem solving in which the agent uses beliefs about actions and their consequences to search for a solution over the more abstract space of plans, rather than over the space of situations.

Planning can be classified as :

- Linear planning
- Non-linear planning
- Hierarchical planning
- Contingency planning

GQ. What are the types of Planning? Explain any one.

Planning is a key ability for intelligent systems, increasing their autonomy and flexibility through the construction of sequences of actions to achieve their goals. It has been an area of research in artificial intelligence for over three decades. Planning techniques have been applied in a variety of tasks including robotics, process planning, wed-based information gathering, autonomous agents and spacecraft mission control. Planning involves the representation of actions and world models, reasoning about the effects of actions, and techniques for efficiently searching the space of possible plans. Some types of planning includes :

1. Hierarchical task network (HTN) planning
2. Heuristic search planning
3. Abstraction, macros, hierarchical planning
4. Mixed-initiative planning
5. Case-based planning

GQ. Compare problem solving and planning.

Problem solving vs. planning : A simple planning agent is very similar to problem solving agents in that it constructs plans that achieve its goals, and then executes them. The limitations of the problem solving approach motivates the design of planning systems.

To solve a planning problem using a state-space search approach we would let the :

- Initial state = initial situation
- Goal-test predicate = goal state description
- Successor function computed from the set of operators.
- Once a goal is found, solution plan is the sequence of operators in the path from the start node to the goal node.

UNIT VI End Se

In searches, operators are used simply to generate successor states and we cannot look "inside" an operator to see how it's defined. The goal-test predicate also is used as a "black box" to test if a state is a goal or not. The search cannot use properties of how a goal is defined in order to reason about finding path to that goal.

Planning is considered different from problem solving because of the difference in the way they represent states, goals, actions, and the differences in the way they construct action sequences.

Remember the search-based problem solver had four basic elements :

- **Representations of actions :** Programs that develop successor state descriptions which represent actions.
- **Representation of state :** Every state description is complete. This is because a complete description of the initial state is given, and actions are represented by a program.
- **Representation of goals :** A problem solving agent has only information about its goal which is in terms of a goal test and the heuristic function.
- **Representation of plans :** In problem solving, the solution is a sequence of actions.

Cordless drill for a problem solving exercise we need to specify :

Initial state : The agent is at home without any objects that he is wanting.

Operator set : Everything the agent can do.

6.5.1 Limits of A.I.

Despite several advantages that AI can offer, there are also some limitations, which cannot be neglected. So, we note some major limitations of AI implementation :

1. AI cannot replace humans

- There is no doubt that machines can perform much more efficiently as compared to a human being. But it is practically not possible to replace humans with AIS, in the coming future.
- Machines work rationally and perform with given algorithm. But they don't have any emotions or moral values.
- The ability to bond with human-beings lacks with machines and that is a critical attribute needed to manage a team of humans.
- Machines can store a lot of data, but to recall and recollect information from them is too difficult a process as compared to human intelligence.

2. It lacks creativity

- Machines simply lack the ability to be creative unlike machines, humans can think and feel. And if guides their decision-making when it comes to being creative.
- AI can assist in terms of helping to determine what sort of imagery to style and price to be used.
- But when it comes to originality and creative thinking, a machine simply cannot compete with the human brain.
- But we need both human and machine.

3. AI cannot think for itself

- AI can only do what it is programmed to do.
- It normally does this extremely well what it is programmed. A person can change or cancel any message instantly, when a tragic event occurs. But machines cannot make 'change-decision' instantly.
- It is because we as humans are capable of showing compassion and empathy to the victims. A machine lacks the ability to show emotion.

4. No one-size-fits-all solutions

One has to rely on individual solutions to perform certain AI-powered tasks. Thus no one-size-fits all sorts of solutions.

Thus one has to use a range of artificially intelligent tasks to carry out a range of solutions to different problems. It becomes then expensive, time-consuming and messy.

5. Required supervision

Algorithms are like an engine : they can run but someone must put the ignition on. One has to feed the AI system with all the new information required for them to learn.

- This is called as **supervised learning.** And AI agents cannot copy the way a human learns naturally. And this is the biggest obstacle when it comes to creating a more-human-like AI agent.

6. Cost and maintenance

- AI is a new form of technology. There can be a significant cost of purchase and a need for on-going maintenance and repair. Artificial intelligence software also requires regular upgrades in order to adapt to the changing business environment.
- To implement any AI system one has to carefully consider the returns as the investment needs.

7. Causes harm

- AT may cause harm if a system fails. For example, consider a driverless car that miscalculates the location of an oncoming vehicle and causes a deadly collision.
- The system can cause harm if the system is used by someone who wishes to cause harm; e.g. consider a terrorist who uses a self-driving car to deploy explosives in a crowded area.

6.5.2 Ethics of A.I.

- Ethics of Artificial intelligence is a system of moral principles and techniques.
- They are intended to inform the development and responsible use of **artificial intelligence technology.**
- Since AI is becoming an **integral part of products and services,** organisation are starting to develop AI codes of ethics An AI code of ethics is also called an AI value platform It is a policy statement that defines the role of artificial intelligence as that is applied to the continued development of the human race.
- The aim of an AI code of ethics is to provide stakeholders with guidance when stakeholders are faced with ethical decision regarding the use of artificial intelligence.
- Isaac Asimov, the science fiction writer, developed the **three laws** of **Robotics** as a means of limiting the potential dangers of Artificial intelligent Agents.

The laws are

1. The first law forbids robots from actively harming humans, and acts if harm comes to humans.

   The second law order robots to obey humans unless the orders are not in accordance with the first law.

   The third law orders robots to protect themselves while acting or performing actions in accordance with the first two laws.

An AI ethics framework is important because if clearly indicates **risks and benefits of AI tools.** It also establishes guide-lines for its responsible use.

The industry and interested parties are supposed to examine major social issues and accordingly come up with a system of moral techniques for using AI responsibly.

UNIT VI End S

Ethical challenges of AI

In the use of AI technology, enterprises face several ethical difficulties :

1. When AI system does not proceed in the planned way, teams must be able locate the failure. The teams have to trace the cause through a complex chain of algorithmic systems.
2. Organisations using AI data must be able to explain (i) the source data, (ii) resulting data and (iii) the functioning of algorithms.
3. According to adam wisniewski, "AI needs to have a strong degree of traceability to ensure that if harms arise, they can be traced back to the cause".

4. Responsibility

- Responsibility for the consequences of AI-based decisions need s to be sorted out in a process that includes lawyers, regulators and citizens.
- The consequences may be catastrophic, including loss of capital, health or life.

5. Fairness

In data sets containing personal identifiable information, there should not be **biases in terms of race, gender** or **ethnicity. This is extremely important.**

6. Misuse

AI algorithms may be used for purposes other than those for which they were created.

According to wisniewski "these scenarios should be analysed at the design stage to minimise the risks and introduce safety measures to reduce the adverse effects in such cases".

Benefits of ethical AI

- The rapid acceleration in AI adoption across businesses has given rise to two major trends : (i) The rise of customer-centricity and (ii) the rise in social activism.
- According to Sudhir Jha, senior vice-president at Mastercard : "Businesses are rewarded not only for providing personalised products and services but also for upholding customer values and doing good for the society in which they operate".
- AI plays an important role in how consumers interact with and perceive a brand.
- In addition to consumers, employees want to feel good about the businesses they work for.
- Jha said, "Responsible AI can go a long way in retaining talent and ensuring smooth execution of a company's operations".

AI code of ethics

According to Jason shephered, vice president of ecosystem at Zededa : A proactive approach to ensuring ethical AI requires addressing three key areas :

(a) Policy

- This includes developing the appropriate framework for driving standardisation and establishing regulations.
- Ethical AI policies have to indicate to deal with legal issues when something goes wrong.
- Companies may adopt AI policies into their own code of conduct. But its effectiveness will depend on employees following the rules, which may not always be realistic.

(b) Education

- Data scientists, front-line employees, executives and consumers all must know policies, key-considerations and negative impact of unethical AI and take data.
- One main concern is the tradeoff between ease of use around data sharing and the potential negative repercussions of over sharing or adverse automations.

- According to shepherd, "ultimately, consumers' willingness to proactively take control of their data and pay attention to potential threats enables by AI is a complex equation based on a combination of instant gratification, value, perception and risk",

(c) Technology

- Executives need to architect AI systems to automatically detect take data and unethical behaviour examples include the take videos and text to undermine a competitor.
- Since AI tools become commoditise, this becomes a serious issue. To combat this threat, organisations need to invest in open, transparent and trusted AI infrastructure.

6.5.3 Future of AI

- Artificial intelligence has gone from fiction to reality in a matter of years machines can think like humans was a trilling idea for storyline.
- But in recent times, it has changed from fiction to reality. People are using AI technologies in their everyday lives, and somehow it has become an integral part of their daily rituals.
- The evolution of artificial Intelligence is not as smooth as one may think.
- We first see how it came into existence and then note its impact on the human race.

1. The Evolution of AI
2. The impact of AI
3. The Impact of AI on society.
4. Artificial Intelligence for the next Decade.
5. Concerns Behind the Rise of AI
6. AI and the future of privacy.
7. Preparing for the future.

1. The evolution of AI

- Alan Turing created the concept that machines can think like humans. He constructed Turing test that helped to understand that machines can think like humans.
- The technology gained momentum when we were introduced to computers.
- The goal was to ensure problem-solving and develop the concept of interpretations.
- The technology was designed to enable machines to keep learning and updating themselves using the data available. If there is a fair amount of data, it amounts to useful information and better algorithm designs.

Let us see why AI is important for society in general.

(i) The present state of AI helps businesses with smooth functioning and better management It helps streamline significant functions essential for prosperity.

(ii) Customer service and other resource support have been improved due to AI functions.

(iii) Research and growing abilities of this technology have led to better functioning.

This technology is used in almost all industries, as it helps with forecasts and faud detections.

2. The Impact of AI

Let us see how AI benefits us and why AI is vital for businesses.

(i) Automation

- At can help us to increase speed and makes us agile. Form building grocers to checking on inventory.
- Thus we can observe greater productivity and better management owing to advances in AI.

UNIT VI End S

(ii) Impressive Experiences

When we incorporate AI in our business, we allows our business to automate interactions. We need not invest in resources, our AI-information is always available. This leads us to better engagement and growth.

(iii) Better healthcare

- AI can ease accessibility and improved awareness in healthcare needs.
- Past data combined with medical advancements can help forecast better and enable. Faster cure.
- Maintaining records and transfer of care has become easier with this technology.

(iv) Problem-solving

- The intelligence combines method and past data to build algorithms that can detect frauds, identity cybercrimes, reduce transaction risks, create better experiences.
- One can notice greater efficiency as a result of this technology.

3. Impact of Artificial intelligence on society

AI is providing better ways to live together and improving the conveniences for the human race.

(i) AI is offering personalisation to users. As a customer, one may need a different solution than someone who is residing with you.

Knowing how you work, what kind of products you use, and how you conduct your daily lives, the artificial agent can process a solution that works best for you and this makes life much easier.

(ii) The doctors can offer their services through AI-based video-conferencing and monitoring tools.

This way, the doctors can keep an eye on their patients and monitor their health.

AI can help the administration to take note of when cleaning is needed inside the hospital.

4. Artificial Intelligence for the next decade

The future of AI in the next decade looks quite promising and eventful. We mention here a few trends to incorporate in the future.

(i) When it comes to businesses, the AI has paved the way for smart monitoring, quicker feedback and improved business lines.

The robotic process automation reduces the time taken to complete repetitive tasks, making the strategies more effective.

(ii) The mobile apps and other mediums that use AI will help you realise what the users want

Data-driven insights lead to personalised solutions and improve interactions.

In the coming years, more businesses will take keen interest in this technology. This will enhance improved business processes.

Thus, AI will help with content creations. Increased solutions, and improve all industry aspects.

5. Concerns Behind the rise of AI

So far we have discussed the advances and benefits in all the fields. Now we discuss how AI can be a possible threat to our society.

(i) AI is cutting down jobs and needs a few jobs for a few roles.

(ii) Also one cannot ignore the risk associated with privacy and personal data intrusion.

(iii) one of the biggest threat facing the people is how to control it. It one creates technology or a machine, one should be able to handle it. If one loses the control over it can create havoc around it.

Hence it is essential to have a boundary set for this technology.

(iv) With advanced automation, a lot many people will be jobless. This will cause financial loses and economic issues.

In the past salesmen used to forecast the future figures using data analysis. With AI one can achieve greater accuracy and one can remove the study of the data from the salesperson's profile.

(v) AI can also cause military and arms issues by releasing intelligent weapons. These can be trained and developed by smart minds.

This can be a threat to the world. Also, keeping these weapons in check can be dangerous. It could lead to wars with little or no scope of human intervention to make them stop.

6. AI and the future of privacy

- If the researchers can create things for your comfort, they can even threaten you with solutions that can endanger your own self.
- Training machines to hack into the data and create the data into an opportunity. A criminal's mind can rank any where from high risk solutions to endangering solutions.
- There is no way to get over this issue. One can master the art of securing data, but one cannot overcome the dangerous species working continuously, creating this technology.

7. Preparing for the future

While implementing AI in your organisation, there are a few things that you have to consider.

(i) Incorporate an open culture in the organisation. It is not enough to adopt the new technology, you also need to prepare your organisation for the artificial intelligence future. You need to boost the performance of the employees.

AI will help you improve efficiency thus leading to better monitoring and enhanced productivity.

If you plan on engaging with Artificial intelligence future, you need to get data from those who have already this technology. You have to understand how they managed to achieve success and what did not work for them.

It is not enough to have the technology; you should work with it. This way, you can innovate and bring better idea to the table.

Advances in AI will reach the super intelligence stage in a couple of years.

A new technology transition has begun and most businesses are using this unique fixture.

It has not only helped with better sales and forecasts but also gives companies new growth opportunities.

NOTES

6.6 COMPONENTS OF ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers "Developing computer programs to store complex problems by application of process that analogues to human reasoning."

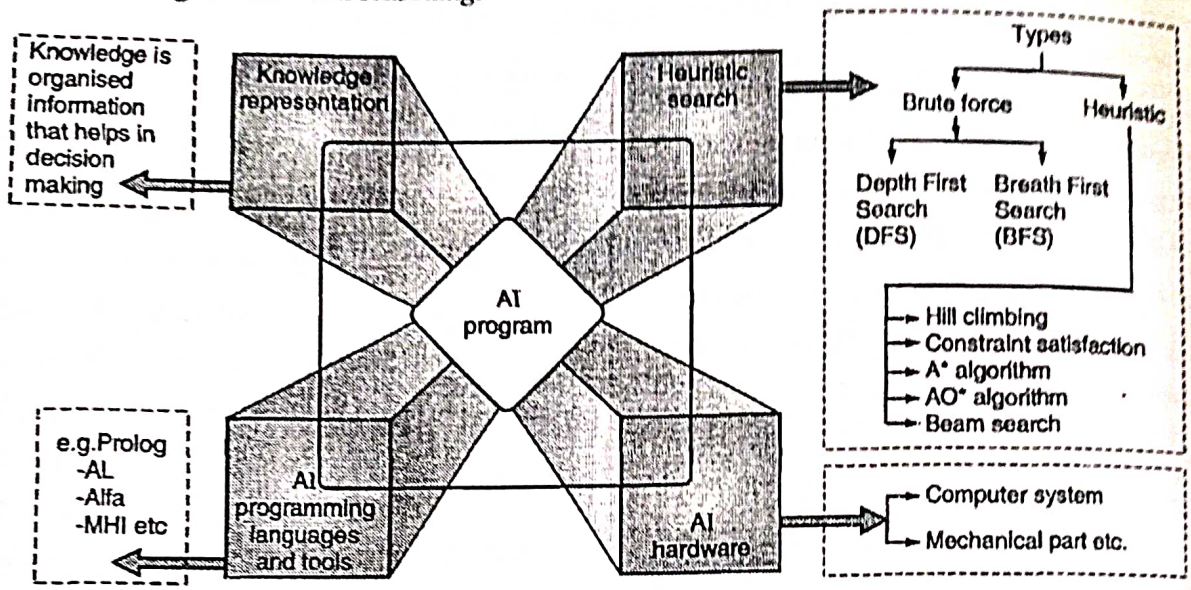


Fig. 6.6.1 : Major components of an AI system

Any AI system has four major components :

1. Knowledge representation.
2. Heuristic search.
3. AI programming languages and tool.
4. AI hardware.

- **Definition :** Artificial intelligence (AI) refers to a **"Developing computer programs to store complex problems by application of process that analogues to human reasoning."** The quality of the results depends on how much knowledge the system possesses. The available knowledge must be represented in a very efficient way. Hence **knowledge representation** is a vital component of the system.
- It is not merely enough that knowledge is represented efficiently. The **inference process** should also be equally good for satisfactory results. The inference process is broadly divided into **brute and heuristic search procedures.**
- Today just like we have specialized languages and programs for data-processing and scientific applications, we encounter specialized languages and tools for AI programming. AI languages provide the basic functions for AI programming and tools for the right environment.
- Most of the AI programs are implemented on **Von Neumann machines** only. However, dedicated workstations have emerged for AI programming. AI provides details on the machine architecture for AI machines.

6.6.1 Functions of Each Components of AI System

(1) Knowledge Representation

- Knowledge representation in AI describes the representation of knowledge. It expresses how the beliefs, intentions, and judgements of an intelligent agent can be expressed suitably for automated reasoning. One of the primary purposes is modelling intelligent behaviour for an agent.

- It represents information from the real world for a computer to understand and then utilise this knowledge to solve complex real-life problems, like communicating with human-beings in natural language.
- It stores data in a database, and allows a machine to learn from that knowledge and behave intelligently like a human being.

(2) Heuristic Search

- In mathematical optimisation and computer science, heuristic is a technique designed for solving a problem more quickly when classic methods fail to find any exact solution.
- This is achieved by trading optionality, completeness, accuracy, or precision for speed. In a way, it can be considered a short-cut.
- A heuristic function ranks alternatives in search algorithms at each balancing step based on available information to decide which branch to follow.

(3) AI programming Language and Tool

(i) Python

- Python language is the forerunner of all other languages. It uses simple syntax.
- The simple syntax allows to spend much more time on planning the core structure, which is why python is an ideal choice for machine learning processes.

(ii) Lisp

- It is the oldest language used for AI processes. Lisp is considered as a tool in AI with its enlarged scope of turning thoughts into reality.
- The language differentiates itself from other AI languages by eying precision.

(iii) R

- This is highly efficient programming language.
- Packages like G models, RODBC, one R and Tm allow huge support for machine learning processes.

(iv) Prolog

- It is completely designed by logic. Prolog requires three important factors : rules, facts and the desired result.
- Once these requirements are provided, the language will figure out the link between the three and design an AI solution.

(v) C++

- One of the major advantages of having C++ as programming language is its processing speed.
- To run complex automated solutions efficiently, C++ is very useful.

(vi) Javascript

- For versatility, Javascript is ahead of Java.
- With continuous developments, multiple domain growth, backend use, ease of use, efficiency etc. Support the above statement.

(vii) Java

- The best benefit of using the JAVA programming language is the presence of virtual machine technology.
- Java virtual machine eases the implementation process.

(viii) Haskell

- This programming language is well-known for resolving errors, during the compilation process and even before that.

- The features like build-in memory and code reusability increase the time allotted for planning the process.

(ix) Julia

- The programming language is well-known for numerical analysis.
- The best feature of Julia is dynamic type system, which allows you to use the language for literally any process.
- Other features involve in-built package manager, macro programming abilities, multiple dispatch support and suitability with C functions.

4. AI Hardware

- An AI accelerator is a specialised hardware accelerator or computer system designed to accelerate AI and machine learning applications, including neural networks and machine vision.
- Applications include algorithms for robotics, internet of things and other data-intensive or sensor-driven tasks.

6.7 AI ARCHITECTURE

- Architectures have undergone profound transformations by Artificial Intelligence and Automation computers and software eliminate tedious repetitive activities, optimising the production of technical material. Even then automation and artificial intelligence would not replace architects. According to ole Bauman, "As long as there are **human beings** and their **challenges,** there will be **Architecture."**
- Artificial intelligence will make an architect's work significantly easier by analysing the whole data and creating models that can save a lot of time and energy of the architect.
- AI can be called an **estimation tool for various aspects** while constructing a building.
- In general, Artificial Intelligence can reshape Architecture in five ways :

(I) Parametric Architecture

- Parametric Architecture creates various types of output designs and such structures which even an architect will fail to imagine.
- It is like an architect's programming language. It an architect considers a building, then AI reframes it to fit it into some other requirements. This way an architect can freely think about different ideas and create something new.

(II) Construction and planning

- Constructing a building not only needs a lot of pre-planning but also some more effort to get an architect's opinion to the construction.
- AI analyses the whole data and creates new models that can save a lot of time and energy of the architect.
- Thus, when it comes to construction part, AI can help so that human. Efforts become negligible.

(III) Laying the foundation

- Earlier the architects had to spends countless hours of research at the starting point of any new project. Now AI steps in and analyses the aggregate data in millisecond and also exhibits some models so that the architect can think about the conceptual design
- For example, to build a home for the family, if one has the whole information about the requirement of the family, one can simply put the whole data using AI and create design variations in a short period of time.

(IV) AI for making homes

At this time, people need everything to be smartly designed. Today's high technology society demand smart homes.

Architects need not bother about how to use AI to create the designs of home only, but they should concentrate on user's experience.

(V) Smart cities

Change is something that should never change. The way your city looks today will be very much different in the coming future. Hence city planning is a challenging task. It needs a lot of precision planning. The fundament task in city planning is to analyse all the possible aspects, like how the population is going to be in the coming years, how a city will flow, how the ecosystem will coexist.

- All these factors indicate that the future architects will have to give less efforts in drawing and more into satisfying all the requirements of the user with the help of Artificial Intelligence.

☞ Some more points

*[AI's ability to use data to make decisions and recommendations will be crucial to the design process, especially in the early stage of an architect's project.]

*[Smart cities will be places driven by real-time data and feedback, communicating with itself like a living organism. The buildings, smart phones, cars, and public places will communicate with each other to improve living conditions, limit waste, increase safety, and also limit traffic. This trend is visible today in some of the world's most advanced cities.]

Chapter Ends...

❑❑❑