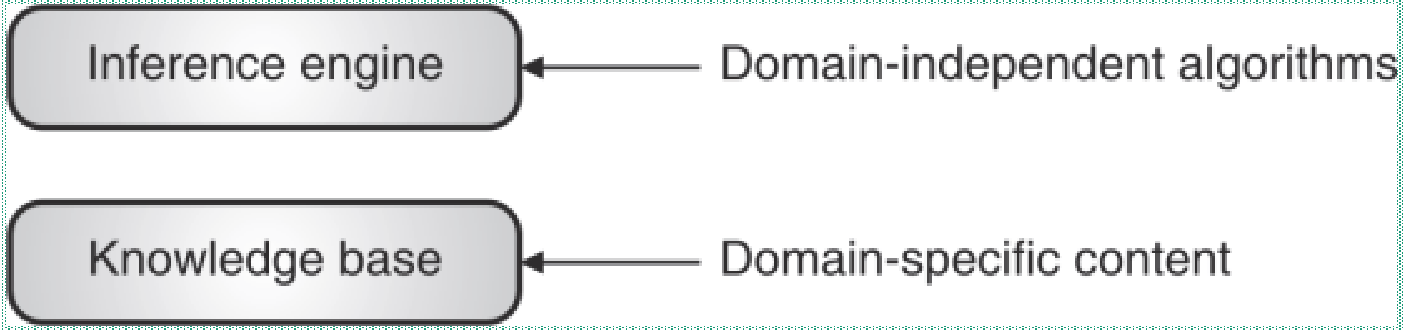
Knowledge and Reasoning

Knowledge-Based Agent in Artificial intelligence

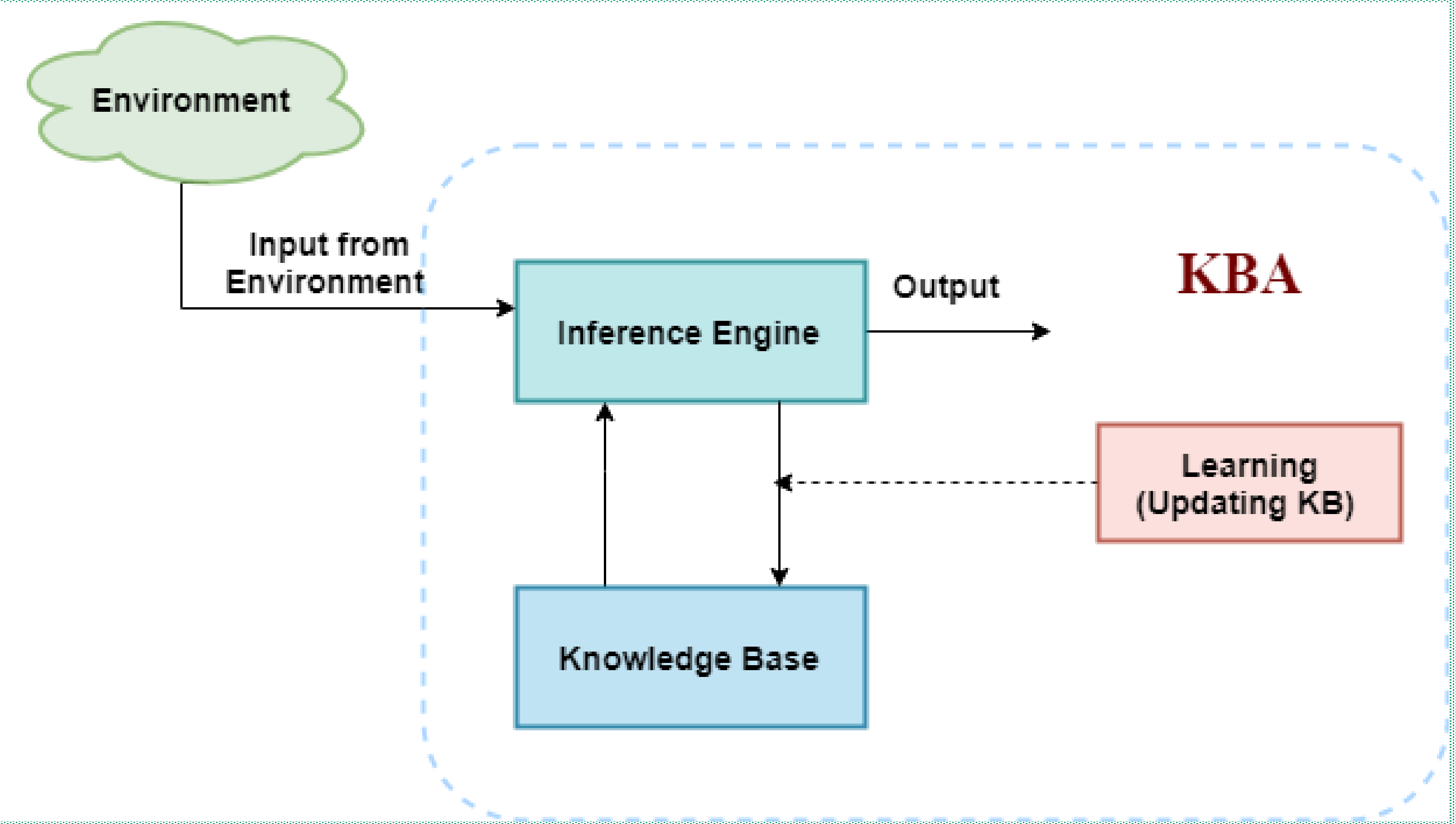
- An intelligent agent needs **knowledge** about the real world for taking decisions and **reasoning** to act efficiently.
- Knowledge-based agents are those agents who have the capability of **maintaining an internal state of knowledge, reason over that knowledge, update their knowledge after observations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

A Knowledge Based Agent



Levels of knowledge base

Architecture of a KB Agent



Knowledge-base

Knowledge-base is a central component of a knowledge-based agent. 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.

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.

Representation of Knowledge using Rules

- Knowledge can be considered to be represented at generally two levels
- **(i) Knowledge level :** This level describes the facts.
- **(ii) Symbol level :** This level deals with using the symbols for representing the objects, which can be manipulated in programs.
- Knowledge can be represented using the following rules
- (a) Logical representations
- (b) Production rule representations
- (c) Semantic networks

Ways of logical representations

- **1. Propositional logic :** These are restricted kinds that make use of propositions (sentences that are either true or false but not both) which can be either true or false. Proposition logic is also known as propositional calculus, sentential calculus or Boolean algebra.
- All propositions are either true or false, **For example :**
- (i) Leaves are green (ii) Violets are blue.

Sentence	Truth Value	Proposition
Sky is blue	true	yes
Roses are red	true	yes
2 + 2 = 5	false	yes

Ways of logical representations

- **2. First order predicate logic :** These are much more expressive and make use of variables, constants, predicates, functions and quantifiers along with the connectives.
- **3. Higher order predicate logic:** Higher order predicate logic is distinguished from first order predicate logic by using additional quantifiers and stronger semantics.
- **4. Fuzzy logic :** These indicate the existence of in between TRUE and FALSE or fuzziness in all logics.

(b) Production rule representation

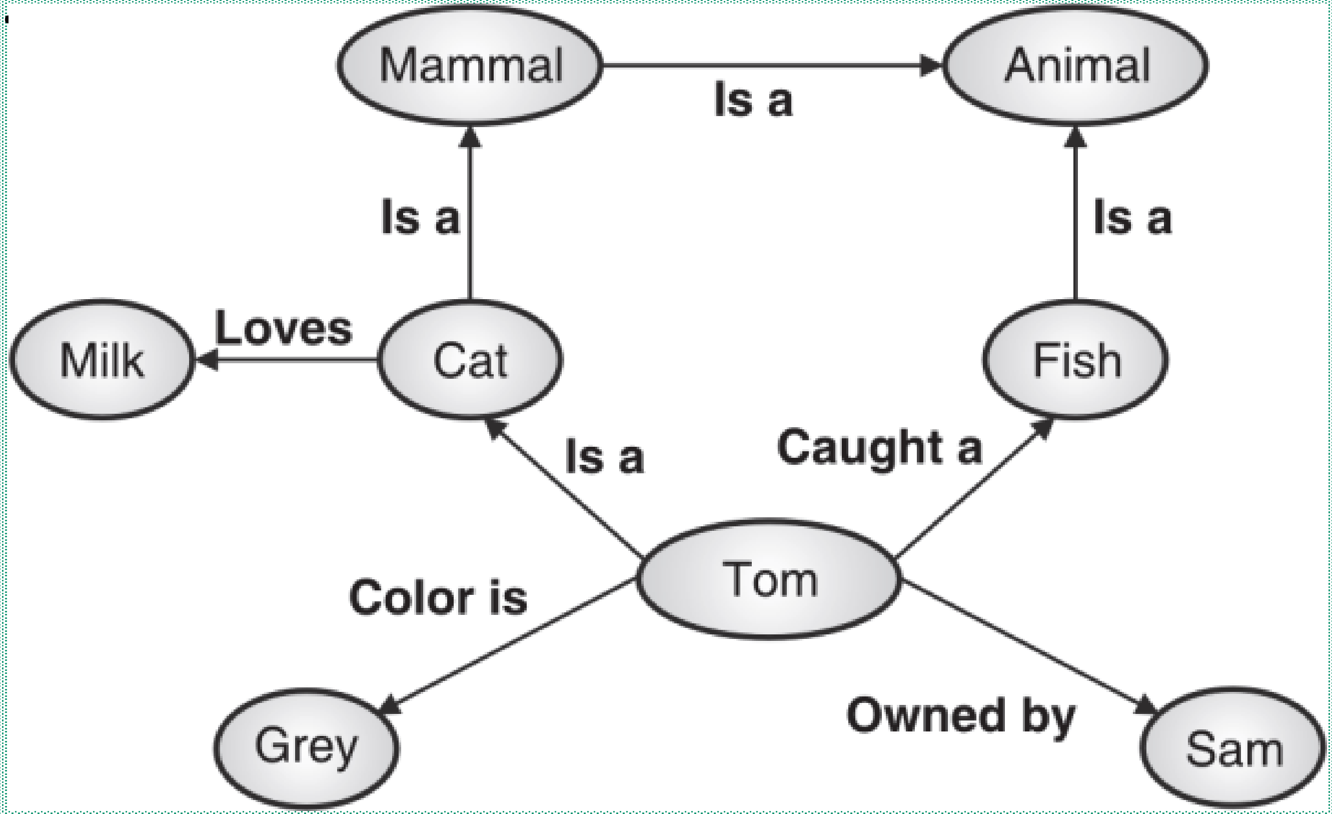
- One of the widest used methods to represent knowledge is to use production rules, it is also known as IF-THEN rules.
- **Syntax :**
- IF condition THEN action
- IF premise THEN conclusion
- IF proposition p1 and proposition p2 are true THEN proposition p3 is true
- **Example :**
- IF pressure is high, THEN volume is small.
- IF the road is slippery, THEN driving is dangerous.
- – Some of the benefits of IF-THEN rules are that they are modular, each defining a relatively small and, at least in principle, independent piece of knowledge. New rules may be added and old ones deleted usually independently of other rules.

(c) Semantic networks

- These represent knowledge in the form of graphical networks, since graphs are easy to be stored inside programs as they are concisely represented by nodes and edges.
- – A semantic network basically comprises of *nodes* that are named and represent concepts, and labelled *links* representing relations between concepts. Nodes represent both types and tokens.

Example

- For example, the semantic network in Fig. expresses the knowledge to represent the following data :
- Tom is a cat.
- Tom caught a fish.
- Tom is grey in color.
- Tom is owned by Sam.
- Tom is a Mammal.
- Fish is an Animal.
- Cats love Milk.
- All mammals are animals



Propositional Logic (PL)

- – **Propositional Logic (PL)** is simple but powerful for some artificial intelligence problems.
- – In case of artificial intelligence propositional logic is not categorized as the study of truth values, but it is based on relativity of truth values. (i.e. The relationship between the truth value of one statement to that of the truth value of other statement)

Proposition

Definition: A **proposition** is a statement that is, by itself, either true or false.

Can be either true or false.

Sample Propositions

- All humans are mortal.
- Ram is married.
- I'll pay for the meal.

Things that aren't propositions

- Come here!
- Why are you crying?

Propositional Connectives

Logical NOT: ¬ P

- Read "not P"
- ¬ P is true if and only if P is false.
- Also called **logical negation**.

Logical AND: P ∧ Q

- Read "P and Q."
- P ∧ Q is true if both P and Q are true.
- Also called **logical conjunction**.

Logical OR: P ∨ Q

- Read "P or Q."
- P ∨ Q is true if at least one of P or Q are true.
- Also called **logical disjunction**.

Propositional Connectives

Implication: $P \rightarrow Q$

- Read "If P then Q".
- False when P is true and Q is false; and is true otherwise.
- Also called **material conditional operator**.

Biconditional: $P \leftrightarrow Q$

- Read "P if and only if Q".
- True if P and Q have the same truth values; and false otherwise.
- Also called **material biconditional operator**

true and **false**:

- The symbol $\top$ is a value that is always true.
- The symbol $\bot$ is a value that is always false.

Implication

For propositions **P** and **Q**, **P → Q** , the **implication** or **conditional statement** is **false** when **P is true** and **Q is false**, and is **true otherwise**.

- P is called the premise or hypothesis.
- Q is called the conclusion.

P	Q	P → Q
F	F	T
F	T	T
T	F	F
T	T	T

We want P → Q to mean "whenever P is true, Q is true as well."

Only way this doesn't happen is if P is true and Q is false.

Biconditional

The **biconditional** of statements **P** and **Q**, denoted **$P \leftrightarrow Q$**, is read "P if and only if Q", and is **true** if **P** and **Q** have the **same truth values**, and **false otherwise**.

The biconditional operator is used to represent a two-directional implication.

P	Q	$P \leftrightarrow Q$
F	F	T
F	T	F
T	F	F
T	T	T

Specifically, $p \leftrightarrow q$ means that p implies q and q implies p.

Conversely if both P implies Q and Q implies P are true, then P if and only if Q is true.

Operator Precedence

Operator precedence for propositional logic:

$\neg$
$\wedge$
$\vee$
$\rightarrow$
$\leftrightarrow$

Translating English Into Logic

User defines a **set of propositional symbols**, like P and Q.

User defines the **semantics** of each **propositional symbol**:

P	It is hot.
Q	It is humid.
R	It is raining.

1. If it is humid, then it is hot
   $Q \rightarrow P$

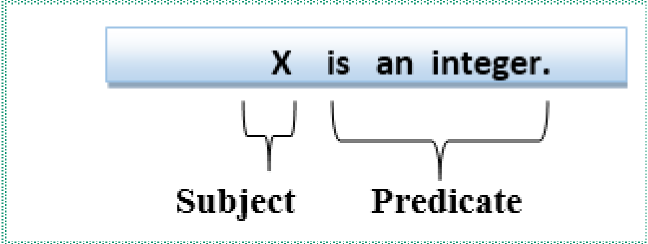
2. If it is hot and humid, then it is raining.
   $(P \wedge Q) \rightarrow R$

First-Order Logic

- Propositional Logic
  - Hard to identify "individuals"
    - E.g., Mary, 3
  - Can't directly talk about properties of individuals or relations between individuals
    - E.g., Ben is fat.
  - Generalizations, patterns, regularities can't easily be represented
    - E.g., All triangles have 3 sides.
- First-Order Logic
  - FOL or FOPC is expressive enough to concisely represent this kind of information
  - FOL adds relations, variables, and quantifiers, e.g.,
    - Every elephant is gray. : $\forall$ x (elephant(x) $\rightarrow$ gray(x))
    - There is a white alligator.: $\exists$ x (alligator(X) ^ white(X))

First-order logic allows us to get at the internal structure of certain propositions in a way that is not possible with propositional logic.

- **First-order logic statements can be divided into two parts:**
- **Subject:** Subject is the main part of the statement.
- **Predicate:** A predicate can be defined as a relation, which binds two atoms together in a statement.
- **Consider the statement: "x is an integer."**, it consists of two parts, the first part x is the subject of the statement and second part "is an integer," is known as a predicate.



Syntax of First-Order Logic

<u>FOL Provides</u>

- Variable symbols
  - E.g., x, y, foo
- Connectives
  - Same as in PL:
  - $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$
- Quantifiers
  - Universal $\forall x$
  - Existential $\exists x$

<u>User Provides</u>

- Constant symbols
  - Mary
  - Green
- Function symbols
  - father-of(Mary) = John
  - color-of(Sky) = Blue
- Predicate symbols
  - greater(5,3)
  - color(Grass, Green)

A predicate becomes a proposition when specific values are assigned to the variables

Functions evaluate to objects, not propositions when specific values are assigned to the variables

Quantifiers

- The biggest change from propositional logic to first-order logic is the use of quantifiers.
- A **quantifier** is a statement that expresses that some property is true for some or all choices that could be made.
  - Turn predicates into propositions by assigning values to all variables:
    - Predicate P(x): x is even.
    - Proposition P(6): 6 is even.
  - The other way to turn a predicate into a proposition:
    - Add a quantifier like "All" or "Some" that indicates the number of values for which the predicate is true.

A formula that contains variables is not simply true or false unless each of these variables is **bound** by a quantifier

The Universal Quantifier

Definition: The **counterexample** for $\forall x\ P(x)$ is any $t \in U$, where U is the domain of discourse, such that $P(t)$ is false.

- Example

  $\forall x,y,z$ sum(x,y,z): `z' is the sum of `x' and `y'.

  For U = non-negative integers.

  Proposition sum(1,7,8) is true.

  sum(5,1,8) is false.

The Existential Quantifier

- **Definition**: The symbol ∃ is the **existential quantifier.**
  - The existential quantification of P(x) is the statement

    P(x) for some values x in the universe, which is written in logical notation as

    $$\exists x\ P(x).$$

  - A statement of the form $\exists x\ P(x)$ asserts that for some choice of x in our domain, P(x) is true.
    - Even and prime in the number series.

      $\exists x.\ (\text{Even}(x) \land \text{Prime}(x))$

English Statment	FOL
Ram is a student	Student(Ram)
Shyam is teacher	Teacher(Shyam)
Ram takes either maths or Bio	$Takes(Ram, maths) \vee Takes(Ram, Bio)$
Ram takes Bio if only if Ram doesnot take maths	$Takes(Ram, Bio) \leftrightarrow \neg Takes(Ram, maths)$
There exist a student	$\exists x\ Student(x)$
There exist a smart student (There exists a student who is smart)	$\exists x (Student(x) \wedge smart(x))$
All ~~smart~~ students are smart	$\forall x\ student(x) \rightarrow smart(x)$

All happy people smile

$\forall x\ happ(x) \rightarrow smile(x)$

Someone is graduating

$\exists x\ graduating(x)$

John is tall

$tall(John)$

All gorillas are black

$\forall x\ gorilla(x) \rightarrow black(x)$

Some gorillas are black

$\exists x\ gorillas(x) \wedge black(x)$

Not all rainy days are cold

Rainy(d) : The day is rainy
cold(d) : The day is cold

Not all ~~days~~ rainy days are cold. This means some days which are rainy but which are not cold.

$$\boxed{\exists d\,(Rainy(d) \wedge \neg cold(d))}$$

- Every gardener likes the sun.

(Ax) gardener(x) => likes(x,Sun)

- All purple mushrooms are poisonous.

(Ax) (mushroom(x) ^ purple(x)) => poisonous(x)

- Deb is not tall.

~tall(Deb)

Inference engine:

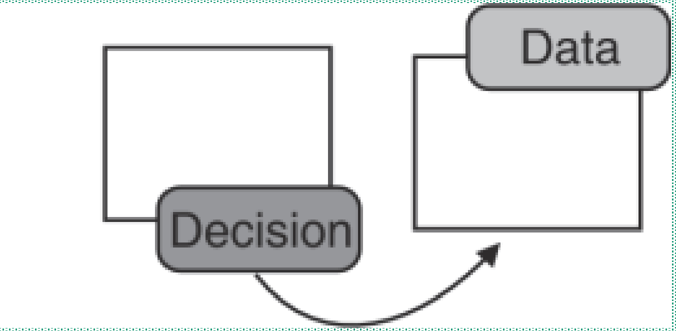
- The inference engine is the component of the intelligent system in artificial intelligence, which applies logical rules to the knowledge base to infer new information from known facts.
- Inference engine commonly proceeds in two modes, which are:

1. **Forward chaining**
2. **Backward chaining**

A. Forward Chaining

- Forward chaining is a form of reasoning which start with atomic sentences in the knowledge base and applies inference rules (Modus Ponens) in the forward direction to extract more data until a goal is reached.
- The Forward-chaining algorithm starts from known facts, triggers all rules whose premises are satisfied, and add their conclusion to the known facts. This process repeats until the problem is solved.
- **Properties of Forward-Chaining:**
- It is a down-up approach, as it moves from bottom to top.
- It is a process of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.
- Forward-chaining approach is also called as data-driven as we reach to the goal using available data.

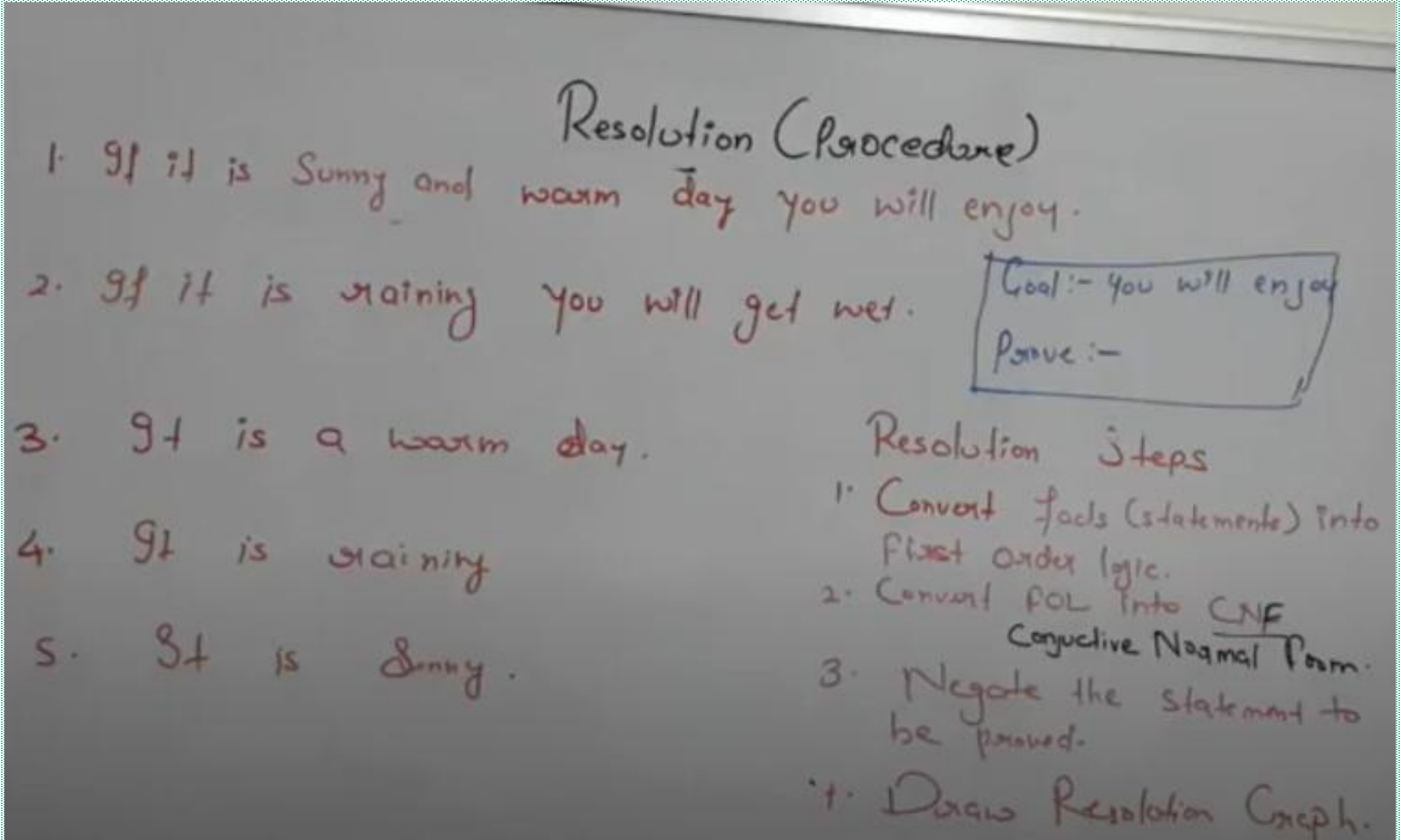
Backward Chaining



If based on the decision the initial data is fetched, then it is called as **backward chaining**. Backward chaining or goal-driven inference works towards a final state, and by looking at the working memory to see if goal already there

For example, If while going out one has taken umbrella. Then based on this decision it can be guessed that it is raining.
Here, “taking umbrella” is a decision based on which the data is generated that “it's raining”. This process is **backward chaining.** “Backward chaining” is called as a decision-driven or goal-driven inference technique.

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



Resolution (Procedure)

1. If it is Sunny and warm day you will enjoy.

   Sunny $\wedge$ warm $\rightarrow$ enjoy

2. If it is raining you will get wet.

   raining $\rightarrow$ wet

3. It is a warm day

   Warm

4. It is raining

   raining

5. It is Sunny.

   Sunny

Goal :- you will enjoy

Prove :-

Resolution Steps

1. Convert facts (statements) into First order logic.
2. Convert FOL into CNF
   Conjuctive Normal Form.
3. Negate the statement to be proved.
4. Draw Resolution Graph.

Resolution (Procedure)

1. If it is Sunny and warm day you will enjoy.

   $Sunny \wedge warm \rightarrow enjoy$

2. If it is raining you will get wet.

   $raining \rightarrow wet$

3. It is a warm day.

   Warm

4. It is raining

   raining

5. It is Sunny.

   Sunny

Goal :- you will enjoy

Prove :-

① ~~¬(sunny ∧ warm) ∨ enjoy~~

¬Sunny ∨ ¬warm ∨ enjoy

② ¬raining ∨ wet

2. Convert FOL into CNF

   Conjuctive Normal Form.

3. Negate the statement to be proved.

4. Draw Resolution Graph.

Resolution (Procedure)

1. If it is Sunny and warm day you will enjoy.

   $Sunny \wedge warm \rightarrow enjoy$

2. If it is raining you will get wet.

   $raining \rightarrow wet$

3. It is a warm day

   Warm

4. It is raining

   raining

5. It is Sunny.

   Sunny

Assume:- $\neg enjoy$ ×

Goal:- you will enjoy

Prove:- enjoy

① $\neg(sunny \wedge warm) \vee enjoy$

$\neg Sunny \vee \neg warm \vee enjoy$

② $\neg raining \vee wet$

3. Negate the statement to be proved.

4. Draw Resolution Graph.

Resolution (Procedure)

1. If it is Sunny and warm day you will enjoy.

   $Sunny \wedge warm \rightarrow enjoy$

2. If it is raining you will get wet.

   $raining \rightarrow wet$

3. It is a warm day.

   $warm$

4. It is raining

   $raining$

5. It is Sunny.

   $Sunny$

Assume :- $\neg enjoy$

Goal :- you will enjoy

Prove :- enjoy

① $\neg(sunny \wedge warm) \vee enjoy$

$\neg Sunny \vee \neg warm \vee enjoy$

② $\neg raining \vee wet$

$\neg enjoy$ $\qquad$ $\neg Sunny \vee \neg warm \vee enjoy$

Resolution (Procedure)

1. If it is Sunny and warm day you will enjoy.

   $Sunny \wedge warm \rightarrow enjoy$

2. If it is raining you will get wet.

   $raining \rightarrow wet$

3. It is a warm day

   $warm$

4. It is raining

   $raining$

5. It is Sunny.

   $Sunny$

Assume :- $\neg enjoy$

Goal :- you will enjoy

Prove :- enjoy

① $\neg(sunny \wedge warm) \vee enjoy$

$\neg Sunny \vee \neg warm \vee enjoy$

② $\neg raining \vee wet$

$\cancel{\neg enjoy}$ $\qquad$ $\neg Sunny \vee \neg warm \vee \cancel{enjoy}$

$\neg Sunny \vee \neg warm$

Resolution (Procedure)

1. If it is Sunny and warm day you will enjoy.

   $Sunny \land warm \rightarrow enjoy$

2. If it is raining you will get wet.

   $raining \rightarrow wet$

3. It is a warm day

   $warm$

4. It is raining

   $raining$

5. It is Sunny.

   $Sunny$

Assume:- $\neg enjoy$

$\boxed{\text{Goal:- you will enjoy} \quad \text{Prove:- enjoy}}$

① $\cancel{\neg(sunny \land warm) \lor enjoy}$

$\neg Sunny \lor \neg warm \lor enjoy$

② $\neg raining \lor wet$

$\cancel{\neg enjoy}$ ① $\neg Sunny \lor \neg warm \lor \cancel{enjoy}$

$\neg Sunny \lor \cancel{\neg warm}$ ③ $\cancel{warm}$

$\neg \cancel{Sunny}$ ⑤ $\cancel{Sunny}$

$\rightarrow \square$

Resolution (Procedure)

1. If it is Sunny and warm day you will enjoy.

$Sunny \wedge warm \rightarrow enjoy$

2. If it is raining you will get wet.

$raining \rightarrow wet$

3. It is a warm day

warm

4. It is raining

raining

5. It is Sunny.

Sunny

Assume:- $\neg enjoy$

Goal:- You will enjoy

Prove:- enjoy

① $\neg(sunny \wedge warm) \vee enjoy$

$\neg Sunny \vee \neg warm \vee enjoy$

② $\neg raining \vee wet$

$\cancel{\neg enjoy}$ ① $\neg Sunny \vee \neg warm \vee \cancel{enjoy}$

$\neg Sunny \vee \cancel{\neg warm}$ ③ $\cancel{warm}$

$\neg Sunny$ ⑤ Sunny

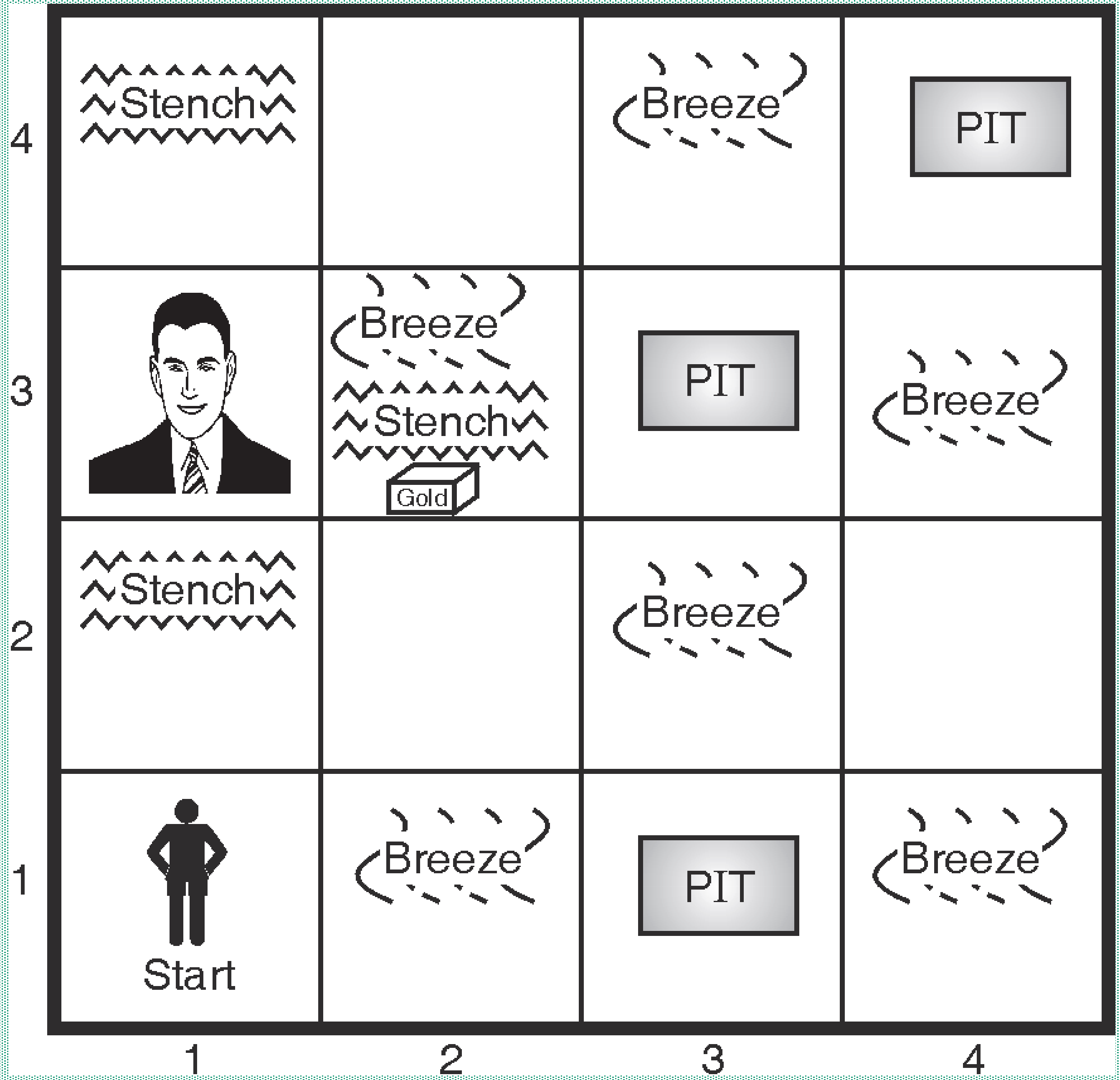
Empty clause (Contradiction) $\longrightarrow \square$

The WUMPUS World Environment

- WUMPUS is an early computer game also known as “Hunt the Wumpus”. WUMPUS was developed by Gregory Yob in 1972/1973. It was originally written in **BASIC (Beginner's All-purpose Symbolic Instruction Code).**
- – WUMPUS is a map-based game. Let's understand the game :
- WUMPUS world is like a cave, which represents number of rooms, rooms, which are connected by passageways.
- We will take a 4 × 4 grid to understand the game.
- WUMPUS is a monster who lives in one of the rooms of the cave. WUMPUS eats the player (agent) if player
- (agent) comes in the same room. Fig. shows that room (3, 1) where WUMPUS is staying.
- Player (agent) starts from any random position in cave and has to explore the cave. We are starting from (1, 1) position.
- – There are various sprites in the game like pit, stench, breeze, gold, and arrow. Every sprite has some feature. Let’s understand this one-by-one :

The WUMPUS World Environment

- Few rooms have bottomless pits which trap the player (agent) if he comes to that room. You can see in the Fig. that room (1,3), (3,3) and (4,4) have bottomless pit. Note that even WUMPUS can fall into a pit.
- Stench experienced in a room which has a WUMPUS in its neighbourhood room. See the Fig, here room (2,1), (3,2) and (4,1) have Stench.
- Breeze is experienced in a room which has a pit in its neighbourhood room. Fig. 3 shows that room (1,2), (1,4), (2,3), (3,2), (3,4) and (4,3) consists of Breeze.
- Player (Agent) has arrows and he can shoot these arrows in straight line to kill WUMPUS.
- One of the rooms consists of gold, this room glitters. Fig. shows that room (3, 2) has Gold.
- Apart from above features player (agent) can accept two types of percepts which are: Bump and scream. A bump is generated if player (agent) walks into a wall. While a sad scream created everywhere in the cave when the WUMPUS is killed.


4
3
2
1
Stench
Breeze
PIT
Breeze
Stench
Gold
PIT
Breeze
Stench
Breeze
Start
Breeze
PIT
Breeze
1
2
3
4

Description of the WUMPUS World

- An agent receives percepts while exploring the rooms of cave. Every percepts can be represented with the help of five element list, which is [stench, breeze, glitter, bump, scream]. Here, player (agent) cannot perceive its own location.
- – If the player (agent) gets percept as [Stench, Breeze, None, None, None]. Then it means that there is a stench and a breeze, but no glitter, no bump, and no scream in the WUMPUS world at that position in the game.
- – Let's take a look at the actions which can be performed by the player(agent) in WUMPUS World :
- o **Move :** To move in forward direction,
- o **Turn :** To turn right by 90 degrees or left by 90 degrees,
- o **Grab :** To pick up gold if it is in the same room as the player(agent),
- o **Shoot :** To Shoot an arrow in a straight line in the direction faced by the player (agent)

Description of the WUMPUS World

- These actions are repeated till the player (agent) kills the WUMPUS or if the player (agent) is killed. If the WUMPUS is killed then it is a winning condition, else if the player (agent) is killed then it is a losing condition and the game is over.
- – Game developer can keep a restriction on the number of arrows which can be used by the player (agent). So if we allow agent to have only one arrow, then only the first shoot action will have some effect. If this shoot action kills the WUMPUS then you win the game, otherwise it reduces the probability of winning the game.
- – Lastly there is a die action : It takes places automatically if the agent enters in a room with a bottomless pit or in a room with WUMPUS.

Goal of the game

- Main aim of the game is that player (agent) should grab the gold and return to starting room (here its (1,1)) without being killed by the monster (WUMPUS).
- – Award and punishment *points* are assigned to a player (Agent) based on the actions it performs. Points can be given as follows :
- o 100 points are awarded if player (agent) comes out of the cave with the gold.
- o 1 point is taken away for every action taken.
- o 10 points are taken away if the arrow is used.
- o 200 points are taken away if the player (agent) gets killed.

Give PEAS descriptors for WUMPUS world **(May 13, Dec. 14)**

- **1. Performance measure**
- + 100 for grabbing the gold and coming back to the starting position,
- – 200 if the player (agent) is killed.
- – 1 per action,
- – 10 for using the arrow.
- **2. Environment**
- – Empty Rooms.
- – Room with WUMPUS.
- – Rooms neighbouring to WUMPUS which are smelly.
- – Rooms with bottomless pits
- – Rooms neighbouring to bottomless pits which are breezy.
- – Room with gold which is glittery.
- – Arrow to shoot the WUMPUS.

Give PEAS descriptors for WUMPUS world

- **Sensors (assuming a robotic agent)**
- – Camera to get the view
- – Odour sensor to smell the stench
- – Audio sensor to listen to the scream and bump
- **4. Effectors (assuming a robotic agent)**
- – Motor to move left, right
- – Robot arm to grab the gold
- – Robot mechanism to shoot the arrow