**Representing simple fact in logic**

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:
  + **Syntax**
  + **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)**.

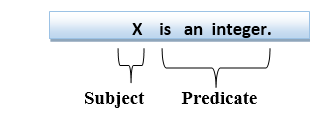
**Complex Sentences:**

* 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:
  1. **Universal Quantifier, (for all, everyone, everything)**
  2. **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 "→".

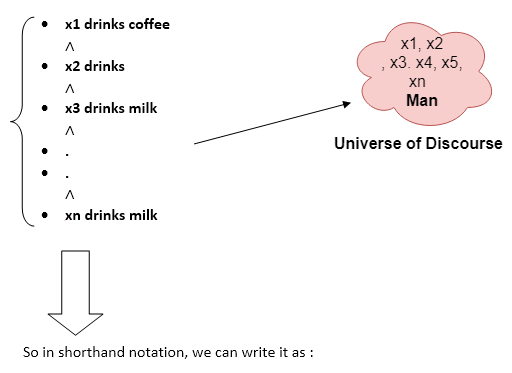
If x is a variable, then ∀x is read as:

* **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.

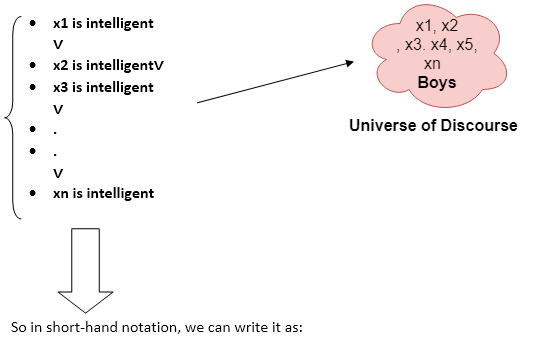
Note: In Existential quantifier we always use AND or Conjunction symbol (∧).

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.

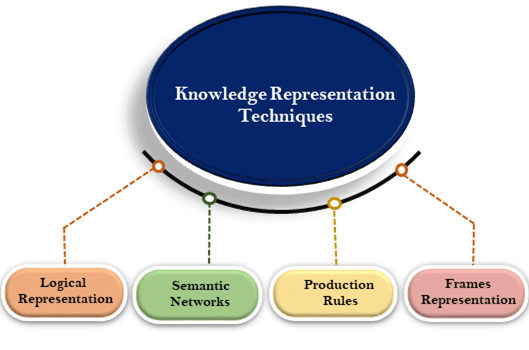
**Example: ∀x [A (x) B( y)], here x and y are the bound variables.**

**Representing instant & ISA relationship**

**Techniques of knowledge representation**

There are mainly four ways of knowledge representation which are given as follows:

1. Logical Representation
2. Semantic Network Representation
3. Frame Representation
4. Production Rules



**1. 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 categorised into mainly two logics:

1. Propositional Logics
2. Predicate logics

Note: We will discuss Prepositional Logics and Predicate logics in later chapters.

Advantages of logical representation:

1. Logical representation enables us to do logical reasoning.
2. Logical representation is the basis for the programming languages.

**Disadvantages of logical Representation:**

1. Logical representations have some restrictions and are challenging to work with.
2. Logical representation technique may not be very natural, and inference may not be so efficient.

Note: Do not be confused with logical representation and logical reasoning as logical representation is a representation language and reasoning is a process of thinking logically.

**2. 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.

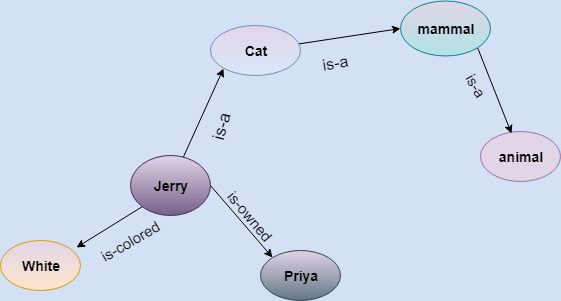
This representation consist of mainly two types of relations:

1. IS-A relation (Inheritance)
2. Kind-of-relation

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

**Statements:**

1. Jerry is a cat.
2. Jerry is a mammal
3. Jerry is owned by Priya.
4. Jerry is brown colored.
5. All Mammals are animal.



In the above diagram, we have represented the different type of knowledge in the form of nodes and arcs. Each object is connected with another object by some relation.

**Drawbacks in Semantic representation:**

1. Semantic networks take more computational time at runtime as we need to traverse the complete network tree to answer some questions. It might be possible in the worst case scenario that after traversing the entire tree, we find that the solution does not exist in this network.
2. Semantic networks try to model human-like memory (Which has 1015 neurons and links) to store the information, but in practice, it is not possible to build such a vast semantic network.
3. These types of representations are inadequate as they do not have any equivalent quantifier, e.g., for all, for some, none, etc.
4. Semantic networks do not have any standard definition for the link names.
5. These networks are not intelligent and depend on the creator of the system.

**Advantages of Semantic network:**

1. Semantic networks are a natural representation of knowledge.
2. Semantic networks convey meaning in a transparent manner.
3. These networks are simple and easily understandable.

**3. 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: 1**

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 |

**Example 2:**

Let's suppose we are taking an entity, Peter. Peter is an engineer as a profession, and his age is 25, he lives in city London, and the country is England. So following is the frame representation for this:

|  |  |
| --- | --- |
| **Slots** | **Filter** |
| **Name** | Peter |
| **Profession** | Doctor |
| **Age** | 25 |
| **Marital status** | Single |
| **Weight** | 78 |

**Advantages of frame representation:**

1. The frame knowledge representation makes the programming easier by grouping the related data.
2. The frame representation is comparably flexible and used by many applications in AI.
3. It is very easy to add slots for new attribute and relations.
4. It is easy to include default data and to search for missing values.
5. Frame representation is easy to understand and visualize.

**Disadvantages of frame representation:**

1. In frame system inference mechanism is not be easily processed.
2. Inference mechanism cannot be smoothly proceeded by frame representation.
3. Frame representation has a much generalized approach.

**4. 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).**

**Advantages of Production rule:**

1. The production rules are expressed in natural language.
2. The production rules are highly modular, so we can easily remove, add or modify an individual rule.

**Disadvantages of Production rule:**

1. Production rule system does not exhibit any learning capabilities, as it does not store the result of the problem for the future uses.
2. During the execution of the program, many rules may be active hence rule-based production systems are inefficient.