ARTIFICIAL INTELLIGENCE

UNIT – 2

Knowledge

Knowledge is a familiarity, awareness, or understanding of someone or something, such as facts(propositional knowledge), skills(Procedural knowledge) or objects (acquaintance knowledge) . Knowledge is awareness or familiarity gained by experiences of facts, data, and situations.

Knowledge Representation

Knowledge Representation in AI describes the representation of knowledge. Basically, it is a study of how the beliefs, intentions, and judgments of an intelligent agent can be expressed suitably for automated reasoning.

Knowledge Representation and Reasoning (KR, KRR) represents information from the real world for a computer to understand and then utilize this knowledge to solve complex real-life problems like communicating with human beings in natural language. Knowledge representation in AI is not just about storing data in a database, it allows a machine to learn from that knowledge and behave intelligently like a human being.

Various Approaches of Knowledge Representation

There are mainly four approaches to knowledge representation, which are given below:

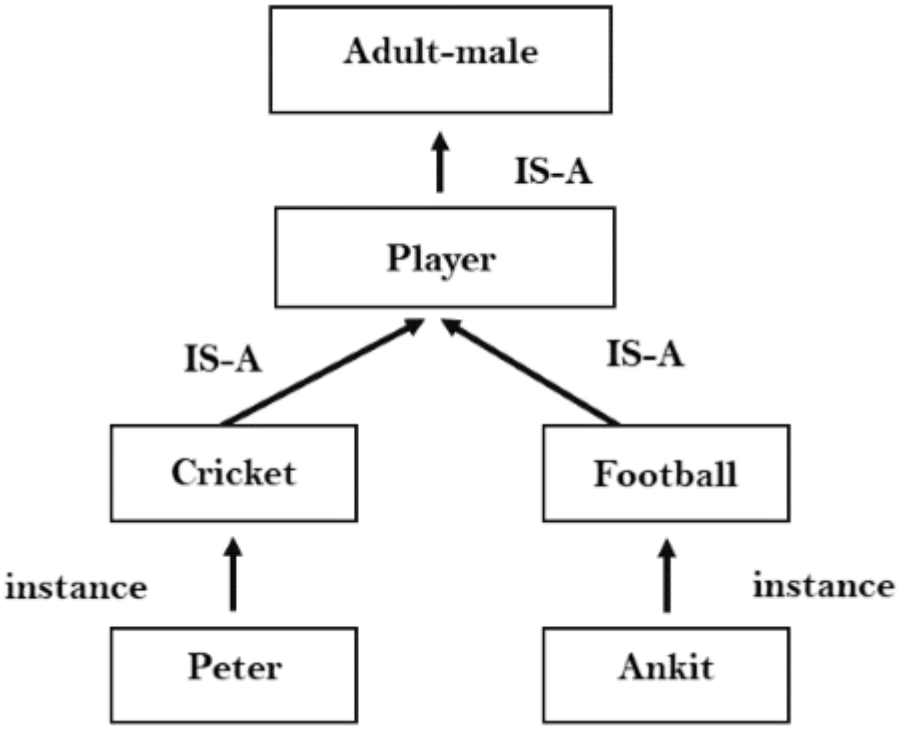
1. Simple relational knowledge:

- 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.

2. Inheritable knowledge:

- In the inheritable knowledge approach, all data must be stored into a hierarchy of classes.
- All classes should be arranged in a generalized form or a hierarchal manner.

- In this approach, we apply inheritance property.
- Elements inherit values from other members of a class.
- This approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation.
- Every individual frame can represent the collection of attributes and its value.
- In this approach, objects and values are represented in Boxed nodes.
- We use Arrows which point from objects to their values.



3. Inferential knowledge:

- Inferential knowledge approach represents knowledge in the form of formal logics.
- This approach can be used to derive more facts.
- It guaranteed correctness.
- **Example:** Let's suppose there are two statements:
    a. Marcus is a man
    b. All men are mortal
    Then it can represent as;

man(Marcus)

∀x = man (x) ----------> mortal (x)s

4. Procedural knowledge:

- Procedural knowledge approach uses small programs and codes which describes how to do specific things, and how to proceed.
- In this approach, one important rule is used which is **If-Then rule**.
- In this knowledge, we can use various coding languages such as **LISP language** and **Prolog language**.
- We can easily represent heuristic or domain-specific knowledge using this approach.
- But it is not necessary that we can represent all cases in this approach.

Issues in Knowledge representation

The issues that arise while using KR techniques are many. Some of these are explained below:

1. **Important Attributed:**

Any attribute of objects so basic that they occur in almost every problem domain?

There are two attributed "instance" and "isa", that are general significance. These attributes are important because they support property inheritance.

2. **Relationship among attributes:**

The attributes we use to describe objects are themselves entities that we represent.

The relationship between the attributes of an object, independent of specific knowledge they encode, may hold properties like:

1. Inverse — This is about consistency check, while a value is added to one attribute. The entities are related to each other in many different ways.
2. Existence in an isa hierarchy — This is about generalization-specification, like, classes of objects and specialized subsets of those classes, there are attributes and specialization of attributes. For example, the attribute height is a specialization of general attribute physical-size which is, in turn, a specialization of physical-attribute. These generalization-specialization relationships are important for attributes because they support inheritance.
3. Technique for reasoning about values — This is about reasoning values of attributes not given explicitly. Several kinds of information are used in reasoning, like, height: must be in a unit of length, Age: of a person cannot be greater than the age of person's parents. The values are often specified when a knowledge base is created.

4. Single valued attributes — This is about a specific attribute that is guaranteed to take a unique value. For example, a baseball player can at time have only a single height and be a member of only one team. KR systems take different approaches to provide support for single valued attributes.

3. Choosing Granularity:

Regardless of the KR formalism, it is necessary to know:

- At what level should the knowledge be represented and what are the primitives?
- Should there be a small number or should there be a large number of low-level primitives or High-level facts.
- High-level facts may not be adequate for inference while Low-level primitives may require a lot of storage.

4. Set of objects:

How should sets of objects be represented?

There are certain properties of objects that are true as member of a set but not as individual;

The reason to represent sets of objects is: if a property is true for all or most elements of a set, then it is more efficient to associate it once with the set rather than to associate it explicitly with every elements of the set.

5. Finding Right structure:

Given a large amount of knowledge stored in a database, how can relevant parts are accessed when they are needed?

This is about access to right structure for describing a particular situation.

This requires, selecting an initial structure and then revising the choice.

Predicate logic or First-order predicate logic

- Predicate logic or First-order predicate 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,

- **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 etc.

Representation of Simple Facts in Logic

Simple Facts can be represented using logics in two ways:

1. Propositional logic: It is useful because it is simple to deal with and a decision procedure for it exists. Also, In order to draw conclusions, facts are represented in a more convenient way as,

1. Marcus is a man. man(Marcus)

2. Plato is a man. man(Plato)

3. All men are mortal. mortal(men)

• But propositional logic fails to capture the relationship between an individual being a man and that individual being a mortal.

How can these sentences be represented so that we can infer the third sentence from the first two? Also, Propositional logic commits only to the existence of facts that may or may not be the case in the world being represented. Moreover, It has a simple syntax and simple semantics. It suffices to illustrate the process of inference.

Propositional logic quickly becomes impractical, even for very small worlds.

2. Predicate logic First-order Predicate logic (FOPL): It models the world in terms of

- Objects, which are things with individual identities.
- Properties of objects that distinguish them from other objects.
- Relations that hold among sets of objects
- Functions, which are a subset of relations where there is only one "value" for any given "input"
- First-order Predicate logic (FOPL) provides:
    - Constants: a, b, dog33. Name a specific object.

- Variables: X, Y. Refer to an object without naming it.
- Functions: Mapping from objects to objects.
- Terms: Refer to objects
- Atomic Sentences: in(dad-of(X), food6) Can be true or false, Correspond to propositional• symbols P, Q.

A well-formed formula (wff) is a sentence containing no “free” variables. So, That is, all variables are “bound” by universal or existential quantifiers. (∀x)P(x, y) has x bound as a universally quantified variable, but y is free

Is-a and instance relationships

- Two attributes is-a and instance play an important role in many aspects of knowledge representation.
- The reason for this is that they support property inheritance.
  Is-a
- -- used to show class inclusion, e.g. isa(mega_star,rich).

- instance
- -- used to show class membership, e.g. instance(prince,mega_star).

Example : A simple sentence like "Joe is a musician" Here "is a" (called IsA) is a way of expressing what logically is called a class-instance relationship between the su bjects represented by the terms "Joe" and "musician". ◊ "Joe" is an instance of the class of things called "musician". " Joe" plays the role of instance, "musician" plays the role of class in that sentence.

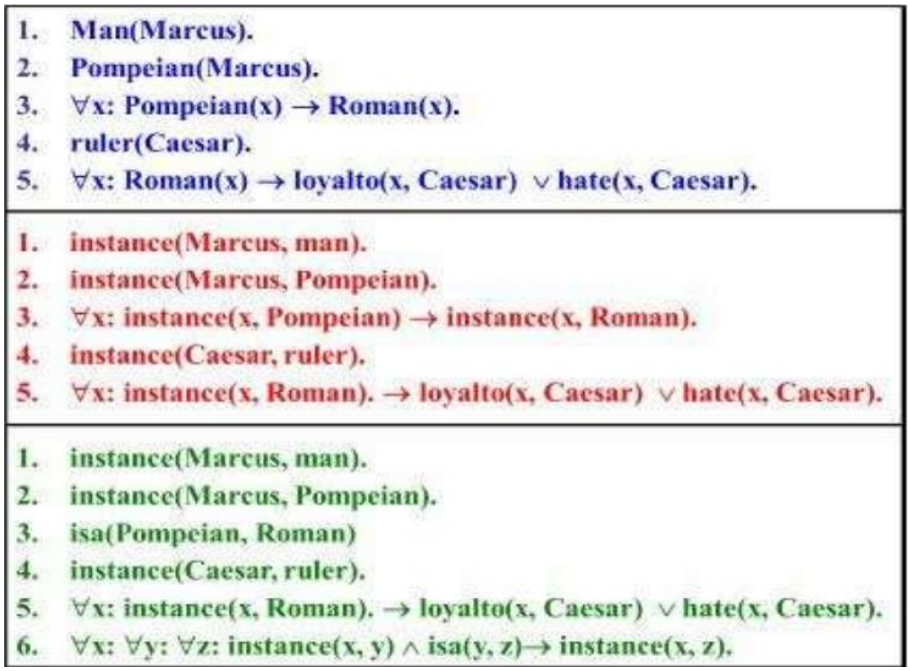


Figure: Three ways of representing class membership: ISA Relationships

- The predicate instance is a binary one, whose first argument is an object and whose second argument is a class to which the object belongs.
- But these representations do not use an explicit isa predicate.
- Instead, subclass relationships, such as that between Pompeians and Romans, described as shown in sentence 3.
- The implication rule states that if an object is an instance of the subclass Pompeian then it• is an instance of the superclass Roman.
  Note that this rule is equivalent to the standard set-theoretic definition of the subclasssuperclass relationship.
- The third part contains representations that use both the instance and isa predicates explicitly.
- The use of the isa predicate simplifies the representation of sentence 3, but it requires that one additional axiom (shown here as number 6) be provided.

Computable Functions and Predicates

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

Consider the following set of facts, again involving Marcus:

1) Marcus was a man. man(Marcus)

2) Marcus was a Pompeian. Pompeian(Marcus)

3) Marcus was born in 40 A.D. born(Marcus, 40)

4) All men are mortal. x: man(x) → mortal(x)

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

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

6) No mortal lives longer than 150 years. x: t1: At2: mortal(x) born(x, t1) gt(t2 – t1,150) → died(x, t2)

7) It is now 1991

now = 1991

So, Above example shows how these ideas of computable functions and predicates can be useful. It also makes use of the notion of equality and allows equal objects to be substituted for each other whenever it appears helpful to do so during a proof.

So, Now suppose we want to answer the question

"Is Marcus alive?"

- The statements suggested here, there may be two ways of deducing an answer.
- Either we can show that Marcus is dead because he was killed by the volcano or we can show that he must be dead because he would otherwise be more than 150 years old, which we know is not possible.
- Also, As soon as we attempt to follow either of those paths rigorously, however, we discover, just as we did in the last example, that we need some additional knowledge. For example, our statements talk about dying, but they say nothing that relates to being alive, which is what the question is asking.
  So we add the following facts:
  8) Alive means not dead. x: t: [alive(x, t) → ¬ dead(x, t)] [¬ dead(x, t) → alive(x, t)]
  9) If someone dies, then he is dead at all later times. x: t1: At2: died(x, t1) gt(t2, t1) → dead(x, t2)
  So, Now let's attempt to answer the question "Is Marcus alive?" by proving: ¬ alive(Marcus, now)