UNIT-3 KNOWLEDGE REPRESENTATION

What is Knowledge?

In order to solve the complex problems encountered in AI, one generally needs a large amount of knowledge, and suitable mechanisms for representing and manipulating all that knowledge.

Knowledge can take many forms.

Some simple examples are:

1. John has an umbrella
2. It is raining
3. An umbrella stops you getting wet when it’s raining
4. An umbrella will only stop you getting wet if it is used properly
5. Umbrellas are not so useful when it is very windy

So, how should an AI agent store and manipulate knowledge like this?

What is a Knowledge Representation?

The object of a knowledge representation is to express knowledge in a computer tractable form, so that it can be used to enable our AI agents to perform well.

Knowledge Representation is a subarea of Artificial Intelligence concerned with understanding, designing and implementing ways of representing information in computers so that programs can use this information

- To derive information that is implied by it,
- To converse with people in natural language,
- To decide what to do next,
- To plan future activities,
- To solve problems in areas that normally requires human expertise.

A knowledge representation language is defined by two aspects:

1. Syntax: The syntax of a language defines which configurations of the components of the language constitute valid sentences.

2. Semantics: The semantics defines which facts in the world the sentences refer to, and hence the statement about the world that each sentence makes.

Approaches to Knowledge Representation

A good system for the representation of knowledge in a particular domain should possess the following four properties:

1. Representational Adequacy – the ability to represent all the different kinds of knowledge that might be needed in that domain.

2. Inferential Adequacy – the ability to manipulate the representational structures to derive new structures (corresponding to new knowledge) from existing structures.

3. Inferential Efficiency – the ability to incorporate additional information into the knowledge structure which can be used to focus the attention of the inference mechanisms in the most promising directions.

4. Acquisitional Efficiency – the ability to acquire new information easily. Ideally the agent should be able to control its own knowledge acquisition, but direct insertion of information by a ‘knowledge engineer’ would be acceptable.

Unfortunately, no single system optimizes all of the capabilities for all kinds of knowledge have yet been found. As a result, multiple techniques for knowledge representation exist. Many programs rely on more than one technique.

Relational Knowledge

- Relational knowledge provides a framework to compare two objects based on equivalent attributes.
- It is the simplest way of storing facts which uses the relational method, and each fact about a set of the object is set out systematically in columns.
- This approach of knowledge representation is famous in database systems where the relationship between different entities is represented.
- This approach has little opportunity for inference.

Example 1: The following is the simple relational knowledge representation.

Player	**Weight (kg)**	**Height (centimeters)**
Player1	65	160.02
Player2	58	168.11
Player3	75	164.29

Given the above table it is not possible to answer simple question such as:

"Who is the heaviest player?"

But if a procedure for finding heaviest player is provided, then these facts will enable that procedure to compute an answer.

Example 2: The following is another relational knowledge representation.

Musician	**Style**	**Instrument**	**Age**
Davis	Jazz	Trumpot	40
John	Avant Garde	Saxophone	35
Frank	Rock	Guitar	55
Milli	Jazz	Guitar	47

- We can ask things like:

  Who is below 40 years in age?
  Who plays Jazz?

- This sort of representation is popular in database systems

First Order Logic

The syntax and semantics of first order logic will be covered in detail in Unit-2.

Some typical sentences in first order logic are:

1. $\text{man}(\text{William}) \vee \text{woman}(\text{Susan})$
2. $\text{married}(\text{William}, \text{Susan})$
3. $\forall x \exists y[\text{person}(x) \Rightarrow \text{has\_mother}(x,y)]$
4. $\forall x \forall y[[\text{parents}(x,y) \wedge \text{man}(x)] \Rightarrow \neg\text{man}(y)$

The language consists of constants {William, Susan, etc.}, variables {x, y, etc.}, functions/predicates {Married(x,y), person(x), etc.}, and the logic symbols:

Logic	$\vee$	$\wedge$	$\Rightarrow$	$\neg$	$\forall$	$\exists$
Natural Language	or	and	implies	not	for all	there exists

We can also manipulate the logic representations to generate new knowledge.

First Order Logic as a Knowledge Representation

We can combine sentences by the 'rules of logic' to produce new sentences, e.g.

$$\begin{array}{l} \neg\text{man}(\text{Chris}) \\ \neg\text{man}(x) \Rightarrow \text{woman}(x) \\ \hline \text{woman}(\text{Chris}) \end{array}$$

As a knowledge representation, first order logic has pros and cons:

Advantages

1. It is very expressive.
2. It has unambiguous syntax and semantics.

Disadvantage

1. There is no generally efficient procedure for processing knowledge.

Procedural Knowledge

Procedural knowledge approach uses small programs and codes which describes how to do specific things, and how to proceed.

The procedural knowledge:

- may have inferential efficiency, but no inferential adequacy and acquisitional efficiency.
- Are represented as small programs that know how to do specific things, how to proceed.

Example: The machine uses the knowledge represented as codes to perform a task.

Knowledge Representation using Semantic Network

- Semantic networks (semantic net) are a commonly used representation in AI. In Semantic net, we can represent our knowledge in the form of graphical networks.
- A semantic network is a graph consisting of nodes that are connected by edges.
- Nodes represent objects, and the links between nodes represent relationships between those objects. The links are usually labelled to indicate the nature of the relationship.

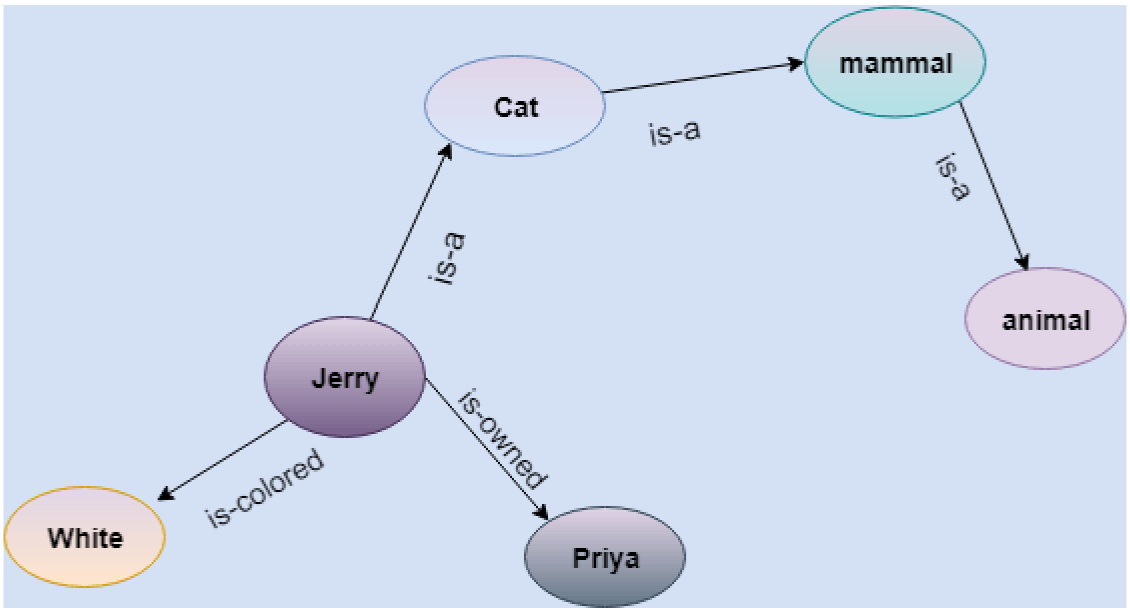
This representation consists of mainly two types of relations:

a. IS-A relation (Inheritance)
b. Kind-of-relation

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

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 edges. 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.

Inheritance in Semantic Net

- Inheritance allows us to specify properties of a superclass and then to define a subclass, which inherits the properties of the superclass.
- Example: If we say that all mammals give birth to live babies and we also say that all dogs are mammals and that Tommy is a dog then we can conclude that Tommy gives birth to live mammals.
  In our example, mammals are the superclass of dogs and Tommy. Dogs are the subclass of mammals and superclass of Tommy.
- Although inheritance is a useful way to express generalization about a class of objects, in some cases we need to express exceptions to those generalizations such as “Male animals do not give birth” or “Female dogs below the age of 6 months do not give birth”.
  In such cases, we say that the default value has been overridden in the subclass.

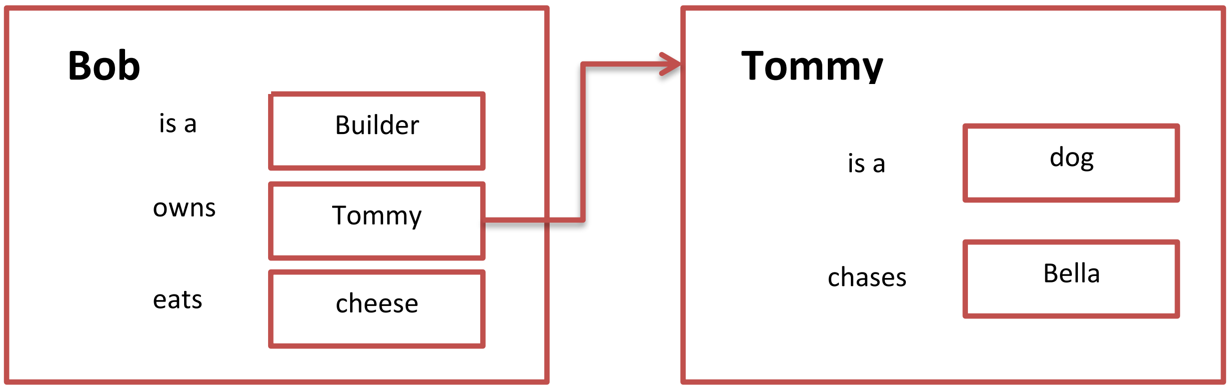
Frames

Frame based representation is a development of semantic nets and allow us to express the idea of inheritance.

A Frame System consists of a set of frames (or nodes), which are connected together by relations. Each frame describes either an instance or a class. Each frame has one or more slots, which are assigned slot values. This is the way in which the frame system is built up. Rather than simply having links between frames, each relationship is expressed by a value being placed in a slot. Example:

Frame Name	**Slot**	**Slot Value**
Bob	is a	Builder
	owns	Tommy
	eats	Cheese
Tommy	is a	dog
	chases	Bella
Bella	is a	Cat
	chases	Mice
Mice	eat	cheese
Cheese		
Builder		
Dog		
Cat		

We can also represent this frame system in a diagrammatic form such a below:



When we say, “Tommy is a dog” we really mean, “Tommy is an instance of the class dog” or “Tommy is a member of the class dogs”.

Why are Frames useful?

The main advantage of using frame-based systems for expert systems is that all information about a particular object is stored in one place.

Inheritance in Frames:

Example: 1

We might extend our frame system with the following additional information:

Dogs chase cats
Cats chase mice
Now, we do not need to state explicitly that Tommy chases Bella or that Tommy chases mice. We can inherit this information because Bella is an instance of the class cats, and Tommy is an instance of class dogs.

Example 2:

Mammals breathe
Dogs are mammals
Cats are mammals
Here we have created superclass mammals, of which dogs and cats are subclasses. Hence, we do not need to explicitly ay that cats and dogs breathe because we can inherit this information. Similarly we do not need to express explicitly that Tommy and Bella breathe as they are instances of the class dogs and cats and therefore they inherit from those classes (superclasses).

Example 3: Let’s take a fact:

Mammals have four legs

Now, this is not true as humans do not have four legs.
Or let’s say Tommy has an unfortunate accident and now has only three legs.
This information might be expressed as:

Frame Name	**Slot**	**Slot Value**
Mammals	*no. of legs	four
Dog	subclass	mammal
Cat	subclass	mammal
Tommy	is a	dog
	number of legs	three
Bella	is a	cat

We have used an asterisk (*) to indicate that the value for the “number of legs” slot for the mammal class is a default value and can be overridden.

Slots as Frames:

- It is also possible to express a range of values that a slot can take. Ex, the no. of legs slot might be allowed a number between 1 & 4.
- Or we can express this restriction by allowing slots to be frames. The no. of legs slot can be represented as a frame such as.

Frame Name	**Slot**	**Slot Value**
No. of legs	Minimum value Maximum value	1 4

Multiple Inheritance in Frame System:

It is possible for a frame to inherit properties from more than one frame, i.e. a class can be a subclass of two superclasses and an object can be an instance of more than one class. This is known as **Multiple Inheritance**.

Example:

Frame Name	**Slot**	**Slot Value**
Human	subclass	Mammal
	No. of legs	two
Builder	builds	houses
Bob	is a	Human
	is a	Builder

Hence, we can inherit the following information about Bob:
He has two legs
He builds houses

In some cases, we encounter conflicts, where multiple inheritance leads us to conclude contradictory information about a frame.

Example:

Frame Name	**Slot**	**Slot Value**
Cheese	is	smelly
Things wrapped in foil	is	not smelly
Cheddar	is a	Cheese
	is a	Things wrapped in foil

Here, cheddar is a type of cheese and that it comes wrapped in foil. Cheddar should inherit its smelliness from the cheese class, but it also inherits non-smelliness from the Things wrapped in foil class.

Thus, we need a mechanism to decide which features to inherit from which superclass.

- One simple method is to say that conflicts are resolved by the order in which they appear.
  So, if a fact is established by instances and then that fact is contradicted by inheritance, the first fact is kept because it appeared first and the contradiction is discarded.
- However, it would be better to build the frame system such that conflicts of this kind cannot occur.