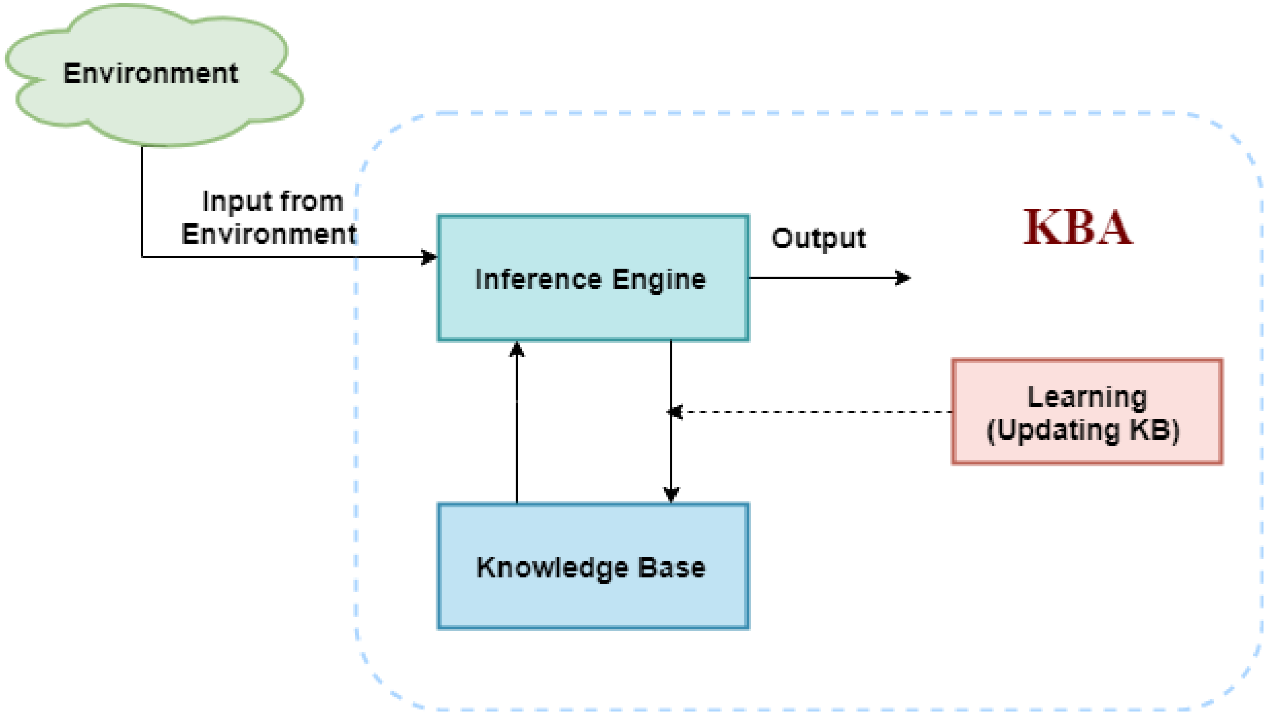
Knowledge-Based Agent in Artificial intelligence

Dr Sujit Das
NIT Warangal

Knowledge-Based Agent

- 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:



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.

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:**
  - **TELL:** This operation tells the knowledge base what it perceives from the environment.
  - **ASK:** This operation asks the knowledge base what action it should perform.
  - **Perform:** It performs the selected action.

Various levels of knowledge-based agent

- 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. 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. 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.

Note: 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

- What is knowledge representation?
- Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world. **But how machines do all these things comes under knowledge representation and reasoning**
- **Knowledge:**
- Knowledge is awareness or familiarity gained by experiences of facts, data, and situations.
- types of knowledge in artificial intelligence:
  - **Declarative Knowledge**
  - **Procedural Knowledge**
  - **Meta-knowledge**
  - **Heuristic knowledge**
  - **Structural knowledge**

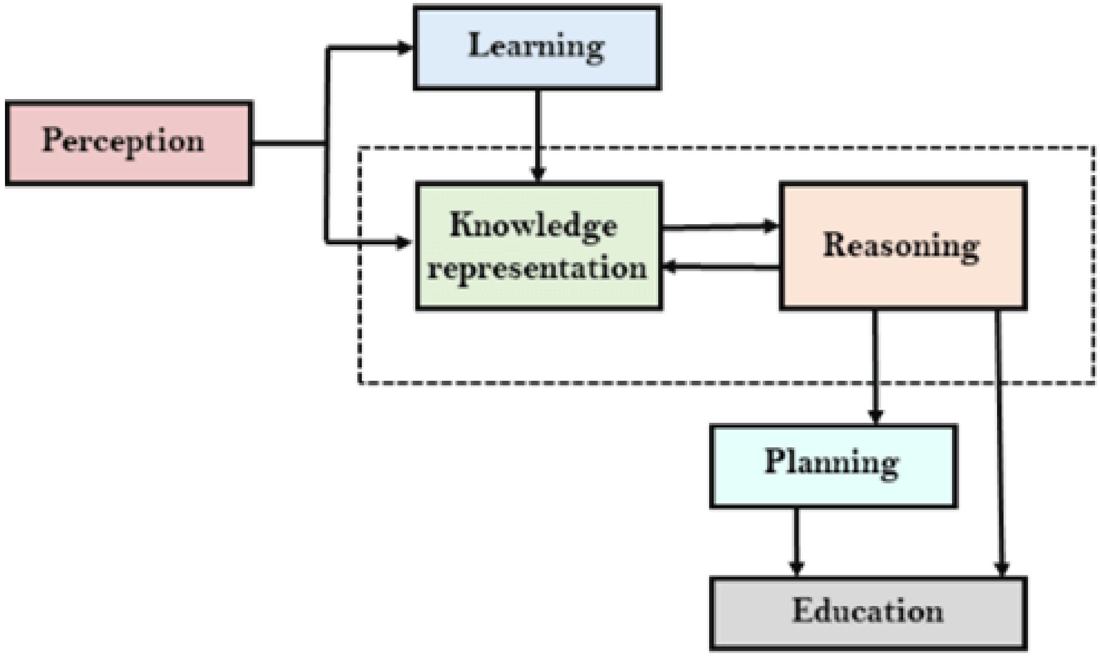
types of knowledge in artificial intelligence

- **Declarative Knowledge**
  - Declarative knowledge is to know about something.
- **Procedural Knowledge**
  - Procedural knowledge is a type of knowledge which is responsible for knowing how to do something.
- **Meta-knowledge**
  - Knowledge about the other types of knowledge is called Meta-knowledge.
- **Heuristic knowledge**
  - Heuristic knowledge is representing knowledge of some experts in a filed or subject.
- **Structural knowledge**
  - Structural knowledge is basic knowledge to problem-solving.

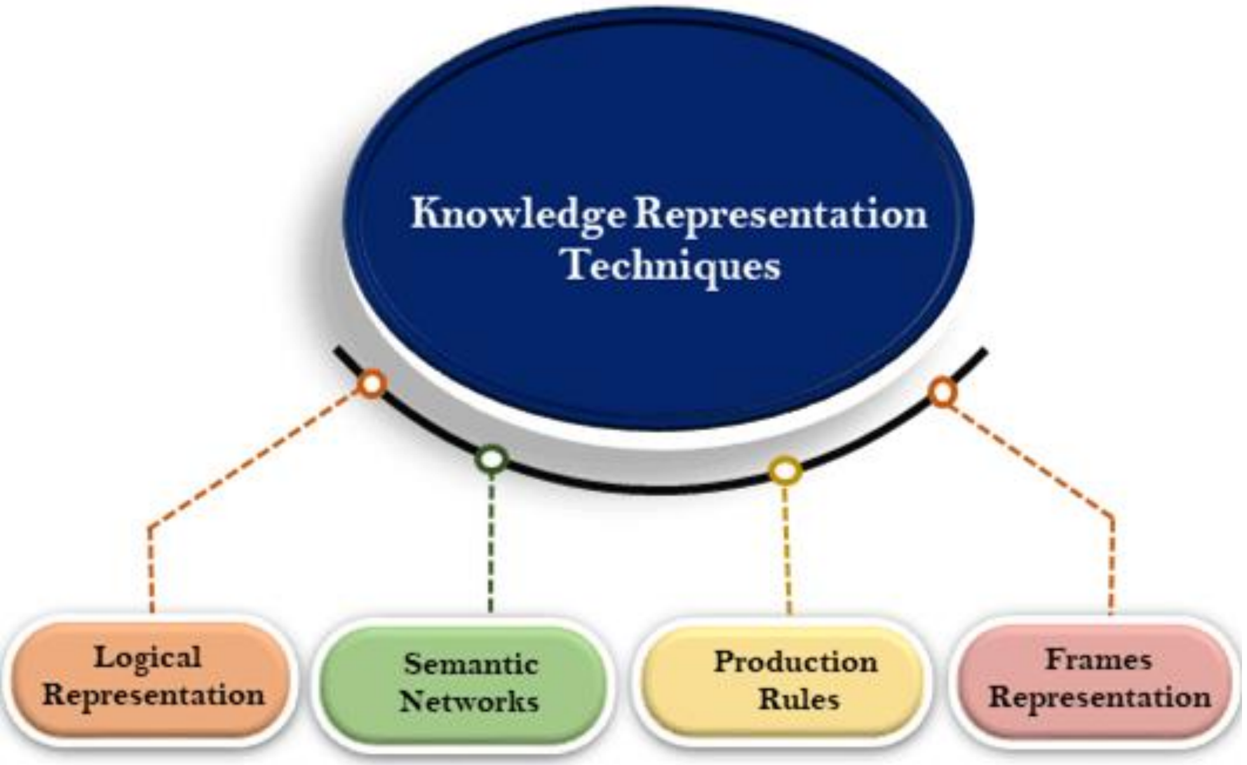
The relation between knowledge and intelligence:

- Knowledge of real-worlds plays a vital role in intelligence and same for creating artificial intelligence.
- Knowledge plays an important role in demonstrating intelligent behavior in AI agents.
- An agent is only able to accurately act on some input when he has some knowledge or experience about that input.
- Let's suppose if you met some person who is speaking in a language which you don't know, then how you will able to act on that.
- The same thing applies to the intelligent behavior of the agents

AI knowledge cycle



Techniques of knowledge representation



Logical Representation

- Logical representation is a language with some concrete rules which deals with propositions and has no ambiguity in representation.
- Logical representation means drawing a conclusion based on various conditions.
- This representation lays down some important communication rules.
- It consists of precisely defined syntax and semantics which supports the sound inference.
- Each sentence can be translated into logics using syntax and semantics.

Semantic Network Representation

- Semantic networks are alternative of predicate logic for knowledge representation.
- In Semantic networks, we can represent our knowledge in the form of graphical networks.
- This network consists of nodes representing objects and arcs which describe the relationship between those objects.
- Semantic networks can categorize the object in different forms and can also link those objects.
- Semantic networks are easy to understand and can be easily extended.

Frame Representation

- A frame is a record like structure which consists of a collection of attributes and its values to describe an entity in the world.
- Frames are the AI data structure which divides knowledge into substructures by representing stereotypes situations.
- It consists of a collection of slots and slot values. These slots may be of any type and sizes. Slots have names and values which are called facets.
- **Facets:** The various aspects of a slot is known as **Facets**.
- Facets are features of frames which enable us to put constraints on the frames.
- A frame may consist of any number of slots, and a slot may include any number of facets and facets may have any number of values.
- A frame is also known as **slot-filter knowledge representation** in artificial intelligence.

Production Rules

- Production rules system consist of (**condition, action**) pairs which mean, "If condition then action". It has mainly three parts:
  - The set of production rules
  - Working Memory
  - The recognize-act-cycle
- In production rules agent checks for the condition and if the condition exists then production rule fires and corresponding action is carried out.
- The condition part of the rule determines which rule may be applied to a problem.
- And the action part carries out the associated problem-solving steps. This complete process is called a recognize-act cycle.
- The working memory contains the description of the current state of problems-solving and rule can write knowledge to the working memory. This knowledge match and may fire other rules.
- If there is a new situation (state) generates, then multiple production rules will be fired together, this is called conflict set.
- In this situation, the agent needs to select a rule from these sets, and it is called a conflict resolution.

Propositional logic in Artificial intelligence

Propositional logic

- Propositional logic (PL) is the simplest form of logic where all the statements are made by propositions.
- A proposition is a declarative statement which is either true or false.
- It is a technique of knowledge representation in logical and mathematical form.
- Example:
- a) It is Sunday.
- b) The Sun rises from West (False proposition)
- c) 3+3= 7(False proposition)
- d) 5 is a prime number.

some basic facts about propositional logic

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

Syntax of propositional logic:

- The syntax of propositional logic defines the allowable sentences for the knowledge representation. There are two types of Propositions:
    - **Atomic Propositions**
    - **Compound propositions**
- **Atomic Proposition:** Atomic propositions are the simple propositions. It consists of a single proposition symbol. These are the sentences which must be either true or false.
- **Example:**
- a) 2+2 is 4, it is an atomic proposition as it is a **true** fact.
- b) "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:**
- a) "It is raining today, and street is wet."
- b) "Ankit is a doctor, and his clinic is in Mumbai."

Logical Connectives:

- Logical connectives are used to connect two simpler propositions or representing a sentence logically. We can create compound propositions with the help of logical connectives. There are mainly five connectives, which are given as follows:
- **Negation:** A sentence such as ¬ P is called negation of P. A literal can be either Positive literal or negative literal.
- **Conjunction:** A sentence which has **∧** connective such as, **P ∧ Q** is called a conjunction.
  Example: Rohan is intelligent and hardworking. It can be written as,
  P= Rohan is intelligent,
  Q= Rohan is hardworking. → P∧ Q.
- **Disjunction:** A sentence which has ∨ connective, such as **P ∨ Q**. is called disjunction, where P and Q are the propositions.
  Example: "Ritika is a doctor or Engineer",
  Here P= Ritika is Doctor. Q= Ritika is Engineer, 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.

Summarized table for Propositional Logic Connectives

Connective symbols	Word	Technical term	Example
$\wedge$	AND	Conjunction	$A \wedge B$
$\vee$	OR	Disjunction	$A \vee B$
$\rightarrow$	Implies	Implication	$A \rightarrow B$
$\Leftrightarrow$	If and only if	Biconditional	$A \Leftrightarrow B$
$\neg$ or $\sim$	Not	Negation	$\neg A$ or $\neg B$

Truth Table

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

For Negation:

P	¬ P
True	**False**
False	**True**

For Conjunction:

P	Q	P∧ Q
True	True	**True**
True	False	**False**
False	True	**False**
False	False	**False**

For disjunction:

P	Q	P V Q.
True	True	**True**
False	True	**True**
True	False	**True**
False	False	**False**

For Implication:

P	Q	P→ Q
True	True	**True**
True	False	**False**
False	True	**True**
False	False	**True**

For Biconditional:

P	Q	P⇔ Q
True	True	**True**
True	False	**False**
False	True	**False**
False	False	**True**

Truth table with three propositions P, Q, and R

P	Q	R	¬R	P∨Q	P∨Q→¬R
True	True	True	False	True	False
True	True	False	True	True	True
True	False	True	False	True	False
True	False	False	True	True	True
False	True	True	False	True	False
False	True	False	True	True	True
False	False	True	False	False	True
False	False	False	True	False	True

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

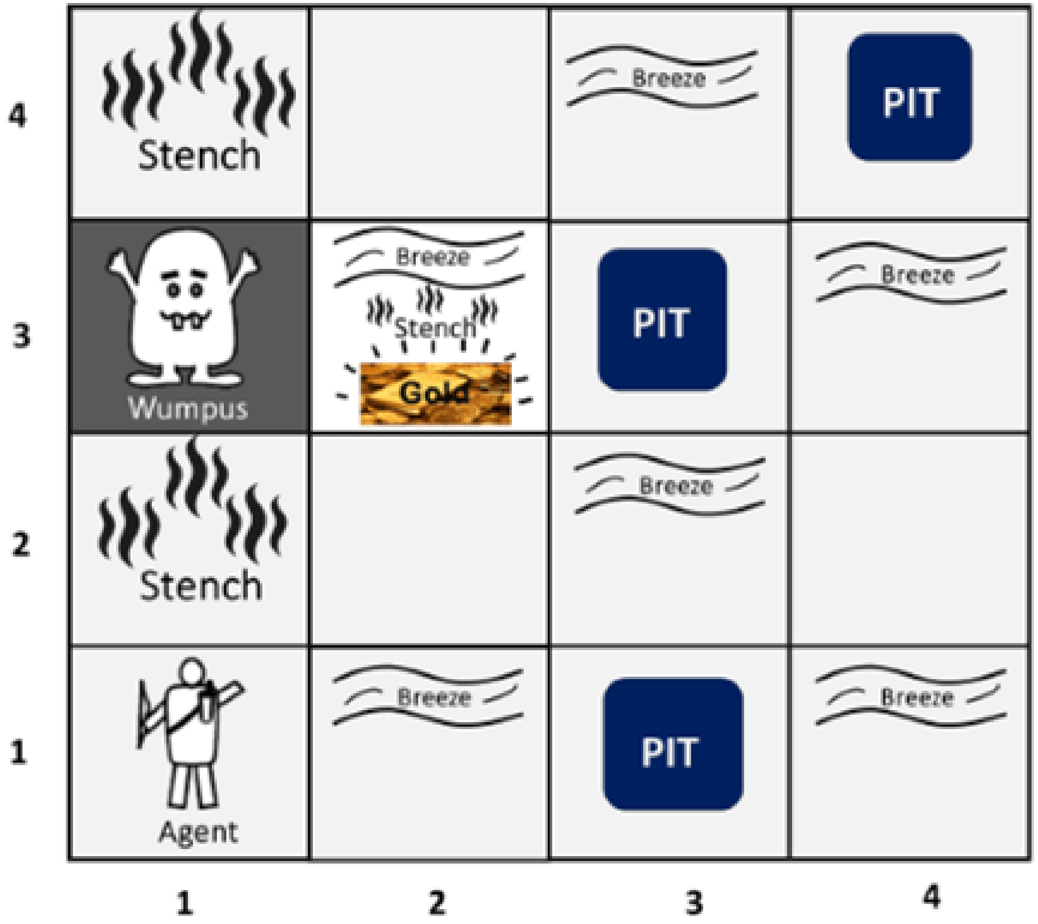
Properties of Operators:

- **Commutativity:**
  - $P \wedge Q = Q \wedge P$, or
  - $P \vee Q = Q \vee P$.
- **Associativity:**
  - $(P \wedge Q) \wedge R = P \wedge (Q \wedge R)$,
  - $(P \vee Q) \vee R = P \vee (Q \vee R)$
- **Identity element:**
  - $P \wedge True = P$,
  - $P \vee True = True$.
- **Distributive:**
  - $P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$.
  - $P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$.
- **DE Morgan's Law:**
  - $\neg (P \wedge Q) = (\neg P) \vee (\neg Q)$
  - $\neg (P \vee Q) = (\neg P) \wedge (\neg Q)$.
- **Double-negation elimination:**
  - $\neg (\neg P) = P$.

Limitations of Propositional logic:

- We cannot represent relations like ALL, some, or none with propositional logic. Example:
  - **All the kids are intelligent.**
  - **Some apples are sweet.**
- Propositional logic has limited expressive power.
- In propositional logic, we cannot describe statements in terms of their properties or logical relationships.

The Wumpus World in Artificial intelligence



The Wumpus World in Artificial intelligence

- 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.

The Wumpus World in Artificial intelligence

- **There are also some components which can help the agent to navigate the cave. These components are given as follows:**
  - The rooms adjacent to the Wumpus room are smelly, so that it would have some stench.
  - The room adjacent to PITs has a breeze, so if the agent reaches near to PIT, then he will perceive the breeze.
  - There will be glitter in the room if and only if the room has gold.
  - 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

- 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.

PEAS description of Wumpus world

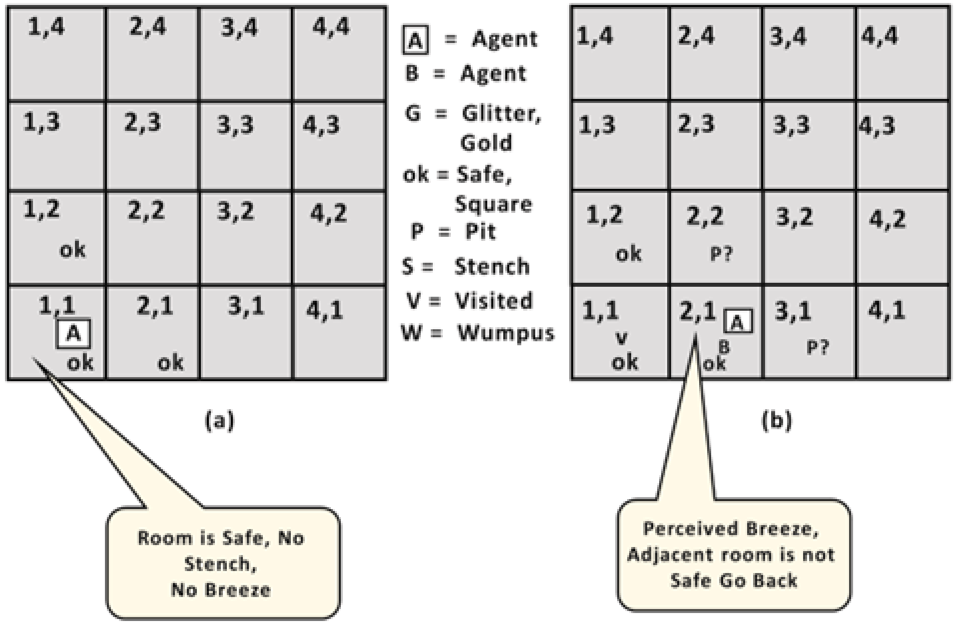
- Actuators:
  - Left turn,
  - Right turn
  - Move forward
  - Grab
  - Release
  - Shoot.
- Sensors:
  - The agent will perceive the **stench** if he is in the room adjacent to the Wumpus. (Not diagonally).
  - The agent will perceive **breeze** if he is in the room directly adjacent to the Pit.
  - 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

- **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.
-

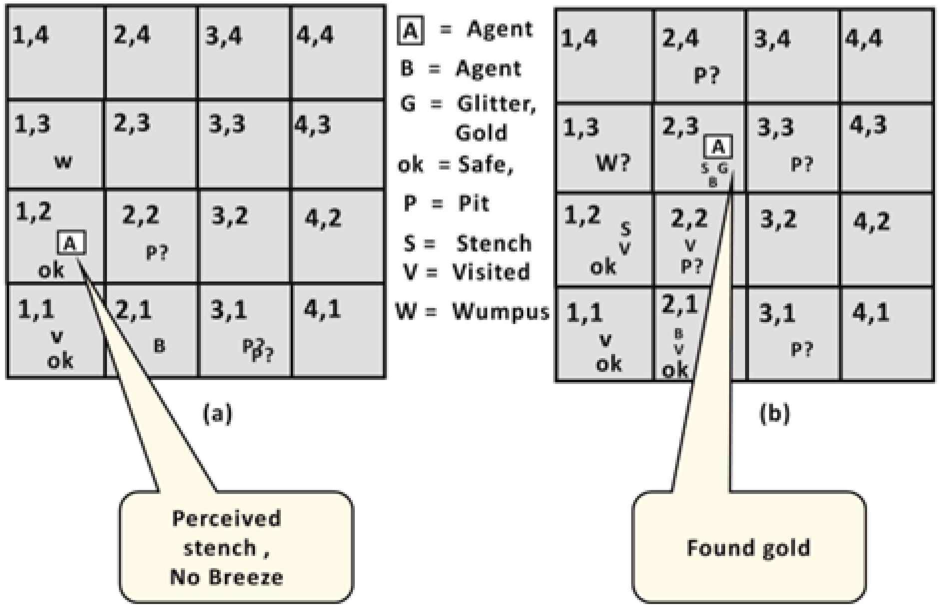


Agent's second Step

- 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].



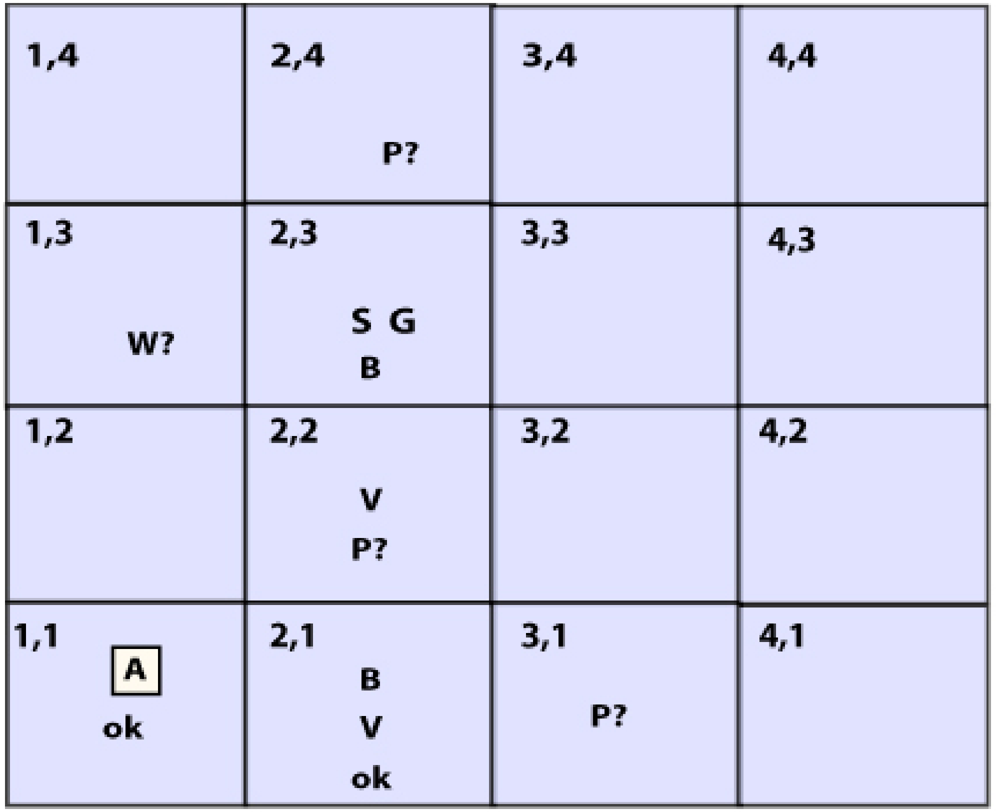
Agent's fourth step:

- 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 Wumpus world

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

Knowledge-base for Wumpus world



Atomic proposition variable for Wumpus world:

- Let $\mathbf{P_{i,j}}$ be true if there is a Pit in the room [i, j].
- Let $\mathbf{B_{i,j}}$ be true if agent perceives breeze in [i, j], (dead or alive).
- Let $\mathbf{W_{i,j}}$ be true if there is wumpus in the square[i, j].
- Let $\mathbf{S_{i,j}}$ be true if agent perceives stench in the square [i, j].
- Let $\mathbf{V_{i,j}}$ be true if that square[i, j] is visited.
- Let $\mathbf{G_{i,j}}$ be true if there is gold (and glitter) in the square [i, j].
- Let $\mathbf{OK_{i,j}}$ be true if the room is safe.

Some Propositional Rules for the wumpus world

(R1) $\neg S_{11} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$

(R2) $\neg S_{21} \rightarrow \neg W_{11} \wedge \neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31}$

(R3) $\neg S_{12} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{22} \wedge \neg W_{13}$

(R4) $S_{12} \rightarrow W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$

Representation of Knowledgebase for Wumpus world:

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

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

- Here in the first row, we have mentioned propositional variables for room[1,1], which is showing that room does not have wumpus($\neg W_{11}$), no stench (**$\neg S_{11}$**), no Pit(**$\neg P_{11}$**), no breeze(**$\neg B_{11}$**), no gold (**$\neg G_{11}$**), visited (**V_{11}**), and the room is Safe(**OK_{11}**).
- In the second row, we have mentioned propositional variables for room [1,2], which is showing 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 propositional variable for room[2,1], which is showing that there is no wumpus(¬ W21), no stench (**$\neg S_{21}$**), no Pit (**$\neg P_{21}$**), Perceives breeze(**B_{21}**), no glitter(**$\neg G_{21}$**), visited (**V_{21}**), and room is safe (**OK_{21}**).

Rules of Inference in Artificial intelligence

- 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 $P \rightarrow Q$. 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 $Q \rightarrow P$.
- **Contrapositive:** The negation of converse is termed as contrapositive, and it can be represented as $\neg Q \rightarrow \neg P$.
- **Inverse:** The negation of implication is called inverse. It can be represented as $\neg P \rightarrow \neg Q$.

Types of Inference rules: 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:

Notation for Modus ponens: $$\frac{P \rightarrow Q, \quad P}{\therefore Q}$$

- **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.

P	Q	$P \rightarrow Q$
0	0	0
0	1	1
1	0	0
1	1	1

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:

Notation for Modus Tollens: $$\frac{P \to Q, \quad \sim Q}{\sim P}$$

- **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

P	Q	$\sim P$	$\sim Q$	$P \to Q$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	0
1	1	0	0	1

Hypothetical Syllogism

- The Hypothetical Syllogism rule state that if P→R is true whenever P→Q is true, and Q→R is true.
- **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**

P	Q	R	$P \rightarrow Q$	$Q \rightarrow R$	$P \rightarrow R$
0	0	0	1	1	1
0	0	1	1	1	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	1	0	0
1	1	1	1	1	1

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:

Notation of Disjunctive syllogism: $\frac{P \vee Q, \quad \neg P}{Q}$

Example:

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

P	Q	$\neg P$	$P \vee Q$
0	0	1	0
0	1	1	1
1	0	0	1
1	1	0	1

Addition:

- The Addition rule is one the common inference rule, and it states that If P is true, then PVQ will be true.
- **Example:**
- **Statement:** I have a vanilla ice-cream. ==> P
  Statement-2: I have Chocolate ice-cream.
  Conclusion: I have vanilla or chocolate ice-cream. ==> (PVQ)

P	Q	$P \vee Q$
0	0	0
1	0	1
0	1	1
1	1	1

Simplification

- The simplification rule state that if **P∧ Q** is true, then **Q or P** will also be true

P	Q	$P \wedge Q$
0	**0**	**0**
1	**0**	**0**
0	**1**	**0**
1	**1**	**1**

Resolution

- The Resolution rule state that if P∨Q and ¬ P∧R is true, then Q∨R will also be true

P	¬ P	Q	R	$P \vee Q$	¬ P∧R	$Q \vee R$
0	1	0	0	0	0	0
0	1	0	1	0	0	1
0	1	1	0	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	0	0
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	0	1

First-Order Logic in Artificial intelligence

- In the topic of Propositional logic, we have seen that how to represent statements using propositional logic.
- But unfortunately, in propositional logic, we can only represent the facts, which are either true or false.
- PL is not sufficient to represent the complex sentences or natural language statements.
- The propositional logic has very limited expressive power.
- Consider the following sentence, which we cannot represent using PL logic.
  - **"Some humans are intelligent",**
- To represent the above statements, PL logic is not sufficient, so we required some more powerful logic, such as first-order logic.

First-Order logic:

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

Basic Elements of First-order logic:

Constant	1, 2, A, John, Mumbai, cat,....
Variables	x, y, z, a, b,....
Predicates	Brother, Father, >,....
Function	sqrt, LeftLegOf, ...
Connectives	$\land$, $\lor$, $\neg$, $\Rightarrow$, $\Leftrightarrow$
Equality	==
Quantifier	$\forall$, $\exists$

Atomic sentences:

- Atomic sentences are the most basic sentences of first-order logic.
- These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms.
- We can represent atomic sentences as **Predicate (term1, term2, ..., term n)**.
- **Example: Ravi and Ajay are brothers: => Brothers(Ravi, Ajay).**
  Chinky is a cat: => cat (Chinky).

Complex Sentences:

- Complex sentences are made by combining atomic sentences using connectives.
- First-order logic statements can be divided into two parts:

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

Quantifiers in First-order logic:

- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression.
- There are two types of quantifier:
  - **Universal Quantifier, (for all, everyone, everything)**
  - **Existential quantifier, (for some, at least one).**

Universal Quantifier:

- Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.
- The Universal quantifier is represented by a symbol ∀, which resembles an inverted A.
- Note: In universal quantifier we use implication "→".
- If x is a variable, then ∀x is read as:
- **For all x**
- **For each x**
- **For every x.**
- Example:
- **All man drink coffee.**
- Let a variable x which refers to a cat so all x can be represented in UOD as below:
- **∀x man(x) → drink (x, coffee).**
- It will be read as: There are all x where x is a man who drink coffee

Existential Quantifier:

- Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.
- It is denoted by the logical operator ∃, which resembles as inverted E. When it is used with a predicate variable then it is called as an existential quantifier.
- Note: In Existential quantifier we always use AND or Conjunction symbol (∧).
- If x is a variable, then existential quantifier will be ∃x or ∃(x). And it will be read as:
- **There exists a 'x.'**
- **For some 'x.'**
- **For at least one 'x.'**
- Example:
- **Some boys are intelligent.**
- **$\exists x$: boys(x) ∧ intelligent(x)**
- It will be read as: There are some x where x is a boy who is intelligent.

Some Examples of FOL using quantifier

- **1. All birds fly.**
  In this question the predicate is "**fly(bird)**."
  And since there are all birds who fly so it will be represented as follows.
  ∀x bird(x) →fly(x).
- **2. Every man respects his parent.**
  In this question, the predicate is "**respect(x, y)," where x=man, and y= parent**.
  Since there is every man so will use ∀, and it will be represented as follows:
  ∀x man(x) → respects (x, parent).
- **3. Some boys play cricket.**
  In this question, the predicate is "**play(x, y)**," where x= boys, and y= game. Since there are some boys so we will use **∃, and it will be represented as**:
  ∃x boys(x) → play(x, cricket).
- **4. Not all students like both Mathematics and Science.**
  In this question, the predicate is "**like(x, y)," where x= student, and y= subject**.
  Since there are not all students, so we will use **∀ with negation, so** following representation for this:
  ¬∀ (x) [student(x) → like(x, Mathematics) ∧ like(x, Science)].