Unit-3

Knowledge Representation

Human beings are good at understanding, reasoning and interpreting knowledge. And using this knowledge, they are able to perform various actions in the real world. But how do machines perform the same? In this article, we will learn about Knowledge Representation in AI and how it helps the machines perform reasoning and interpretation using Artificial Intelligence in the following sequence:

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. One of the primary purposes of Knowledge Representation includes modelling intelligent behaviour for an agent.

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.

The different kinds of knowledge that need to be represented in AI include:

- **Objects**
- **Events**
- **Performance**
- **Facts**
- **Meta-Knowledge**
- **Knowledge-base**

Now that you know about Knowledge representation in AI, let's move on and know about the different types of Knowledge.

Different Types of Knowledge-

There are 5 types of Knowledge such as:

- **Declarative Knowledge** – It includes concepts, facts, and objects and expressed in a declarative sentence.
- **Structural Knowledge** – It is a basic problem-solving knowledge that describes the relationship between concepts and objects.
- **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.

These are the important types of Knowledge Representation in AI. Now, let's have a look at the cycle of knowledge representation and how it works.

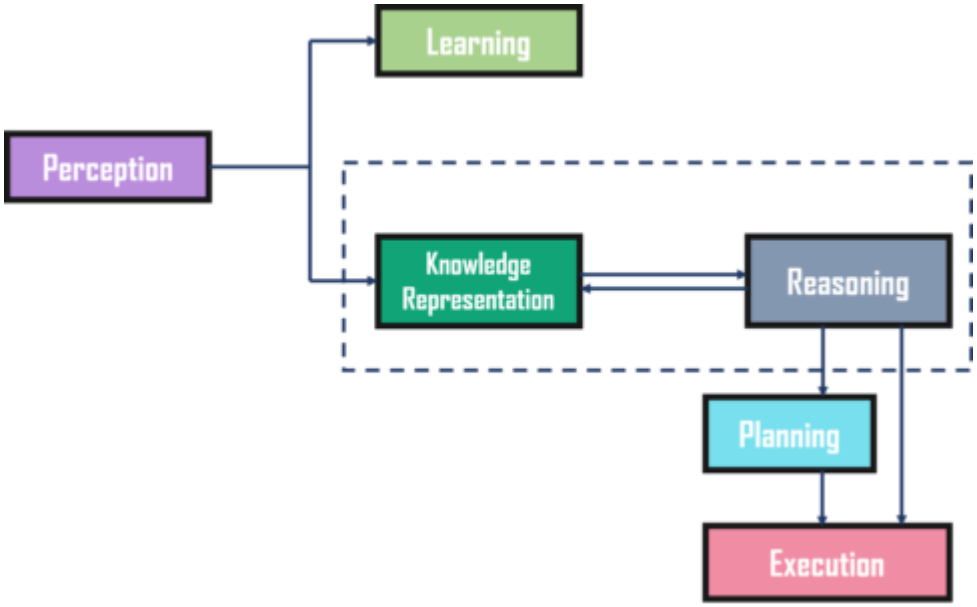
Cycle of Knowledge Representation in AI –

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

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

Here is an example to show the different components of the system and how it works:

Example



The above diagram shows the interaction of an AI system with the **real world** and the **components** involved in showing intelligence.

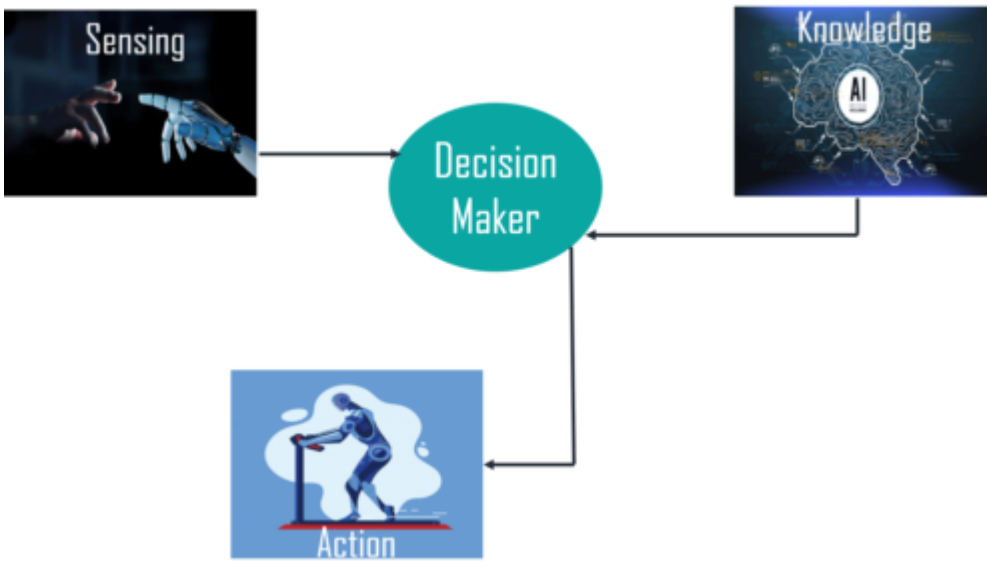
- The **Perception component** retrieves data or information from the environment. with the help of this component, you can retrieve data from the environment, find out the source of noises and check if the AI was damaged by anything. Also, it defines how to respond when any sense has been detected.
- 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.
- 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. Instead of trying to understand or build brains from the bottom up, its goal is to understand and build intelligent behavior from the top-down 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.

So, these are the different components of the cycle of Knowledge Representation in AI. Now, let's understand the relationship between knowledge and intelligence.

What is the Relation between Knowledge & Intelligence?

In the real world, knowledge plays a vital role in intelligence as well as creating artificial intelligence. It demonstrates the intelligent behaviour in **AI agents or systems**. It is possible for an agent or system to act accurately on some input only when it has the knowledge or experience about the input.

Let's take an example to understand the relationship:



In this example, there is one **decision-maker** whose actions are justified by sensing the environment and using knowledge. But, if we remove the knowledge part here, it will not be able to display any intelligent behaviour.

Now that you know the relationship between knowledge and intelligence, let's move on to the techniques of Knowledge Representation in AI.

Techniques of Knowledge Representation in AI

There are four techniques of representing knowledge such as:

Now, let's discuss these techniques in detail.

Logical Representation -

Logical representation is a language with some **definite rules** which deal with propositions and has no ambiguity in representation. It represents a conclusion based on various conditions and lays down some important **communication rules**. Also, 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	Semantics
• It decides how we can construct legal sentences in logic. • It determines which symbol we can use in knowledge representation. • Also, how to write those symbols.	• Semantics are the rules by which we can interpret the sentence in the logic. • It assigns a meaning to each sentence.

Advantages:

- Logical representation helps to perform logical reasoning.
- This representation is the basis for the programming languages.

Disadvantages:

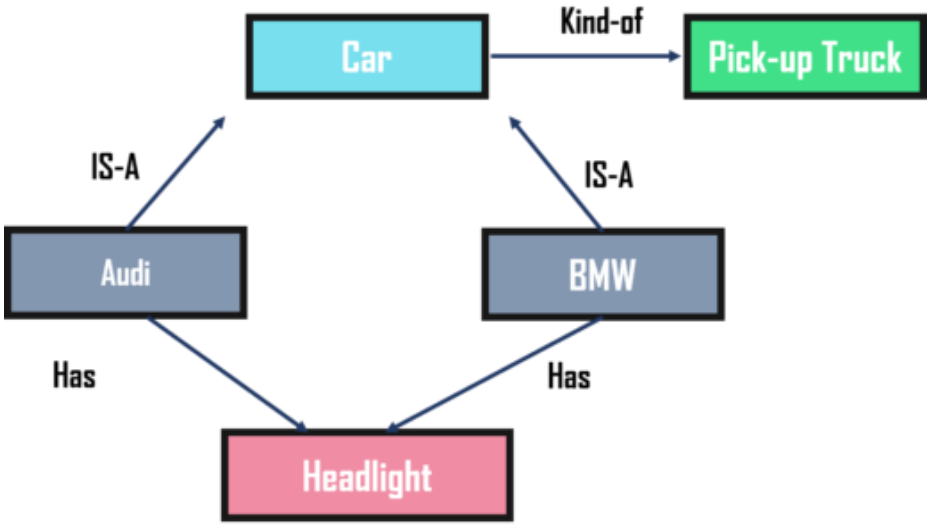
- Logical representations have some restrictions and are challenging to work with.
- This technique may not be very natural, and inference may not be very efficient.

Semantic Network Representation

Semantic networks work as an **alternative** of **predicate logic** for knowledge representation. In Semantic networks, you can represent your knowledge in the form of graphical networks. This network consists of nodes representing objects and arcs which describe the relationship between those objects. Also, it categorizes the object in different forms and links those objects.

This representation consists of two types of relations:

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



Advantages:

- Semantic networks are a natural representation of knowledge.
- Also, it conveys meaning in a transparent manner.
- These networks are simple and easy to understand.

Disadvantages:

- Semantic networks take more computational time at runtime.
- Also, these are inadequate as they do not have any equivalent quantifiers.

- These networks are not intelligent and depend on the creator of the system.

.

Frame Representation

A frame is a **record** like structure that consists of a **collection of attributes** and values to describe an entity in the world. These are the AI data structure that divides knowledge into substructures by representing stereotypes situations. Basically, it consists of a collection of slots and slot values of any type and size. Slots have names and values which are called facets.

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 cannot be easily processed.
- The inference mechanism cannot be smoothly proceeded by frame representation.
- It has a very generalized approach.

Production Rules

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. Whereas, the action part carries out the associated problem-solving steps. This complete process is called a recognize-act cycle.

The production rules system consists of three main parts:

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

Advantages:

- The production rules are expressed in natural language.
- The production rules are highly modular and can be easily removed or modified.

Disadvantages:

- It does not exhibit any learning capabilities and does not store the result of the problem for future uses.
- During the execution of the program, many rules may be active. Thus, rule-based production systems are inefficient.

So, these were the important techniques for Knowledge Representation in AI. Now, let's have a look at the requirements for these representations.

Representation Requirements

A good knowledge representation system must have properties such as:

- **Representational Accuracy:** It should represent all kinds of required knowledge.
- **Inferential Adequacy**: It should be able to manipulate the representational structures to produce new knowledge corresponding to the existing structure.
- **Inferential Efficiency**: The ability to direct the inferential knowledge mechanism into the most productive directions by storing appropriate guides.
- **Acquisitional efficiency**: The ability to acquire new knowledge easily using automatic methods.

Now, let's have a look at some of the approaches to Knowledge Representation in AI along with different examples.

Approaches to Knowledge Representation in AI-

There are different approaches to knowledge representation such as:

1. 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.

Example:

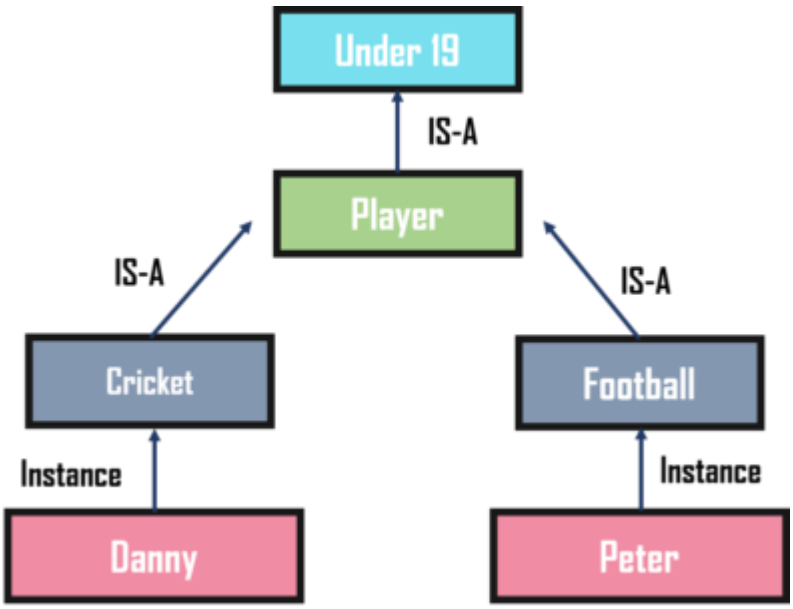
Name	Age	Emp ID
John	25	100071
Amanda	23	100056
Sam	27	100042

This is an example of representing simple relational knowledge.

2. 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.

Example:



3. 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

.

Logic in AI

Logic is defined as a scientific study of the process of reasoning and the system of rules and procedures that help in reasoning process. Logic is the process of reasoning representations using expressions in formal logic to represent the knowledge required. Inference rules and proof procedures can apply this knowledge to solve specific problems.
We can derive new piece of knowledge by proving that it is a consequence of knowledge that is already known. We generate logical statements to prove the certain assertions.

Algorithm = logic + control

Role of Logic in AI

1. Computer scientists are familiar with the idea that logic provides techniques for analyzing the inferential properties of languages. Logic can provide specification for a programming language by characterizing a mapping from programs to the computations that they implement.
2. A compiler that implements the language can be incomplete as long as it approximates the logical requirements of given problem. This makes it possible to involve logic in AI applications to vary from relatively weak uses in which logic informs the implementation process with analysis in depth .
3. Logical theories in AI are independent from implementations. They provide insights into the reasoning problem without directly informing the implementation.
4. Ideas from logic theorem proving and model construction techniques are used in AI.
5. Logic works as a analysis tool, knowledge representation technique for automated reasoning and developing Expert Systems. Also it gives the base to programming language like Prolog to develop AI software's.

Propositional logic in Artificial intelligence

Propositional logic (PL) is the simplest form of logic where all the statements are made by propositions. A proposition is a declarative statement which is either true or false. It is a technique of knowledge representation in logical and mathematical form.

Example:

1. a) It is Sunday.
2. b) The Sun rises from West (False proposition)
3. c) 3+3= 7(False proposition)

4. d) 5 is a prime number.

Following are some basic facts about propositional logic:

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

Syntax of propositional logic:

The syntax of propositional logic defines the allowable sentences for the knowledge representation. There are two types of Propositions:

1. **Atomic Propositions**
2. **Compound propositions**

- **Atomic Proposition:** Atomic propositions are the simple propositions. It consists of a single proposition symbol. These are the sentences which must be either true or false.

Example:

a) 2+2 is 4, it is an atomic proposition as it is a **true** fact.
b) "The Sun is cold" is also a proposition as it is a **false** fact.

- **Compound proposition:** Compound propositions are constructed by combining simpler or atomic propositions, using parenthesis and logical connectives.

Example:

a) "It is raining today, and street is wet."

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

Logical Connectives:

Logical connectives are used to connect two simpler propositions or representing a sentence logically. We can create compound propositions with the help of logical connectives. There are mainly five connectives, which are given as follows:

1. **Negation:** A sentence such as ¬ P is called negation of P. A literal can be either Positive literal or negative literal.
2. **Conjunction:** A sentence which has **∧** connective such as, **P ∧ Q** is called a conjunction.
   Example: Rohan is intelligent and hardworking. It can be written as,
   P= Rohan is intelligent,
   Q= Rohan is hardworking. → P∧ Q.
3. **Disjunction:** A sentence which has ∨ connective, such as **P ∨ Q**. is called disjunction, where P and Q are the propositions.
   Example: "Ritika is a doctor or Engineer",
   Here P= Ritika is Doctor. Q= Ritika is Doctor, so we can write it as **P ∨ Q**.
4. **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
5. **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.

Following is the summarized table for Propositional Logic 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:

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

For Negation:

P	¬ P
True	**False**
False	**True**

For Conjunction:

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

For disjunction:

P	Q	P ∨ Q.
True	True	**True**
False	True	**True**
True	False	**True**
False	False	**False**

For Implication:

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

For Biconditional:

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

Truth table with three propositions:

We can build a proposition composing three propositions P, Q, and R. This truth table is made-up of 8n Tuples as we have taken three proposition symbols.

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

Precedence of connectives:

Just like arithmetic operators, there is a precedence order for propositional connectors or logical operators. This order should be followed while evaluating a propositional problem. Following is the list of the precedence order for operators:

Precedence	Operators
First Precedence	Parenthesis
Second Precedence	Negation
Third Precedence	Conjunction(AND)
Fourth Precedence	Disjunction(OR)
Fifth Precedence	Implication
Six Precedence	Biconditional

Note: For better understanding use parenthesis to make sure of the correct interpretations. Such as ¬R∨ Q, It can be interpreted as (¬R) ∨ Q.

Logical equivalence:

Logical equivalence is one of the features of propositional logic. Two propositions are said to be logically equivalent if and only if the columns in the truth table are identical to each other.

Let's take two propositions A and B, so for logical equivalence, we can write it as A⇔B. In below truth table we can see that column for ¬A∨ B and A→B, are identical hence A is Equivalent to B

A	B	¬A	¬A∨ B	A→B
T	T	F	**T**	**T**
T	F	F	**F**	**F**
F	T	T	**T**	**T**
F	F	T	**T**	**T**

Properties of Operators:

- **Commutativity:**
  - P∧ Q= Q ∧ P, or
  - P ∨ Q = Q ∨ P.
- **Associativity:**
  - (P ∧ Q) ∧ R= P ∧ (Q ∧ R),
  - (P ∨ Q) ∨ R= P ∨ (Q ∨ R)
- **Identity element:**
  - P ∧ True = P,
  - P ∨ True= True.
- **Distributive:**
  - P∧ (Q ∨ R) = (P ∧ Q) ∨ (P ∧ R).
  - P ∨ (Q ∧ R) = (P ∨ Q) ∧ (P ∨ R).
- **DE Morgan's Law:**
  - ¬ (P ∧ Q) = (¬P) ∨ (¬Q)
  - ¬ (P ∨ Q) = (¬ P) ∧ (¬Q).
- **Double-negation elimination:**
  - ¬ (¬P) = P.

Limitations of Propositional logic:

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

  a. **All the girls are intelligent.**

b. **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.

First-Order Logic in Artificial intelligence

In the topic of Propositional logic, we have seen that how to represent statements using propositional logic. But unfortunately, in propositional logic, we can only represent the facts, which are either true or false. PL is not sufficient to represent the complex sentences or natural language statements. The propositional logic has very limited expressive power. Consider the following sentence, which we cannot represent using PL logic.

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

    a. **Syntax**

    b. **Semantics**

Syntax of First-Order logic:

The syntax of FOL determines which collection of symbols is a logical expression in first-order logic. The basic syntactic elements of first-order logic are symbols. We write statements in short-hand notation in FOL.

Basic Elements of First-order logic:

Following are the basic elements of FOL syntax:

Constant	1, 2, A, John, Mumbai, cat,....
Variables	x, y, z, a, b,....
Predicates	Brother, Father, >,....
Function	sqrt, LeftLegOf, ...
Connectives	∧, ∨, ¬, ⇒, ⇔
Equality	==
Quantifier	∀, ∃

Atomic sentences:

- Atomic sentences are the most basic sentences of first-order logic. These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms.
- We can represent atomic sentences as **Predicate (term1, term2, ..., term n)**.

Example: Ravi and Ajay are brothers: => Brothers(Ravi, Ajay).
Chinky is a cat: => cat (Chinky).

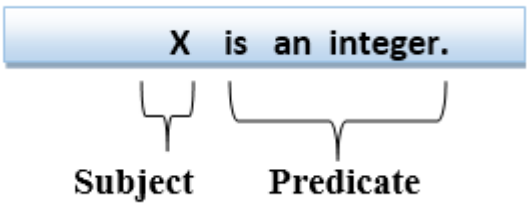
Complex Sentences:

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

First-order logic statements can be divided into two parts:

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

Consider the statement: "x is an integer.", it consists of two parts, the first part x is the subject of the statement and second part "is an integer," is known as a predicate.



Quantifiers in First-order logic:

- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression. There are two types of quantifier:

a. **Universal Quantifier, (for all, everyone, everything)**

b. **Existential quantifier, (for some, at least one).**

Universal Quantifier:

Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.

The Universal quantifier is represented by a symbol ∀, which resembles an inverted A.

Note: In universal quantifier we use implication "→".

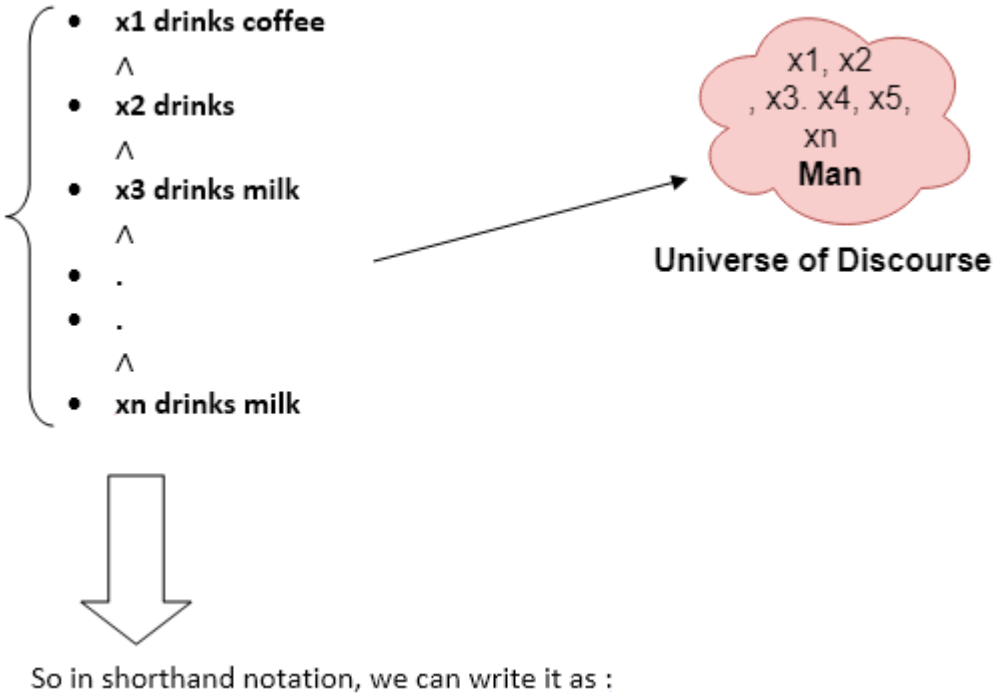
If x is a variable, then ∀x is read as:

- **For all x**
- **For each x**
- **For every x.**

Example:

All man drink coffee.

Let a variable x which refers to a cat so all x can be represented in UOD as below:



∀x man(x) → drink (x, coffee).

It will be read as: There are all x where x is a man who drink coffee.

Existential Quantifier:

Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.

It is denoted by the logical operator ∃, which resembles as inverted E. When it is used with a predicate variable then it is called as an existential quantifier.

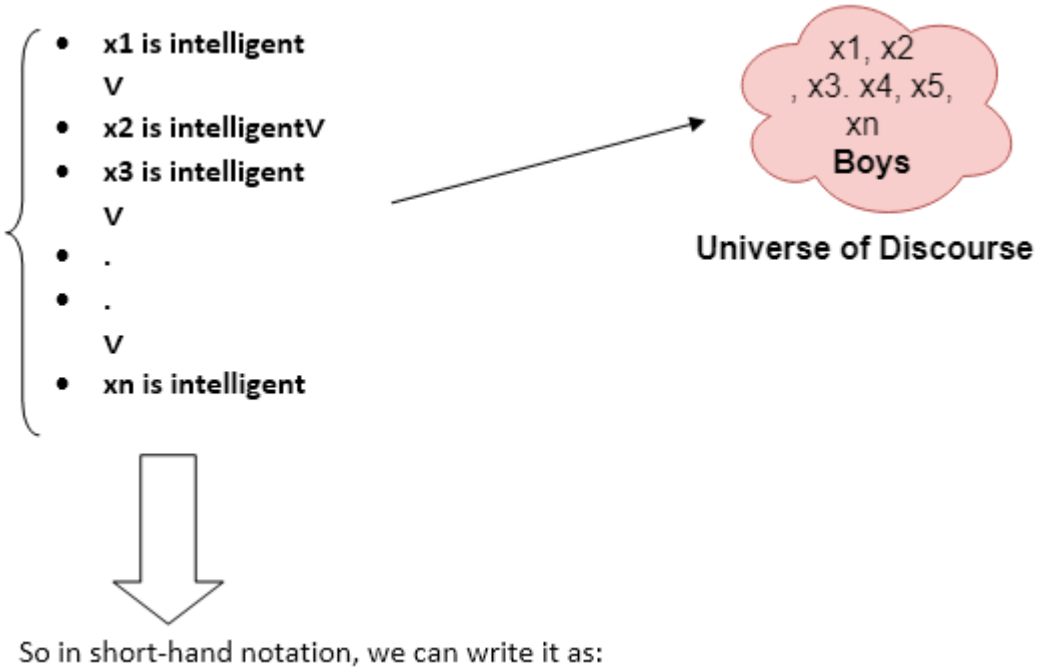
Note: In Existential quantifier we always use AND or Conjunction symbol (∧).

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

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

Example:

Some boys are intelligent.



∃x: boys(x) ∧ intelligent(x)

It will be read as: There are some x where x is a boy who is intelligent.

Points to remember:

- The main connective for universal quantifier ∀ is implication →.
- The main connective for existential quantifier ∃ is and ∧.

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.

Some Examples of FOL using quantifier:

1. All birds fly.
In this question the predicate is "**fly(bird)**."
And since there are all birds who fly so it will be represented as follows.

∀x bird(x) →fly(x).

2. Every man respects his parent.
In this question, the predicate is "**respect(x, y)," where x=man, and y= parent**.
Since there is every man so will use ∀, and it will be represented as follows:

∀x man(x) → respects (x, parent).

3. Some boys play cricket.

In this question, the predicate is "**play(x, y)**," where x= boys, and y= game. Since there are some boys so we will use **∃, and it will be represented as**:

∃x boys(x) → play(x, cricket).

4. Not all students like both Mathematics and Science.
In this question, the predicate is "**like(x, y)," where x= student, and y= subject**.
Since there are not all students, so we will use **∀ with negation, so** following representation for this:

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

5. Only one student failed in Mathematics.
In this question, the predicate is "**failed(x, y)," where x= student, and y= subject**.
Since there is only one student who failed in Mathematics, so we will use following representation for this:

∃(x) [student(x) → failed (x, Mathematics) ∧∀ (y) [¬(x==y) ∧ student(y) → ¬failed (x, Mathematics)].

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 which are given below:

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.

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

Differentiation between Propositional Logic and First-Order Logic

Propositional Logic (PL)

Propositional logic is an analytical statement which is either true or false. It is basically a technique that represents the knowledge in logical & mathematical form. There are two types of propositional logic; Atomic and Compound Propositions.

Facts about Propositional Logic

- Since propositional logic works on 0 and 1 thus it is also known as ‘Boolean Logic’.
- Proposition logic can be either true or false it can never be both.
- In this type of logic, symbolic variables are used in order to represent the logic and any logic can be used for representing the variable.
- t is comprised of objects, relations, functions, and logical connectives.
- Proposition formula which is always false is called ‘Contradiction’ whereas a proposition formula which is always true is called ‘Tautology’.

First-Order Logic (FOL)

First-Order Logic is another knowledge representation in AI which is an extended part of PL. FOL articulates the natural language statements briefly. Another name of First-Order Logic is ‘Predicate Logic’.

Facts about First Order Logic

- FOL is known as the powerful language which is used to develop information related to objects in a very easy way.
- Unlike PL, FOL assumes some of the facts that are related to objects, relations, and functions.
- FOL has two main key features or you can say parts that are; ‘Syntax’ & ‘Semantics’.

Key differences between PL and FOL

- Propositional Logic converts a complete sentence into a symbol and makes it logical whereas in First-Order Logic relation of a particular sentence will be made that involves relations, constants, functions, and constants.
- The limitation of PL is that it does not represent any individual entities whereas FOL can easily represent the individual establishment that means if you are writing a single sentence then it can be easily represented in FOL.
- PL does not signify or express the generalization, specialization or pattern for example 'QUANTIFIERS' cannot be used in PL but in FOL users can easily use quantifiers as it does express the generalization, specialization, and pattern.

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.

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:

1. **Forward chaining**
2. **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 Chaining

Forward chaining is also known as a forward deduction or forward reasoning method when using an inference engine. Forward chaining is a form of reasoning which start with atomic sentences in the knowledge base and applies inference rules (Modus Ponens) in the forward direction to extract more data until a goal is reached.

The Forward-chaining algorithm starts from known facts, triggers all rules whose premises are satisfied, and add their conclusion to the known facts. This process repeats until the problem is solved.

Properties of Forward-Chaining:

- It is a down-up approach, as it moves from bottom to top.
- It is a process of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.
- Forward-chaining approach is also called as data-driven as we reach to the goal using available data.
- Forward -chaining approach is commonly used in the expert system, such as CLIPS, business, and production rule systems.

Consider the following famous example which we will use in both approaches:

Example:

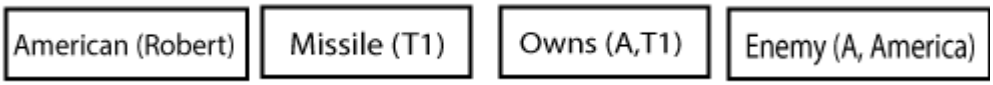
"As per the law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missiles, and all the missiles were sold to it by Robert, who is an American citizen."

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

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 as below.



Step-2:

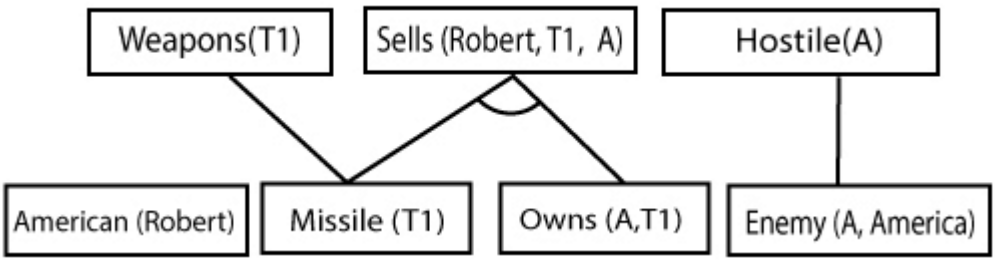
At the second step, we will see those facts which infer from available facts and with satisfied premises.

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

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

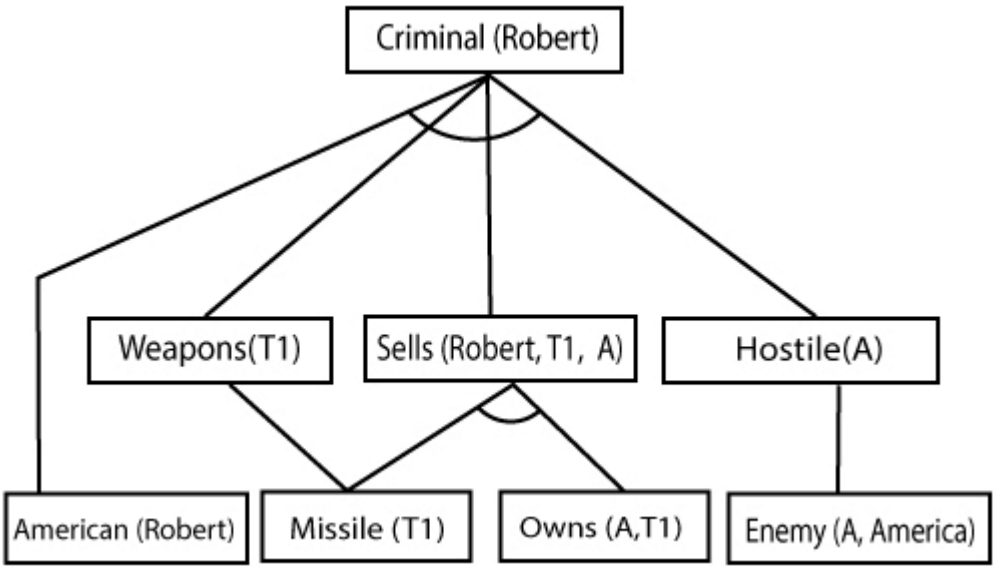
Rule-(4) satisfy with the substitution {p/T1}, **so Sells (Robert, T1, A)** is added, which infers from the conjunction of Rule (2) and (3).

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.

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.

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:

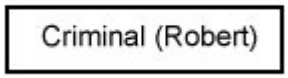
- It is known as a top-down approach.
- Backward-chaining is based on modus ponens inference rule.
- In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true.
- It is called a goal-driven approach, as a list of goals decides which rules are selected and used.
- Backward -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.

Backward-Chaining proof:

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

Step-1:

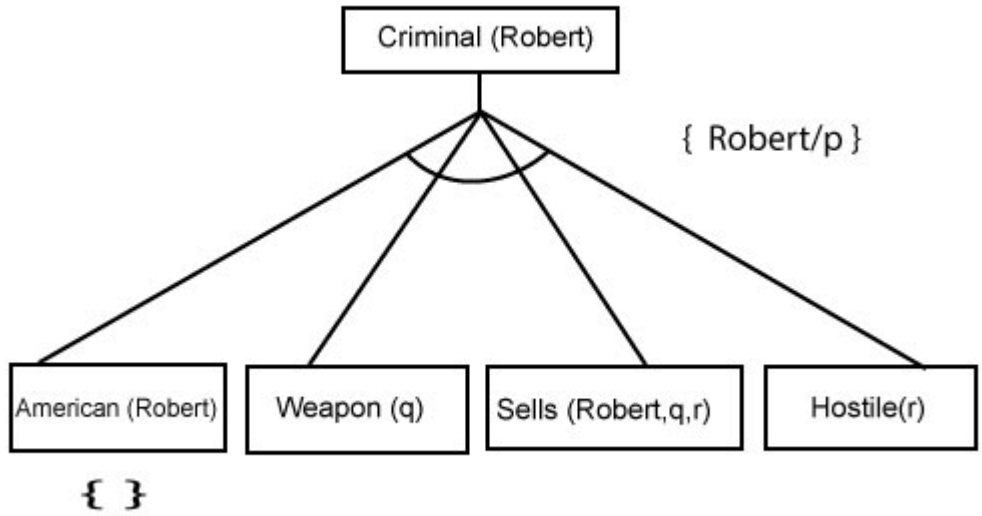
At the first step, we will take the goal fact. And from the goal fact, we will infer other facts, and at last, we will prove those facts true. So our goal fact is "Robert is Criminal," so following is the predicate of it.



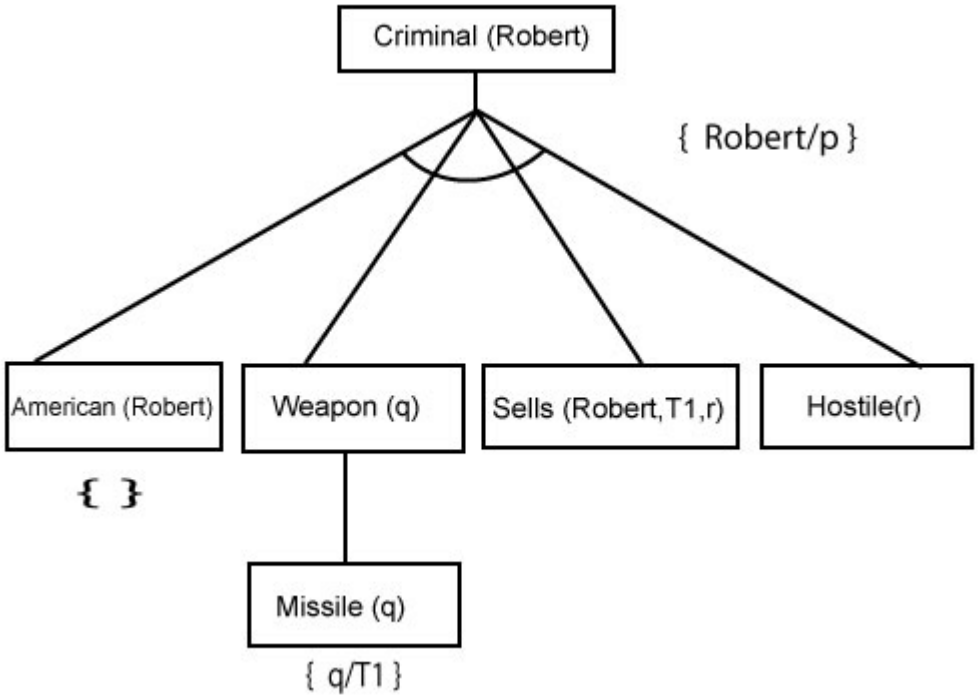
Step-2:

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.

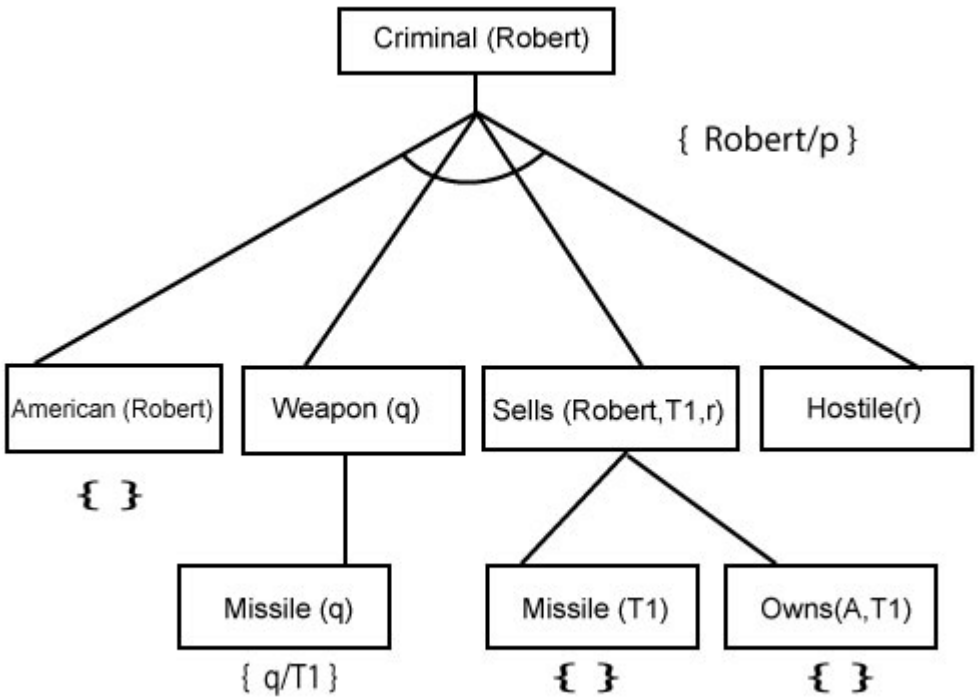


Step-3:t At step-3, we will extract further fact Missile(q) which infer from Weapon(q), as it satisfies Rule-(5). Weapon (q) is also true with the substitution of a constant T1 at q.



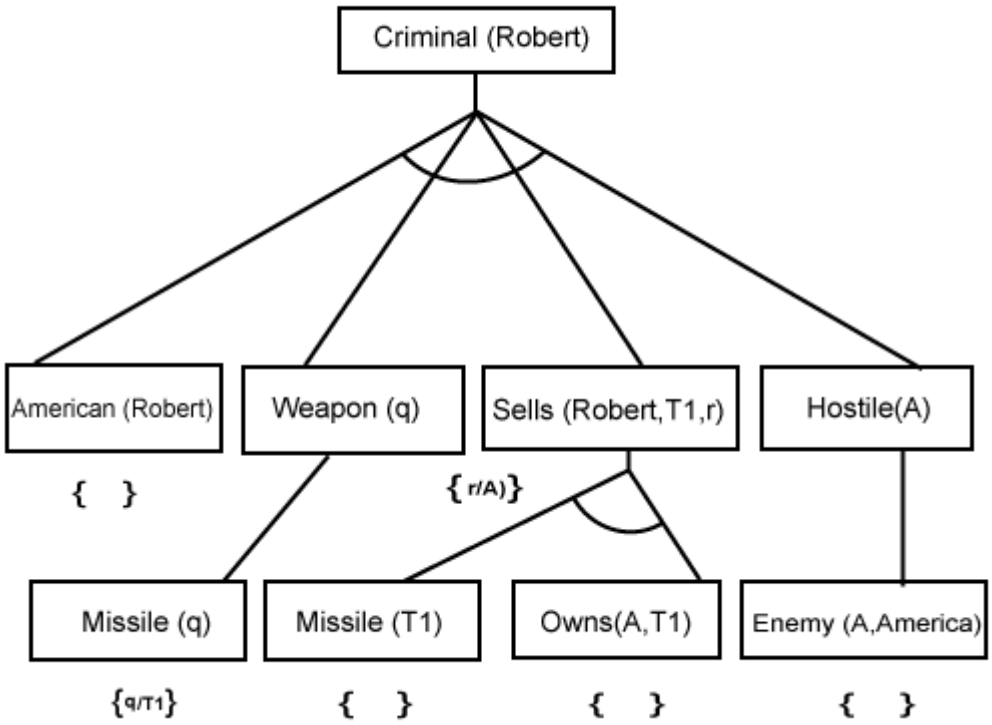
Step-4:

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.



Difference between backward chaining and forward chaining

Following is the difference between the forward chaining and backward chaining:

- Forward chaining as the name suggests, start from the known facts and move forward by applying inference rules to extract more data, and it continues until it reaches to the goal, whereas backward chaining starts from the goal, move backward by using inference rules to determine the facts that satisfy the goal.
- Forward chaining is called a **data-driven** inference technique, whereas backward chaining is called a **goal-driven** inference technique.
- Forward chaining is known as the **down-up** approach, whereas backward chaining is known as a **top-down** approach.

	it reaches to the goal.	required facts that support the goal.
2.	It is a bottom-up approach	It is a top-down approach
3.	Forward chaining is known as data-driven inference technique as we reach to the goal using the available data.	Backward chaining is known as goal-driven technique as we start from the goal and divide into sub-goal to extract the facts.
4.	Forward chaining reasoning applies a breadth-first search strategy.	Backward chaining reasoning applies a depth-first search strategy.
5.	Forward chaining tests for all the available rules	Backward chaining only tests for few required rules.
6.	Forward chaining is suitable for the planning, monitoring, control, and interpretation application.	Backward chaining is suitable for diagnostic, prescription, and debugging application.

- Forward chaining is slow as it checks for all the rules, whereas backward chaining is fast as it checks few required rules only.

7.	Forward chaining can generate an infinite number of possible conclusions.	Backward chaining generates a finite number of possible conclusions.
8.	It operates in the forward direction.	It operates in the backward direction.
9.	Forward chaining is aimed for any conclusion.	Backward chaining is only aimed for the required data.

Resolution

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**.

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.

So to represent uncertain knowledge, where we are not sure about the predicates, we need uncertain reasoning or probabilistic reasoning.

Causes of uncertainty:

Following are some leading causes of uncertainty to occur in the real world.

1. Information occurred from unreliable sources.
2. Experimental Errors

3. Equipment fault
4. Temperature variation
5. Climate change.

Probabilistic reasoning:

Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge. In probabilistic reasoning, we combine probability theory with logic to handle the uncertainty.

We use probability in probabilistic reasoning because it provides a way to handle the uncertainty that is the result of someone's laziness and ignorance.

In the real world, there are lots of scenarios, where the certainty of something is not confirmed, such as "It will rain today," "behaviour of someone for some situations," "A match between two teams or two players." These are probable sentences for which we can assume that it will happen but not sure about it, so here we use probabilistic reasoning.

Need of probabilistic reasoning in AI:

- When there are unpredictable outcomes.
- When specifications or possibilities of predicates becomes too large to handle.
- When an unknown error occurs during an experiment.

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

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

-
- **Event:** Each possible outcome of a variable is called an event.
- **Sample space:** The collection of all possible events is called sample space.

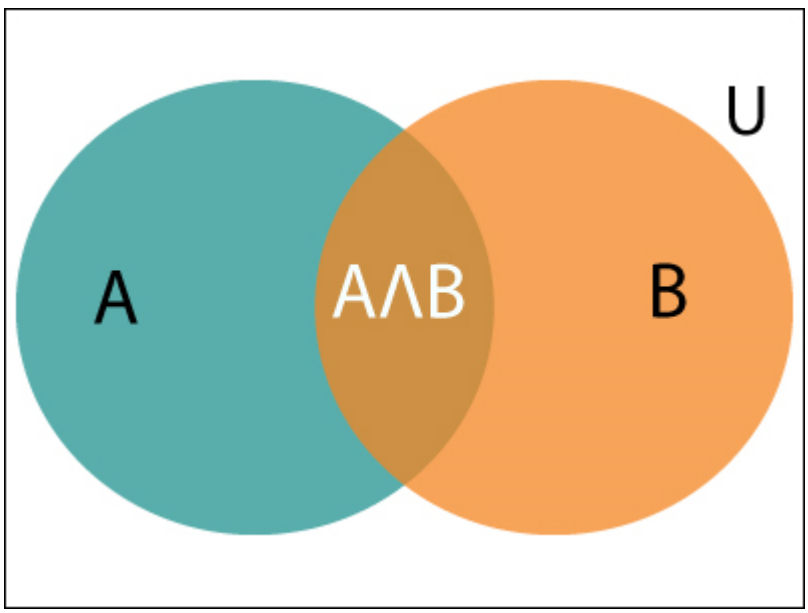
- **Random variables:** Random variables are used to represent the events and objects in the real world.
- **Prior probability:** The prior probability of an event is probability computed before observing new information.
- **Posterior Probability:** The probability that is calculated after all evidence 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:
- $$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

- **Where P($A \wedge B$)= 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:
- $$P(B|A) = \frac{P(A \wedge B)}{P(A)}$$

- 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$\wedge B$) by P(B)**.



- **Example:**

- In a class, there are 70% of the students who like English and 40% of the students who likes English and mathematics, and then what is the percent of students those who like English also like mathematics?
- **Solution:**
- Let, A is an event that a student likes Mathematics
- B is an event that a student likes English.
- $$P(A|B) = \frac{P(A \wedge B)}{P(B)} = \frac{0.4}{0.7} = 57\%$$
- **Hence, 57% are the students who like English also like Mathematics.**

Probabilistic reasoning is used in **AI** when we are unsure of the predicates, when the possibilities of predicates become too large to list down, when it is known that an error occurs during an experiment. Bayesian network is a directed acyclic graph model which helps us represent **probabilistic** data.

Bayesian Belief Network in artificial intelligence

Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty. We can define a Bayesian network as:

"A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph."

It is also called a **Bayes network, belief network, decision network**, or **Bayesian model**.

Bayesian networks are probabilistic, because these networks are built from a **probability distribution**, and also use probability theory for prediction and anomaly detection.

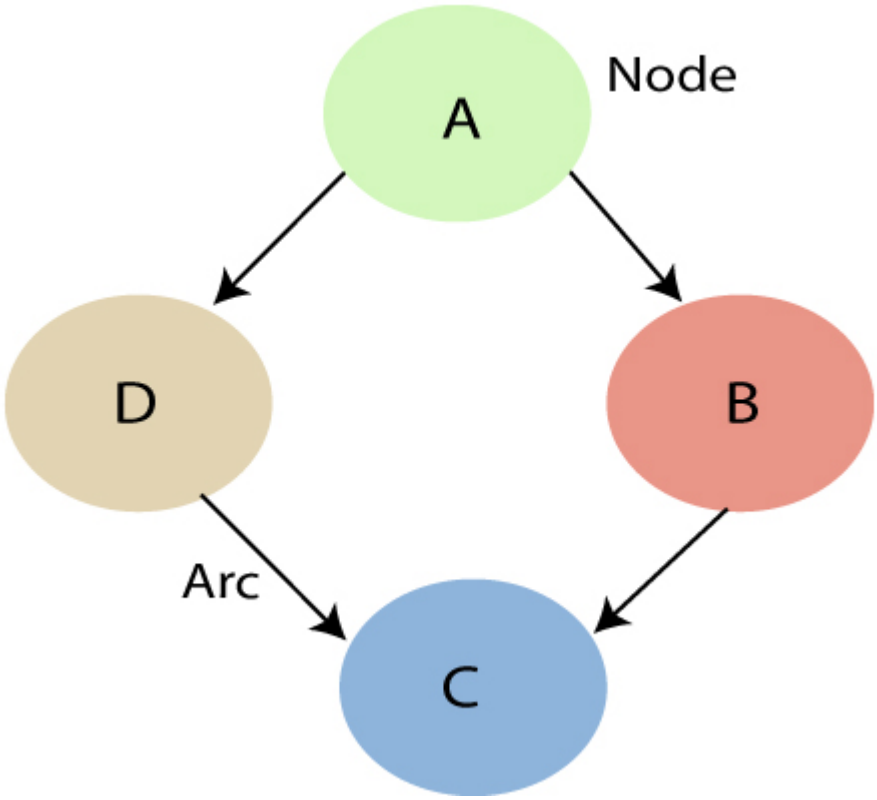
Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network. It can also be used in various tasks including **prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction**, and **decision making under uncertainty**.

Bayesian Network can be used for building models from data and experts opinions, and it consists of two parts:

- **Directed Acyclic Graph**
- **Table of conditional probabilities.**

The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an **Influence diagram**.

A Bayesian network graph is made up of nodes and Arcs (directed links), where:



- Each **node** corresponds to the random variables, and a variable can be **continuous** or **discrete**.
- **Arc or directed arrows** represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph.
  These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other
  - **In the above diagram, A, B, C, and D are random variables represented by the nodes of the network graph.**
  - **If we are considering node B, which is connected with node A by a directed arrow, then node A is called the parent of Node B.**
  - **Node C is independent of node A.**

> Note: The Bayesian network graph does not contain any cyclic graph. Hence, it is known as a **directed acyclic graph or DAG**.

The Bayesian network has mainly two components:

- **Causal Component**
- **Actual numbers**

Each node in the Bayesian network has condition probability distribution $\mathbf{P(X_i \mid Parent(X_i))}$, which determines the effect of the parent on that node.

Bayesian network is based on Joint probability distribution and conditional probability. So let's first understand the joint probability distribution:

Explanation of Bayesian network:

Let's understand the Bayesian network through an example by creating a directed acyclic graph:

Example: Harry installed a new burglar alarm at his home to detect burglary. The alarm reliably responds at detecting a burglary but also responds for minor earthquakes. Harry has two neighbors David and Sophia, who have taken a responsibility to inform Harry at work when they hear the alarm. David always calls Harry when he hears the alarm, but sometimes he got confused with the phone ringing and calls at that time too. On the other hand, Sophia likes to listen to high music, so sometimes she misses to hear the alarm. Here we would like to compute the probability of Burglary Alarm.

Problem:

Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Sophia both called the Harry.

Solution:

- The Bayesian network for the above problem is given below. The network structure is showing that burglary and earthquake is the parent node of the alarm and directly affecting the probability of alarm's going off, but David and Sophia's calls depend on alarm probability.
- The network is representing that our assumptions do not directly perceive the burglary and also do not notice the minor earthquake, and they also not confer before calling.
- The conditional distributions for each node are given as conditional probabilities table or CPT.
- Each row in the CPT must be sum to 1 because all the entries in the table represent an exhaustive set of cases for the variable.
- In CPT, a boolean variable with k boolean parents contains 2^K probabilities. Hence, if there are two parents, then CPT will contain 4 probability values

List of all events occurring in this network:

- **Burglary (B)**
- **Earthquake(E)**
- **Alarm(A)**
- **David Calls(D)**
- **Sophia calls(S)**

We can write the events of problem statement in the form of probability: **P[D, S, A, B, E]**, can rewrite the above probability statement using joint probability distribution:

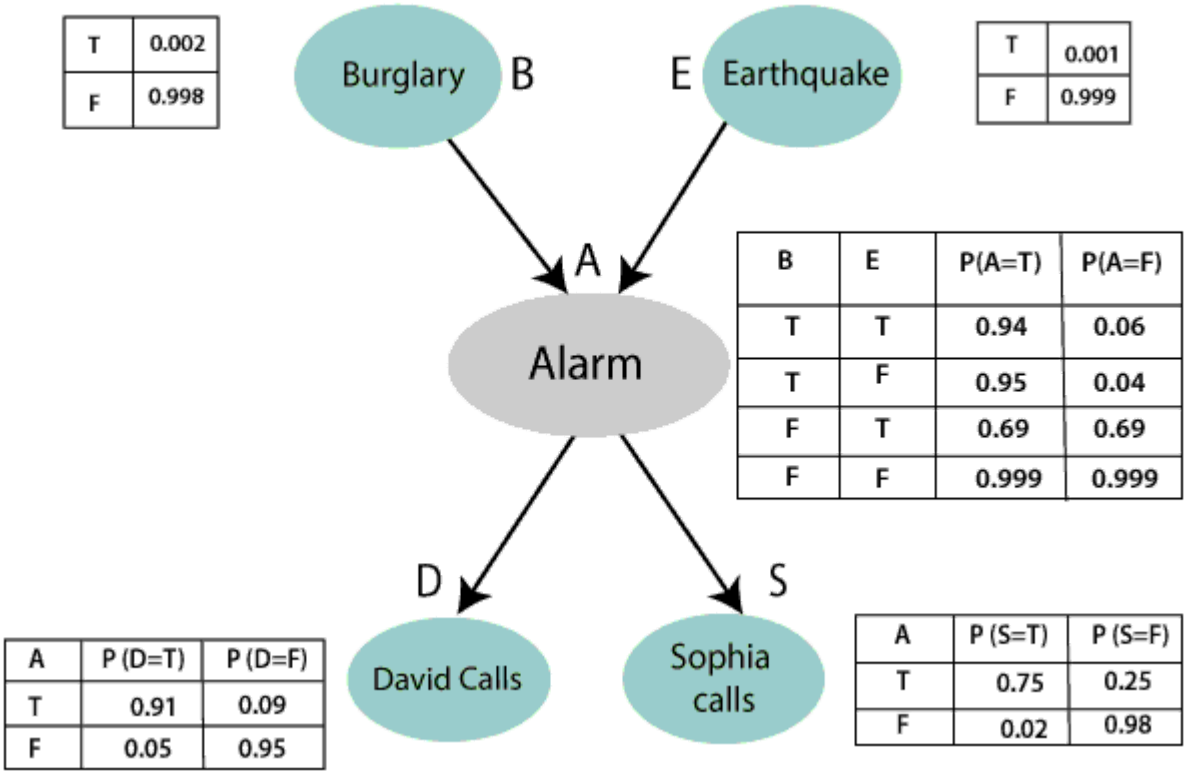
P[D, S, A, B, E]= P[D | S, A, B, E]. P[S, A, B, E]

=P[D | S, A, B, E]. P[S | A, B, E]. P[A, B, E]

= P [D| A]. P [S| A, B, E]. P[A, B, E]

= P[D | A]. P[S | A]. P[A| B, E]. P[B, E]

= P[D | A]. P[S | A]. P[A| B, E]. P[B |E]. P[E]



T	0.002
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

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.

We can provide the conditional probabilities as per the below tables:

Conditional probability table for Alarm A:

The Conditional probability of Alarm A depends on Burglar and earthquake:

A	P(S= True)	P(S= False)
True	0.75	0.25
False	0.02	0.98

True	False	0.95	0.04
False	True	0.31	0.69
False	False	0.001	0.999

Conditional probability table for David Calls:

A	P(D= True)	P(D= False)
True	0.91	0.09
False	0.05	0.95

The Conditional probability of David that he will call depends on the probability of Alarm.

Conditional probability table for Sophia Calls:

The Conditional probability of Sophia that she calls is depending on its Parent Node "Alarm."

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.

Hence, a Bayesian network can answer any query about the domain by using Joint distribution.

The semantics of Bayesian Network:

There are two ways to understand the semantics of the Bayesian network, which is given below:

1. To understand the network as the representation of the Joint probability distribution.

It is helpful to understand how to construct the network.

2. To understand the network as an encoding of a collection of conditional independence statements.

It is helpful in designing inference procedure.

Utility theory and its importance in AI

Utility theory is concerned with people's choices and decisions. It is concerned also with people's preferences and with judgments of preferability, worth, value, goodness or any of a number of similar concepts. Utility means quality of being useful. So as per this each state in environment has a degree of usefulness to an agent, that agent will prefer states with higher utility.

Decision Theory = Probability theory + Utility Theory.

Interpretations of utility theory are often classified under two headings, prediction and prescription:
(i) The predictive approach is interested in the ability of a theory to predict actual choice behaviour.
(ii) The prescriptive approach is interested in saying how a person ought to make a decision.

E.g : Psychologists are primarily interested in **prediction.**

Economists in both **prediction and prescription**. **In statistics** the emphasis is on **prescription in decision making** under uncertainty. The emphasis in **management science is prescriptive** also.
Sometimes it is useful to ignore uncertainty, focus on ultimate choices. Other times, must model uncertainty explicitly. **Examples: Insurance markets, Financial markets., Game theory.** Rather than choosing outcome directly, decision-maker chooses uncertain prospect (or lottery). **A lottery is a probability distribution over outcomes**.
Expected Utility : Expected utility of action A , given evidence E , E ∪(A | E) is calculated as follows **: E ∪(A | E) = Σ (Result i(A) | D0 (A) , E) ∪ ((** (**)**), where ,
P (Resulti (A) | D0 (A)) is probability assigned by agent for action A to be executed.
D0(A) : Proposition that A is executed in current state.
This has two basic components; consequences (or outcomes) and lotteries.
(a) **Consequences**: These are what the decision-maker ultimately cares about.

Example: **"I get pneumonia, my health insurance company covers most of the costs, but I have to pay a $500 deductible."** Consumer does not choose consequences directly.
Lotteries Consumer chooses a lottery, p

(b) **Lotteries are probability distributions** over consequences: p : C → [0, 1] ;

with Σc ∈ C p (c) = 1. Set of all lotteries is denoted by P. **Example: "A gold-level health insurance plan, which covers all kinds of diseases, but has a $500 deductible."** Makes sense because consumer assumed to rank health insurance plans only insofar as lead to different probability distributions over consequences

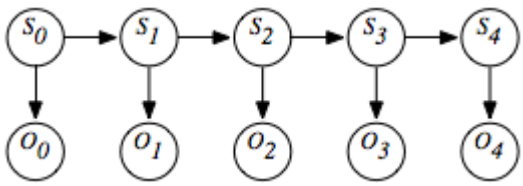
Utility Function : U : P → R has an expected utility form if there exists a function u : C → R such that U (p) = Σ p (c) u (c) for all p ∈ P. c ∈ C . In this case, the **function U is called an expected utility function, and the function u is call a von Neumann-Morgenstern utility function.** These functions are used to capture agent's preferences between various world states .This **f**unction assigns a single number to express desirability of a state utilities**.** Utilities are combined with outcome probabilities of actions to give an expected utility for each action. U (s) : Means utility of state S , for agent's Decision.

Maximum expected Utility (MEU) : This represents that a rational agent should select an action that maximizes the agent's expected utility. MEU principle says **" If an agent maximizes a utility function that correctly reflects the performance measure by which its behavior is being judged , then it will achieve the highest possible performance score if we average over the environment of agent."**

Hidden Markov Models –

HMMs are probabilistic models. They allow us to compute the joint probability of a set of hidden states given a set of observed states. The hidden states are also referred to as latent states. Once we know the joint probability of a sequence of hidden states, we determine the best possible sequence i.e. the sequence with the highest probability and choose that sequence as the best sequence of hidden states. A **hidden Markov model (HMM)** is an augmentation of the Markov chain to include observations. Just like the state transition of the Markov chain, an HMM also includes observations of the state. These observations can be **partial** in that different states can map to the same observation and **noisy** in that the same state can stochastically map to different observations at different times.

The assumptions behind an HMM are that the state at time $t+1$ only depends on the state at time t, as in the Markov chain. The observation at time t only depends on the state at time t. The observations are modeled using the variable O_t for each time t whose domain is the set of possible observations.
The belief network representation of an HMM is depicted in. Although the belief network is shown for four stages, it can proceed indefinitely.



A stationary HMM includes the following probability distributions:

- $P(S_0)$ specifies initial conditions.
- $P(S_{t+1}|S_t)$ specifies the dynamics.
- $P(O_t|S_t)$ specifies the sensor model.

There are a number of tasks that are common for HMMs.

The problem of **filtering** or belief-state **monitoring** is to determine the current state based on the current and previous observations, namely to determine

$P(S_i|O_0,...,O_i)$.

Note that all state and observation variables after S_i are irrelevant because they are not observed and can be ignored when this conditional distribution is computed.
The problem of **smoothing** is to determine a state based on past and future observations. Suppose an agent has observed up to time k and wants to determine the state at time i for $i<k$; the smoothing problem is to determine

$P(S_i|O_0,...,O_k)$.

All of the variables S_i and V_i for $i>k$ can be ignored.
The problem of **filtering** or belief-state **monitoring** is to determine the current state based on the current and previous observations, namely to determine

$P(S_i|O_0,...,O_i)$.

Note that all state and observation variables after S_i are irrelevant because they are not observed and can be ignored when this conditional distribution is computed.
The problem of **smoothing** is to determine a state based on past and future observations. Suppose an agent has observed up to time k and wants to determine the state at time i for $i<k$; the smoothing problem is to determine

$P(S_i|O_0,...,O_k)$.

All of the variables S_i and V_i for $i>k$ can be ignored.
The problem of **filtering** or belief-state **monitoring** is to determine the current state based on the current and previous observations, namely to determine

$P(S_i|O_0,...,O_i)$.

Note that all state and observation variables after S_i are irrelevant because they are not observed and can be ignored when this conditional distribution is computed.
The problem of **smoothing** is to determine a state based on past and future observations. Suppose an agent has observed up to time k and wants to determine the state at time i for $i<k$; the smoothing problem is to determine

$P(S_i|O_0,...,O_k)$.

All of the variables S_i and V_i for $i>k$ can be ignored.