Unit-1

Introduction to AI

Why Study AI?

- AI makes computers more useful
- Intelligent computer would have huge impact on civilization
- AI cited as “field I would most like to be in” by scientists in all fields
- Computer is a good metaphor for talking and thinking about intelligence
- Turning theory into working programs forces us to work out the details
- AI yields good results for Computer Science
- AI yields good results for other fields
- Computers make good experimental subjects

What is the definition of AI?

- It is a branch of Computer Science that pursues creating the computers or machines as intelligent as human beings.
- It is the science and engineering of making intelligent machines, especially intelligent computer programs.
- Artificial Intelligence is the study of how to make computers do things, which, at the moment, people do better.
- According to the father of Artificial Intelligence, **John McCarthy**, it is “The science and engineering of making intelligent machines, especially intelligent computer programs”.
- Artificial Intelligence is a way of making a computer, a computer-controlled robot, or a software think intelligently, in the similar manner the intelligent humans think.

What is the definition of AI?

- Artificial Intelligence is concerned with the design of intelligence in an artificial device.
- The term was coined by McCarthy in 1956.
- There are two ideas in the definition.

1. Intelligence
2. artificial device

What is the definition of AI?

- AI is that sector in computer science that emphasizes the creation of intelligent machines that work, operate and react like human beings.
- AI is used in decision making by the machines considering the real-time scenario. An Artificially Intelligent machine reads the real-time data, understands the business scenario and reacts accordingly.
- Some of the activities that the artificially intelligent machines are designed for are:

➢ Speech recognition

➢ Learning

➢ Planning

➢ Problem-solving

What is the definition of AI?

Charniak & McDermott, 1985

"The study of mental faculties through the use of computational models"

Bellman, 1978

"[The automation of] activities that we associate with human thinking, activities such as decision making, problem solving, learning"

Dean et al., 1995

"The design and study of computer programs that behave intelligently. These programs are constructed to perform as would a human or an animal whose behavior we consider intelligent"

Haugeland, 1985

"The exciting new effort to make computers think *machines with minds*, in the full and literal sense"

Kurzweil, 1990

"The art of creating machines that perform functions that require intelligence when performed by people"

What is the definition of AI?

Nilsson, 1998

“Many human mental activities such as writing computer programs, doing mathematics, engaging in common sense reasoning, understanding language, and even driving an automobile, are said to demand intelligence. We might say that [these systems] exhibit artificial intelligence”

Rich & Knight, 1991

“The study of how to make computers do things at which, at the moment, people are better”

Schalkoff, 1990

“A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes”

Winston, 1992

“The study of the computations that make it possible to perceive, reason, and act”

Goals of Artificial Intelligence

- To Create Intelligent & Expert Systems: The development began to making systems that exhibit intelligent behavior. The expected functions of these machines are learning, demonstrating, explaining, and advising its users.
- To Inculcate Human Intelligence into the Machines: Creating systems and developing software that understands, think, learn, and behave like humans.

History of AI

- 1950 : The Turing test, originally called the imitation game by Alan Turing
- 1956 : The Scientist John McCarthy defined the term AI
- 1981 : Japan was the first to take this initiative and launched a scheme called 5th Generation

➢ Britain created a project called “Elvi” for this scheme

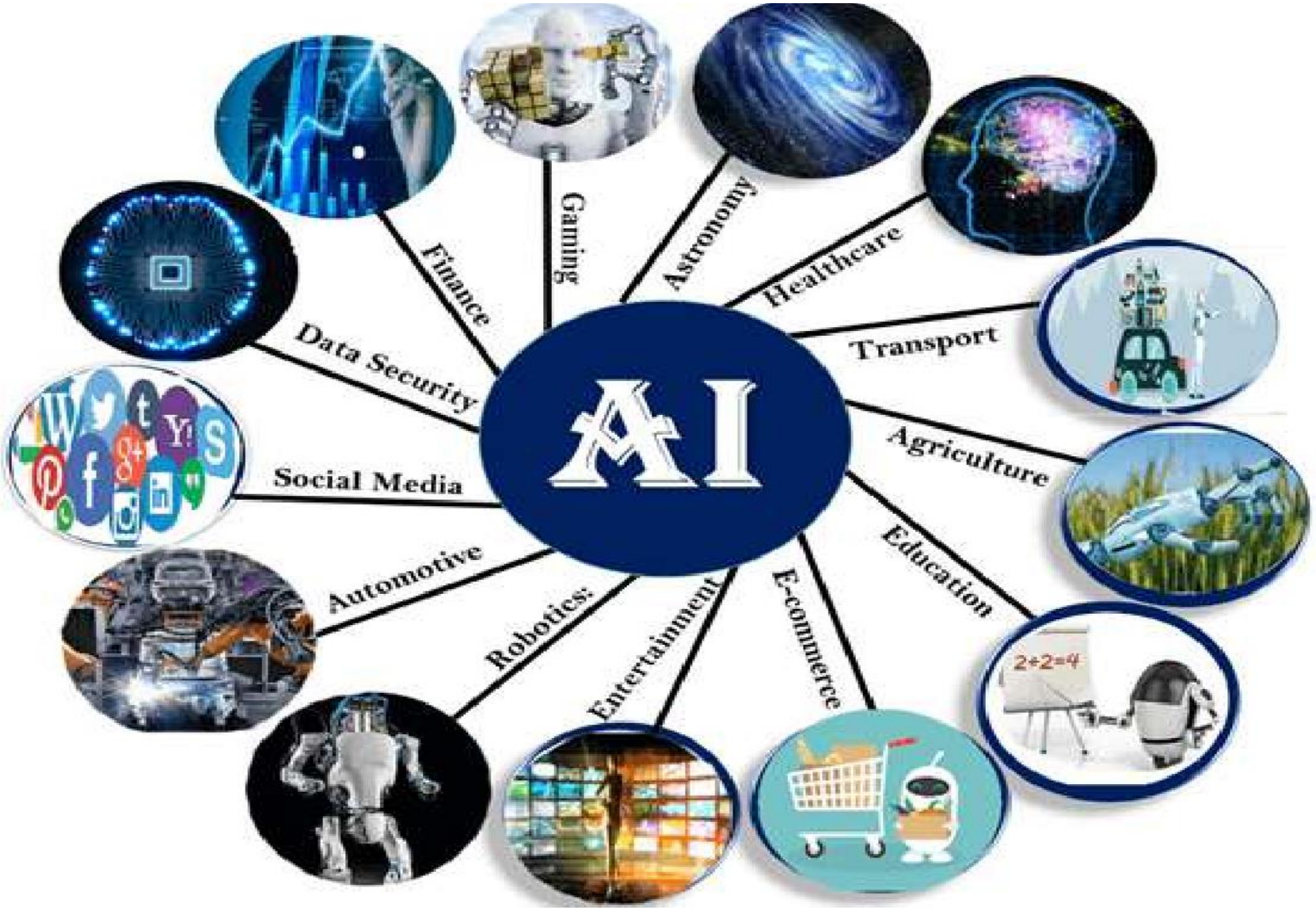
➢ European Union countries also started a program called “Esprit”

- 1983 : some private organizations jointly established a consortium “ Micro Electronics and Computer Technology” to develop advanced technologies applicable to AI such as VLSI Circuits

History of AI

- 2010 : Google DeepMind was british company founded in 2010. Its headquartered in London. In 2014, the American Company Google bought it and its name was changed from DeepMind to Google DeepMind.
- 2011 : Apple has launched Siri
- 2014 : Amazon has launched Alexa
- 2016 : AlphaGo program by Google's DeepMind AI, has won Korean Lee Sedol, one of Go's most dominant player.
- 2016 : Google Assistant is an AI powered virtual assitant developed by Google
- 2016 : Sophia Robot in HongKong

Applications



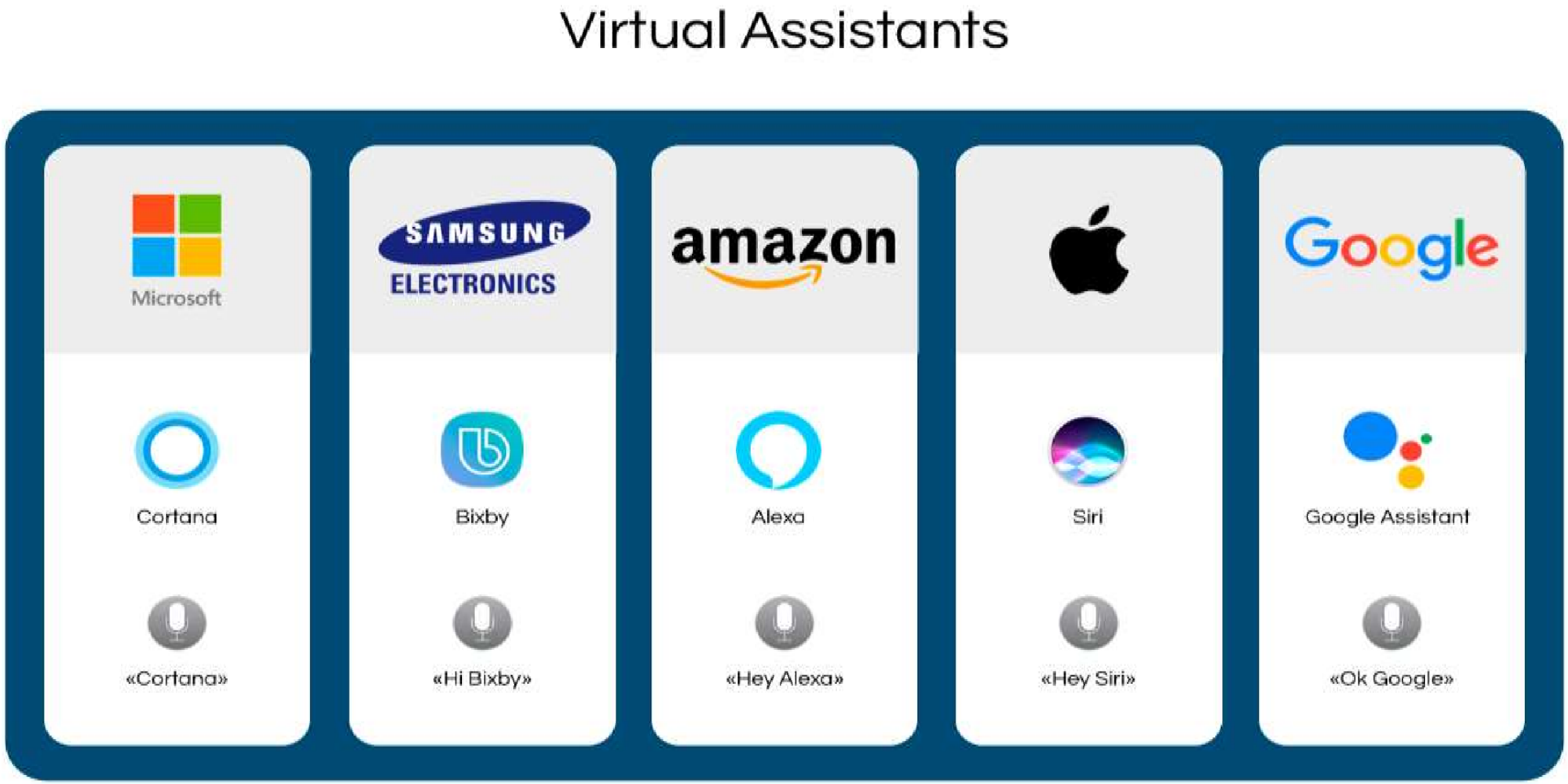
Applications

- AI is used in the **finance** industry, where personal data is collected, which can be later used to provide financial advice.
- AI is used in the field of **education**, where the grading system can be automated, and the performance of the students can be assessed based on which the learning process can be improved.
- In the field of **Healthcare**, AI is used to perform a better diagnosis where the technologies used to understand the natural language and respond to the questions asked. Also, computer programs like chatbots are used to assist customers in scheduling appointments and ease of billing process, etc.
- AI is used in **Business** to automate the repetitive tasks performed by humans with the help of Robotic Process Automation. To increase customer satisfaction, machine learning algorithms are integrated with analytics to gather information which helps in understanding customer needs.
- AI is used in **Smart Home devices**, **security and surveillance, navigation and travel, music and media streaming and video games**, etc.

Examples

- Self-driving cars
- Robo-advisors
- Conversational bots
- Email spam filters
- Netflix's recommendations
- A drone, spying camera or a spying airplane takes photographs, videos, which are used to understand the map of the area or figure out spatial information.
- Clinical expert systems use cameras inside the body and are often used by the doctors to diagnose the patient.
- Use of computer software is used in Police investigations for facial recognition. This program can identify the face of the suspect having a record in the police system called with the portrait mode with the description the witness gives to the forensic artist.

Examples


Virtual Assistants
Microsoft
Cortana
«Cortana»
SAMSUNG
ELECTRONICS
Bixby
«Hi Bixby»
amazon
Alexa
«Hey Alexa»
Siri
«Hey Siri»
Google
Google Assistant
«Ok Google»

Examples


LEFT REARWARD VEHICLE CAMERA
MEDIUM RANGE VEHICLE CAMERA
RIGHT REARWARD VEHICLE CAMERA

AI, a Multi-disciplinary domain

- Engineering: robotics, vision, control-expert systems, biometrics,
- Computer Science: AI-languages , knowledge representation, algorithms, …
- Pure Sciences: statistics approaches, neural nets, fuzzy logic, …
- Linguistics: computational linguistics, phonetics en speech, …
- Psychology: cognitive models, knowledge-extraction from experts, …
- Medicine: human neural models, neuro-science,...

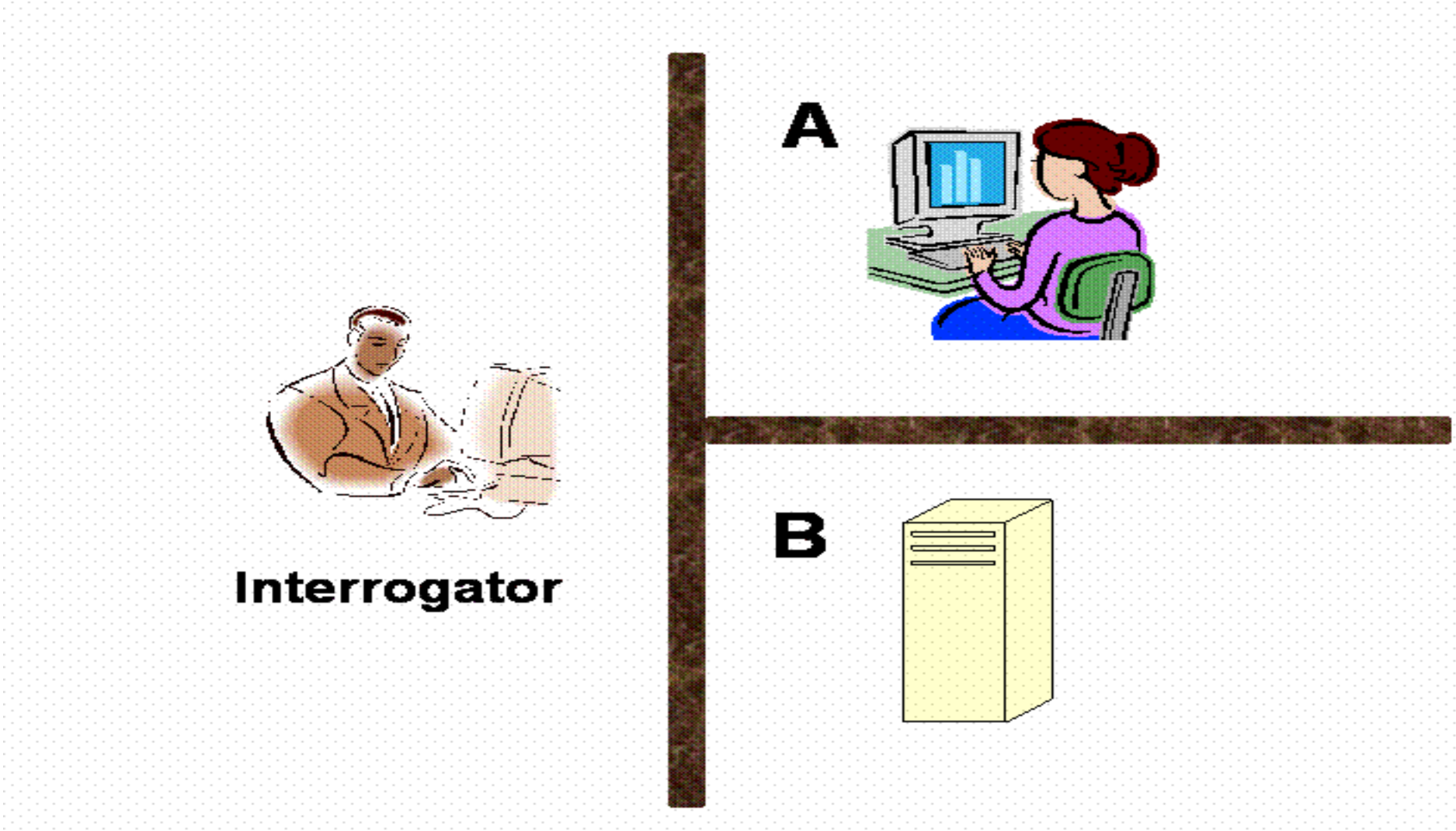
Approaches of AI

Systems that think like humans	Systems that think rationally
Systems that act like humans	Systems that act rationally

Scope and view of Artificial Intelligence

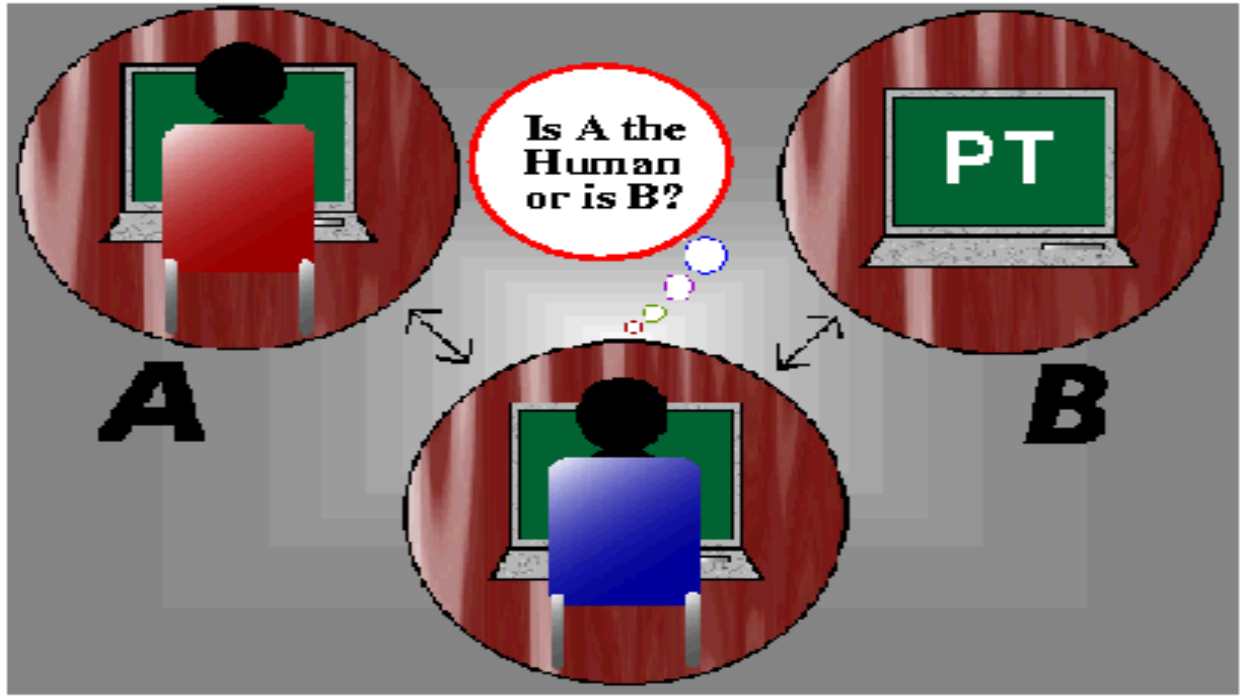
1. One view is that artificial intelligence is about designing systems that are as intelligent as humans. (emulate the human thought process)
2. The second approach is best embodied by the concept of the Turing Test. (Alan Turing, 1912)
3. Logic and laws of thought deals with studies of ideal or rational thought process and inference.
   - Knowledge representation
4. The fourth view of AI is that it is the study of rational agents. This view deals with building machines that act rationally (sensibly/logically).

Turing Test (The Imitation Game)



Turing Test : Acting Humanly

- Turing test: ultimate test for acting humanly
  - Computer and human both interrogated by judge
  - Computer passes test if judge can't tell the difference



Turing test

- In 1950, Alan Turing Proposed the following method for determining whether a machine can think. His method has since become known as the Turing Test.
- To conduct this Test, we need two people and the machine to be evaluated. One person plays the role of the interrogator, who is in a separate room from the computer and the other person.
- The interrogator can ask questions of either the person or the computer by typing questions and receiving types responses. However interrogator knows them only as A and B and aims to determine which is the person and which is the machine.
- The goal of the machine is to fool the interrogator into believing that it is the person. If the machine is succeeds at this, then we will conclude that the machine can think.

Typical AI problems

- "common-place" tasks (*These tasks are done routinely by people and some other animals*)
  - *Recognizing people, objects.*
  - Communicating (through *natural language).*
  - *Navigating around obstacles on the streets*
- "expert tasks" (*can only be performed by skilled specialists*)
  - Medical diagnosis
  - Mathematical problem solving
  - Playing games like chess

- Which of these tasks are easy and which ones are hard?
  - Computer system can perform sophisticated tasks like medical diagnosis, performing symbolic integration, proving theorems and playing chess
  - It has proved to be very hard to make computer systems perform many routine tasks that all humans and a lot of animals can do.

Intelligent behavior

- What constitutes intelligent behavior?
- Some of these tasks and applications are:
  - Perception involving image recognition and computer vision
  - Reasoning
  - Learning
  - Understanding language involving natural language processing, speech processing
  - Solving problems
  - Robotics

Which of these exhibits intelligence?

- You beat somebody at chess.
- You prove a mathematical theorem using a set of known axioms.
- You need to buy some supplies, meet three different colleagues, return books to the library, and exercise. You plan your day in such a way that everything is achieved in an efficient manner.
- You are a lawyer who is asked to defend someone. You recall three similar cases in which the defendant was guilty, and you turn down the potential client.
- A stranger passing you on the street notices your watch and asks, “Can you tell me the time?” You say, “It is 3:00.”
- You are told to find a large Phillips screwdriver in a cluttered workroom. You enter the room (you have never been there before), search without falling over objects, and eventually find the screwdriver.

Which of these exhibits intelligence?

- You are a six-month-old infant. You can produce sounds with your vocal organs, and you can hear speech sounds around you, but you do not know how to make the sounds you are hearing. In the next year, you figure out what the sounds of your parents' language are and how to make them.
- You are a one-year-old child learning Arabic. You hear strings of sounds and figure out that they are associated with particular meanings in the world. Within two years, you learn how to segment the strings into meaningful parts and produce your own words and sentences.
- Someone taps a rhythm, and you are able to beat along with it and to continue it even after it stops.
- You are some sort of primitive invertebrate. You know nothing about how to move about in your world, only that you need to find food and keep from bumping into walls. After lots of reinforcement and punishment, you get around just fine.

Practical Impact of AI

- AI components are embedded in numerous devices
  - Copy machines (for automatic copy quality improvement)
  - Identifying credit card fraud
  - Advising doctors
  - Recognizing speech
  - Helping complex planning tasks
  - Game playing
  - ….

What can AI systems do

- Today's AI systems have been able to achieve limited success in some of these tasks
  - In Computer vision, the systems are capable of face recognition
  - In Robotics, we have been able to make vehicles that are mostly autonomous.
  - In Natural language processing, we have systems that are capable of simple machine translation.
  - Today's Expert systems can carry out medical diagnosis in a narrow domain
  - Speech understanding systems are capable of recognizing several thousand words
  - Planning and scheduling systems
  - The Learning systems are capable of doing text categorization into about a 1000 topics
  - In Games, AI systems can play at the Grand Master level in chess (world champion), checkers, etc.

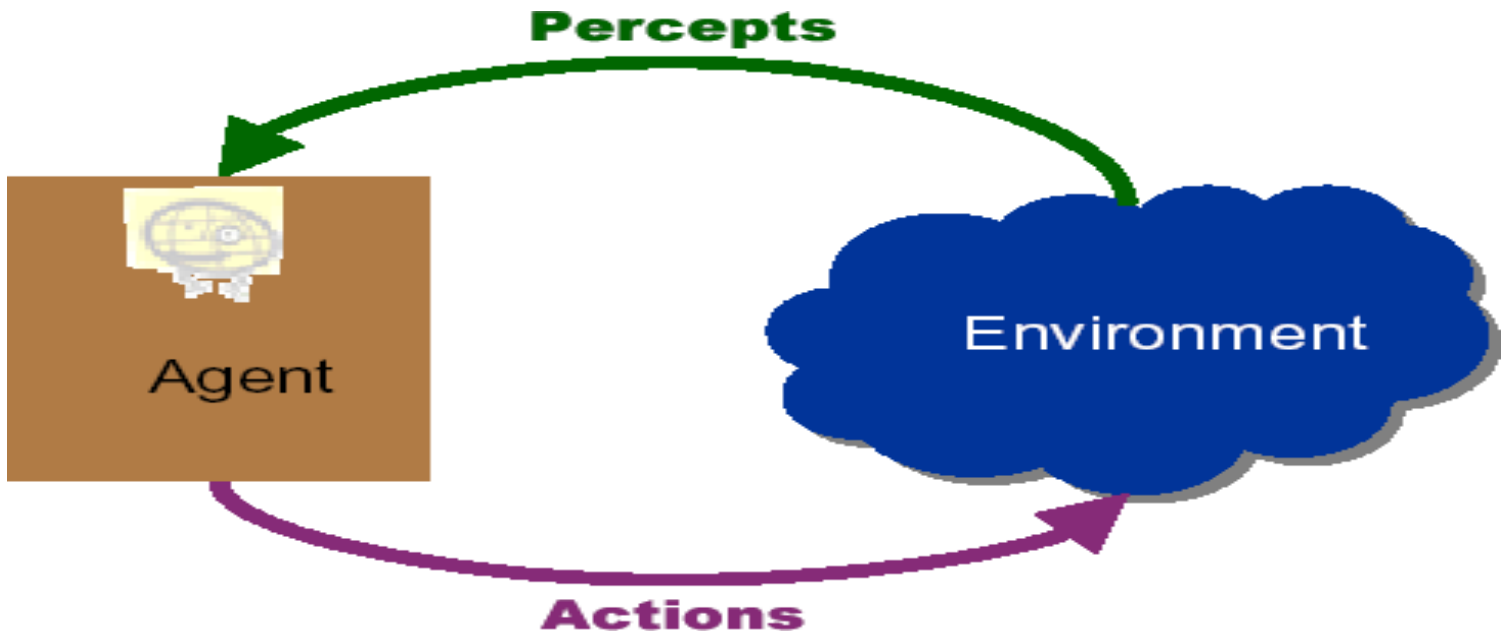
Introduction to Agent

Agents

- Agents in Artificial Intelligence are the associated concepts that the AI technologies work upon.
- The AI software or AI-enabled devices with sensors generally captures the information from the environment setup and process the data for further actions.
- There are mainly two ways the agents interact with the environment, such as perception and action. The perception is only passive for capturing the information without changing the actual environment, whereas action is the active form of interaction by changing the actual environment.
- AI technologies such as virtual assistance chat boats, AI-enabled devices to work based on the previous persecution data processing and learning for the actions.

What is an Agent

- An Agent is anything that takes actions according to the information that it gains from the environment. A human agent has sensory organs to sense the environment and the body parts to act while a robot agent has sensors to perceive the environment.
- An agent acts in an environment
- An agent perceives its environment through sensors.



How does the Agent Interact with the Environment?

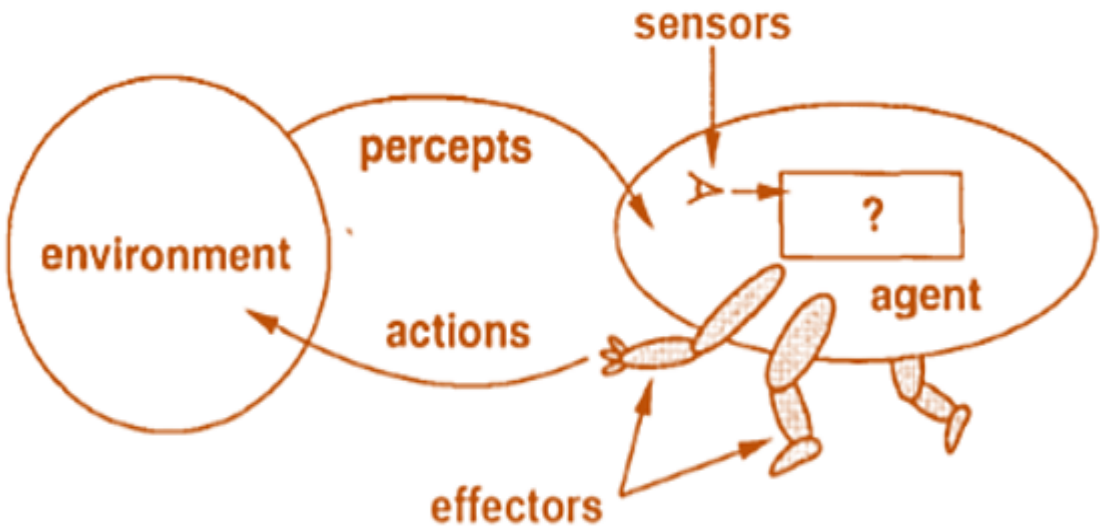
- The agents interact with the environment in two ways:

➢ Perception: Perception is a passive interaction, where the agent gains information about the environment without changing the environment. The sensors of the robot help it to gain information about the surroundings without affecting the surrounding. Hence, gaining information through sensors is called perception.

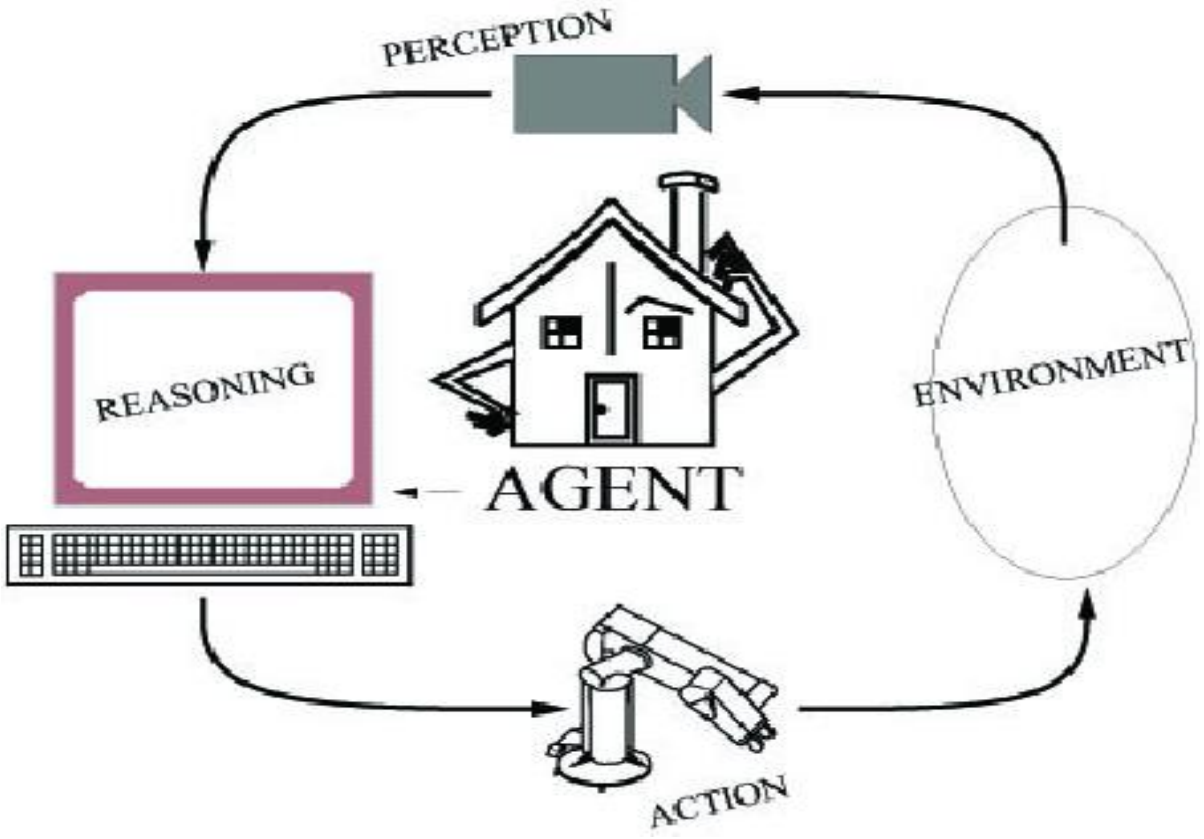
➢ Action: Action is an active interaction where the environment is changed. When the robot moves an obstacle using its arm, it is called an action as the environment is changed. The arm of the robot is called an “Effector” as it performs the action.

How does the Agent Interact with the Environment?

- The interaction of the Agent with the Environment is through Sensors and Effectors.
- Consider the example of a chatbot which is a virtual assistant. When it reads and understands the meaning of a user's messages, it is called perception. And when it replies to the user after analyzing the user's message, it is called the action.



How does the Agent Interact with the Environment?



An agent perceives its environment through sensors and acts on the environment through actuators.

Human: sensors are eyes, ears, actuators (effectors) are hands, legs, mouth.

Robot: sensors are cameras, sonar, lasers, ladar, bump, effectors are grippers, manipulators, motors

The agent's behavior is described by its function that maps percept to action.

Examples of Agents

- Humans : They have eyes, ears, skin, taste buds, etc. for sensors; and hands, fingers, legs, mouth for effectors.
- Robots : Robots may have camera, sonar, infrared, bumper, etc. for sensors. They can have grippers, wheels, lights, speakers, etc. for actuators.
- Software agents (softbots): have some functions as sensors and some functions as actuators.
- Expert systems
- Autonomous spacecrafts

PEAS

- Use PEAS to describe task environment
  - Performance measure
  - Environment
  - Actuators
  - Sensors
- Example: Taxi driver
  - Performance measure: safe, fast, comfortable (maximize profits)
  - Environment: roads, other traffic, pedestrians, customers
  - Actuators: steering, accelerator, brake, signal, horn
  - Sensors: cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors

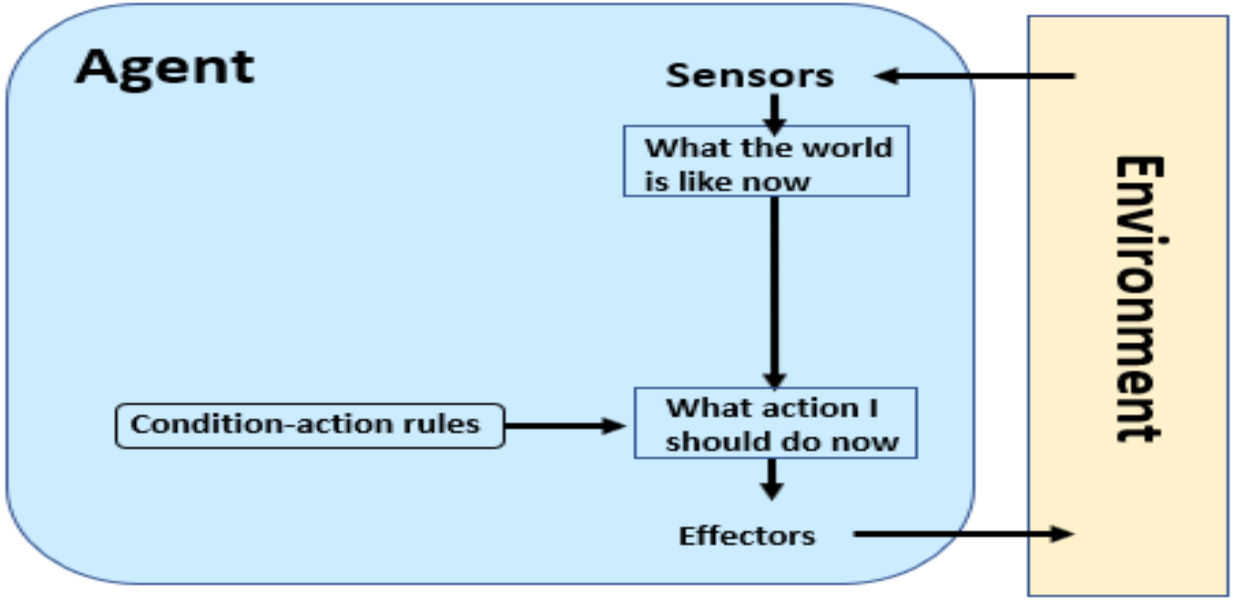
Agent Types

- Depending on the problem statement and ability to perceive the agents can be categorized into 5 categories.

- **Simple Reflex agent**: works on current perception
- **Reflex Agent With State**: represents the current state based on history.
- **Goal-based agents**: They are proactive agents and works on planning and searching.
- **Utility-based agents**: Have extra component of utility measurement over goal-based agent
- **Learning agent**: able to learn and adapt the new decision-making capabilities based on experience.

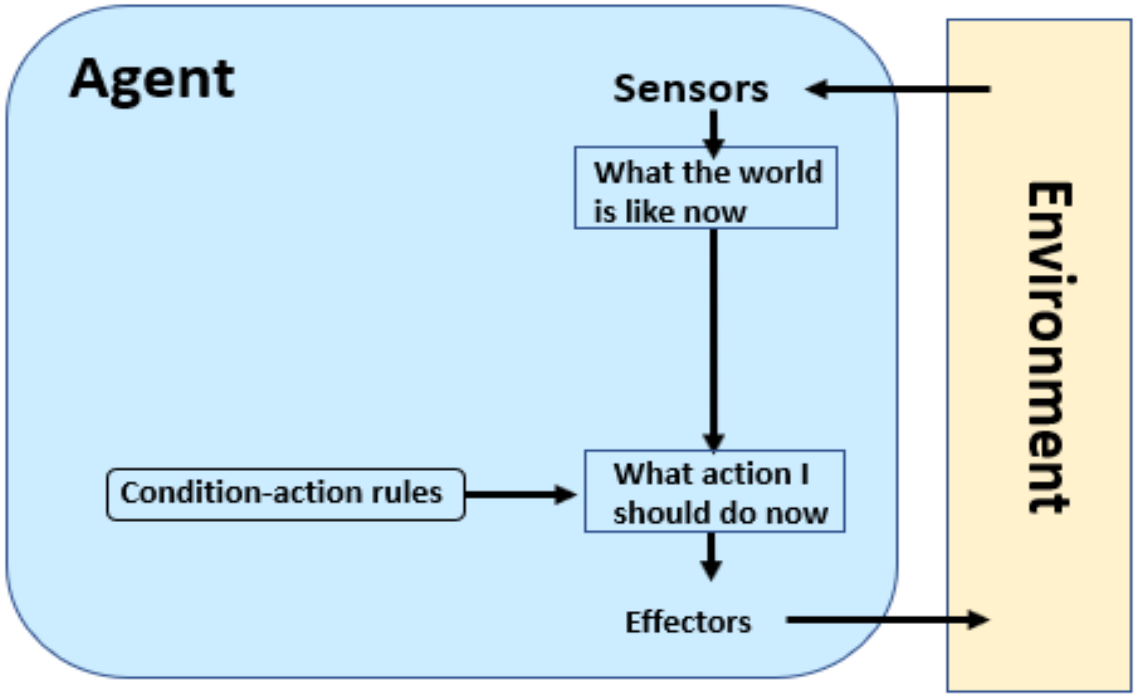
Reflex Agent

- Reflex Agent works similar to our body's reflex action (e.g. when we immediately lift our finger when it touches the tip of the flame). Just as the prompt response of our body based on the current situation, the **agent also responds based on the current environment irrespective of the past state of the environment.** The reflex agent can work properly only if the decisions to be made are based on the **current percept**.



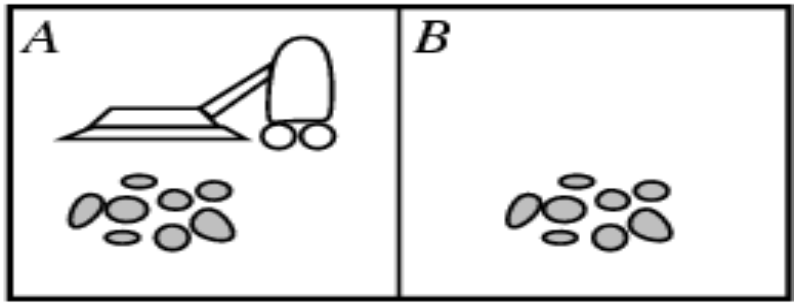
Simple Reflex Agent

- Use simple “if then” rules
- Can be short sighted




```
SimpleReflexAgent(percept)
  state  = InterpretInput(percept)
  rule   = RuleMatch(state, rules)
  action = RuleAction(rule)
  Return action
```

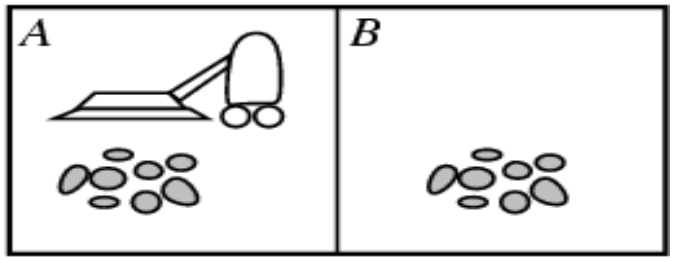
Example: Vacuum Agent



- Performance?
  - 1 point for each square cleaned in time T?
  - #clean squares per time step - #moves per time step?
- Environment: Two rooms that can be dirty or clean
- Agent: A robotic vacuum cleaner
- Actions: left, right, suck, idle
- Percepts: location and contents
  - [A, dirty]

Reflex Vacuum Agent

- If status=Dirty then return Suck
  else if location=A then return Right
  else if location=B then right Left

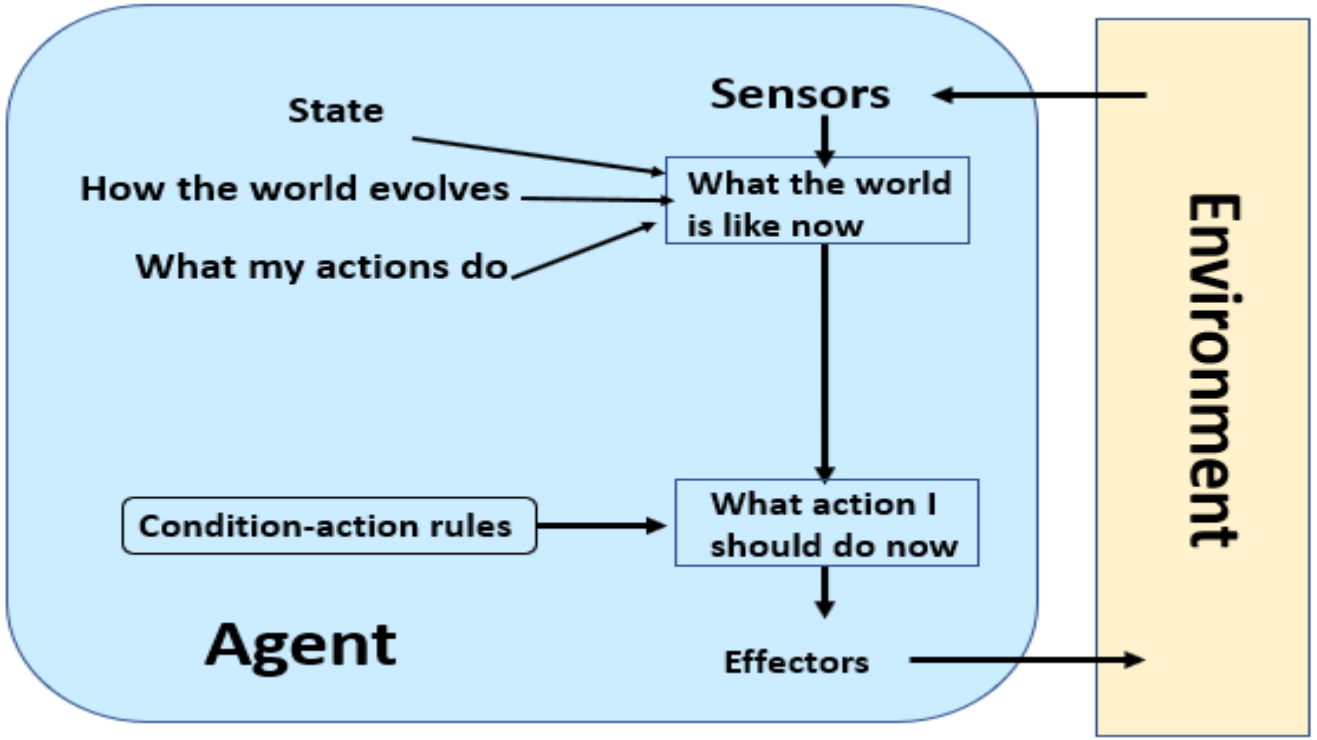


Reflex Agent With State

- These are the **agents with memory**. It stores the information about the **previous state, the current state and performs the action accordingly**.
- Just as while driving, if the driver wants to change the lane, he looks into the mirror to know the present position of vehicles behind him. While looking in front, he can only see the vehicles in front, and as he already has the information on the position of vehicles behind him (from the mirror a moment ago), he can safely change the lane. The previous and the current state get updated quickly for deciding the action.

Reflex Agent With State

- Store previously-observed information
- Can reason about unobserved aspects of current state



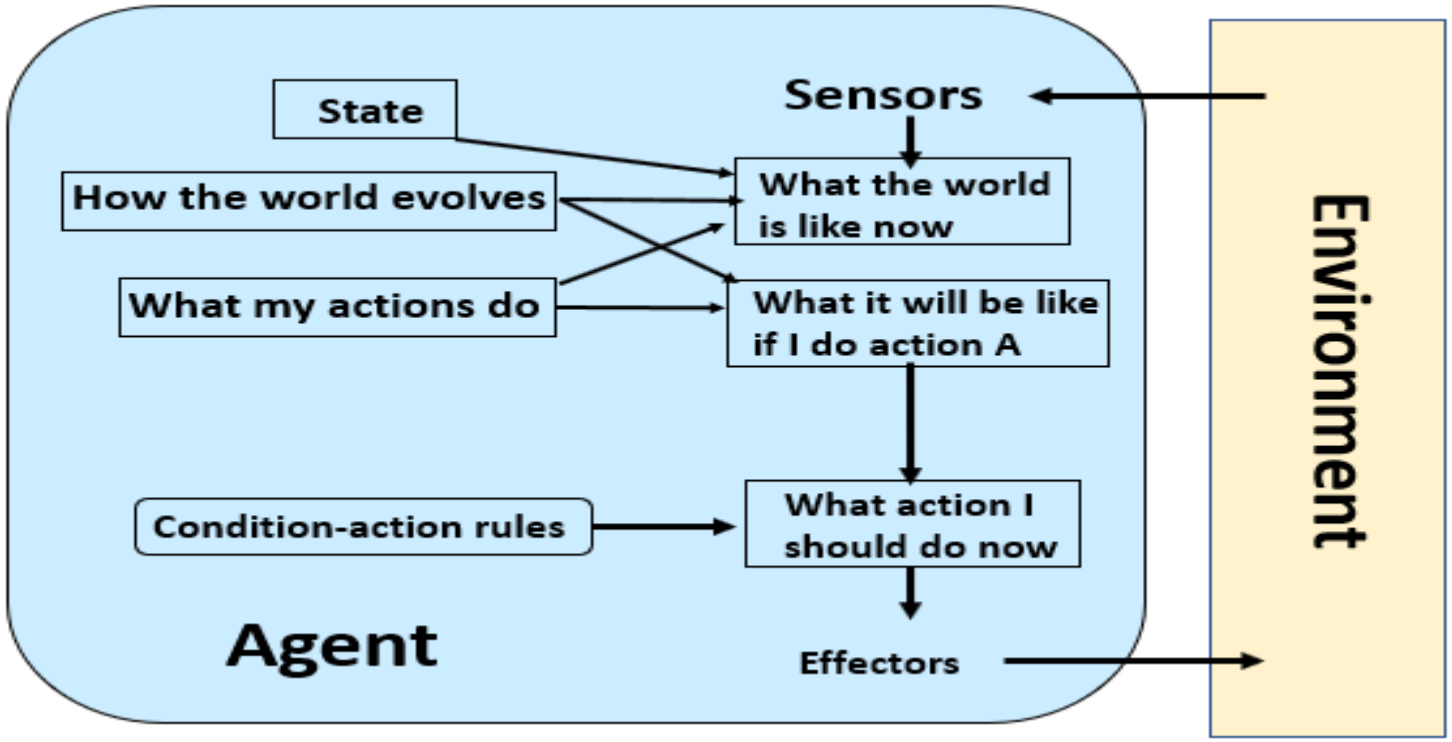

```
ReflexAgentWithState(percept)
  state  = UpdateDate(state,action,percept)
  rule   = RuleMatch(state, rules)
  action = RuleAction(rule)
  Return action
```

Goal-based Agents

- In some circumstances, just the information of the current state may not help in making the right decision**. If the goal is known, then the agent takes into account the goal information besides the current state information to make the right decision.**
- For, e.g., if the agent is a self-driving car and the goal is the destination, then the information of the route to the destination helps the car in deciding when to turn left or right.
- ‘Search’ and ‘planning’ are the two subfields of AI that help the agent achieve its goals. Though the goal-based agent may appear less efficient, yet it is flexible. Considering the same example mentioned above, if the destination changes then the agent will manipulate its actions accordingly. This will not be the case with the reflex agent as all the rules need to be rewritten with the change in goal.

Goal-Based Agents

- Goal reflects desires of agents
- May project actions to see if consistent with goals
- Takes time, world may change during reasoning

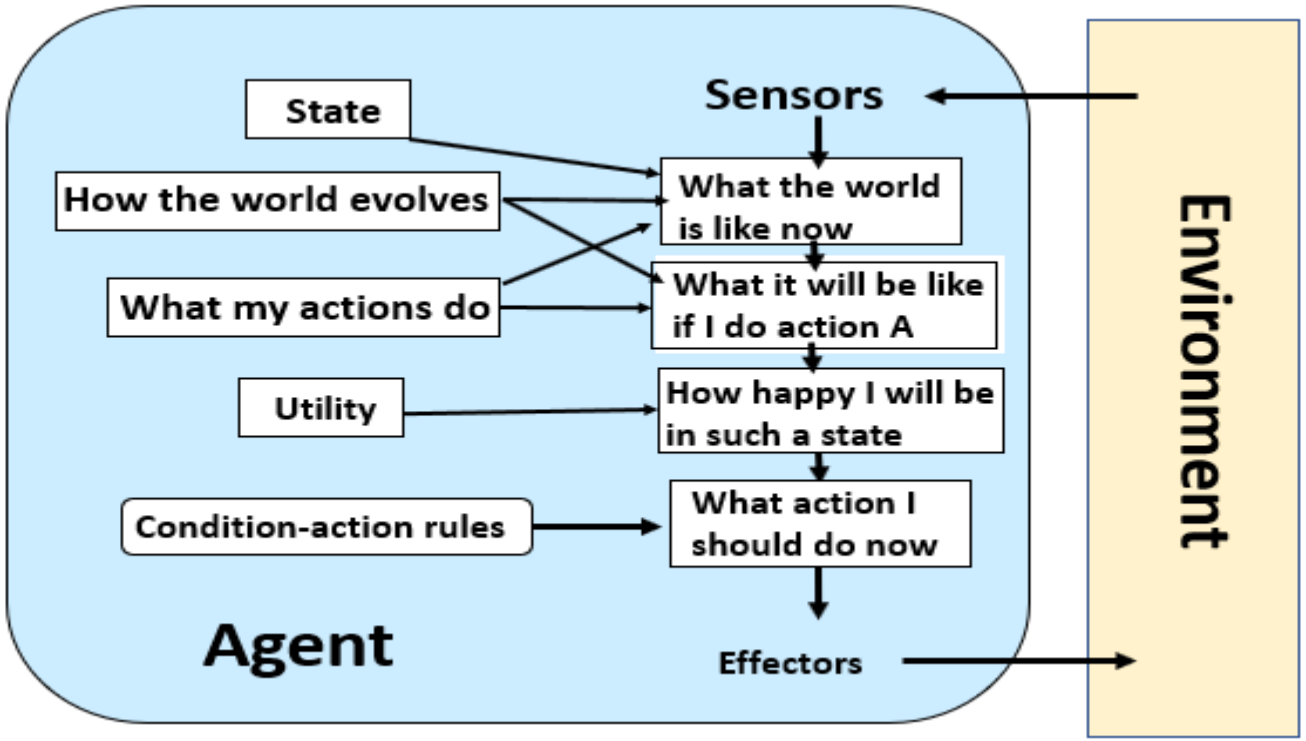


Utility Agents

- There can be many possible sequences to achieve the goal, but some will be better than others. Considering the same example mentioned above, **the destination is known**, but there are **multiple routes**. **Choosing an appropriate route** also matters to the overall success of the agent.
- There are many factors in deciding the route like the **shortest one**, the **comfortable one**, etc. The success depends on the utility of the agent-based on user preferences.
- The utility is a function that maps a state to a real number that describes the degree of happiness. The utility function specifies the appropriate trade-off in case the goals are conflicting.

Utility-Based Agents

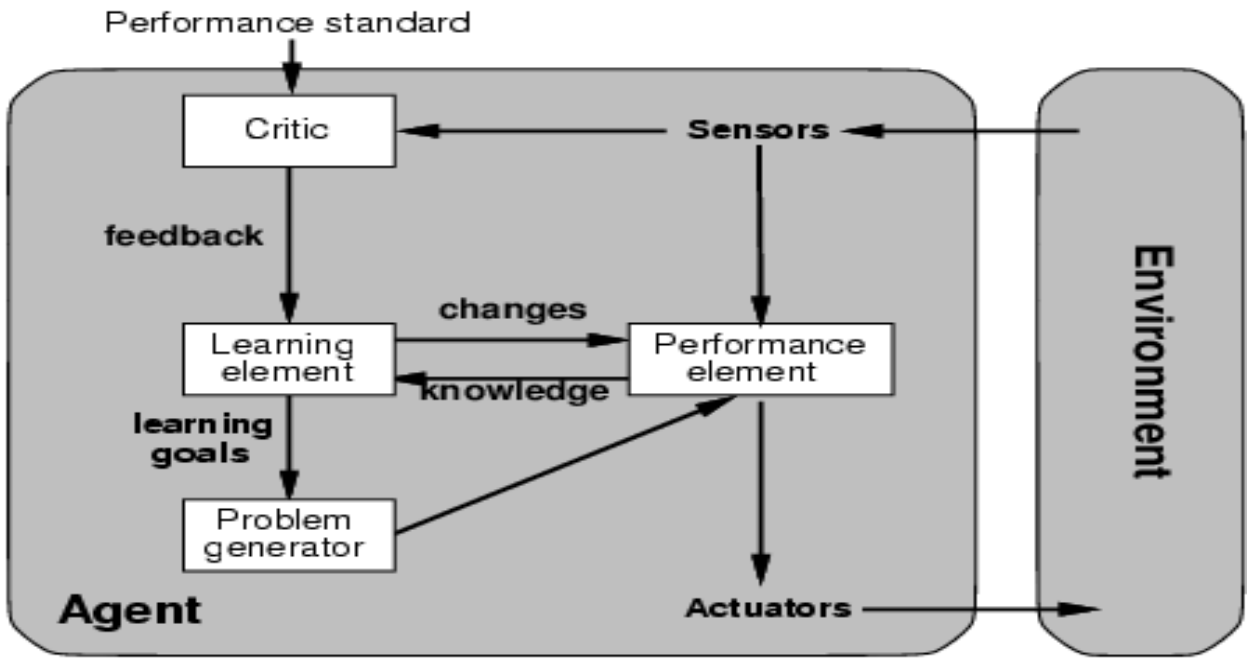
- Evaluation function to measure utility f(state) -> value
- Useful for evaluating competing goals



Learning Agents

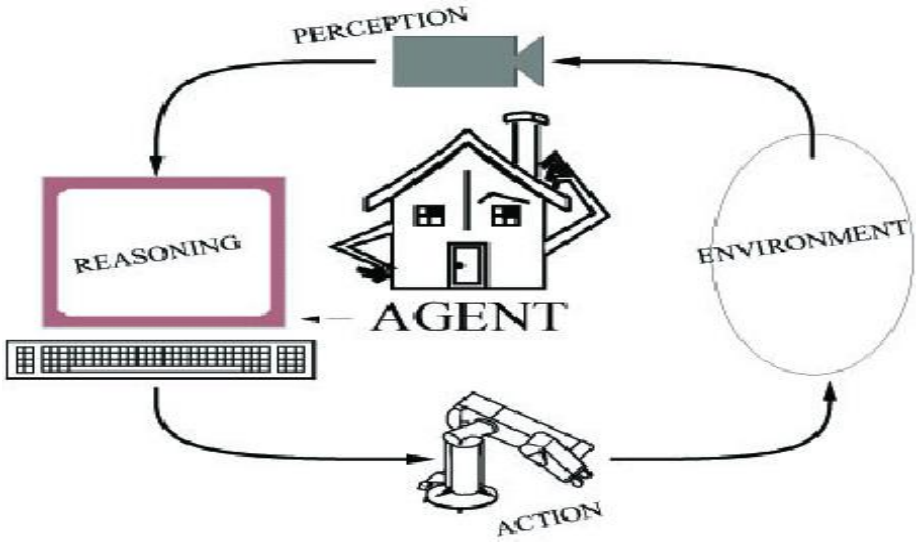
- This agent is capable of learning from the experience that is whatever the actions it has performed; it takes feedback and adapts accordingly.
- For a Learning agent to work the way, it has four components. First is the **learning element**, which learns from experience. Second is the **critic**, which is a feedback system about how well the agent is doing. The third is the **performance** element, which decides what external action should be taken. The last one is a **problem generator** which is a feedback agent that keeps history and makes new suggestions.

Learning Agents



Xavier mail delivery robot

- Performance: Completed tasks
- Environment: See for yourself
- Actuators: Wheeled robot actuation
- Sensors: Vision, sonar, dead reckoning
- Reasoning: Markov model induction, A* search, Bayes classification



Pathfinder Medical Diagnosis System

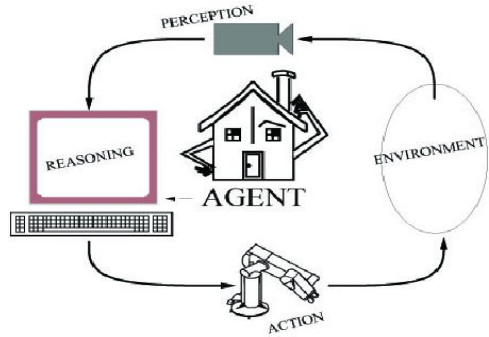
- Performance: Correct Hematopathology diagnosis
- Environment: Automate human diagnosis, partially observable, deterministic, episodic, static, continuous, single agent
- Actuators: Output diagnoses and further test suggestions
- Sensors: Input symptoms and test results
- Reasoning: Bayesian networks, Monte-Carlo simulations

Alvinn

- ALVINN (Autonomous Land Vehicle In a Neural Network) is a 3-layer back-propagation network designed for the task of road following. Currently ALVINN takes images from a camera and a laser range finder as input and produces as output the direction the vehicle should travel in order to follow the road.
- Performance: Stay in lane, on road, maintain speed
- Environment: Driving Hummer on and off road without manual control (Partially observable, stochastic, episodic, dynamic, continuous, single agent), Autonomous automobile
- Actuators: Speed, Steer
- Sensors: Stereo camera input
- Reasoning: Neural networks

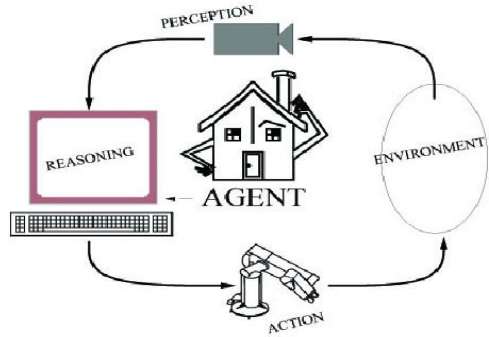
Robot Soccer

- Robot soccer competition
- Sensors: Camera image, messages from other players
- Reasoning: Planning, image processing
- Action: Robot 2D move or kick ball



Webcrawler Softbot

- Search web for items of interest
- Perception: Web pages
- Reasoning: Pattern matching
- Action: Select and traverse hyperlinks



Summary

- A goal of AI is to build intelligent agents that act so as to optimize performance.
- An agent perceives and acts in an environment, has an architecture, and is implemented by an agent program.
- An ideal agent always chooses the action which maximizes its expected performance, given its percept sequence so far.
- An autonomous agent uses its own experience rather than built-in knowledge of the environment by the designer.
- An agent program maps from percept to action and updates its internal state.
  - Reflex agents respond immediately to percepts.
  - Goal-based agents act in order to achieve their goal(s).
  - Utility-based agents maximize their own utility function.
- Representing knowledge is important for successful agent design.

Agent Environment

- Agent and Environment are two pillars in Artificial Intelligence, our aim is to build intellectual agents and work in an environment. If you consider broadly agent is the solution and environment is the problem.
- In simple terms, even starter or researcher can understand that and is defined **Agent as game and Environment as ground.**
- An environment can be described as a situation in which an agent is present.
- The environment is where agent lives, operate and provide the agent with something to sense and act upon it.
- Environment is the place where the agent is going to work. In general, Environment gives possible rewards, states, actions to the agents.

Agent Environment

- Before we start let us define few terms

➢ **Perception**: what agent see the environment.

➢ **Perception history**: It is the history of perception which comes in a specific period.

➢ **Actuators / Effectors** : A mechanism that puts something into action / Agents organs (hands and legs) that becomes active.

Examples

- Examples of Agent : It can be Program, Chatbot, Robot, Machine, Car, Players etc.
- Examples of Environment: It depends on the application for example- chess, maze, outer space etc.

Examples

- Examples of Agents and Environments are many due to their contexts, applications and necessity. Possible and well known agents and environments listed in the below diagram.
- **Agents ---------- Environments**

➢ Robot ----------→Room

➢ Chatbot----------> Chatting

➢ Vehicle----------→ Road

➢ Program---------→ Data and Rules

➢ Machine ----------→ Working Field

Types of Environment

- As per Russell and Norvig, an environment can have various features from the point of view of an agent:

- Fully observable vs Partially Observable
- Single-agent vs Multi-agent
- Deterministic vs Stochastic
- Episodic vs sequential
- Static vs Dynamic
- Discrete vs Continuous
- Known vs Unknown
- Accessible vs Inaccessible

Types of Environment

- **Fully observable Vs Partially Observable:**

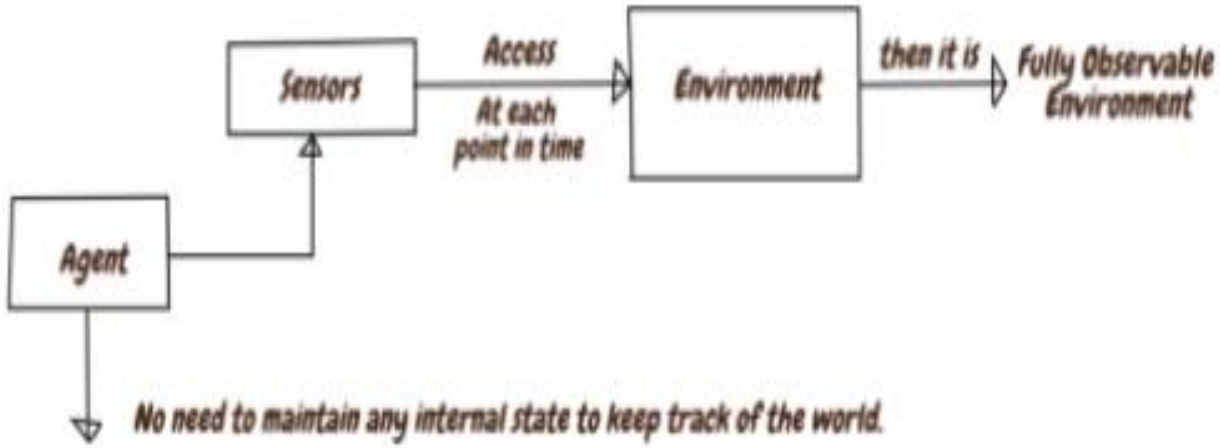
➢ If an agent sensor can sense or access the complete state of an environment at each point of time then it is **a fully observable** environment, else it is **partially observable**.

➢ An Environment is Partially Observable due to noise and inaccurate sensors or because parts of the state are simply missing from the sensor data.

➢ **Unobservable**: If the agent has no sensors at all then the Environment is Unobservable

Types of Environment

Fully Observable



Partially Observable

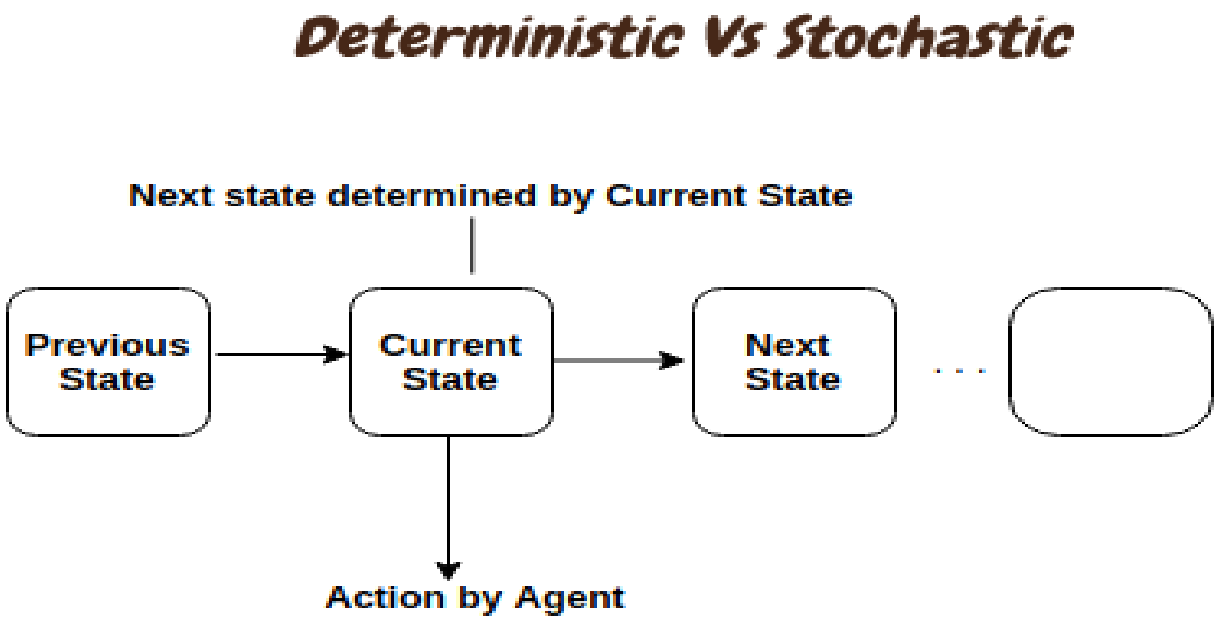
Types of Environment

- **Single Vs Multi Agent:** Only one agent participates in the environment is Single Agent. More than one agent interact the with the environment is Multi Agent.

- If only one agent is involved in an environment, and operating by itself then such an environment is called single agent environment.
- However, if multiple agents are operating in an environment, then such an environment is called a multi-agent environment.
- The agent design problems in the multi-agent environment are different from single agent environment.

Types of Environment

- **Deterministic Vs Stochastic :** If the next state of the environment is completely determined by the current state & the action executed by the agent, then we say the environment is deterministic, otherwise it is stochastic.



Types of Environment

- **Deterministic Vs Stochastic:**

- Deterministic AI environments are those on which the outcome can be determined base on a specific state. In other words, deterministic environments ignore uncertainty. Most real world AI environments are not deterministic. Instead, they can be classified as stochastic. Tic Tac Toe game, chess is an example of Deterministic environment.
- A stochastic environment is random in nature and cannot be determined completely by an agent. For e.g., if agent kicks the ball in a particular direction, then the ball may or may not be stopped by other players in soccer game, self driving car, the growth of a bacterial population, an electric current fluctuating due to thermal noise or the movement of a gas molecule.
- In a deterministic, fully observable environment, agent does not need to worry about uncertainty.

Types of Environment

- **Uncertainty**: An environment is Uncertain if it is not fully observable or Not Deterministic.
- In a multi-agent environment, Uncertainty arises purely from the actions of other agents. In deterministic, actions of other agents unable to predict by any other agent (each agent).

Types of Environment

- **Episodic vs Sequential:**

➢ In an episodic environment, there is a series of one-shot actions, and only the current percept is required for the action. For e.g., An AI that looks at radiology images to determine if there is a sickness.

➢ However, in Sequential environment, an agent requires memory of past actions to determine the next best actions.

➢ In an episodic environment, each agent's performance is the result of a series of independent tasks performed. There is no link between the agent's performance and other different scenarios. In other words, the agent decides which action is best to take, it will only consider the task at hand and doesn't have to consider the effect it may have on future tasks.

- Examples:

➢ Episodic environment: mail sorting system

➢ Non-episodic environment: chess game

Types of Environment

- **Static Vs Dynamic:**

- Static AI environments rely on data-knowledge sources that **don't change frequently over time**. Speech analysis is a problem that operates on static AI environments.
- If the environment can change itself while an agent is deliberating then such environment is called a dynamic environment else it is called a static environment.
- Static environments are easy to deal because an agent does not need to continue looking at the world while deciding for an action. However for dynamic environment, agents need to keep looking at the world at each action.
- Taxi driving is an example of a dynamic environment whereas Crossword puzzles are an example of a static environment.

Types of Environment

- **Discrete vs Continuous:**

- If in an environment there are a finite number of percepts and actions that can be performed within it, then such an environment is called a discrete environment else it is called continuous environment.
- A chess game comes under discrete environment as there is a finite number of moves that can be performed.
- Continuous AI environments rely on unknown and rapidly changing data sources. Vision systems in drones or self-driving cars operate on continuous AI environments.

Types of Environment

- **Known vs Unknown**
- Known and unknown are not actually a feature of an environment, but it is an agent's state of knowledge to perform an action.
- In a known environment, the results for all actions are known to the agent. While in unknown environment, agent needs to learn how it works in order to perform an action.
- It is quite possible that a known environment to be partially observable and an Unknown environment to be fully observable.
- **Accessible vs Inaccessible**
- If an agent can obtain complete and accurate information about the state's environment, then such an environment is called an Accessible environment else it is called inaccessible.
- An empty room whose state can be defined by its temperature is an example of an accessible environment.
- Information about an event on earth is an example of Inaccessible environment.

Introduction to State Space Search

State space search

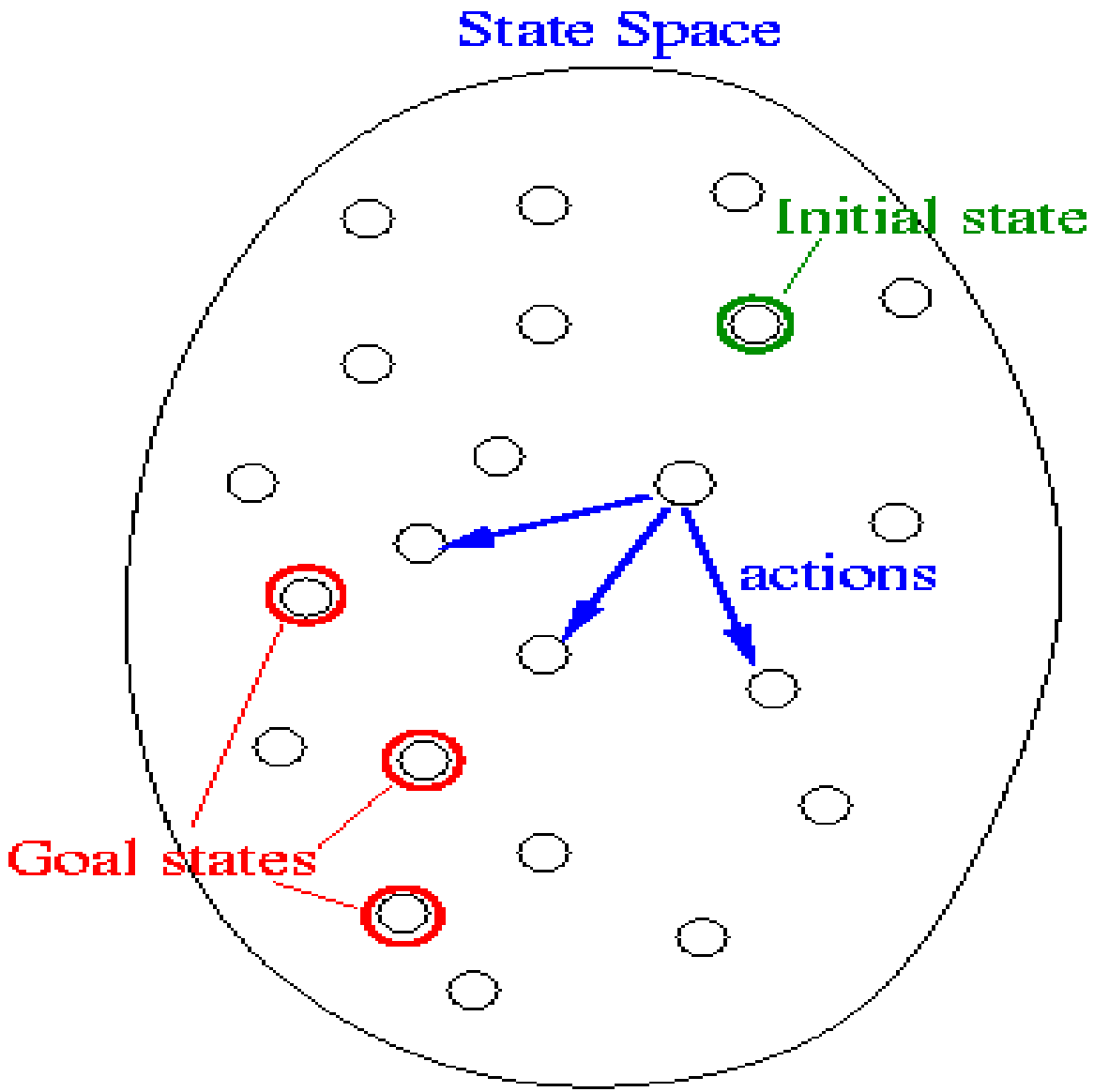
- Formulate a problem as a state space search by showing the legal problem states, the legal operators, and the initial and goal states
- A state is defined by the specification of the values of all attributes of interest in the world
- An operator changes one state into the other;
- The initial state is where you start
- The goal state is the partial description of the solution

Goal Directed Agent

- A goal directed agent needs to achieve certain goals.
- Such an agent selects its actions based on the goal it has.
- Few examples of goal directed agents
  - 15-puzzle:
  - to navigate a maze and reach the HOME position
- The agent must choose a sequence of actions to achieve the desired goal

Search Problem

- A search problem consists of the following:
  - S: the full set of states
  - s_0 : the initial state
  - A: S→Sis a set of operators
  - G is the set of final states. Note that G ⊆S
- Figure Shown in next slide


State Space
Initial state
actions
Goal states

Search Problem

- The search problem is to find a sequence of actions which transforms the agent from the initial state to a goal state $g \in G$.
- A search problem is represented by a 4-tuple $\{S, s_0, A, G\}$.
- This sequence of actions is called a solution plan (path from the initial state to a goal state)
- The cost of a path is a positive number.

Representation of search problems

- A search problem is represented using a directed graph.
  - The states are represented as nodes.
  - The allowed actions are represented as arcs.

Searching process

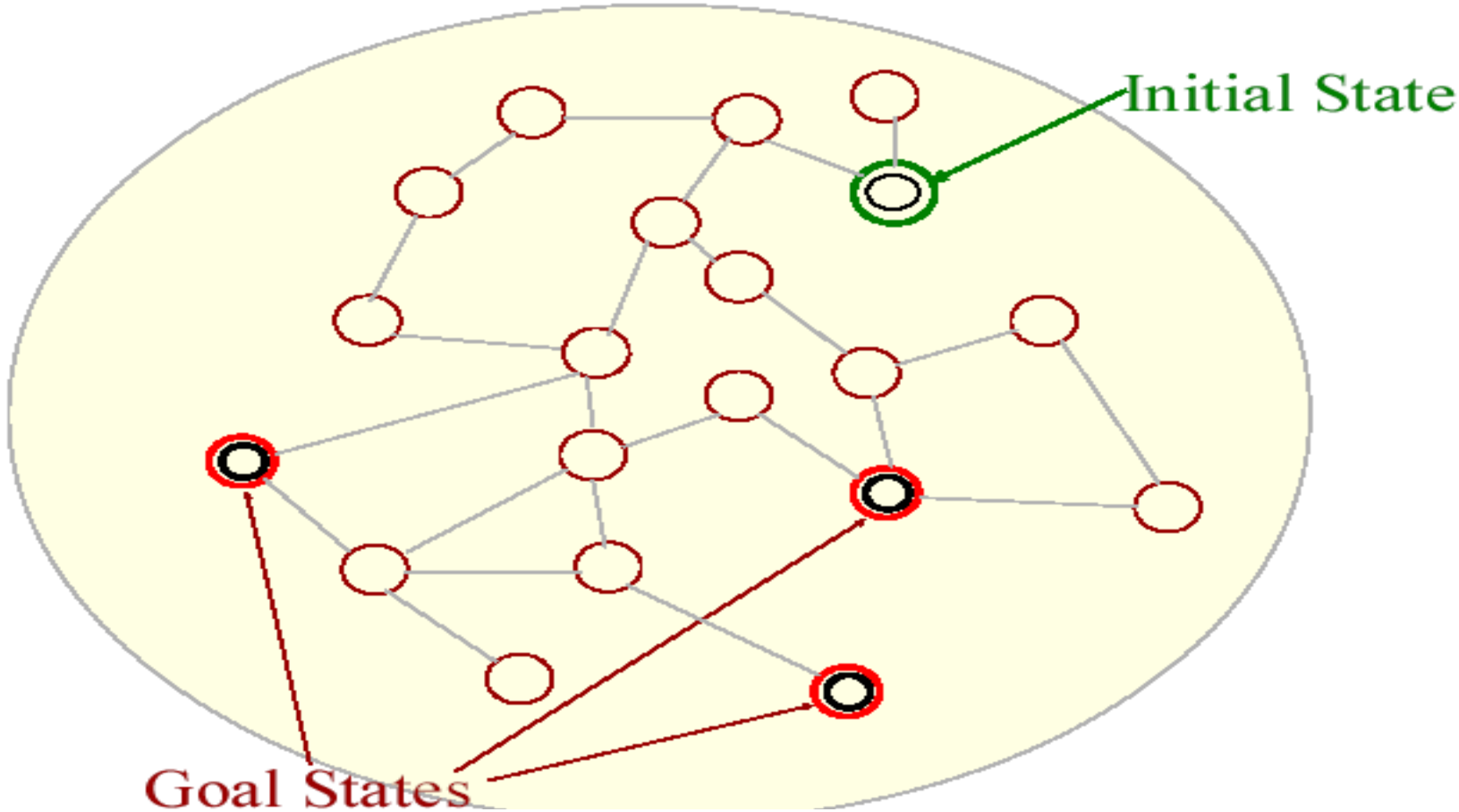
- The generic searching process

Do until a solution is found or the state space is exhausted.

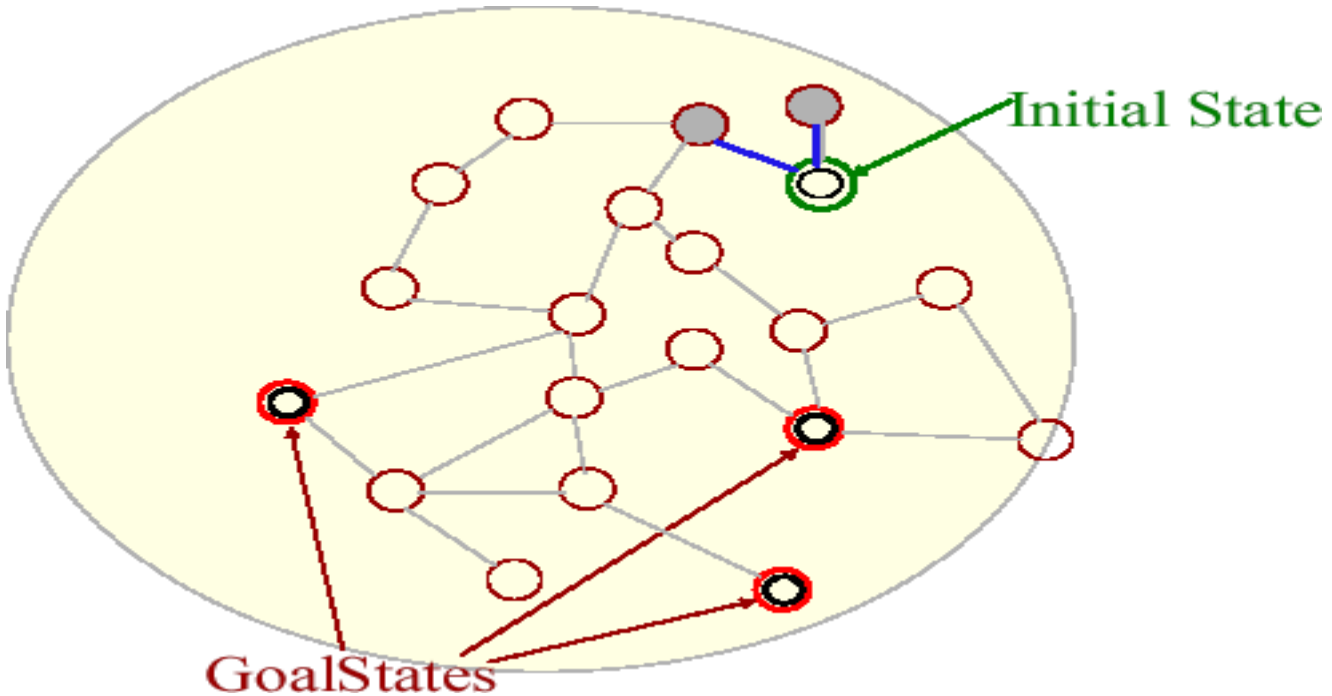
1. Check the current state
2. Execute allowable actions to find the successor states.
3. Pick one of the new states.
4. Check if the new state is a solution state

   If it is not, the new state becomes the current state and the process is repeated

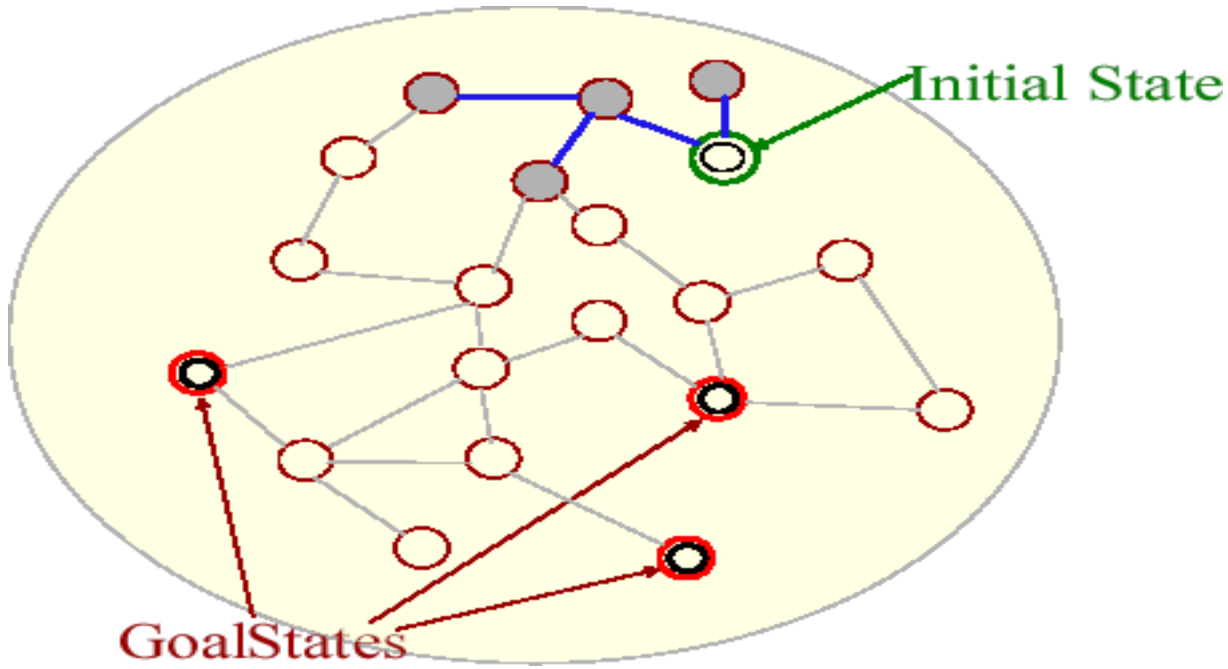
Illustration of a search process



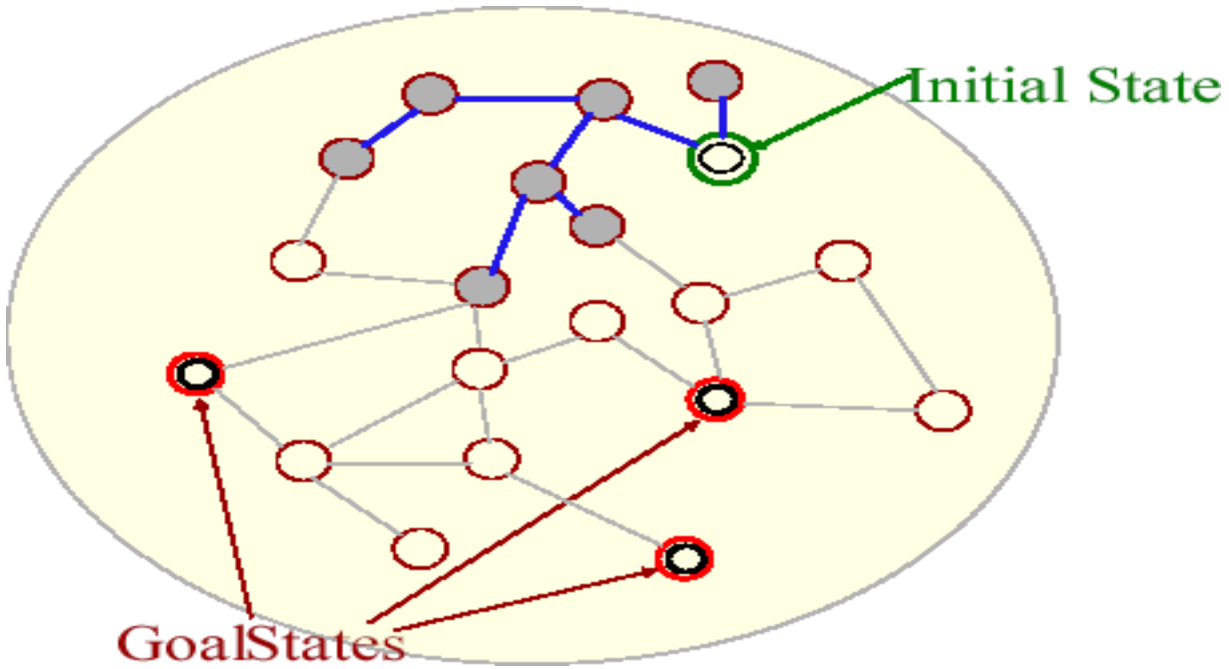
s_0 is the initial state.
The successor states are the adjacent states in the graph.
There are three goal states.



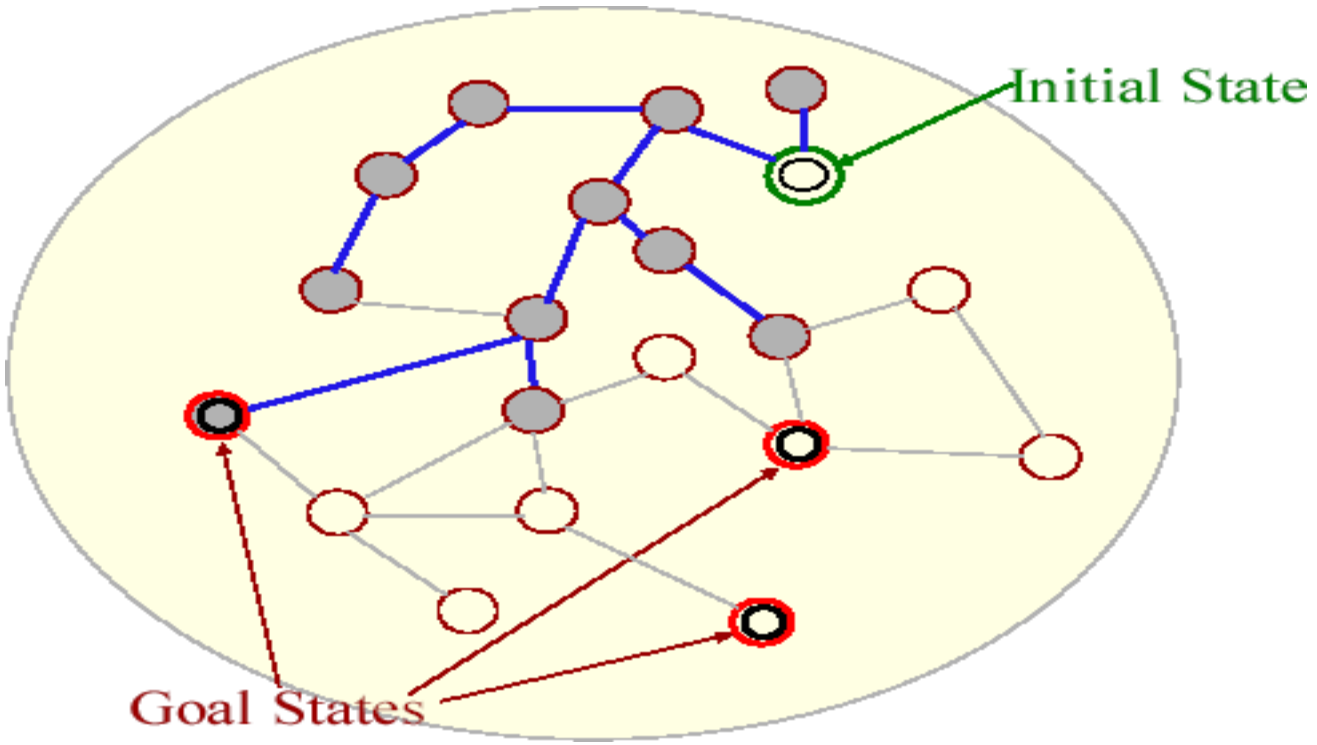
The two successor states of the initial state are generated.



The successors of these states are picked and their successors are generated.



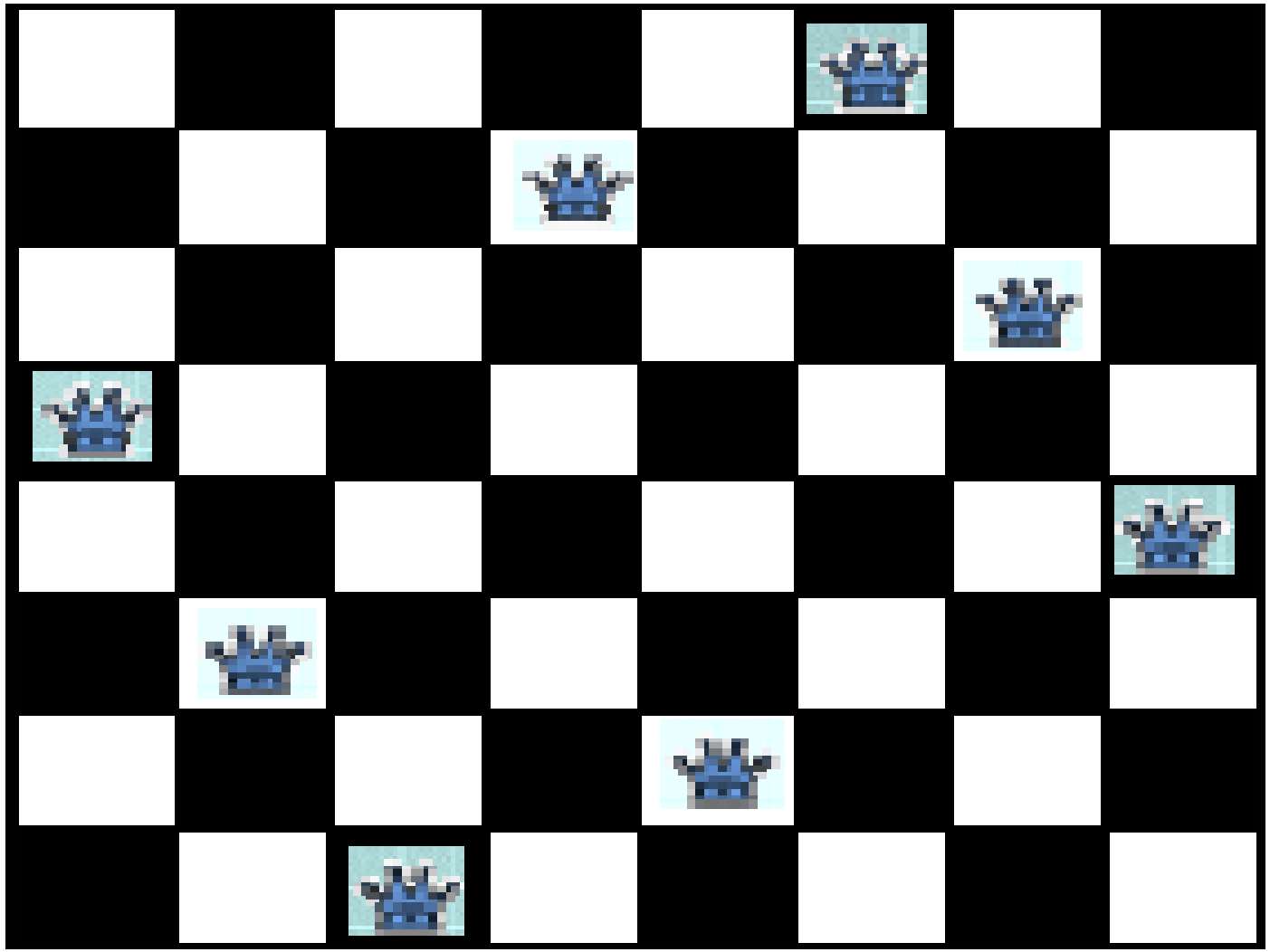
Successors of all these states are generated.



A goal state has been found.

8 queens problem

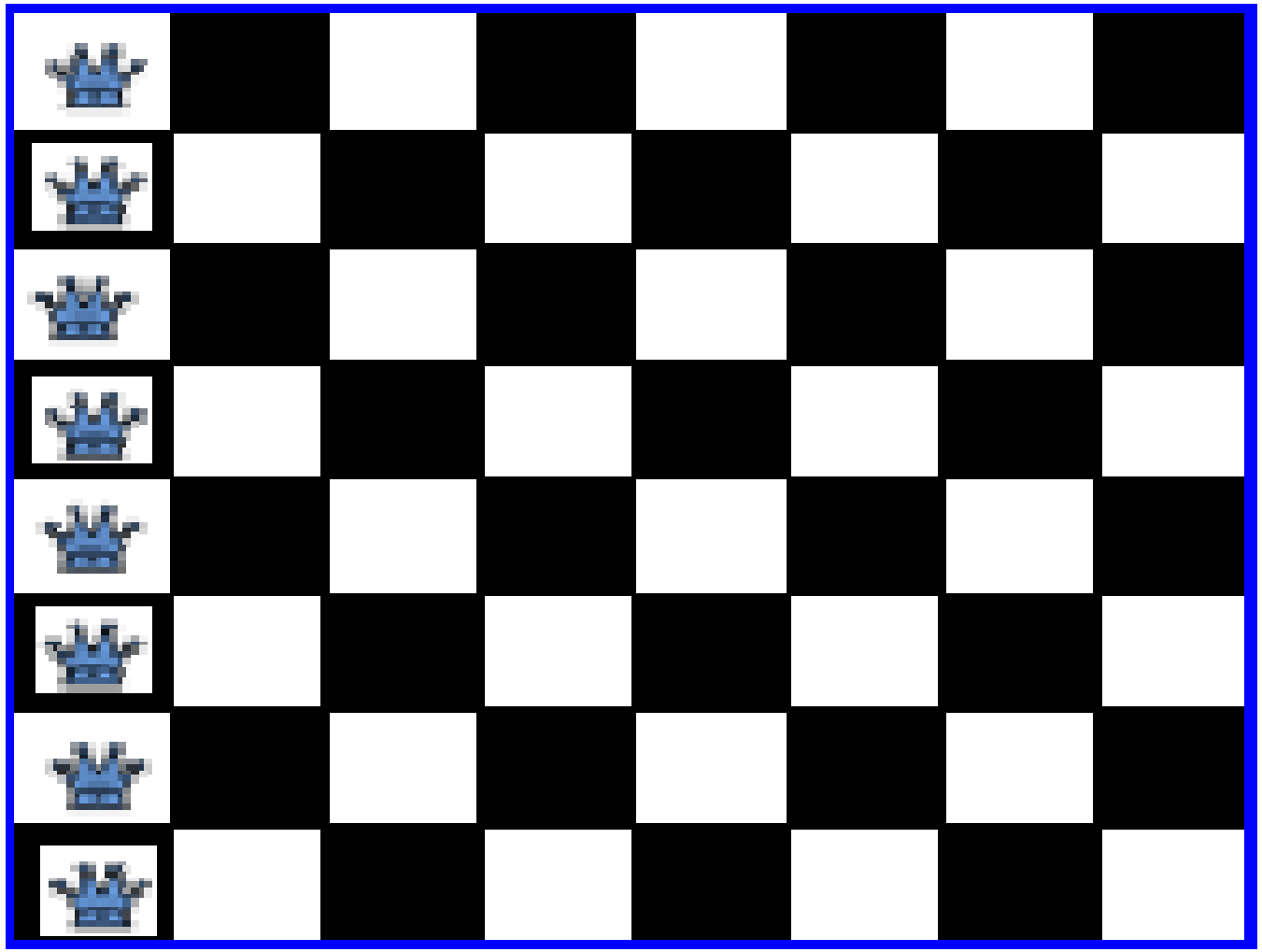
- The problem is to place 8 queens on a chessboard so that no two queens are in the same row, column or diagonal



- Is this state a solution? Yes

N queens problem formulation

- States: Any arrangement of 8 queens on the board
- Initial state: All queens are at column 1
- Successor function: Change the position of any one queen
- Goal test: 8 queens on the board, none are attacked



- If we consider moving the queen at column 1, it may move to any of the seven remaining columns.

Thank you.

Unit-2

Knowledge Representation and Reasoning

Outline

- What is knowledge representation
- Different types of knowledge
- Cycle of knowledge representation
- Relation between knowledge and intelligence
- Techniques of knowledge representation
- Representation Requirements
- Approaches with example

Knowledge Representation

- Knowledge Representation in Artificial Intelligence refers to that concept where ways are identified to provide machines with the knowledge that humans possess so that AI systems can become better. As it is a universal fact that more a person knows a subject matter, the chances of taking a correct action or decision will be higher.
- AI developers represent the knowledge of the human world in a way that machines can understand and can make the AI systems smarter to solve complex real-world problems. The problem is that we humans process information in a highly complex manner.
- One of the primary purposes of Knowledge Representation includes modelling intelligent behaviour for an agent.

Knowledge Representation

- Knowledge Representation in AI describes the representation of knowledge. Basically, it is a study of how the beliefs, intentions, and judgments of an intelligent agent can be expressed suitably for automated reasoning.
- Knowledge Representation and Reasoning (KR, KRR) represents information from the real world for a computer to understand and then utilize this knowledge to solve complex real-life problems like communicating with human beings in natural language.
- Knowledge representation in AI is not just about storing data in a database, it allows a machine to learn from that knowledge and behave intelligently like a human being.

What to Represent

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

➢ **Object:** All the facts about objects in our world domain. For example, cars have wheels, the piano has keys, Guitars contains strings, etc.

➢ **Events:** Events are the actions which occur in our world. Our perception of the world is based on what we know regarding the various events that have taken place in our world. This knowledge is regarding all those events. The wars, achievements, advancement of societies, etc., are an example of this knowledge.

➢ **Performance:** It describe behavior which involves knowledge about how to do things. It deals with how humans and other beings and things perform certain actions in different situations. Thus, it helps in understanding the behavior side of the knowledge.

What to Represent

- **Meta-knowledge:** It is knowledge about what we know. knowledge can be divided into 3 categories: What we know, What we know that we don’t know, and knowledge that we even are unaware of and Meta knowledge deals with the first concept. Thus, meta-knowledge is the knowledge of what we know.
- **Facts:** Facts are the truths about the real world and what we represent.

Knowledge-Based Agent in AI

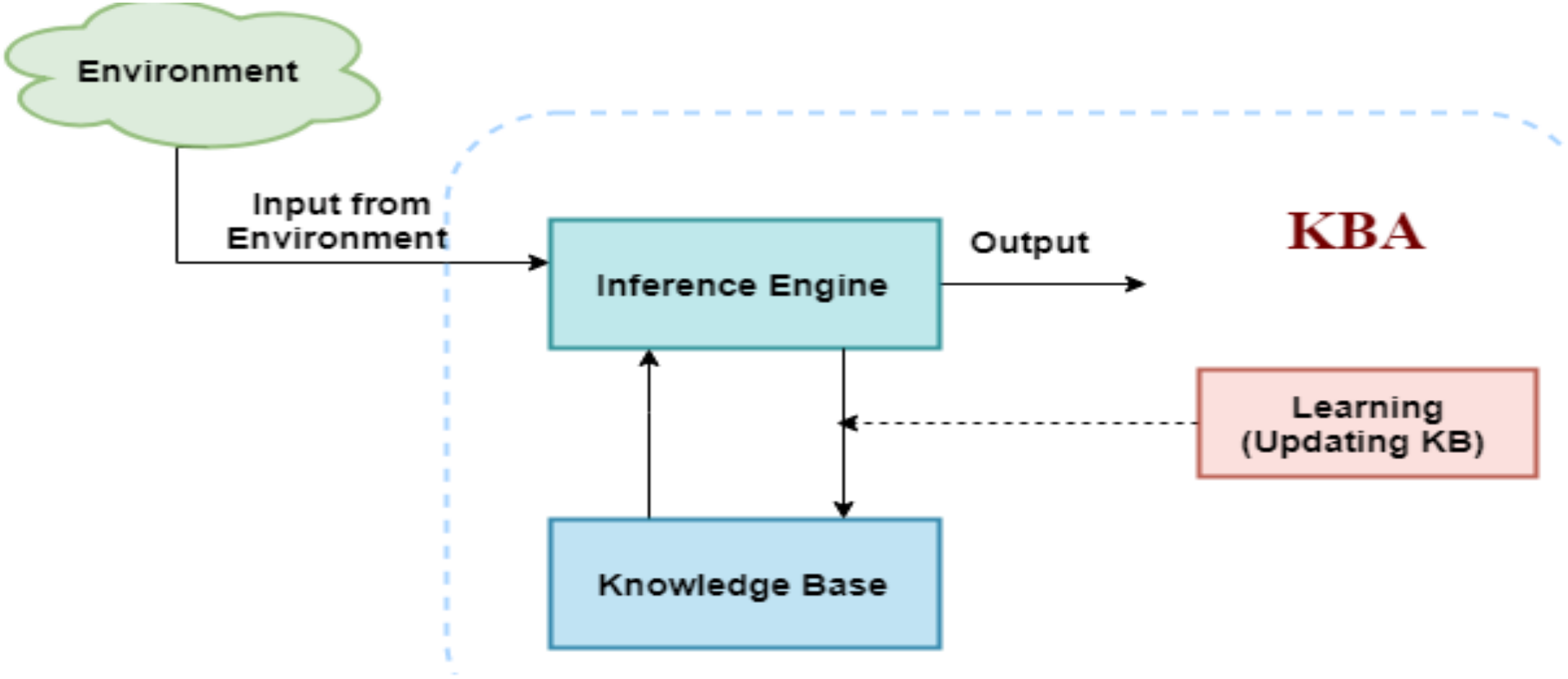
- An intelligent agent needs **knowledge** about the real world for taking decisions and **reasoning** to act efficiently.
- Knowledge-based agents are those agents who have the capability of **maintaining an internal state of knowledge, reason over that knowledge, update their knowledge after observations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

➢ Inference system.

The architecture of knowledge-based agent

- Knowledge-base is a central component of a knowledge-based agent, it is also known as KB. It is a collection of sentences. These sentences are expressed in a language which is called a knowledge representation language. The Knowledge-base of KBA stores fact about the world.



The architecture of knowledge-based agent

- **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

Knowledge-based agent

- Approaches to designing a knowledge-based agent:

- **Declarative approach:** Declarative knowledge refers to facts or information stored in the memory, that is considered static in nature. Declarative knowledge, also referred to as conceptual, propositional or descriptive knowledge, describes things, events, or processes; their attributes; and their relation to each other.
- **Procedural approach:** Procedural Knowledge refers to the knowledge of how to perform a specific skill or task, and is considered knowledge related to methods, procedures, or operation of equipment.

Declarative and Procedural Knowledge

- Declarative knowledge involves knowing THAT something is the case - that J is the tenth letter of the alphabet, that Paris is the capital of France. Declarative knowledge is conscious; it can often be verbalized.
- Procedural knowledge involves knowing HOW to do something - ride a bike, for example. We may not be able to explain how we do it. Procedural knowledge involves implicit learning, which a learner may not be aware of, and may involve being able to use a particular form to understand or produce language without necessarily being able to explain it.

Declarative and Procedural Knowledge

- Real world Example: I need a cup of tea.

➢ Declarative:

1. Get me a cup of tea.

➢ Procedural:

1. Go to kitchen
2. Get sugar, milk and tea.
3. Mix them and heat over the fire till it boils
4. Put that in a cup and bring it to me

- In a declarative language, we just set the command or order and let it be on system how to complete that order. We just need our result without digging into how it should be done.

Declarative and Procedural Knowledge

- In a procedural language, we define the whole process and provide the steps how to do it. We just provide orders and define how the process will be served.
- The main difference between two approaches are, in declarative approach, we tell the computer what problem we want solved and in procedural approach, we tell the computer how to solve the problem.

Declarative and Procedural Knowledge

- Different types of knowledge can be more or less effective, given the scenario in which they're used. For example, you can score 100% in your driving theory test, yet still not be able to actually drive a car. In that case, your declarative knowledge of driving is almost useless, as you can't actually put it into practice until you have an understanding of the procedural knowledge involved in driving the car itself.
- You might know what every road sign in the US means, what every button on your dashboard does, and what lies underneath the hood, but you don't know how to parallel park or shift from 1st to 2nd gear.

Different Types of Knowledge

Different Types of Knowledge

- **Declarative Knowledge** – It includes concepts, facts, and objects and expressed in a declarative sentence.
- **Procedural Knowledge** – This is responsible for knowing how to do something and includes rules, strategies, procedures, etc.
- **Meta Knowledge** – Meta Knowledge defines knowledge about other types of Knowledge.
- **Heuristic Knowledge** – This represents some expert knowledge in the field or subject.
- **Structural Knowledge** – It is a basic problem-solving knowledge that describes the relationship between concepts and objects

Different Types of Knowledge

- **Declarative Knowledge**

➢ It is the knowledge that represents the facts, objects, concepts that help us describe the world around us. Thus it deals with the description of something.

- **Procedural Knowledge**

➢ This type of knowledge is more complex than declarative knowledge as it refers to a more complex idea, i.e., how things behave and work. Thus this knowledge is used to accomplish any task using certain procedures, rules, and strategies, making the system using this knowledge work efficiently. Also, this type of knowledge highly depends on the task we are trying to accomplish.

Different Types of Knowledge

- **Meta Knowledge**

- The knowledge of pre-defined knowledge is known as meta knowledge. A study of planning, tagging and learning are some of the examples of meta knowledge. This model tends to change with time and utilize a different specification
- For example, bibliographic data are considered as a meta-knowledge.
- The main usage of meta-knowledge is to understand and improve the nature of user interface components and to maintain the knowledge bases that are used alongside inference engines as well.

Different Types of Knowledge

- **Heuristic Knowledge**

- The knowledge provided by experts of certain domains, subjects, disciplines, and fields is known as the Heuristic knowledge, which they have been obtained after years of experience. This type of knowledge helps in taking the best approach to particular problems and making decisions.
- Heuristic knowledge is seen as a helpmate to what you know. Some examples of heuristic knowledge are a hypothesis, common sense, rule of thumb, etc.
- A heuristic, is any approach to problem-solving that uses a practical method or various shortcuts in order to produce solutions that may not be optimal but are sufficient given a limited timeframe or deadline.

Different Types of Knowledge

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

➢ Declarative- explains facts

➢ Procedural- explain the behaviour

➢ Meta knowledge of other topics of knowledge

➢ Heuristic-knowledge of specific fields and domains

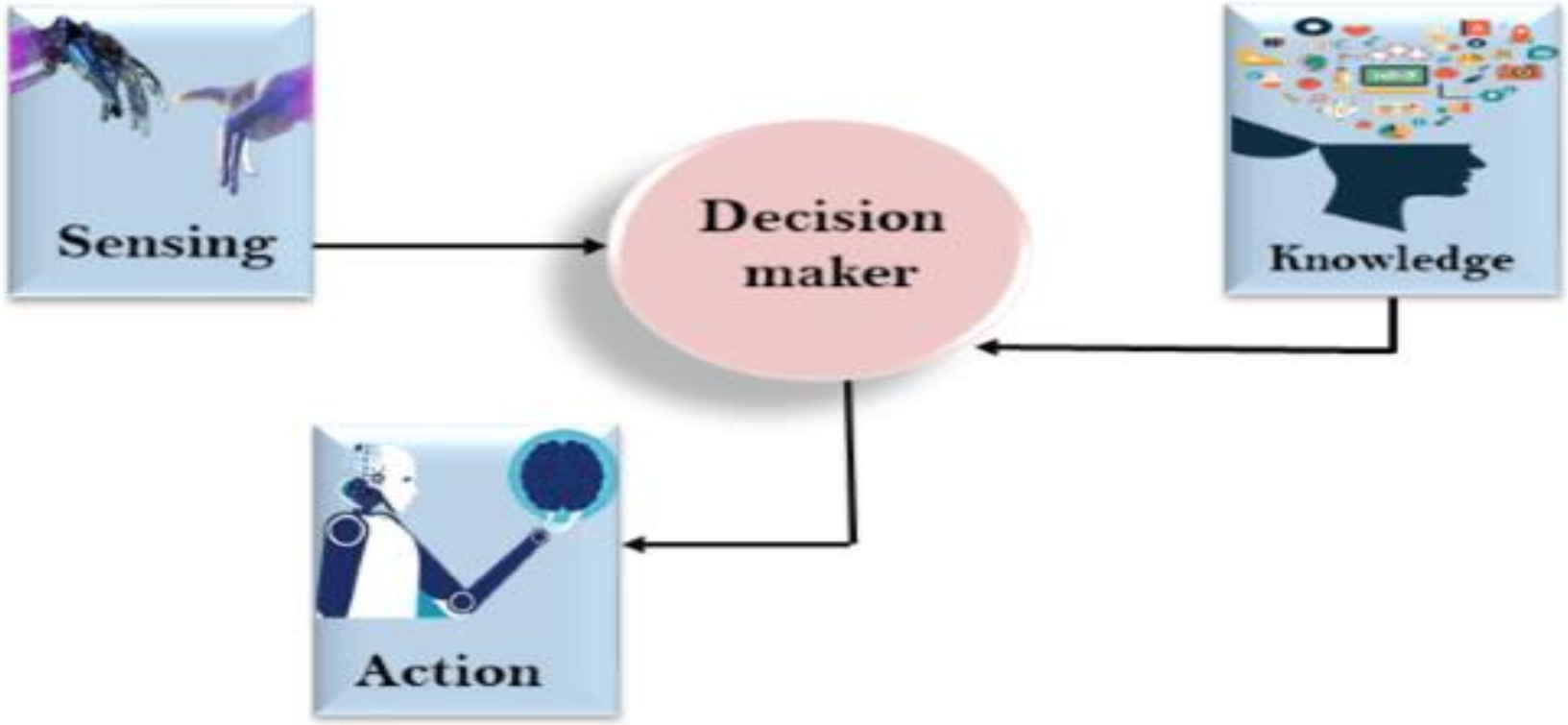
➢ Structural- knowledge for seeing the relations between different objects

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 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 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 behaviour of the agents.

Relation between Knowledge and Intelligence

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



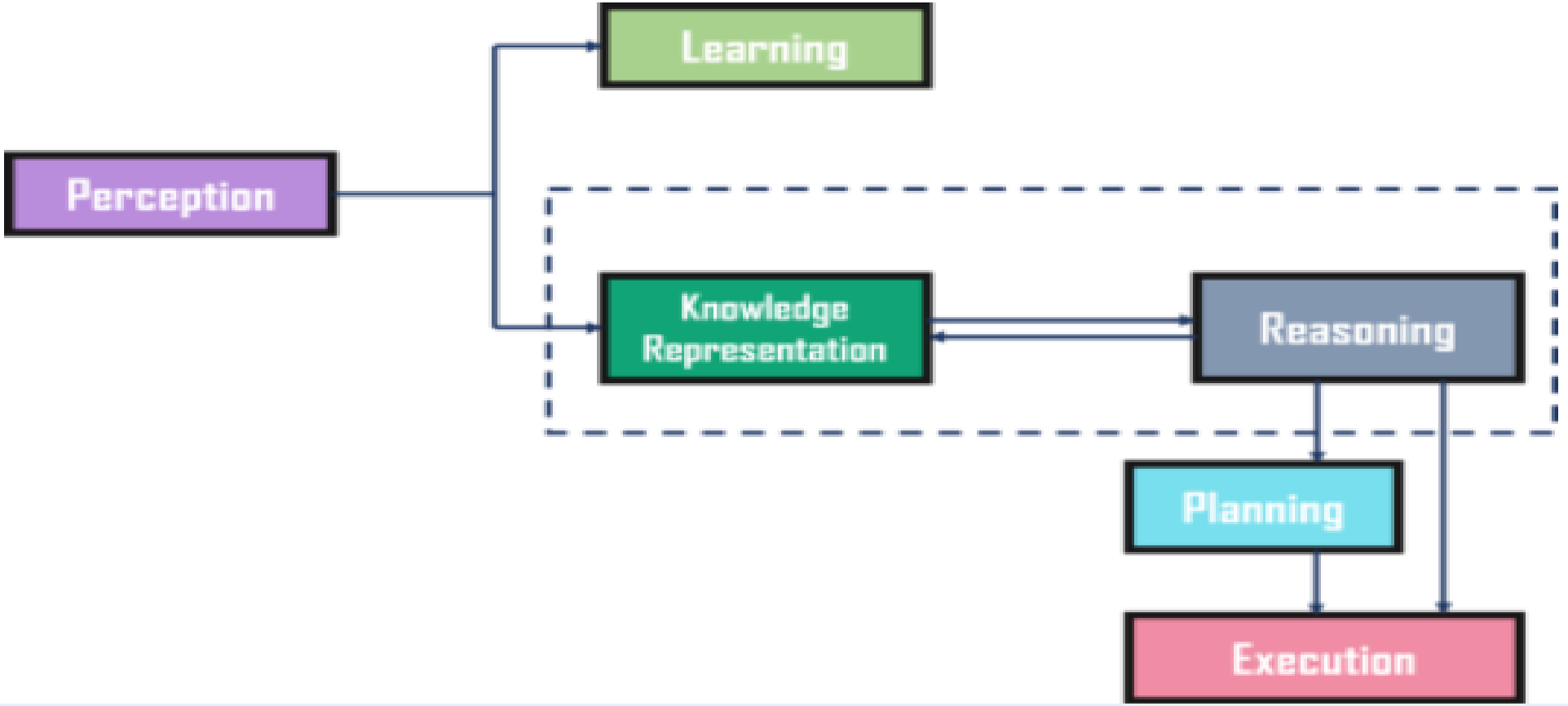
Cycle of Knowledge Representation

- Artificial Intelligent Systems usually consist of various components to display their intelligent behaviour. Some of these components include:

- **Perception**
- **Learning**
- **Knowledge Representation & Reasoning**
- **Planning**

Cycle of Knowledge Representation

- **Cycle of Knowledge Representation in AI**



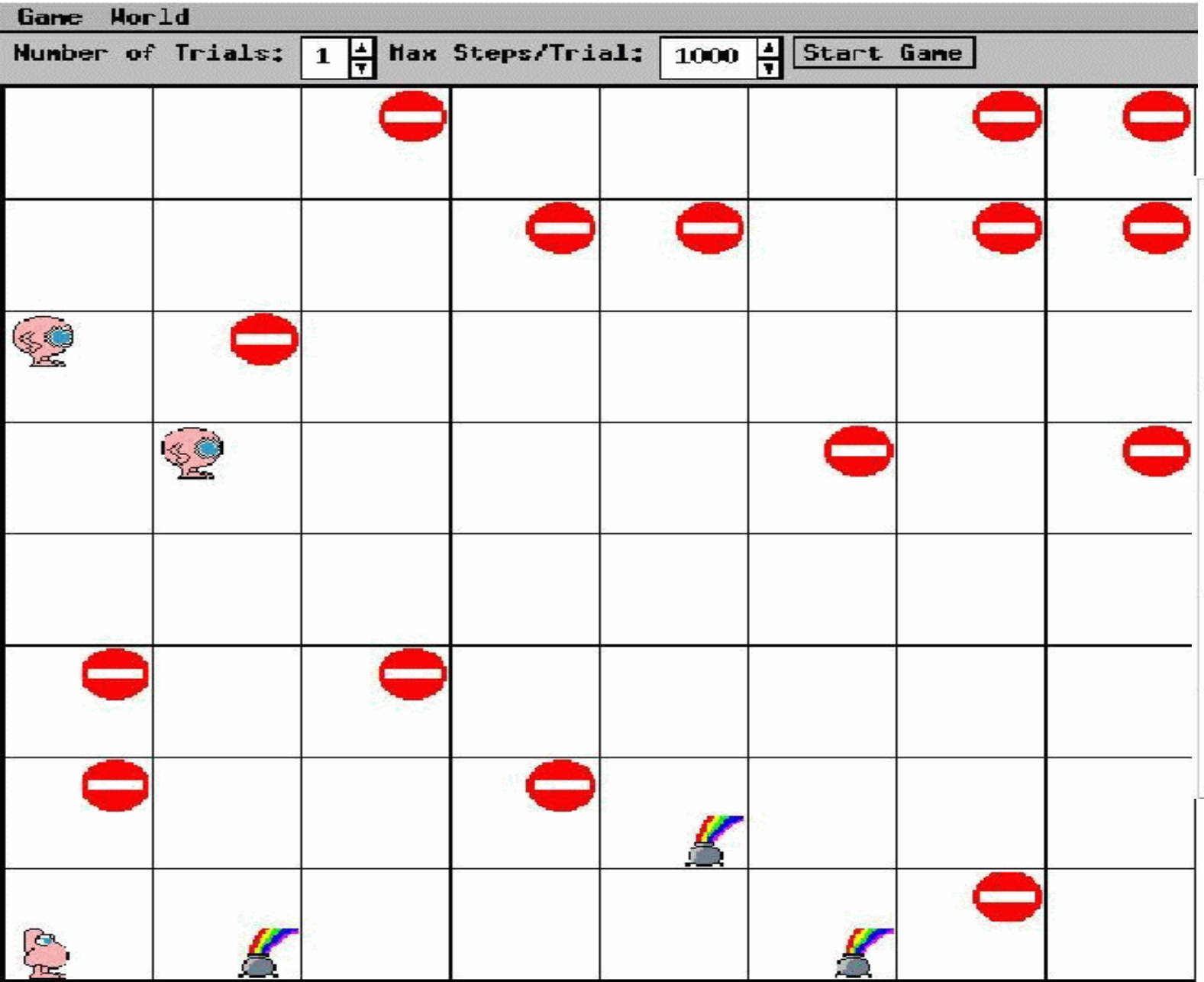
Cycle of Knowledge Representation

- The diagram shows the interaction of an AI system with the **real world** and the **components** involved in showing intelligence.
- The **Perception component** retrieves data or information from the environment. with the help of this component, we can retrieve data from the environment
- Then, there is the **Learning Component** that learns from the captured data by the perception component. The goal is to build computers that can be taught instead of programming them. Learning focuses on the process of self-improvement. In order to learn new things, the system requires knowledge acquisition, inference, acquisition of heuristics, faster searches, etc.

Cycle of Knowledge Representation

- The main component in the cycle is **Knowledge Representation and Reasoning** which shows the human-like intelligence in the machines. Knowledge representation is all about understanding intelligence and focus on what an agent needs to know in order to behave intelligently. Also, it defines how automated reasoning procedures can make this knowledge available as needed.
- The **Planning and Execution** components depend on the analysis of knowledge representation and reasoning. Here, planning includes giving an initial state, finding their preconditions and effects, and a sequence of actions to achieve a state in which a particular goal holds. Now once the planning is completed, the final stage is the execution of the entire process.

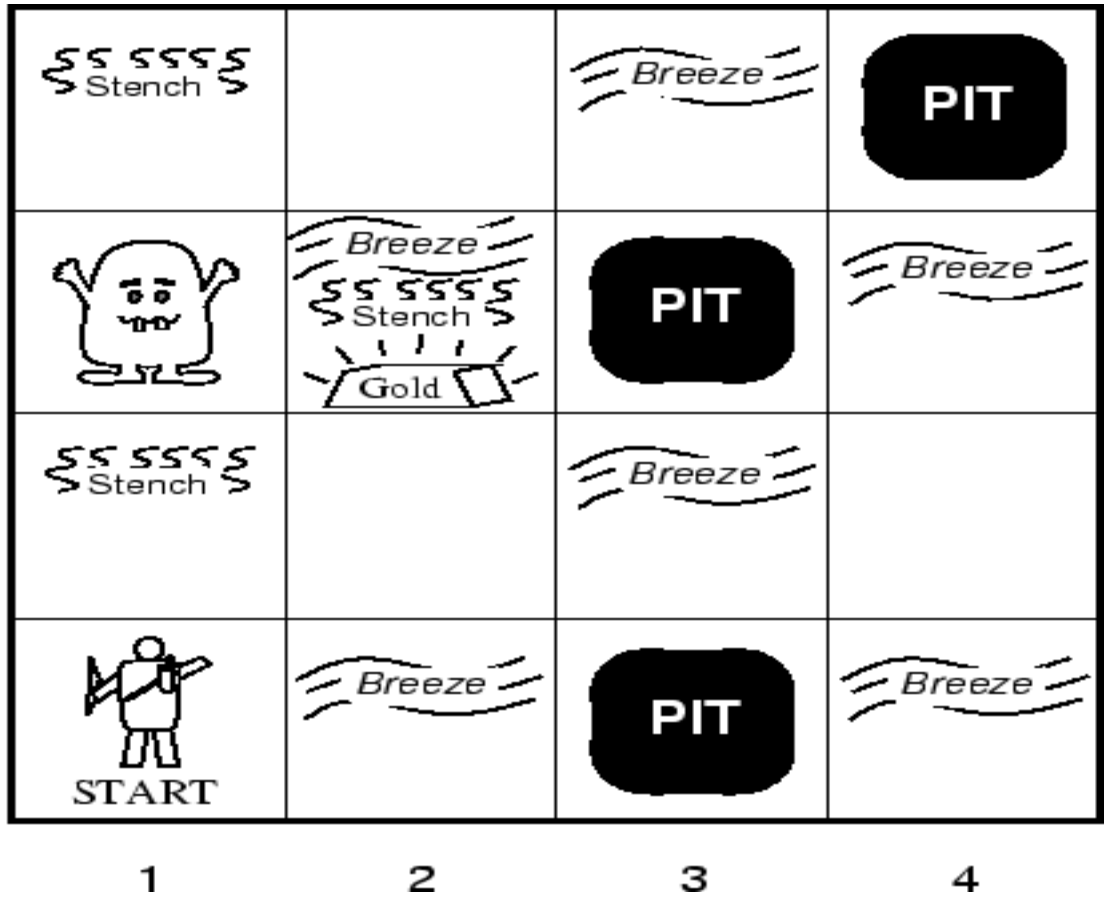
The Wumpus World Environment



Player (Facing Down)	Player (Facing Up)
Wumpus (Facing Down)	Wumpus (Facing Up)
Player (Facing Left)	Player (Facing Right)
Wumpus (Facing Left)	Wumpus (Facing Right)
Player (Dead)	Pot of Gold
Wumpus (Dead)	Bottomless Pit

WW Agent Description

- Performance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using arrow
- Environment
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in same square
  - Shooting kills wumpus if agent facing it
  - Shooting uses up only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops gold in same square
- Actuators
  - Left turn, right turn, forward, grab, release, shoot
- Sensors
  - Breeze, glitter, smell, bump, scream



WW Environment Properties

- Observable?
  - Partial
- Deterministic?
  - Yes
- Episodic?
  - Sequential
- Static?
  - Yes (for now), wumpus and pits do not move
- Discrete?
  - Yes
- Single agent?
  - Multi (wumpus, eventually other agents)

Sample Run

1,4	2,4	3,4	4,4
1,3	2,3	2,4	2,5
1,2 OK	2,2	3,2	4,2
1,1 A OK	2,1 OK	3,1	4,1

Percept: [None, None, None, None, None]

Deduce: Agent alive, so (1,1) OK
No breeze, so (1,2) and (2,1) OK

Sample Run

1,4	2,4	3,4	4,4
1,3	2,3	2,4	2,5
1,2 OK	2,2	3,2	4,2
1,1 A OK	2,1 OK	3,1	4,1

Percept: [None, None, None, None, None]

Deduce: Agent alive, so (1,1) OK
No breeze, so (1,2) and (2,1) OK

Action: Move East (turnright, goforward)

Sample Run

1,4	2,4	3,4	4,4
1,3	2,3	2,4	2,5
1,2 OK	2,2 P?	3,2	4,2
1,1 OK	2,1 OK	3,1 P?	4,1

Percept: [None, Breeze, None, None, None]

Deduce: Pit in (2,2) or (3,1)

Sample Run

1,4	2,4	3,4	4,4
1,3	2,3	2,4	2,5
1,2 OK	2,2 P?	3,2	4,2
1,1 OK	2,1 OK	3,1 P?	4,1

Percept: [None, Breeze, None, None, None]

Deduce: Pit in (2,2) or (3,1)

Action: Back to (1,1) then to (1,2)
(turnleft, turnleft, goforward,
turnright, goforward)

Sample Run

1,4	2,4	3,4	4,4
1,3 W	2,3	2,4	2,5
1,2 (A) OK	2,2 OK	3,2	4,2
1,1 OK	2,1 OK	3,1 P	4,1

Percept: [Stench, None, None, None, None]

Deduce: Wumpus in (1,3)
No pit in (2,2), pit in (3,1)

Sample Run

1,4	2,4	3,4	4,4
1,3 W	2,3	3,3	4,3
1,2 (A) OK	2,2 OK	3,2	4,2
1,1 OK	2,1 OK	3,1 P	4,1

Percept: [Stench, None, None, None, None]

Deduce: Wumpus in (1,3)
No pit in (2,2), pit in (3,1)

Action: Move to (2,2) (turnright, goforward)
Ignore percept for now
Move to (2,3) (turnleft, goforward)

Sample Run

1,4	2,4 P?	3,4	4,4
1,3 W	2,3 (A) OK G	3,3 P?	4,3
1,2 OK	2,2 OK	3,2	4,2
1,1 OK	2,1 OK	3,1 P	4,1

Percept: [Stench, Breeze, Glitter, None, None]

Deduce: Pit in (2,4) or (3,3)
Gold in (2,3)

Action: Move to (2,2) (turnright, goforward)
Ignore percept for now
Move to (2,3) (turnleft, goforward)

Sample Run

1,4	2,4 P?	3,4	4,4
1,3 W	2,3 (A) OK G	3,3 P?	4,3
1,2 OK	2,2 OK	3,2	4,2
1,1 OK	2,1 OK	3,1 P	4,1

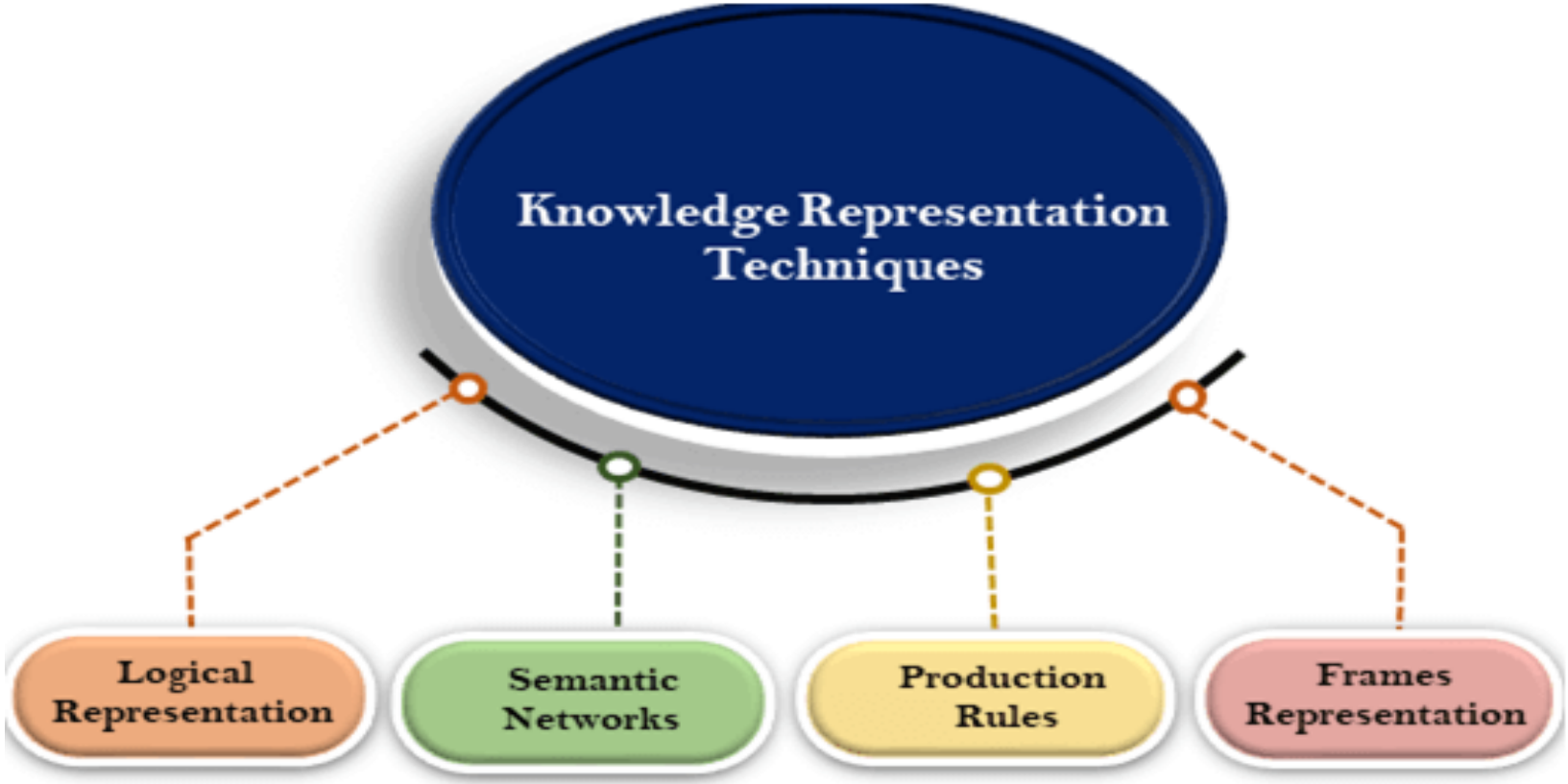
Percept: [Stench, Breeze, Glitter, None, None]

Deduce: Pit in (2,4) or (3,3)
Gold in (2,3)

Action: Move to (1,1) through OK locations

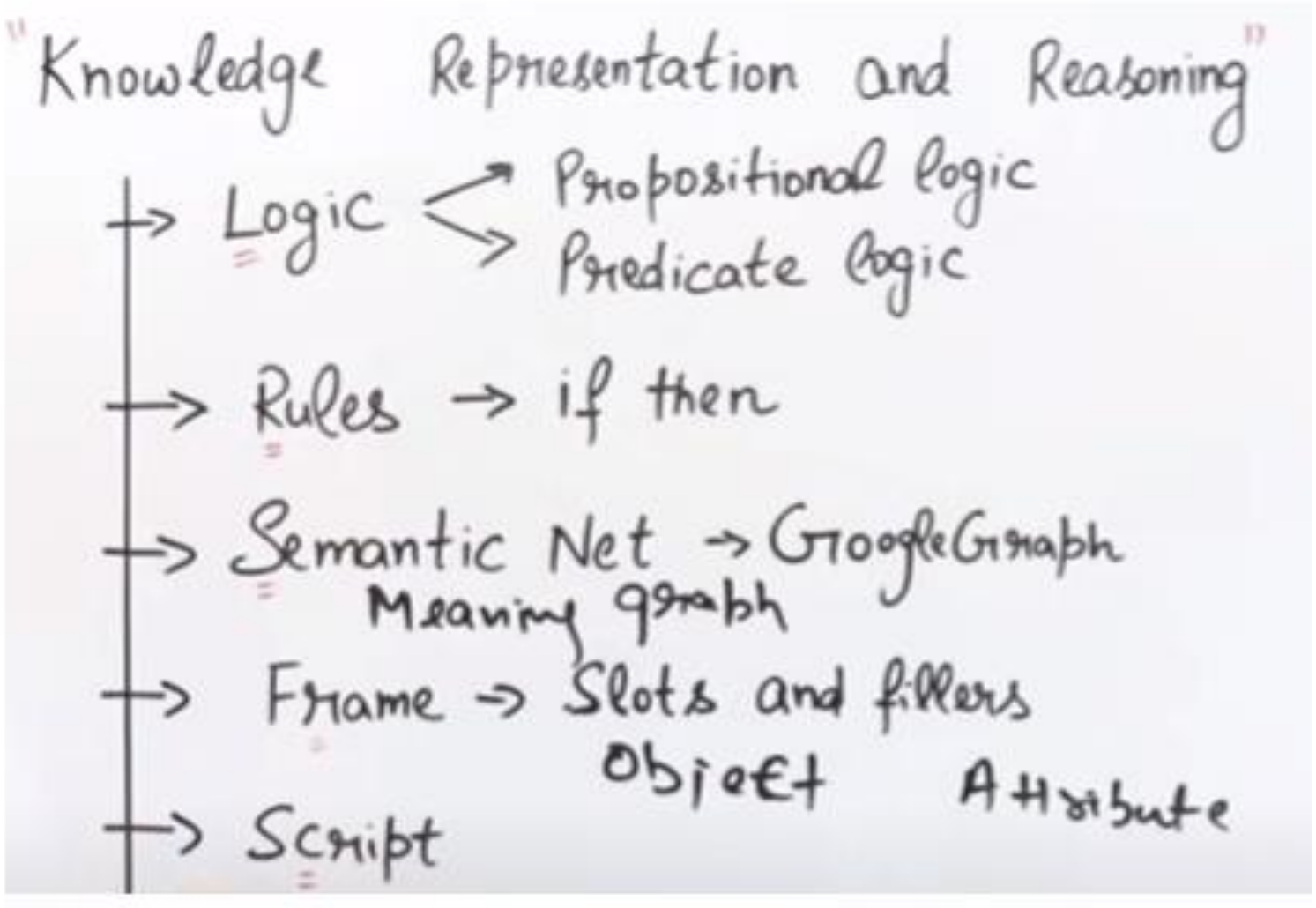
Knowledge Representation Techniques

- How this knowledge can be represented so that a machine can make sense of it. One has to keep in mind that there are numerous ways to achieve this, and no method is perfect and has its own disadvantages.



Knowledge Representation Techniques

- How to represent knowledge



Knowledge Representation Techniques

- **Logical Representation**
- It is the most basic form of representing knowledge to machines where a well-defined syntax with proper rules is used.
- This syntax needs to have no ambiguity in its meaning and must deal with prepositions.
- Thus, this logical form of presentation acts as communication rules and is why it can be best used when representing facts to a machine.
- Logical representation can be categorised into mainly two logics:

➢ Propositional Logics

➢ Predicate logics

Knowledge Representation Techniques

- Logical Representation can be of two types-

➢ **Propositional Logic**: This type of logical representation is also known as propositional calculus or statement logic. This works in a Boolean, i.e., True or False method.

➢ **First-order Logic**: This type of logical representation is also known as the First Order Predicate Calculus Logic (FOPL). This logical representation represents the objects in quantifiers and predicates and is an advanced version of propositional logic.

Knowledge Representation Techniques

- Logical Representation
- Advantages:

➢ Logical representation helps to perform logical reasoning

➢ This representation is the basis for the programming languages.

- Disadvantages:

➢ Logical representations have some restrictions and are challenging to work with

➢ This technique may not be very natural and inference may not be very efficient.

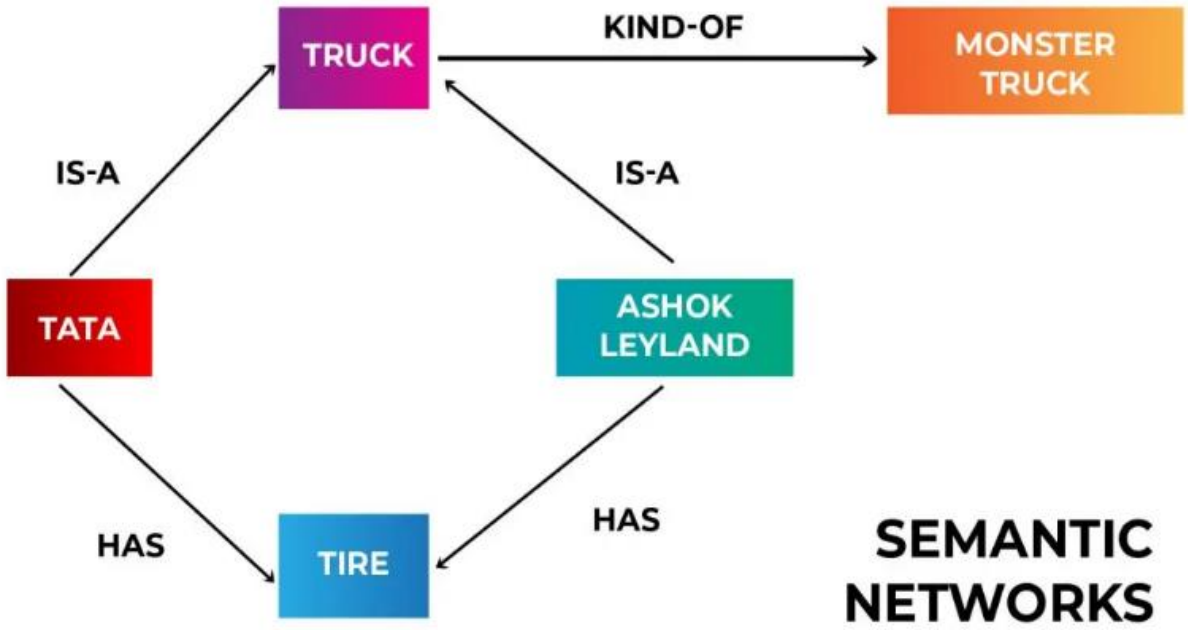
Knowledge Representation Techniques

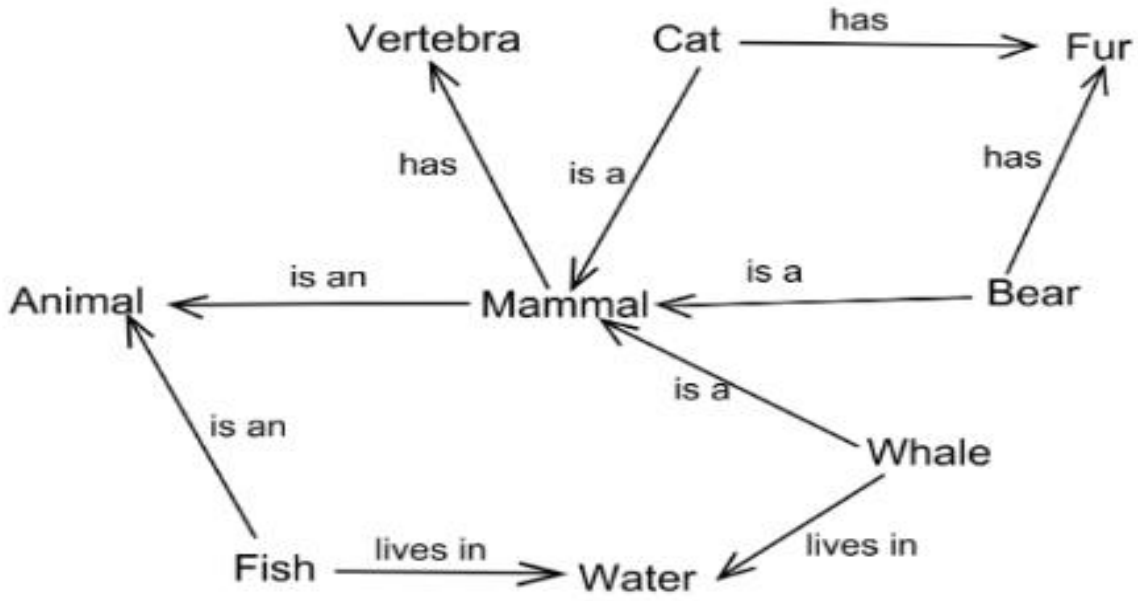
- **Semantic Networks**

- In this form, a graphical representation conveys how the objects are connected and are often used with a data network.
- The Semantic networks consist of node/block (the objects) and arcs/edges (the connections) that explain how the objects are connected. This form of representation is also known as an alternative to the FPOL form of representation.
- The relationships found in the Semantic Networks can be of two types – IS-A and instance (KIND-OF). This form of representation is more natural than logical. It is simple to understand however suffers from being computationally expensive and do not have the equivalent of quantifiers found in the logical representation.

Knowledge Representation Techniques

- **Semantic Networks:** a knowledge representation that represents relationships between concepts and ideas in the form of a network. It is generally shown as a graph where concepts/ideas are “nodes” and relationships are “edges” or arrows



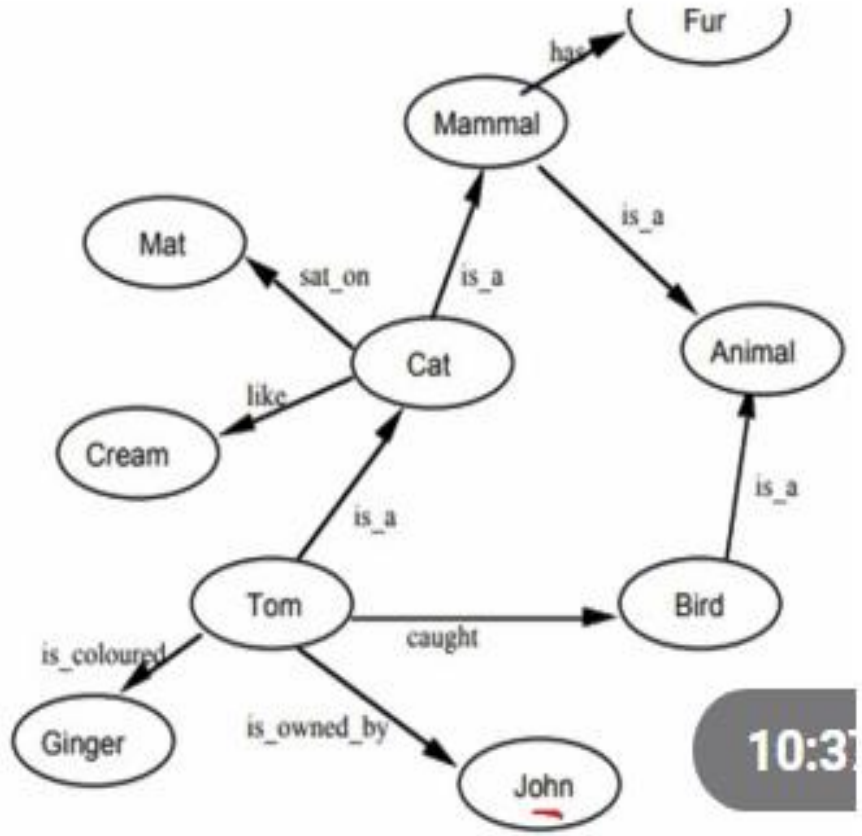


Knowledge Representation Techniques

- **Semantic Networks**

Example :

Tom is a cat.
Tom caught a bird.
Tom is owned by John.
Tom is ginger in colour.
Cats like cream.
The cat sat on the mat.
A cat is a mammal.
A bird is an animal.
All mammals are animals.
Mammals have fur.



Knowledge Representation Techniques

- Advantages:
  - Semantic networks are a natural representation of knowledge
  - It conveys meaning in a transparent manner
  - These networks are simple and easy to understand
- Disadvantages:
  - Semantic networks takes more computational time at runtime
  - These are inadequate as they do not have any equivalent quantifiers.
  - These networks are not intelligent and depends on the creator of the system.

Knowledge Representation Techniques

- **Frame Representation**
- it is a collection of attributes and values linked to it. This AI-specific data structure uses slots and fillers (i.e., slot values, which can be of any data type and shape).
- it has a similar concept to how information is stored in a typical DBMS. These slots and fillers form a structure – a frame. The slots here have the name (attributes), and knowledge related to it is stored in the fillers.
- The biggest advantage of this form of representation is that due to its structure, similar data can be combined in groups as frame representation can divide the knowledge in structures and then further into sub-structures. Also, being like any typical data structure can be understood, visualized, manipulated easily, and typical concepts such as adding, removing, deleting slots can be done effortlessly.

Knowledge Representation Techniques

- Advantages:
- ➢ It makes the programming easier by grouping the related data.
- ➢ Frame representation is easy to understand and visualize.
- ➢ It is very easy to add slots for new attributes and relations also it is easy to include default data and search for missing values.
- Disadvantages:
- ➢ In frame system inference, the mechanism can not be easily processed.
- ➢ The inference mechanism can not be smoothly processed by frame representation.
- ➢ It is very generalized approach.

Knowledge Representation Techniques

- **Production Rules**
- It is among the most common ways in which knowledge is represented in AI systems. In the simplest form, it can be understood as a simple if-else rule-based system and, in a way, is the combination of Propositional and FOPL logics.
- This system comprises a set of production rules, rule applier, working memory, and a recognize act cycle. For every input, conditions are checked from the set of a production rule, and upon finding a suitable rule, an action is committed. This cycle of selecting the rule based on some conditions and consequently acting to solve the problem is known as a recognition and act cycle, which takes place for every input.

Knowledge Representation Techniques

- Advantages:
- ➢ Expressed in natural language
- ➢ Production rules are highly modular and can be easily removed or modified.
- Disadvantages:
- ➢ It does not exhibit any leaning capabilities and does not store the result of the problem for future uses.
- ➢ During the execution of program many rules may be active. Thus rule based production systems are inefficient.

Approaches to Knowledge Representation

- There are different approaches to knowledge representation such as:
- **Simple Relational Knowledge**

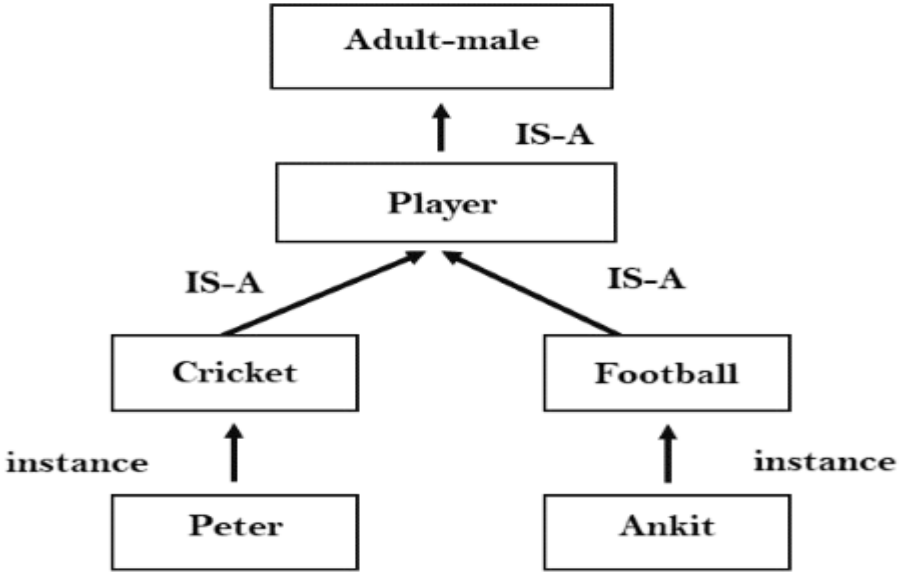
➢ It is the simplest way of storing facts which uses the relational method. Here, all the facts about a set of the object are set out systematically in columns. Also, this approach of knowledge representation is famous in database systems where the relationship between different entities is represented. Thus, there is little opportunity for inference.

Name	Age	Emp ID
John	25	100071
Amanda	23	100056
Sam	27	100042

Approaches to Knowledge Representation

- **Inheritable Knowledge**

➢ In the inheritable knowledge approach, all data must be stored into a **hierarchy of classes** and should be arranged in a generalized form or a hierarchal manner. Also, this approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation. In this approach, objects and values are represented in Boxed nodes.



Approaches to Knowledge Representation

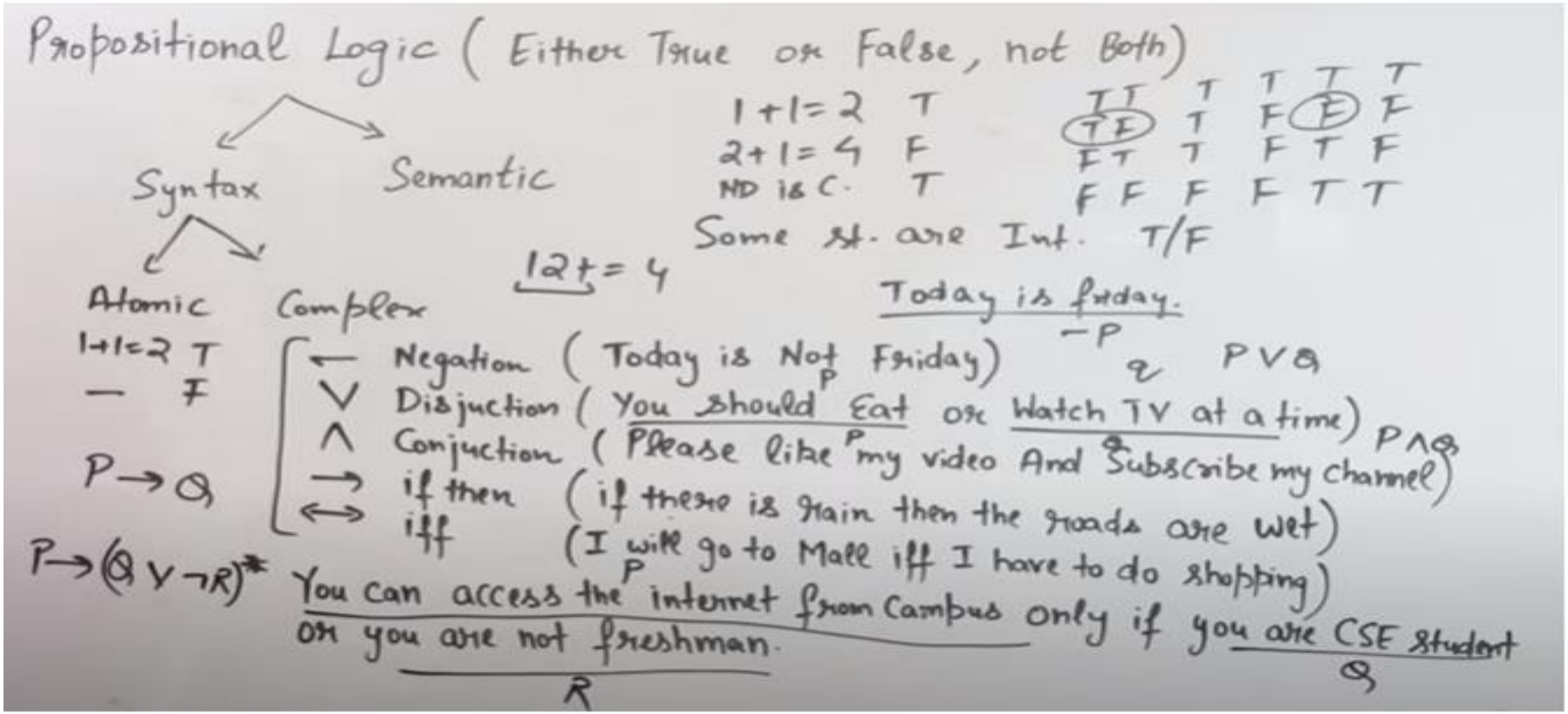
- **Inferential Knowledge**
- The inferential knowledge approach represents **knowledge** in the form of **formal logic**. Thus, it can be used to derive more facts. Also, it guarantees correctness.
- **Example: Statement 1**: John is a cricketer.

  Statement 2: All cricketers are athletes.
- Then it can be represented as; **Cricketer(John)**

  ∀x = Cricketer (x) ———-> Athelete (x)s

Logical Representation-Propositional Logic

- Propositional Logic - One of the simplest method for representing knowledge



Logical Representation-Propositional Logic

- A statement can be defined as a declarative sentence, or part of a sentence, that is capable of having a truth-value, such as being true or false. So Propositions can be either true or false, but it cannot be both.
- **Example**, the following are statements:

➢ George W. Bush is the 43rd President of the United States.

➢ Paris is the capital of France.

➢ Everyone born on Monday has purple hair.

- Sometimes, a statement can contain one or more other statements as parts. Consider for example, the following statement:

➢ Either Ganymede is a moon of Jupiter or Ganymede is a moon of Saturn.

Logical Representation-Propositional Logic

- **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: 2+2 is 4, it is an atomic proposition as it is a **true** fact.

  "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: "It is raining today, and street is wet."

  "Ankit is a doctor, and his clinic is in Mumbai."

Paris is the capital of France and Paris has a population of over two million.

Logical Representation-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.
- Propositional logic consists of an object 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.

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 **Λ** connective such as, **P Λ Q** is called a conjunction.
  Example: Rohan is intelligent and hardworking. It can be written as,
  P= Rohan is intelligent,
  Q= Rohan is hardworking. → PΛ Q.

Logical Connectives

- **Disjunction:** A sentence which has V connective, such as **P V Q**. is called disjunction, where P and Q are the propositions.
  Example: "Ritika is a doctor or Engineer",
  Here P= Ritika is Doctor. Q= Ritika is Doctor, so we can write it as **P V 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.

Logical Connectives

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

- Truth Table

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**

Logical Connectives

- Truth Table

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**

Propositional Logic

- For propositional logic, a row in the truth table is one interpretation

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
False	*False*	*True*	*False*	*False*	*True*	*True*
False	*True*	*True*	*False*	*True*	*True*	*False*
True	*False*	*False*	*False*	*True*	*False*	*False*
True	*True*	*False*	*True*	*True*	*True*	*True*

Logical Connectives

- Truth table with three propositions:

P	Q	R	¬R	P v Q	P v 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

- Logical equivalence:

A	B	¬A	¬A V B	A → B
T	T	F	**T**	**T**
T	F	F	**F**	**F**
F	T	T	**T**	**T**
F	F	T	**T**	**T**

Example

- Translation

Translation of English sentences to propositional logic:

(1) identify atomic sentences that are propositions

(2) Use logical connectives to translate more complex composite sentences that consist of many atomic sentences

Assume the following sentence:

- It is not sunny this afternoon and it is colder than yesterday.

Atomic sentences:

- p = It is sunny this afternoon
- q = it is colder than yesterday

Translation: $\neg p \wedge q$

Example

- Translation

Assume the following sentences:

- It is not sunny this afternoon and it is colder than yesterday. $\neg p \wedge q$
- We will go swimming only if it is sunny. $r \rightarrow p$
- If we do not go swimming then we will take a canoe trip. $\neg r \rightarrow s$
- If we take a canoe trip, then we will be home by sunset. $s \rightarrow t$

Denote:

- p = It is sunny this afternoon
- q = it is colder than yesterday
- r = We will go swimming
- s= we will take a canoe trip
- t= We will be home by sunset

Formalizing English Sentences

- Let’s consider a propositional language where p means ”Paola is happy”, q means ”Paola paints a picture”, and r means ”Renzo is happy”.
- Formalize the following sentences:

➢ ”if Paola is happy and paints a picture then Renzo isn’t happy”

- $p \wedge q \rightarrow \neg r$ 2

➢ ”if Paola is happy, then she paints a picture”

- $p \rightarrow q$

➢ ”Paola is happy only if she paints a picture”

- $\neg(p \wedge \neg q)$ which is equivalent to $p \rightarrow q$!!!

Formalizing English Sentences

- Let A =”Angelo comes to the party”, B =”Bruno comes to the party”, C =”Carlo comes to the party”, and D =”Davide comes to the party”.

1 ”If Davide comes to the party then Bruno and Carlo come too”

2 ”Carlo comes to the party only if Angelo and Bruno do not come”

3 ”If Davide comes to the party, then, if Carlo doesn’t come then Angelo comes”

4 ”Carlo comes to the party provided that Davide doesn’t come, but, if Davide comes, then Bruno doesn’t come”

5 ”A necessary condition for Angelo coming to the party, is that, if Bruno and Carlo aren’t coming, Davide comes”

6 ”Angelo, Bruno and Carlo come to the party if and only if Davide doesn’t come, but, if neither Angelo nor Bruno come, then Davide comes only if Carlo comes”

Formalizing English Sentences

- Let A =”Angelo comes to the party”, B =”Bruno comes to the party”, C =”Carlo comes to the party”, and D =”Davide comes to the party”.

1 ”If Davide comes to the party then Bruno and Carlo come too”

- $D \rightarrow B \wedge C$

2 ”Carlo comes to the party only if Angelo and Bruno do not come”

- $C \rightarrow \neg A \wedge \neg B$

3 ”If Davide comes to the party, then, if Carlo doesn’t come then Angelo comes”

- $D \rightarrow (\neg C \rightarrow A)$

Formalizing English Sentences

- Let A =”Angelo comes to the party”, B =”Bruno comes to the party”, C =”Carlo comes to the party”, and D =”Davide comes to the party”.
- ”Carlo comes to the party provided that Davide doesn’t come, but, if Davide comes, then Bruno doesn’t come”

➢ $(C \rightarrow \neg D) \wedge (D \rightarrow \neg B)$

- ”A necessary condition for Angelo coming to the party, is that, if Bruno and Carlo aren’t coming, Davide comes”

➢ $A \rightarrow (\neg B \wedge \neg C \rightarrow D)$

- ”Angelo, Bruno and Carlo come to the party if and only if Davide doesn’t come, but, if neither Angelo nor Bruno come, then Davide comes only if Carlo comes”

➢ $(A \wedge B \wedge C \leftrightarrow \neg D) \wedge (\neg A \wedge \neg B \rightarrow (D \rightarrow C))$

Contradiction and Tautology

- Some composite sentences may always (under any interpretation) evaluate to a single truth value:

- **Contradiction** (always ***False***)

$$P \wedge \neg P$$

- **Tautology** (always ***True***)

$$P \vee \neg P$$

$$\left.\begin{array}{l} \neg(P \vee Q) \Leftrightarrow (\neg P \wedge \neg Q) \\ \neg(P \wedge Q) \Leftrightarrow (\neg P \vee \neg Q) \end{array}\right\} \text{DeMorgan's Laws}$$

Rules of Inference

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

Rules of Inference

- **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:

P	Q	P → Q	Q → P	¬ Q → ¬ P	¬ P → ¬ Q.
T	T	T	T	T	T
T	F	F	T	F	T
F	T	T	F	T	F
F	F	T	T	T	T

Types of Inference Rules

- **Modus Ponens**: if P and P → Q is true, then we can infer that Q will be true.

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.
- **Proof by Truth table:**

P	Q	$P \rightarrow Q$
0	**0**	**0**
0	**1**	**1**
1	**0**	**0**
1	**1**	**1** ←

Types of Inference Rules

- **Modus Tollens**: The Modus Tollens rule state that if P→ Q is true and **¬ Q is true, then ¬ P** will also true.

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

- **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	$\sim P$	$\sim Q$	$P \rightarrow Q$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	0
1	1	0	0	1

Types of Inference Rules

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

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	←

Types of Inference Rules

- **Disjunctive Syllogism**: The Disjunctive syllogism rule state that if PVQ is true, and ¬P is true, then Q will be true.

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

- Example
- **Statement-1:** Today is Sunday or Monday. ==>PVQ
  Statement-2: Today is not Sunday. ==> ¬P
  Conclusion: Today is Monday. ==> Q
- **Proof by Truth table:**

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

Types of Inference Rules

- **Addition**: The Addition rule is one the common inference rule, and it states that If P is true, then PVQ will be true.

Notation of Addition: $\frac{P}{P \vee Q}$

- **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)
- **Proof by Truth table:**

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

Types of Inference Rules

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

Notation of Simplification rule: $\frac{P \wedge Q}{Q}$ Or $\frac{P \wedge Q}{P}$

- **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**

Notation of Resolution $\frac{P \vee Q,\ \neg P \wedge R}{Q \vee R}$

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	←

Logical Representation-Propositional Logic

- In order to draw conclusions, facts are represented in a more convenient way as,

1. Marcus is a man.

➢ man(Marcus)

2. Plato is a man.

➢ man(Plato)

3. All men are mortal.

➢ mortal(men)

- Caesar was a ruler

➢ ruler(Caesar)

Limitations of Propositional logic

- We cannot represent relations like ALL, some, or none with propositional logic.
- Example:

➢ All the girls 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.

Limitations of Propositional Logic

- Propositional logic cannot express general-purpose knowledge briefly
- We need 32 sentences to describe the relationship between wumpus and stenches
- We would need another 32 sentences for pits and breezes
- We would need at least 64 sentences to describe the effects of actions
- Difficult to identify specific individuals (Mary, among 3)
- Generalizations, patterns, regularities difficult to represent (all triangles have 3 sides)
- Can’t directly talk about properties of individuals or relations between individuals (e.g., “Bill is tall”)

Limitations of Propositional logic

- Wumpus World and propositional logic
- Find Pits in Wumpus world
  - $B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$ (Breeze next to Pit) 16 rules
- Find Wumpus
  - $S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$ (stench next to Wumpus) 16 rules
- At least one Wumpus in world
  - $W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,4}$ (at least 1 Wumpus) 1 rule
- At most one Wumpus
  - $\neg W_{1,1} \vee \neg W_{1,2}$ (155 RULES)

Propositional Logic VS Predicate Logic

Propositional Logic	Predicate Logic
Propositional logic is the logic that deals with a collection of declarative statements which have a truth value, true or false.	Predicate logic is an expression consisting of variables with a specified domain. It consists of objects, relations and functions between the objects.
It is the basic and most widely used logic. Also known as Boolean logic.	It is an extension of propositional logic covering predicates and quantification.
A proposition has a specific truth value, either true or false.	A predicate's truth value depends on the variables' value.
Scope analysis is not done in propositional logic.	Predicate logic helps analyze the scope of the subject over the predicate. There are two quantifiers : Universal Quantifier (∀) depicts for all, Existential Quantifier (∃) depicting there exists some
Propositions are combined with Logical Operators or Logical Connectives like Negation(¬), Disjunction(∨), Conjunction(∧), Exclusive OR(⊕), Implication(⇒), Bi-Conditional or Double Implication(⇔).	Predicate Logic adds by introducing quantifiers to the existing proposition.

Thank you.

Unit-2

Predicate Logic

Predicate Logic / Firs order Logic / First order Predicate Logic

First-Order Logic

- In propositional logic, we can only represent the facts, which are either true or false. The propositional logic has very limited expressive power. Consider the following sentence, which we cannot represent using PL logic.
- **"Some humans are intelligent", or**
- **"Sachin likes cricket."**
- To represent the above statements, PL logic is not sufficient, so we required some more powerful logic, such as first-order logic.

First-Order Logic

- First-order logic 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:

First-Order Predicate Calculus

- First-Order Logic includes:
  - Objects: peoples, numbers, places, ideas (atoms)
  - Relations: relationships between objects (predicates, T/F value)
    - Example: father(fred, mary)
    - Properties: properties of atoms (predicates, T/F value)
      Example: red(ball)
  - Functions: father-of(mary), next(3), (any value in range)
    - Constant: function with no parameters, MARY

Logics in General

- Ontological Commitment:
  - What exists in the world — TRUTH
  - PL : facts hold or do not hold.
  - FOL : objects with relations between them that hold or do not hold

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0, 1]$
Fuzzy logic	degree of truth $\in [0, 1]$	known interval value

Syntax of FOL: Basic elements

- Constant Symbols:
  - Stand for objects
  - e.g., KingJohn, 2, UCI,...

- Predicate Symbols
  - Stand for relations
  - E.g., Brother(Richard, John), greater_than(3,2)...

- Function Symbols
  - Stand for functions
  - E.g., Sqrt(3), LeftLegOf(John),...

Syntax of FOL: Basic elements

- Constants KingJohn, 2, UCI,...
- Predicates Brother, >,...
- Functions Sqrt, LeftLegOf,...
- Variables x, y, a, b,...
- Connectives $\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$
- Equality =
- Quantifiers $\forall, \exists$

Free and Bound Variables

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

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

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

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

➢ A variable x is bound in a wff φ if and only if it is in the scope of a quantifier ∀x or ∃x in φ; otherwise it is free.

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

Syntax: First-Order Logic

- **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: Chinky is a cat: => cat (Chinky)**.
- **Complex Sentences:**

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

Syntax: First-Order 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.
- A predicate is a property that is affirmed or denied about the subject (in logic, we say “variable” or “argument”) of a statement
- **Example: "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.
- “x is greater than 3”
- x |subject is greater than 3” | predicate

Syntax: First-Order Logic

- We introduce a (functional) symbol for the predicate, and put the subject as an argument (to the functional symbol): P(x)
- Examples:

- Father(x): unary predicate
- Brother(x,y): binary predicate
- Sum(x,y,z): ternary predicate
- P(x,y,z,t): n-ary predicate

Syntax: First-Order Logic

- **Quantifiers in First-order logic**

➢ A predicate becomes a proposition when we assign it fixed values. However, another way to make a predicate into a proposition is to quantify it. That is, the predicate is true (or false) for **all possible values** in the universe of discourse or for **some value(s)** 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).

Syntax: First-Order Logic

- **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.

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

  – **For all x**
  – **For each x**
  – **For every x.**

- Example:
- **All man drink coffee.** ∀x man(x) → drink (x, coffee).

Syntax: First-Order Logic

- **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 ∃, When it is used with a predicate variable then it is called as an existential quantifier.

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

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

- Example:
- **Some boys are intelligent.** ∃x: boys(x) ∧ intelligent(x)

Syntax: First-Order Logic

- **Well Formed Formula (wff)**:
- A well-formed formula, is obtained by composing atoms with logical connectives and quantifiers. Therefore, a well-formed formula is a predicate with the following properties:

➢ All atomic formulas (atoms) are well-formed formulas.

➢ Combines using propositional connectives such as conjunction, disjunction and negation.

➢ If x is a variable and F is a well formed formula function, then both quantifiers are well formed formulas.

- A well-formed formula (wff) is a sentence containing no “free” variables. That is, all variables are “bound” by universal or existential quantifiers.

Syntax: First-Order Logic

- The Syntactic Rules (Individual terms are just the individual constants and individual variables)
- If P is an n-place predicate, and t1, t2, …, tn are terms, then P(t1, t2, …, tn) is a wff
- If φ and ψ are wff, then $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \leftrightarrow \psi)$ are wffs
- If φ is a wff and x is an individual variable, then $\forall x\, \varphi$ and $\exists x\, \varphi$ are wffs
- Nothing else is a wff

Syntax: First-Order Logic

- **Well Formed Formula**: A well-formed formula (wff) is a sentence containing no “free” variables. That is, all variables are “bound” by universal or existential quantifiers.
- **Quantifiers**

➢ **Universal quantification**

• (∀x)P(x) means that P holds for **all values of x** in the domain associated with that variable

• E.g., (∀x) dolphin(x) → mammal(x)

➢ **Existential quantification**

• (∃ x)P(x) means that P holds for **some value of x** in the domain associated with that variable

• E.g., (∃ x) mammal(x) ∧ lays-eggs(x)

Syntax: First-Order Logic

- **Properties of Quantifiers:**

➢ In universal quantifier, ∀x∀y is similar to ∀y∀x.

➢ In Existential quantifier, ∃x∃y is similar to ∃y∃x.

➢ ∃x∀y is not similar to ∀y∃x. Why?

Example:

- 1) ∀x∃y: For every x there exists a y (such that...)
  Everyone is married to someone (ie. For every person there exists a person to whom he/she is married)
- 2) ∃y∀x: There exists a y (such that) for every x...
  Someone is married to everyone (ie. There is someone who is married to everyone)

Syntax: First-Order Logic

- **Some Predicate Calculus Equivalence**

Suppose p and q are predicates; X and Y are variables

- $\neg \exists X\ p(X) = \forall X \neg p(X)$
- $\neg \forall X\ p(X) = \exists X \neg p(X)$
- $\exists X\ p(X) = \exists Y\ p(Y)$
- $\forall X\ p(X) = \forall Y\ p(Y)$
- $X\ (p(X) \wedge q(X)) = \forall X\ p(X) \wedge \forall Y\ q(Y)$
- $\exists X\ (p(X) \wedge q(X)) = \exists X\ p(X) \wedge \forall Y\ q(Y)$

Examples

- All men are mortal

Examples

- All men are mortal
  - $\forall$x [Man(x) -> Mortal(x)]

Examples

- All men are mortal
  - $\forall$x [Man(x) -> Mortal(x)]
- Socrates is a man
  - Man(Socrates)
- Socrates is mortal
  - Mortal(Socrates)

Examples

- All men are mortal
  - $\forall$x [Man(x) -> Mortal(x)]
- Socrates is a man
  - Man(Socrates)
- Socrates is mortal
  - Mortal(Socrates)
- All purple mushrooms are poisonous
  - $\forall$ x [(Purple(x) ^ Mushroom(x)) -> Poisonous(x)]

Examples

- All men are mortal
  - ∀x [Man(x) -> Mortal(x)]
- Socrates is a man
  - Man(Socrates)
- Socrates is mortal
  - Mortal(Socrates)
- All purple mushrooms are poisonous
  - ∀x [(Purple(x) ^ Mushroom(x)) -> Poisonous(x)]
- A mushroom is poisonous only if it is purple
  - ∀x [(Mushroom(x) ^ Poisonous(x)) -> Purple(x)]

Examples

- No human enjoys golf

$$\forall x[Human(x) \rightarrow \neg Enjoys(x, Golf)$$

- Some professor that is not a historian writes programs

$$\exists x[\mathrm{Pr}ofessor(x) \wedge \neg Historian(x) \wedge Writes(x, \mathrm{Pr}ograms)]$$

- All birds fly. ∀x bird(x) →fly(x).
- Every man respects his parent. ∀x man(x) → respects (x, parent).
- Some boys play cricket. ∃x boys(x) ^ play(x, cricket).
- Not all students like both Mathematics and Science.

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

Representing facts with Predicate Logic - Example

Ram is a man.

man(Ram)

God loves Ram.

loves(God, Ram)

Ram eats an apple.

eats(Ram, apple)

Sita drinks water.

drinks(sita, water)

Every man loves god.

$\forall$ x (x, man(x) --> loves (x, god))

Everyone loves someone

$\forall x \exists y$ Loves (x,y)

There is one person everyone loves

$\exists y \forall x$ Loves (x,y)

All birds have feathers.

$\forall$ x [Feathers(x) $\Rightarrow$ Bird(x)]

Every dog is an animal.

$\forall$ x (dog(x) $\Rightarrow$ animal (x))

Representing facts with Predicate Logic - Example

- Marcus was a man
- Marcus was a Pompeian
- All Pompeians were Romans
- Caesar was a ruler.
- All Romans were either loyal to Caesar or hated him.
- Everyone is loyal to someone.
- Men only try to assassinate rulers they are not loyal to.
- Marcus tried to assassinate Caesar

Representing facts with Predicate Logic - Example

$$\text{Man}(\text{Marcus})$$

$$\text{Pompeian}(\text{Marcus})$$

$$\forall\ x\ \text{Pompeian}(x) \Rightarrow\ \text{Roman}(x)$$

$$\text{Ruler}(\text{Caesar})$$

$$\forall\ x\ \text{Romans}(x) \Rightarrow\ \text{Loyalto}(x,\text{Caesar}) \vee \text{Hate}(x,\text{Caesar})$$

$$\forall\ x\ \exists y\ \text{Loyalto}(x,y)$$

$$\forall\ x\ \forall\ y\ \text{Man}(x) \wedge\ \text{Ruler}(y) \wedge\ \text{Tryassassinate}(x,y) \Rightarrow \neg\ \text{Loyalto}(x,y)$$

$$\text{Tryassassinate}(\text{Marcus},\text{Caesar})$$

Unification

- Unification is a process of making two different logical atomic expressions identical by finding a substitution. Unification depends on the substitution process.
- It takes two literals as input and makes them identical using substitution.
- Let Ψ_1 and Ψ_2 be two atomic sentences and σ be a unifier such that, $\boldsymbol{\Psi_1\sigma = \Psi_2\sigma}$, then it can be expressed as **UNIFY(Ψ_1, Ψ_2)**.
- **Example:** Let Ψ_1 = King(x), Ψ_2 = King(John),
- **Substitution θ = {John/x}** is a unifier for these atoms and applying this substitution, and both expressions will be identical.

Unification

- The UNIFY algorithm is used for unification, which takes two atomic sentences and returns a unifier for those sentences (If any exist).
- Unification is a key component of all first-order inference algorithms.
- It returns **fail** if the expressions **do not match** with each other.
- The substitution variables are called Most General Unifier or MGU.

➢ **E.g.** Let's say there are two different expressions, **P(x, y), and P(a, f(z))**.

- Substitute x with a, and y with f(z) in the first expression, and it will be represented as **a/x** and f(z)/y.
- With both the substitutions, the first expression will be identical to the second expression and the substitution set will be: **[a/x, f(z)/y]**.

Unification

- Conditions for Unification:

➢ Predicate symbol must be same, atoms or expression with different predicate symbol can never be unified.

➢ Number of Arguments in both expressions must be identical.

➢ Unification will fail if there are two similar variables present in the same expression.

Unification Algorithm

1. Initial predicate symbols must match.
2. For each pair of predicate arguments:

- different constants cannot match.
- a variable may be replaced by a constant.
- a variable may be replaced by another variable.
- a variable may be replaced by a function as long as the function does not contain an instance of the variable.

- When attempting to match 2 literals, all substitutions must be made to the entire literal.
- There may be many substitutions that unify 2 literals, the most general unifier is always desired.

Unification Examples

- P(x) and P(y): substitution = (x/y) (“substitute x for y”)
- P(x,f(y)) and P(Joe,z): (Joe/x, f(y)/z)
- P(x) V Q(Jane) and P(Bill) V Q(y): (Bill/x, Jane/y)
- Man(Marcus) and ¬Man(spot) (Not unified)
- Man(Marcus) and Man(Marcus, X) (Not unified)
- Man(Marcus) and Man(X) (X is substitute to Marcus so unified)
- Man(hate(X,Y)) and Man(hate(Marcus, spot)) (Marcus/X and spot/Y)

Normal Forms

- Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms
- Conjunctive Normal Form (CNF)
  conjunction of disjunctions of literals (conjunction of clauses)
  For example, (A v –B) ^ (B v –C v –D)
- Disjunctive Normal Form (DNF)
  disjunction of conjunctions of literals (disjunction of terms)
  For example, (A ^ B) v (A ^ -C) v (A ^ -D) v (-B ^ -C) v (-B ^ -D)
- Horn Form (restricted)
  conjunction of *Horn clauses* (clauses with <= 1 positive literal)
  For example, (A v –B) ^ (B v –C v –D)
  Often written as a set of implications:
  B -> A and (C ^ D) -> B

Convert first-order logic expressions to normal form

- **Why CNF (Conjunctive Normal Form)?**

➢ To make our representation simple.

➢ In wff, our representation of fact is very complicated.

➢ In CNF form, simplicity should be there, embedding should not be there and all quantifiers should not be there.

Example: All Romans who know Markus either hate Ceaser or think that any one who hates anyone is crazy.

$\forall$ x: [roman(x) ^ know(x , Markus)] -> [hate(x, ceaser) V ($\forall$ y: $\exists$ z: hate(y,z) -> thinkcrazy(x,y)]

Convert first-order logic expressions to normal form

- **Rules to convert wff to CNF**

Step-1: Eliminate biconditionals and implications:

- Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

Step-2: Move $\neg$ inwards:

- $\neg(\forall\ x\ p) \equiv \exists\ x\ \neg p$,
- $\neg(\exists\ x\ p) \equiv \forall\ x\ \neg p$,
- $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$,
- $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$,
- $\neg\neg\alpha \equiv \alpha$.

Convert first-order logic expressions to normal form

- **Rules to convert wff to CNF**

Step-3: Standardize variables apart by renaming them: each quantifier should use a different variable.

Step-4. **Skolemize**: each existential variable is replaced by a Skolem constant or Skolem function of the enclosing universally quantified variables.

- For instance, ∃x Rich(x) becomes Rich(G1) where G1 is a new Skolem constant.

- "Everyone has a heart" ∀ x Person(x) ⇒ ∃ y Heart(y) ∧ Has(x, y) becomes ∀ x Person(x) ⇒ Heart(H(x)) ∧ Has(x, H(x)), where H is a new symbol (Skolem function).

Convert first-order logic expressions to normal form

- **Rules to convert wff to CNF**

Step-5: Convert to prenex form by moving all universal quantifiers to the beginning of the wff.

➢ wff in prenex form = Prefix (string of quantifiers) + Matrix (quantifier-free formula)

Step-6: Drop universal quantifiers

• For instance, ∀ x Person(x) becomes Person(x).

Step-7. Distribute ∧ over ∨: Use distributive laws and equivalence rules of propositional logic to transform the matrix to CNF.

• $(\alpha \wedge \beta) \vee \gamma \equiv (\alpha \vee \gamma) \wedge (\beta \vee \gamma)$.

Skolem functions

- ∃ y President(y)
- We replace y with a new function func:
- President(func())
- func is called a skolem function.

Example:

" ∀ x ∃ y Father(y,x)

create a new function named foo and replace y with the function.

" ∀ x Father(foo(x),x)

Convert first-order logic expressions to normal form

- **Example Conversion to CNF**

$[(\forall x)Q(x)] \Rightarrow (\forall x)(\forall y)[(\exists z)[P(x,y,z) \Rightarrow (\forall u)R(x,y,u,z)]$

- Eliminate implications $\Leftrightarrow\Leftrightarrow$, $\Rightarrow$

$\neg[(\forall x)Q(x)] \vee (\forall x)(\forall y)[(\exists z)[\neg P(x,y,z) \vee (\forall u)R(x,y,u,z)]$

- Reduce scope of negations:

$[(\exists x)\neg Q(x)] \vee (\forall x)(\forall y)[(\exists z)[\neg P(x,y,z) \vee (\forall u)R(x,y,u,z)]$

- Standardize variables

$[(\exists w)\neg Q(w)] \vee (\forall x)(\forall y)[(\exists z)[\neg P(x,y,z) \vee (\forall u)R(x,y,u,z)]$

Convert first-order logic expressions to normal form

- Skolemization:

¬Q(A)V(∀x,y)[¬P(x,y,f(x,y))V(∀u)R(x,y,u,f(x,y))]

- Prenex form:

(∀x,y,u)[¬Q(A)V¬P(x,y,f(x,y))VR(x,y,u,f(x,y))

- Drop universal quantifiers

¬Q(A)V¬P(x,y,f(x,y))VR(x,y,u,f(x,y))

- Convert to CNF: wff already in CNF

Convert first-order logic expressions to normal form

- **Example2 Conversion to CNF**

∀ x: ([roman(x) ^ know(x , Markus)] -> [hate(x, ceaser)) V (∀ y: ∃ z: hate(y,z)) -> thinkcrazy(x,y)]

- Eliminate implications ⇔⇔, ⇒

∀ x: ¬ [roman(x) ^ know(x , Markus)] V [hate(x, ceaser) V ¬ (∀ y: ∃ z: hate(y,z)) V thinkcrazy(x,y)]

- Reduce scope of negations:

∀ x: [¬ roman(x) V ¬ know(x , Markus)] V [hate(x, ceaser) V ∃y: ∀ z: ¬ hate(y,z)) V thinkcrazy(x,y)]

- Standardize variables

Skip this step

Convert first-order logic expressions to normal form

- Skolemization:

∀ x: [¬ roman(x) V ¬ know(x , Markus)] V [hate(x, ceaser) V ∀ y ∀ z: ¬ hate(y,z)) V thinkcrazy(x,y)]

- Prenex form:

(∀x,y,z) [¬ roman(x) V ¬ know(x , Markus)] V [hate(x, ceaser) V (¬ hate(y,z)) V thinkcrazy(x,y)]

- Drop universal quantifiers

¬ roman(x) V ¬ know(x , Markus) V [hate(x, ceaser) V (¬ hate(y,z)) V thinkcrazy(x,y)]

- Convert to CNF

¬ roman(x) V ¬ know(x , Markus) V hate(x, ceaser) V ¬ hate(y,z) V thinkcrazy(x,y)

Convert first-order logic expressions to normal form

- **Example3**

∀ x: (¬ literate(x) -> (¬write(x) ^ (¬ ∃y: (read(x,y) ^ book(y)))

➢ Eliminate implications ⇔⇔, ⇒

∀ x: literate(x) V (¬write(x) ^ ∀y: ¬(read(x,y) ^ book(y))

➢ Reduce scope of negations:

∀ x: literate(x) V (¬write(x) ^ ∀y: ¬read(x,y) V ¬book(y))

➢ Standardize variables

Skip this step

Convert first-order logic expressions to normal form

- Prenex form:

∀ x: ∀y: literate(x) V (¬write(x)) ^ (¬read(x,y) V ¬book(y))

- Drop universal quantifiers

literate(x) V (¬write(x)) ^ (¬read(x,y) V ¬book(y))

- Conjuction and disjunction

A V (B ^ (C VD)) = (A V B) ^ (A V (C V D))

where (C V D) = X

(literate(x) V ¬write(x)) ^ (literate (x) V (¬read(x,y) V ¬book(y)))

Here, A ^ B so A and B both are considered as separate facts.

i) literate(x) V ¬write(x)

ii) literate (x) V (¬read(x,y) V ¬book(y))

Resolution

- Resolution is a procedure to prove a statement.
- Steps for Resolution:
- Conversion of facts into first-order logic.
- Convert FOL statements into CNF
- Negate the statement which needs to prove (proof by contradiction)
- Draw resolution graph (unification). (From Knowledge base find out fact/term as negative meaning so both will be elemented)
- Resolvent: From two parent clauses which was derived is called as resolvent.

Resolution

- **Example**
- Marcus was a man
- Marcus was a Pompeian
- All Pompeians were Romans
- Caesar was a ruler.
- All Romans were either loyal to Caesar or hated him.
- Everyone is loyal to someone.
- Men only try to assassinate rulers they are not loyal to.
- Marcus tried to assassinate Caesar

Knowledge Base in First Order Logic

$\text{Man}(\text{Marcus})$

$\text{Pompeian}(\text{Marcus})$

$\forall\ x\ \text{Pompeian}(x) \Rightarrow\ \text{Roman}(x)$

$\text{Ruler}(\text{Caesar})$

$\forall\ x\ \text{Romans}(x) \Rightarrow\ \text{Loyalto}(x,\text{Caesar}) \vee \text{Hate}(x,\text{Caesar})$

$\forall\ x\ \exists y\ \text{Loyalto}(x,y)$

$\forall\ x\ \forall\ y\ \text{Man}(x) \wedge\ \text{Ruler}(y) \wedge\ \text{Tryassassinate}(x,y) \Rightarrow$
$\neg\ \text{Loyalto}(x,y)$

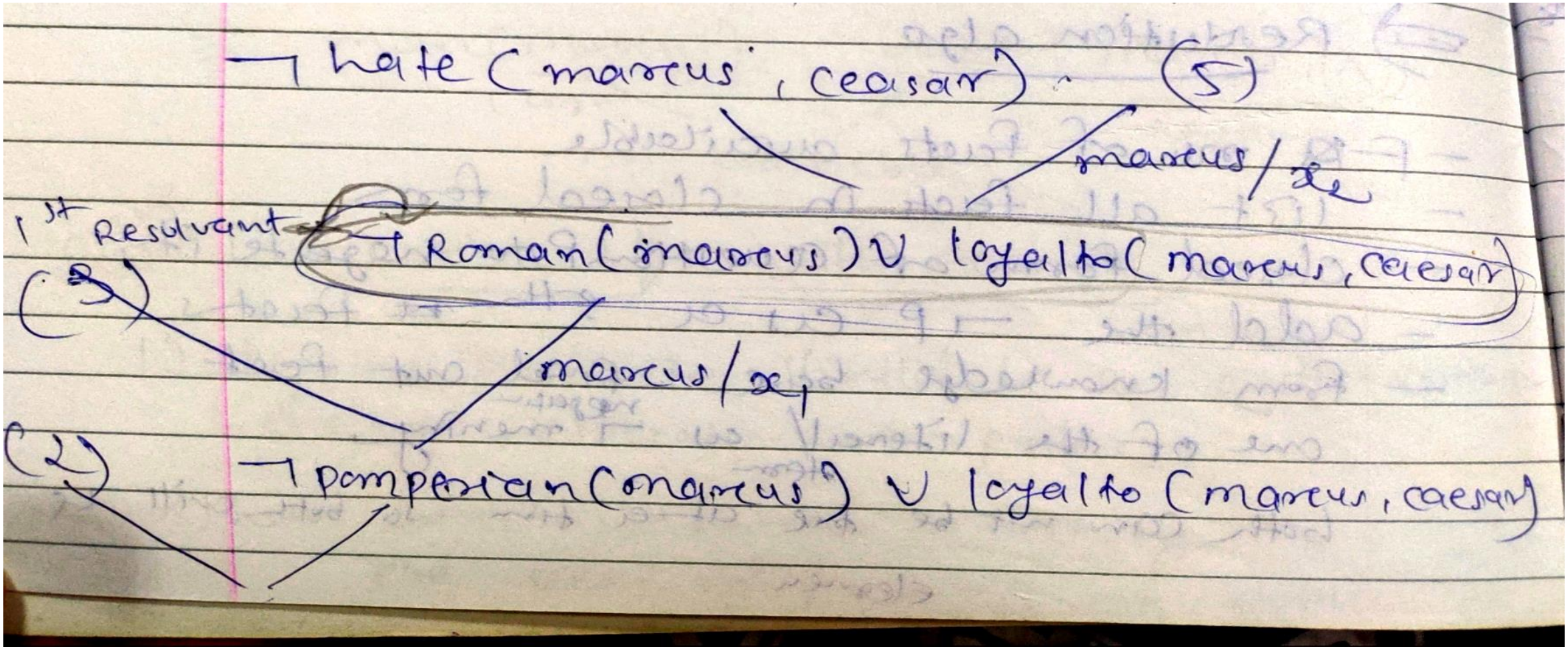
$\text{Tryassassinate}(\text{Marcus},\text{Caesar})$

Knowledge Base in CNF

- Man(Marcus)
- Pompeian(Marcus)
- ¬ Pompeian(x_1) ∨ Roman(x_1)
- Ruler(Caesar)
- ¬ Romans(x_2) ∨ Loyalto(x_2,Caesar) ∨ Hate(x_2,Caesar)
- Loyalto(x_3, $f(x_3)$)
- ¬ Man(x_4) ∨ ¬ Ruler(y_1) ∨ ¬ Tryassassinate(x_4,y_1) ∨ Loyalto(x_4,y_1)

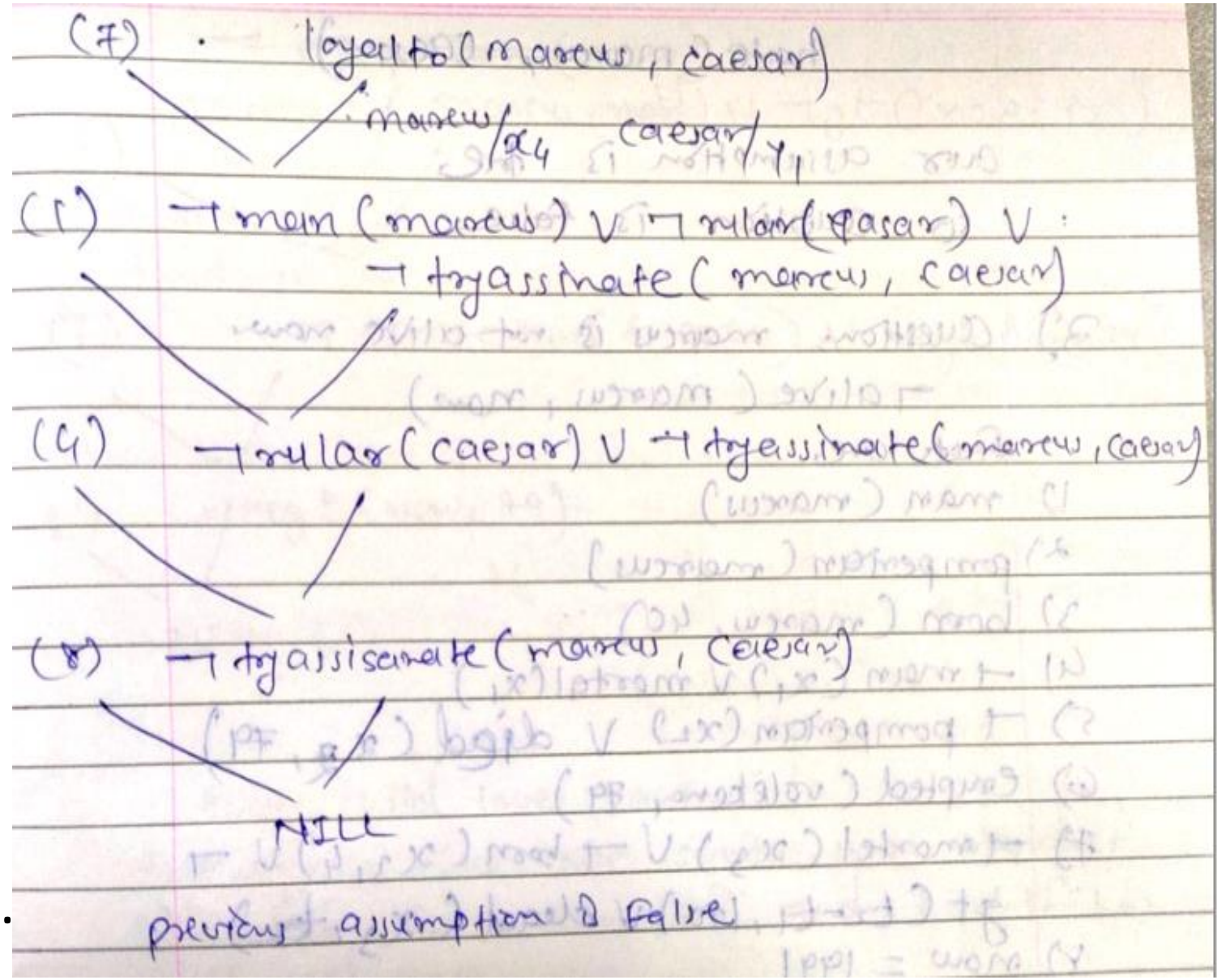
Knowledge Base in CNF

- Query: Does Marcus hate Caesar?
- Negation of Query ¬Hate(Marcus, Caesar)



Resolution

- Does Marcus hate Caesar?

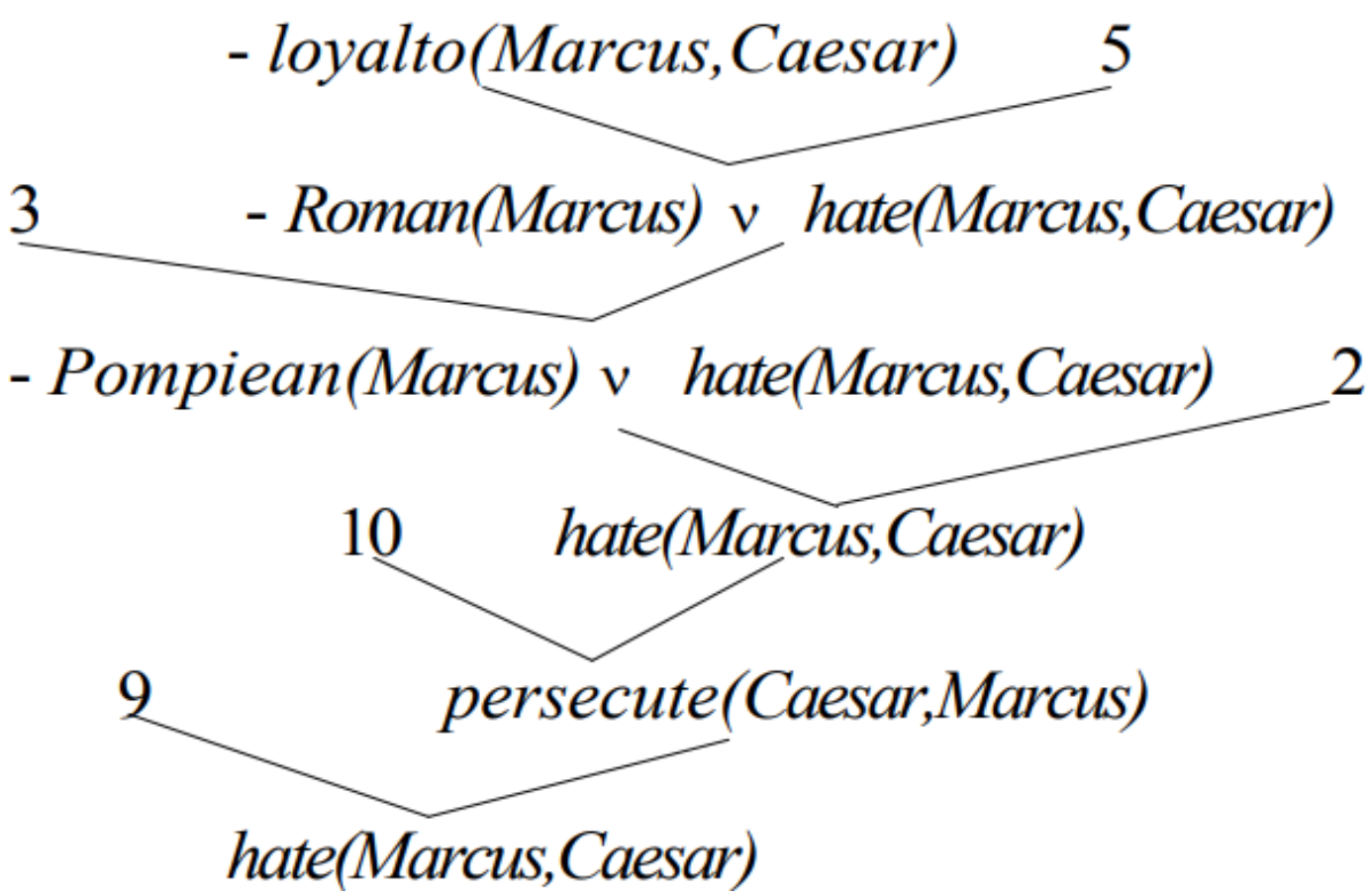


- So hate(Markus, ceaser) is true.

Resolution

- An Unsuccessful Attempt at Resolution

Prove: *loyalto(Marcus,Caesar)*



Resolution

- Example
- John likes all kind of food.
- Apple and vegetable are food
- Anything anyone eats and not killed is food.
- Anil eats peanuts and still alive
- Harry eats everything that Anil eats.
- Prove by resolution that:
- John likes peanuts.

Conversion of Facts into FOL

- Step-1: Conversion of Facts into FOL

a. $\forall x$: food(x) $\rightarrow$ likes(John, x)

b. food(Apple) $\wedge$ food(vegetables)

c. $\forall x \forall y$: eats(x, y) $\wedge \neg$ killed(x) $\rightarrow$ food(y)

d. eats (Anil, Peanuts) $\wedge$ alive(Anil).

e. $\forall x$: eats(Anil, x) $\rightarrow$ eats(Harry, x)

f. $\forall x$: $\neg$ killed(x) $\rightarrow$ alive(x) } **added predicates.**

g. $\forall x$: alive(x) $\rightarrow \neg$ killed(x) }

h. likes(John, Peanuts)

Conversion of FOL into CNF

- Step-2: Conversion of FOL into CNF

Eliminate all implication (→) and rewrite

a. ∀x ¬ food(x) V likes(John, x)

b. food(Apple) Λ food(vegetables)

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

d. eats (Anil, Peanuts) Λ alive(Anil)

e. ∀x ¬ eats(Anil, x) V eats(Harry, x)

f. ∀x¬ [¬ killed(x)] V alive(x)

g. ∀x ¬ alive(x) V ¬ killed(x)

h. likes(John, Peanuts).

Move negation (¬)inwards and rewrite

a. ∀x ¬ food(x) V likes(John, x)

b. food(Apple) Λ food(vegetables)

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

d. eats (Anil, Peanuts) Λ alive(Anil)

e. ∀x ¬ eats(Anil, x) V eats(Harry, x)

f. ∀x ¬killed(x)] V alive(x)

g. ∀x ¬ alive(x) V ¬ killed(x)

h. likes(John, Peanuts).

Resolution Example

- Step-2: Conversion of FOL into CNF
- **Eliminate existential instantiation quantifier by elimination.**

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

- **Drop Universal quantifiers.**

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

Resolution Example

a. ¬ food(x) V likes(John, x)

b. food(Apple)

c. food(vegetables)

d. ¬ eats(y, z) V killed(y) V food(z)

e. eats (Anil, Peanuts)

f. alive(Anil)

g. ¬ eats(Anil, w) V eats(Harry, w)

h. killed(g) V alive(g)

i. ¬ alive(k) V ¬ killed(k)

j. likes(John, Peanuts).

- **Distribute conjunction Λ over disjunction ¬.**

➢ This step will not make any change in this problem.

Resolution Example

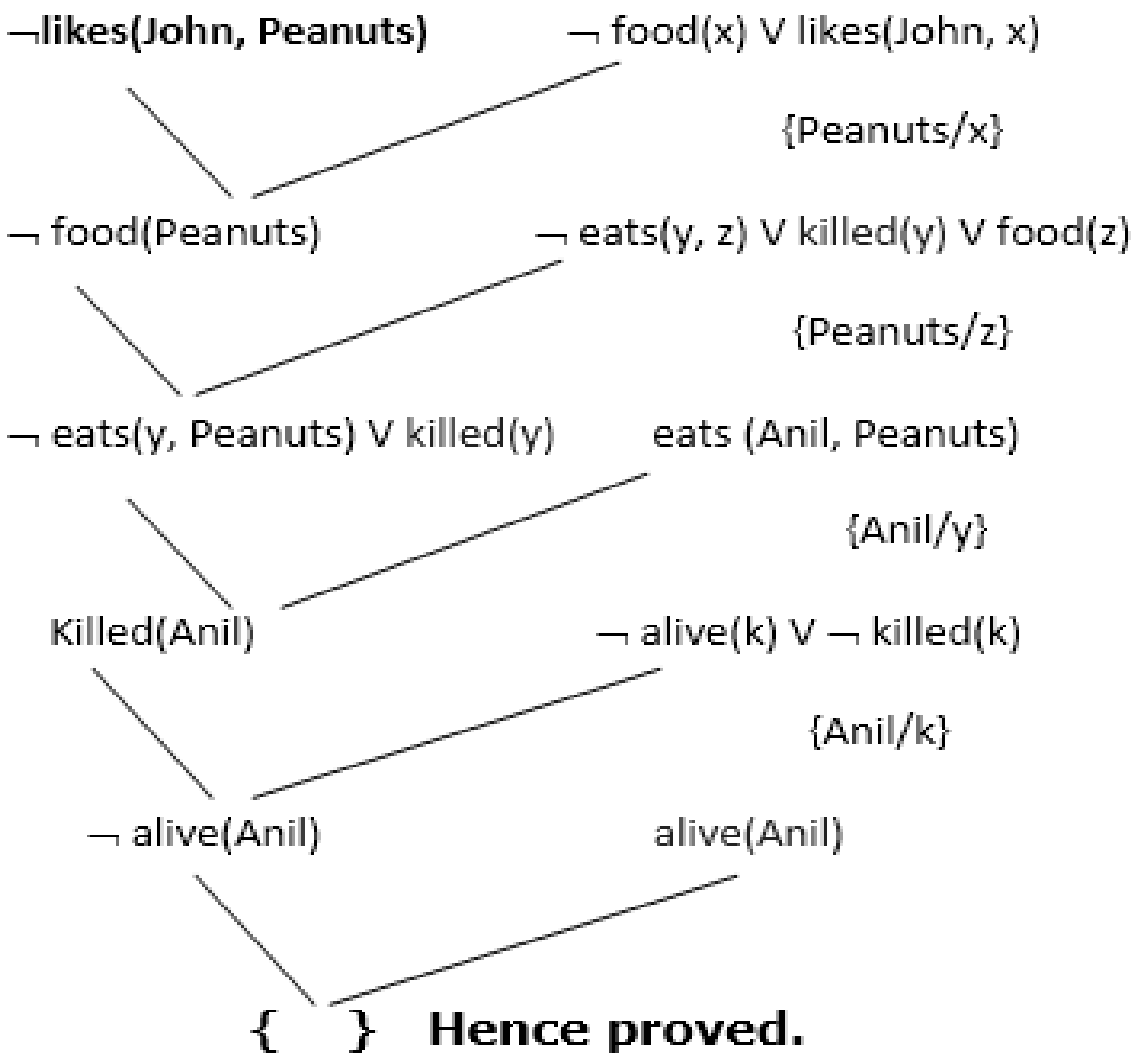
- Step-3: Negate the statement to be proved

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

- Step-4: Draw Resolution graph:

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

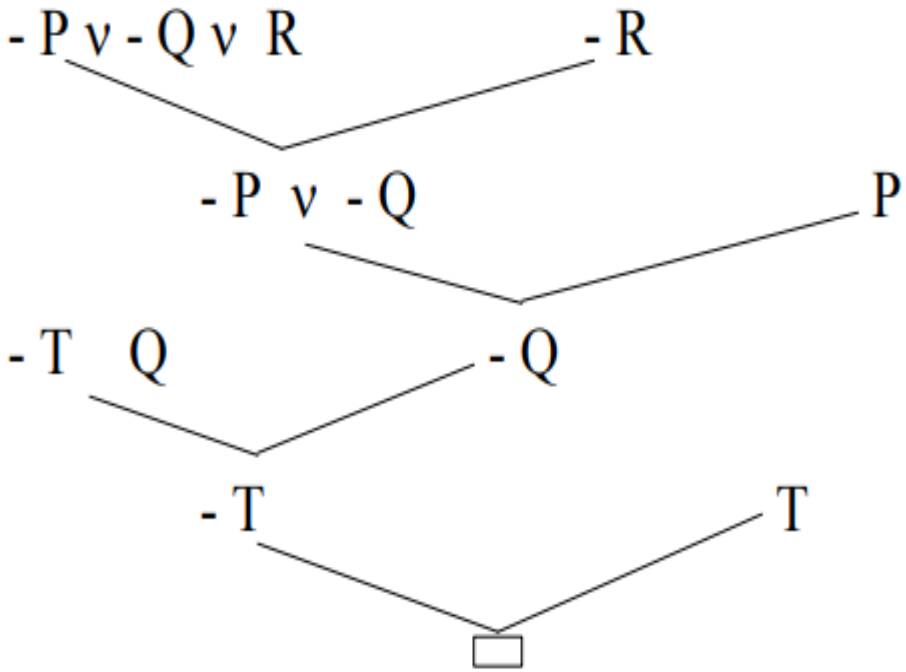
Resolution Example



Resolution

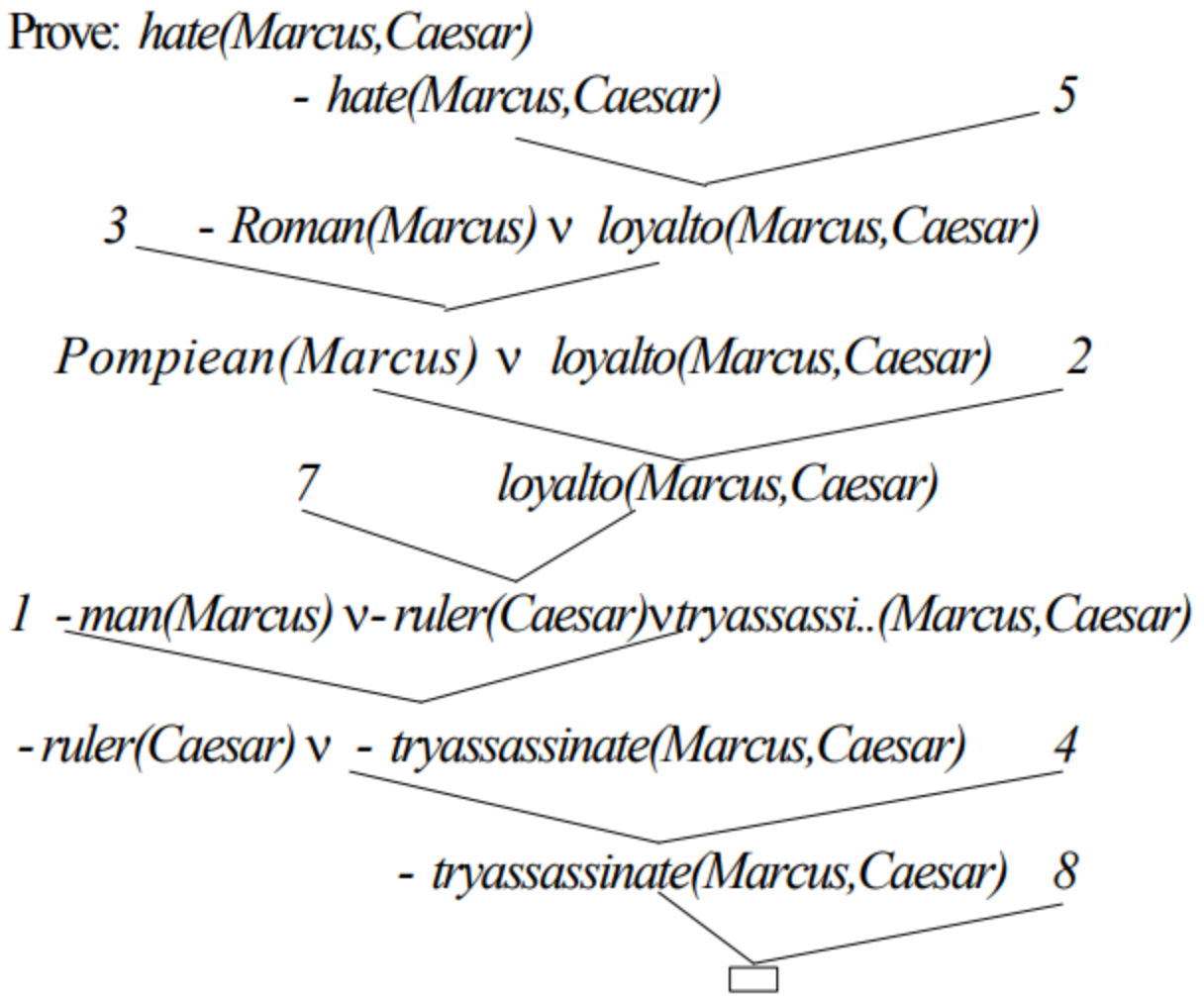
- Example

Given Axioms	Converted to Clause Form
P	P
(P $\wedge$ Q) -> R	not P $\vee$ not Q $\vee$ R
(S $\vee$ T) -> Q	not S $\vee$ Q
	not T $\vee$ Q
T	T



Resolution in Predicate Logic

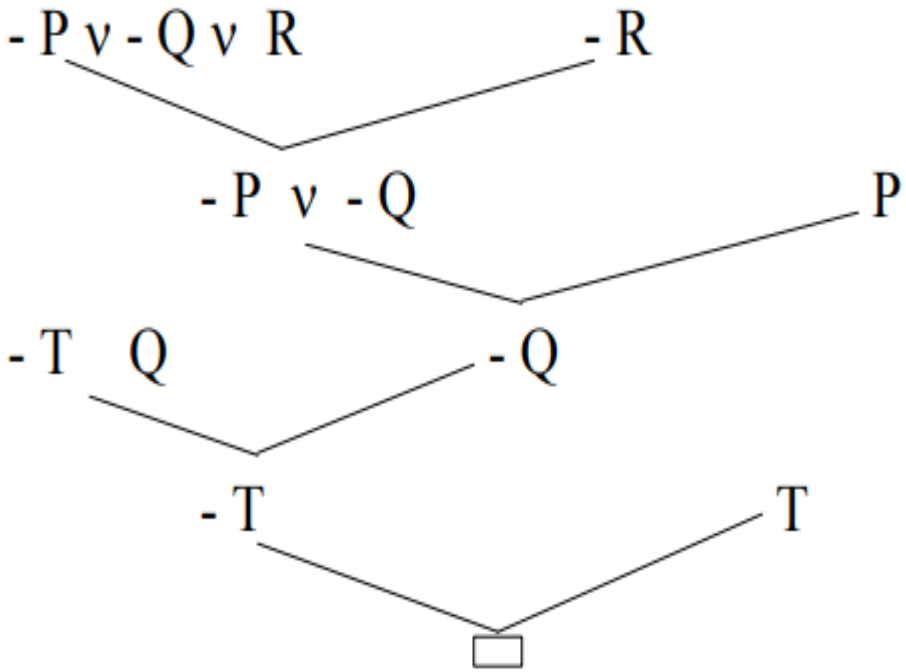
1. $man(Marcus)$
2. $Pompiean(Marcus)$
3. $\neg Pompiean(x1) \vee Roman(x1)$
4. $ruler(Caesar)$
5. $\neg Roman(x2) \vee loyalto(x2,Caesar) \vee hate(x2,Caesar)$
6. $loyal(x3,f(x3))$
7. $\neg man(x4) \vee \neg ruler(y1) \vee \neg tryassassinate(x4,y1) \vee loyalto(x4,y1)$
8. $tryassassinate(Marcus,Caesar)$



Resolution

- Example

Given Axioms	Converted to Clause Form
P	P
(P $\wedge$ Q) -> R	not P $\vee$ not Q $\vee$ R
(S $\vee$ T) -> Q	not S $\vee$ Q
	not T $\vee$ Q
T	T



Representing Instance and ISA Relationships

- Specific attributes instance and isa play an important role particularly in a useful form of reasoning called property inheritance.
- The predicates instance and isa explicitly captured the relationships they used to express, namely class membership and class inclusion.

1. Man(Marcus).
2. Pompeian(Marcus).
3. $\forall x$: Pompeian(x) $\rightarrow$ Roman(x).
4. ruler(Caesar).
5. $\forall x$: Roman(x) $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).

1. instance(Marcus, man).
2. instance(Marcus, Pompeian).
3. $\forall x$: instance(x, Pompeian) $\rightarrow$ instance(x, Roman).
4. instance(Caesar, ruler).
5. $\forall x$: instance(x, Roman). $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).

1. instance(Marcus, man).
2. instance(Marcus, Pompeian).
3. isa(Pompeian, Roman)
4. instance(Caesar, ruler).
5. $\forall x$: instance(x, Roman). $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar).
6. $\forall x$: $\forall y$: $\forall z$: instance(x, y) $\wedge$ isa(y, z) $\rightarrow$ instance(x, z).

Representing Instance and ISA Relationships

- The first part of the figure contains, class membership represented with unary predicates (such as Roman), each of which corresponds to a class.
- Asserting that P(x) is true is equivalent to asserting that x is an instance (or element) of P.
- The second part of the figure contains representations that use the instance predicate explicitly.
- The predicate instance is a binary one, whose first argument is an object and whose second argument is a class to which the object belongs.
- But these representations do not use an explicit isa predicate.
- Instead, subclass relationships, such as that between Pompeians and Romans, described as shown in sentence 3.

Representing Instance and ISA Relationships

- The implication rule states that if an object is an instance of the subclass Pompeian then it is an instance of the superclass Roman.
- Note that this rule is equivalent to the standard set-theoretic definition of the subclasssuperclass relationship.
- The third part contains representations that use both the instance and isa predicates explicitly.
- The use of the isa predicate simplifies the representation of sentence 3, but it requires that one additional axiom (shown here as number 6) be provided.

Computable Functions and Predicates

- To express simple facts, such as the following greater-than and less-than relationships:
- gt(1,O) It(0,1) gt(2,1) It(1,2) gt(3,2) It(2,3)
- It is often also useful to have computable functions as well as computable predicates.
- Thus we might want to be able to evaluate the truth of gt(2 + 3,1)
- To do so requires that we first compute the value of the plus function given the arguments 2 and 3, and then send the arguments 5 and 1 to gt.

Computable Functions and Predicates

- Consider the following set of facts, again involving Marcus:

1) Marcus was a man.

man(Marcus)

2) Marcus was a Pompeian.

Pompeian(Marcus)

3) Marcus was born in 40 A.D.

born(Marcus, 40)

4) All men are mortal.

x: man(x) → mortal(x)

5) All Pompeians died when the volcano erupted in 79 A.D.

erupted(volcano, 79) ∧ ∀ x : [Pompeian(x) → died(x, 79)]

6) No mortal lives longer than 150 years.

x: t1: At2: *mortal(x) born(x, t1) gt(t2 – t1,150) → died(x, t2)*

7) It is now 1991.

***now* = 1991**

Computable Functions and Predicates

- So, Now suppose we want to answer the question “Is Marcus alive?”
- The statements suggested here, there may be two ways of deducing an answer.
- Either we can show that Marcus is dead because he was killed by the volcano or we can show that he must be dead because he would otherwise be more than 150 years old, which we know is not possible.
- Also, As soon as we attempt to follow either of those paths rigorously, however, we discover, just as we did in the last example, that we need some additional knowledge. For example, our statements talk about dying, but they say nothing that relates to being alive, which is what the question is asking.

Computable Functions and Predicates

- So we add the following facts:

8) Alive means not dead.

x: t: [alive(x, t) → ¬ dead(x, t)] [¬ dead(x, t) → alive(x, t)]

9) If someone dies, then he is dead at all later times.

x: t1: At2: *died(x, t1) gt(t2, t1) → dead(x, t2)*

So, based on these facts, we can answer the question "Is Marcus alive?" by proving: ¬ ***alive(Marcus, now)***

Thank you.

Resolution Proof Example.

(a) Marcus was a man.

(b) Marcus was a Roman.

(c) All men are people.

(d) Caesar was a ruler.

(e) All Romans were either loyal to Caesar or hated him (or both).

(f) Everyone is loyal to someone.

(g) People only try to assassinate rulers they are not loyal to.

(h) Marcus tried to assassinate Caesar.

Convert to First order Logic.

(a) Marcus was a man.

(b) Marcus was a Roman.

(c) All men are people.

(d) Caesar was a ruler.

(e) All Romans were either loyal to Caesar or hated him (or both).

(f) Everyone is loyal to someone.

(g) People only try to assassinate rulers they are not loyal to.

(h) Marcus tried to assassinate Caesar.

(a) man(marcus)

Convert to First order Logic

(a) Marcus was a man.

(b) Marcus was a Roman.

(c) All men are people.

(d) Caesar was a ruler.

(e) All Romans were either loyal to Caesar or hated him (or both).

(f) Everyone is loyal to someone.

(g) People only try to assassinate rulers they are not loyal to.

(h) Marcus tried to assassinate Caesar.

(a) man(marcus)
(b) roman(marcus)

Convert to First order Logic

(a) Marcus was a man.

(b) Marcus was a Roman.

(c) All men are people.

(d) Caesar was a ruler.

(e) All Romans were either loyal to Caesar or hated him (or both).

(f) Everyone is loyal to someone.

(g) People only try to assassinate rulers they are not loyal to.

(h) Marcus tried to assassinate Caesar.

(a) man(marcus)

(b) roman(marcus)

(c) $\forall X.man(X) \rightarrow person(X)$

Convert to First order Logic

(a) Marcus was a man.

(b) Marcus was a Roman.

(c) All men are people.

(d) Caesar was a ruler.

(e) All Romans were either loyal to Caesar or hated him (or both).

(f) Everyone is loyal to someone.

(g) People only try to assassinate rulers they are not loyal to.

(h) Marcus tried to assassinate Caesar.

(a) man(marcus)
(b) roman(marcus)
(c) $\forall X.man(X) \rightarrow person(X)$
(d) ruler(caesar)

Convert to First order Logic

(a) Marcus was a man.

(b) Marcus was a Roman.

(c) All men are people.

(d) Caesar was a ruler.

(e) All Romans were either loyal to Caesar or hated him (or both).

(f) Everyone is loyal to someone.

(g) People only try to assassinate rulers they are not loyal to.

(h) Marcus tried to assassinate Caesar.

(a) $man(marcus)$

(b) $roman(marcus)$

(c) $\forall X.man(X) \rightarrow person(X)$

(d) $ruler(caesar)$

(e) $\forall X.roman(x) \rightarrow loyal(X,caesar) \vee hate(X,caesar)$

Convert to First order Logic

(a) Marcus was a man.

(b) Marcus was a Roman.

(c) All men are people.

(d) Caesar was a ruler.

(e) All Romans were either loyal to Caesar or hated him (or both).

(f) Everyone is loyal to someone.

(g) People only try to assassinate rulers they are not loyal to.

(h) Marcus tried to assassinate Caesar.

(a) $man(marcus)$

(b) $roman(marcus)$

(c) $\forall X.man(X) \rightarrow person(X)$

(d) $ruler(caesar)$

(e) $\forall X.roman(x) \rightarrow loyal(X,caesar) \vee hate(X,caesar)$

(f) $\forall X \exists Y.\ loyal(X,Y)$

Convert to First order Logic

(a) Marcus was a man.

(b) Marcus was a Roman.

(c) All men are people.

(d) Caesar was a ruler.

(e) All Romans were either loyal to Caesar or hated him (or both).

(f) Everyone is loyal to someone.

(g) People only try to assassinate rulers they are not loyal to.

(h) Marcus tried to assassinate Caesar.

(a) $man(marcus)$

(b) $roman(marcus)$

(c) $\forall X.\ man(X) \rightarrow person(X)$

(d) $ruler(caesar)$

(e) $\forall X.\ roman(x) \rightarrow loyal(X,caesar) \vee hate(X,caesar)$

(f) $\forall X \exists Y.\ loyal(X,Y)$

(g) $\forall X \forall Y.\ person(X) \wedge tryassasin(X,Y) \rightarrow \neg loyal(X,Y)$

Convert to First order Logic

(a) Marcus was a man.

(b) Marcus was a Roman.

(c) All men are people.

(d) Caesar was a ruler.

(e) All Romans were either loyal to Caesar or hated him (or both).

(f) Everyone is loyal to someone.

(g) People only try to assassinate rulers they are not loyal to.

(h) Marcus tried to assassinate Caesar.

(a) $man(marcus)$

(b) $roman(marcus)$

(c) $\forall X.\ man(X) \rightarrow person(X)$

(d) $ruler(caesar)$

(e) $\forall X.\ roman(x) \rightarrow loyal(X,caesar) \vee hate(X,caesar)$

(f) $\forall X \exists Y.\ loyal(X,Y)$

(g) $\forall X \forall Y.\ person(X) \wedge ruler(Y) \ tryassasin(X,Y) \rightarrow \neg loyal(X,Y)$

(h) $tryassasin(marcus,caesar)$

Convert to Clausal Form

(a) man(marcus)
(b) roman(marcus)
(c) $\forall$X. man(X) $\rightarrow$ person(X)
(d) ruler(caesar)
(e) $\forall$X. roman(x) $\rightarrow$ loyal(X,caesar) $\vee$ hate(X,caesar)
(f) $\forall$X$\exists$Y. loyal(X,Y)
(g) $\forall$X$\forall$Y. person(X) $\wedge$ ruler(Y) $\wedge$ tryassasin(X,Y) $\rightarrow$ $\neg$loyal(X,Y)
(h) tryassasin(marcus,caesar)

Convert to Clausal Form

(a) man(marcus)
(b) roman(marcus)
(c) $\forall$X. man(X) $\rightarrow$ person(X)
(d) ruler(caesar)
(e) $\forall$X. roman(x) $\rightarrow$ loyal(X,caesar) $\vee$ hate(X,caesar)
(f) $\forall$X$\exists$Y. loyal(X,Y)
(g) $\forall$X$\forall$Y. person(X) $\wedge$ ruler(Y)$\wedge$ tryassasin(X,Y) $\rightarrow$ $\neg$loyal(X,Y)
(h) tryassasin(marcus,caesar)

Convert to Clausal Form

(a) man(marcus)
(b) roman(marcus)
(c) $\forall X.\ man(X) \rightarrow person(X)$
    $(\neg man(X), person(X))$
(d) ruler(caesar)
(e) $\forall X.\ roman(x) \rightarrow loyal(X,caesar) \vee hate(X,caesar)$
(f) $\forall X \exists Y.\ loyal(X,Y)$
(g) $\forall X \forall Y.\ person(X) \wedge ruler(Y) \wedge tryassasin(X,Y) \rightarrow \neg loyal(X,Y)$
(h) tryassasin(marcus,caesar)

Convert to Clausal Form

(a) man(marcus)
(b) roman(marcus)
(c) $\forall X.\ man(X) \rightarrow person(X)$
    $(\neg man(X), person(X))$
(d) ruler(caesar)
(e) $\forall X.\ roman(x) \rightarrow loyal(X,caesar) \vee hate(X,caesar)$
(f) $\forall X \exists Y.\ loyal(X,Y)$
(g) $\forall X \forall Y.\ person(X) \wedge ruler(Y) \wedge tryassasin(X,Y) \rightarrow \neg loyal(X,Y)$
(h) tryassasin(marcus,caesar)

Convert to Clausal Form

(a) man(marcus)
(b) roman(marcus)
(c) $\forall$X. man(X) $\rightarrow$ person(X)
    ($\neg$man(X), person(X))
(d) ruler(caesar)
(e) $\forall$X. roman(X) $\rightarrow$ loyal(X,caesar) $\vee$ hate(X,caesar)
    ($\neg$roman(X), loyal(X,caesar), hate(X,caesar))
(f) $\forall$X$\exists$Y. loyal(X,Y)
(g) $\forall$X$\forall$Y. person(X) $\wedge$ ruler(Y)$\wedge$ tryassasin(X,Y) $\rightarrow$ $\neg$loyal(X,Y)
(h) tryassasin(marcus,caesar)

Convert to Clausal Form

(a) man(marcus)
(b) roman(marcus)
(c) $\forall$X. man(X) $\rightarrow$ person(X)
    ($\neg$man(X), person(X))
(d) ruler(caesar)
(e) $\forall$X. roman(X) $\rightarrow$ loyal(X,caesar) $\vee$ hate(X,caesar)
    ($\neg$roman(X), loyal(X,caesar), hate(X,caesar))
(f) $\forall$X$\exists$Y. loyal(X,Y)
    (loyal(X,f(X))
(g) $\forall$X$\forall$Y. person(X) $\wedge$ ruler(Y) $\wedge$ tryassasin(X,Y) $\rightarrow$ $\neg$loyal(X,Y)
(h) tryassasin(marcus,caesar)

Convert to Clausal Form

(a) man(marcus)
(b) roman(marcus)
(c) $\forall$X. man(X) $\rightarrow$ person(X)
    ($\neg$man(X), person(X))
(d) ruler(caesar)
(e) $\forall$X. roman(X) $\rightarrow$ loyal(X,caesar) $\vee$ hate(X,caesar)
    ($\neg$roman(X), loyal(X,caesar), hate(X,caesar))
(f) $\forall$X$\exists$Y. loyal(X,Y)
    (loyal(X,f(X))
(g) $\forall$X$\forall$Y. person(X) $\wedge$ ruler(Y) $\wedge$ tryassasin(X,Y) $\rightarrow$ $\neg$loyal(X,Y)
    ($\neg$person(X), $\neg$ruler(Y), $\neg$tryassasin(X,Y), $\neg$loyal(X,Y))
(h) tryassasin(marcus,caesar)

Convert to Clausal Form

(a) man(marcus)
(b) roman(marcus)
(c) $\forall X.\ man(X) \rightarrow person(X)$
    $(\neg man(X), person(X))$
(d) ruler(caesar)
(e) $\forall X.\ roman(X) \rightarrow loyal(X,caesar) \vee hate(X,caesar)$
    $(\neg roman(X), loyal(X,caesar), hate(X,caesar))$
(f) $\forall X \exists Y.\ loyal(X,Y)$
    $(loyal(X,f(X))$
(g) $\forall X \forall Y.\ person(X) \wedge ruler(Y) \wedge tryassasin(X,Y) \rightarrow \neg loyal(X,Y)$
    $(\neg person(X), \neg ruler(Y), \neg tryassasin(X,Y), \neg loyal(X,Y))$
(h) tryassasin(marcus,caesar)

Convert to Clausal Form

1. man(marcus)
2. roman(marcus)
3. (¬man(X), person(X))
4. ruler(caesar)
5. (¬roman(X), loyal(X,caesar), hate(X,caesar))
6. (loyal(X,f(X))
7. (¬person(X), ¬ruler(Y), ¬tryassasin(X,Y), ¬loyal(X,Y))
8. tryassasin(marcus,caesar)

Query

Who hated Caesar?

Query

Who hated Caesar?

In First Order logic.

1. $\exists X.\ hate(X,caesar)$

Query

Who hated Caesar?

In First Order logic.

1. $\exists X.\ hate(X,caesar)$

Negate!

1. $\forall X.\ \neg hate(X,caesar)$

Query

Who hated Caesar?

In First Order logic.

1. $\exists X.\ hate(X,caesar)$

Negate!

1. $\forall X.\ \neg hate(X,caesar)$

Clausal Form, with answer literal.

9. $(\neg hate(X,caesar),\ ans(X))$

Resolution Proof

1. man(marcus)
2. roman(marcus)
3. (¬man(X), person(X))
4. ruler(caesar)
5. (¬roman(X), loyal(X,caesar), hate(X,caesar))
6. (loyal(X,f(X))
7. (¬person(X), ¬ruler(Y), ¬tryassasin(X,Y), ¬loyal(X,Y))
8. tryassasin(marcus,caesar)
9. (¬hate(X,caesar), ans(X))
10. R[9,5c] (¬roman(X), loyal(X,caesar), ans(X))

Note: In general we have to be cautious about variable names. The X in clause 5 is NOT the same as the X in clause 9!

Resolution Proof

Note: In general we have to be cautious about variable names. The X in clause 5 is NOT the same as the X in clause 9!

1. (p(X), h(Y))
2. (¬p(X), q(Y))

3. R[1,2] (h(Y), q(Y))

This is incorrect, as now h and q seem to have the same variable. In fact the Y in 1 and the Y in 2 are different. The correct operation is to first rename the variables in one of the clauses so that the two clauses each have distinct variable names.

Resolution Proof

Note: In general we have to be cautious about variable names. The X in clause 5 is NOT the same as the X in clause 9!

1. ~~(p(X), h(Y))~~ (p(Z), h(W))
2. (¬p(X), q(Y))

3. R[1,2] {Z=X} (h(W), q(Y))

Note that 3 is now more general than (h(Y), q(Y)).

In our example if we applied this more correct rule we would still get the same answer…so I took a short cut.

Resolution Proof

1. man(marcus)
2. roman(marcus)
3. (¬man(X), person(X))
4. ruler(caesar)
5. (¬roman(X), loyal(X,caesar), hate(X,caesar))
6. (loyal(X,f(X))
7. (¬person(X), ¬ruler(Y), ¬tryassasin(X,Y), ¬loyal(X,Y))
8. tryassasin(marcus,caesar)
9. ¬hate(X,caesar)
10. R[9,5c] (¬roman(X), loyal(X,caesar), ans(X))

Resolution Proof

1. man(marcus)
2. roman(marcus)
3. (¬man(X), person(X))
4. ruler(caesar)
5. (¬roman(X), loyal(X,caesar), hate(X,caesar))
6. (loyal(X,f(X))
7. (¬person(X), ¬ruler(Y), ¬tryassasin(X,Y), ¬loyal(X,Y))
8. tryassasin(marcus,caesar)
9. ¬hate(X,caesar)
10. R[9,5c] (¬roman(X), loyal(X,caesar), ans(X))
11. R[10a,2] {X=marcus} (loyal(markus,caesar), ans(markus))

Resolution Proof

1. man(marcus)
2. roman(marcus)
3. (¬man(X), person(X))
4. ruler(caesar)
5. (¬roman(X), loyal(X,caesar), hate(X,caesar))
6. (loyal(X,f(X))
7. (¬person(X), ¬ruler(Y), ¬tryassasin(X,Y), ¬loyal(X,Y))
8. tryassasin(marcus,caesar)
9. ¬hate(X,caesar)
10. R[9,5c] (¬roman(X), loyal(X,caesar), ans(X))
11. R[10a,2] {X=marcus} (loyal(markus,caesar), ans(markus))
12. R[11,7c] {X=markus, Y=caesar} (¬person(markus), ¬ruler(caesar), ¬tryassasin(markus,caesar), ans(markus))

Resolution Proof

1. man(marcus)
2. roman(marcus)
3. (¬man(X), person(X))
4. ruler(caesar)
5. (¬roman(X), loyal(X,caesar), hate(X,caesar))
6. (loyal(X,f(X))
7. (¬person(X), ¬ruler(Y), ¬tryassasin(X,Y), ¬loyal(X,Y))
8. tryassasin(marcus,caesar)
9. ¬hate(X,caesar)
10. R[9,5c] (¬roman(X), loyal(X,caesar), ans(X))
11. R[10a,2] {X=marcus} (loyal(markus,caesar), ans(markus))
12. R[11,7c] {X=markus, Y=caesar} (¬person(markus), ¬ruler(caesar), ¬tryassasin(markus,caesar), ans(markus))
13. R[12a,3b] {X=markus} (¬man(markus), ¬ruler(caesar), ¬tryassasin(markus,caesar), ans(markus))

Resolution Proof

1. man(marcus)
2. roman(marcus)
3. (¬man(X), person(X))
4. ruler(caesar)
5. (¬roman(X), loyal(X,caesar), hate(X,caesar))
6. (loyal(X,f(X))
7. (¬person(X), ¬ruler(Y), ¬tryassasin(X,Y), ¬loyal(X,Y))
8. tryassasin(marcus,caesar)
9. ¬hate(X,caesar)
10. R[9,5c] (¬roman(X), loyal(X,caesar), ans(X))
11. R[10a,2] {X=marcus} (loyal(markus,caesar), ans(markus))
12. R[11,7c] {X=markus, Y=caesar} (¬person(markus), ¬ruler(caesar), ¬tryassasin(markus,caesar), ans(markus))
13. R[12a,3b] {X=markus} (¬man(markus), ¬ruler(caesar), ¬tryassasin(markus,caesar), ans(markus))
14. R[13a,1] (¬ruler(caesar), ¬tryassasin(markus,caesar), ans(markus))

Resolution Proof

1. man(marcus)
2. roman(marcus)
3. (¬man(X), person(X))
4. ruler(caesar)
5. (¬roman(X), loyal(X,caesar), hate(X,caesar))
6. (loyal(X,f(X))
7. (¬person(X), ¬ruler(Y), ¬tryassasin(X,Y), ¬loyal(X,Y))
8. tryassasin(marcus,caesar)
9. ¬hate(X,caesar)
10. R[9,5c] (¬roman(X), loyal(X,caesar), ans(X))
11. R[10a,2] {X=marcus} (loyal(markus,caesar), ans(markus))
12. R[11,7c] {X=markus, Y=caesar} (¬person(markus), ¬ruler(caesar), ¬tryassasin(markus,caesar), ans(markus))
13. R[12a,3b] {X=markus} (¬man(markus), ¬ruler(caesar), ¬tryassasin(markus,caesar), ans(markus))
14. R[13a,1] (¬ruler(caesar), ¬tryassasin(markus,caesar), ans(markus))
15. R[13a,2] (¬tryassasin(markus,caesar), ans(markus))

Resolution Proof

1. man(marcus)
2. roman(marcus)
3. (¬man(X), person(X))
4. ruler(caesar)
5. (¬roman(X), loyal(X,caesar), hate(X,caesar))
6. (loyal(X,f(X))
7. (¬person(X), ¬ruler(Y), ¬tryassasin(X,Y), ¬loyal(X,Y))
8. tryassasin(marcus,caesar)
9. ¬hate(X,caesar)
10. R[9,5c] (¬roman(X), loyal(X,caesar), ans(X))
11. R[10a,2] {X=marcus} (loyal(markus,caesar), ans(markus))
12. R[11,7c] {X=markus, Y=caesar} (¬person(markus), ¬ruler(caesar), ¬tryassasin(markus,caesar), ans(markus))
13. R[12a,3b] {X=markus} (¬man(markus), ¬ruler(caesar), ¬tryassasin(markus,caesar), ans(markus))
14. R[13a,1] (¬ruler(caesar), ¬tryassasin(markus,caesar), ans(markus))
15. R[13a,2] (¬tryassasin(markus,caesar), ans(markus))
16. R[15a,8] ans(markus)