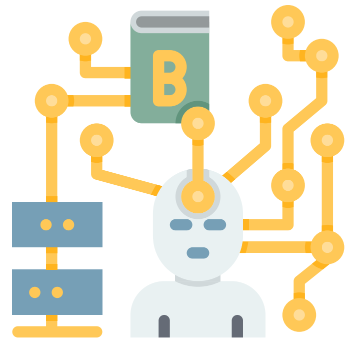
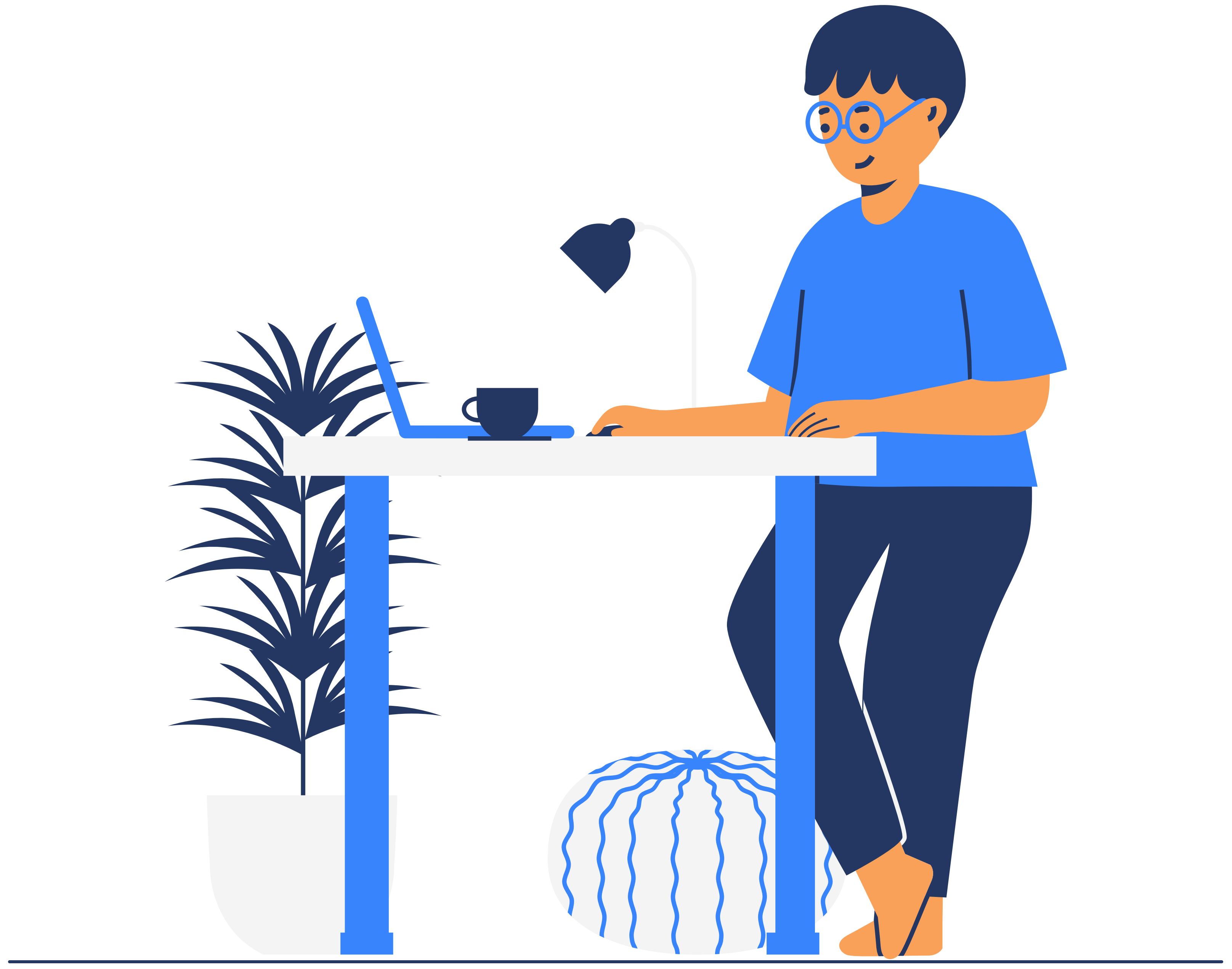
Artificial Intelligence

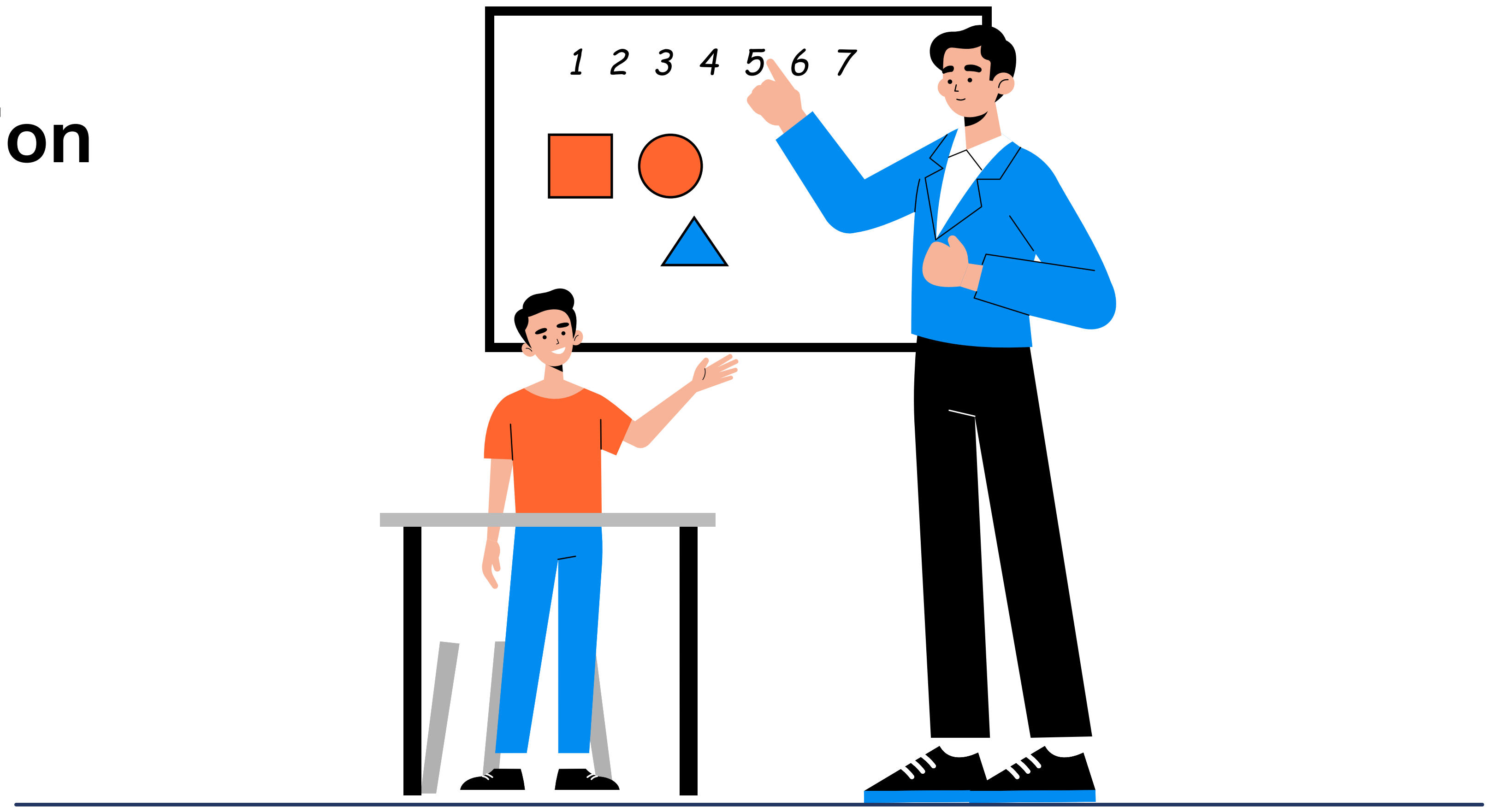
Unit 3

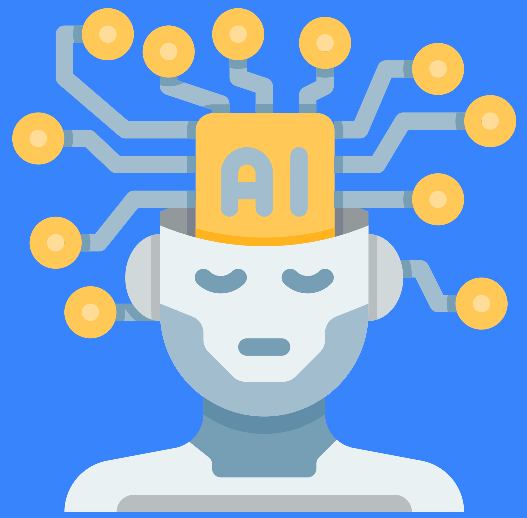




Topics to be covered...

First order logic
Predicate logic and casual form question
Well-formed formula(WFF)
Prolog Programming
Unification
Forward and Backward chaining
Resolution in Predicate logic
Properties of Good knowledge representation
Happy Ending!





Inference rules in first order logic

Inference rules in first order logic

- First-order logic is another way of knowledge representation in artificial intelligence. It is an extension to propositional logic.
- 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.
- FOL assumes the following things in the world:
    - **Objects:** A, B, people, numbers, colors, wars, theories, squares, pits, wumpus
    - **Relations:** The sister of, brother of, has color, comes between
    - **Function:** Father of, best friend, third inning of, end of

Inference rules in first order logic

- Examples:

Predicate logic and casual form question

Predicate logic and casual form question

- John likes all kind of food
- Apples are food
- Bill eats mango and is still alive
- Sue eats everything Bill eats

Well-formed formula(WFF)

Well-formed formula(WFF)

Propositional logic uses a symbolic “language” to represent the logical structure, or form, of a compound proposition.

Fortunately, the syntax of propositional logic is easy to learn. It has only three rules:

- Any capital letter by itself is a WFF.
- Any WFF can be prefixed with “~”. (The result will be a WFF too.)
- Any two WFFs can be put together with “•”, “∨”, “⊃”, or “≡” between them, enclosing the result in parentheses. (This will be a WFF too.)

Well-formed formula(WFF)

WFF	**explanation**
A	by rule 1
~A	by rule 2, since A is a WFF
~~A	by rule 2 again, since ~A is a WFF
(~A • B)	by rule 3, joining ~A and B
((~A • B) ⊃ ~~C)	by rule 3, joining (~A • B) and ~~C
~((~A • B) ⊃ ~~C)	by rule 2, since ((~A • B) ⊃ ~~C) is a WFF

non-WFF	**explanation**
A~	the ~ belongs on the left side of the negated proposition
(A)	parentheses are only introduced when joining two WFFs with •, ∨, ⊃, or ≡
(A •)	there’s no WFF on the right side of the •
(A • B) ⊃ C)	missing paranthesis on the left side
(A • ⊃ B)	cannot be formed by the rules of syntax

Prolog Programming

Prolog Programming

- Prolog stands for programming in logic.
- Prolog is a declarative language, which means that a program consists of data based on the facts and rules (Logical relationship) rather than computing how to find a solution.
- Prolog is a declarative language that means we can specify what problem we want to solve rather than how to solve it.
- Prolog is used in some areas like database, natural language processing, artificial intelligence, but it is pretty useless in some areas like a numerical algorithm or instance graphics.

- **Applications of Prolog**
  - Robot Planning
  - Natural language understanding
  - Machine Learning
  - Problem Solving
  - Intelligent Database retrieval

Prolog Programming

- ?- write('Welcome to EIOV'),nl,write('Example of Prolog'),nl.

- The prompt is used to show that the Prolog system is ready to specify one or more goals of sequence to the user.
- Using a full stop, we can terminate the sequence of goals.

- Welcome to EIOV
- Example of Prolog

- prolog1.pl.

• Boolean	true, fail
• Variables	variables
• Integer	integers
• Atom	character sequence
• Real	floating point number

Unification

Unification

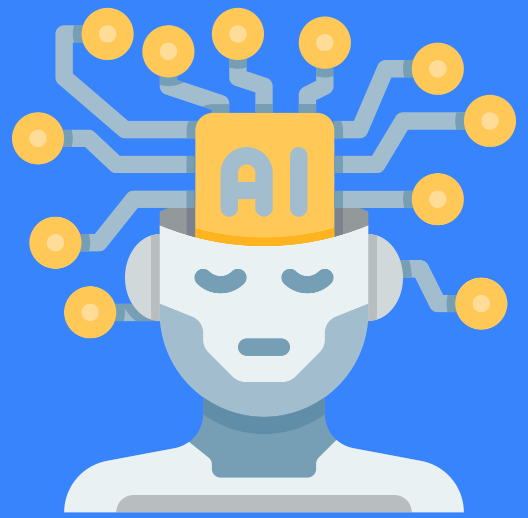
- Unification is the process of finding a substitute that makes two separate logical atomic expressions identical. The substitution process is necessary for unification.
- It accepts two literals as input and uses substitution to make them identical.
- Let Ψ1 and Ψ2 be two atomic sentences, and be a unifier such that Ψ1 = Ψ2 , then UNIFY(Ψ1, Ψ2)can be written.
- Example: Find the MGU for Unify{King(x), King(John)}
  - Let Ψ1 = King(x), Ψ2 = King(John),

- **Conditions for Unification**
  - The following are some fundamental requirements for unification:
  - Atoms or expressions with various predicate symbols can never be united.
  - Both phrases must have the same number of arguments.
  - If two comparable variables appear in the same expression, unification will fail.

Unification Algorithm

- Step 1: Begin by making the substitute set empty.
- Step 2: Unify atomic sentences in a recursive manner:
  - a.Check for expressions that are identical.
  - b.If one expression is a variable vΨi, and the other is a term ti which does not contain variable vi, then:
    - a. Substitute ti / vi in the existing substitutions
    - b.Add ti / vi to the substitution setlist.
    - c. If both the expressions are functions, then function name must be similar, and the number of arguments must be the same in both the expression.
- Find the most general unifier for each pair of the following atomic statements (If exist).

- **UNIFY(knows(Richard, x), knows(Richard, John))**
  - Here, Ψ1 = knows(Richard, x), and Ψ2 = knows(Richard, John)
  - S0 => { knows(Richard, x); knows(Richard, John)}
  - S SUBST θ= {John/x}
  - S1 => { knows(Richard, John); knows(Richard, John)}, Successfully Unified.
  - Unifier: {John/x}.



Forward and Backward chaining

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.

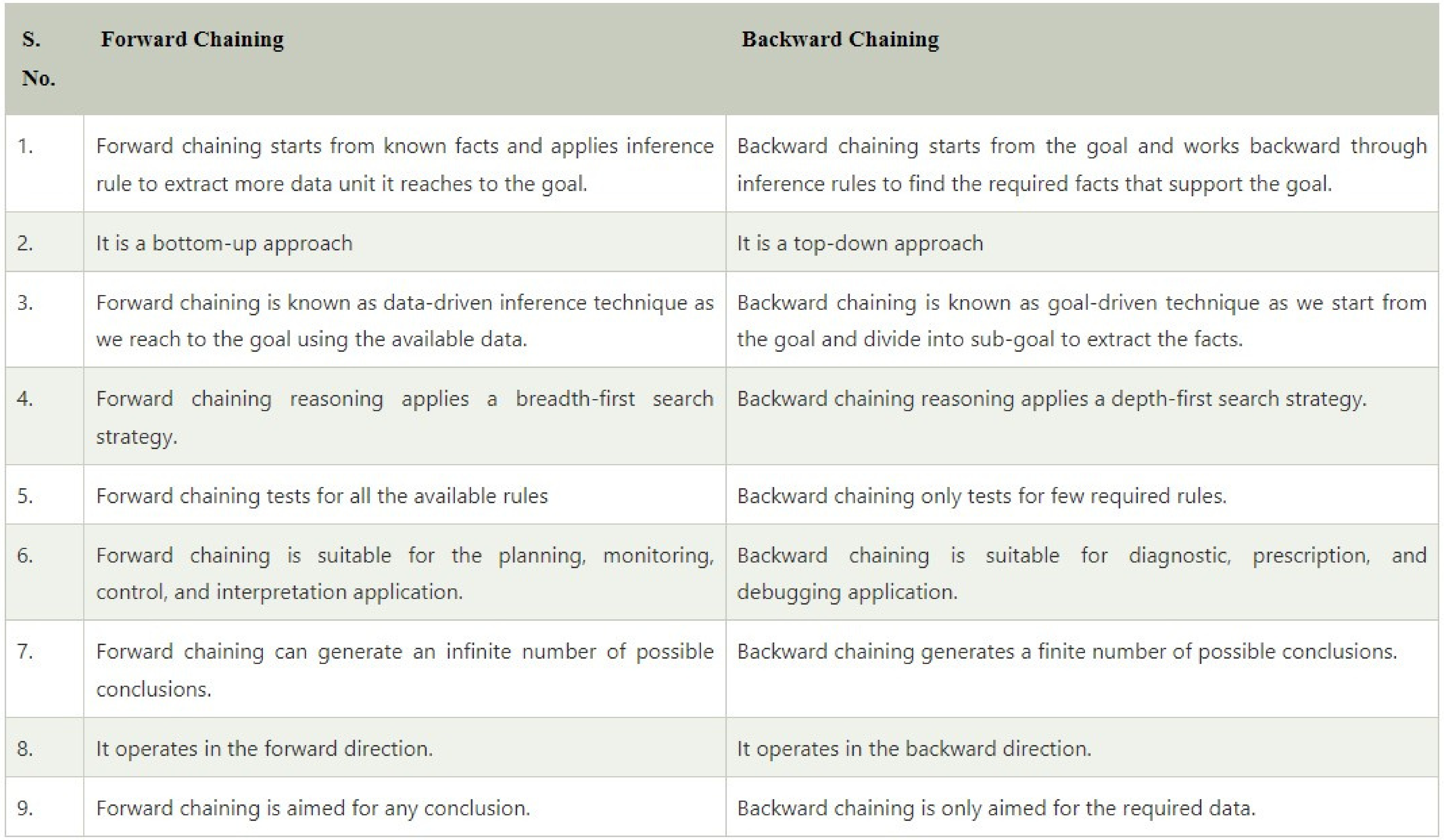
Backward chaining

- 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.

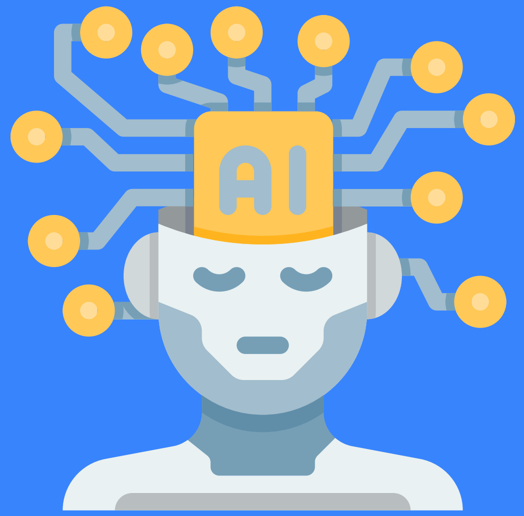
Properties of backward chaining:

- 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.

Forward vs Backward chaining



S. No.	Forward Chaining	Backward Chaining
1.	Forward chaining starts from known facts and applies inference rule to extract more data unit it reaches to the goal.	Backward chaining starts from the goal and works backward through inference rules to find the 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.
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 in Predicate logic

Resolution in Predicate logic

- 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.

Resolution in Predicate logic

- **Steps for Resolution:**
  - Conversion of facts into first-order logic.
  - Convert FOL statements into CNF
  - Negate the statement which needs to prove (proof by contradiction)
  - Draw resolution graph (unification).

Properties of Good knowledge representation

Properties of Good knowledge representation

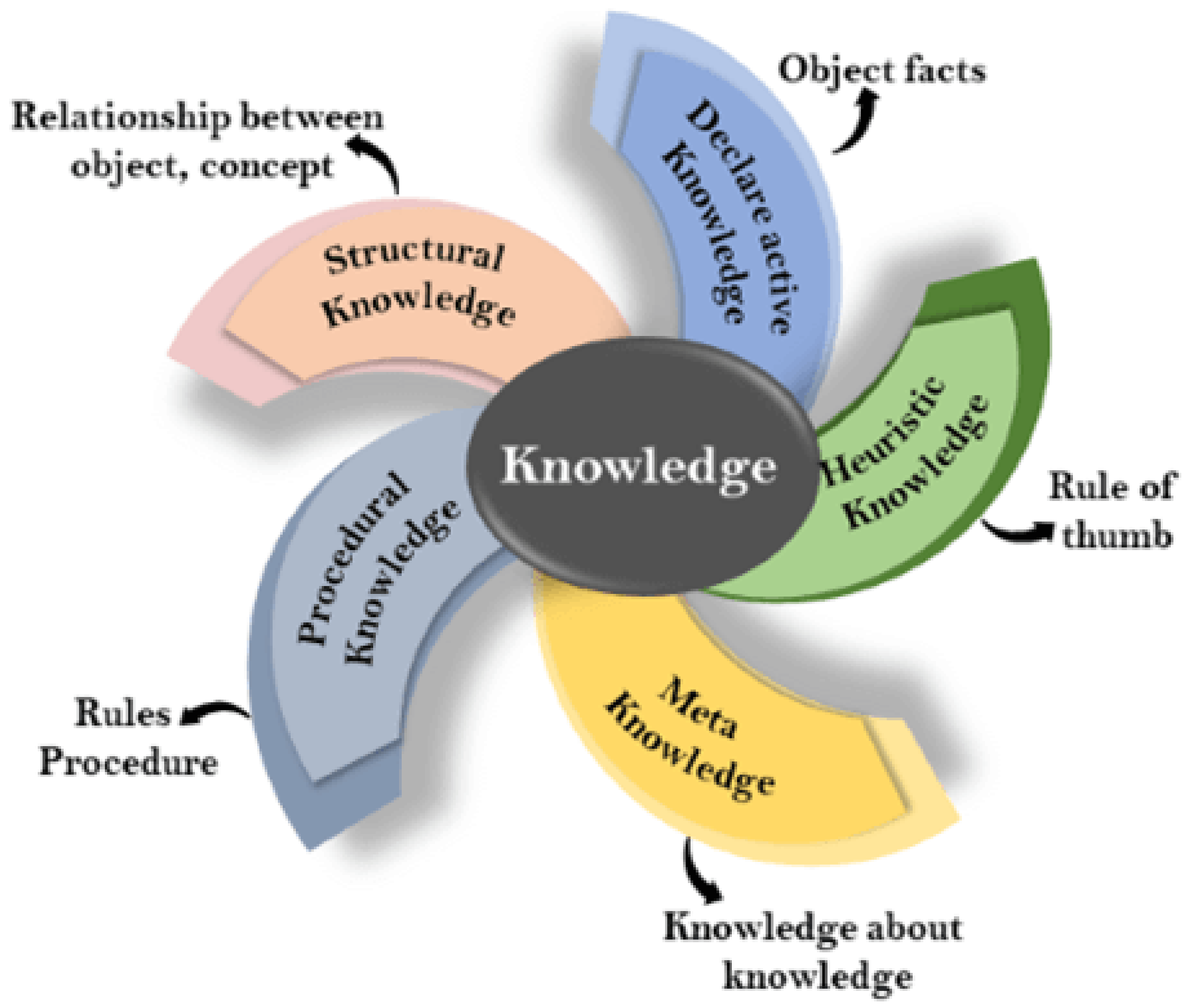
- Humans are best at understanding, reasoning, and interpreting knowledge. Human knows things, which is knowledge and as per their knowledge they perform various actions in the real world. But how machines do all these things comes under knowledge representation and reasoning. Hence we can describe Knowledge representation as following:
- Knowledge representation and reasoning (KR, KRR) is the part of Artificial intelligence which concerned with AI agents thinking and how thinking contributes to intelligent behavior of agents.
- It is responsible for representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real world problems such as diagnosis a medical condition or communicating with humans in natural language.
- It is also a way which describes how we can represent knowledge in artificial intelligence. Knowledge representation is not just storing data into some database, but it also enables an intelligent machine to learn from that knowledge and experiences so that it can behave intelligently like a human.

Properties of Good knowledge representation

What to Represent:

- Following are the kind of knowledge which needs to be represented in AI systems:
- **Object**: All the facts about objects in our world domain. E.g., Guitars contains strings, trumpets are brass instruments.
- **Events**: Events are the actions which occur in our world.
- **Performance**: It describe behavior which involves knowledge about how to do things.
- **Meta-knowledge:** It is knowledge about what we know.
- **Facts**: Facts are the truths about the real world and what we represent.
- **Knowledge-Base:** The central component of the knowledge-based agents is the knowledge base. It is represented as KB. The Knowledgebase is a group of the Sentences (Here, sentences are used as a technical term and not identical with the English language).

Types of knowledge



Happy Ending!