Knowledge based agents in AI

Last Updated : 21 Sep, 2023

Humans claim that how intelligence is achieved- not by purely reflect mechanisms but by process of reasoning that operate on internal representation of knowledge. In AI these techniques for intelligence are present in Knowledge Based Agents.

Knowledge-Based System

- A knowledge-based system is a system that uses artificial intelligence techniques to store and reason with knowledge. The knowledge is typically represented in the form of rules or facts, which can be used to draw conclusions or make decisions.
- One of the key benefits of a knowledge-based system is that it can help to automate decision-making processes. For example, a knowledge-based system could be used to diagnose a medical condition, by reasoning over a set of rules that describe the symptoms and possible causes of the condition.
- Another benefit of knowledge-based systems is that they can be used to explain their decisions to humans. This can be useful, for example, in a customer service setting, where a knowledge-based system can help a human agent understand why a particular decision was made.

Knowledge-Based System in Artificial Intelligence

- An intelligent agent needs knowledge about the real world to make 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 action. These agents can represent the world with some formal representation and act intelligently.

Why use a knowledge base?

- A knowledge base **inference** is required for updating knowledge for an agent to learn with experiences and take action as per the knowledge.
- Inference means deriving new sentences from old. The inference-**based** system allows us to add a new sentence to the knowledge base. A sentence is a proposition about the world. The inference system applies logical rules to the KB to deduce new information.
- **The inference** system generates new facts so that an agent can update the KB. An inference system works mainly in two rules which are given:
    - Forward chaining
    - Backward chaining

Various levels of knowledge-based agents

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.

Knowledge-based agents have explicit representation of knowledge that can be reasoned. They maintain internal state of knowledge, reason over it, update it and perform actions accordingly. These agents act intelligently according to requirements.

Knowledge based agents give the current situation in the form of sentences. They have complete knowledge of current situation of mini-world and its surroundings. These agents manipulate knowledge to infer new things at "Knowledge level".

knowledge-based system has following features

Knowledge base (KB): *It is the key component of a knowledge-based agent. These deal with real facts of world. It is a mixture of sentences which are explained in knowledge representation language.*

Actions performed by an agent

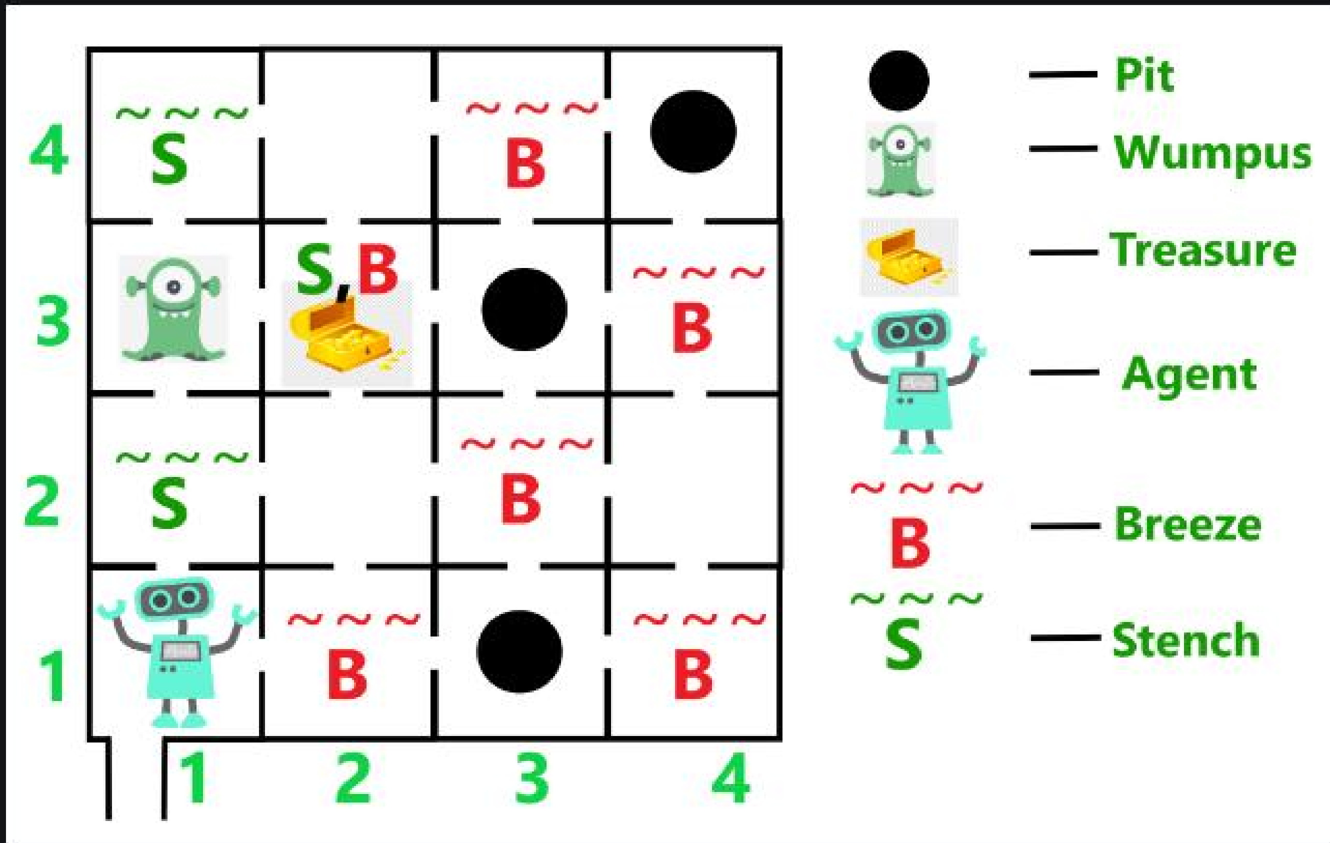
Inference System is used when we want to update some information (sentences) in Knowledge-Based System and to know the already present information. This mechanism is done by **TELL** and **ASK** operations. They include inference i.e. producing new sentences from old. Inference must accept needs when one asks a question to KB and answer should follow from what has been Told to KB. Agent also has a KB, which initially has some background Knowledge. Whenever, agent program is called, it performs some actions.

Actions done by KB Agent:

1. It TELLS what it recognized from the environment and what it needs to know to the knowledge base.
2. It ASKS what actions to do? and gets answers from the knowledge base.
3. It TELLS the which action is selected , then agent will execute that action.

Understanding Wumpus World Problem

Wumpus World is a 4x4 grid consisting of **16 rooms**. Agent starts at Room[1,1] facing right and its goal is to retrieve treasure while avoiding hazards such as pits and the Wumpus. Agent must navigate through grid using its limited sensory input to make decisions that will keep it safe and allow it to successfully collect the treasure and exit the cave.



Key Elements:

- **Pits**: If the agent steps into a pit it falls and dies. A **breeze** in adjacent rooms suggests nearby pits.
- **Wumpus**: A creature that kills agent if it enters its room. Rooms next to the Wumpus have a **stench**. Agent can use an **arrow** to kill the Wumpus.
- **Treasure**: Agent's main objective is to **collect the treasure** (gold) which is located in one room.
- **Breeze**: Indicates a pit is nearby.
- **Stench**: Indicates the Wumpus is nearby.

PEAS Description

PEAS stands for **Performance Measures**, **Environment**, **Actuators** and **Sensors** which describe agent's capabilities and environment.

1. Performance measures: Rewards or Punishments

- Agent gets gold and return back safe = *+1000 points*
- Agent dies (pit or Wumpus)= *-1000 points*
- Each move of the agent = *-1 point*
- Agent uses the arrow = *-10 points*

2. Environment: A setting where everything will take place.

- A cave with *16(4x4)* rooms.
- Rooms adjacent (not diagonally) to the Wumpus are stinking.
- Rooms adjacent (not diagonally) to the pit are breezy.
- Room with 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 agent to perform following actions in the environment.

- Move forward: Move to next room.
- Turn right/left: Rotate agent 90 degrees.
- Shoot: Kill Wumpus with arrow.
- Grab: Take treasure.
- Release: Drop treasure

4. Sensors: Devices help the agent in sensing following from the environment.

- Breeze: Detected near a pit.
- Stench: Detected near the Wumpus.
- Glitter: Detected when treasure is in the room.
- Scream: Triggered when Wumpus is killed.
- Bump: Occurs when hitting a wall.

How the Agent Operates with PEAS

1. **Perception**: Agent uses sensory inputs (breeze, stench, glitter) to detect its surroundings and understand the environment.
2. **Inference**: Agent applies logical reasoning to find location of hazards. For example if it detects a **breeze**, it warns that a pit is nearby or if there's a **stench** it suspects the Wumpus is in an adjacent room.
3. **Planning**: Based on its deductions agent plans its next move avoiding risky areas like rooms with suspected pits or the Wumpus.
4. **Action**: Agent performs planned action such as moving to a new room, shooting arrow at the Wumpus or taking the treasure.

Understanding Propositional Logic in Artificial Intelligence

Propositional logic works with statements called propositions that can be true or false. These propositions represent facts or conditions about a situation. We use symbols to represent the propositions and logical operations to connect those propositions. It help us understand how different facts are related to each other in complex statements or problem. Proposition operators like conjunction (∧), disjunction (∨), negation (¬), implication(→) and biconditional (↔) helps combine various proposition to represent logical relations.

Example of Propositions Logic

- **P**: "The sky is blue." (This statement can be either true or false.)
- **Q**: "It is raining right now." (This can also be true or false.)
- **R**: "The ground is wet." (This is either true or false.)

These can be combined using logical operations to create more complex statements. For example:

- **P ∧ Q**: "The sky is blue AND it is raining." (This is true only if both P and Q are true.)
- **P ∨ Q**: "The sky is blue OR it is raining." (This is true if at least one of P or Q is true.)
- **¬P**: "It is NOT true that the sky is blue." (This is true if P is false means the sky is not blue.)

Logical Equivalence

Two statements are logically equivalent if they always have the same truth values in every possible situation. For example:

- The statement "S → T" (if S then T) is equivalent to "¬S ∨ T" (not S or T). This means "if S is true, then T must be true" is the same as "either S is false or T is true."
- The biconditional "P ↔ Q" (P if and only if Q) is equivalent to "(P → Q) ∧ (Q → P)" (P implies Q and Q implies P).

Basic Concepts of Propositional Logic

1. Propositions

A proposition is a statement that can either be true or false. It does not matter how complicated statement is if it can be classified as true or false then it is a proposition. For example:

- "The sky is blue." (True)
- "It is raining." (False)

2. Logical Connectives

Logical connectives are used to combine simple propositions into more complex ones. The main connectives are:

- **AND (∧)**: This operation is true if both propositions are true.
  Example: "It is sunny ∧ it is warm" is true only if both "It is sunny" and "It is warm" are true.
- **OR (∨)**: This operation is true if at least one of the propositions is true.
  Example: "It is sunny ∨ it is raining" is true if either "It is sunny" or "It is raining" is true.
- **NOT (¬)**: This operation reverses the truth value of a proposition.
  Example: "¬It is raining" is true if "It is raining" is false.
- **IMPLIES (→)**: This operation is true if the first proposition leads to the second.
  Example: "If it rains then the ground is wet" (It rains → The ground is wet) is true unless it rains and the ground is not wet.
- **IF AND ONLY IF (↔)**: This operation is true if both propositions are either true or false together.
  Example: "It is raining ↔ The ground is wet" is true if both "It is raining" and "The ground is wet" are either true or both false.

3. Truth Tables

They are used to find the truth value of complex propositions by checking all possible combinations of truth values for their components. They systematically list every possible combinations which helps in making it easy to find how different logical operators affect the overall outcome. This approach ensures that no combination is given extra importance which provides a clear and complete picture of the logic at work.

4. Tautologies, Contradictions and Contingencies

- **Tautology**: A proposition that is always true no matter the truth values of the individual components.
  Example: "P ∨ ¬P" (This is always true because either P is true or P is false).
- **Contradiction**: A proposition that is always false.
  Example: "P ∧ ¬P" (This is always false because P can't be both true and false at the same time).
- **Contingency**: A proposition that can be true or false depending on the truth values of its components.
  Example: "P ∧ Q" (This is true only if both P and Q are true).

Properties of Operators

Logical operators in propositional logic have various important properties that help to simplify and analyze complex statements:

1. **Commutativity**: Order of propositions doesn't matter when using AND (∧) or OR (∨).

- $P \land Q \equiv Q \land P$
- $P \lor Q \equiv Q \lor P$

2. **Associativity**: Grouping of propositions doesn't matter when using multiple ANDs or ORs.

- $(P \land Q) \land R \equiv P \land (Q \land R)$
- $(P \lor Q) \lor R \equiv P \lor (Q \lor R)$

3. **Distributivity**: AND (∧) and OR (∨) can distribute over each other which is similar to multiplication and addition in math.

- $P \land (Q \lor R) \equiv (P \land Q) \lor (P \land R)$
- $P \lor (Q \land R) \equiv (P \lor Q) \land (P \lor R)$

4. **Identity**: A proposition combined with "True" or "False" behaves predictably.

- P ∧ true ≡ P
- P ∨ false ≡ P

5. **Domination**: When combined with "True" or "False" some outcomes are always fixed.

- P ∨ true ≡ true
- P ∧ false ≡ false

6. **Double Negation:** Negating a proposition twice cancels out the negation.

¬ (¬P) ≡ P

7. **Idempotence**: Repeating same proposition with AND or OR doesn’t change its value.

- P ∧ P ≡ P
- P ∨ P ≡ P

Applications of Propositional Logic in AI

1. Knowledge Representation: Propositional logic is used to represent knowledge in a structured way. It allows AI systems to store and manipulate facts about the world. For example in expert systems knowledge is encoded as a set of propositions and logical rules.

2. Automated Reasoning: AI uses logical rules such as Modus Ponens and Modus Tollens which help systems to find new conclusions from existing fact and to "think" logically. For example:

- **Modus Ponens:** If "P → Q" and "P" are true then "Q" must be true.
- **Modus Tollens:** If "P → Q" and "¬Q" are true then "¬P" must be true.

3. Problem Solving and Planning: It allows AI planners to solve problems and to create action sequences by representing goals. For example the STRIPS planning system helps propositional logic to represent preconditions and effects of actions.

4. Decision Making: It helps to evaluate various options and find the best course of action. Logical rules can encode decision criteria and truth tables can be used to assess the outcomes of different choices.

5. Natural Language Processing (NLP): It is applied in NLP for tasks like semantic parsing where natural language sentences are converted into logical representations. This helps in

Propositional Logic

Last Updated : 27 Jan, 2025

Logic is the basis of all mathematical reasoning and all automated reasoning. The rules of logic specify the meaning of mathematical statements. These rules help us understand and reason with statements such as -

> $\exists\, x \text{ such that } x \neq a^2 + b^2, \text{ where } x, a, b \in Z$
> *Which in Simple English means "**There exists an integer that is not the sum of two squares**".*

Importance of Mathematical Logic

The rules of logic give precise meaning to mathematical statements. These rules are used to distinguish between valid and invalid mathematical arguments. Apart from its importance in understanding mathematical reasoning, logic has numerous applications in Computer Science, varying from the design of digital circuits to the construction of computer programs and verification of the correctness of programs.

Types of Propositions

In propositional logic, propositions are statements that can be evaluated as true or false. The are the building blocks of more complex logical statements. Here's a breakdown of the two main types of propositions:

- Atomic Propositions
- Compound Propositions

Propositional Logic

Propositional logic is a branch of mathematics that studies the logical relationships between propositions (or statements, sentences, assertions) taken as a whole, and connected via logical connectives.

For Example,

1. *The sun rises in the East and sets in the West.*
2. *1 + 1 = 2*
3. *'b' is a vowel.*

Above 3 sentences are propositions, where the first two are Valid(True) and the third one is Invalid(False).

All of the above sentences are propositions, where the first two are Valid(True) and the third one is Invalid(False). Some sentences that do not have a truth value or may have more than one truth value are not propositions. For Example,

1. What time is it?
2. Go out and Play
3. x + 1 = 2

The above sentences are not propositions as the first two do not have a truth value, and the third one may be true or false. To represent propositions, **propositional variables** are used. By Convention, these variables are represented by small alphabets such as $p,\ q,\ r,\ s$. The area of logic which deals with propositions is called **propositional calculus** or **propositional logic**. It also includes producing new propositions using existing ones. Propositions constructed using one or more propositions are called **compound propositions**. The propositions are combined together using **Logical Connectives** or **Logical Operators**.

Truth Table of Propositional Logic

Since we need to know the truth value of a proposition in all possible scenarios, we consider all the possible combinations of the propositions which are joined together by Logical Connectives to form the given compound proposition. This compilation of all possible scenarios in a tabular format is called a **truth table**. Most Common Logical Connectives-

1. Negation

If p is a proposition, then the negation of p is denoted by $\neg p$, which when translated to simple English means- "It is not the case that **p**" or simply "not **p**". The truth value of **-p** is the opposite of the truth value of **p**. The truth table of **-p** is:

p	¬p
T	F
F	T

2. Conjunction

For any two propositions p and q, their conjunction is denoted by $p \wedge q$, which means "p and q". The conjunction $p \wedge q$ is True when both p and q are True, otherwise False. The truth table of $p \wedge q$ is:

p	q	p ∧ q
T	T	T
T	F	F
F	T	F
F	F	F

3. Disjunction

For any two propositions p and q, their disjunction is denoted by $p \vee q$, which means "p or q". The disjunction $p \vee q$ is True when either p or q is True, otherwise False. The truth table of $p \vee q$ is:

p	q	p ∨ q
T	T	T
T	F	T
F	T	T
F	F	F

4. Exclusive Or

For any two propositions p and q, their exclusive or is denoted by $p \oplus q$, which means "either p or q but not both". The exclusive or $p \oplus q$ is True when either p or q is True, and False when both are true or both are false. The truth table of $p \oplus q$ is:

p	q	p ⊕ q
T	T	F
T	F	T
F	T	T
F	F	F

5. Implication

For any two propositions p and q , the statement "if p then q " is called an implication and it is denoted by $p \to q$. In the implication $p \to q$, p is called the **hypothesis** or **antecedent** or **premise** and q is called the **conclusion** or **consequence**. The implication is $p \to q$ is also called a **conditional statement**. The implication is false when p is true and q is false otherwise it is true. The truth table of $p \to q$ is:

p	q	p → q
T	T	T
T	F	F
F	T	T
F	F	T

One might wonder that why is $p \to q$ true when p is false. This is because the implication guarantees that when p and q are true then the implication is true. But the implication does not guarantee anything when the premise p is false. There is no way of knowing whether or not the implication is false since p did not happen. This situation is similar to the "Innocent until proven Guilty" stance, which means that the implication $p \to q$ is considered true until proven false. Since we cannot call the implication $p \to q$ false when p is false, our only alternative is to call it true.

This follows from the **Explosion Principle** which says: "A False statement implies anything" Conditional statements play a very important role in mathematical reasoning, thus a variety of terminology is used to express $p \to q$, some of which are listed below.

> *"If p, then "q"p is sufficient for q""q when p""a necessary condition for p is q""p only if q""q unless ≠p""q follows from p"*

Example, "If it is Friday then it is raining today" is a proposition which is of the form $p \to q$. The above proposition is true if it is not Friday(premise is false) or if it is Friday and it is raining, and it is false when it is Friday but it is not raining.