Knowledge-Based Agent and Probabilistic reasoning in Artificial intelligence

Table of Contents

Knowledge-Based Agent in Artificial intelligence

Artificial Intelligence (AI) is one of the most rapidly developing fields in computer science, with knowledge-based agents being one of the most important components of AI. Knowledge-based agents are artificial intelligence systems that use knowledge to perform their tasks. They are composed of two main parts, a **knowledge base**, and **an inference system,** and they operate by making deductions, decisions, and conclusions based on the available knowledge.

Knowledge-Base

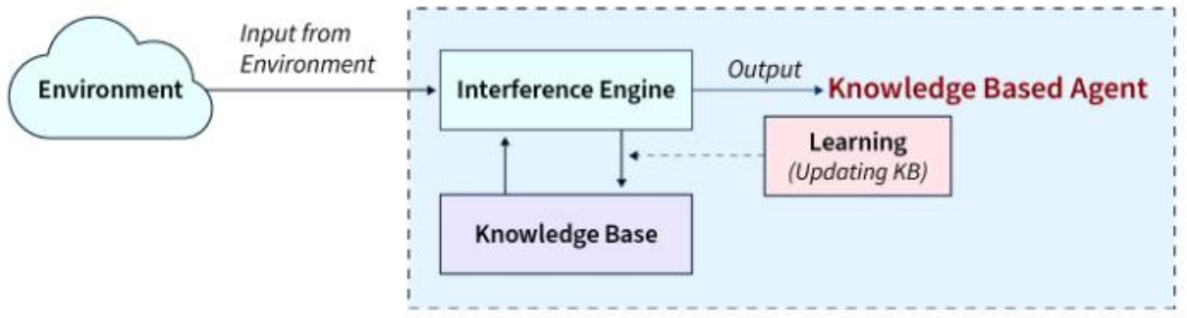
The knowledge base of a knowledge-based agent is a collection of knowledge that the agent uses to make decisions. The knowledge can be explicit, such as rules or facts, or implicit, such as relationships or patterns. The knowledge base is stored in a database or a knowledge representation system.

Inference System

The inference system of a knowledge-based agent is responsible for using the knowledge in the knowledge base to make decisions. The inference system uses reasoning methods, such as deduction, induction, and abduction, to infer new knowledge from existing knowledge.

The Architecture of Knowledge-Based Agent

The diagram below depicts a general architecture for a knowledge-based agent (KBA):



The KBA receives input from the environment through perception, which the inference engine processes. The inference engine communicates with the knowledge base (KB) to determine the appropriate action based on the knowledge stored in the KB. The learning element of the KBA regularly updates the KB by incorporating new knowledge.

Why Use a Knowledge Base?

A knowledge base is used in a knowledge-based agent to provide a structured way to store and retrieve information. The knowledge base allows the agent to make decisions based on available knowledge rather than relying on hard-coded rules. Additionally, the knowledge base can be updated as new information becomes available, making the agent more adaptable to changing situations.

Inference System

The inference system of a knowledge-based agent is responsible for using the knowledge in the knowledge base to make decisions. The inference system uses reasoning methods, such as deduction, induction, and abduction, to infer new knowledge from existing knowledge. Generating new facts by the inference system allows the agent to update its knowledge base. The inference system operates primarily through two rules, which are referred to as forward chaining and backward chaining.

Operations Performed by KBA

There are three main operations that a knowledge-based agent (KBA) performs to demonstrate intelligent behavior. The first operation is called **TELL**, where the KBA informs the knowledge base about the information it has perceived from the environment. The second operation is called **ASK**, where the KBA requests the knowledge base to suggest appropriate action. The third operation is **PERFORM**, where the KBA executes the selected action.

The Structure Outline of a Generic Knowledge-Based Agents Program

Below is the structure outline of a generic knowledge-based agents program:

```
function KB-AGENT(percept):
persistent: KB, a knowledge base
        t, a counter, initially 0, indicating the time
TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
Action = ASK(KB, MAKE-ACTION-QUERY(t))
TELL(KB, MAKE-ACTION-SENTENCE(action, t))
 t = t + 1
 return action
```

The knowledge-based agent receives a percept and produces an action as output. The agent is equipped with a knowledge base (KB) with background knowledge of the real world. Additionally, the agent has a time counter initially set to zero, indicating the time elapsed during the process.

Each time the function is executed, the agent performs three operations:

Firstly, it reports to the KB what it has perceived. Secondly, it asks the KB what action it should take.

Thirdly, it reports to the KB which action it has chosen.

The **MAKE-PERCEPT-SENTENCE** generates a sentence that indicates that the agent perceived the given percept at the given time.

The **MAKE-ACTION-QUERY** generates a sentence that asks which action should be taken at the current time.

The **MAKE-ACTION-SENTENCE** generates a sentence that asserts that the chosen action has been executed.

Various Levels of Knowledge-Based Agent

Knowledge-based agents can be classified into three levels:

Knowledge Level

The knowledge level is the highest level of abstraction in a knowledge-based agent. It describes what the agent knows and how it uses it to perform tasks. The knowledge level concerns the representation and organization of knowledge rather than the implementation details.

Logical Level

The logical level is the intermediate level of abstraction in a knowledge-based agent. It describes how the knowledge is represented and manipulated by the inference engine. The logical story concerns the formal logic used to represent knowledge and make inferences.

Implementation Level

The implementation level is the lowest level of abstraction in a knowledge-based agent. It describes how the knowledge and inference engine is implemented using a programming language. The implementation level is concerned with the details of the programming language and the algorithms used to implement the knowledge and inference engine.

Approaches to Designing a Knowledge-Based Agent

There are two main approaches to designing a knowledge-based agent:

Declarative Approach

The declarative approach to designing a knowledge-based agent focuses on representing knowledge in a declarative form, such as rules or facts. The knowledge is represented independently of the algorithms used to manipulate it.

Procedural Approach

The procedural approach to designing a knowledge-based agent focuses on representing knowledge in a procedural form, such as a sequence of instructions. The knowledge is represented in terms of the algorithms used to manipulate it. The inference engine uses these algorithms to make inferences from the knowledge.

Knowledge representation

Knowledge representation is a crucial element of Artificial Intelligence. It is believed that an intelligent system needs to have an explicit representation of its knowledge to reason and make decisions.

Knowledge representation provides a framework for representing, organizing, and manipulating knowledge that can be used to solve complex problems, make decisions, and learn from data.

For example, when you see a hot tea cup, a signal immediately comes from your brain cautioning you against picking it up. If we were to make AI more sophisticated(or humanist), we would be required to feed them with more and often complex information about our world to perform the complex task, which leads to the concept of Knowledge Representation in Artificial Intelligence.

There are various approaches to knowledge representation in AI, including:

- **Logical representation:** This involves representing knowledge in a symbolic logic or rule-based system, which uses formal languages to express and infer new knowledge.
- **Semantic networks:** This involves representing knowledge through nodes and links, where nodes represent concepts or objects, and links represent their relationships.
- **Frames:** This approach involves representing knowledge in the form of structures called frames, which capture the properties and attributes of objects or concepts and the relationships between them.
- **Ontologies:** This involves representing knowledge in the form of a formal, explicit specification of the concepts, properties, and relationships between them within a particular domain.
- **Neural networks:** This involves representing knowledge in the form of patterns or connections between nodes in a network, which can be used to learn and infer new knowledge from data.

The Different Kinds of Knowledge: What to Represent

- **Object:** The AI needs to know all the facts about the objects in our world domain. E.g., A keyboard has keys, a guitar has strings, etc.
- **Events:** The actions which occur in our world are called events.
- **Performance**: It describes a behavior involving knowledge about how to do things.
- **Meta-knowledge:** The knowledge about what we know is called meta-knowledge.
- **Facts:** The things in the real world that are known and proven true.
- **Knowledge Base:** A knowledge base in artificial intelligence aims to capture human expert knowledge to support decision-making, problem-solving, and more.

Types of Knowledge in AI

In AI, various types of knowledge` are used for different purposes. Here are some of the main types of knowledge in AI:

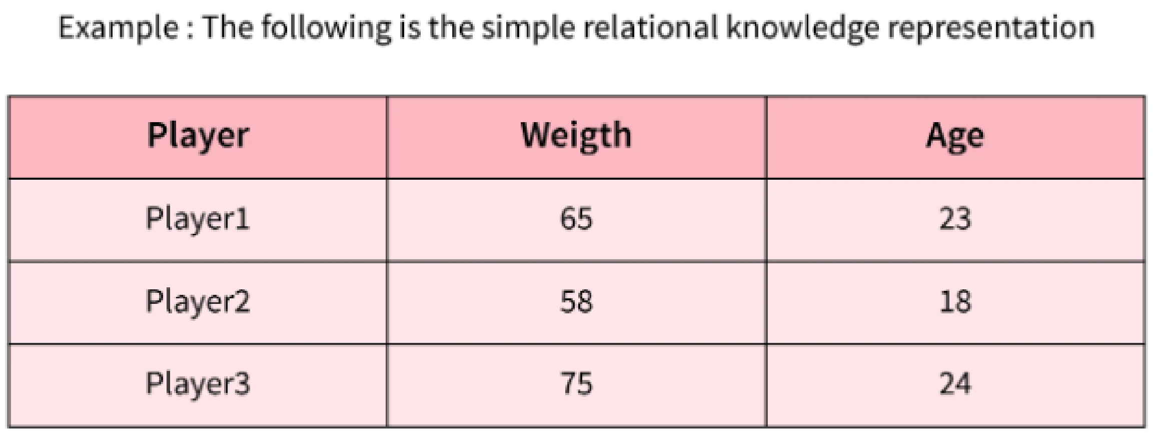
- **Declarative Knowledge:** This knowledge can be expressed in a declarative form, such as facts, rules, or propositions. It is also called descriptive knowledge and is expressed in declarative sentences. It is often represented using logic-based representations such as knowledge graphs or ontologies. Example: The capital of France is Paris. This statement represents declarative knowledge because it is a fact that can be explicitly stated and written down. It is not based on personal experience or practical skills, but rather on an established piece of information that can be easily communicated to others.
- **Procedural Knowledge:** This knowledge is used to perform specific tasks or actions and is often represented using algorithms or programming languages. It is responsible for knowing how to do something. It includes rules, strategies, procedures, agendas, etc. Example: How to change a flat tire on a car, including the steps of loosening the lug nuts, jacking up the car, removing the tire, and replacing it with a spare. This is a practical skill that involves specific techniques and steps that must be followed to successfully change a tire.
- **Meta-knowledge:** This is knowledge about knowledge and is often used to reason about and improve the performance of AI systems. Example: To remember new information, it is helpful to use strategies such as repetition, visualization, and elaboration. This statement represents metaknowledge because it is knowledge about how to learn and remember new information, rather than knowledge about a specific fact or concept. It acknowledges that some specific techniques and strategies can be used to enhance memory and learning, and encourages the use of these techniques to improve learning outcomes.
- **Heuristic Knowledge:** Heuristics are based on past experiences or domain knowledge and are often used in decision-making processes to guide an AI system toward a solution. Heuristic knowledge is a type of knowledge in AI that refers to rules of thumb or strategies that are used to solve problems quickly and efficiently, but only sometimes optimally. Heuristics are often used when there is too much complexity or uncertainty in a problem to use an exact algorithm or solution. Example: When packing for a trip, it is helpful to make a list of essential items, pack versatile clothing items that can be mixed and matched, and leave room in the suitcase for any souvenirs or purchases. This statement represents heuristic knowledge because it is a practical set of rules of thumb that can be used to guide decision-making in a specific situation (packing for a trip).
- **Structural Knowledge:** This is knowledge about the structure of a problem or system and is often used to help AI systems decompose complex problems into simpler sub-problems that can be solved more easily. It is the basic knowledge of problem-solving. It also describes relationships between concepts such as kind of, part of, and grouping of something. Example: In the field of biology, living organisms can be classified into different taxonomic groups based on shared characteristics. These taxonomic groups include domains, kingdoms, phyla, classes, orders, families, genera, and species. This statement represents structural knowledge because it describes the hierarchical structure of the taxonomic classification system used in biology. It acknowledges that there are specific levels of organization within this system and that each level has its unique characteristics and relationships to other levels.

Approaches to Knowledge Representation

Simple Relational Knowledge

This type of knowledge uses relational methods to store facts. It is one of the simplest types of knowledge representation. The facts are systematically set out in terms of rows and columns. This type of knowledge representation is used in database systems where the relationship between different entities is represented. There is a low opportunity for inference.

Change the design and the numbers

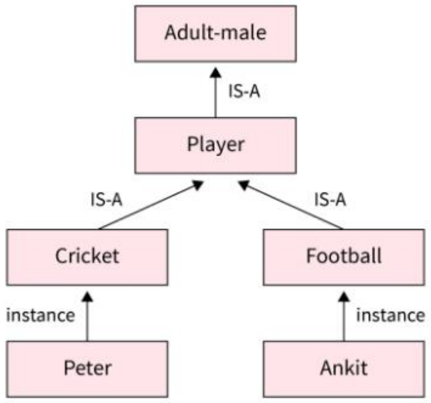


Example : The following is the simple relational knowledge representation

Player	Weigth	Age
Player1	65	23
Player2	58	18
Player3	75	24

Inheritable Knowledge

Inheritable knowledge in AI refers to knowledge acquired by an AI system through learning and can be transferred or inherited by other AI systems. This knowledge can include models, rules, or other forms of knowledge that an AI system learns through training or experience. In this approach, all data must be stored in a hierarchy of classes. Boxed nodes are used to represent objects and their values. We use Arrows that point from objects to their values. Rather than starting from scratch, an AI system can inherit knowledge from other systems, allowing it to learn faster and avoid repeating mistakes that have already been made. Inheritable knowledge also allows for knowledge transfer across domains, allowing an AI system to apply knowledge learned in one domain to another.



Inferential Knowledge

Inferential knowledge refers to the ability to draw logical conclusions or make predictions based on available data or information. In artificial intelligence, inferential knowledge is often used in machine learning algorithms, where models are trained on large amounts of data and then used to make predictions or decisions about new data. For example, in image recognition, a machine learning model can be trained on a large dataset of labeled images and then used to predict the contents of new images that it has never seen before. The model can draw inferences based on the patterns it has learned from the training data. It represents knowledge in the form of formal logic.

Example:

Statement 1: Alex is a footballer.

Statement 2: All footballers are athletes.

Then it can be represented as; Footballer(Alex) ∀x = Footballer (x) -> Athelete (x)s

Procedural Knowledge:

In artificial intelligence, procedural knowledge refers to the knowledge or instructions required to perform a specific task or solve a problem. This knowledge is often represented in algorithms or rules dictating how a machine processes data or performs tasks. For example, in natural language processing, procedural knowledge might involve the steps required to analyze and understand the meaning of a sentence. This could include tasks such as identifying the parts of speech in the sentence, identifying relationships between different words, and determining the overall structure and meaning of the sentence.

One of the most important rules used is the If-then rule. This knowledge allows us to use various coding languages such as LISP and Prolog. Procedural knowledge is an important aspect of artificial intelligence, as it allows machines to perform complex tasks and make decisions based on specific instructions.

Techniques of Knowledge Representation in AI

Logical Representation

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

Syntax: Syntaxes are the rules which decide how we can construct legal sentences in the logic. It determines which symbol we can use in knowledge representation. How to write those symbols.

Semantics: Semantics are the rules by which we can interpret the sentence in the logic. Semantic also involves assigning a meaning to each sentence.

Logical representation can be categorized into mainly two logics:

a) Propositional Logics.
b) Predicate logics.

Semantic Network Representation

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

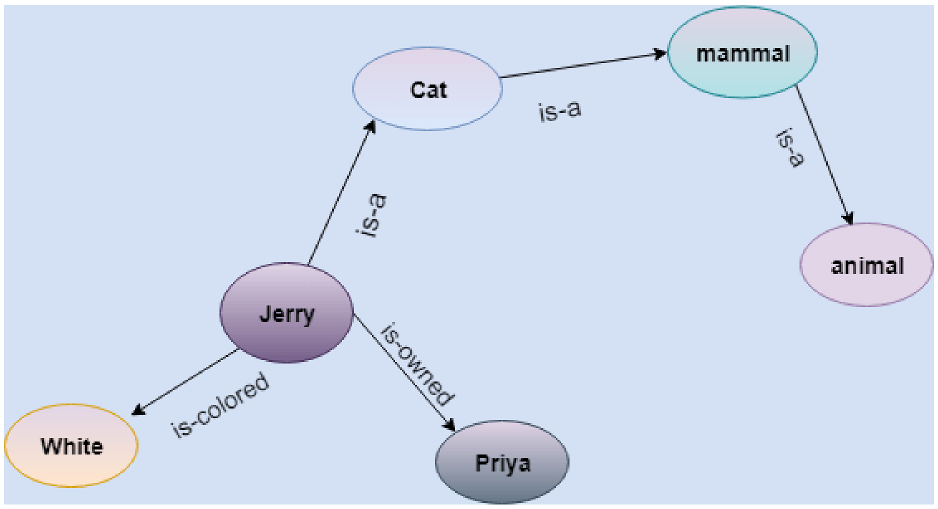
This representation consist of mainly two types of relations:

- IS-A relation (Inheritance)
- Kind-of-relation

Example: Following are some statements which we need to represent in the form of nodes and arcs.

Statements:

a) Jerry is a cat.
b) Jerry is a mammal
c) Jerry is owned by Priya.
d) Jerry is brown colored.
e) All Mammals are animal.



Frame Representation

A frame is a record like structure which consists of a collection of attributes and its values to describe an entity in the world. Frames are the AI data structure which divides knowledge into substructures by representing stereotypes situations. It consists of a collection of slots and slot values. These slots may be of any type and sizes. Slots have names and values which are called facets.

Facets: The various aspects of a slot is known as Facets. Facets are features of frames which enable us to put constraints on the frames. Example: IF-NEEDED facts are called when data of any particular slot is needed. A frame may consist of any number of slots, and a slot may include any number of facets and facets may have any number of values. A frame is also known as slot-filter knowledge representation in artificial intelligence.

Frames are derived from semantic networks and later evolved into our modern-day classes and objects. A single frame is not much useful. Frames system consist of a collection of frames which are connected. In the frame, knowledge about an object or event can be stored together in the knowledge base. The frame is a type of technology which is widely used in various applications including Natural language processing and machine visions.

Example:

Let's take an example of a frame for a book

Slots	Filters
Title	Artificial Intelligence
Genre	Computer Science
Author	Peter Norvig
Edition	Third Edition
Year	1996
Page	1152

Production Rules

Production rules system consist of (condition, action) pairs which mean, "If condition then action". It has mainly three parts:

- The set of production rules
- Working Memory
- The recognize-act-cycle

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

Example:

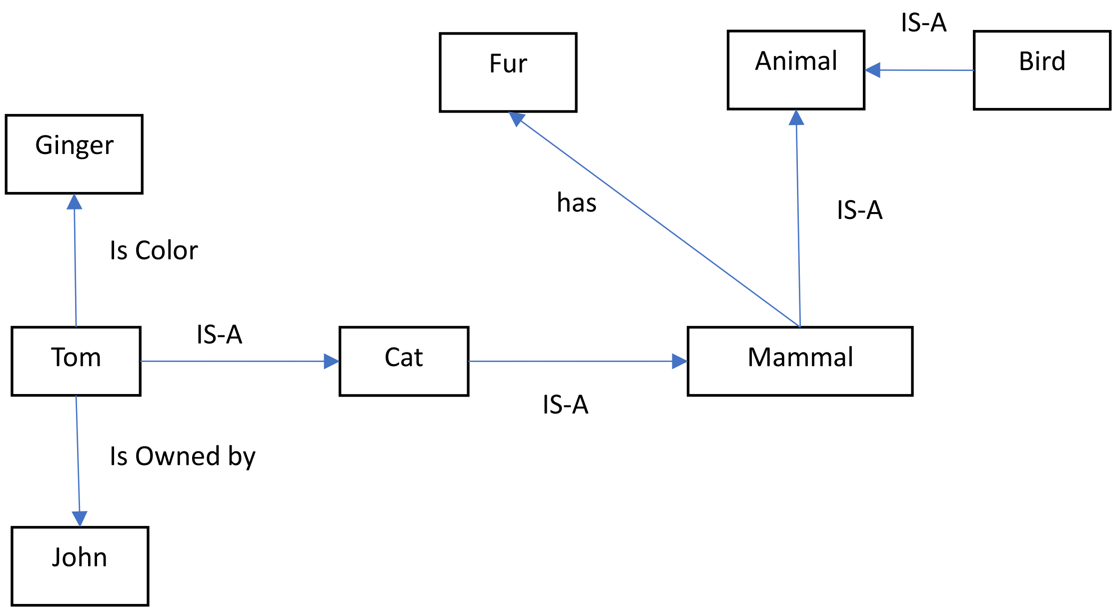
- IF (at bus stop AND bus arrives) THEN action (get into the bus)
- IF (on the bus AND paid AND empty seat) THEN action (sit down).
- IF (on bus AND unpaid) THEN action (pay charges).
- IF (bus arrives at destination) THEN action (get down from the bus).

Knowledge Representation in AI (Exam) Sample Problems:

Problem 1: Draw the semantic network that represents the data given below:

Mammals have fur. All mammals are animals. A bird is an animal. A cat is a mammal. Tom is a cat. Tom is owned by John. Tom is ginger in color.

Solution:



Problem 2: Find whether the meaning of the statement is true or false. “If the earth moves round the sun or the sun moves round the earth, then Copernicus might be a mathematician but was not an astronomer.”

Solution:

P = "The Earth moves round the Sun." = T

Q = "The Sun moves round the Earth." = T

M = "Copernicus is a mathematician." = T

A = "Copernicus is an astronomer." = T

$(T \lor T) \rightarrow (T \land \neg T)$

$= T \rightarrow (T \land F)$

$= T \rightarrow F$

$= F$

So the meaning of the statement is False

What is an Inference?

Inference in AI refers to the process of reasoning and making decisions based on available information or data. It involves deriving new knowledge or conclusions from existing knowledge or data. In other words, it is the process of going beyond the information provided to make predictions or draw conclusions based on that information.

In AI, inference can be categorized into two types: deductive inference and inductive inference. Deductive inference involves reasoning from general principles to specific conclusions, while inductive inference involves inferring general principles or rules based on specific observations or data.

Inference is a crucial process in AI and plays a vital role in various applications, including natural language processing, computer vision, robotics, and expert systems. For example, in natural language processing, inference is used to understand the meaning of a sentence based on the context and previous knowledge. In computer vision, inference is used to recognize objects in an image based on patterns and features. In robotics, inference is used to plan and execute actions based on the perception of the environment.

Different Types of Inference Rules in AI

Inference rules in AI are used to make logical deductions from given premises. Here are the different types of inference rules in AI:

Modus Ponens

It is a deductive inference rule in which if A implies B and A is true, then B must also be true. It is also known as affirming the antecedent. For example, "If it's raining, then the ground is wet" (A implies B), "It's raining" (A is true), therefore "The ground is wet" (B is true).

Symbolic Notation:

$(P \rightarrow Q), P \vdash Q$

Explanation:

The symbol "→" means "implies" or "if...then". So

$(P \rightarrow Q)$ means "if P, then Q". The symbol "⊢" means "entails" or "leads to", and is used to indicate that Q can be deduced from the premises $(P \rightarrow Q)$ and P.

Modus Tollens

It is a deductive inference rule in which if A implies B and B is false, then A must also be false. It is also known as denying the consequent. For example, "If it's raining, then the ground is wet" (A implies B), "The ground is not wet" (B is false), therefore "It's not raining" (A is false).

Symbolic Notation: $(P \rightarrow Q), \neg Q \vdash \neg P$

Explanation:

The symbol "¬" means "not", so $\neg Q$ means "it is not the case that Q". The symbol "⊢" has the same meaning as in Modus Ponens. This rule can be read as "If P implies Q and Q is false, then P is false".

Hypothetical Syllogism

It is a deductive inference rule that allows us to conclude from two conditional statements. If A implies B and B implies C, then A implies C. For example, "If it's raining, then the ground is wet" (A implies B), "If the ground is wet, then the grass will be green" (B implies C), therefore "If it's raining, then the grass will be green" (A implies C).

Symbolic Notation: (P → Q), (Q → R) ⊢ (P → R)

Explanation:

This rule says that if P implies Q, and Q implies R, then P implies R. The symbol "→" means "implies", and the symbol "⊢" means "entails".

Disjunctive Syllogism

It is a deductive inference rule that allows us to conclude a disjunctive statement. If it's either A or B and A is false, then B must be true.

For example, "It's either raining or snowing" (A or B), "It's not raining" (A is false), therefore "It's snowing" (B is true).

Symbolic Notation: (P ∨ Q), ¬P ⊢ Q

Explanation:

The symbol "∨" means " or", so (P ∨ Q) means "either P or Q (or both) is true". The symbol "¬" means "not". This rule can be read as "If either P or Q is true, and P is false, then Q is true".

Addition

It is a deductive inference rule that allows us to add a statement to conjunction (A conjunction implies that both statements are true). If A is true, then A or B is also true. For example, "It's raining" (A is true), therefore "It's either raining or snowing" (A or B is true).

Symbolic Notation: P ⊢ (P ∨ Q)

Explanation:

This rule says that if P is true, then P or Q is true. The symbol "∨" means " or".

Simplification

It is a deductive inference rule that allows us to derive a simpler statement from conjunction. If A and B are true, then A is true. For example, "It's raining and the ground is wet" (A and B are true), therefore "It's raining" (A is true).

Symbolic Notation: (P ∧ Q) ⊢ P

Explanation:

The symbol "∧" means " and", so (P ∧ Q) means "both P and Q are true". This rule can be read as "If both P and Q are true, then P is true".

Resolution

It is a deductive inference rule that allows us to resolve a disjunction(disjunction implies that at least one statement is true). If A implies B and C implies not B, then A implies not C. For example, "If it's raining, then the ground is wet" (A implies B), "If the ground is not wet, then it's not raining" (C implies not B), therefore "If it's raining, then the ground is not dry" (A implies not C).

Symbolic Notation: $(P \vee Q), (\neg P \vee R) \vdash (Q \vee R)$

Explanation:

This rule can be read as "If either P or Q is true, and not-P or R is true, then Q or R is true". The symbol "¬" means "not", and the symbol "∨" means " or".

Wumpus World in AI

The Wumpus World in AI is a classic problem demonstrating various ideas such as search algorithms, planning, and decision-making. The wumpus world in AI is a straightforward environment in which an agent (a computer program or a robot) must traverse a grid world filled with obstacles, hazards, and dangerous wumpus. Wumpus is a fictional character that kills the player in the game. The agent must travel the globe for a safe route to the treasure without falling into pits or being killed by the wumpus.

Properties of the Wumpus World

- Partially observable: The Wumpus world in AI is partially observable because the agent can only sense the immediate surroundings, such as an adjacent room.
- Deterministic: It is deterministic because the result and end of the world are already known.
- Sequential: It is sequential because the order is essential.
- Static: It is motionless because Wumpus and Pits are not moving.
- Discrete: The surroundings are distinct.
- One agent: The environment is a single agent because we only have one agent, and Wumpus is not regarded as an agent.

PEAS Description of Wumpus World

To build an intelligent agent for the Wumpus World, we must first define the problem's Performance, Environment, Actuators, and Sensors (PEAS).

Performance:

- +1000 bonus points if the agent returns from the tunnel with the gold.
- Being eaten by the wumpus or plummeting into the pit results in a -1000 point penalty.
- Each move is worth -1, and using an arrow is worth -10.
- The game is over if either agent dies or exits the tunnel.

Environment:

- A four-by-four grid of chambers.
- The operative begins in room square [1, 1], facing the right.
- Wumpus and gold locations are selected randomly except for the first square [1,1].
- Except for the first square, each square in the tunnel has a 0.2 chance of being a pit.

Actuators: They are the actions that the agent can take to interact with the world. The worker in Wumpus World in AI can carry out the following tasks:

- Left turn
- Right turn
- Move forward
- Grab
- Release
- Shoot

Sensors: They are how the agent senses its surroundings. The agent's instruments in the Wumpus World provide the following information:

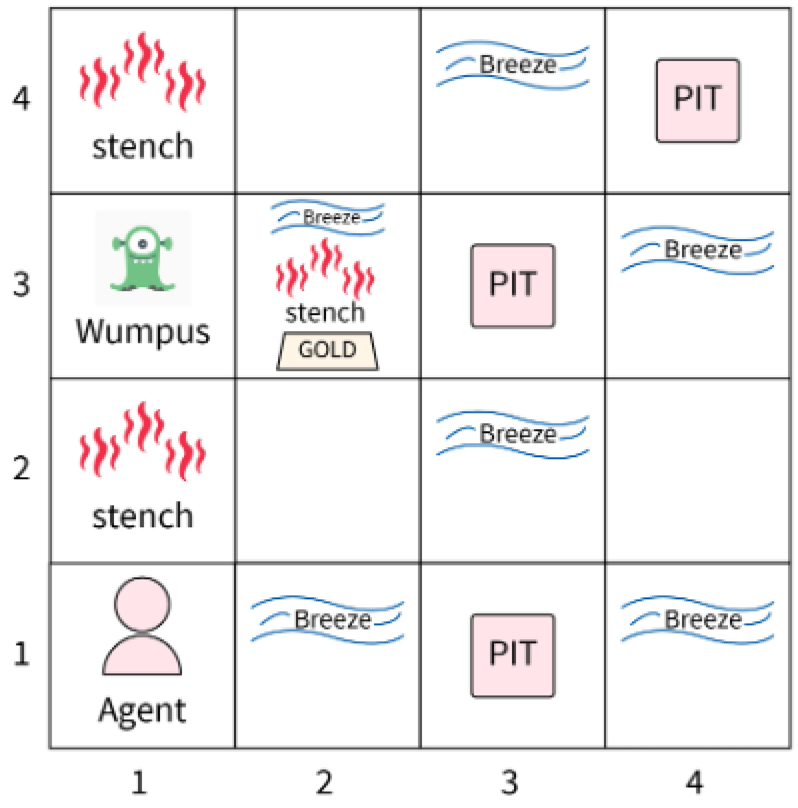
- If the agent is in the chamber next to the wumpus, he will notice the stench. (Not diagonally).
- If the agent is in the room immediately adjacent to the pit, he will notice a breeze.
- The agent will notice the glitter in the chamber with the gold.
- The agent will notice the bump if he runs into a wall.
- When the Wumpus is shot, it lets out a horrifying scream that can be heard throughout the tunnel.
- These perceptions can be represented as a five-element list with distinct indicators for each sensor.
- For example, if an agent detects stench and breeze but not glitter, bump, or scream, it can be depicted as [Stench, Breeze, None, None].

Wumpus World Cave Problem

The Wumpus world in AI is a cave with four chambers linked by passageways. So there are a total of 16 chambers that are linked to one another. We now have a knowledge-based agent who will advance in this universe. The cave has a chamber with a beast named Wumpus, who eats anyone who enters it. The agent can shoot the wumpus, but the agent only has one projectile. Some pit rooms in the Wumpus world in AI are bottomless, and if the agent falls into one of them, he will be stuck there eternally. The exciting aspect of this cave is discovering a heap of gold in one of its rooms. So the agent's objective is to locate the gold and climb out of the cave without being eaten by wumpus or falling into Pits. The agent will be rewarded if he returns with gold, but he will be punished if he is eaten by wumpus or slips into the pit.

Some elements can assist the agent in navigating the tunnel. These elements are listed below:

- The rooms adjacent to the Wumpus chamber are stinky, so there will be a stench.
- The room closest to the PITs has a breeze, so if the agent gets close to the PIT, he will notice the breeze.
- Glitter will be present in the chamber if the room contains gold.
- If the agent confronts the wumpus, it can be killed, and the wumpus will scream horribly, which can be heard throughout the cave.

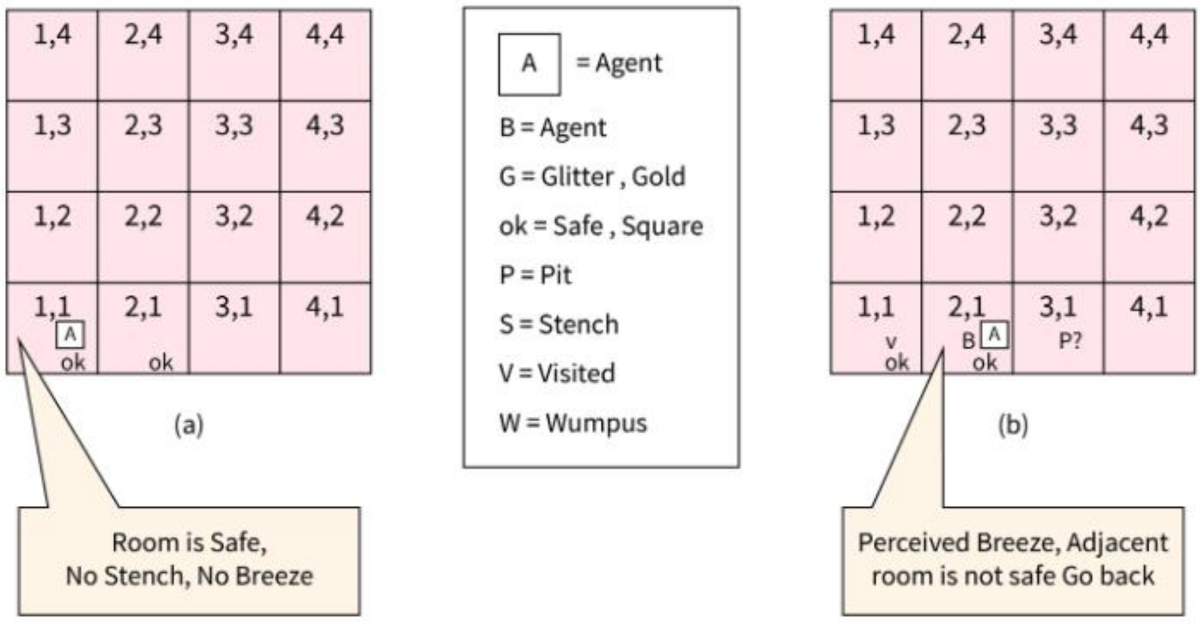


Exploring the Wumpus World

We will now explore the Wumpus world in AI and use logical reasoning to determine how the agent will reach its objective.

Agent's first step: Initially, the agent is in the first room or on the square [1,1], and we already know that this room is safe for the agent, so we will add the symbol OK to the below diagram (a) to indicate that room is safe. Then, the agent is represented by symbol A, the breeze by symbol B, the glitter or gold by symbol G, the visited chamber by symbol V, the pits by symbol P, and the wumpus by symbol W.

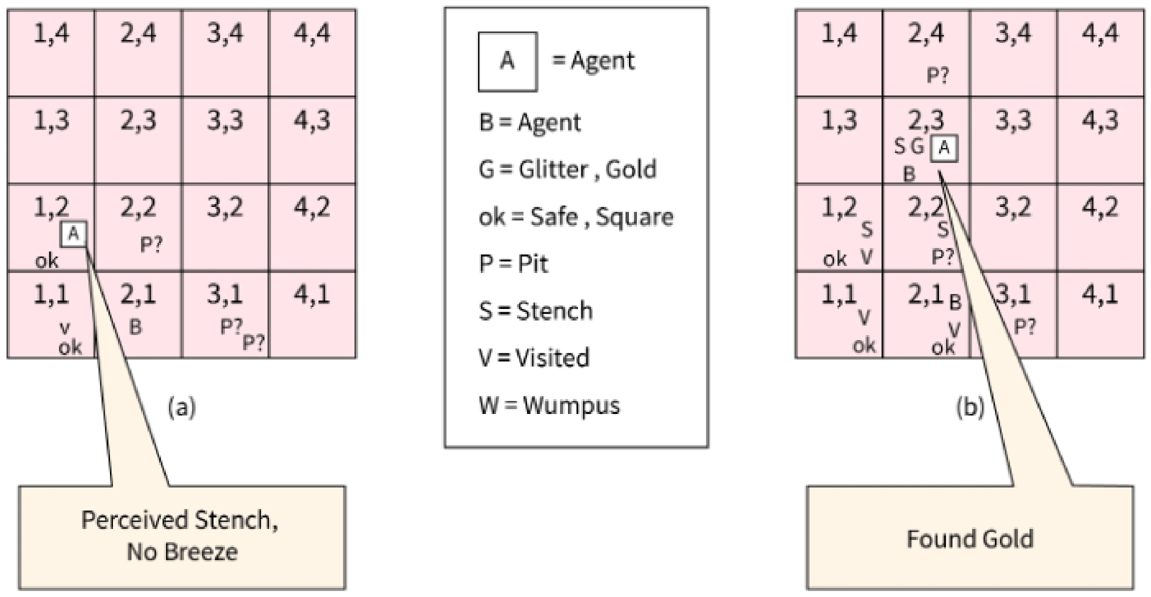
The agent does not detect any breeze or Stench in Room [1,1], implying that the neighboring squares are also fine.



Agent's second step: Now that the agent has to proceed forward, it will either go to [1, 2] or [2, 1]. Assume the agent moves to room [2, 1]. The agent detects a breeze in this chamber, indicating that the pit is nearby. The pit can be in [3, 1] or [2, 2], so we'll put the symbol P? to indicate whether or not this is a Pit room.

Now, the agent will pause and reflect before making any bad moves. Finally, the agent will return to the [1, 1] chamber. The agent visits rooms [1,1] and [2,1], so we will use V to symbolize the visited squares.

Agent's third step: At the third stage, the agent will proceed to room [1,2], which is fine. The agent detects a stench in the area [1,2], indicating the presence of a Wumpus nearby. But, according to the game's regulations, wumpus cannot be in the room [1,1] nor in [2,2]. (Agent had not detected any stench when he was at [2,1]). As a result, the agent deduces that the wumpus is in room [1,3], and in the present state, there is no breeze, implying that there is no Pit and no Wumpus in [2,2]. So it's safe, and we'll label it OK, and the agent will move in further [2,2].



Agent's fourth move: Because there is no stench or breeze in room [2,2], let us assume the agent chooses to relocate to [2,3]. The agent detects glitter in the room [2,3], so it should take the gold and climb out of the cave.

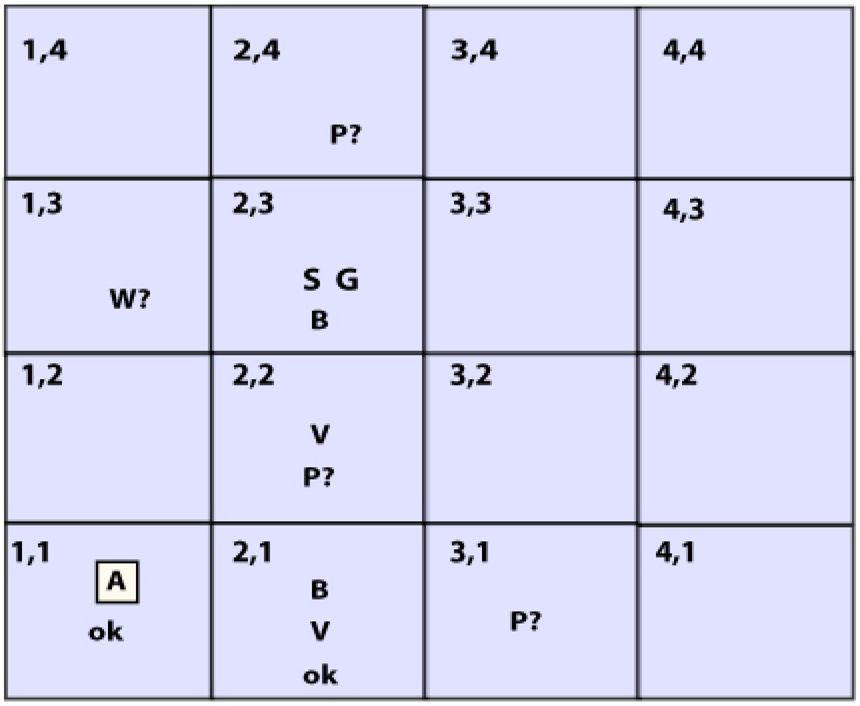
Applications of Wumpus World in AI

The Wumpus World in AI is a classic problem with multiple uses, including:

- Developing intelligent agents: The Wumpus World in AI is an excellent platform for creating intelligent agents capable of navigating complicated environments, reasoning in uncertainty, and planning actions.
- Testing AI algorithms: Wumpus World is a benchmark issue for testing and comparing various AI algorithms, such as search, planning, and reinforcement learning.
- Education and training: Because it is simple to use and offers hands-on experience, the Wumpus World in AI is a popular tool for teaching AI concepts and algorithms to students.
- Game Development: Wumpus World can motivate developers to create challenging and engaging games requiring strategic thinking and problem-solving.
- Robotics: The Wumpus World can be used as a testing and development setting for robotics algorithms such as pathfinding and mapping.

Knowledge-base for Wumpus world Problem (Exam Problem)

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



Atomic proposition variable for Wumpus world:

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

Some Propositional Rules for the wumpus world:

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

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

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

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

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

Solution:

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

Apply Modus Ponens with ¬S11 and R1:

We will firstly apply MP rule with R1 which is $\neg S_{11} \rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$, and $\boldsymbol{\neg S_{11}}$ which will give the output $\neg W_{11} \wedge W_{12} \wedge W_{12}$.

$$\begin{tikzcd}
\boxed{\neg S_{11} \Rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}} \arrow[dr] & & \boxed{\neg S_{11}} \arrow[dl] \\
& \boxed{\neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}} &
\end{tikzcd}$$

Apply And-Elimination Rule:

After applying And-elimination rule to $\neg W_{11} \land \neg W_{12} \land \neg W_{21}$, we will get three statements:
$\boldsymbol{\neg W_{11}, \neg W_{12},}$ **and** $\boldsymbol{\neg W_{21}}$.

Apply Modus Ponens to $\boldsymbol{\neg S_{21}}$, and R2:

Now we will apply Modus Ponens to $\neg S_{21}$ and R2 which is $\neg S_{21} \rightarrow \neg W_{21} \land \neg W_{22} \land \neg W_{31}$, which will give the Output as $\boldsymbol{\neg W_{21} \land \neg W_{22} \land \neg W_{31}}$

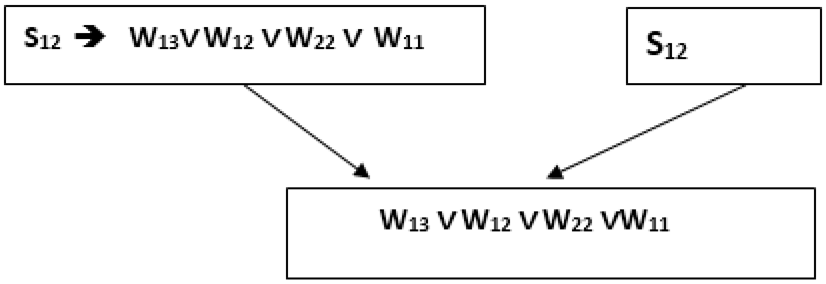
$$\begin{tikzcd}
\boxed{\neg S_{21} \Rightarrow \neg W_{21} \land \neg W_{22} \land \neg W_{31}} \arrow[dr] & & \boxed{\neg S_{21}} \arrow[dl] \\
& \boxed{\neg W_{21} \land \neg W_{22} \land \neg W_{31}} &
\end{tikzcd}$$

Apply And -Elimination rule:

Now again apply And-elimination rule to $\boldsymbol{\neg W_{21} \land \neg W_{22} \land \neg W_{31}}$, we will get three statements:
$\boldsymbol{\neg W_{21}, \neg W_{22},}$ **and** $\boldsymbol{\neg W_{31}}$.

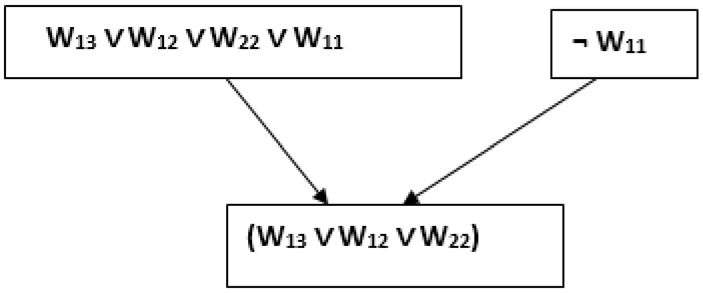
Apply MP to $\boldsymbol{S_{12}}$ and R4:

Apply Modus Ponens to $\boldsymbol{S_{12}}$ and $\boldsymbol{R_4}$ which is $\boldsymbol{S_{12} \rightarrow W_{13} \lor W_{12} \lor W_{22} \lor W_{11}}$, we will get the output as $\boldsymbol{W_{13} \lor W_{12} \lor W_{22} \lor W_{11}}$.


$S_{12} \Rightarrow W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$
S_{12}
$W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$

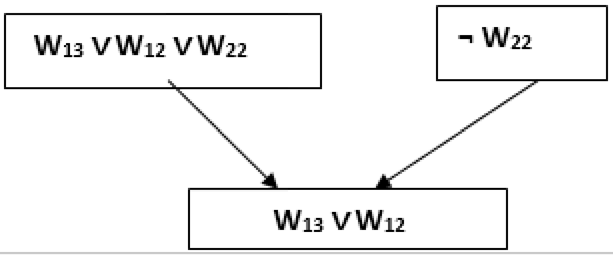
Apply Unit resolution on $W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$ and $\neg W_{11}$:

After applying Unit resolution formula on $W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$ and $\neg W_{11}$ we will get $W_{13} \vee W_{12} \vee W_{22}$.


$W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$
$\neg W_{11}$
$(W_{13} \vee W_{12} \vee W_{22})$

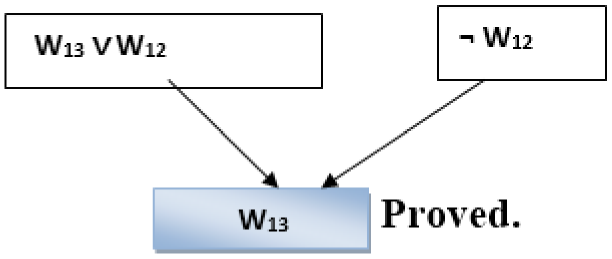
Apply Unit resolution on $W_{13} \vee W_{12} \vee W_{22}$ and $\neg W_{22}$:

After applying Unit resolution on **$W_{13} \vee W_{12} \vee W_{22}$**, and **$\neg W_{22}$**, we will get **$W_{13} \vee W_{12}$** as output.



Apply Unit Resolution on $W_{13} \vee W_{12}$ and $\neg W_{12}$:

After Applying Unit resolution on **$W_{13} \vee W_{12}$ and $\neg W_{12}$**, we will get **W_{13}** as an output, hence it is proved that the Wumpus is in the room [1, 3].



Wumpus World (Exam) Problem

We can construct a knowledge base for the wumpus world. We focus first on the immutable aspects of the wumpus world, leaving the mutable aspects for a later section. For now, we need the following symbols for each [x, y] location: (Section 7.4. Propositional Logic: A Very Simple Logic Page 247)

- Px,y is true if there is a pit in [x, y].
- Wx,y is true if there is a wumpus in [x, y], dead or alive.
- Bx,y is true if the agent perceives a breeze in [x, y].
- Sx,y is true if the agent perceives a stench in [x, y].

The sentences we write will suffice to derive ¬P1,2 (there is no pit in [1,2]), We label each sentence Ri so that we can refer to them:

- There is no pit in [1,1]:
    - R1 : ¬P1,1 .
- A square is breezy if and only if there is a pit in a neighboring square. This has to be stated for each square; for now, we include just the relevant squares:
    - R2 : B1,1 ⇔ (P1,2 ∨ P2,1) .
    - R3 : B2,1 ⇔ (P1,1 ∨ P2,2 ∨ P3,1) .
- The preceding sentences are true in all wumpus worlds. Now we include the breeze percepts for the first two squares visited in the specific world the agent is in, leading up to the situation in Figure 7.3(b).
    - R4 : ¬B1,1 .
    - R5 : B2,1 .

Prove that no Pit at [1,2] ¬P1,2.

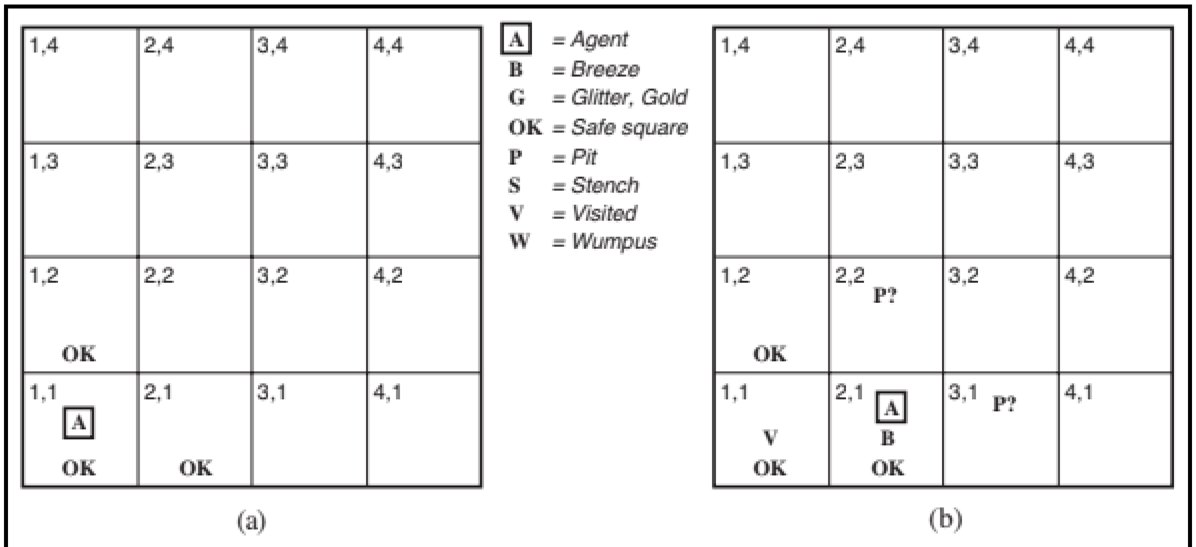


Figure 7.3 The first step taken by the agent in the wumpus world. (a) The initial situation, after percept [*None*, *None*, *None*, *None*, *None*]. (b) After one move, with percept [*None*, *Breeze*, *None*, *None*, *None*].

Solution:

We start with the knowledge base containing R1 through R5 and show how to prove $\neg P_{1,2}$, that is, there is no pit in [1,2].

First, we apply biconditional elimination to R2 to obtain

R2 : $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$

Convert to, $B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}$

Then we apply And-Elimination to obtain

$(P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}$

Logical equivalence for contrapositives gives (from propositional rules)

$\neg B_{1,1} \Rightarrow \neg(P_{1,2} \lor P_{2,1})$

Now we can apply Modus Ponens and the percept R4 (i.e., $\neg B_{1,1}$), to obtain

$\neg(P_{1,2} \lor P_{2,1})$

Finally, we apply De Morgan's rule, giving the conclusion

$\neg P_{1,2} \land \neg P_{2,1}$.

That is, neither [1,2] nor [2,1] contains a pit.

Inference in First-Order Logic

Inference in First-Order Logic is used to deduce new facts or sentences from existing sentences. Before understanding the FOL inference rule, let's understand some basic terminologies used in FOL.

Substitution: Substitution is a fundamental operation performed on terms and formulas. It occurs in all inference systems in first-order logic. The substitution is complex in the presence of quantifiers in FOL. If we write **F[a/x]**, so it refers to substitute a constant "**a**" in place of variable "**x**".

Equality: First-Order logic does not only use predicate and terms for making atomic sentences but also uses another way, which is equality in FOL. For this, we can use **equality symbols** which specify that the two terms refer to the same object.

Example: Brother (John) = Smith.

As in the above example, the object referred by the **Brother (John)** is similar to the object referred by **Smith**. The equality symbol can also be used with negation to represent that two terms are not the same objects.

Example: ¬(x=y) which is equivalent to x ≠y.

FOL inference rules for quantifier:

As propositional logic we also have inference rules in first-order logic, so following are some basic inference rules in FOL:

- Universal Generalization
- Universal Instantiation
- Existential Instantiation
- Existential introduction

1. Universal Generalization:

Universal generalization is a valid inference rule which states that if premise P(c) is true for any arbitrary element c in the universe of discourse, then we can have a conclusion as ∀ x P(x).

It can be represented as: $\frac{P(c)}{\forall x\, P(x)}$.

This rule can be used if we want to show that every element has a similar property.

In this rule, x must not appear as a free variable.

Example: Let's represent, P(c): "**A byte contains 8 bits**", so for **∀ x P(x)** "**All bytes contain 8 bits.**", it will also be true.

2. Universal Instantiation

Universal instantiation is also called universal elimination or UI is a valid inference rule. It can be applied multiple times to add new sentences.

The new KB is logically equivalent to the previous KB.

As per UI, **we can infer any sentence obtained by substituting a ground term for the variable**.

The UI rule state that we can infer any sentence P(c) by substituting a ground term c (a constant within domain x) from **∀ x P(x) for any object in the universe of discourse**.

It can be represented as: $\dfrac{\forall x\, P(x)}{P(c)}$.

Example:1.

IF "Every person like ice-cream"=> ∀x P(x) so we can infer that
"John likes ice-cream" => P(c)

Example: 2.

Let's take a famous example,

"All kings who are greedy are Evil." So let our knowledge base contains this detail as in the form of FOL:

∀x king(x) ∧ greedy (x) → Evil (x),

So from this information, we can infer any of the following statements using Universal Instantiation:

King(John) ∧ Greedy (John) → Evil (John),

King(Richard) ∧ Greedy (Richard) → Evil (Richard),

King(Father(John)) ∧ Greedy (Father(John)) → Evil (Father(John)),

3. Existential Instantiation

Existential instantiation is also called Existential Elimination, which is a valid inference rule in first-order logic.

It can be applied only once to replace the existential sentence.

The new KB is not logically equivalent to old KB, but it will be satisfiable if old KB was satisfiable.

This rule states that one can infer P(c) from the formula given in the form of ∃x P(x) for a new constant symbol c.

The restriction with this rule is that c used in the rule must be a new term for which P(c) is true.

It can be represented as: $\dfrac{\exists x\, P(x)}{P(c)}$

Example:

From the given sentence: **∃x Crown(x) ∧ OnHead(x, John),**

So we can infer: **Crown(K) ∧ OnHead(K, John),** as long as K does not appear in the knowledge base.

The above used K is a constant symbol, which is called **Skolem constant**.

The Existential instantiation is a special case of **Skolemization process**.

4. Existential introduction

An existential introduction is also known as an existential generalization, which is a valid inference rule in first-order logic.

This rule states that if there is some element c in the universe of discourse which has a property P, then we can infer that there exists something in the universe which has the property P.

It can be represented as: $\frac{P(c)}{\exists x P(x)}$

Example: Let's say that,
"Priyanka got good marks in English."
"Therefore, someone got good marks in English."

Generalized Modus Ponens Rule:

For the inference process in FOL, we have a single inference rule which is called Generalized Modus Ponens. It is lifted version of Modus ponens.

Generalized Modus Ponens can be summarized as, " P implies Q and P is asserted to be true, therefore Q must be True."

According to Modus Ponens, for atomic sentences **pi, pi', q**. Where there is a substitution θ such that SUBST **(θ, pi',) = SUBST(θ, pi)**, it can be represented as:

$$\frac{p1', p2', \ldots, pn', (p1 \land p2 \land \ldots \land pn \Rightarrow q)}{\mathrm{SUBST}(\theta, q)}$$

Example:

We will use this rule for Kings are evil, so we will find some x such that x is king, and x is greedy so we can infer that x is evil.

Here let say,

p1' is king(John)   p1 is king(x)

p2' is Greedy(y)   p2 is Greedy(x)

θ is {x/John, y/John}   q is evil(x)

SUBST(θ,q).

What is 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: Find the MGU for Unify{King(x), King(John)}

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.

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))**.

In this example, we need to make both above statements identical to each other. For this, we will perform the substitution.

P(x, y)......... (i)
P(a, f(z))......... (ii)

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]**.

Conditions for Unification:

Following are some basic 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.

Implementation of the Algorithm

Step.1: Initialize the substitution set to be empty.

Step.2: Recursively unify atomic sentences:

(a) Check for Identical expression match.
(b) If one expression is a variable v_i, and the other is a term t_i which does not contain variable vi, then:
    a. Substitute t_i / v_i in the existing substitutions
    b. Add t_i / v_i to the substitution setlist.
    c. If both the expressions are functions, then function name must be similar, and the number of arguments must be the same in both the expressions.

Unification Exam Problems

For each pair of the following atomic sentences find the most general unifier (If exist).

Problem 1: Find the MGU of {p(f(a), g(Y)) and p(X, X)}

Solution:

S0 => Here, Ψ1 = p(f(a), g(Y)), and Ψ2 = p(X, X)

SUBST θ= {f(a) / X}

S1 => Ψ1 = p(f(a), g(Y)), and Ψ2 = p(f(a), f(a))

SUBST θ= {f(a) / g(y)}, Unification failed.

Unification is not possible for these expressions.

Problem 2: Find the MGU of {p(b, X, f(g(Z))) and p(Z, f(Y), f(Y))}

Solution:

Here, Ψ1 = p(b, X, f(g(Z))) , and Ψ2 = p(Z, f(Y), f(Y))

S0 => { p(b, X, f(g(Z))); p(Z, f(Y), f(Y))}

SUBST θ={b/Z}

S1 => { p(b, X, f(g(b))); p(b, f(Y), f(Y))}

SUBST θ={f(Y) /X}

S2 => { p(b, f(Y), f(g(b))); p(b, f(Y), f(Y))}

SUBST θ= {g(b) /Y}

S2 => { p(b, f(g(b)), f(g(b)); p(b, f(g(b)), f(g(b))} Unified Successfully.

And Unifier = { b/Z, f(Y) /X , g(b) /Y}.

Problem 3: Find the MGU of {p (X, X), and p (Z, f(Z))}

Solution:

Here, Ψ1 = {p (X, X), and Ψ2 = p (Z, f(Z))

S0 => {p (X, X), p (Z, f(Z))}

SUBST θ= {X/Z}

S1 => {p (Z, Z), p (Z, f(Z))}

SUBST θ= {f(Z) / Z}, Unification Failed.

Hence, unification is not possible for these expressions.

Problem 4: Find the MGU of UNIFY(prime (11), prime(y))

Solution:

Here, Ψ1 = {prime(11) , and Ψ2 = prime(y)}

S0 => {prime(11) , prime(y)}

SUBST θ= {11/y}

S1 => {prime(11) , prime(11)} , Successfully unified.

Unifier: {11/y}.

Problem 5: Find the MGU of Q(a, g(x, a), f(y)), Q(a, g(f(b), a), x)}

Solution:

Here, Ψ1 = Q(a, g(x, a), f(y)), and Ψ2 = Q(a, g(f(b), a), x)

S0 => {Q(a, g(x, a), f(y)); Q(a, g(f(b), a), x)}

SUBST θ= {f(b)/x}

S1 => {Q(a, g(f(b), a), f(y)); Q(a, g(f(b), a), f(b))}

SUBST θ= {b/y}

S1 => {Q(a, g(f(b), a), f(b)); Q(a, g(f(b), a), f(b))}, Successfully Unified.

Unifier: [a/a, f(b)/x, b/y].

Problem 6: UNIFY(knows(Richard, x), knows(Richard, John))

Solution:

Here, Ψ1 = knows(Richard, x), and Ψ2 = knows(Richard, John)

S0 => { knows(Richard, x); knows(Richard, John)}

SUBST θ= {John/x}

S1 => { knows(Richard, John); knows(Richard, John)}, Successfully Unified.

Unifier: {John/x}.

Resolution in FOL

Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions. It was invented by a Mathematician John Alan Robinson in the year 1965.

Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule which can efficiently operate on the **conjunctive normal form or clausal form**.

Clause: Disjunction of literals (an atomic sentence) is called a **clause**. It is also known as a unit clause.

Conjunctive Normal Form: A sentence represented as a conjunction of clauses is said to be **conjunctive normal form** or **CNF**.

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).

Proof by resolution (Exam) Example:

Problem 1:

Knowledge Base

- **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:

a. **John likes peanuts.**

Step-1: Conversion of Facts into FOL

In the first step we will convert all the given statements into its first order logic.

a. ∀x: food(x) → likes(John, x)

b. food(Apple) Λ food(vegetables)

c. ∀x ∀y: eats(x, y) Λ ¬ killed(x) → food(y)

d. eats (Anil, Peanuts) Λ alive(Anil).

e. ∀x : eats(Anil, x) → eats(Harry, x)

f. ∀x: ¬ killed(x) → alive(x)

g. ∀x: alive(x) →¬ killed(x)

added predicates.

h. likes(John, Peanuts)

Step-2: Conversion of FOL into CNF

In First order logic resolution, it is required to convert the FOL into CNF as CNF form makes easier for resolution proofs.

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).

Rename variables or standardize variables

(a) ∀x ¬ food(x) V likes(John, x)

(b) food(Apple) Λ food(vegetables)

(c) ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)

(d) eats (Anil, Peanuts) Λ alive(Anil)

(e) ∀w¬ eats(Anil, w) V eats(Harry, w)

(f) ∀g ¬killed(g)] V alive(g)

(g) ∀k ¬ alive(k) V ¬ killed(k)

(h) likes(John, Peanuts).

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.

(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).

Note: Statements "food(Apple) Λ food(vegetables)" and "eats (Anil, Peanuts) Λ alive(Anil)" can be written in two separate statements. Distribute conjunction Λ over disjunction ¬.

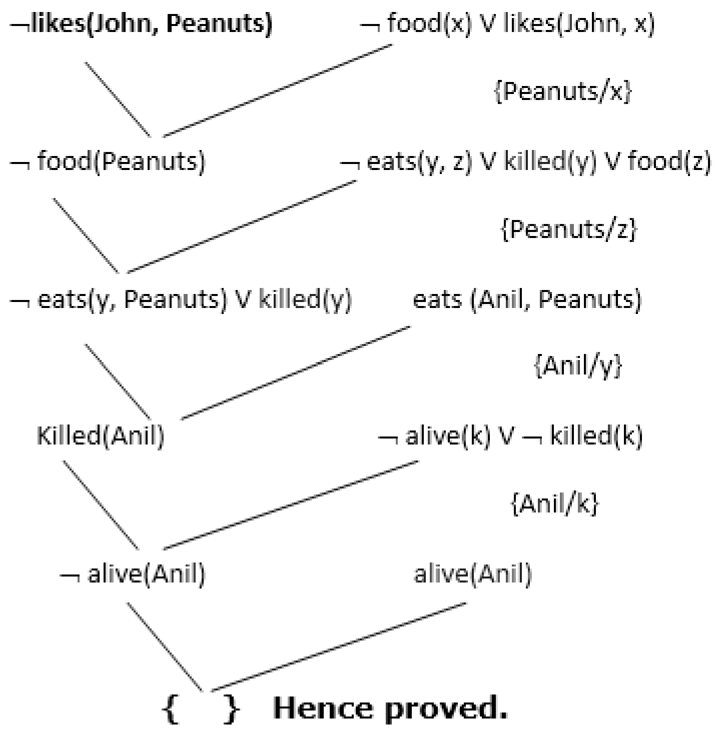
This step will not make any change in this problem.

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:



Hence the negation of the conclusion has been proved as a complete contradiction with the given set of statements.

Explanation of Resolution graph:

- In the first step of resolution graph, **¬likes(John, Peanuts)** , and **likes(John, x)** get resolved(canceled) by substitution of **{Peanuts/x}**, and we are left with **¬ food(Peanuts)**
- In the second step of the resolution graph, **¬ food(Peanuts)** , and **food(z)** get resolved (canceled) by substitution of **{ Peanuts/z}**, and we are left with **¬ eats(y, Peanuts) V killed(y)** .

- In the third step of the resolution graph, **¬ eats(y, Peanuts)** and **eats (Anil, Peanuts)** get resolved by substitution **{Anil/y}**, and we are left with **Killed(Anil)** .
- In the fourth step of the resolution graph, **Killed(Anil)** and **¬ killed(k)** get resolve by substitution **{Anil/k}**, and we are left with **¬ alive(Anil)** .
- In the last step of the resolution graph **¬ alive(Anil)** and **alive(Anil)** get resolved.

Problem 2: Knowledge Base

- All humans are mortal. ∀x (Human(x) → Mortal(x))
- Socrates is a human. Human(Socrates)
- Some philosophers are humans. ∃x (Philosopher(x) ∧ Human(x))
- If someone is a philosopher, they are wise. ∀y (Philosopher(y) → Wise(y))
- Socrates is a philosopher. Philosopher(Socrates)
- Wise individuals are knowledgeable. ∀z (Wise(z) → Knowledgeable(z))
- All knowledgeable people are educated. ∀w (Knowledgeable(w) → Educated(w))
- All educated people succeed. ∀v (Educated(v) → Succeed(v))

Goal:

- Prove that Socrates succeeds. Succeed(Socrates)

Solution:

Conversion to CNF (for Predicate Logic).

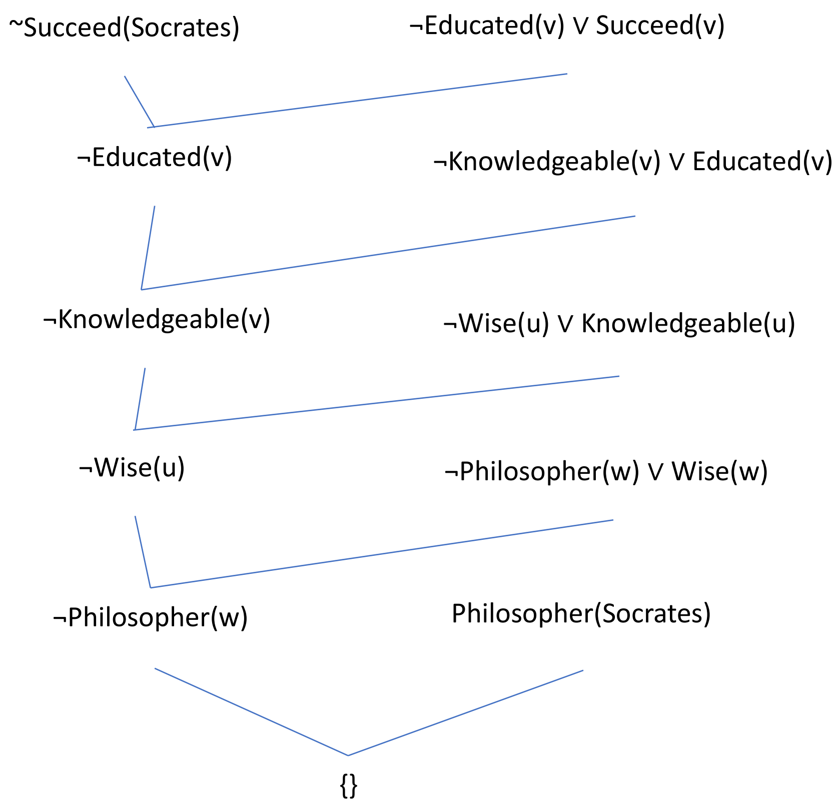
Knowledge Base:

- Clause 1: ¬Human(x) ∨ Mortal(x)
- Clause 2: Human(Socrates)
- Clause 3: Philosopher(z) ∧ Human(z)
- Clause 4: ¬Philosopher(w) ∨ Wise(w)
- Clause 5: Philosopher(Socrates)
- Clause 6: ¬Wise(u) ∨ Knowledgeable(u)
- Clause 7: ¬Knowledgeable(v) ∨ Educated(v)
- Clause 8: ¬Educated(v) ∨ Succeed(v)

Goal:

- Goal Clause: Succeed(Socrates)

Draw Resolution graph. Proof by contradiction


~Succeed(Socrates)
¬Educated(v) ∨ Succeed(v)
¬Educated(v)
¬Knowledgeable(v) ∨ Educated(v)
¬Knowledgeable(v)
¬Wise(u) ∨ Knowledgeable(u)
¬Wise(u)
¬Philosopher(w) ∨ Wise(w)
¬Philosopher(w)
Philosopher(Socrates)
{}

Problem 3:

Knowledge Base:

- All students study. ∀x (Student(x) → Study(x))
- Jane is a student. Student(Jane)
- Some students study mathematics. ∃y (Student(y) ∧ Mathematics(y))
- All students studying mathematics take calculus. ∀z (Mathematics(z) → Takes(z, Calculus))
- Jane studies mathematics. Mathematics(Jane)
- Calculus is a course. Course(Calculus)
- All students who take calculus are serious students. ∀w (Takes(w, Calculus) → Serious(w))
- Serious students succeed. ∀v (Serious(v) → Succeed(v))

Goal:

- Prove that Jane succeeds. Succeed(Jane)

Solution:

Conversion to CNF (for Predicate Logic):

Converting the statements to CNF:

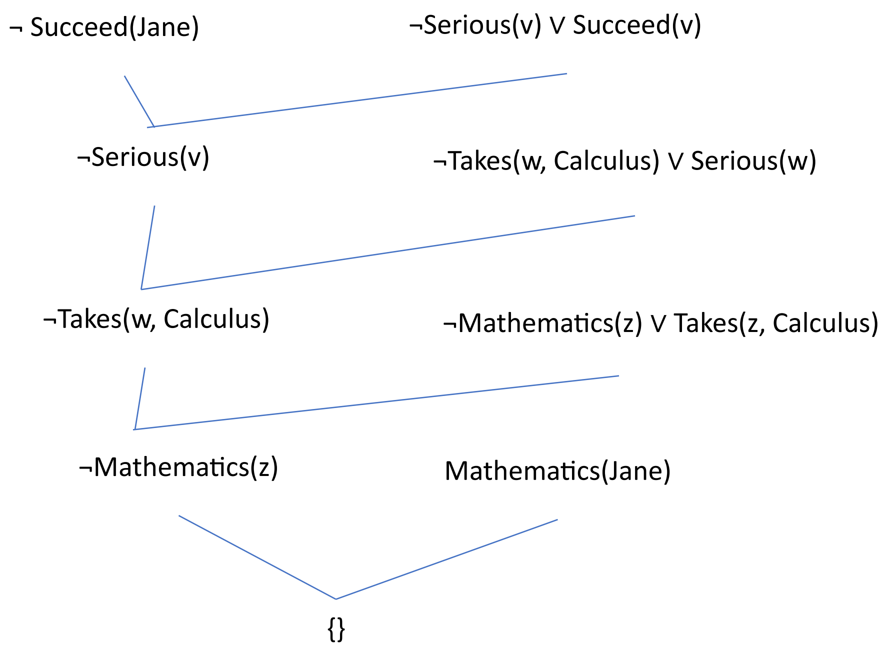
Knowledge Base:

- Clause 1: ¬Student(x) ∨ Study(x)
- Clause 2: Student(Jane)
- Clause 3: Student(y) ∧ Mathematics(y)
- Clause 4: ¬Mathematics(z) ∨ Takes(z, Calculus)
- Clause 5: Mathematics(Jane)
- Clause 6: Course(Calculus)
- Clause 7: ¬Takes(w, Calculus) ∨ Serious(w)
- Clause 8: ¬Serious(v) ∨ Succeed(v)

Goal:

Goal Clause: Succeed(Jane)

Solution:



Problem 4:

The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

Prove that Colonel West is a Criminal

Solution:

It is a crime for an American to sell weapons to hostile nations:

R1:American(x) $\wedge$ Weapon(y) $\wedge$ Sells(x,y,z) $\wedge$ Hostile(z) $\Rightarrow$ Criminal(x)

Nono … has some missiles, i.e.,

$\exists$x Owns(Nono,x) $\wedge$ Missile(x):

R2: Owns(Nono,M_1) and Missile(M_1)

R3: Missile(M1)

… all of its missiles were sold to it by Colonel West

R4: Missile(x) $\wedge$ Owns(Nono,x) $\Rightarrow$ Sells(West,x,Nono)

Missiles are weapons:

R5: Missile(x) $\Rightarrow$ Weapon(x)

An enemy of America counts as "hostile“:

R6: Enemy(x,America) $\Rightarrow$ Hostile(x)

West, who is American …

R7: American(West)

The country Nono, an enemy of America …

R8: Enemy(Nono,America)

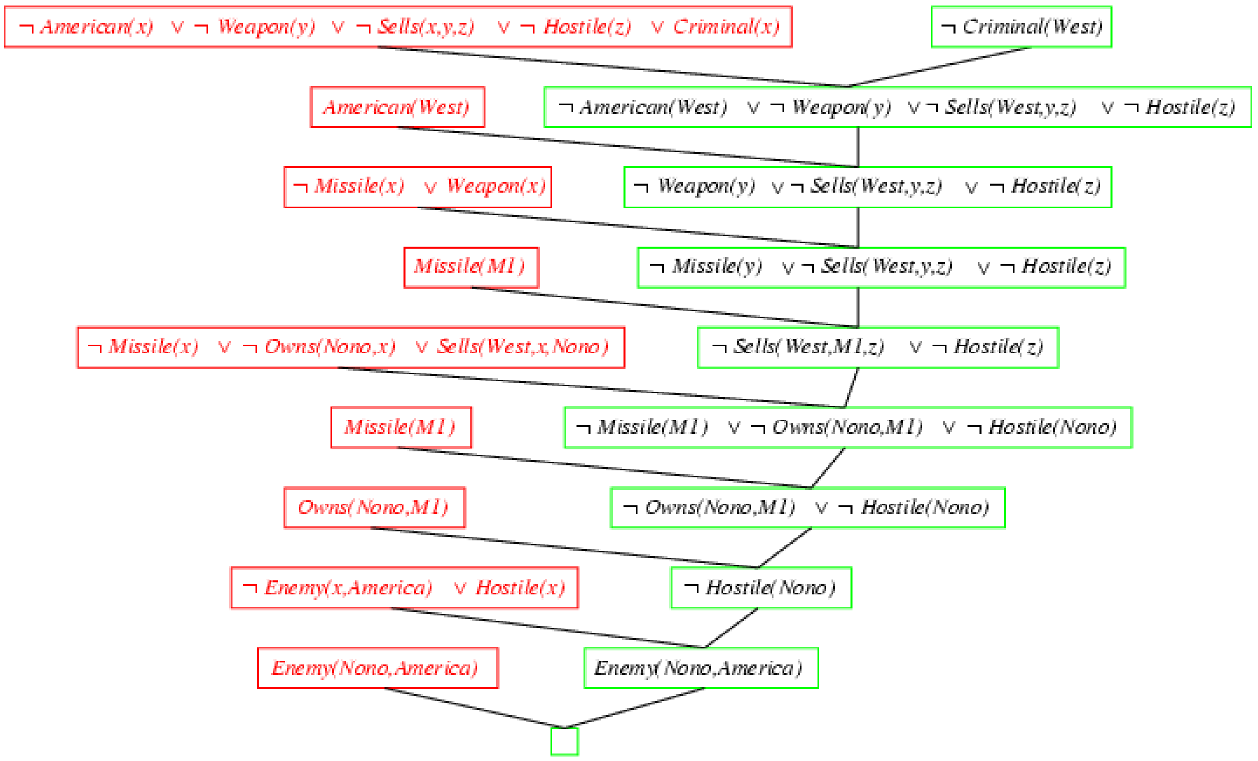
Convert to CNF

- ¬American(x) ∨ ¬Weapon(y) ∨ ¬Sells(x, y, z) ∨ ¬Hostile(z) ∨ Criminal(x)
- ¬Missile(x) ∨ ¬Owns(Nono, x) ∨ Sells(West, x, Nono)
- ¬Enemy(x, America) ∨ Hostile(x)
- ¬Missile(x) ∨ Weapon(x)
- Owns(Nono, M1)
- Missile(M1)

- American(West)
- Enemy(Nono, America) .

Query: Criminal(West)

Proof by Contradiction:


¬ American(x) ∨ ¬ Weapon(y) ∨ ¬ Sells(x,y,z) ∨ ¬ Hostile(z) ∨ Criminal(x)
¬ Criminal(West)
American(West)
¬ American(West) ∨ ¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ Weapon(x)
¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ ¬ Owns(Nono,x) ∨ Sells(West,x,Nono)
¬ Sells(West,M1,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(M1) ∨ ¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
Owns(Nono,M1)
¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
¬ Enemy(x,America) ∨ Hostile(x)
¬ Hostile(Nono)
Enemy(Nono,America)
Enemy(Nono,America)

Forward Chaining and backward chaining in AI

In artificial intelligence, forward and backward chaining is one of the important topics, but before understanding forward and backward chaining lets first understand that from where these two terms came.

Inference engine:

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

a) Forward chaining
b) Backward chaining

Horn Clause and Definite clause:

Horn clause and definite clause are the forms of sentences, which enables knowledge base to use a more restricted and efficient inference algorithm. Logical inference algorithms use forward and backward chaining approaches, which require KB in the form of the **first-order definite clause**.

Definite clause: A clause which is a disjunction of literals with **exactly one positive literal** is known as a definite clause or strict horn clause.

Horn clause: A clause which is a disjunction of literals with **at most one positive literal** is known as horn clause. Hence all the definite clauses are horn clauses.

Example: (¬ p V ¬ q V k). It has only one positive literal k.

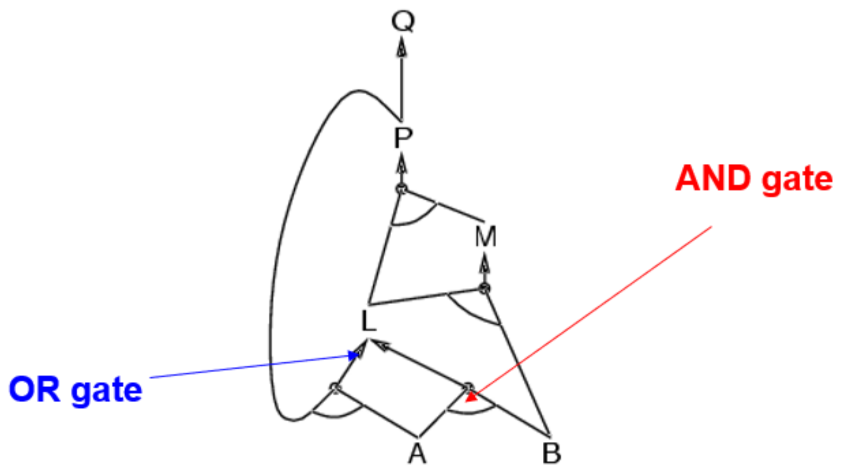
It is equivalent to p ∧ q → k.

Forward and backward Problem for Propositional Logic

Problem:

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Solution:



Forward and backward Problem for FOL

Problem

“As per the law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missiles, and all the missiles were sold to it by Robert, who is an American citizen."

Prove that **"Robert is criminal."**

To solve the above problem, first, we will convert all the above facts into first-order definite clauses, and then we will use a forward-chaining algorithm to reach the goal.

Facts Conversion into FOL:

It is a crime for an American to sell weapons to hostile nations. (Let's say p, q, and r are variables)
American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p)   ...(1)

Country A has some missiles. **Owns(A, p) ∧ Missile(p)**. It can be written in two definite clauses by using Existential Instantiation, introducing new Constant T1.
Owns(A, T1)   ......(2)
Missile(T1)   ....(3)

All of the missiles were sold to country A by Robert.
Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A)   ......(4)

Missiles are weapons.
Missile(p) → Weapons (p)   ....(5)

Enemy of America is known as hostile.
Enemy(p, America) →Hostile(p)   ....(6)

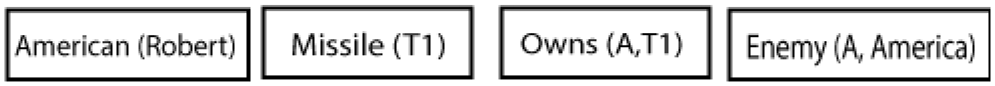
Country A is an enemy of America.
Enemy (A, America)   ....(7)

Robert is American
American(Robert). **..........(8)**

Forward chaining proof

Step-1:

In the first step we will start with the known facts and will choose the sentences which do not have implications, such as: **American(Robert), Enemy(A, America), Owns(A, T1), and Missile(T1)**. All these facts will be represented below.



Step-2:

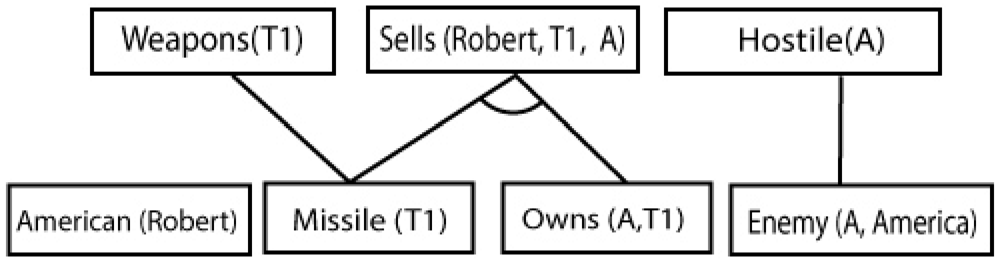
At the second step, we will see those facts which infer from available facts and with satisfied premises.

Rule-(1) does not satisfy premises, so it will not be added in the first iteration.

Rule-(2) and (3) are already added.

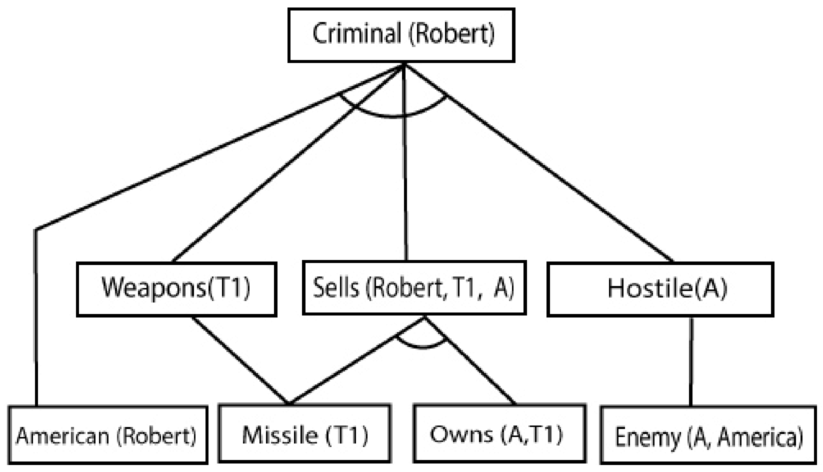
Rule-(4) satisfy with the substitution {p/T1}, **so Sells (Robert, T1, A)** is added, which infers from the conjunction of Rule (2) and (3).

Rule-(6) is satisfied with the substitution(p/A), so Hostile(A) is added and which infers from Rule-(7).



Step-3:

At step-3, as we can check Rule-(1) is satisfied with the substitution **{p/Robert, q/T1, r/A}, so we can add Criminal(Robert)** which infers all the available facts. And hence we reached our goal statement.



Hence it is proved that Robert is Criminal using forward chaining approach.

Backward Chaining:

Backward-chaining is also known as a backward deduction or backward reasoning method when using an inference engine. A backward chaining algorithm is a form of reasoning, which starts with the goal and works backward, chaining through rules to find known facts that support the goal.

Properties of backward chaining:

It is known as a top-down approach.

Backward-chaining is based on modus ponens inference rule.

In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true.

It is called a goal-driven approach, as a list of goals decides which rules are selected and used.

Backward -chaining algorithm is used in game theory, automated theorem proving tools, inference engines, proof assistants, and various AI applications.

The backward-chaining method mostly used a **depth-first search** strategy for proof.

Example:

In backward chaining, we will use the same above example, and will rewrite all the rules.

American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p) ...(1)
Owns(A, T1) ....(2)

Missile(T1) ....(3)

Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A) ......(4)

Missile(p) → Weapons (p) ....(5)

Enemy(p, America) →Hostile(p) ....(6)

Enemy (A, America) ....(7)

American(Robert) ....(8)

Backward-Chaining proof:

In Backward chaining, we will start with our goal predicate, which is **Criminal(Robert)**, and then infer further rules.

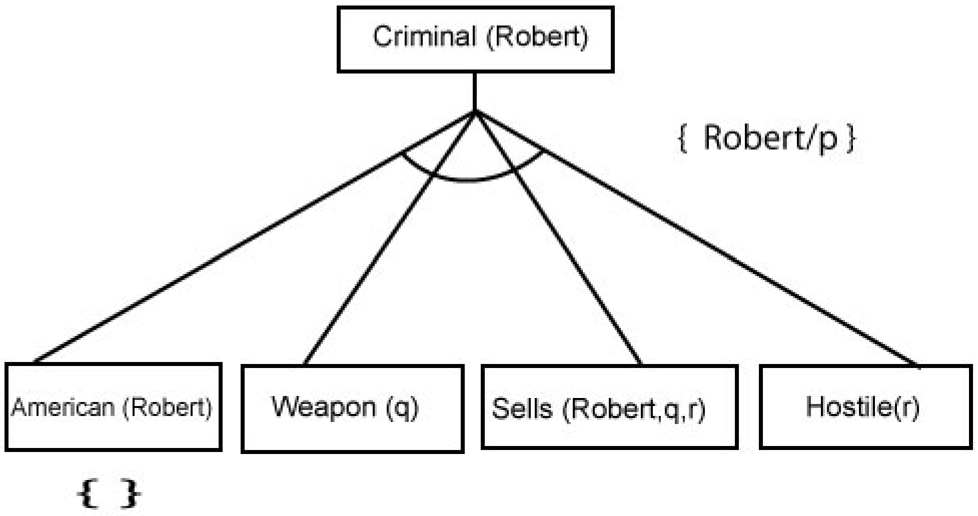
Step-1:

At the first step, we will take the goal fact. And from the goal fact, we will infer other facts, and at last, we will prove those facts true. So our goal fact is "Robert is Criminal," so following is the predicate of it.

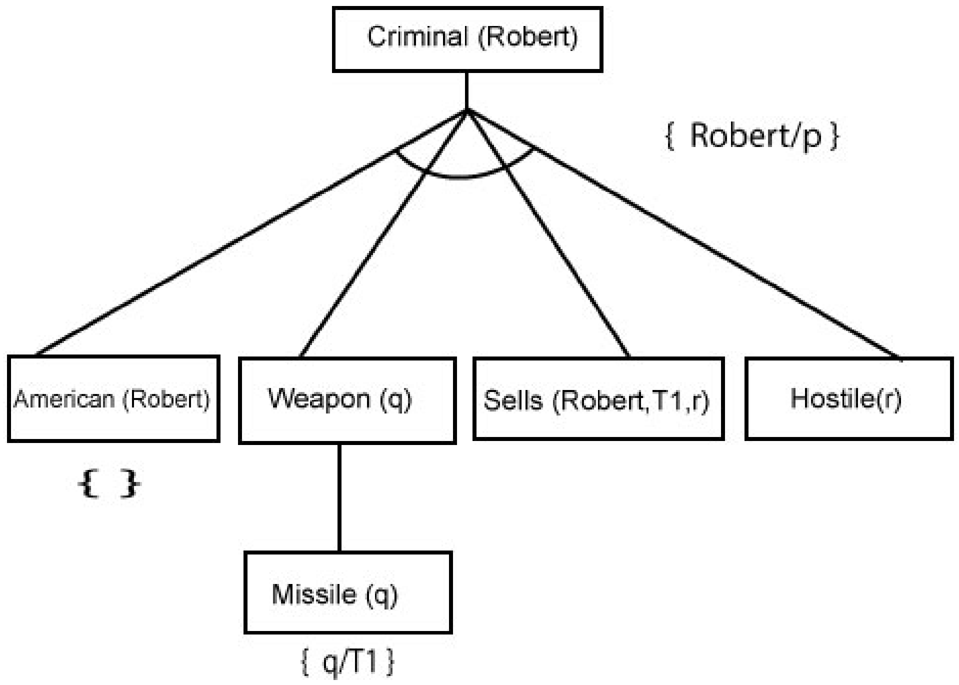
Step-2:

At the second step, we will infer other facts form goal fact which satisfies the rules. So as we can see in Rule-1, the goal predicate Criminal (Robert) is present with substitution {Robert/P}. So we will add all the conjunctive facts below the first level and will replace p with Robert.

Here we can see American (Robert) is a fact, so it is proved here.

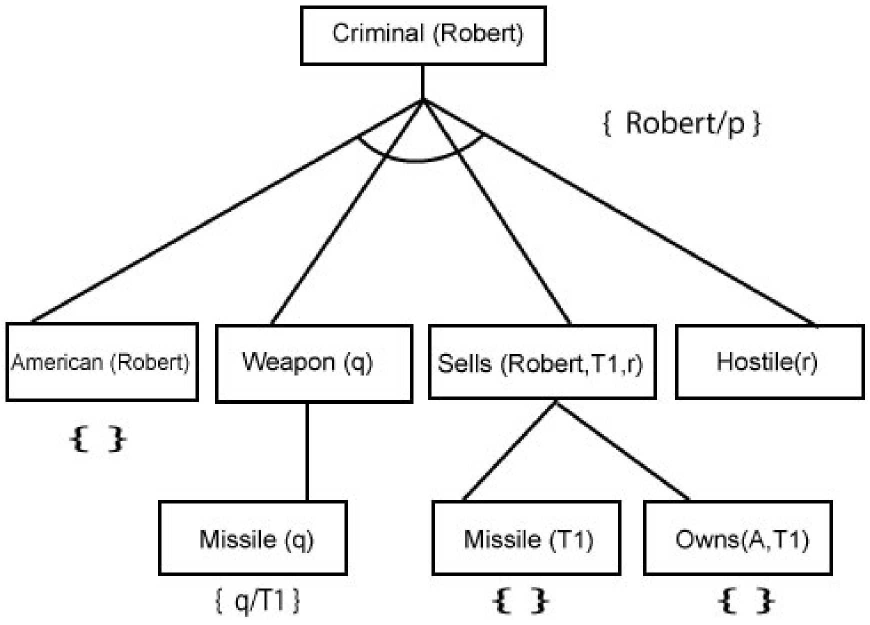


Step-3:t At step-3, we will extract further fact Missile(q) which infer from Weapon(q), as it satisfies Rule-(5). Weapon (q) is also true with the substitution of a constant T1 at q.



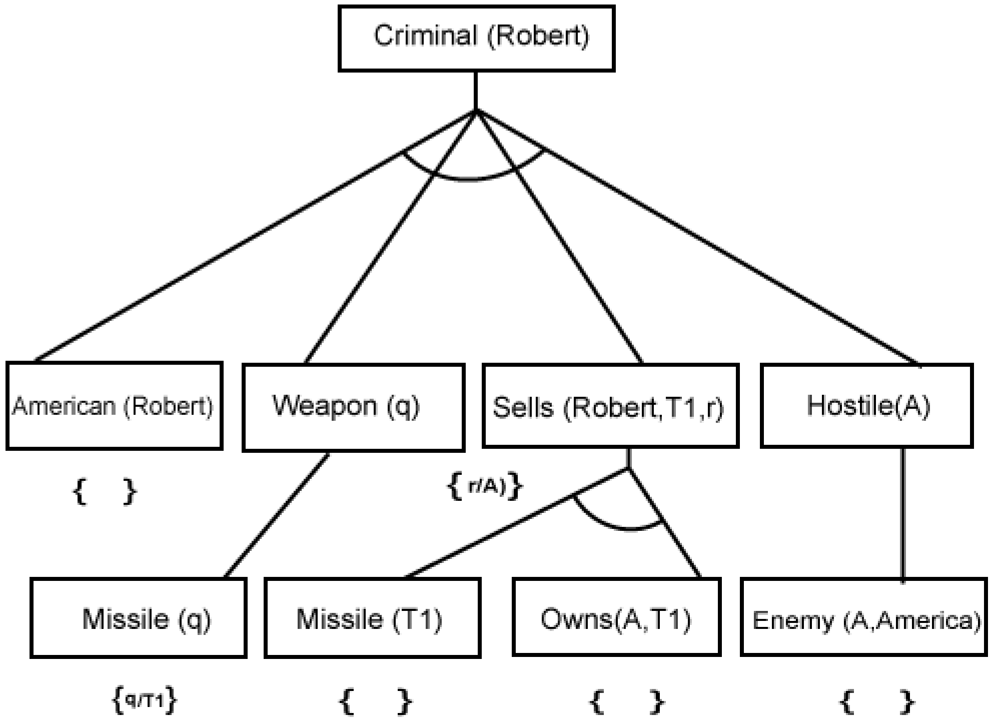
Step-4:

At step-4, we can infer facts Missile(T1) and Owns(A, T1) form Sells(Robert, T1, r) which satisfies the **Rule- 4**, with the substitution of A in place of r. So these two statements are proved here.



Step-5:

At step-5, we can infer the fact **Enemy(A, America)** from **Hostile(A)** which satisfies Rule- 6. And hence all the statements are proved true using backward chaining.


Criminal (Robert)
American (Robert)
Weapon (q)
Sells (Robert,T1,r)
Hostile(A)
{ }
{r/A)}
Missile (q)
Missile (T1)
Owns(A,T1)
Enemy (A,America)
{q/T1}
{ }
{ }
{ }

Probabilistic reasoning in Artificial intelligence

Uncertainty:

Till now, we have learned knowledge representation using first-order logic and propositional logic with certainty, which means we were sure about the predicates. With this knowledge representation, we might write A→B, which means if A is true then B is true but consider a situation where we are not sure about whether A is true or not then we cannot express this statement, this situation is called uncertainty.

Probabilistic Reasoning

Probabilistic reasoning is a technique used in AI to address uncertainty by modeling and reasoning with probabilistic Information. It allows AI systems to make decisions and predictions based on the probabilities of different outcomes, taking into account uncertain or incomplete Information. Probabilistic reasoning provides a principled approach to handling uncertainty, allowing machines to reason about uncertain situations in a rigorous and quantitative manner.

In probabilistic reasoning, there are two ways to solve problems with uncertain knowledge:

- Bayes' rule
- Bayesian Statistics

As probabilistic reasoning uses probability and related terms, so before understanding probabilistic reasoning, let's understand some common terms:

Probability: Probability can be defined as a chance that an uncertain event will occur. It is the numerical measure of the likelihood that an event will occur. The value of probability always remains between 0 and 1 that represent ideal uncertainties.

$0 \leq P(A) \leq 1$, where P(A) is the probability of an event A.

$P(A) = 0$, indicates total uncertainty in an event A.

$P(A) = 1$, indicates total certainty in an event A.

We can find the probability of an uncertain event by using the below formula.

$$\textbf{Probability of occurrence} = \frac{\text{Number of desired outcomes}}{\text{Total number of outcomes}}$$

$P(\neg A)$ = probability of a not happening event.

$P(\neg A) + P(A) = 1$.

Event: Each possible outcome of a variable is called an event.

Sample space: The collection of all possible events is called sample space.

Random variables: Random variables are used to represent the events and objects in the real world.

Prior probability: The prior probability of an event is probability computed before observing new information.

Posterior Probability: The probability that is calculated after all evidence or information has taken into account. It is a combination of prior probability and new information.

Conditional probability

Conditional probability is a probability of occurring an event when another event has already happened.

Let's suppose, we want to calculate the event A when event B has already occurred, "the probability of A under the conditions of B", it can be written as:

$$P(A|B) = \frac{P(A \land B)}{P(B)}$$

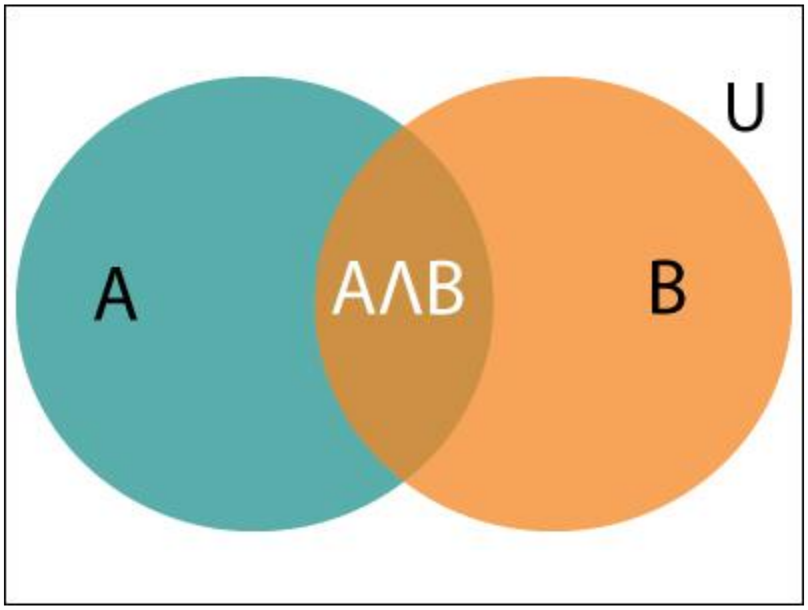
Where P($A \land B$)= Joint probability of a and B

P(B)= Marginal probability of B.

If the probability of A is given and we need to find the probability of B, then it will be given as:

$$P(B|A) = \frac{P(A \land B)}{P(A)}$$

It can be explained by using the below Venn diagram, where B is occurred event, so sample space will be reduced to set B, and now we can only calculate event A when event B is already occurred by dividing the probability of **P(A$\land$*B*) by P(B)**.



Bayes' theorem in Artificial intelligence

Bayes' theorem is also known as Bayes' rule, Bayes' law, or Bayesian reasoning, which determines the probability of an event with uncertain knowledge. In probability theory, it relates the conditional probability and marginal probabilities of two random events. Bayes' theorem was named after the

British mathematician Thomas Bayes. The Bayesian inference is an application of Bayes' theorem, which is fundamental to Bayesian statistics.

It is a way to calculate the value of P(B|A) with the knowledge of P(A|B).

Bayes' theorem allows updating the probability prediction of an event by observing new information of the real world.

As from product rule we can write:

P(A ⋀ B)= P(A|B) P(B) or Similarly, the probability of event B with known event A:

P(A ⋀ B)= P(B|A) P(A)

Equating right hand side of both the equations, we will get:

$$\boxed{P(A|B) = \frac{P(B|A)\, P(A)}{P(B)}} \quad \text{....(a)}$$

Bayes theorem in Artificial intelligence: The above equation (a) is called as Bayes' rule or Bayes' theorem. This equation is basic of most modern AI systems for probabilistic inference. It shows the simple relationship between joint and conditional probabilities. Here, P(A|B) is known as posterior, which we need to calculate, and it will be read as Probability of hypothesis A when we have occurred an evidence B.

P(B|A) is called the likelihood, in which we consider that hypothesis is true, then we calculate the probability of evidence.

P(A) is called the prior probability, probability of hypothesis before considering the evidence

P(B) is called marginal probability, pure probability of an evidence.

In the equation (a), in general, we can write P (B) = P(A)*P(B|Ai), hence the Bayes' rule can be written as:

Bayes theorem in Artificial intelligence

$$\boxed{P(A_i|B) = \frac{P(A_i) * P(B|A_i)}{\sum_{i=1}^{k} P(A_i) * P(B|A_i)}}$$

Where A1, A2, A3,........, An is a set of mutually exclusive and exhaustive events.

Simple Bayes Rule’s Problem(Exam Problem)

Problem 1: A doctor is aware that disease meningitis causes a patient to have a stiff neck, and it occurs 80% of the time. He is also aware of some more facts, which are given as follows:

- The Known probability that a patient has meningitis disease is 1/30,000.
- The Known probability that a patient has a stiff neck is 2%.

what is the probability that a patient has diseases meningitis with a stiff neck?

Solution:

A doctor is aware that disease meningitis causes a patient to have a stiff neck, and it occurs 80% of the time. He is also aware of some more facts, which are given as follows:

The Known probability that a patient has meningitis disease is 1/30,000.

The Known probability that a patient has a stiff neck is 2%.

Let a be the proposition that patient has stiff neck and b be the proposition that patient has meningitis. , so we can calculate the following as:

P(a|b) = 0.8

P(b) = 1/30000

P(a)= .02

$$\mathbf{P(b|a)} = \frac{P(a|b)P(b)}{P(a)} = \frac{0.8*(\frac{1}{30000})}{0.02} = 0.001333333.$$

Problem 2: From a standard deck of playing cards, a single card is drawn. The probability that the card is king is 4/52, then calculate posterior probability P(King|Face), which means the drawn face card is a king card.

Solution:

$$\mathbf{P(king|face)} = \frac{P(Face|king)*P(King)}{P(Face)} \quad ....(i)$$

(king): probability that the card is King= 4/52= 1/13

P(face): probability that a card is a face card= 3/13

P(Face|King): probability of face card when we assume it is a king = 1

Putting all values in equation (i) we will get:

$$P(king|face) = \frac{1*(\frac{1}{13})}{(\frac{3}{13})} = 1/3,$$ **it is a probability that a face card is a king card.**

Problem 3: Let's take System *Y*, which has a malfunction rate of 0.5%. There is a diagnostic test available for detecting malfunctions in System *Y*. It correctly identifies 98% of System *Y's* malfunctions. Also, the test has a false positive rate of 3%. What is the probability of System *Y* having a malfunction given that the test result is positive?

Solution:

We can apply Bayes' theorem to calculate $P(Malfunction|Positivetest)$.
$P(Malfunction|Positivetest)=(P(Malfunction)*P(Positivetest|Malfunction))/P(Positivetest)$

$P(Malfunction)=0.5\%=0.005$

$P(Positivetest|Malfunction)=0.98$

$P(NoMalfunction)=100\%-0.5\%$

$P(NoMalfunction)=99.5\%$

$P(Positivetest|NoMalfunction)=0.03$

Since we don't have P(Positive Test), we'll have to calculate it using the rest of the values. For this, we need to consider the probabilities of getting a positive test result in both malfunction and non-malfunction cases.
$P(Positivetest)=(P(Malfunction)*P(Positivetest|Malfunction))+(P(NoMalfunction)*P(Positivetest|NoMalfunction))$

$P(PositiveTest)=(0.005*0.98)+(0.995*0.03)$

$P(PositiveTest)=0.035$

Plugging in the values in Bayes' theorem gives us our result:

$P(Malfunction|PositiveTest)=(0.005*0.98)/0.035$

$P(Malfunction|PositiveTest)=0.142857.$

Bayesian Belief Network in artificial intelligence

In the field of artificial intelligence and decision-making, Bayesian Belief Networks (BBNs) have emerged as a powerful tool for probabilistic reasoning and inference. BBNs provide a framework for representing and analyzing complex systems by explicitly modelling the relationships between uncertain variables. With their ability to reason under uncertainty, BBNs have found wide-ranging applications in areas such as healthcare, finance, environmental management, and more. In this technical article, we will explore the fundamentals of Bayesian Belief Networks, their construction, inference algorithms, and real-world applications. Whether you are a researcher, practitioner or enthusiast in the field of AI, this article will provide you with a comprehensive understanding of BBNs and their potential for solving real-world problems.

Bayesian Network Consists Of Two Parts

Together, the DAG and the conditional probability tables allow us to perform probabilistic inference in the network, such as computing the probability of a particular variable given the values of other variables in the network. Bayesian networks have many applications in machine learning, artificial intelligence, and decision analysis.

Directed Acyclic Graph

This is a graphical representation of the variables in the network and the causal relationships between them. The nodes in the DAG represent variables, and the edges represent the dependencies between the variables. The arrows in the graph indicate the direction of causality.

Table of Conditional Probabilities

For each node in the DAG, there is a corresponding table of conditional probabilities that specifies the probability of each possible value of the node given the values of its parents in the DAG. These tables encode the probabilistic relationships between the variables in the network.

A Bayesian Belief Network Graph

A Bayesian network is a probabilistic graphical model that represents a set of variables and their conditional dependencies through a directed acyclic graph (DAG). The nodes in the graph represent variables, while the edges indicate the probabilistic relationships between them.

Table of Conditional Probabilities

For each node in the DAG, there is a corresponding table of conditional probabilities that specifies the probability of each possible value of the node given the values of its parents in the DAG. These tables encode the probabilistic relationships between the variables in the network.

A Bayesian Belief Network Graph

A Bayesian network is a probabilistic graphical model that represents a set of variables and their conditional dependencies through a directed acyclic graph (DAG). The nodes in the graph represent variables, while the edges indicate the probabilistic relationships between them.

- The nodes of the network graph in the preceding diagram stand in for the random variables A, B, C, and D, respectively.
- Node A is referred to as the parent of Node B if we are thinking about node B, which is linked to node A by a directed arrow.
- Node C is independent of node A.

Joint Probability Distribution

In Bayesian network modeling, joint probability distribution is a crucial concept that describes the probability of all possible configurations of the network's variables.

The joint probability distribution of a Bayesian network is the product of the conditional probabilities of each node given its parents in the network. This means that the joint probability distribution provides a complete description of the probability distribution of all the variables in the network.

The joint probability distribution of a Bayesian network can be used to perform a variety of tasks, such as:

Inference: Given observed values of some variables in the network, the joint probability distribution can be used to calculate the probabilities of the remaining variables.

Parameter Estimation: The joint probability distribution can be used to estimate the parameters of the network, such as the conditional probabilities of each node given its parents.

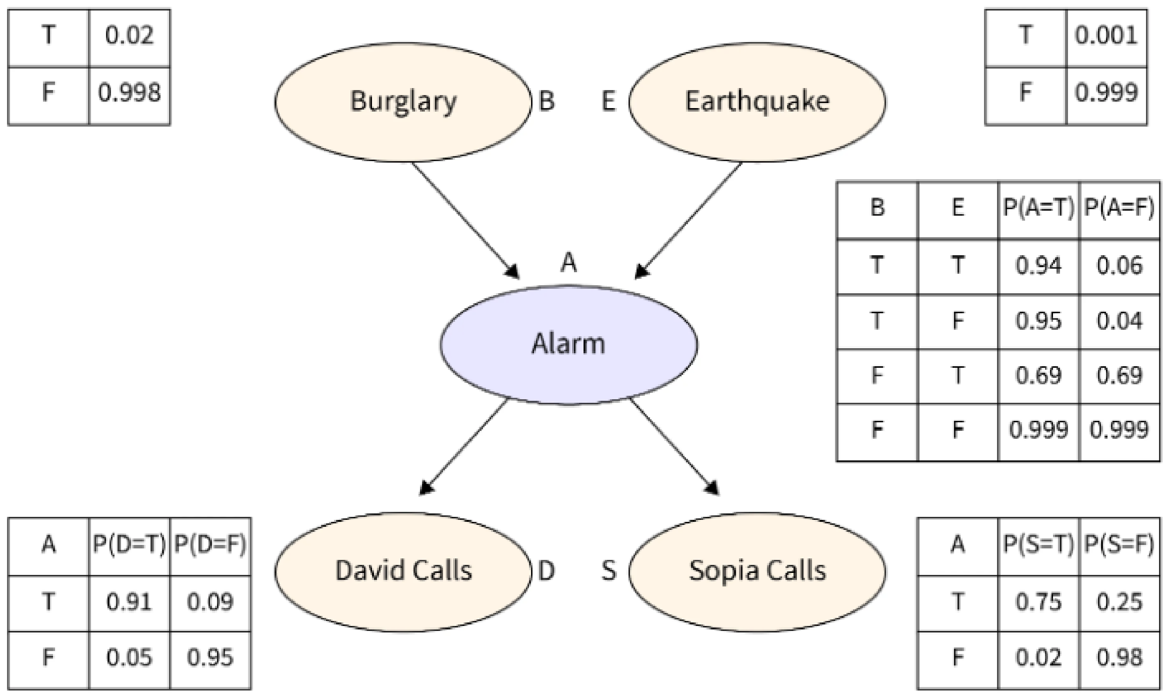
Model Selection: The joint probability distribution can be used to compare different Bayesian network models and choose the best one that fits the observed data.

Prediction: The joint probability distribution can be used to make predictions about the behavior of the variables in the network.

Example of Bayesian Network

Problem 1: To prevent break-ins, Harry put a brand-new burglar alarm at his house. The alarm consistently reacts to a break-in, but it also reacts to little earthquakes. James and Safina, two of Harry's neighbours, have agreed to call Harry at work when they hear the alarm. James always calls Harry when the alarm goes off, but occasionally he gets distracted by the phone ringing and calls at other times. Safina, on the other hand, enjoys listening to loud music, so occasionally she doesn't hear the alarm. Here, we'd want to calculate the likelihood of a burglar alarm.

Determine the likelihood that the alarm went off but that neither a burglary nor an earthquake had taken place, and that both James and Safina had phoned Harry.



T	0.02
F	0.998

T	0.001
F	0.999

B	E	P(A=T)	P(A=F)
T	T	0.94	0.06
T	F	0.95	0.04
F	T	0.69	0.69
F	F	0.999	0.999

A	P(D=T)	P(D=F)
T	0.91	0.09
F	0.05	0.95

A	P(S=T)	P(S=F)
T	0.75	0.25
F	0.02	0.98

Solution:

List of All Events Occurring in a Bayesian Network

- Burglary (B)
- Earthquake(E)
- Alarm(A)
- James Calls(D)

- Safina calls(S)

Let's take the observed probability for the Burglary and earthquake component:

- $P(B=True)=0.002$, which is the probability of burglary.
- $P(B=False)=0.998$, which is the probability of no burglary.
- $P(E=True)=0.001$, which is the probability of a minor earthquake
- $P(E=False)=0.999$, Which is the probability that an earthquake not occurred.

From the formula of joint distribution, we can write the problem statement in the form of probability distribution:

*P(S, D, A, ¬B, ¬E) = P (S|A) *P (D|A)*P (A|¬B ^ ¬E) *P (¬B) *P (¬E).*

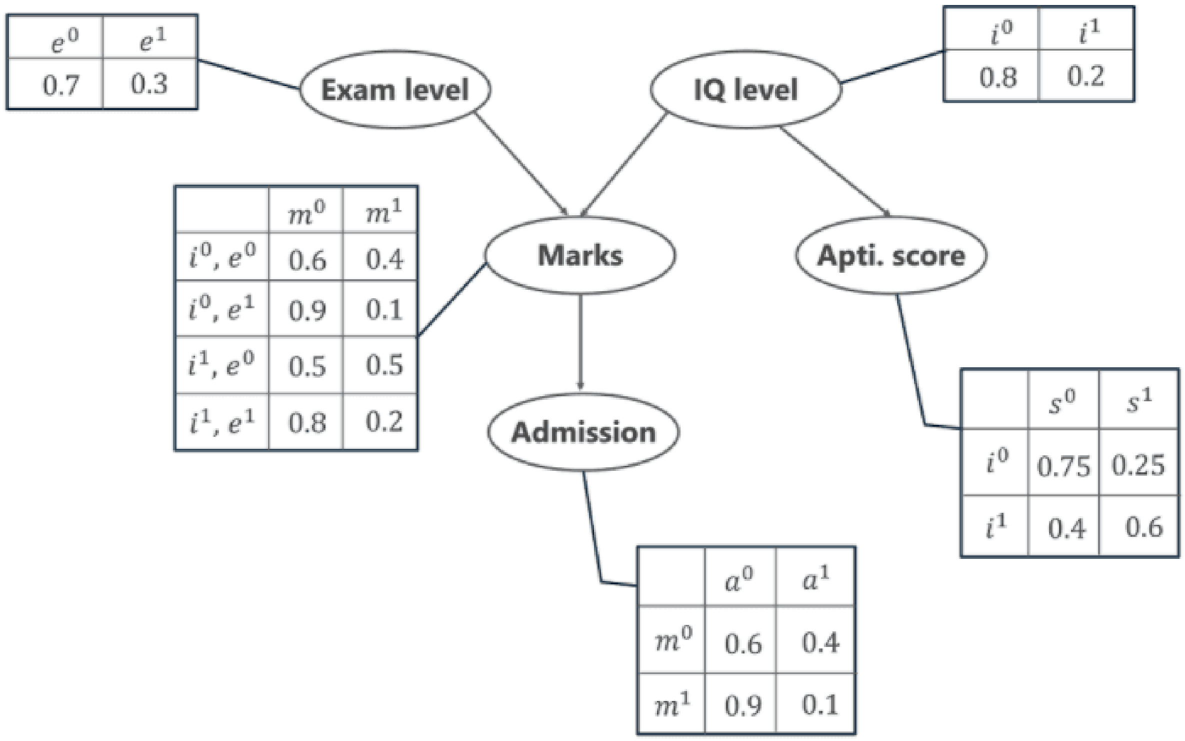
= 0.75 0.91* 0.001* 0.998*0.999*

= 0.00068045.

Problem 2: Let us now understand the mechanism of Bayesian Networks and their advantages with the help of a simple example. In this example, let us imagine that we are given the task of modeling a student's marks (m) for an exam he has just given. From the given Bayesian Network Graph below, we see that the marks depend upon two other variables. They are,

- Exam Level (e)– This discrete variable denotes the difficulty of the exam and has two values (0 for easy and 1 for difficult)
- IQ Level (i) – This represents the Intelligence Quotient level of the student and is also discrete in nature having two values (0 for low and 1 for high)

Additionally, the IQ level of the student also leads us to another variable, which is the Aptitude Score of the student (s). Now, with marks the student has scored, he can secure admission to a particular university. The probability distribution for getting admitted (a) to a university is also given below.



e^0	e^1
0.7	0.3

i^0	i^1
0.8	0.2

	m^0	m^1
i^0, e^0	0.6	0.4
i^0, e^1	0.9	0.1
i^1, e^0	0.5	0.5
i^1, e^1	0.8	0.2

	s^0	s^1
i^0	0.75	0.25
i^1	0.4	0.6

	a^0	a^1
m^0	0.6	0.4
m^1	0.9	0.1

Case 1: Calculate the probability that in spite of the exam level being difficult, the student having a low IQ level and a low Aptitude Score, manages to pass the exam and secure admission to the university.

Case 2: In another case, calculate the probability that the student has a High IQ level and Aptitude Score, the exam being easy yet fails to pass and does not secure admission to the university.

Solution:

In the above graph, we see several tables representing the probability distribution values of the given 5 variables. These tables are called the Conditional Probabilities Table or CPT. There are a few properties of the CPT given below –

- The sum of the CPT values in each row must be equal to 1 because all the possible cases for a particular variable are exhaustive (representing all possibilities).
- If a variable that is Boolean in nature has k Boolean parents, then in the CPT it has 2K probability values.

Coming back to our problem, let us first list all the possible events that are occurring in the above-given table.

1. Exam Level (e)
2. IQ Level (i)

3. Aptitude Score (s)
4. Marks (m)
5. Admission (a)

These five variables are represented in the form of a Directed Acyclic Graph (DAG) in a Bayesian Network format with their Conditional Probability tables. Now, to calculate the Joint Probability Distribution of the 5 variables the formula is given by,

P[a, m, i, e, s]= P(a | m) . P(m | i, e) . P(i) . P(e) . P(s | i)

From the above formula,

- P(a | m) denotes the conditional probability of the student getting admission based on the marks he has scored in the examination.
- P(m | i, e) represents the marks that the student will score given his IQ level and difficulty of the Exam Level.
- P(i) and P(e) represent the probability of the IQ Level and the Exam Level.
- P(s | i) is the conditional probability of the student's Aptitude Score, given his IQ Level.

With the following probabilities calculated, we can find the Joint Probability Distribution of the entire Bayesian Network.

Case 1:

From the above word problem statement, the Joint Probability Distribution can be written as below,

P[a=1, m=1, i=0, e=1, s=0]

From the above Conditional Probability tables, the values for the given conditions are fed to the formula and is calculated as below.

P[a=1, m=1, i=0, e=0, s=0] = P(a=1 | m=1) . P(m=1 | i=0, e=1) . P(i=0) . P(e=1) . P(s=0 | i=0)

= 0.1 * 0.1 * 0.8 * 0.3 * 0.75

= 0.0018

Case 2:

The formula for the JPD is given by

P[a=0, m=0, i=1, e=0, s=1]

Thus,

P[a=0, m=0, i=1, e=0, s=1]= P(a=0 | m=0) . P(m=0 | i=1, e=0) . P(i=1) . P(e=0) . P(s=1 | i=1)

= 0.6 * 0.5 * 0.2 * 0.7 * 0.6

= 0.0252

Reference

[1] Peter Norvig, “Artificial Intelligence-A Modern Approach(3rd Edition)”, Chapter 7,8,9

[2] David L. Poole & Alan K. Mackworth, “Artificial Intelligence 3e Foundations of Computational Agents”, https://artint.info/

[2] https://www.scaler.com/