# Artificial Intelligence

## MODULE -IV

**Knowledge and reasoning:** Logical Agents: Knowledge -based Agents, The WUMPUS world, Logic, Propositional Logic, Reasoning Patterns in Propositional logic, Resolution, Forward and Backward chaining. First-order Logic: Syntax and Semantics of First-Order Logic

## What is knowledge representation?

Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world. **But how machines do all these things comes under knowledge representation and reasoning**. Hence we can describe Knowledge representation as following:

- Knowledge representation and reasoning (KR, KRR) is the part of Artificial intelligence which concerned with AI agents thinking and how thinking contributes to intelligent behavior of agents.
- It is responsible for representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real world problems such as diagnosis a medical condition or communicating with humans in natural language.
- It is also a way which describes how we can represent knowledge in artificial intelligence. Knowledge representation is not just storing data into some database, but it also enables an intelligent machine to learn from that knowledge and experiences so that it can behave intelligently like a human.

### What to Represent:

Following are the kind of knowledge which needs to be represented in AI systems:

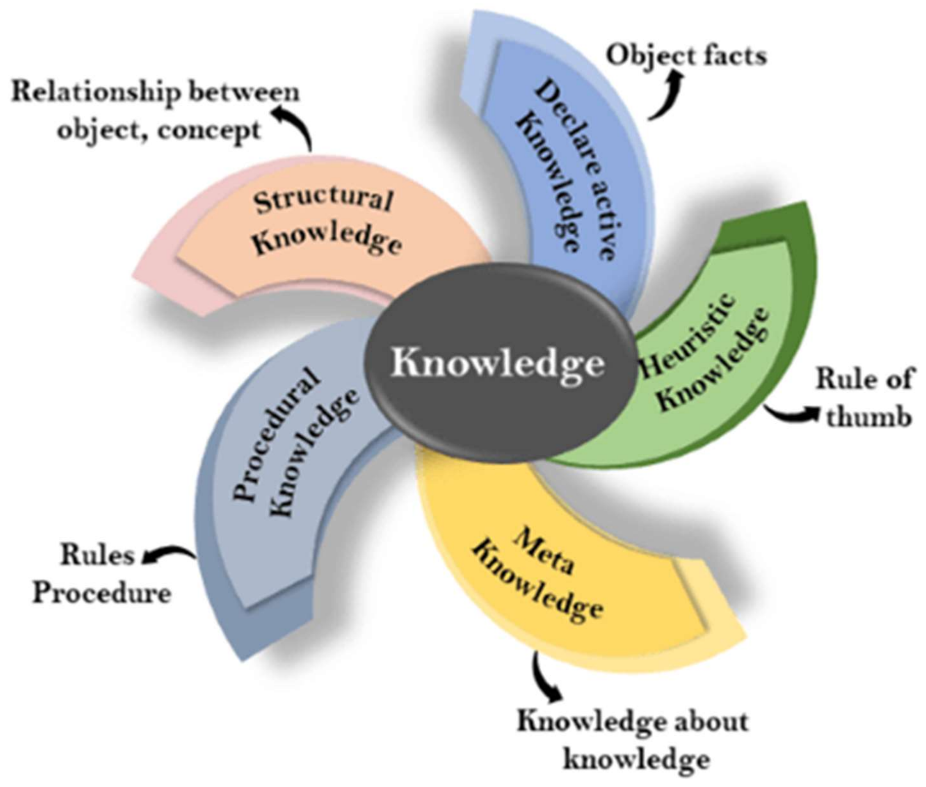
- **Object:** All the facts about objects in our world domain. E.g., Guitars contains strings, trumpets are brass instruments.
- **Events:** Events are the actions which occur in our world.
- **Performance:** It describe behavior which involves knowledge about how to do things.
- **Meta-knowledge:** It is knowledge about what we know.

- **Facts:** Facts are the truths about the real world and what we represent.
- **Knowledge-Base:** The central component of the knowledge-based agents is the knowledge base. It is represented as KB. The Knowledgebase is a group of the Sentences (Here, sentences are used as a technical term and not identical with the English language).

**Knowledge:** Knowledge is awareness or familiarity gained by experiences of facts, data, and situations. Following are the types of knowledge in artificial intelligence:

## Types of knowledge:

Following are the various types of knowledge:



### 1. Declarative Knowledge:

- Declarative knowledge is to know about something.
- It includes concepts, facts, and objects.
- It is also called descriptive knowledge and expressed in declarative sentences.
- It is simpler than procedural language.

## 2. Procedural Knowledge

- It is also known as imperative knowledge.
- Procedural knowledge is a type of knowledge which is responsible for knowing how to do something.
- It can be directly applied to any task.
- It includes rules, strategies, procedures, agendas, etc.
- Procedural knowledge depends on the task on which it can be applied.

## 3. Meta-knowledge:

- Knowledge about the other types of knowledge is called Meta-knowledge.

## 4. Heuristic knowledge:

- Heuristic knowledge is representing knowledge of some experts in a filed or subject.
- Heuristic knowledge is rules of thumb based on previous experiences, awareness of approaches, and which are good to work but not guaranteed.

## 5. Structural knowledge:

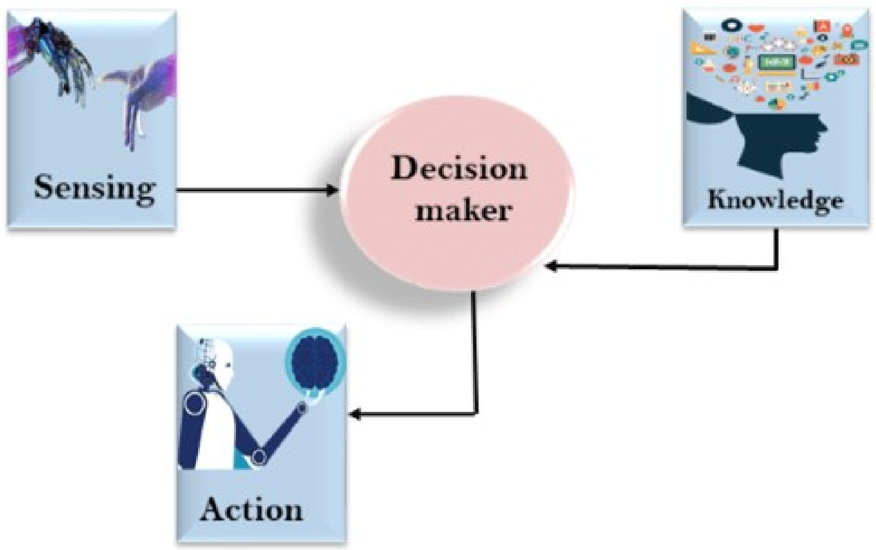
- Structural knowledge is basic knowledge to problem-solving.
- It describes relationships between various concepts such as kind of, part of, and grouping of something.
- It describes the relationship that exists between concepts or objects.

## The relation between knowledge and intelligence:

Knowledge of real-worlds plays a vital role in intelligence and same for creating artificial intelligence. Knowledge plays an important role in demonstrating intelligent behavior in AI agents. An agent is only able to accurately act on some input when he has some knowledge or experience about that input.

Let's suppose if you met some person who is speaking in a language which you don't know, then how you will able to act on that. The same thing applies to the intelligent behavior of the agents.

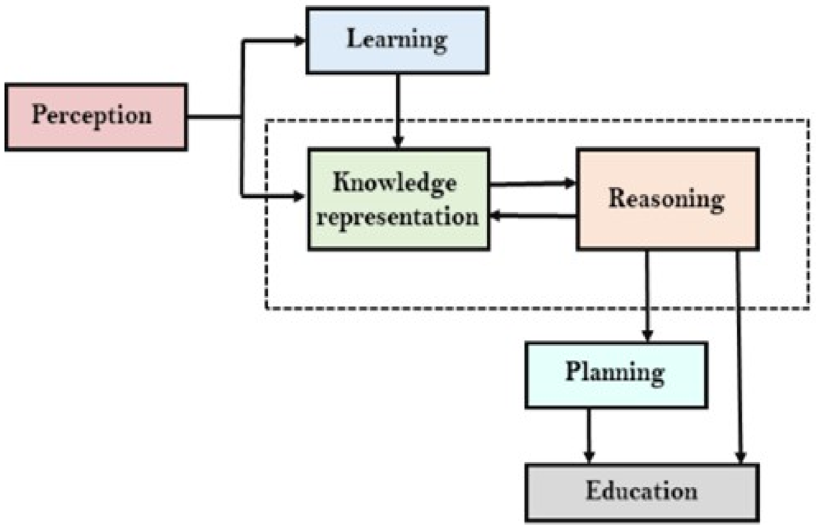
As we can see in below diagram, there is one decision maker which act by sensing the environment and using knowledge. But if the knowledge part will not present then, it cannot display intelligent behavior.



## AI knowledge cycle:

An Artificial intelligence system has the following components for displaying intelligent behavior:

- Perception
- Learning
- Knowledge Representation and Reasoning
- Planning
- Execution



The above diagram is showing how an AI system can interact with the real world and what components help it to show intelligence. AI system has Perception component by which it retrieves information from its environment. It can be visual, audio or another form of sensory input. The learning component is responsible for learning from data captured by Perception comportment. In the complete cycle, the main components are knowledge representation and Reasoning. These two components are involved in showing the intelligence in machine-like humans. These two components are independent with each other but also coupled together. The planning and execution depend on analysis of Knowledge representation and reasoning.

## Approaches to knowledge representation:

There are mainly four approaches to knowledge representation, which are given below:

### 1. Simple relational knowledge:

- It is the simplest way of storing facts which uses the relational method, and each fact about a set of the object is set out systematically in columns.
- This approach of knowledge representation is famous in database systems where the relationship between different entities is represented.
- This approach has little opportunity for inference.

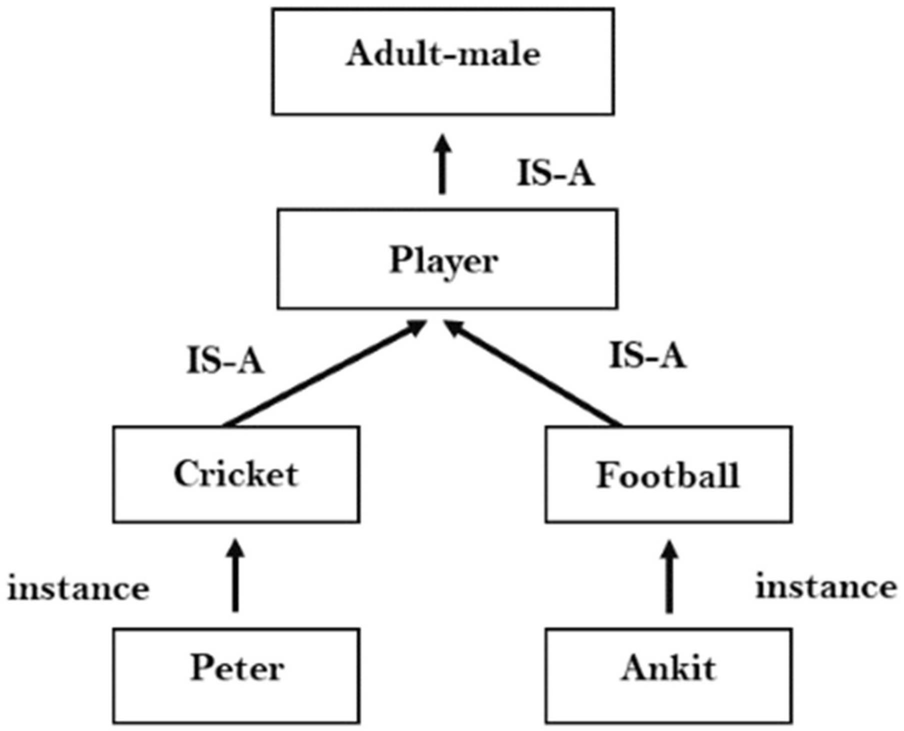
**Example: The following is the simple relational knowledge representation.**

| Player | Weight | Age |
|---|---|---|
| Player1 | 65 | 23 |
| Player2 | 58 | 18 |
| Player3 | 75 | 24 |

### 2. Inheritable knowledge:

- In the inheritable knowledge approach, all data must be stored into a hierarchy of classes.
- All classes should be arranged in a generalized form or a hierarchal manner.
- In this approach, we apply inheritance property.
- Elements inherit values from other members of a class.
- This approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation.
- Every individual frame can represent the collection of attributes and its value.
- In this approach, objects and values are represented in Boxed nodes.
- We use Arrows which point from objects to their values.

- **Example:**



## 3. Inferential knowledge:

- Inferential knowledge approach represents knowledge in the form of formal logics.
- This approach can be used to derive more facts.
- It guaranteed correctness.
- **Example:** Let's suppose there are two statements:
  a. Marcus is a man
  b. All men are mortal
  Then it can represent as;

  **man(Marcus)**
  **∀x = man (x) ----------> mortal (x)s**

## 4. Procedural knowledge:

- Procedural knowledge approach uses small programs and codes which describes how to do specific things, and how to proceed.
- In this approach, one important rule is used which is **If-Then rule**.

- In this knowledge, we can use various coding languages such as **LISP language** and **Prolog language**.
- We can easily represent heuristic or domain-specific knowledge using this approach.
- But it is not necessary that we can represent all cases in this approach.

## Requirements for knowledge Representation system:

A good knowledge representation system must possess the following properties.

1. **Representational Accuracy:**
   KR system should have the ability to represent all kind of required knowledge.
2. **Inferential Adequacy:**
   KR system should have ability to manipulate the representational structures to produce new knowledge corresponding to existing structure.
3. **Inferential Efficiency:**
   The ability to direct the inferential knowledge mechanism into the most productive directions by storing appropriate guides.
4. **Acquisitional efficiency**- The ability to acquire the new knowledge easily using automatic methods.

## Knowledge-Based Agent in Artificial intelligence:

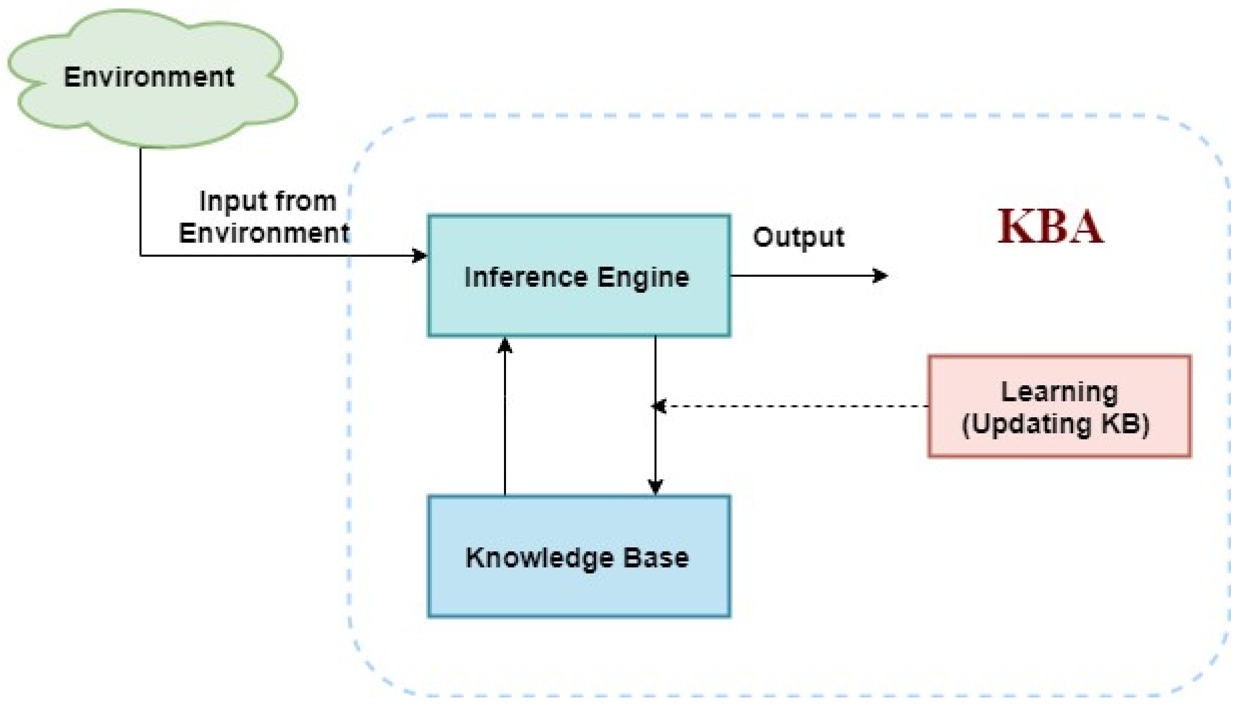
- An intelligent agent needs knowledge about the real world for taking decisions and reasoning to act efficiently.
- Knowledge-based agents are those agents who have the capability of maintaining an internal state of knowledge, reason over that knowledge, update their knowledge after observations and take actions. These agents can represent the world with some formal representation and act intelligently.
- Knowledge-based agents are composed of two main parts:
  - Knowledge-base and
  - Inference system.

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

- An agent should be able to represent states, actions, etc.
- An agent Should be able to incorporate new percepts

- An agent can update the internal representation of the world
- An agent can deduce the internal representation of the world
- An agent can deduce appropriate actions.

## The architecture of knowledge-based agent:



The above diagram is representing a generalized architecture for a knowledge-based agent. The knowledge-based agent (KBA) take input from the environment by perceiving the environment. The input is taken by the inference engine of the agent and which also communicate with KB to decide as per the knowledge store in KB. The learning element of KBA regularly updates the KB by learning new knowledge.

Knowledge base: Knowledge-base is a central component of a knowledge-based agent, it is also known as KB. It is a collection of sentences (here 'sentence' is a technical term and it is not identical to sentence in English). These sentences are expressed in a language which is called a knowledge representation language. The Knowledge-base of KBA stores fact about the world.

### Why use a knowledge base?

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

## Inference system:

Inference means deriving new sentences from old. Inference system allows us to add a new sentence to the knowledge base. A sentence is a proposition about the world. Inference system applies logical rules to the KB to deduce new information.

Inference system generates new facts so that an agent can update the KB. An inference system works mainly in two rules which are given as:

- **Forward chaining**
- **Backward chaining**

## Operations Performed by KBA:

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

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.

## A generic knowledge-based agent:

Following is the structure outline of a generic knowledge-based agents program:

```
1. function KB-AGENT(percept):
2. persistent: KB, a knowledge base
3.         t, a counter, initially 0, indicating time
4. TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
5. Action = ASK(KB, MAKE-ACTION-QUERY(t))
6. TELL(KB, MAKE-ACTION-SENTENCE(action, t))
7.  t = t + 1
8.  return action
```

The knowledge-based agent takes percept as input and returns an action as output. The agent maintains the knowledge base, KB, and it initially has some background knowledge of the real world. It also has a counter to indicate the time for the whole process, and this counter is initialized with zero.

Each time when the function is called, it performs its three operations:

- Firstly it TELLs the KB what it perceives.
- Secondly, it asks KB what action it should take
- Third agent program TELLS the KB that which action was chosen.

The MAKE-PERCEPT-SENTENCE generates a sentence as setting that the agent perceived the given percept at the given time.

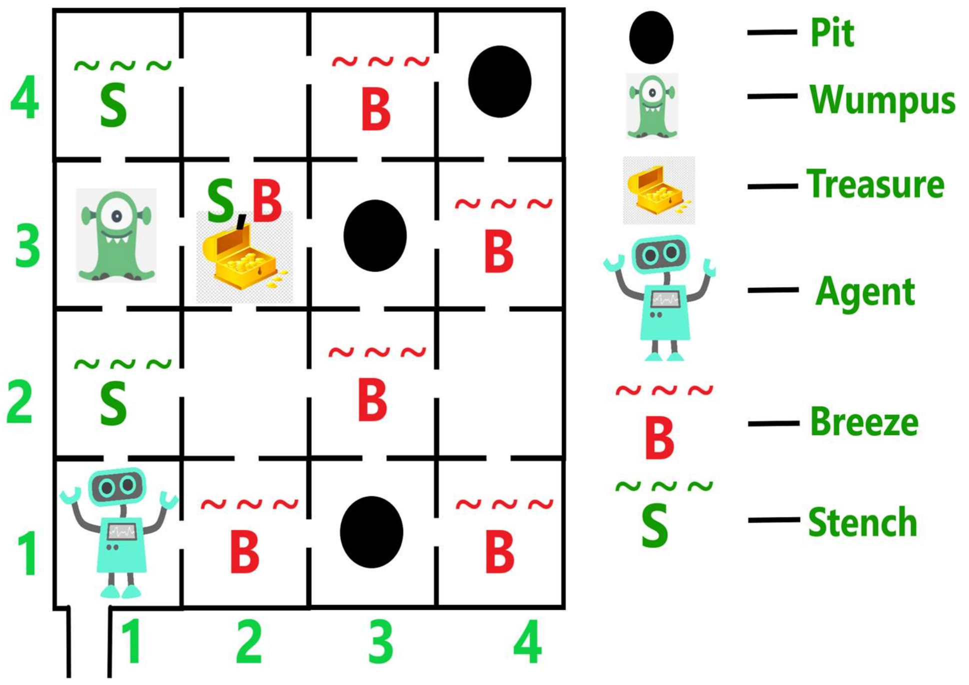
The MAKE-ACTION-QUERY generates a sentence to ask which action should be done at the current time.

## The Wumpus World:

The Wumpus World's agent is an example of a knowledge-based agent that represents Knowledge representation, reasoning and planning. Knowledge-Based agent links general knowledge with current percepts to infer hidden characters of current state before selecting actions. Its necessity is vital in partially observable environments.

### Problem Statement:

The Wumpus world is a cave with 16 rooms (4×4). Each room is connected to others through walkways (no rooms are connected diagonally). The knowledge-based agent starts from Room[1, 1]. The cave has – some pits, a treasure and a beast named Wumpus. The Wumpus can not move but eats the one who enters its room. If the agent enters the pit, it gets stuck there. The goal of the agent is to take the treasure and come out of the cave. The agent is rewarded, when the goal conditions are met. The agent is penalized, when it falls into a pit or being eaten by the Wumpus. Some elements support the agent to explore the cave, like -The wumpus's adjacent rooms are stenchy. -The agent is given one arrow which it can use to kill the wumpus when facing it (Wumpus screams when it is killed). – The adjacent rooms of the room with pits are filled with breeze.
-The treasure room is always glittery.



PEAS represents Performance Measures, Environment, Actuators, and Sensors. The PEAS description helps in grouping the agents.

## PEAS Description for the Wumpus World problem:

1. **Performance measures:**
    - Agent gets the gold and return back safe = +1000 points
    - Agent dies = -1000 points
    - Each move of the agent = -1 point
    - Agent uses the arrow = -10 points
2. **Environment:**
    - A cave with 16(4×4) rooms
    - Rooms adjacent (not diagonally) to the Wumpus are stinking
    - Rooms adjacent (not diagonally) to the pit are breezy
    - The room with the gold glitters
    - Agent's initial position – Room[1, 1] and facing right side
    - Location of Wumpus, gold and 3 pits can be anywhere, except in Room[1, 1].
3. **Actuators:**
    Devices that allow the agent to perform the following actions in the environment.
    - Move forward
    - Turn right
    - Turn left
    - Shoot
    - Grab
    - Release

4. **Sensors:**
   Devices which helps the agent in sensing the following from the environment.
   - Breeze
   - Stench
   - Glitter
   - Scream (When the Wumpus is killed)
   - Bump (when the agent hits a wall)

## Propositional logic in Artificial intelligence:

Propositional logic (PL) is the simplest form of logic where all the statements are made by propositions. A proposition is a declarative statement which is either true or false. It is a technique of knowledge representation in logical and mathematical form.

**Example:**

1. a) It is Sunday.
2. b) The Sun rises from West (False proposition)
3. c) 3+3= 7(False proposition)
4. d) 5 is a prime number.

Following are some basic facts about propositional logic:

- Propositional logic is also called Boolean logic as it works on 0 and 1.
- In propositional logic, we use symbolic variables to represent the logic, and we can use any symbol for a representing a proposition, such A, B, C, P, Q, R, etc.
- Propositions can be either true or false, but it cannot be both.
- Propositional logic consists of an object, relations or function, and logical connectives.
- These connectives are also called logical operators.
- The propositions and connectives are the basic elements of the propositional logic.
- Connectives can be said as a logical operator which connects two sentences.
- A proposition formula which is always true is called tautology, and it is also called a valid sentence.
- A proposition formula which is always false is called Contradiction.
- A proposition formula which has both true and false values is called

- Statements which are questions, commands, or opinions are not propositions such as "Where is Rohini", "How are you", "What is your name", are not propositions.

## Syntax of propositional logic:

The syntax of propositional logic defines the allowable sentences for the knowledge representation. There are two types of Propositions:

**a. Atomic Propositions**

**b. Compound propositions**

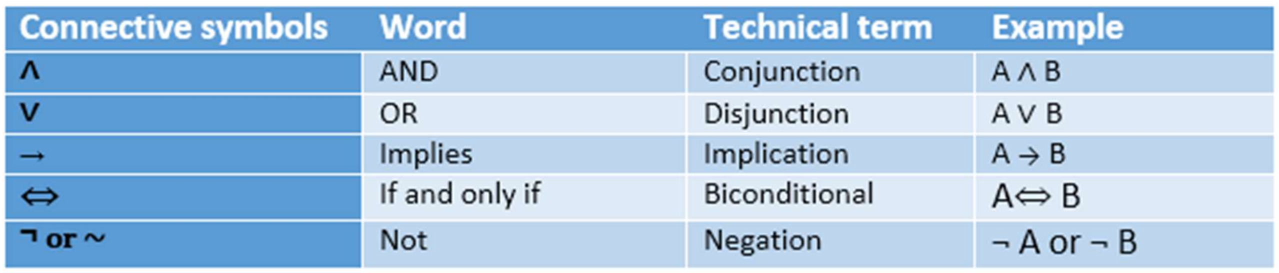
- **Atomic Proposition:** Atomic propositions are the simple propositions. It consists of a single proposition symbol. These are the sentences which must be either true or false.

## Logical Connectives:

Logical connectives are used to connect two simpler propositions or representing a sentence logically. We can create compound propositions with the help of logical connectives. There are mainly five connectives, which are given as follows:

1. **Negation:** A sentence such as ¬ P is called negation of P. A literal can be either Positive literal or negative literal.
2. **Conjunction:** A sentence which has ∧ connective such as, P ∧ Q is called a conjunction.
   Example: Rohan is intelligent and hardworking. It can be written as,
   P= Rohan is intelligent,
   Q= Rohan is hardworking. → P∧ Q.
3. **Disjunction:** A sentence which has ∨ connective, such as P ∨ Q. is called disjunction, where P and Q are the propositions.
   Example: "Ritika is a doctor or Engineer",
   Here P= Ritika is Doctor. Q= Ritika is Doctor, so we can write it as P ∨ Q.
4. **Implication:** A sentence such as P → Q, is called an implication. Implications are also known as if-then rules. It can be represented as
   If it is raining, then the street is wet.
   Let P= It is raining, and Q= Street is wet, so it is represented as P → Q
5. **Biconditional:** A sentence such as P⇔ Q is a Biconditional sentence, example If I am breathing, then I am alive
   P= I am breathing, Q= I am alive, it can be represented as P ⇔ Q.

Following is the summarized table for Propositional Logic Connectives:



| Connective symbols | Word | Technical term | Example |
|---|---|---|---|
| ∧ | AND | Conjunction | A ∧ B |
| ∨ | OR | Disjunction | A ∨ B |
| → | Implies | Implication | A → B |
| ⇔ | If and only if | Biconditional | A⇔ B |
| ¬ or ~ | Not | Negation | ¬ A or ¬ B |

## First-Order Logic in Artificial intelligence:

In the topic of Propositional logic, we have seen that how to represent statements using propositional logic. But unfortunately, in propositional logic, we can only represent the facts, which are either true or false. PL is not sufficient to represent the complex sentences or natural language statements. The propositional logic has very limited expressive power. Consider the following sentence, which we cannot represent using PL logic.

- **"Some humans are intelligent", or**
- **"Sachin likes cricket."**

To represent the above statements, PL logic is not sufficient, so we required some more powerful logic, such as first-order logic.

## Reasoning in Artificial intelligence

In previous topics, we have learned various ways of knowledge representation in artificial intelligence. Now we will learn the various ways to reason on this knowledge using different logical schemes.

### Reasoning:

The 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.

## Types of Reasoning:

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

- Deductive reasoning
- Inductive reasoning
- Abductive reasoning
- Common Sense Reasoning
- Monotonic Reasoning
- Non-monotonic Reasoning

## 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.

Deductive reasoning is a type of propositional logic in AI, and it requires various rules and facts. 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, which can be explained as below example.

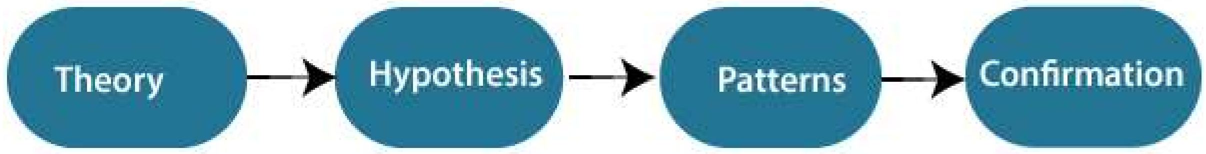
**Example:**

**Premise-1: All the human eats veggies**

**Premise-2: Suresh is human.**

**Conclusion: Suresh eats veggies.**

The general process of deductive reasoning is given below:



## 2. Inductive Reasoning:

Inductive reasoning is a form of reasoning to arrive at a conclusion using limited sets of facts by the process of generalization. It starts with the series of specific facts or data and reaches to a general statement or conclusion.

Inductive reasoning is a type of propositional logic, which is also known as cause-effect reasoning or bottom-up reasoning.

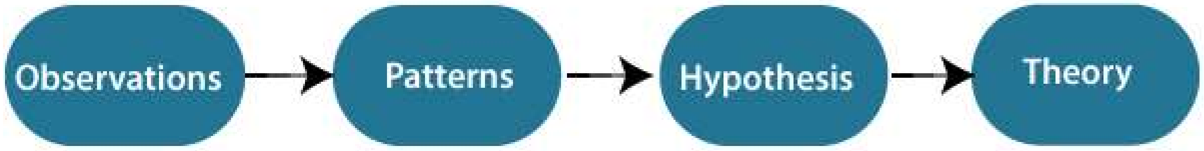
In inductive reasoning, we use historical data or various premises to generate a generic rule, for which premises support the conclusion.

In inductive reasoning, premises provide probable supports to the conclusion, so the truth of premises does not guarantee the truth of the conclusion.

**Example:**

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

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



## 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, the 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.

## 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 events which occurs on every day.

It relies on good judgment 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. 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 not useful for the real-time systems, as in real time, facts get changed, so we cannot use monotonic reasoning.

Monotonic reasoning is used in conventional reasoning systems, and a logic-based system is monotonic.

Any theorem proving is an example of monotonic reasoning.

**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:

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

### Disadvantages of Monotonic Reasoning:

- We cannot represent the real world scenarios using Monotonic reasoning.
- Hypothesis knowledge cannot be expressed with monotonic reasoning, which means facts should be true.
- Since we can only derive conclusions from the old proofs, so new knowledge from the real world cannot be added.

## 6. Non-monotonic Reasoning

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

Logic will be said as non-monotonic if some conclusions can be invalidated by adding more knowledge into 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 suppose the knowledge base contains the following knowledge:

- **Birds can fly**
- **Penguins cannot fly**
- **Pitty is a bird**

So from the above sentences, we can conclude that **Pitty can fly**.

However, 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:

- For real-world systems such as Robot navigation, we can use non-monotonic reasoning.
- In Non-monotonic reasoning, we can choose probabilistic facts or can make assumptions.

### Disadvantages of Non-monotonic Reasoning:

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

## First-Order logic:

- First-order logic is another way of knowledge representation in artificial intelligence. It is an extension to propositional logic.
- FOL is sufficiently expressive to represent the natural language statements in a short way.
- First-order logic is also known as Predicate logic or First-order predicate logic. First-order logic is a powerful language that develops information about the objects in a more easy way and can also express the relationship between those objects.
- First-order logic (like natural language) does not only assume that the world contains facts like propositional logic but also assumes the following things in the world:
  - **Objects:** A, B, people, numbers, colors, wars, theories, squares, pits, wumpus, ......
  - **Relations:** It can be unary relation such as: red, round, is adjacent, or n-any relation such as: the sister of, brother of, has color, comes between
  - **Function:** Father of, best friend, third inning of, end of, ......
- As a natural language, first-order logic also has two main parts:

a. **Syntax**

b. **Semantics**

### Syntax of First-Order logic:

The syntax of FOL determines which collection of symbols is a logical expression in first-order logic. The basic syntactic elements of first-order logic are symbols. We write statements in short-hand notation in FOL.

## Basic Elements of First-order logic:

Following are the basic elements of FOL syntax:

| | |
|---|---|
| Constant | 1, 2, A, John, Mumbai, cat,.... |
| Variables | x, y, z, a, b,.... |
| Predicates | Brother, Father, >,.... |
| Function | sqrt, LeftLegOf, .... |
| Connectives | ∧, ∨, ¬, ⇒, ⇔ |
| Equality | == |
| Quantifier | ∀, ∃ |

## Atomic sentences:

- Atomic sentences are the most basic sentences of first-order logic. These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms.
- We can represent atomic sentences as Predicate (term1, term2, ......, term n).

Example: Ravi and Ajay are brothers: => Brothers(Ravi, Ajay).
Chinky is a cat: => cat (Chinky).

## Complex Sentences:

- Complex sentences are made by combining atomic sentences using connectives.

First-order logic statements can be divided into two parts:

- Subject: Subject is the main part of the statement.
- Predicate: A predicate can be defined as a relation, which binds two atoms together in a statement.

Consider the statement: "x is an integer.", it consists of two parts, the first part x is the subject of the statement and second part "is an integer," is known as a predicate.

## Quantifiers in First-order logic:

- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression. There are two types of quantifier:

a. Universal Quantifier, (for all, everyone, everything)

b. Existential quantifier, (for some, at least one).

## Universal Quantifier:

Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.

The Universal quantifier is represented by a symbol ∀, which resembles an inverted A.

Note: In universal quantifier we use implication "→".

If x is a variable, then ∀x is read as:

- For all x
- For each x
- For every x.

**Example:**

All man drink coffee.

Let a variable x which refers to a cat so all x can be represented in UOD as below:

$$\forall x \; man(x) \rightarrow drink(x, coffee).$$

It will be read as: There are all x where x is a man who drink coffee.

## Existential Quantifier:

Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.

It is denoted by the logical operator ∃, which resembles as inverted E. When it is used with a predicate variable then it is called as an existential quantifier.

Note: In Existential quantifier we always use AND or Conjunction symbol (∧).

If x is a variable, then existential quantifier will be ∃x or ∃(x). And it will be read as:

- There exists a 'x.'
- For some 'x.'
- For at least one 'x.'

Example:

Some boys are intelligent.

∃x: boys(x) ∧ intelligent(x)

It will be read as: There are some x where x is a boy who is intelligent.

**Points to remember:**

- The main connective for universal quantifier ∀ is implication →.
- The main connective for existential quantifier ∃ is and ∧.

## Properties of Quantifiers:

- In universal quantifier, ∀x∀y is similar to ∀y∀x.
- In Existential quantifier, ∃x∃y is similar to ∃y∃x.
- ∃x∀y is not similar to ∀y∃x.

**Some Examples of FOL using quantifier:**

1. All birds fly.
In this question the predicate is "fly(bird)."
And since there are all birds who fly so it will be represented as follows.
∀x bird(x) →fly(x).

2. Every man respects his parent.
In this question, the predicate is "respect(x, y)," where x=man, and y= parent.
Since there is every man so will use ∀, and it will be represented as follows:

∀x man(x) → respects (x, parent).

3. Some boys play cricket.
In this question, the predicate is "play(x, y)," where x= boys, and y= game. Since there are some boys so we will use ∃, and it will be represented as:

∃x boys(x) → play(x, cricket).

4. Not all students like both Mathematics and Science.
In this question, the predicate is "like(x, y)," where x= student, and y= subject.
Since there are not all students, so we will use ∀ with negation, so following representation for this:

¬∀ (x) [ student(x) → like(x, Mathematics) ∧ like(x, Science)].

5. Only one student failed in Mathematics.
In this question, the predicate is "failed(x, y)," where x= student, and y= subject.
Since there is only one student who failed in Mathematics, so we will use following representation for this:

∃(x) [ student(x) → failed (x, Mathematics) ∧∀ (y) [¬(x==y) ∧ student(y) → ¬failed (x, Mathematics)].

## Free and Bound Variables:

The quantifiers interact with variables which appear in a suitable way. There are two types of variables in First-order logic which are given below:

**Free Variable:** A variable is said to be a free variable in a formula if it occurs outside the scope of the quantifier.

Example: ∀x ∃(y)[P (x, y, z)], where z is a free variable.

**Bound Variable:** A variable is said to be a bound variable in a formula if it occurs within the scope of the quantifier.

Example: ∀x [A (x) B( y)], here x and y are the bound variables

## Resolution in FOL:

### Resolution

Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions. It was invented by a Mathematician John Alan Robinson in the year 1965.

Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule which can efficiently operate on the **conjunctive normal form or clausal form**.

**Clause**: Disjunction of literals (an atomic sentence) is called a **clause**. It is also known as a unit clause.

**Conjunctive Normal Form**: A sentence represented as a conjunction of clauses is said to be **conjunctive normal form** or **CNF**.

**Note:** To better understand this topic, firstly learns the FOL in AI.

### The resolution inference rule:

The resolution rule for first-order logic is simply a lifted version of the propositional rule. Resolution can resolve two clauses if they contain complementary literals, which are assumed to be standardized apart so that they share no variables.

$$\frac{l_1 \vee \ldots\ldots \vee l_k, \qquad m_1 \vee \ldots\ldots \vee m_n}{SUBST(\theta, l_1 \vee \ldots\ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots\ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots\ldots \vee m_n)}$$

Where **li** and **mj** are complementary literals.

This rule is also called the **binary resolution rule** because it only resolves exactly two literals.

### Example:

We can resolve two clauses which are given below:

**[Animal (g(x) V Loves (f(x), x)]  and  [¬ Loves(a, b) V ¬Kills(a, b)]**

Where two complimentary literals are: **Loves (f(x), x) and ¬ Loves (a, b)**

These literals can be unified with unifier **θ= [a/f(x), and b/x]** , and it will generate a resolvent clause:

**[Animal (g(x) V ¬ Kills(f(x), x)].**

### Steps for Resolution:

1.Conversion of facts into first-order logic.

2.Convert FOL statements into CNF

3.Negate the statement which needs to prove (proof by contradiction)

4.Draw resolution graph (unification).

To better understand all the above steps, we will take an example in which we will apply resolution.

**Example:**

**John likes all kind of food.**

**Apple and vegetable are food**

**Anything anyone eats and not killed is food.**

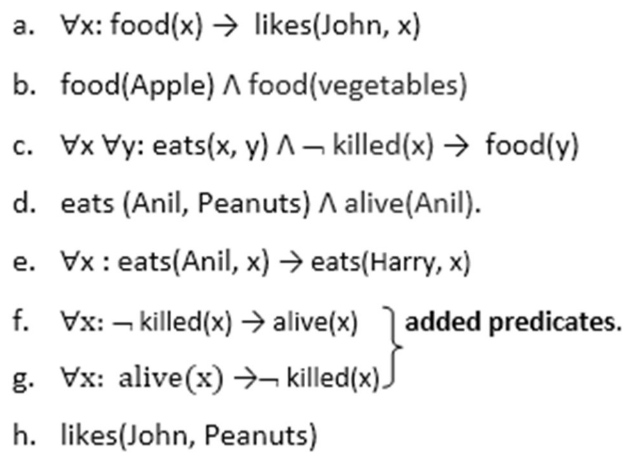
**Anil eats peanuts and still alive**

**Harry eats everything that Anil eats.**
**Prove by resolution that:**

**John likes peanuts.**

**Step-1: Conversion of Facts into FOL**

In the first step we will convert all the given statements into its first order logic.



a. $\forall x: \text{food}(x) \rightarrow \text{likes}(\text{John}, x)$
b. $\text{food}(\text{Apple}) \land \text{food}(\text{vegetables})$
c. $\forall x \forall y: \text{eats}(x, y) \land \neg \text{killed}(x) \rightarrow \text{food}(y)$
d. $\text{eats}(\text{Anil}, \text{Peanuts}) \land \text{alive}(\text{Anil})$.
e. $\forall x : \text{eats}(\text{Anil}, x) \rightarrow \text{eats}(\text{Harry}, x)$
f. $\forall x: \neg \text{killed}(x) \rightarrow \text{alive}(x)$ } **added predicates.**
g. $\forall x: \text{alive}(x) \rightarrow \neg \text{killed}(x)$
h. $\text{likes}(\text{John}, \text{Peanuts})$

**Step-2: Conversion of FOL into CNF**

In First order logic resolution, it is required to convert the FOL into CNF as CNF form makes easier for resolution proofs.

**Eliminate all implication (→) and rewrite**

∀x ¬ food(x) V likes(John, x)

food(Apple) Λ food(vegetables)

∀x ∀y ¬ [eats(x, y) Λ ¬ killed(x)] V food(y)

eats (Anil, Peanuts) Λ alive(Anil)

∀x ¬ eats(Anil, x) V eats(Harry, x)

∀x¬ [¬ killed(x) ] V alive(x)

∀x ¬ alive(x) V ¬ killed(x)

likes(John, Peanuts).

**Move negation (¬)inwards and rewrite**

∀x ¬ food(x) V likes(John, x)

food(Apple) Λ food(vegetables)

∀x ∀y ¬ eats(x, y) V killed(x) V food(y)

eats (Anil, Peanuts) Λ alive(Anil)

∀x ¬ eats(Anil, x) V eats(Harry, x)

∀x ¬killed(x) ] V alive(x)

∀x ¬ alive(x) V ¬ killed(x)

likes(John, Peanuts).

**Rename variables or standardize variables**

∀x ¬ food(x) V likes(John, x)

food(Apple) Λ food(vegetables)

∀y ∀z ¬ eats(y, z) V killed(y) V food(z)

eats (Anil, Peanuts) Λ alive(Anil)

∀w¬ eats(Anil, w) V eats(Harry, w)

∀g ¬killed(g) ] V alive(g)

∀k ¬ alive(k) V ¬ killed(k)

likes(John, Peanuts).

**Eliminate existential instantiation quantifier by elimination.**
In this step, we will eliminate existential quantifier ∃, and this process is known as Skolemization. But in this example problem since there is no existential quantifier so all the statements will remain same in this step.

**Drop Universal quantifiers.**
In this step we will drop all universal quantifier since all the statements are not implicitly quantified so we don't need it.

1.¬ food(x) V likes(John, x)

2.food(Apple)

3.food(vegetables)

4.¬ eats(y, z) V killed(y) V food(z)

5.eats (Anil, Peanuts)

6.alive(Anil)

7.¬ eats(Anil, w) V eats(Harry, w)

8.killed(g) V alive(g)

9.¬ alive(k) V ¬ killed(k)

10.likes(John, Peanuts).

**Note:** Statements "food(Apple) Λ food(vegetables)" and "eats (Anil, Peanuts) Λ alive(Anil)" can be written in two separate statements.

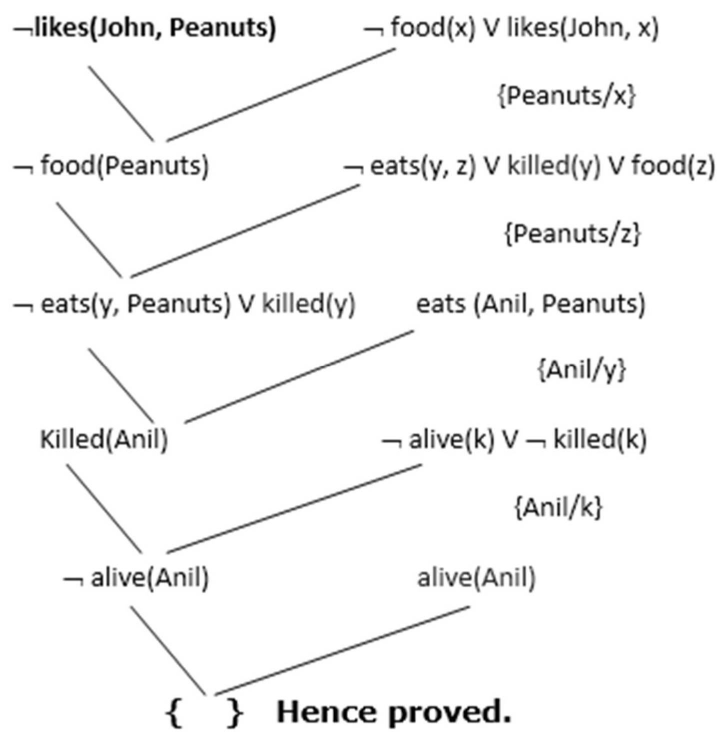
**Distribute conjunction ∧ over disjunction ¬.**
This step will not make any change in this problem.

**Step-3: Negate the statement to be proved**

In this statement, we will apply negation to the conclusion statements, which will be written as ¬likes(John, Peanuts)

**Step-4: Draw Resolution graph:**

Now in this step, we will solve the problem by resolution tree using substitution. For the above problem, it will be given as follows:



Hence the negation of the conclusion has been proved as a complete contradiction with the given set of statements.

## Explanation of Resolution graph:

In the first step of resolution graph, **¬likes(John, Peanuts)** , and **likes(John, x)** get resolved(canceled) by substitution of **{Peanuts/x}**, and we are left with **¬ food(Peanuts)**

In the second step of the resolution graph, **¬ food(Peanuts)** , and **food(z)** get resolved (canceled) by substitution of **{ Peanuts/z}**, and we are left with **¬ eats(y, Peanuts) V killed(y)** .

In the third step of the resolution graph, **¬ eats(y, Peanuts)** and **eats (Anil, Peanuts)** get resolved by substitution **{Anil/y}**, and we are left with **Killed(Anil)** .

In the fourth step of the resolution graph, **Killed(Anil)** and **¬ killed(k)** get resolve by substitution **{Anil/k}**, and we are left with **¬ alive(Anil)** .

In the last step of the resolution graph **¬ alive(Anil)** and **alive(Anil)** get resolved.

## Forward Chaining and backward chaining in AI:

In artificial intelligence, forward and backward chaining is one of the important topics, but before understanding forward and backward chaining lets first understand that from where these two terms came.

### Inference engine:

The inference engine is the component of the intelligent system in artificial intelligence, which applies logical rules to the knowledge base to infer new information from known facts. The first inference engine was part of the expert system. Inference engine commonly proceeds in two modes, which are:

a. **Forward chaining**
b. **Backward chaining**

### Horn Clause and Definite clause:

Horn clause and definite clause are the forms of sentences, which enables knowledge base to use a more restricted and efficient inference algorithm. Logical inference algorithms use forward and backward chaining approaches, which require KB in the form of the **first-order definite clause**.

**Definite clause:** A clause which is a disjunction of literals with **exactly one positive literal** is known as a definite clause or strict horn clause.

**Horn clause:** A clause which is a disjunction of literals with **at most one positive literal** is known as horn clause. Hence all the definite clauses are horn clauses.

**Example: (¬ p V ¬ q V k)**. It has only one positive literal k.

It is equivalent to p ∧ q → k.

### A. Forward Chaining

Forward chaining is also known as a forward deduction or forward reasoning method when using an inference engine. Forward chaining is a form of reasoning which start with atomic sentences in the knowledge base and applies inference rules (Modus Ponens) in the forward direction to extract more data until a goal is reached.

The Forward-chaining algorithm starts from known facts, triggers all rules whose premises are satisfied, and add their conclusion to the known facts. This process repeats until the problem is solved.

## Properties of Forward-Chaining:

- It is a down-up approach, as it moves from bottom to top.
- It is a process of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.
- Forward-chaining approach is also called as data-driven as we reach to the goal using available data.
- Forward -chaining approach is commonly used in the expert system, such as CLIPS, business, and production rule systems.

Consider the following famous example which we will use in both approaches:

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

### Facts Conversion into FOL:

- It is a crime for an American to sell weapons to hostile nations. (Let's say p, q, and r are variables)

  **American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p) ...(1)**

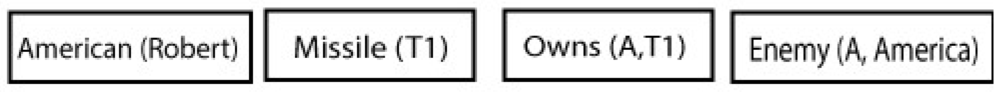
- Country A has some missiles. **?p Owns(A, p) ∧ Missile(p)**. It can be written in two definite clauses by using Existential Instantiation, introducing new Constant T1.
  **Owns(A, T1) ......(2)**
  **Missile(T1) .......(3)**
- All of the missiles were sold to country A by Robert.
  **?p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A) ......(4)**
- Missiles are weapons.
  **Missile(p) → Weapons (p) .......(5)**

- Enemy of America is known as hostile. **Enemy(p, America) →Hostile(p)** **........(6)**
- Country A is an enemy of America. **Enemy (A, America)** **.........(7)**
- Robert is American **American(Robert).** **..........(8)**

**Forward chaining proof:**

**Step-1:**

In the first step we will start with the known facts and will choose the sentences which do not have implications, such as: **American(Robert), Enemy(A, America), Owns(A, T1), and Missile(T1)**. All these facts will be represented as below.



**Step-2:**

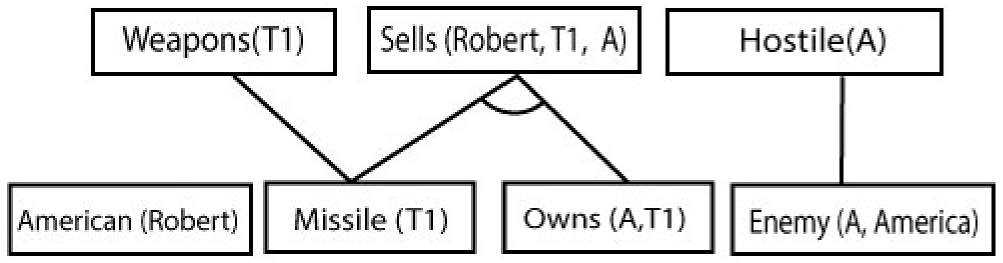
At the second step, we will see those facts which infer from available facts and with satisfied premises.

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

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

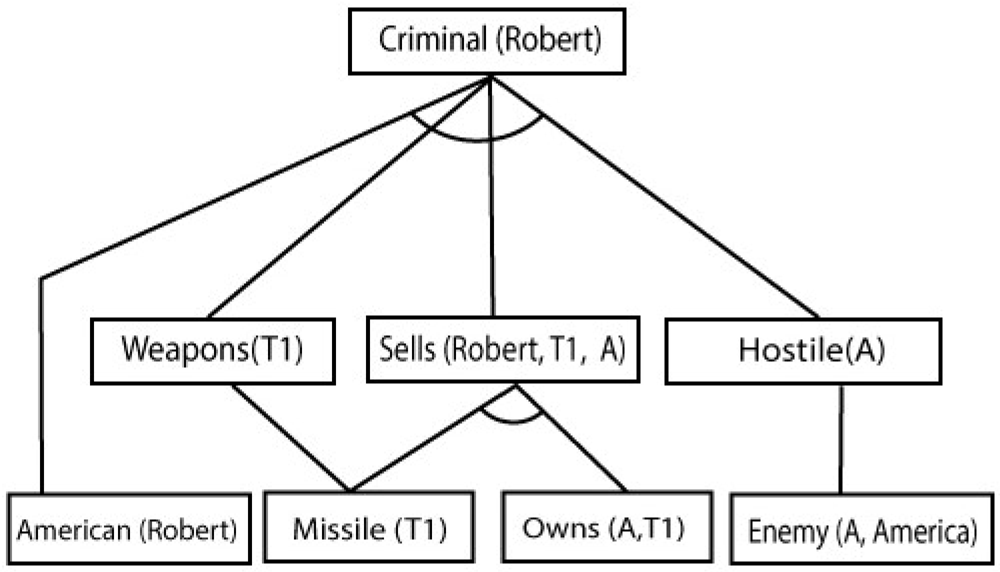
Rule-(4) satisfy with the substitution {p/T1}, **so Sells (Robert, T1, A)** is added, which infers from the conjunction of Rule (2) and (3).

Rule-(6) is satisfied with the substitution(p/A), so Hostile(A) is added and which infers from Rule-(7).



**Step-3:**

At step-3, as we can check Rule-(1) is satisfied with the substitution **{p/Robert, q/T1, 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 chaining approach.**

## B. Backward Chaining:

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

### Properties of backward chaining:

- It is known as a top-down approach.
- Backward-chaining is based on modus ponens inference rule.
- In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true.
- It is called a goal-driven approach, as a list of goals decides which rules are selected and used.
- Backward -chaining algorithm is used in game theory, automated theorem proving tools, inference engines, proof assistants, and various AI applications.
- The backward-chaining method mostly used a **depth-first search** strategy for proof.

Example:

In backward-chaining, we will use the same above example, and will rewrite all the rules.

- **American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p) ...(1)**
  **Owns(A, T1) ........(2)**
- **Missile(T1)**
- **?p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A) ......(4)**
- **Missile(p) → Weapons (p) .......(5)**
- **Enemy(p, America) →Hostile(p) ........(6)**
- **Enemy (A, America) .........(7)**
- **American(Robert). ..........(8)**

## Backward-Chaining proof:

In Backward chaining, we will start with our goal predicate, which is **Criminal(Robert)**, and then infer further rules.

**Step-1:**

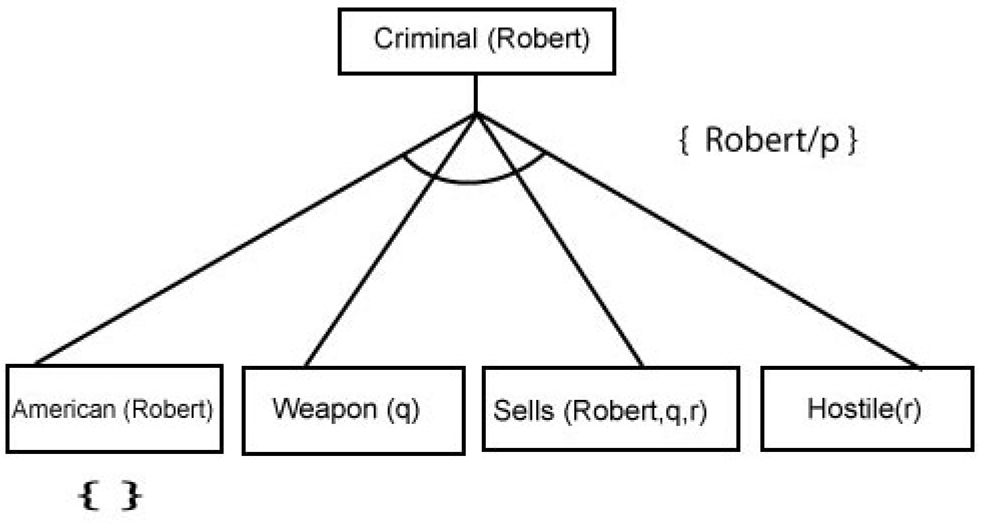
At the first step, we will take the goal fact. And from the goal fact, we will infer other facts, and at last, we will prove those facts true. So our goal fact is "Robert is Criminal," so following is the predicate of it.

Criminal (Robert)

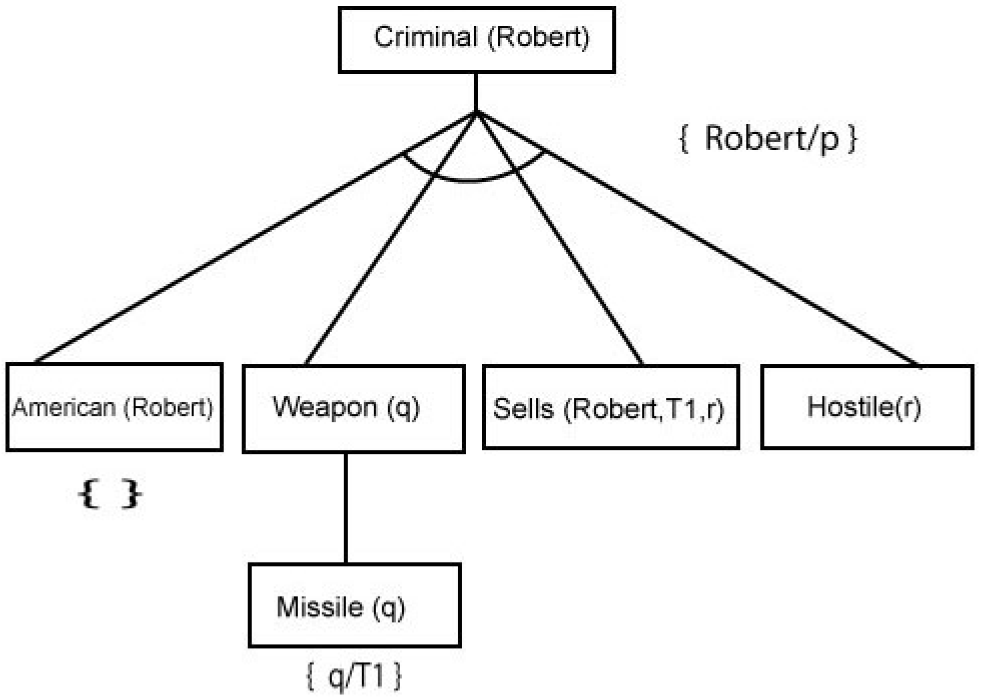
**Step-2:**

At the second step, we will infer other facts form 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 will replace p with Robert.

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

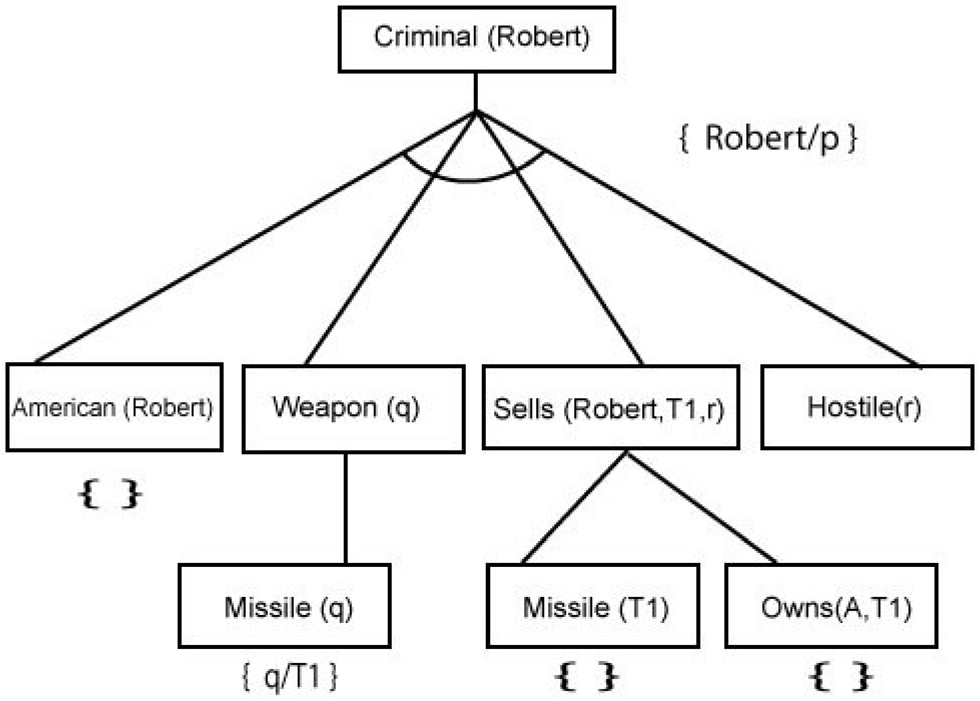


**Step-3:**t At step-3, we will extract further fact Missile(q) which infer from Weapon(q), as it satisfies Rule-(5). Weapon (q) is also true with the substitution of a constant T1 at q.



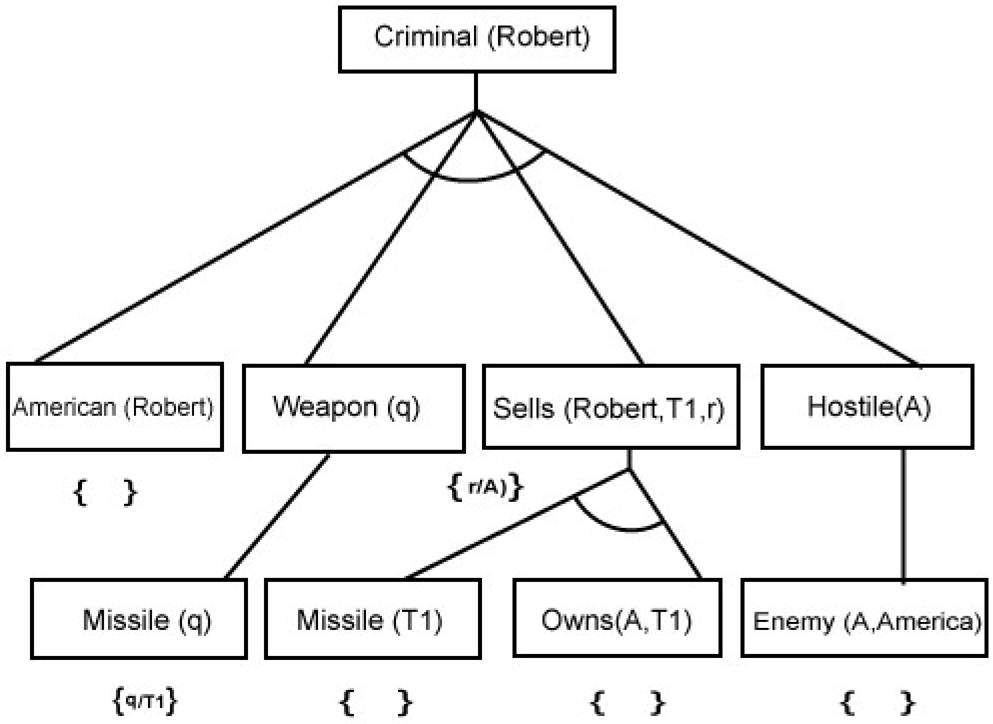
**Step-4:**

At step-4, we can infer facts Missile(T1) and Owns(A, T1) form Sells(Robert, T1, r) which satisfies the **Rule- 4**, with the substitution of A in place of r. So these two statements are proved here.



**Step-5:**

At step-5, 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.


Criminal (Robert)
American (Robert)
Weapon (q)
Sells (Robert,T1,r)
Hostile(A)
{ }
{ r/A)}
Missile (q)
Missile (T1)
Owns(A,T1)
Enemy (A,America)
{q/T1}
{ }
{ }
{ }