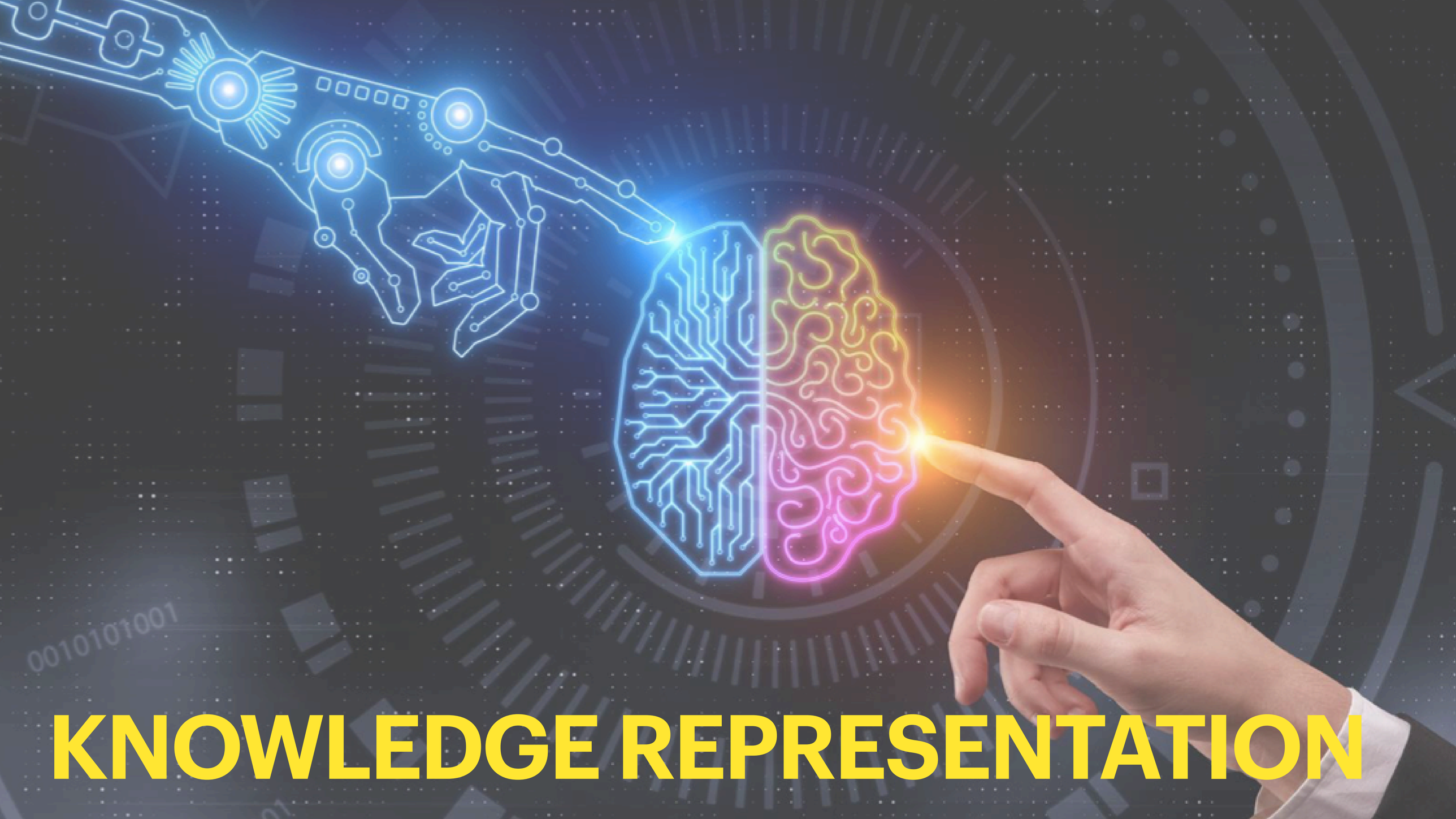
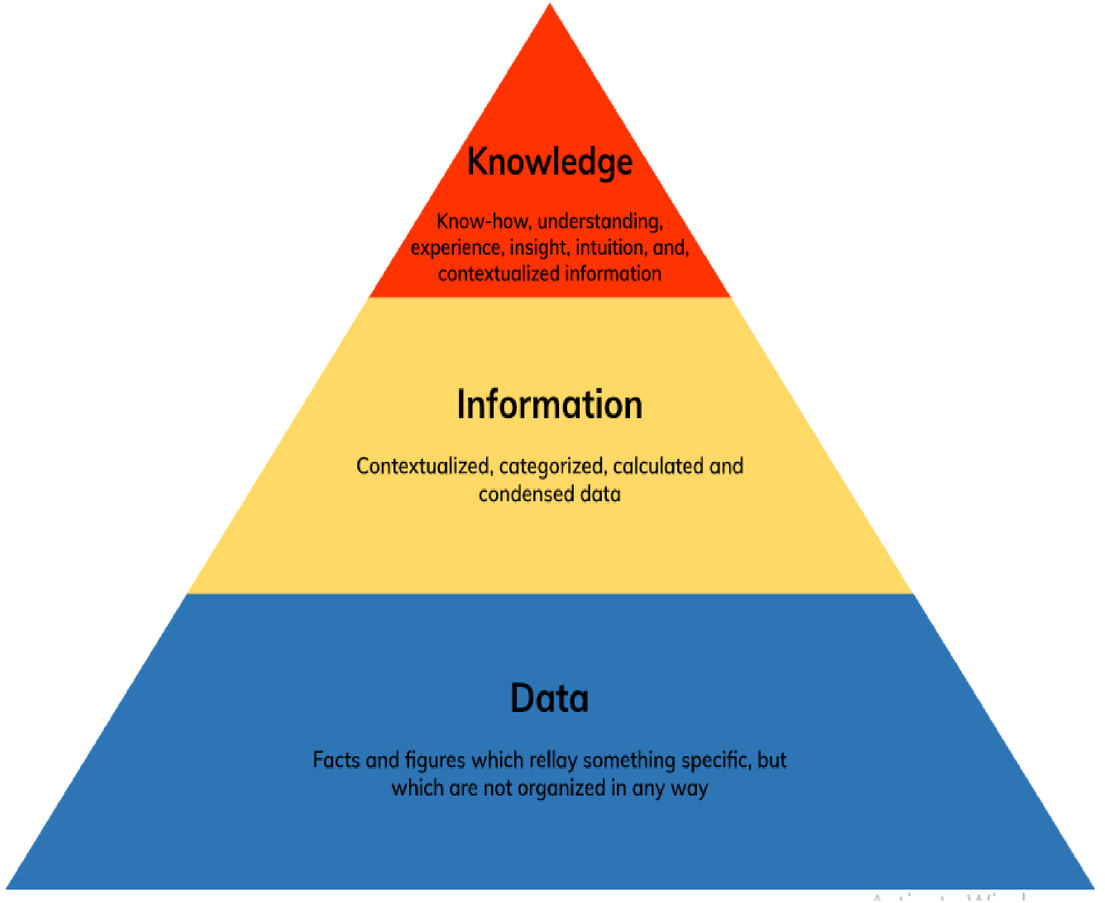


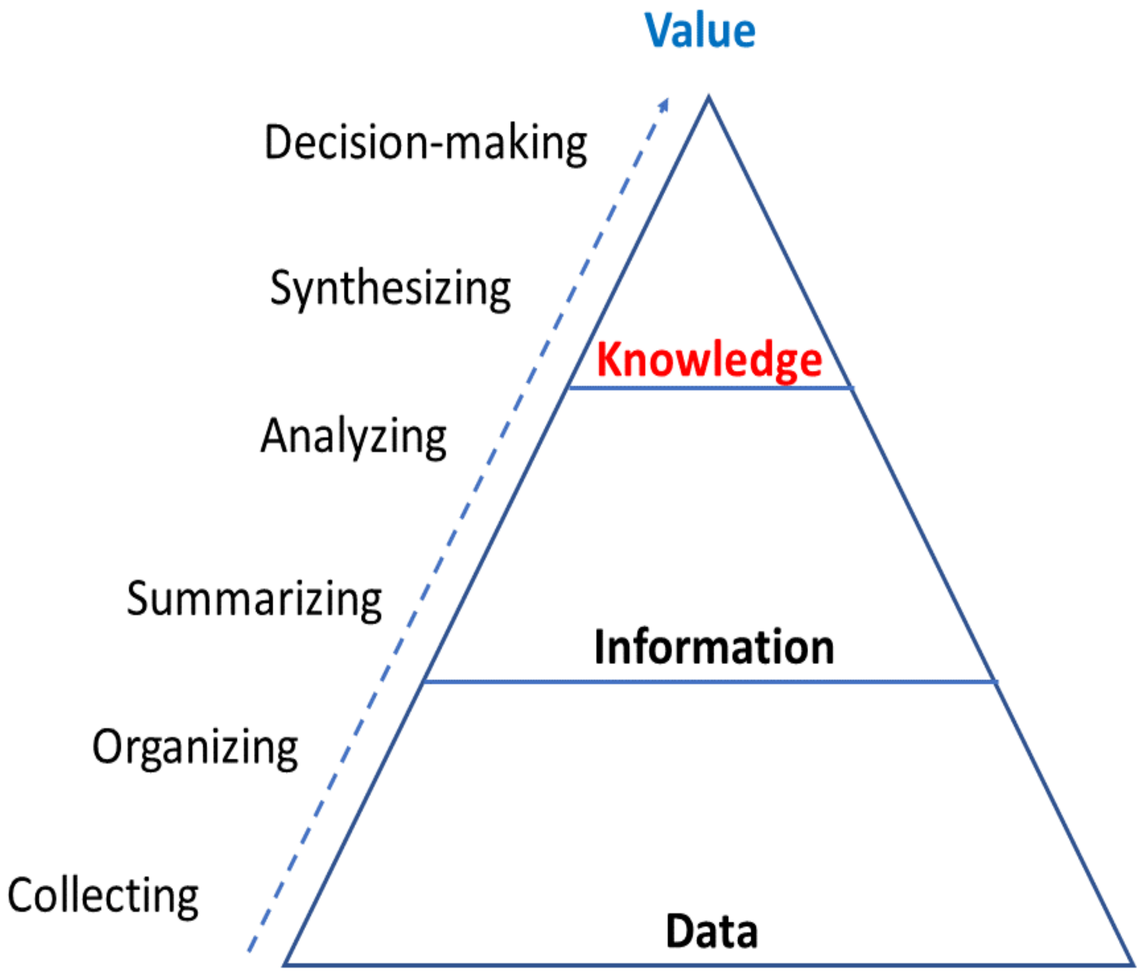
KNOWLEDGE REPRESENTATION

Data vs. Information vs Knowledge



Example

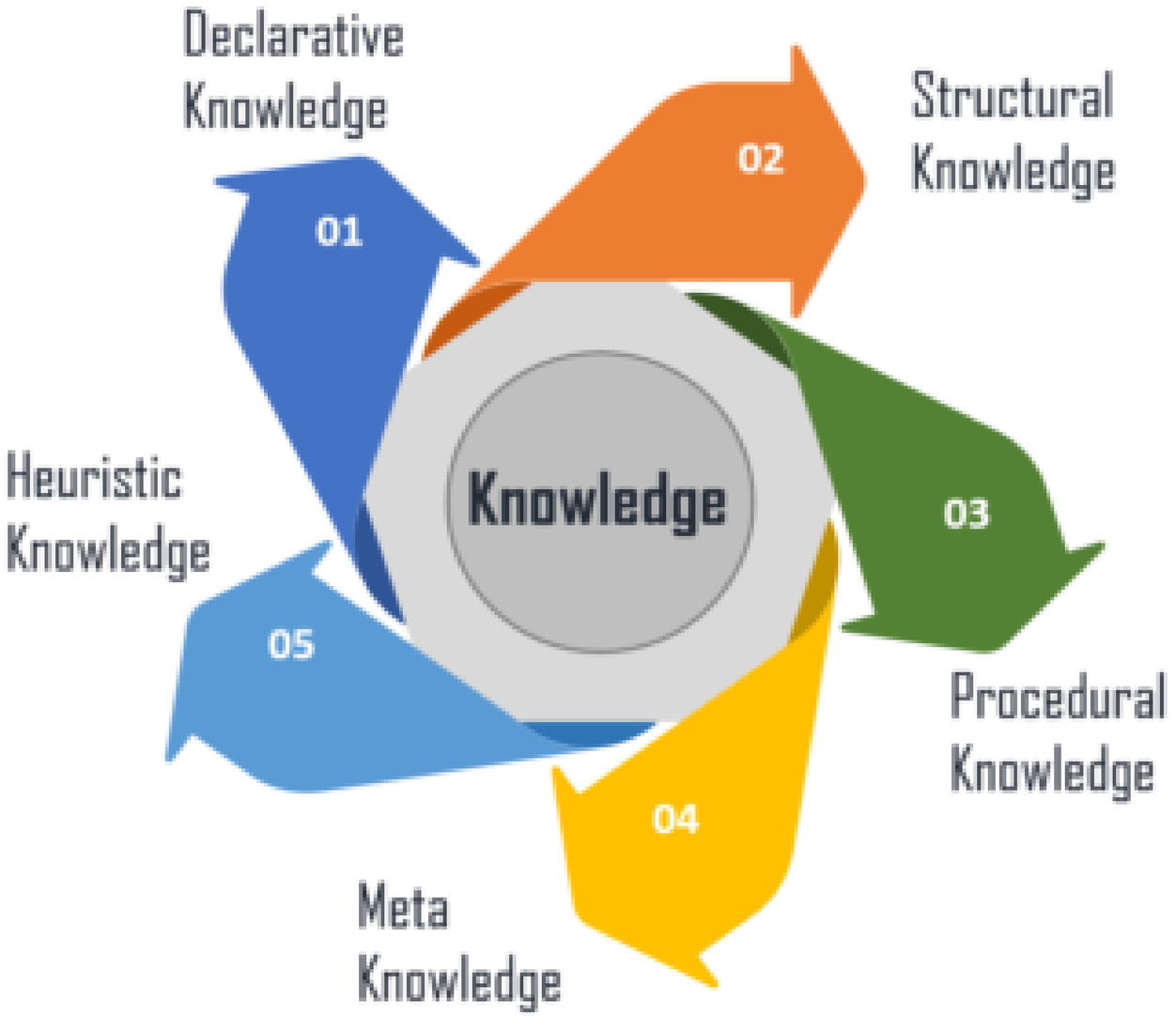
- Look at the examples given for **data**:
  - Dog, cat, cow.
  - 161.2, 175.3, 166.4, 164.7, 169.3
- Only when we assign a context or meaning that's when the data become information. It all becomes meaningful .
  - Dog, cat, cow is a list of household pets
  - 165, 175.2, 186.3, 164.3, 169.3 are the height of 14-year old students.
- If you apply this information to gain further **knowledge**, we could say that:
  - A tiger is not a household pet as it is not on the list, and it lives in the wild forest.
  - The tallest student is 186.3cm.



Knowledge

Knowledge is awareness or familiarity gained by experiences of facts, data, and situations.

Types of knowledge



Declarative Knowledge

- Declarative knowledge is to know about something.
- It includes concepts, facts, and objects.
- It is also called descriptive knowledge and expressed in declarativesentences.
- It is simpler than procedural language.

Procedural Knowledge

- It is also known as imperative knowledge.
- Procedural knowledge is a type of knowledge which is responsible for knowing how to do something.
- It can be directly applied to any task.
- It includes rules, strategies, procedures, agendas, etc.
- Procedural knowledge depends on the task on which it can be applied.

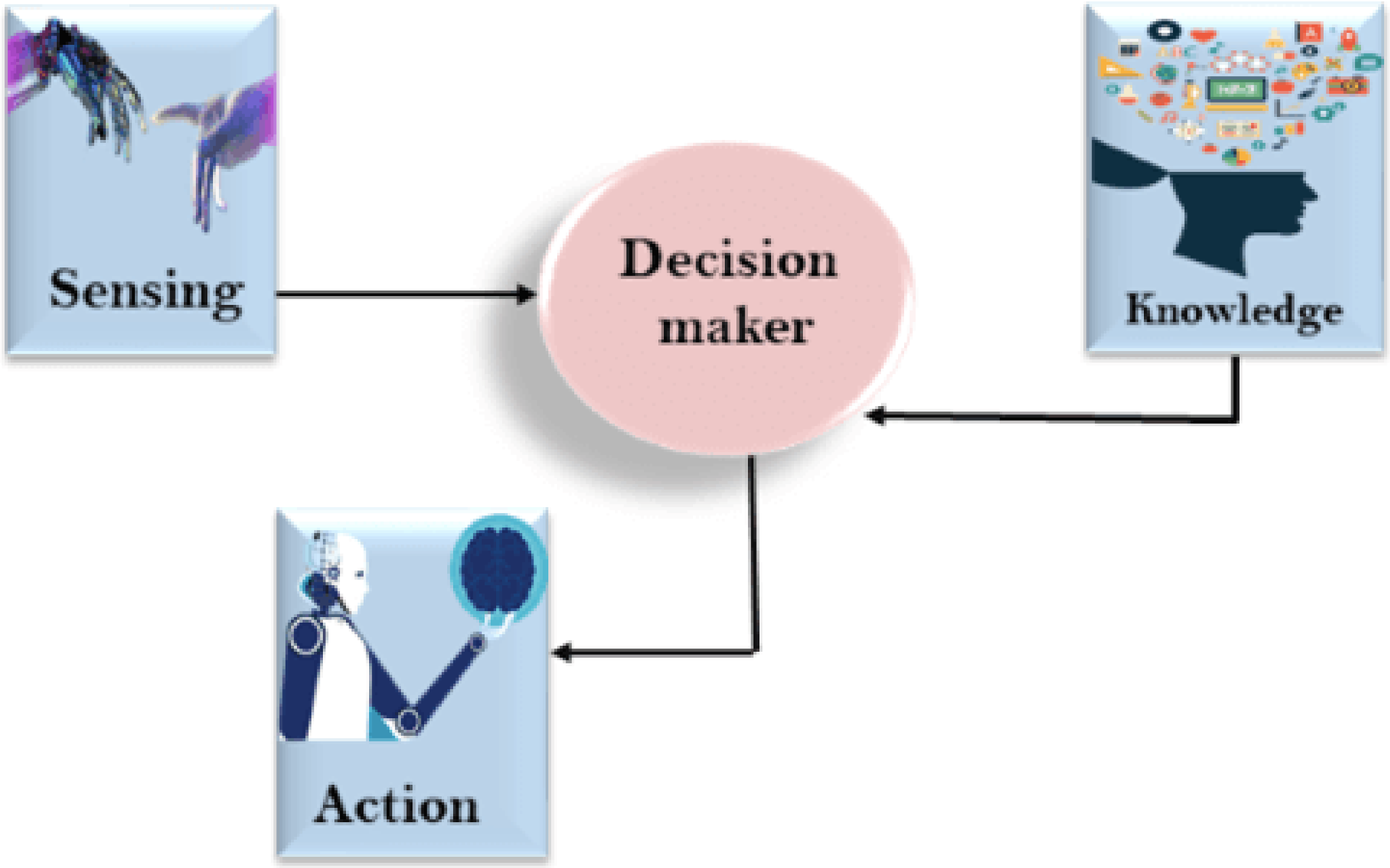
Meta and Heuristic Knowledge

- Knowledge about the other types of knowledge is called **Meta-knowledge**.
- Heuristic knowledge is representing knowledge of some experts in a filed or subject.
- Heuristic knowledge is rules of thumb based on previous experiences, awareness of approaches, and which are good to work but not guaranteed.

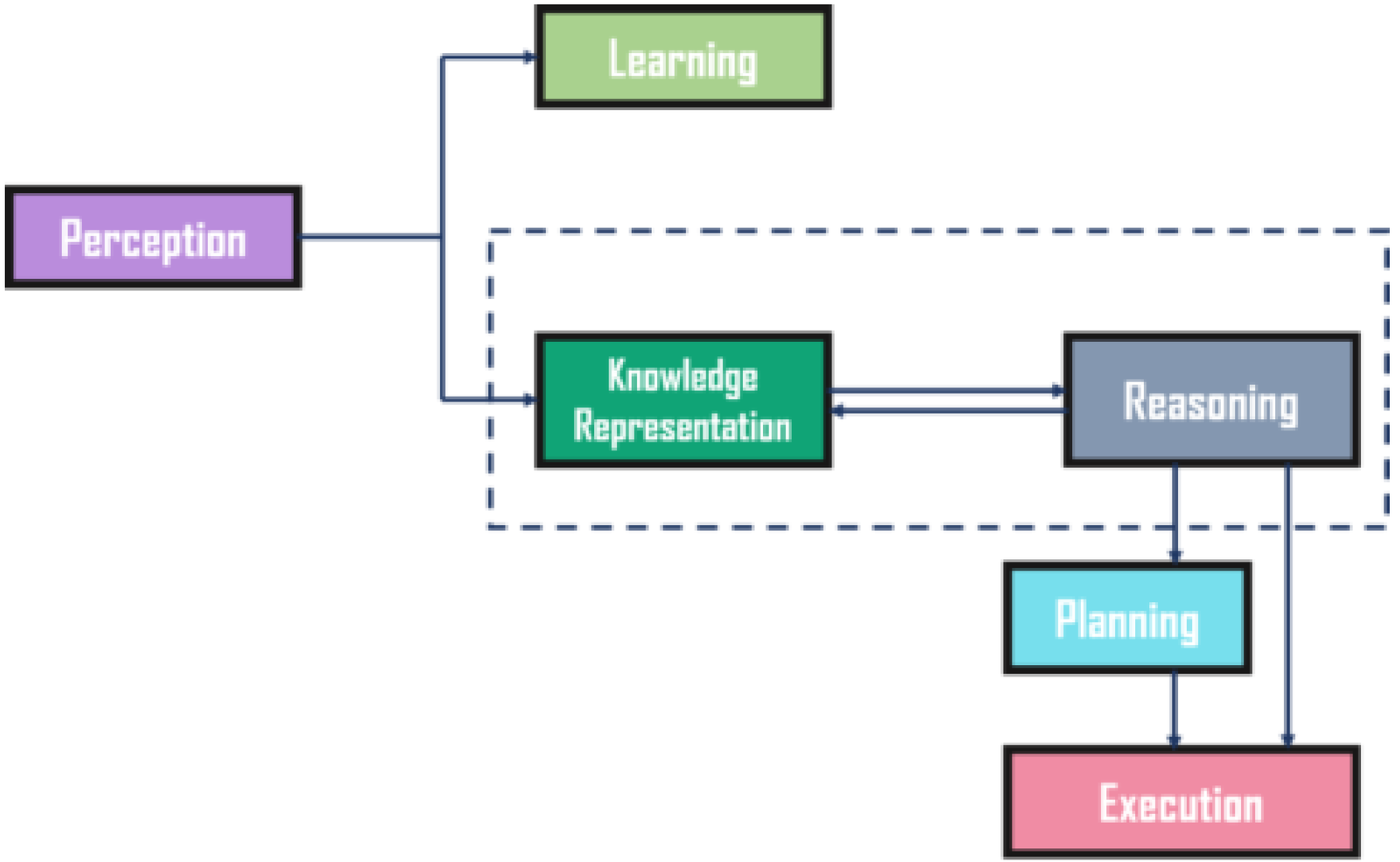
Structural knowledge

- Structural knowledge is basic knowledge to problem-solving.
- It describes relationships between various concepts such as kind of, part of, and grouping of something.
- It describes the relationship that exists between concepts or objects.

Relation between knowledge and AI



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



- 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.

Knowledge Representation

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.

Techniques of KR

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:**

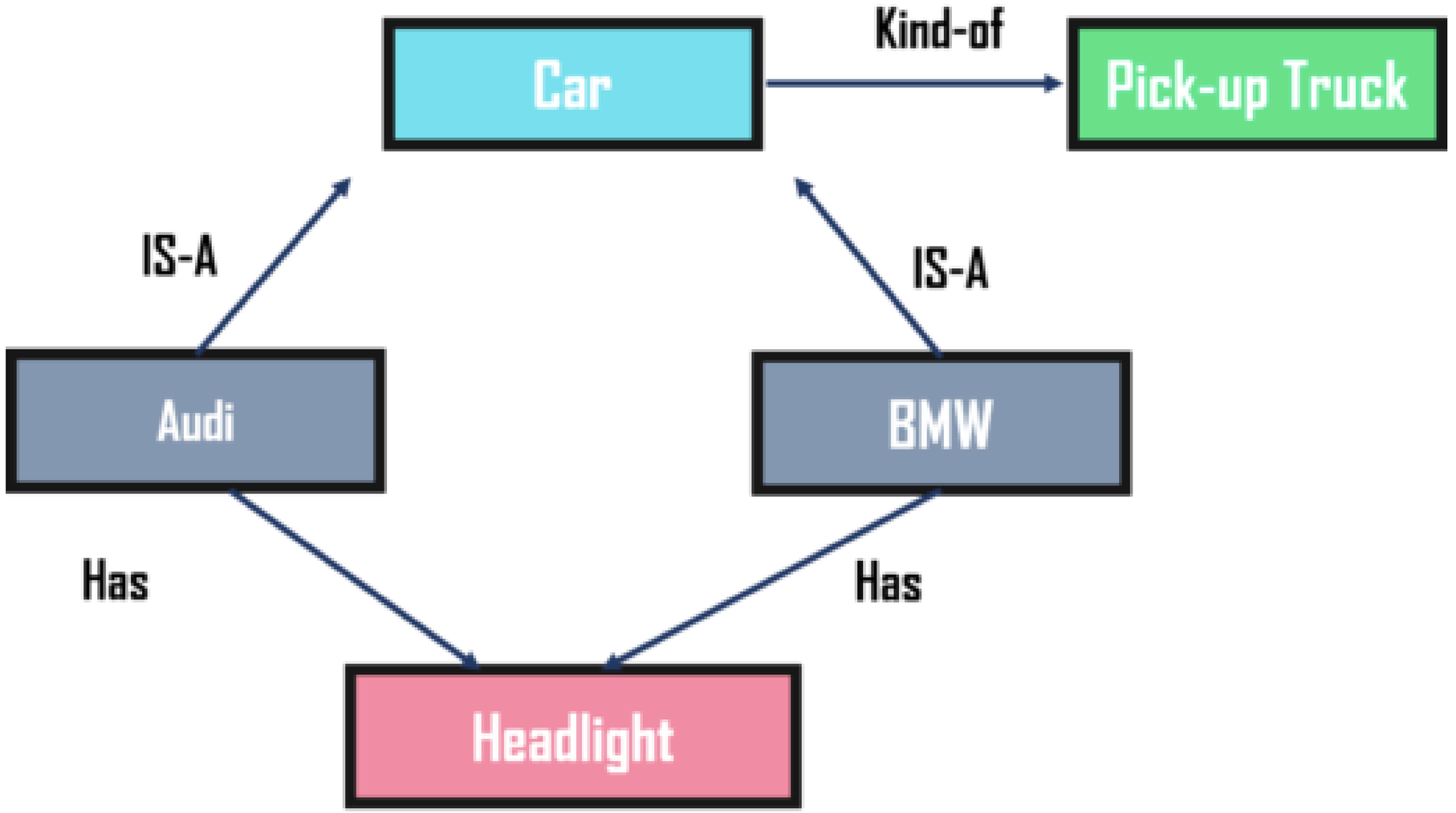
- Logical representation helps to perform logical reasoning.
- This representation is the basis for the programming languages.

✳ **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 consist of two types of relations:
    1. IS-A relation (Inheritance)
    2. Kind-of-relation


Car
Kind-of
Pick-up Truck
IS-A
IS-A
Audi
BMW
Has
Has
Headlight

✳ 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 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**.

✳ **Advantages:**

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

✳ **Disadvantages:**

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

Production Rule

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

1. The set of production rules
2. Working Memory
3. 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.

Representation Requirements

- A good knowledge representation system must have properties such as:

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

Approaches to Knowledge Representation

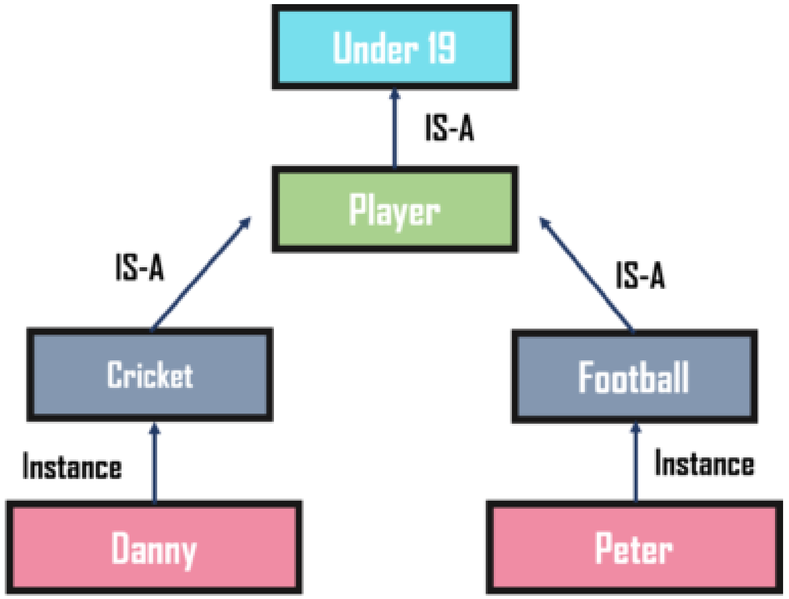
Simple Relational Knowledge

- 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.

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

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.



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

Example: Let's suppose there are two statements:

a. Marcus is a man

b. All men are mortal
Then it can represent as;

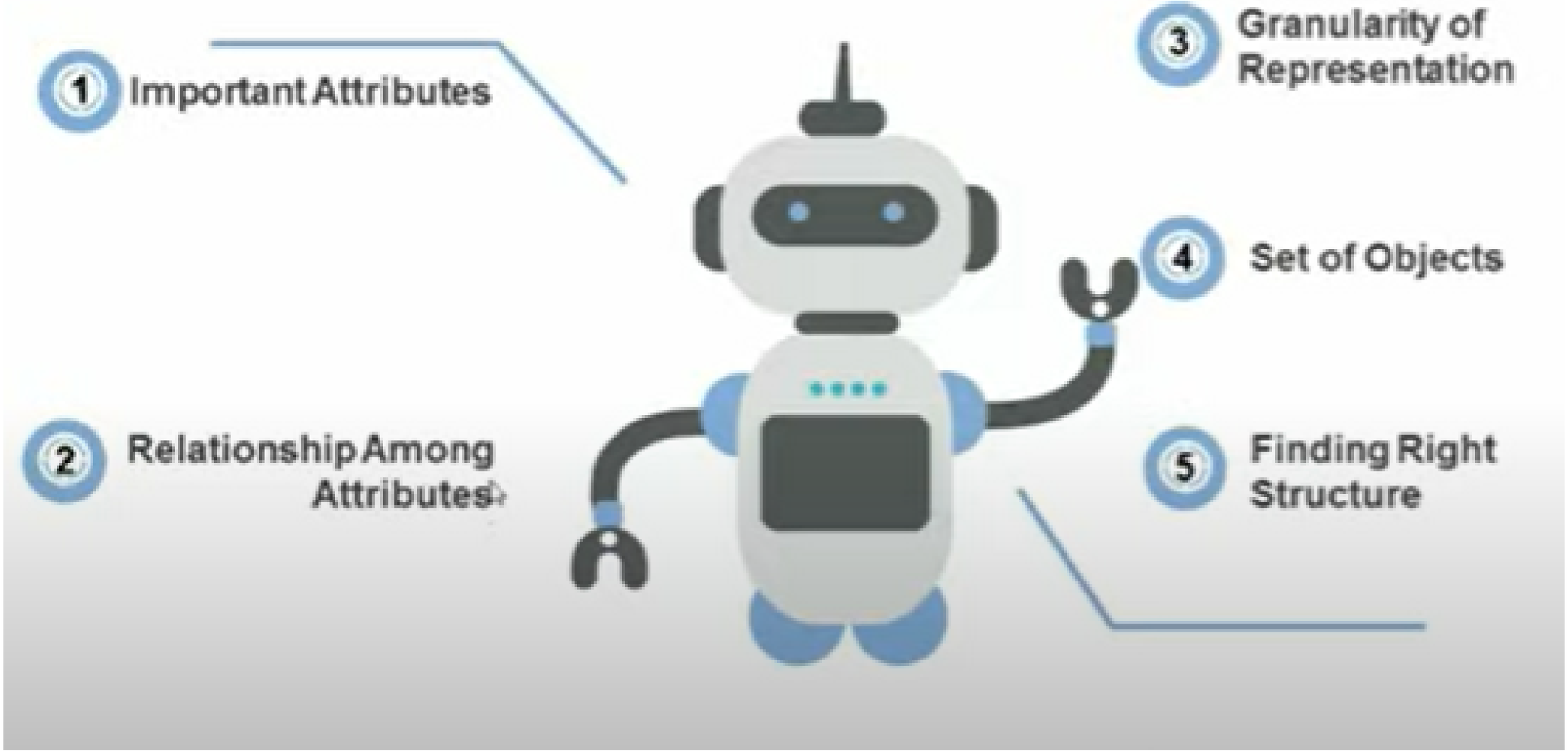
man(Marcus)

∀x = man (x) ----------> mortal (x)s

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 KR


1 Important Attributes
2 Relationship Among Attributes
3 Granularity of Representation
4 Set of Objects
5 Finding Right Structure

1. **Important attributes** : There are two attributes shown in the diagram, **instance** and **isa.** Since these attributes support property of inheritance, they are of prime importance.
2. **Relationships among attributes** : Basically, the attributes used to describe objects are nothing but the entities. However, the attributes of an object do not depend on the encoded specific knowledge.

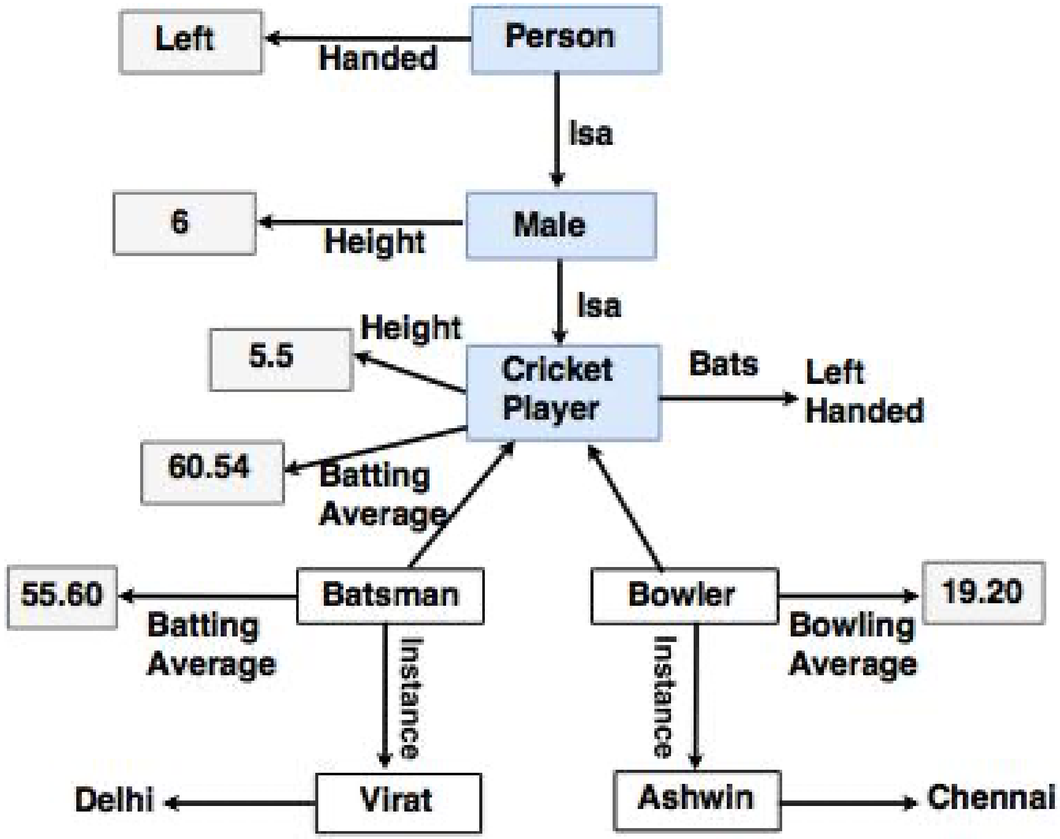


Fig: Inheratable Knowledge Representation

3. Choosing the granularity of representation

- While deciding the granularity of representation, it is necessary to know the following:

1. What are the primitives and at what level should the knowledge be represented?
2. What should be the number (small or large) of low-level primitives or high-level facts?

- High-level facts may be insufficient to draw the conclusion while Low-level primitives may require a lot of storage.

For example: Suppose that we are interested in following facts:

John spotted Alex.

Now, this could be represented as "Spotted (agent(John), object (Alex))”

Such a representation can make it easy to answer questions such as: Who spotted Alex?

Suppose we want to know : "Did John see Sue?"

Given only one fact, user cannot discover that answer.

Hence, the user can add other facts, such as "Spotted (x, y) → saw (x, y)"

4. Representing sets of Objects

- There are some properties of objects which satisfy the condition of a set together but not as individual;

Example: Consider the assertion made in the sentences:

"There are more sheep than people in Australia", and "English speakers can be found all over the world."

These facts can be described by including an assertion to the sets representing people, sheep, and English.

5. Finding the right structure as needed

- To describe a particular situation, it is always important to find the access of right structure. This can be done by selecting an initial structure and then revising the choice.
- While selecting and reversing the right structure, it is necessary to solve following problem statements. **They include the process on how to:**

1. Select an initial appropriate structure.
2. Fill the necessary details from the current situations.
3. Determine a better structure if the initially selected structure is not appropriate to fulfill other conditions.
4. Find the solution if none of the available structures is appropriate.
5. Create and remember a new structure for the given condition.
6. There is no specific way to solve these problems, but some of the effective knowledge representation techniques have the potential to solve them.

Propositional Logic

Propositional Logic

- 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. It is Sunday.
    2. The Sun rises from West (False proposition)
    3. 3+3= 7(False proposition)
    4. 5 is a prime number.

Basic facts about PL

- 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(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**.
- 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.

Types of PL

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 :
    1. 2+2 is 4, it is an atomic proposition as it is a **true** fact.
    2. "The Sun is cold" is also a proposition as it is a **false** fact.

Compound Propositions

- Compound propositions are constructed by combining simpler or atomic propositions, using parenthesis and logical connectives.
- Example :
    1. "It is raining today, and street is wet."
    2. "Ankit is a doctor, and his clinic is in Mumbai."

Logical 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

You should eat or (∨) watch TV at a time.

Please like my video and (∧) subscribe it.

Truth Table

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**

Precedence Order

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

Properties of operators

- **Commutativity:**

  $P \wedge Q = Q \wedge P$, or

  $P \vee Q = Q \vee P$.

- **Associativity:**

  $(P \wedge Q) \wedge R = P \wedge (Q \wedge R)$,

  $(P \vee Q) \vee R = P \vee (Q \vee R)$

- **Identity element:**

  $P \wedge \text{True} = P$,

  $P \vee \text{True} = \text{True}$.

- **Distributive:**

  $P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$.

  $P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$.

- **DE Morgan's Law:**

  $\neg (P \wedge Q) = (\neg P) \vee (\neg Q)$

  $\neg (P \vee Q) = (\neg P) \wedge (\neg Q)$.

- **Double-negation elimination:**

  $\neg (\neg P) = P$.

Limitations

- We cannot represent relations like ALL, some, or none with propositional logic. Example:
    1. All the girls are intelligent.
    2. 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

- 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 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:

  1. Syntax
  2. Semantics

Syntax of FOL

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

1. if 'S' is a sentence then ¬S is also a sentence.
2. if S1 & S2 are sentence] S1∧S2 (conjection) ~ S1 ∨ S2 (∨)

- 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:

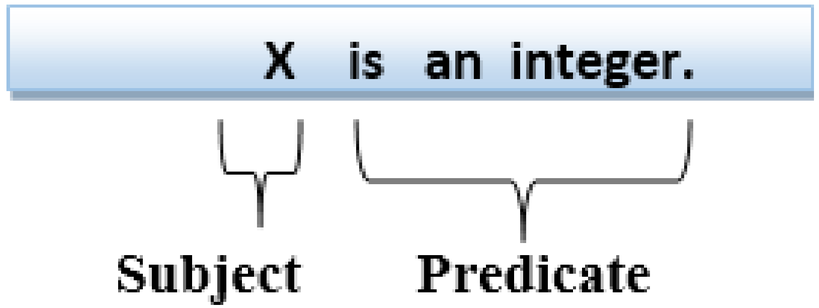
  Ravi and Ajay are brothers: => Brothers(Ravi, Ajay).

  Chinky is a cat: => cat (Chinky).

3. S1 → S2 is also sentence
4. S1 = S2 ..
5. if 'S' is a sentence & 'x' is a variable

Complex Sentences

- Complex sentences are made by combining atomic sentences using connectives.
- First-order logic statements can be divided into two parts:
  1. **Subject:** Subject is the main part of the statement.
  2. **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 ∀
- If x is a variable, then ∀x is read as:

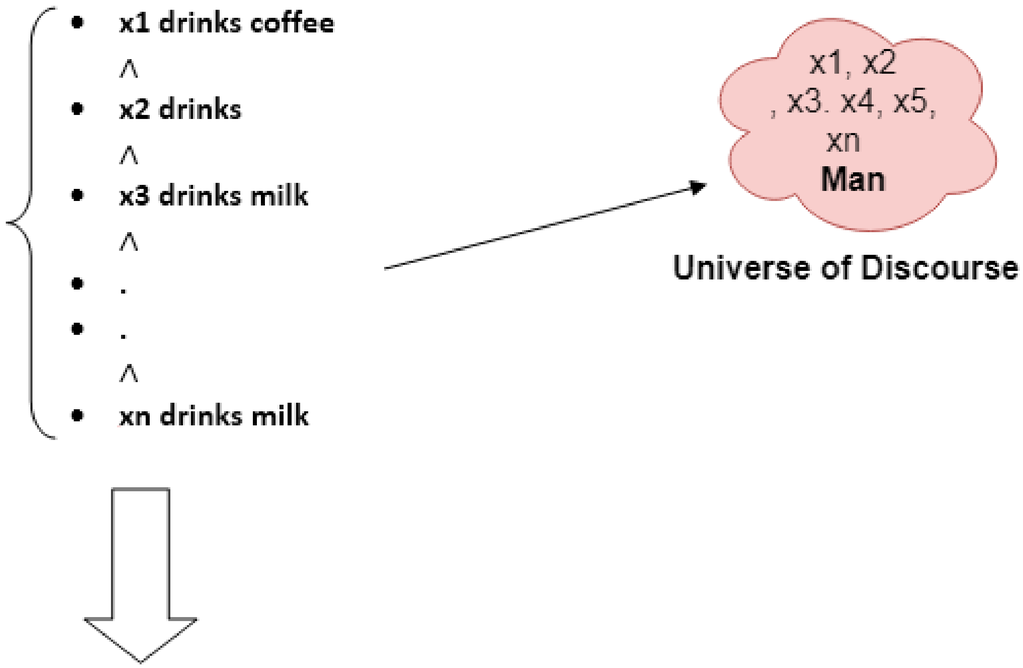
  For all x

  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 Quantifiers

- 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 ∃. When it is used with a predicate variable then it is called as an existential quantifier.
- If x is a variable, then existential quantifier will be ∃x or ∃(x). And it will be read as:

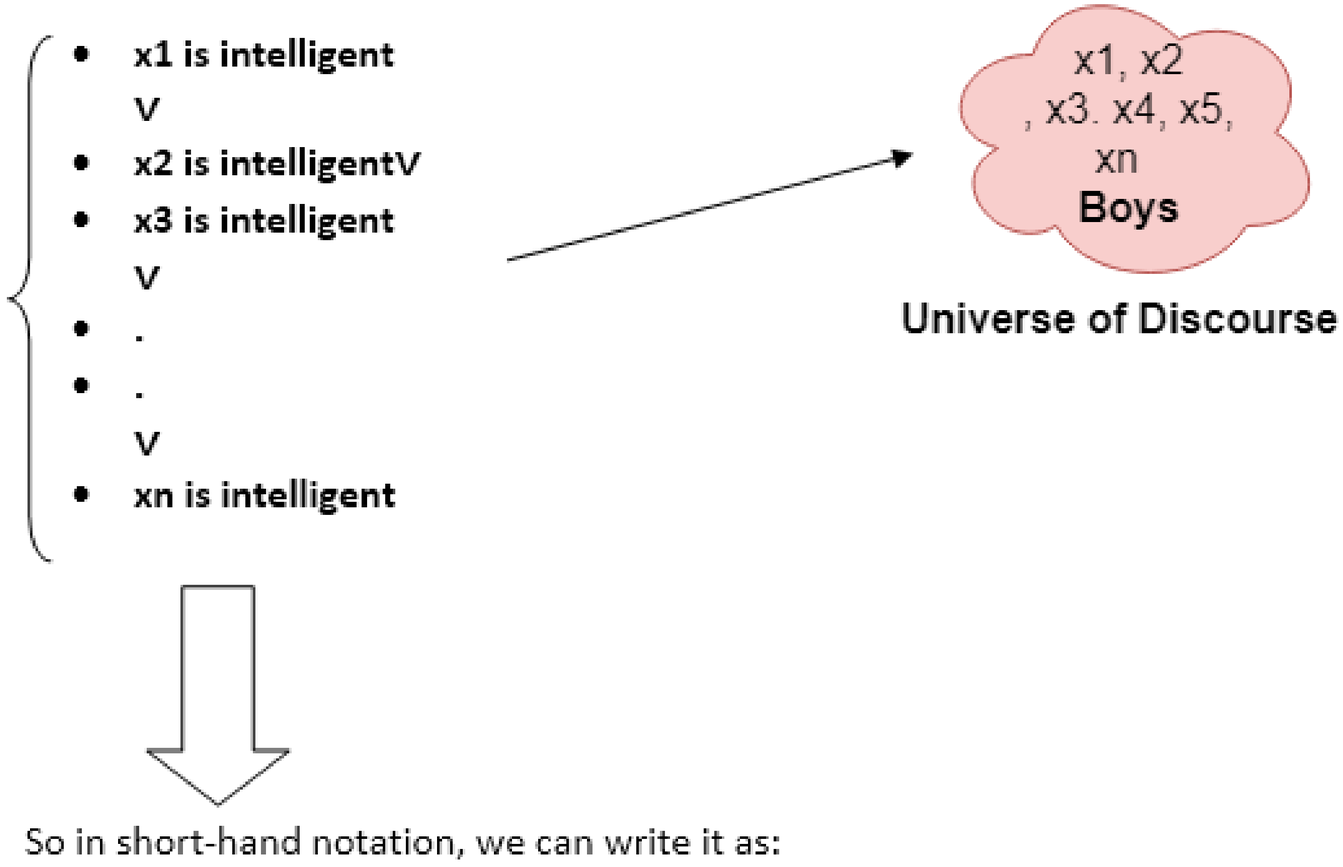
  There exists a ‘x.'

  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.

All Boys like football $\leftarrow \forall x : Boys(x) \rightarrow Like(x, football)$

Some Boys like football $\exists x : Boy(x) \wedge Like(x, football)$

Points to remember

1. The main connective for universal quantifier $\forall$ is implication $\rightarrow$.
2. The main connective for existential quantifier $\exists$ is and $\wedge$.

Properties of Quantifiers

1. In universal quantifier, $\forall x \forall y$ is similar to $\forall y \forall x$.
2. In Existential quantifier, $\exists x \exists y$ is similar to $\exists y \exists x$.
3. $\exists x \forall y$ is not similar to $\forall y \exists x$.

Free and Bound Variables

- **Free Variable:** A variable is said to be a free variable in a formula if it occurs outside the scope of the quantifier.

  Example:

  $\forall x \exists(y)[P(x, y, z)]$, 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:

  $\forall x [A(x) B(y)]$, here **x** and **y** are the bound variables.

Examples of FOL

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)].