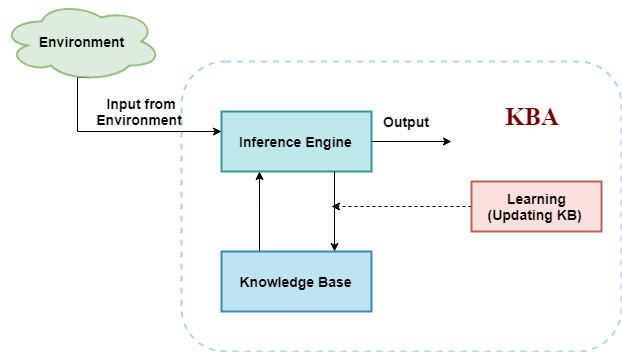
Knowledge-Based Agent in Artificial intelligence

* 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.

MAKE-ACTION-SENTENCE generates a sentence which asserts that the chosen action was executed.

Various levels of knowledge-based agent:

A knowledge-based agent can be viewed at different levels which are given below:

1. Knowledge level

Knowledge level is the first level of knowledge-based agent, and in this level, we need to specify what the agent knows, and what the agent goals are. With these specifications, we can fix its behavior. For example, suppose an automated taxi agent needs to go from a station A to station B, and he knows the way from A to B, so this comes at the knowledge level.

2. Logical level:

At this level, we understand that 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. At the logical level we can expect to the automated taxi agent to reach to the destination B.

3. Implementation level:

This is the physical representation of logic and knowledge. At the implementation level agent perform actions as per logical and knowledge level. 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:

There are mainly two approaches to build a knowledge-based agent:

1. **1. Declarative approach:** 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. This approach is called Declarative approach.
2. **2. Procedural approach:** In the procedural approach, we directly encode desired behavior as a program code. Which means we just need to write a program that already encodes the desired behavior or agent.

 **Declarative Approach** → প্রথমে ফাঁকা KB দিয়ে শুরু করে, ধীরে ধীরে বাক্য যোগ করা।

 **Procedural Approach** → সরাসরি প্রোগ্রামে আচরণ (behavior) কোড করে দেওয়া।

* বাস্তবে, দুই পদ্ধতির মিশ্রণ বেশি কার্যকর।

However, 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.

Knowledge Representation in AI

10 Jun 2025 |  9 min read

Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge, and based on 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 follows:

Knowledge representation and reasoning (KR, KRR) is the part of Artificial Intelligence which concerned with [AI agents](https://www.tpointtech.com/ai-agents)' thinking and how thinking contributes to the intelligent behaviour of agents.

It is responsible for representing information about the real world so that a computer can understand and utilise this knowledge to solve complex real-world problems, such as diagnosis a medical condition or communicating with humans in natural language.

It is also a way to describe how we can represent knowledge in [artificial intelligence](https://www.tpointtech.com/artificial-intelligence-ai). Knowledge representation is not just storing data in a database, but it also enables an intelligent machine to learn from that knowledge and experience so that it can behave intelligently like a human.

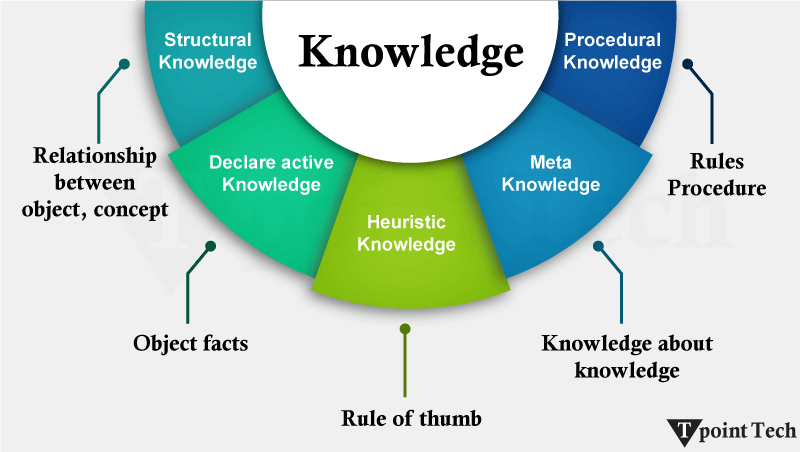
What to Represent?

The following are the kind of knowledge that needs to be represented in AI systems:

* **Object:** All the facts about objects in our world domain. E.g., Guitars contain strings, and trumpets are brass instruments.
* **Events:** Events are the actions that occur in our world.
* **Performance:** It describes behaviour that 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 Sentences (Here, sentences are used as a technical term and not identical with the English language).
* **Knowledge:** Knowledge is awareness gained by experiences of facts, data, and situations. The following are the types of knowledge in artificial intelligence:

Types of Knowledge

The following are the various types of knowledge:



1. Declarative Knowledge

* Declarative knowledge is knowing 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 that 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 to which it can be applied.

3. Meta-Knowledge

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

4. Heuristic Knowledge

* Heuristic knowledge represents the knowledge of some experts in a field or subject.
* Heuristic knowledge is rules Rules based on the experience of experts, which may not always be correct.

5. Structural Knowledge

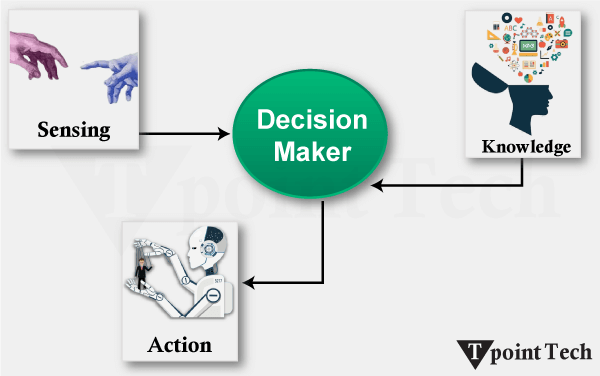
* Structural knowledge is basic knowledge for problem-solving.
* It describes relationships between various concepts such as kind of, part of, and grouping of something.
* It describes the relationship that exists between concepts or objects.

The Relation between Knowledge and Intelligence

Knowledge of the real world plays a vital role in intelligence, and the same applies to creating artificial intelligence. Knowledge plays an important role in demonstrating intelligent behaviour 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 that you met a person who is speaking in a language that you don't know; then how would you be able to act on that? The same thing applies to the intelligent behaviour of the agents.

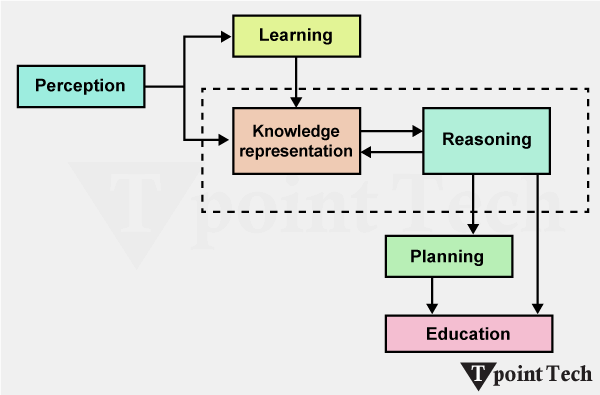
As we can see in the diagram below, there is one decision-maker who acts by sensing the environment and using knowledge. But if the knowledge part is not present, then it cannot display intelligent behaviour.



AI Knowledge Cycle

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

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



The above diagram shows how an AI system can interact with the real world and what components help it to show intelligence. An AI system has a 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 the Perception component.

In the complete cycle, the main components are knowledge representation and Reasoning. These two components are involved in showing the intelligence of machine-like humans. These two components are independent of each other but also coupled together. The planning and execution depend on the 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 objects 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:**

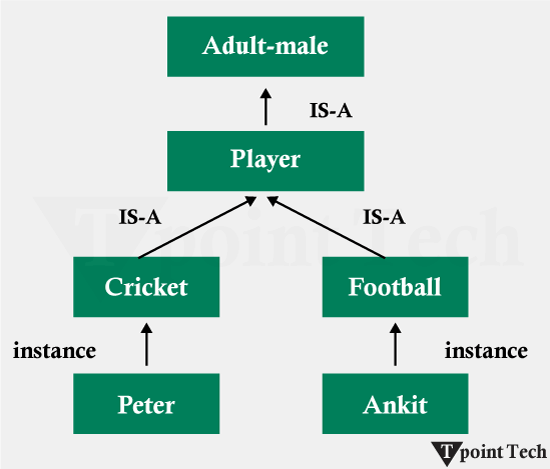
The following is a simple relational knowledge representation.

|  |  |  |
| --- | --- | --- |
| **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 in a hierarchy of classes.
* All classes should be arranged in a generalised form or a hierarchal manner.
* In this approach, we apply the 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 the instance relation.
* Every individual frame can represent the collection of attributes and their value.
* In this approach, objects and values are represented in Boxed nodes.
* We use Arrows that point from objects to their values.

**Example:**



3. Inferential Knowledge

* The inferential knowledge approach represents knowledge in the form of formal logic.
* This approach can be used to derive more facts.
* It guaranteed correctness.

**Example:**

Let's suppose there are two statements:

1. Marcus is a man
2. All men are mortal

Then it can be represented as;

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

4. Procedural Knowledge

* The procedural knowledge approach uses small programs and codes that describe how to do specific things and how to proceed.
* In this approach, one important rule is used, which is the If-Then rule.
* With this knowledge, we can use various coding languages such as LISP and Prologue.
* We can easily represent heuristic or domain-specific knowledge using this approach.
* However, it is not necessary that we can represent all cases in this approach.

Requirements for a knowledge Representation system:

A good knowledge representation system must possess the following properties.

**1. Representational Accuracy:** The KR system should have the ability to represent all kinds of required knowledge.

**2. Inferential Adequacy:** The KR system should have the ability to manipulate the representational structures to produce new knowledge corresponding to the existing structure.

**3. Inferential Efficiency:** The ability to direct the inferential knowledge mechanism in the most productive directions by storing appropriate guides.

**4. Acquisitional efficiency:** The ability to acquire new knowledge easily using automatic methods.

 **Representational Accuracy** → Ability to accurately represent all necessary knowledge.

 **Inferential Adequacy** → Ability to derive new knowledge from existing knowledge.

 **Inferential Efficiency** → Ability to perform inference quickly and effectively.

 **Acquisitional Efficiency** → Ability to easily acquire new knowledge.

Challenges in Knowledge Representation

Handling Ambiguity and Uncertainty

* **Ambiguity:** In other words, language, symbols, concepts, and other sorts of signifiers can be given more than one meaning, depending on the context. In this case, the word can be any of a financial institution or the side of a river as a bank.
* **Uncertainty:** But, unsurprisingly, incomplete, imprecise, or contradictory information is the fact of the matter. For example, the prediction of stock market trends is uncertainty of economic data, geopolitical factors, as well as human behavior.

They are commonly used to represent and infer under uncertainty with systems that are based on making reasoned guesses using what is known as **Bayesian networks and probabilistic graphical models.**

Scalability of Representation

* **Volume of Knowledge:** For example, there are plenty of high-dimensional domains involving lots of data to be managed by AI systems, which need to be made available in a timely fashion for use in answering questions (e.g., healthcare or autonomous vehicles).

As we store and retrieve knowledge at a large scale, we can do that using knowledge graphs and distributed storage systems, such as Neo4j, due to which knowledge can be stored and retrieved efficiently.

* **Complexity of Interrelations:** As the knowledge increases, but prior to a point at which computational efficiency is reached, the number of relationships between entities increases.

Clustering, as well as hierarchical representations and modular ontologies, can reduce the complexity in terms of complex relationships without destroying the complexity.

Balancing Expressiveness with Efficiency

**Expressiveness:**  
To explain something complicated, you need a lot of details. But if you use too many details, the computer gets slow. So, the best way is to use a mix of clear rules and learning from examples. This way, the computer understands well without getting too slow.

**Efficiency:**  
Computers work faster if they don’t think about every small thing. Sometimes, they miss little details and make small mistakes. But they use smart tricks to work fast and still do a good job. These tricks help the computer save time and remember things so it doesn’t repeat work.

.

Dynamic Knowledge Updates and Maintenance

**Dynamic Updates:**  
In fast-changing areas (like weather forecasting or social media), new information needs to be added constantly.  
**Solution:** Use Incremental Learning or Online Learning — update the system with new data without retraining the whole system from scratch.

**Maintenance:**  
Maintaining a large knowledge base is hard because data comes from different sources and may have duplicates or conflicts.  
**Solution:** Regular audits, conflict resolution, and using data deduplication tools.

Applications of Knowledge Representation

Here’s the English translation of the Knowledge Representation application areas you provided:

**Applications of Knowledge Representation**

**1. Problem Solving and Decision Making**

* **Structured Problem Analysis:** Breaking down problems into simpler parts using logic systems or production rules.
* **Decision Support Systems:** Using knowledge graphs and ontologies to analyze different solutions and provide recommendations.

**2. Robotics and Automation Systems**

* **Environmental Mapping:** Robots understand maps of their surroundings to find paths.
* **Task Automation:** Automatically performing tasks like assembling parts or cleaning.
* **Human-Robot Collaboration:** Robots use ontologies to understand human intentions and work accordingly.
* **Autonomous Vehicles:** Cars understand roads, traffic rules, and environment to drive themselves.

**3. Healthcare and Industry**

* **Clinical Decision Support Systems:** Diagnosis, treatment advice, and patient outcome prediction.
* **Drug Discovery:** Finding new medicines and analyzing causes of diseases.
* **Predictive Maintenance:** Predicting when machines will break down before it happens.
* **Process Optimization:** Improving manufacturing and production processes.

**4. Search Engines and Recommendation Systems**

* **Knowledge Graph in Search Engines:** For example, typing “Leonardo da Vinci” shows not just web pages but also detailed info about his life and work.
* **Content-Based Filtering:** Suggesting content based on user preferences.
* **Collaborative Filtering:** Making recommendations based on the preferences of others.

If you want, I can make it even simpler!

Techniques of Knowledge Representation

27 May 2025 |  9 min read

Artificial intelligence refers to the process of experiencing intelligence through machines to execute specific functions such as perceiving, understanding, deciding, and deciding. However, this becomes a challenge when it comes to accomplishing this goal because machines need human knowledge to accomplish such tasks. Knowledge representation, which can be defined as the ways and methods that enable the storage and understanding of human knowledge by machines, falls under [AI](https://www.tpointtech.com/artificial-intelligence-ai).

There is clearly a significant task with regard to knowledge representation in the context of making it understandable by machines for subsequent use in reasoning and problem-solving. To meet this challenge, several techniques of representing knowledge in artificial intelligence have been formulated, including the rule-based system, semantic network, frame knowledge representation, ontology, and logic-based knowledge representation.

They help organize the information in a manner that such knowledge can be processed and used for the application of varying levels of inference or reasoning.

What is Knowledge Representation?

Knowledge representation was put into practice in an attempt to capture and represent the extent of the relationship between certain concepts, ideas or objects in a way that can elicit inferences or conclusions. In order to do this, four different representation techniques can be employed: logical representation, semantic network representation, frame representation, and production rules.

This makes AI useful in practice in the sense that many intelligent systems are designed using the techniques of knowledge representation in order to reason, understand language, find patterns, learn, and make decisions. For instance, the KRS can be of help in constructing an application that would enable the user to ask questions related to a specific area of interest or create a recommender system to be used to recommend items of interest to the user.

Different Kinds of Knowledge That Need to Be Represented in AI

The knowledge that needs to be represented in AI can be classified as Objects, Events, Performance, Facts, Meta-knowledge or knowledge-base.

Objects

It is a nominal variable defined as things in the external environment that can be viewed in terms of their characteristics or can be GET IT tangible and inert. Some of the objects will be cars, buildings, and people. Various techniques, such as [object-oriented programming](https://www.tpointtech.com/r-object-oriented-programming) techniques, represent knowledge in AI.

Events

In wider terms, they refer to activities that occur in the world or actions that happen in the world. Some of the things that are associated with events include driving a car, preparing a meal or going to a concert. Event-based systems are used to represent knowledge in AI, and the use of events does this.

Performance

Performance can also be defined as the manner in which agents or systems act in terms of executing tasks. It consists of the purpose and aims of the task as well as the measures that will be employed to assess productivity. These systems rely on performance as the basis of knowledge in AI.

Facts

Facts mean statements that can either be true or false statements. It is common knowledge that a preposition is a part of speech that involves an adverbial modification of a verb, and it can be confirmed using a fact or as an argument from the conclusion. Some examples of facts include "the sky is blue", "the earth orbits around the sun", and "water boils at 100 degrees Celsius". Invariably, facts are used to model knowledge in AI knowledge-based systems.

Meta-Knowledge

Meta-knowledge refers to knowledge about knowledge. The first subtopic is the structure and organization of knowledge, which is more detailed about the structuring of information and how knowledge institutions are arranged. The meta-knowledge is crucial to AI since it facilitates the evaluation of the quality of knowledge for adequate reasoning to be applied.

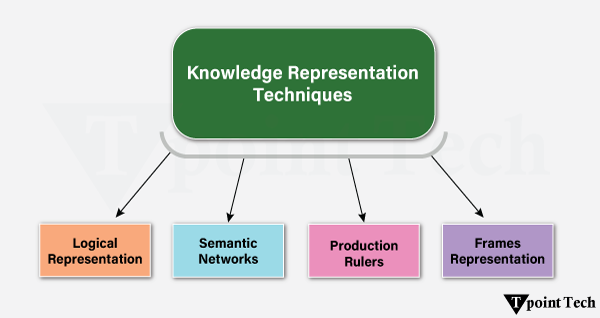
Knowledge-Base

A knowledge base is also referred to as 'artificial knowledge' and can be described as a pool of information in a format that can be accessed and utilized by machines. It is the information that is embedded within an entity and is pertinent to a specific subject area of activity. One of the most used knowledge representations in AI is the use of a knowledge base to represent knowledge in KBS.

Knowledge Representation Techniques

There are four main ways of knowledge representation, which are given as follows:

* Logical Representation
* Semantic Network Representation
* Frame Representation
* Production Rules



### ****1. Logical Representation****

Here, knowledge is expressed in a logical format, like “if … then …” type rules.

Two types of logic:

* **Propositional Logic** – works with whole statements.
* **Predicate Logic** – works with objects and their properties or relations.

**Example:**  
“All humans are mortal” — this kind of knowledge is written in a logical form.

**Advantages:**

* Allows clear and precise reasoning.
* Forms the basis of many programming languages.

**Cons:**

* Can be hard to use in some cases.
* Not like natural language, so harder for humans to read.

Semantic Network Representation

**📌 What is Semantic Network Representation?**

A **Semantic Network** is a method of representing knowledge in the form of a **graph**.

* **Node:** An object or concept
* **Arc:** The relationship between nodes

This way, objects and their relationships are shown visually, making it easy to understand and easy to add new information when needed.

**🔗 Types of Relationships**

In Semantic Networks, two main types of relationships are used:

1. **IS-A Relation (Inheritance)**  
   Meaning: *"is a …"*
   * Example: **Jerry IS-A Cat** → Jerry is a cat.
2. **Kind-of Relation**  
   Meaning: *"is a kind of …"*
   * Example: **Cat Kind-of Mammal** → A cat is a kind of mammal.

**📍 Example**

We can represent the following sentences as nodes and arcs:

* Jerry is a cat.
* Jerry is a mammal.
* Jerry is owned by Priya.
* Jerry is brown-coloured.
* All mammals are animals.

Here, the **nodes** are: Jerry, Cat, Mammal, Priya, Brown, Animal  
The **arcs** are: IS-A, owned-by, colour-of

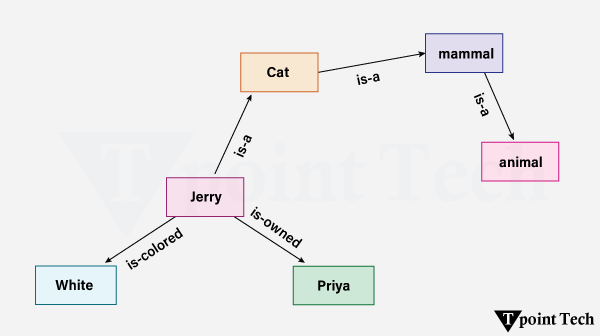
These can be shown together in a graph.

**✅ Advantages**

* Represents knowledge in a way similar to human memory.
* Clearly expresses meaning.
* Easy to understand and extend.

**❌ Drawbacks**

* Takes time to find answers (may need to search the whole network).
* Creating a huge memory network like humans is practically impossible.
* Cannot use quantifiers like “all” or “some”.
* No fixed standard for naming links.
* Not intelligent on its own — fully depends on the creator.



In the above diagram, we have represented the different types of knowledge in the form of nodes and arcs. Each object is connected with another object by some relation.

## 📌 What is Frame Representation?

A **Frame** is a type of **record-like structure** where detailed information about an **entity** or an **event** is stored in the form of **Attributes** and their **Values**.

* Each attribute is called a **Slot**.
* Each Slot’s value or additional information is called a **Facet**.

This method organizes knowledge into small **sub-structures** so that it can be used easily when needed.  
It actually originated from **Semantic Networks** and is now used in the form of **Classes** and **Objects** in Object-Oriented Programming (OOP).

### 🔍 What are Facets?

Facets are additional properties or conditions of a Slot.

* Example: **IF-NEEDED Facet** → The information of a Slot will only be retrieved when it is required.

A Slot can have multiple Facets, and each Facet can have multiple values.

### 📝 Examples

**Example 1 – Frame for a Book:**

| **Slot** | **Value** |
| --- | --- |
| Title | Artificial Intelligence |
| Genre | Computer Science |
| Author | Peter Norvig |
| Edition | Third Edition |
| Year | 1996 |
| Page | 1152 |
|  |  |
| Name | Peter |
| Profession | Doctor |
| Age | 25 |
| Marital Status | Single |
| Weight | 78 |

### ✅ Advantages

* Makes programming easier by keeping related information together.
* **Flexible** → Easy to add new Slots or Attributes.
* Easy to assign default values.
* Easy to handle missing values.
* Easy to understand and visualize.

### ❌ Disadvantages

* Reasoning (inference) is not straightforward.
* Does not work smoothly for large-scale reasoning.
* The method is quite **generalized** in nature.

## 📌 What is ****Production Rules****?

A **Production Rule system** is a type of knowledge representation method where rules are stored in **(Condition, Action)** pairs.  
Meaning: **"If the condition is true, then perform the action"** (IF condition → THEN action).

It is the foundation of **Rule-based Systems**.

### 🔹 Main Parts

1. **Set of Production Rules**
   * Contains the list of all rules (e.g., IF-THEN statements).
2. **Working Memory**
   * Stores the information about the current state or situation.
3. **Recognize-Act Cycle**
   * Recognizes the conditions and performs the corresponding actions.

### 🔄 How it Works

1. The **agent** takes the current state from the **Working Memory**.
2. If a rule’s condition matches, that rule is **fired** and its action is performed.
3. Performing the action may create a new state.
4. If the new state matches another rule’s condition, that rule will also fire.

### ⚠️ Conflict Set & Conflict Resolution

* Sometimes, more than one rule can match at the same time.
* The process of choosing which rule to execute first is called **Conflict Resolution**.

### 📝 Example

* IF (at bus stop AND bus arrives) THEN get into the bus
* IF (on the bus AND paid AND empty seat) THEN sit down
* IF (on bus AND unpaid) THEN pay charges
* IF (bus arrives at destination) THEN get down from the bus

### ✅ Advantages

* Can be written in natural language, making it easy to understand.
* Very modular → rules can be easily added, removed, or modified.

### ❌ Disadvantages

* No learning capability → cannot remember previous solutions.
* If too many rules are active at once, the system becomes slow.

Propositional Logic in Artificial Intelligence

## 📌 What is ****Propositional Logic****?

**Propositional Logic** is a type of logic where we use **propositions** (statements) to express whether something is **true** or **false**.  
Each proposition must be either **true** or **false** — nothing in between.

It is used in **Artificial Intelligence (AI)** for **knowledge representation**, **reasoning**, and **deriving new information** (**inference**).

Syntax of propositional Logic:

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

* Atomic Propositions
* Compound propositions

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

**Example 1:**

1. 2+2 is 4; it is an atomic proposition as it is a fact.
2. "The Sun is cold" is also a proposition as it is a false fact.

**Compound proposition:** Compound propositions are constructed by combining simpler or atomic propositions, using parenthesis and logical connectives.

**Example 2:**

1. "It is raining today, and the street is wet."
2. "Ankit is a doctor, and his clinic is in Mumbai."

Logical Connectives

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

* **Negation:** A sentence such as ¬ P is called negation of P. A literal can be either Positive literal or negative literal.
* **Conjunction:** A sentence that has ∧ a 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.

* **Disjunction:** A sentence that has ∨ a connective, such as P ∨, 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.

* **Implication:**A sentence such as P → Q is called an implication. Implications are also known as if-then rules. It can be represented as

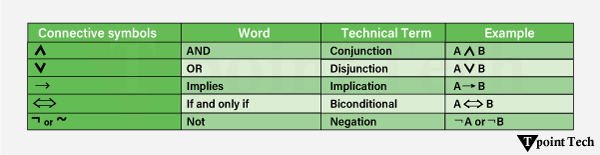
If it is raining, then the street is wet.

Let P= It is raining, and Q= Street is wet, so it is represented as P → Q

* **Biconditional:** A sentence such as P⇔ Q is a Biconditional sentence example: If I am breathing, then I am alive

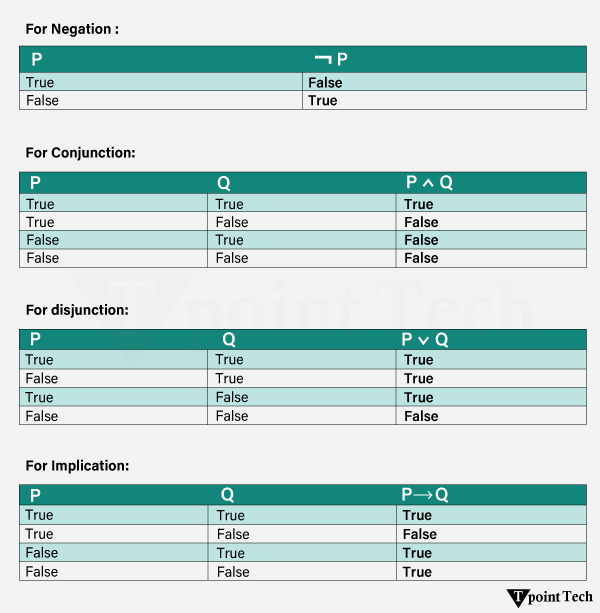
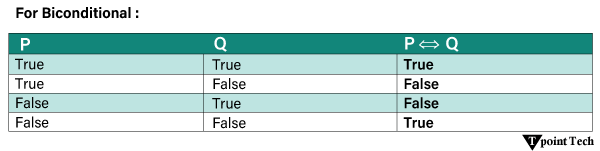
P= I am breathing, Q= I am alive, it can be represented as P ⇔ Q.

Following is the summarized table for Propositional Logic Connectives:



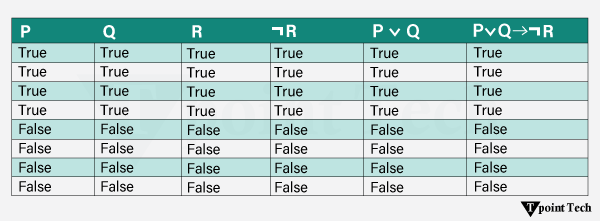
Truth Table

In propositional logic, we need to know the truth values of propositions in all possible scenarios. We can combine all the possible combinations with logical connectives, and the representation of these combinations in a tabular format is called a Truth table. Following is the truth table for all logical connectives:

Truth Table with Three Propositions

We can build a proposition composing three propositions: P, Q, and R. This truth table is made up of 8n Tuples as we have taken three proposition symbols.



Precedence of connectives:

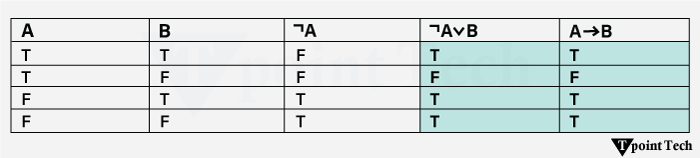
Just like arithmetic operators, there is a precedence order for propositional connectors or logical operators. This order should be followed while evaluating a propositional problem. Following is the list of the precedence order for operators:

|  |  |
| --- | --- |
| **Precedence** | **Operators** |
| First Precedence | Parenthesis |
| Second Precedence | Negation |
| Third Precedence | Conjunction(AND) |
| Fourth Precedence | Disjunction(OR) |
| Fifth Precedence | Implication |
| Six Precedence | Biconditional |

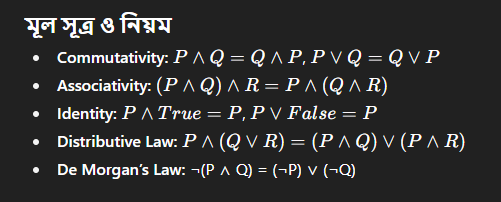
Logical Equivalence

Propositional logic is one of the features that have logical equivalence. The definition of logical equivalence is given by saying two propositions are logically equivalent if and only if the columns of a truth table are the same.

Assuming two propositions, A and B, we'll mark it by A⇔B, and this is the logical equivalence. From the below truth table, we see that the columns of ¬A∨ B and B→A are identical, so A is Equivalent to B.



Properties of Operators

****

**Commutativity:**

P∧ Q= Q ∧ P, or

P ∨ Q = Q ∨ P.

**Associativity:**

(P ∧ Q) ∧ R= P ∧ (Q ∧ R),

(P ∨ Q) ∨ R= P ∨ (Q ∨ R)

**Identity element:**

P ∧ True = P,

P ∨ True= True.

**Distributive:**

P∧ (Q ∨ R) = (P ∧ Q) ∨ (P ∧ R).

P ∨ (Q ∧ R) = (P ∨ Q) ∧ (P ∨ R).

**DE Morgan's Law:**

¬ (P ∧ Q) = (¬P) ∨ (¬Q)

¬ (P ∨ Q) = (¬ P) ∧ (¬Q).

**Double-negation elimination:**

¬ (¬P) = P.

### ****Uses of Propositional Logic in AI****

#### **1. Knowledge Representation**

* Propositional Logic is used to store information in a structured way.
* This makes it easier to store and process facts about the world.
* **Example:** In a knowledge-based system, the knowledge base contains propositions and logical rules.

#### **2. Problem Solving & Planning**

* In AI planning, propositional logic helps arrange steps according to the goal.
* **Example:** In the STRIPS planning system, actions are defined using preconditions and effects in logical form.

#### **3. Decision Making**

* When applied correctly, it helps choose the best solution among different options.
* Decision rules can be defined using logical rules or truth tables.

#### **4. Natural Language Processing (NLP)**

* Sentences are converted into logical form (**semantic parsing**).
* This helps understand the meaning of a sentence and apply reasoning to reach new conclusions.

### ****Important Facts About Propositional Logic****

* Also called **Boolean Logic**, because it works with 0 and 1 (True/False).
* Propositions are usually represented by symbols like **A, B, C, P, Q, R**, etc.
* A proposition can be True or False, but not both at the same time.
* It consists of **objects**, **relations**, **functions**, and **logical connectives**.
* Logical connectives (or operators) join two or more statements.
* If a proposition is always True, it is called a **Tautology** (Valid Sentence).
* If a proposition is always False, it is called a **Contradiction**.
* Questions, commands, or opinions (e.g., "How are you?", "What is your name?") are **not** propositions.

### ****Limitations of Propositional Logic****

* **Limited expression** – Cannot express general rules like “All humans are mortal.”
* **Scalability problem** – As the number of propositions grows, the truth table becomes very large.
* **Restricted reasoning** – Works only with True/False, cannot handle probabilities or uncertainty.
* **No quantifiers** – Unlike Predicate Logic, it cannot use ∀ (for all) or ∃ (there exists).
* **Cannot handle uncertainty** – Struggles with incomplete or probabilistic information.
* **Lacks context understanding** – Cannot interpret meaning or context in complex situations.
* Cannot express relationships like **ALL**, **SOME**, or **NONE**.

# Rules of Inference in Artificial intelligence

17 Mar 2025 | 3 min read

## Inference:

In artificial intelligence, we need intelligent computers which can create new logic from old logic or by evidence, **so generating the conclusions from evidence and facts is termed as Inference**.

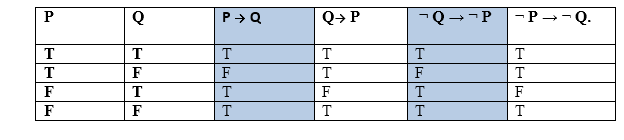
## Inference rules:

Inference rules are the templates for generating valid arguments. Inference rules are applied to derive proofs in artificial intelligence, and the proof is a sequence of the conclusion that leads to the desired goal.

In inference rules, the implication among all the connectives plays an important role. Following are some terminologies related to inference rules:

* **Implication:** It is one of the logical connectives which can be represented as P → Q. It is a Boolean expression.
* **Converse:** The converse of implication, which means the right-hand side proposition goes to the left-hand side and vice-versa. It can be written as Q → P.
* **Contrapositive:** The negation of converse is termed as contrapositive, and it can be represented as ¬ Q → ¬ P.
* **Inverse:** The negation of implication is called inverse. It can be represented as ¬ P → ¬ Q.

From the above term some of the compound statements are equivalent to each other, which we can prove using truth table:



Hence from the above truth table, we can prove that P → Q is equivalent to ¬ Q → ¬ P, and Q→ P is equivalent to ¬ P → ¬ Q.

## Types of Inference rules:

### 1. Modus Ponens:

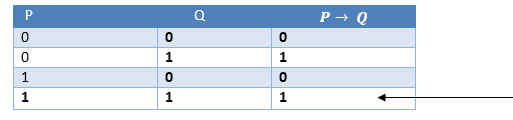
The Modus Ponens rule is one of the most important rules of inference, and it states that if P and P → Q is true, then we can infer that Q will be true. It can be represented as:

Rules of Inference in Artificial intelligence

**Example:**

Statement-1: "If I am sleepy then I go to bed" ==> P→ Q  
Statement-2: "I am sleepy" ==> P  
Conclusion: "I go to bed." ==> Q.  
Hence, we can say that, if P→ Q is true and P is true then Q will be true.

**Proof by Truth table:**



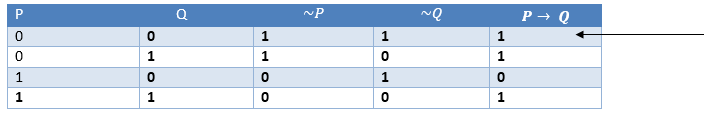
### 2. Modus Tollens:

The Modus Tollens rule state that if P→ Q is true and **¬ Q is true, then ¬ P** will also true. It can be represented as:

Rules of Inference in Artificial intelligence

**Statement-1:** "If I am sleepy then I go to bed" ==> P→ Q  
**Statement-2:** "I do not go to the bed."==> ~Q  
**Statement-3:** Which infers that "**I am not sleepy**" => ~P

**Proof by Truth table:**



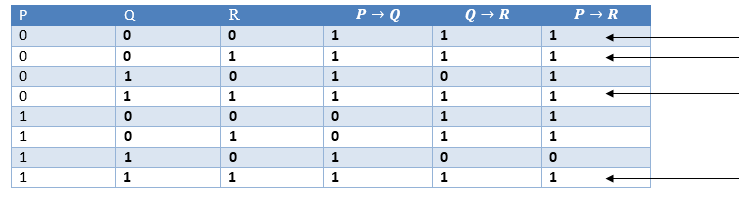
### 3. Hypothetical Syllogism:

The Hypothetical Syllogism rule state that if P→R is true whenever P→Q is true, and Q→R is true. It can be represented as the following notation:

**Example:**

**Statement-1:** If you have my home key then you can unlock my home. **P→Q**  
**Statement-2:** If you can unlock my home then you can take my money. **Q→R**  
**Conclusion:** If you have my home key then you can take my money. **P→R**

**Proof by truth table:**



### 4. Disjunctive Syllogism:

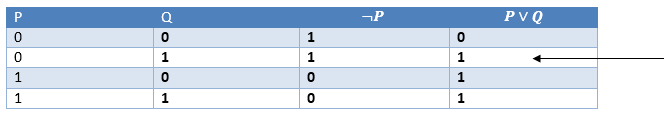
The Disjunctive syllogism rule state that if P∨Q is true, and ¬P is true, then Q will be true. It can be represented as:

Rules of Inference in Artificial intelligence

**Example:**

**Statement-1:** Today is Sunday or Monday. ==>P∨Q  
**Statement-2:** Today is not Sunday. ==> ¬P  
**Conclusion:** Today is Monday. ==> Q

**Proof by truth-table:**



### 5. Addition:

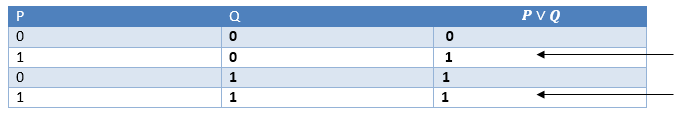
The Addition rule is one the common inference rule, and it states that If P is true, then P∨Q will be true.

Rules of Inference in Artificial intelligence

**Example:**

**Statement:** I have a vanilla ice-cream. ==> P  
**Statement-2:** I have Chocolate ice-cream.  
**Conclusion:** I have vanilla or chocolate ice-cream. ==> (P∨Q)

**Proof by Truth-Table:**

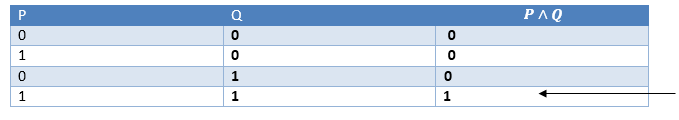


### 6. Simplification:

The simplification rule state that if **P∧ Q** is true, then **Q or P** will also be true. It can be represented as:

Rules of Inference in Artificial intelligence

**Proof by Truth-Table:**

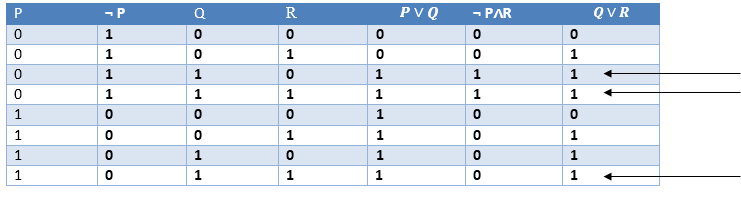


### 7. Resolution:

The Resolution rule state that if P∨Q and ¬ P∧R is true, then Q∨R will also be true. **It can be represented as**

Rules of Inference in Artificial intelligence

**Proof by Truth-Table:**



# The Wumpus World in Artificial intelligence

17 Mar 2025 | 5 min read

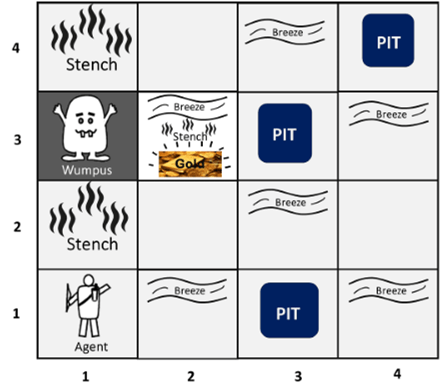
## Wumpus world:

The Wumpus world is a simple world 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 Wumpus world, there are some Pits 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. So 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.

#### Note: Here Wumpus is static and cannot move.

Following is a sample diagram for representing the Wumpus world. It is showing some rooms with Pits, one room with Wumpus and one agent at (1, 1) square location of the world.



**There are also some components which can help the agent to navigate the cave. These components are given as follows:**

1. The rooms adjacent to the Wumpus room are smelly, so that it would have some stench.
2. The room adjacent to PITs has a breeze, so if the agent reaches near to PIT, then he will perceive the breeze.
3. There will be glitter in the room if and only if the room has gold.
4. 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.

## PEAS description of Wumpus world:

To explain the Wumpus world we have given PEAS description as below:

### Performance measure:

* +1000 reward points if the agent comes out of the cave with the gold.
* -1000 points penalty for being eaten by the Wumpus or falling into the pit.
* -1 for each action, and -10 for using an arrow.
* The game ends if either agent dies or came out of the cave.

### Environment:

* A 4\*4 grid of rooms.
* The agent initially in room square [1, 1], facing toward the right.
* Location of Wumpus and gold are chosen randomly except the first square [1,1].
* Each square of the cave can be a pit with probability 0.2 except the first square.

### ****Actuators (এজেন্টের কার্যকরী অঙ্গ)****

এজেন্ট যেসব কাজ করতে পারে:

* **Left turn** – বাম দিকে ঘোরা
* **Right turn** – ডান দিকে ঘোরা
* **Move forward** – সামনে যাওয়া
* **Grab** – জিনিস তোলা (যেমন Gold)
* **Release** – নামানো
* **Shoot** – তীর ছোড়া

### Sensors:

* The agent will perceive the **stench** if he is in the room adjacent to the Wumpus. (Not diagonally).
* The agent will perceive **breeze** if he is in the room directly adjacent to the Pit.
* The agent will perceive the **glitter** in the room where the gold is present.
* The agent will perceive the **bump** if he walks into a wall.
* When the Wumpus is shot, it emits a horrible **scream** which can be perceived anywhere in the cave.
* These percepts can be represented as five element list, in which we will have different indicators for each sensor.
* Example if agent perceives stench, breeze, but no glitter, no bump, and no scream then it can be represented as:  
  **[Stench, Breeze, None, None, None]**.

## The Wumpus world Properties:

* **Partially observable:** The Wumpus world is partially observable because the agent can only perceive the close environment such as an adjacent room.
* **Deterministic:** It is deterministic, as the result and outcome of the world are already known.
* **Sequential:** The order is important, so it is sequential.
* **Static:** It is static as Wumpus and Pits are not moving.
* **Discrete:** The environment is discrete.
* **One agent:** The environment is a single agent as we have one agent only and Wumpus is not considered as an agent.

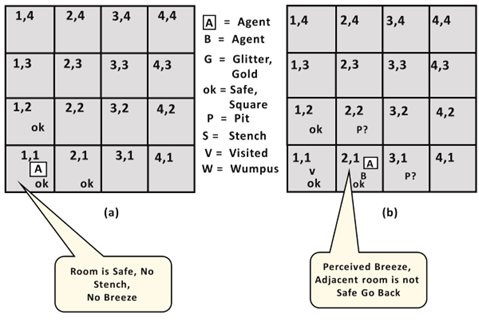
## Exploring the Wumpus world:

Now we will explore the Wumpus world and will determine how the agent will find its goal by applying logical reasoning.

**Agent's First step:**

Initially, the agent is in the first room or on the square [1,1], and we already know that this room is safe for the agent, so to represent on the below diagram (a) that room is safe we will add 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 OK.



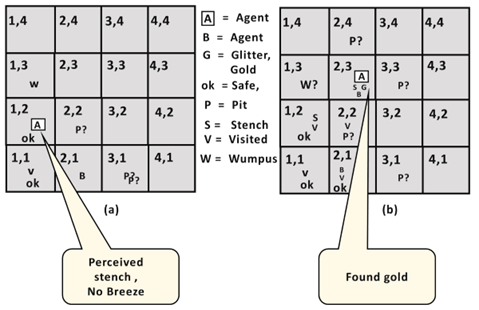
**Agent's second Step:**

Now agent needs to move forward, so it will either move to [1, 2], or [2,1]. Let's 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 the [1, 1] room. The room [1,1], and [2,1] are visited by the agent, so we will use symbol V to represent the visited squares.

**Agent's third step:**

At the third step, now agent will move to the room [1,2] which is OK. 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 current state, there is no breeze which means in [2,2] there is no Pit and no Wumpus. So it is safe, and we will mark it OK, and the agent moves further in [2,2].



**Agent's fourth step:**

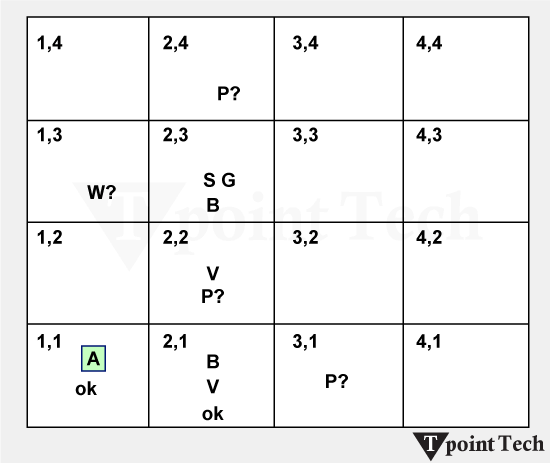
At room [2,2], here no stench and no breezes present so let's suppose agent decides to move to [2,3]. At room [2,3] agent perceives glitter, so it should grab the gold and climb out of the cave.

# Knowledge Base for the Wumpus World

10 Jun 2025 | 7 min read

As in the previous topic, we have learned about the wumpus world and how a knowledge-based agent evolves the world. Now, in this topic, we will create a knowledge base for the Wumpus world and will derive some proof for the Wumpus world using propositional logic.

The agent starts visiting from the first square [1, 1], and we already know that this room is safe for the agent. To build a knowledge base for the Wumpus world, we will use some rules and atomic propositions. We need a symbol [i, j] for each location in the wumpus world, where i is for the location of rows, and j is for the column location.

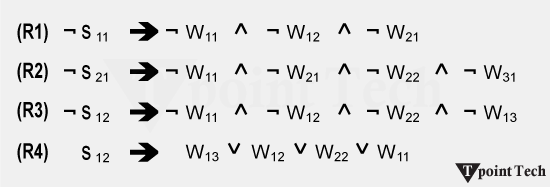


Atomic proposition variable for the Wumpus world:

* Let **P**i,j be true if there is a Pit in the room [i, j].
* Let **B**i,j be true if the agent perceives a breeze in [i, j] (dead or alive).
* Let **W**i,j be true if there is a wumpus in the square[i, j].
* Let **S**i,j be true if the agent perceives a stench in the square [i, j].
* Let **V**i,j be true if that square[i, j] is visited.
* Let **G**i,j be true if there is gold (and glitter) in the square [i, j].
* Let **OK**i,j be true if the room is safe.

#### Note: For a 4 \* 4 square board, there will be 7\*4\*4= = 112 propositional variables.

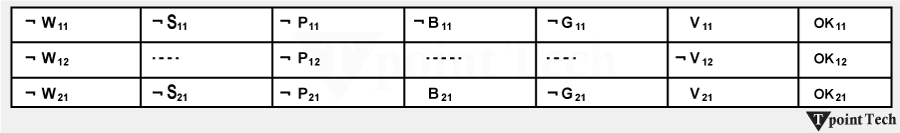
## Some Propositional Rules for the Wumpus World



#### Note: lack of variables gives us similar rules for each cell.

### Representation of Knowledgebase for Wumpus World

Following is the Simple KB for the wumpus world when an agent moves from room [1, 1] to room [2,1]:



Here in the first row, we have mentioned propositional variables for room[1,1], which shows that the room does not have wumpus(¬ W11), no stench (**¬S**11), no Pit(**¬P**11), no breeze(**¬B**11), no gold (**¬G**11), visited (**V**11), and the room is Safe(**OK**11).

In the second row, we have mentioned propositional variables for room [1,2], which shows that there is no wumpus, stench, and breeze are unknown as an agent has not visited room [1,2], no Pit, not visited yet, and the room is safe.

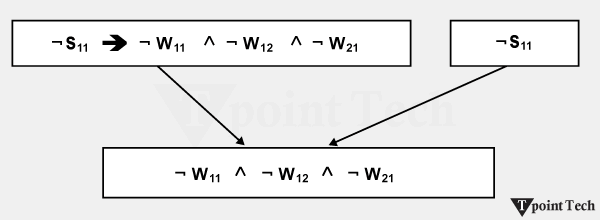
In the third row, we have mentioned the propositional variable for room[2,1], which shows that there is no wumpus(**¬ W**21), no stench (**¬S**21), no Pit (**¬P**21), Perceives breeze(**B**21), no glitter(**¬G**21), visited (**V**21), and room is safe (**OK**21).

## Prove that Wumpus is in the room (1, 3)

We can prove that the wumpus is in the room (1, 3) using propositional rules that we have derived for the wumpus world and using the inference rule.

Apply Modus Ponens with ¬S11 and R1:

We will firstly apply MP rule with R1 which is ¬S11 → ¬ W11 ^ ¬ W12 ^ ¬ W21, and ¬S11 which will give the output ¬ W11 ^ W12 ^ W12.



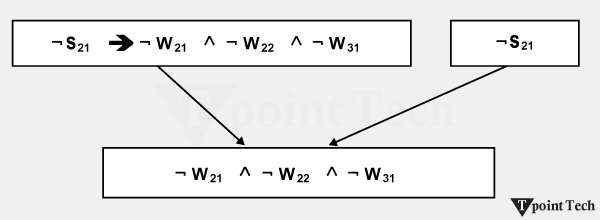
**Apply And-Elimination Rule:**

After applying And-elimination rule to ¬ W11 ∧ ¬ W12 ∧ ¬ W21, we will get three statements:

1. ¬ W11, ¬ W12, and ¬W21

**Apply Modus Ponens to ¬S21, and R2:**

Now we will apply Modus Ponens to ¬S21 and R2 which is ¬S21 → ¬ W21 ∧¬ W22 ∧ ¬ W31, which will give the Output as ¬ **W21** ∧ ¬ **W22** ∧¬ **W31**



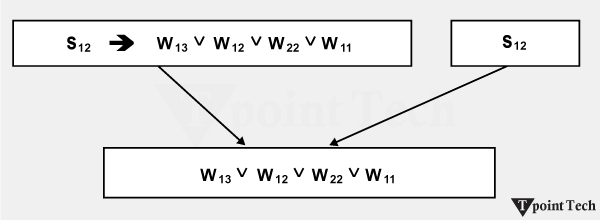
**Apply -Elimination Rule:**

Now again apply And-elimination rule to **¬ W21 ∧ ¬ W22 ∧¬ W31**, We will get three statements:

1. ¬ W21, ¬ W22, and ¬ W31

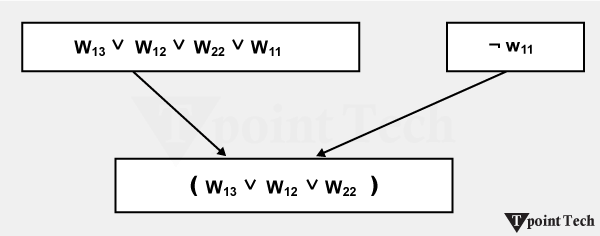
**Apply MP to S12 and R4:**

Apply Modus Ponens to **S12** and **R4**, which is **S12 → W13 ∨. W12 ∨. W22 ∨.W11**, we will get the output as **W13∨ W12 ∨ W22 ∨.W11.**



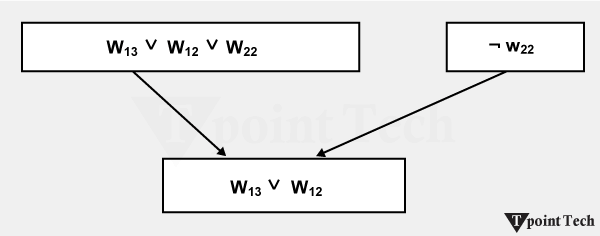
**Apply Unit resolution on W13 ∨ W12 ∨ W22 ∨W11 and ¬ W11 :**

After applying the Unit resolution formula on W13 ∨ W12 ∨ W22 ∨W11 and ¬ W11 we will get W13 ∨ W12 ∨ W22.



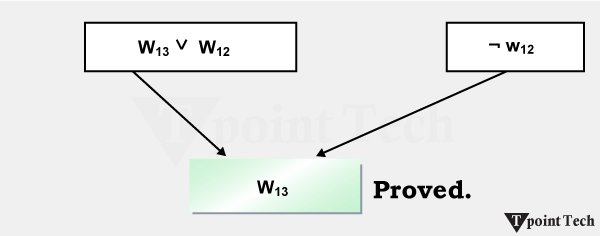
**Apply Unit resolution on W13 ∨ W12 ∨ W22 and ¬ W22:**

After applying Unit resolution on **W13 ∨ W12 ∨ W22, and ¬W22**, we will get **W13 ∨ W12** as output.



**Apply Unit Resolution on W13 ∨ W12 and ¬ W12 :**

After Applying Unit resolution on **W13 ∨ W12 and ¬ W12**, we will get **W13** as an output, hence it is proved that the Wumpus is in the room [1, 3].



## Building the Knowledge Base for the Wumpus World

### Initial Knowledge Setup

**Encoding the Starting State of the Wumpus World**

In the Wumpus World, we start with a predefined grid environment and a certain scheme of putting the agent, the Wumpus, the pits, and the gold. Logical representations of this setup should be included in the initial state of the knowledge base.

**Agent's Position:**

The agent initially begins his journey from the bottom left corner, say Room [1,1].

**Logical Representation:** At(Agent, [1,1]).

**Wumpus and Pits:**

At the start, they are unknown where the Wumpus and pits are, but the existence of the Wumpus and pits is inferred from percepts like "stench" and "breeze."

**Example:** ¬Pit([1,1]) (No pit in the starting room).

**Gold:**

At first, nothing is known about the location of the gold.

**Example:** ¬Gold([1,1]).

### Representing Safe and Unsafe Zones

The agent is allowed to move in safe zones without stepping on a Wumpus or falling into a pit. Inferred unsafe zones based on percepts received from adjacent rooms.

**Safe Zone Representation:**

**Logical Representation:** Safe([x,y]) ⇔ ¬Pit([x,y]) ∧ ¬Wumpus([x,y]).

**Unsafe Zone Indicators:**

The ceiling breezes in a room mean a pit in one of the adjoining rooms.

**Example:** Breeze([x,y]) ⇒ Pit([x±1,y]) ∨ Pit([x,y±1]).

A Wumpus is in one of the adjoining rooms if there is a room with a stench.

**Example:** Stench([x,y]) ⇒ Wumpus([x±1,y]) ∨ Wumpus([x,y±1]).

### Dynamic Knowledge Updates

**Updating Knowledge Based on Percepts**

The temple's knowledge base of the environment pertains to the dynamic agent's perceived environment. Every time a percept happens (e.g., breeze, stench, glitter) updates fire:

**Example of Percept Processing:**

However, if a stench is detected in [2,2], the knowledge base will include:

* Stench([2,2])
* Wumpus([1,2]) ∨ Wumpus([2,1]) ∨ Wumpus([2,3]) ∨ Wumpus([3,2]).

**Safe Zone Identification:**

All adjacent rooms are marked as safe if no percepts are detected in them in Room [1,1].

1. ¬Stench([1,1]) ∧ ¬Breeze([1,1]) ⇒ Safe([1,2]) ∧ Safe([2,1]).

### Handling Uncertainty in the Environment

When there are multiple hypotheses for a percept, uncertainty occurs. This can be resolved by [Probabilistic reasoning](https://www.tpointtech.com/probabilistic-reasoning-in-artifical-intelligence) or prioritized exploration:

**Ambiguity Example:**

* When there is a stench in Room [2,2], there can be several possible places where the Wumpus can be.
* Logical Representation: Wumpus([1,2]) ∨ Wumpus([2,3]) ∨ Wumpus([3,2]).
* However, the agent has to gather more information by exploring adjacent rooms.

**Key Logical Statements**

**Examples of Logical Sentences in the Knowledge Base**

Knowledge base [include] logical rules that govern the agent's reasoning and actions:

* **Safe Movement:** Safe([x,y]) ⇒ Move(Agent, [x,y]).
* **Avoid Unsafe Zones:** ¬Safe([x,y]) ⇒ ¬Move(Agent, [x,y]).
* **Gold Detection:** Glitter([x,y]) ⇒ Gold([x,y]).
* **Shooting the Wumpus:** Wumpus([x,y]) ⇒ Shoot(Agent, [x,y]).

**Practical Example:**

The Agent detects a stench in Room [2,1].

* Add: Stench([2,1]).
* Infer: Wumpus([1,1]) ∨ Wumpus([2,2]) ∨ Wumpus([3,1]).

The Agent detects no stench in Room [3,1].

* Add: ¬Stench([3,1]).
* Infer: ¬Wumpus([2,1]).

It is a structured approach to the evolution of the knowledge base so as to have the agent traverse the Wumpus World efficiently while modelling uncertainties.

## Applications of the Wumpus World Knowledge Base

**Logical Representation:**

The Wumpus World is an exciting domain with the environment being a grid with specific rules and hazards that allow for using logical statements encoding facts and regulations. For instance, we can use propositional logic to denote conditions like if there is a stench in a room, then there is a possibility of the Wumpus nearby.

**Inference Mechanisms:**

In particular, the knowledge base shows how, through employing forward chaining, backward chaining, and resolution, AI systems can deduce new facts from existing facts.

**Uncertainty and Incompleteness:**

The Wumpus World also models situations without complete information. AI [applications](https://www.tpointtech.com/applications-of-artificial-intelligence) such as medical diagnosis and risk analysis need to know how to deal with uncertainty using probabilities or assumptions, and this knowledge is taught in the knowledge base.

### Developing Problem-Solving Strategies

**Goal-Oriented Search:**

Using its knowledge base, the agent in the Wumpus World selects its safe paths, spots the gold, and exits the grid, avoiding the hazards. This respects the idea of goal-oriented search strategies such as [breadth-first search](https://www.tpointtech.com/breadth-first-search-in-python), depth-first search, and A\* algorithms.

**Heuristic Development:**

The agent will be able to optimize its actions by incorporating heuristics in its decision-making process, such that there is as much exploration as safety. For instance, a heuristic might suggest that rooms with minimum perceived risk, according to the currently available knowledge, need to be explored first.

**Iterative Decision-Making:**

The agent deals with dynamic updates to the knowledge base so that the agent can modify its strategy based on new available information. Real-world problem solving with game AI or [robotics](https://www.tpointtech.com/robotics-tutorial) navigation uses this iterative approach to problem solving.

### Enhancing Autonomous Agent Design

**Environment Awareness:**

Then, the agent constructs a mental model of the environment by utilising percepts (smells, breezes, glitter, etc. ). A key aspect of situational awareness in dynamic real-world robotics and self-driving cars is watching their surroundings.

**Decision Autonomy:**

The decision of an agent, having or not having a sufficient knowledge base, determines whether it is autonomous. Logical rules and inference mechanisms demonstrated in the inference mechanisms of the Wumpus World serve as a backbone of such systems as automated drones and industrial robots.

**Risk Assessment:**

In the Wumpus World, agents are taught to consider the risks versus rewards. For example, upon entering a high-reward room, we may need to enter an adjacent room that is potentially dangerous. This also applies to financial AI systems that decide on investing risks or healthcare bots that decide on treatment options.

# First-Order Logic in Artificial Intelligence

10 Jun 2025 |  9 min read

**🧠 First-Order Logic in Artificial Intelligence**

**First-Order Logic (FOL)** is a powerful **knowledge representation formalism** that extends Propositional Logic.

**❌ Why is Propositional Logic not enough?**

In Propositional Logic, only whole sentences are treated as **true** or **false**, and we cannot reason about *parts* of those sentences.

For example, the following sentences **cannot be properly expressed** in Propositional Logic:

* “Some humans are intelligent”
* “Sachin likes cricket”

This is because these sentences involve **individual objects/entities** (like *Sachin*, *humans*) and **relations** between them (like *likes*, *are intelligent*), which Propositional Logic cannot handle.

### First-Order logic

First-order logic is another way of [knowledge representation](https://www.tpointtech.com/knowledge-representation-in-ai) in artificial intelligence. It is an extension of propositional logic. FOL is sufficiently expressive to represent the natural language statements concisely.

It is also known as predicate logic or first-order [predicate logic](https://www.tpointtech.com/predicate-logic). First-order logic is a powerful language that develops information about objects in an easier way and can also express the relationship between those objects.

First-order logic (like natural language) not only assumes that the world contains facts like propositional logic but also assumes the following things in the world:

* **Objects:** A, B, people, numbers, colours, wars, theories, squares, pits, wumpus, ......
* **Relations:** It can be a unary relation, such as red, round, is adjacent, or n-any relation, such as the sister of, brother of, has colour, comes between
* **Function:** Father of, best friend, third inning of, end of, ......

As a natural language, first-order logic also has two main parts:

1. Syntax
2. 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:

The 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** | ∀, ∃ |

## ****Semantics (Meaning)****

**Syntax** tells us how to write, whereas **semantics** tells us what that writing actually means.

Whether predicates are true or false depends on the **real world**.

By using **quantifiers**, we can specify whether we are talking about **all objects** or **only some objects**.

### 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).

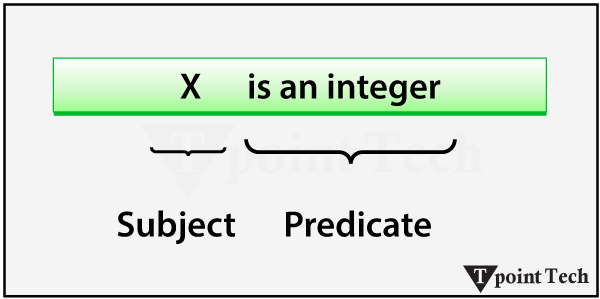
### Complex Sentences

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 that 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 the second part, "is an integer," is known as a predicate.



**Quantifiers in First-order logic:**

A quantifier is a language element that generates quantification, and quantification specifies the quantity of specimens in the universe of discourse. These are the symbols that permit the determination or identification of the range and scope of the variable in the logical expression. There are two types of [quantifiers](https://www.tpointtech.com/quantifiers-in-discrete-mathematics):

1. Universal Quantifier (for all, everyone, everything)
2. Existential quantifier (for some, at least one).

### Universal Quantifier

**The 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 ∀that resembles an inverted A.

#### **Note: In the universal quantifier, we use implication "→".**

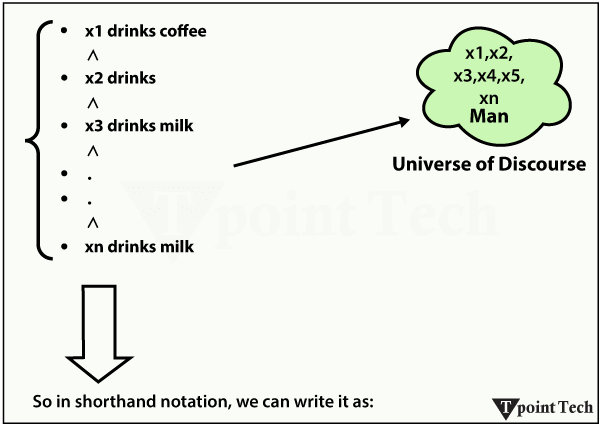
If x is a variable, then ∀x is read as:

* For all x
* For each x
* For every x

**Example**

All men drink coffee.

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



**∀x man(x) → drink (x, coffee).**

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

### Existential Quantifier

**Existential quantifiers are a type of quantifiers that 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 an inverted E. When it is used with a predicate variable, it is called an existential quantifier.

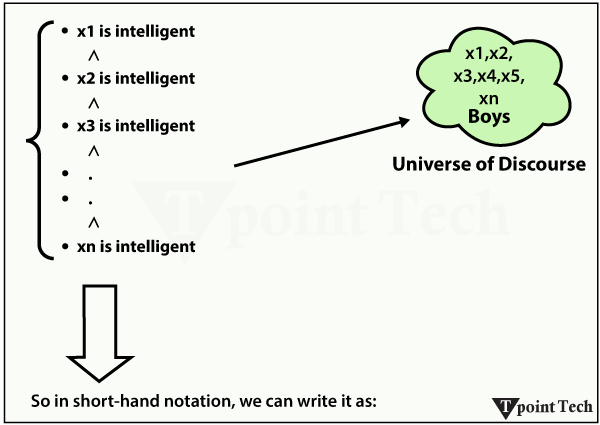
#### **Note: In the Existential quantifier, we always use AND or the Conjunction symbol (∧).**

If x is a variable, then the 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.



1. ∃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 the universal quantifier **∀** is implication **→**.
* The main connective for the existential quantifier **∃** is and **∧**.

**Properties of Quantifiers:**

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

**Example:**

Some Examples of FOL using quantifiers:

**1. All birds fly**

In this question, the predicate is "fly(bird)."

And since there are all birds who fly, it will be represented as follows:

1. ∀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:

1. ∀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 we will use ∃, and it will be represented as:

1. ∃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, we will use ∀ negation, so the following representation for:

**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, we will use the following representation for this:

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

## ✂️ এটাকে দুই ভাগে ভাঙলে সহজ হবে:

### ➊ ∃x … → “****কমপক্ষে একজন ছাত্র আছে**** যে Mathematics-এ ফেল করেছে”

👉 অর্থ: At least one student failed in math

### ➋ ∀y … → “****সেই ছাত্র ছাড়া অন্য কেউ (ইউনিক y ≠ x) যদি student হয় তাহলে সে ফেল করেনি****”

👉 অর্থ: No other student failed in math

## 🔁 মিলিয়ে দেখো:

| **অংশ** | **অর্থ** |
| --- | --- |
| ∃x | কোনো একজন x আছে |
| student(x) → failed(x, Mathematics) | সে একজন ছাত্র এবং ম্যাথে ফেল করেছে |
| ∀y | সকল y এর জন্য |
| ¬(x = y) ∧ student(y) → ¬failed(y, Mathematics) | যদি y ≠ x এবং y স্টুডেন্ট হয় → তাহলে y ম্যাথে ফেল করেনি |

### Free and Bound Variables:

The quantifiers interact with variables that appear suitably. 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**

1. ∀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**

1. ∀x [A (x) B( y)]

Here, x and y are the bound variables.

## Applications of First-Order Logic in Artificial Intelligence

### Knowledge Representation and Reasoning

**Role in AI:**

For a simple, clear, logical, structured representation of real-world knowledge, FOL is a robust framework. It can encode facts, relationships, and rules about facts about entities that are about such a domain.

**Example:**

Familiar relationships represented in the knowledge base:

* Facts: Parent(John, Mary)
* Rule: ∀x ∀y (Parent(x, y) → Ancestor(x, y))

**Reasoning:**

It allows for deriving conclusions from known facts and rules. For instance, if we have Parent(John, Mary) and the rule above, then the system will be able to infer that Ancestor(John, Mary).

**Use Case:**

Creating systems that could make intelligent decisions themselves ¬- in other words, diagnostic systems in medicine or fraud detection in finance, for example.

### Natural Language Processing (NLP)

**Role in AI:**

FOL formalises the constructs of natural language into logical representations and hence helps in understanding and processing natural language.

**Example:**

* Sentence: "Every student in the class has submitted the assignment."
* Logical Form: ∀x (Student(x) → Submitted(x, Assignment))

**Applications:**

* Semantic parsing: Deriving logic that a machine can understand from natural language.
* Question answering systems: Matching questions with knowledge base facts using FOL.

**Use Case:**

One good example of a couple of people deploying such types of reasoning is Virtual Assistants like [Siri](https://www.tpointtech.com/siri) and [Google Assistant](https://www.tpointtech.com/google-assistant-app-for-android), which reason from user queries through the principles of FOL.

### Semantic Web Technologies

**Role in AI:**

FOL is used to underpin ontologies and rules to the effect that the relationship between web entities is specified.

**Example:**

In turn, FOL creates structured, machine-readable web content by means of (especially) RDF (Resource Description Framework) and OWL (Web Ontology Language) ontologies.

* Fact: Book(Book1) ∧ Author(Book1, "AuthorName")
* Rule: ∀x (Book(x) → HasPublisher(x, "DefaultPublisher"))

**Applications:**

* Intelligent search engines: Understanding relationships of concepts, which will affect search.
* Data integration: To combine different datasets via logical reasoning.

**Use Case:**

Though the knowledge graph setting of Google, or the linked open data effort, is the most prominent application, FOL is being used for meaningful information retrieval.

### Expert Systems

**Role in AI:** Knowledge in some particular domain is represented in an FOL, and then solutions for problems are inferred through logical reasoning in the FOL.

**Example:** A medical diagnostic expert system:

* Knowledge Base:
  + Fact: Symptom(John, Fever)
  + Rule: ∀x (Symptom(x, Fever) ∧ Symptom(x, Cough) → Diagnosis(x, Flu))
* Reasoning: If we know fact Symptom(John, Fever) and Symptom(John, Cough), the system infers Diagnosis(John, Flu).

**Applications:**

* Healthcare: Assisting doctors with diagnoses.
* Engineering: Troubleshooting and system maintenance.

**Use Case:**

The use of FOL was known in MYCIN, a famous early expert system used for the diagnosis of bacterial infections.

### Automated Theorem Proving

**Role in AI:** FOL is used within automated theorem proving to formalise and then prove mathematical theorems or logical assertions.

**Example:** Proving a theorem:

* Hypothesis: ∀x (P(x) → Q(x))
* Given: P(a)
* Goal: Prove Q(a)
* Inference: Using resolution, the system concludes Q(a).

**Applications:**

* Verifying software correctness.
* Formal proofs of mathematical conjectures.

**Use Case:** For example, Coq and Prover9 use FOL to generate automated proofs.

## Limitations of First-Order Logic in Artificial Intelligence

### Decidability and Computational Complexity

**Decidability Issues:**

* We cannot create effective tests in FOL to find out whether a given statement is true in all models or not. It can be stated that there doesn't exist an algorithm in general that might work out the truth or falsity of every first-order statement.
* For example, a logical problem in FOL that has some structure so that the reasoning cycle leads to an infinite loop will require infinite time/resources to solve.

**Computational Complexity:**

* In some cases in which it's possible to find a solution, the time required to solve the problem may grow exponentially worse than the size of the problem.
* With an increasingly complex domain, the cost of the task with respect to both time and memory grows.

### Expressiveness for Certain Real-World Problems

**Temporal and Dynamic Aspects:**

* Reasoning about temporal or sequential things doesn't play too well with FOL. For example, it is not simple to say, "If an event occurs, B event should run after 10 minutes."
* However, temporal reasoning necessitates, for example, temporal logic or higher-order logic.

**Continuous Domains:**

Many real-world problems deal with continuous variables (e.g., physics-based systems and machine learning models). The real-world domains in which these problems live are constant; FOL, as is, simply does not work well there.

**Nested and Self-Referencing Statements:**

FOL can be difficult (or impossible) to describe the composition of complex relationships with self-references or nested conditions. For example, the sentence 'this statement is false' is a logical paradox.

### Limitations in Representing Uncertain or Probabilistic Knowledge

**Deterministic Nature:**

On the other hand, FOL is a deterministic world wherein if you can prove something that just isn't the case true, then it is true; vice versa, it is also false. Unfortunately, this binary approach cannot be applied to all aspects of uncertainty since many AI applications easily show a lack of ambiguity and partial truth.

For example, in medical diagnosis, each symptom gives rise to several potential diseases, each with differing probabilities - a structure that's too complex for FOL to represent without extra mechanisms.

**Lack of Probabilistic Framework:**

Crucial [probabilistic reasoning](https://www.tpointtech.com/probabilistic-reasoning-in-artifical-intelligence), machine learning, decision-making under uncertainty, etc, depend on it. Unfortunately, FOL doesn't handle concepts like probability natively.

**Uncertainty in Knowledge Representation:**

For example, in natural language processing or social systems, information is either incomplete or uncertain in multiple domains. Yet FOL is unsuitable when information is partial, fuzzy, or interpretation-dependent.

# Knowledge Engineering in First-order logic

23 May 2025 |  8 min read

### What is Knowledge Engineering?

The process of constructing a knowledge base in first-order logic is called knowledge engineering. In knowledge engineering, someone who investigates a particular domain, learns important concepts of that domain, and generates a formal representation of the objects is known as a knowledge engineer.

In this topic, we will understand the Knowledge engineering process in the electronic circuit domain, which is already familiar. This approach is mainly suitable for creating a special-purpose knowledge base.

### The Knowledge Engineering Process

An important area of [AI](https://www.tpointtech.com/artificial-intelligence-ai) and expert systems is knowledge engineering, a domain in which an intelligent [computer system](https://www.tpointtech.com/what-is-a-computer-system) tries to mimic human expert behavior. For a digital circuit of some sort, such as a one-bit full adder, the knowledge engineer must first grasp what the circuit does and the components involved: inputs A, B, and Carry-in; gates (AND, OR, XOR); and corresponding outputs: Sum and Carry-out. This clear understanding of the knowledge must then be formalized through first-order logic.

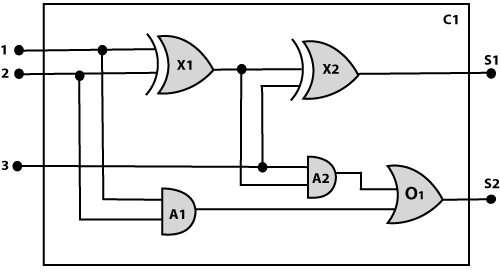
Such formal representation is being utilized in automated reasoning systems to conclude or detect inconsistencies, to predict output values for any given set of inputs, or for fault detection in circuit design. Various scenarios are conducted to test the knowledge base and validate whether the system will behave as expected.

Knowledge engineering also involves optimizing and simulating designs apart from circuit analysis. This ensures knowledge reusability in related domains and lessens manual expertise. The engineer attempts to represent the systems' understanding using logic and rules and simulate human understanding to improve decision-making accuracy in real-time applications.

Knowledge engineering finds application in numerous fields such as medical diagnosis, robotics, finance, and others. Its set methods are perfectly suited for expert-based decision-making applications. Hence, knowledge-based systems are thereby evolving, as the present knowledge base may be inadequate, and knowledge can be updated as the domain knowledge tends to grow. In support of the knowledge engineer, will also assist in building a rule-based system, an ontology, and semantic networks.

Finally, knowledge engineering looks to bridge the gap between human expertise and machine reasoning, enabling AI systems to intelligently decide and deal with changes while providing accurate outputs on highly complex problem domains.

The following are some main steps of the knowledge-engineering process. Using these steps, we will develop a knowledge base that will allow us to reason about a digital circuit (One-bit full adder), which is given below:



**1. Identify the Task:**

The first step of the process is to identify the task, and for the digital [circuit](https://www.tpointtech.com/electric-circuit-definition), there are various reasoning tasks.

At the first level or highest level, we will examine the functionality of the circuit:

* **Does the circuit add properly?**
* **What will the output of gate A2 be if all the inputs are high?**

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

* **Which gate is connected to the first input terminal?**
* **Does the circuit have feedback loops?**

Task identification is required, as it defines the extent and direction of the knowledge engineering process. With a clarifying understanding of the functional and structural questions, a full-fledged analysis follows. The functional tasks ask whether the circuit actually performs the intended operation properly. The structural tasks solve out the physical or logical placement of the element, reduce to another type of design flaw, optimize performance, or substantiate a behavior of the circuit.

These questions will also assist in documenting the knowledge systematically for reuse in AI training or simulation. This clarity facilitates efficient reasoning as well as smart troubleshooting within a circuit system.

**2. Assemble the Relevant Knowledge:**

In the second step, we will assemble the relevant knowledge that is required for digital circuits. So, for digital circuits, we have the following required knowledge:

* Logic circuits are made up of wires and gates.
* Signal flows through wires to the input terminal of the gate, and each gate produces the corresponding output, which flows further.
* In this logic circuit, four types of gates are used: AND, OR, XOR, and NOT.
* All these gates have one output terminal and two input terminals (except the NOT gate, which has one input terminal).

Gaining associated knowledge ensures the knowledge base is formed on accurate and complete information. Understanding how the signals propagate, how individual gates operate, and how things are connected is required in logically modeling the circuit. Familiarity with timing, delay of signals, and logic levels (high or low), besides rudimentary gate operation, can model the representation even more accurately.

Moreover, the identification of common sub-circuits, such as multiplexers or half-adders, can also be employed to simplify complex designs. Recording facts in a proper format suitable for first-order logic is also a part of this step, which makes inference work effectively and enables AI systems to perform accurate reasoning over the digital circuit.

**3. Decide on Vocabulary:**

The next step of the process is to select functions, predicates, and constants to represent the circuits, terminals, signals, and gates. Firstly, we will distinguish the gates from each other and from other objects. Each gate is represented as an object named by a constant, such as Gate(X1). 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 (C1).

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

For gate input, we will use the function In(1, X1) to denote the first input terminal of the gate, and for the output terminal, we will use Out (1, X1).

The function Arity(c, i, j) is used to denote that circuit c has i input, j output.

The connectivity between gates can be represented by the predicate Connect(Out(1, X1), In(1, X1)).

We use a unary predicate On (t), which is true if the signal at a terminal is on.

This conversion converts physical and logical structures of a circuit into formalized expressions that the reasoning system is able to read. The application of constants, predicates, and functions enables us to define advanced relationships exactly. For instance, by using functions such as In and Out to indicate inputs and outputs, signal flow can be graphically represented. By representing behavior, structure, and connectivity in logical forms, the system can infer circuit behavior for a variety of conditions.

In addition, formalization of this kind facilitates automatic verification, fault analysis, and performance assessment. Logical functions ensure that the knowledge base can be scaled and extended in such a manner as to include larger and more complicated digital circuits in an economical fashion.

**4. Encode General Knowledge About the Domain:**

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

* If two terminals are connected, then they have the same input signal, which can be represented as:

1. ∀  t1, t2 Terminal (t1) ∧ Terminal (t2) ∧ Connect (t1, t2) → Signal (t1) = Signal (2).

* Signal at every terminal will have either value 0 or 1; it will be represented as:

1. ∀  t Terminal (t) →Signal (t) = 1 ∨Signal (t) = 0.

* Connect predicates are commutative:

1. ∀  t1, t2 Connect(t1, t2)  →  Connect (t2, t1).

* Representation of types of gates:

1. ∀  g Gate(g) ∧ r = Type(g) → r = OR ∨r = AND ∨r = XOR ∨r = NOT.

* The output of an AND gate will be zero if and only if any of its inputs is zero.

1. ∀  g Gate(g) ∧ Type(g) = AND →Signal (Out(1, g))= 0 ⇔  ∃n Signal (In(n, g))= 0.

* The output of the OR gate is 1 if and only if any of its input is 1:

1. ∀  g Gate(g) ∧ Type(g) = OR → Signal (Out(1, g))= 1 ⇔  ∃n Signal (In(n, g))= 1

* The output of the XOR gate is 1 if and only if its inputs are different:

1. ∀  g Gate(g) ∧ Type(g) = XOR → Signal (Out(1, g)) = 1 ⇔  Signal (In(1, g)) ≠ Signal (In(2, g)).

* The output of a NOT gate is the invert of its input:

1. ∀  g Gate(g) ∧ Type(g) = NOT →   Signal (In(1, g)) ≠ Signal (Out(1, g)).

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

1. ∀  g Gate(g) ∧ Type(g) = NOT →   Arity(g, 1, 1)
2. ∀  g Gate(g) ∧ r =Type(g)  ∧ (r= AND ∨r= OR ∨r= XOR) →  Arity (g, 2, 1).

* All gates are logic circuits:

1. ∀  g Gate(g) → Circuit (g).

**5. Encode a Description of the Problem Instance:**

Now we encode the problem of circuit C1; firstly, we categorize the circuit and its gate components. This step is easy if an ontology about the problem is already thought out. This step involves writing simple atomic sentences of instances of concepts, which is known as ontology.

For the given circuit C1, we can 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:

1. For XOR gate: Type(x1)= XOR, Type(X2) = XOR
2. For AND gate: Type(A1) = AND, Type(A2)= AND
3. For OR gate: Type (O1) = OR.

And then represent the connections between all the gates.

#### **Note: Ontology defines a particular theory of the nature of existence.**

Encoding the instance of the problem enables the system to operate using real configurations of components, instantiating general rules for reasoning. This step also projects the abstract structure onto a real example-circuit C1. It not only refers to mapping types to gates but also defines their interconnectivities through predicates such as Connect. These interconnectivities determine paths of signal flows, which are critical for examining logic.

With a structured ontology, complex circuits may also be reduced into basic logical primitives. Through the linkage of gate families and links, this encoding can enable simulation, fault diagnosis, and optimization techniques to be implemented, and for this reason, it is an integral part of the knowledge-engineering process.

**6. Pose Queries to the Inference Procedure and Get Answers:**

In this step, we will find all the possible sets of values of all the terminals for the adder circuit. The first query will be:

What should be the combination of inputs that would generate the first output of circuit C1, as 0 and a second output to be 1?

1. ∃i1, i2, i3 Signal (In(1, C1))=i1 ∧ Signal (In(2, C1))=i2 ∧ Signal (In(3, C1))= i3
2. ∧ Signal (Out(1, C1)) =0 ∧ Signal (Out(2, C1))=1

This is a demonstration of the strength of logical inference in knowledge systems. Through questioning, we can make helpful inferences, such as identifying some conditions on inputs that lead to desired outputs. For a one-bit full adder, it is necessary to pose questions regarding combinations of A, B, and Carry-in that lead to a given Sum and Carry-out for testing and verification.

The inference engine checks via all logical rules and relations already programmed to give accurate responses. Backward reasoning is also possible, where the outputs are given and the inputs need to be deduced. Such a query is greatly helpful in debugging circuits, optimization, and constructing systems with predetermined logical outputs.

**7. Debug the Knowledge Base:**

Now we will debug the knowledge base, and this is the last step of the complete process. In this step, we will try to debug the issues with the knowledge base.

In the knowledge base, we may have omitted assertions like 1 ≠ 0.