Knowledge Representation
UNIT-3

KNOWLEDGE REPRESENTATION

- First Order Predicate Logic –
- Prolog Programming –
- Unification –
- Forward Chaining-Backward Chaining –
- Resolution –
- Knowledge Representation –
- Ontological Engineering-
- Categories and Objects –
- Events – Mental Events and Mental Objects –
- Reasoning Systems for Categories –
- Reasoning with Default Information

What is knowledge representation?

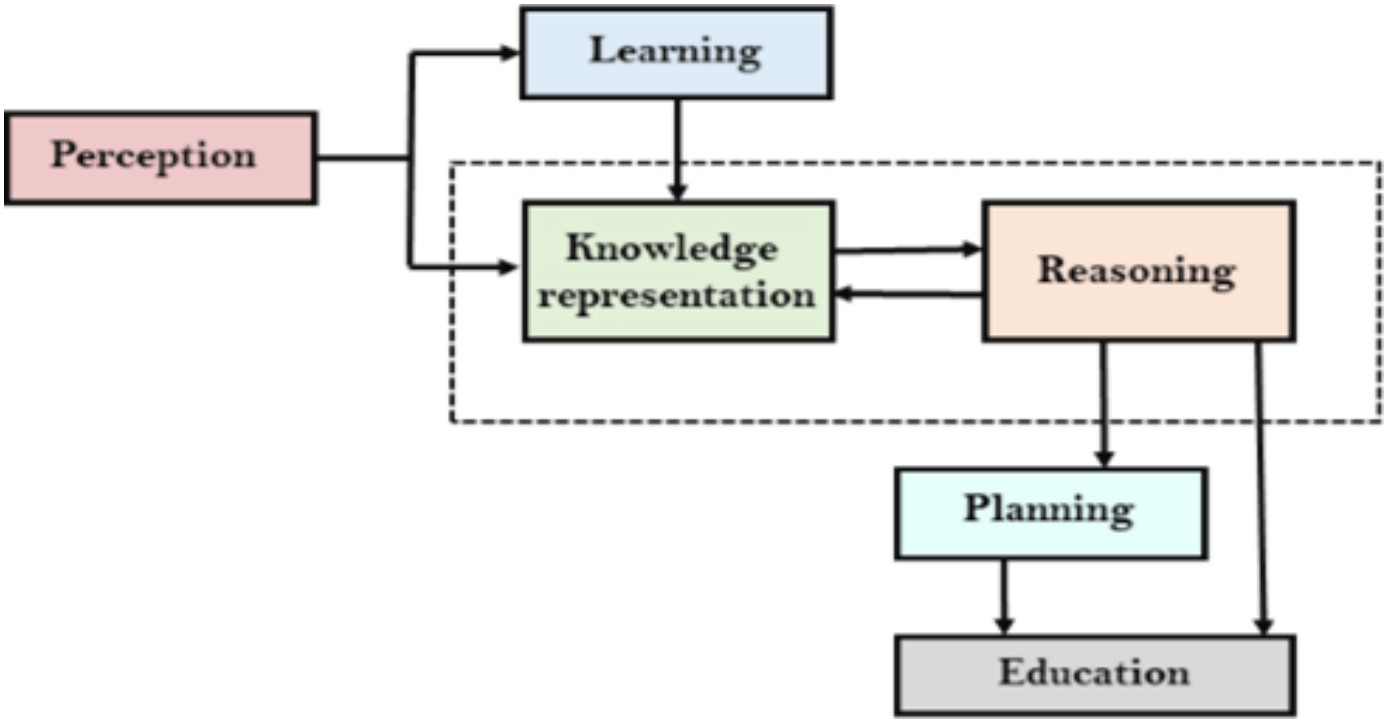
- Knowledge is cognizance or understanding expanded by experiences of facts, data, and circumstances.
- Humans are best at understanding, reasoning, and interpreting knowledge. **But how machines will be able to do all these comes under knowledge representation and reasoning**.

Following are the types of knowledge in artificial intelligence:

- **Declarative Knowledge**
- **Procedural Knowledge**
- **Meta-knowledge**
- **Heuristic knowledge**
- **Structural knowledge**


Knowledge
Declare active Knowledge
Object facts
Structural Knowledge
Relationship between object, concept
Heuristic Knowledge
Rule of thumb
Procedural Knowledge
Rules Procedure
Meta Knowledge
Knowledge about knowledge

AI Knowledge Cycle



Approaches to Knowledge Representation

1.Simple Relational Knowledge:

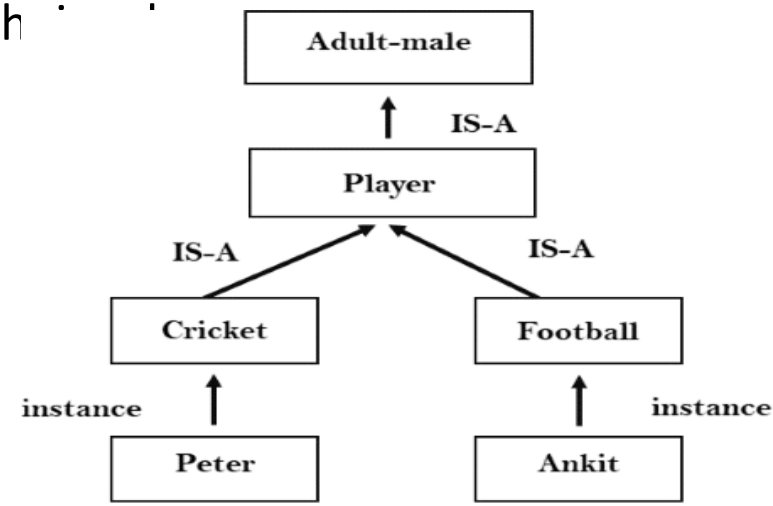
- It is the simplest way of storing facts which uses the relational method, and each fact about a set of the object is set out systematically in columns.
- This approach of knowledge representation is famous in database systems where the relationship between different entities is represented.
- This approach has little opportunity for inference.
- **Example:**

Player	Weight	Age
Player1	65	23
Player2	58	18
Player3	75	24

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 th

Example:



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:
  - Marcus is a man
  - All men are mortal
    Then it can represent as;

    man(Marcus)
    ∀x :man (x)→mortal (x)

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-specificknowledge using this approach.
- But it is not necessary that we can represent all cases in this approach.

Requirements for knowledge Representation system

- **Representational Accuracy:** Ability to represent all kind of required knowledge.
- **InferentialAdequacy:** Ability to manipulatethe representational structures to produce new knowledge corresponding to existing structure.
- **Inferential Efficiency:** Ability to direct the inferential knowledge appliance into the most productive orders by storing suitable guides.
- **Acquisitional Efficiency-** The ability to obtain the new knowledge easily
  using instinctive methods.

Techniques of Knowledge Representation

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

1. Logical Representation

There are two type of logical representation:

- Propositional Logics
- Predicate logics

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 determines which symbol we can use in knowledge representation.
- Semantics are the rules by which we can interpret the sentence in the logic.

2. Semantic Network Representation

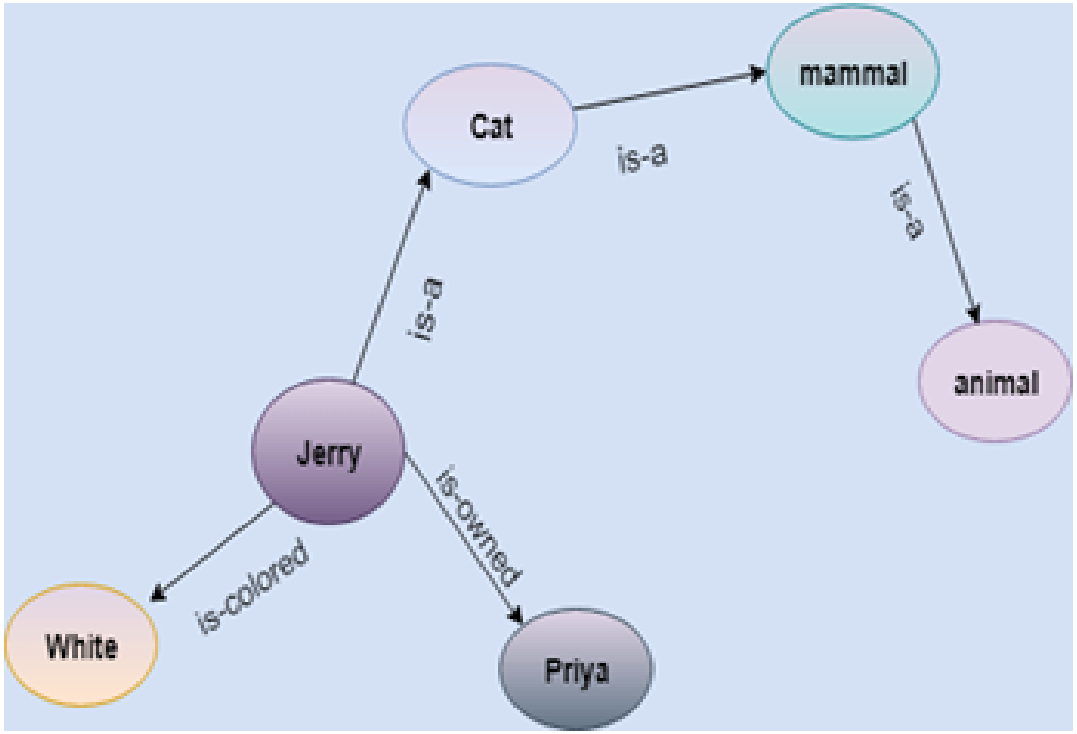
- It is alternative of predicate logic for knowledge representation.
- In Semantic net we represent knowledge in graphical network form.
- This network consists of nodes representing objects and arcs which describe the relationship between those objects. Semantic networks can categorize the object in different forms and can also link those objects.
- This representation consists of mainly two types of relations:

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

Example:

Statements:

- Jerry is a cat.
- Jerry is a mammal
- Jerry is owned by Priya.
- Jerry is brown colored.
- All Mammals are animal.



3. Frame Representation

- The structure of frame is like record which contains 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.
- A frame is also known as **slot-filter knowledge representation** in artificial intelligence.

Example: Frame for a book

Slots	Filters
Title	Artificial Intelligence
Genre	Computer Science
Author	Peter Norvig
Edition	Third Edition
Year	1996
Page	1152

4. Production Rules

It is based on pair of (**condition, action**) that is "If condition then action".

It has mainly three parts:

- Set of production rules
- Working Memory
- The recognize-act-cycle

Condition part - Determines which rule may be applied to a problem

Action part - Carries out the associated problem-solving steps.

This complete process is known as recognize-act cycle.

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).

Propositional Logic

- **Propositional logic** is used for solving complex problems using simple statements.
- These statements can either be true or false but cannot be both at same time. These propositions form knowledge representation, reasoning and decision-making in AI systems
- Propositional logic works with statements called propositions that can be true or false.
- These propositions represent facts or conditions about a situation.
- We use symbols to represent the propositions and logical operations to connect those propositions.
- It help us understand how different facts are related to each other in complex statements or problem.
- Proposition operators like conjunction (∧), disjunction (∨), negation (¬), implication(→) and biconditional (↔) helps combine various proposition to represent logical relations

Example of Propositions Logic

P: "The sky is blue." (This statement can be either true or false.)
Q: "It is raining right now." (This can also be true or false.)
R: "The ground is wet." (This is either true or false.)

These can be combined using logical operations to create more complex statements. For example:

- **P ∧ Q**: "The sky is blue AND it is raining." (This is true only if both P and Q are true.)
- **P ∨ Q**: "The sky is blue OR it is raining." (This is true if at least one of P or Q is true.)
- **¬P**: "It is NOT true that the sky is blue." (This is true if P is false means the sky is not blue.)

Tautologies, Contradictions and Contingencies

- **Tautology**: A proposition that is always true no matter the truth values of the individual components.
  Example: "P ∨ ¬P" (This is always true because either P is true or P is false).
- **Contradiction**: A proposition that is always false.
  Example: "P ∧ ¬P" (This is always false because P can't be both true and false at the same time).
- **Contingency**: A proposition that can be true or false depending on the truth values of its components.
  Example: "P ∧ Q" (This is true only if both P and Q are true)

Rules of Inference:

→ ① **Modus Ponens**: If 'P' and 'P→Q' is given to be true, then we can infer that 'Q' is true.

P: It is a holiday ✓ (T)

Q: The school is closed] → we can infer that it is true.

P→Q: If it is a holiday, then School is closed ✓ (T)

② **Modus Tollens**: If '~Q' and '~P→~Q' are given to be true, then we can infer that ~P is true.

~Q = School is not closed.

if it is not a holiday, then School is not closed.

(~P→~Q) → ~P] it is not a holiday (True)

Limitations of Propositional Logic

1. **Lack of Expressiveness:** It cannot handle relationships like "All humans are mortal.“

2. **Scalability:** Truth tables expands exponentially with the number of propositions.

3. **Limited Inference:** It only handles binary truth values (true/false) and cannot represent probabilities.

4. **No Quantifiers:** Unlike predicate logic it does not allow the use of quantifiers such as “for all” (denoted by ∀) or “there exists” (denoted by ∃).

5. **Inability to Handle Uncertainty:** It cannot represent probabilities or partial truths which helps in making it unsuitable for uncertain situations.

6. **Lack of Context Awareness:** It ignores meaning or context of statements which limits its ability to interpret nuanced scenarios.

First Order Predicate 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

User provides

- **Constant symbols,** which represent individuals in the world
  - Mary
  - 3
  - Green
- **Function symbols,** which map individuals to individuals
  - father-of(Mary) = John
  - color-of(Sky) = Blue
- **Predicate symbols,** which map individuals to truth values
  - greater(5,3)
  - green(Grass)
  - color(Grass, Green)

FOL Provides

- **Variable symbols**
  - E.g., x, y, foo
- **Connectives**
  - Same as in PL: not ($\neg$), and ($\wedge$), or ($\vee$), implies ($\rightarrow$), if and only if (biconditional $\leftrightarrow$)
- **Quantifiers**
  - Universal $\forall$**x** or **(Ax)**
  - Existential $\exists$**x** or **(Ex)**

Sentences are built from terms and atoms

- A **term** (denoting a real-world individual) is a constant symbol, a variable symbol, or an n-place function of n terms.

  x and $f(x_1, ..., x_n)$ are terms, where each x_i is a term.

  A term with no variables is a **ground term**
- An **atomic sentence** (which has value true or false) is an n-place predicate of n terms
- A **complex sentence** is formed from atomic sentences connected by the logical connectives:

  $\neg P$, $P \vee Q$, $P \wedge Q$, $P \rightarrow Q$, $P \leftrightarrow Q$ where P and Q are sentences
- A **quantified sentence** adds quantifiers $\forall$ and $\exists$
- A **well-formed formula (wff)** is a sentence containing no "free" variables. That is, all variables are "bound" by universal or existential quantifiers.

  $(\forall x)P(x,y)$ has x bound as a universally quantified variable, but y is free.

Characteristics of WFF

•**Syntax Compliance**: WFFs adhere to the syntax rules of first-order logic, which define how terms, predicates, quantifiers, and logical connectives can be combined to form valid expressions.

•**Symbolic Representation**: WFFs consist of symbols representing terms (constants, variables, and functions), predicates (relations), quantifiers (∀, ∃), and logical connectives (∧, ∨, →, ¬).

•**Quantifier Scope**: WFFs maintain clear quantifier scope, ensuring that quantifiers bind variables appropriately within the formula. The scope of quantifiers affects the interpretation and meaning of the formula.

•**Complexity and Nesting**: WFFs can range from simple atomic formulas to complex nested structures involving multiple quantifiers and connectives. Proper nesting and grouping of subformulas are essential for clarity and unambiguous interpretation.

Quantifiers

- **Universal quantification**
  - $(\forall x)P(x)$ means that P holds for **all** values of x in the domain associated with that variable
  - E.g., $(\forall x)$ dolphin(x) $\rightarrow$ mammal(x)
- **Existential quantification**
  - $(\exists x)P(x)$ means that P holds for **some** value of x in the domain associated with that variable
  - E.g., $(\exists x)$ mammal(x) $\wedge$ lays-eggs(x)
  - Permits one to make a statement about some object without naming it

Quantifiers

- Universal quantifiers are often used with "implies" to form "rules":

  $(\forall x)$ student(x) $\rightarrow$ smart(x) means "All students are smart"

- Universal quantification is *rarely* used to make blanket statements about every individual in the world:

  $(\forall x)$student(x)$\wedge$smart(x) means "Everyone in the world is a student and is smart"

- Existential quantifiers are usually used with "and" to specify a list of properties about an individual:

  $(\exists x)$ student(x) $\wedge$ smart(x) means "There is a student who is smart"

- A common mistake is to represent this English sentence as the FOL sentence:

  $(\exists x)$ student(x) $\rightarrow$ smart(x)

  – But what happens when there is a person who is *not* a student?

Quantifier Scope

- Switching the order of universal quantifiers *does not* change the meaning:
  - $(\forall x)(\forall y)P(x,y) \leftrightarrow (\forall y)(\forall x)\ P(x,y)$
- Similarly, you can switch the order of existential quantifiers:
  - $(\exists x)(\exists y)P(x,y) \leftrightarrow (\exists y)(\exists x)\ P(x,y)$
- Switching the order of universals and existentials *does* change meaning:
  - Everyone likes someone: $(\forall x)(\exists y)$ likes(x,y)
  - Someone is liked by everyone: $(\exists y)(\forall x)$ likes(x,y)

Connections between All and Exists

We can relate sentences involving ∀ and ∃ using De Morgan's laws:

- $(\forall x)\ \neg P(x) \leftrightarrow \neg(\exists x)\ P(x)$
- $\neg(\forall x)\ P(x) \leftrightarrow (\exists x)\ \neg P(x)$
- $(\forall x)\ P(x) \leftrightarrow \neg\ (\exists x)\ \neg P(x)$
- $(\exists x)\ P(x) \leftrightarrow \neg(\forall x)\ \neg P(x)$

Quantified inference rules

- Universal instantiation
- Universal generalization
- Existential instantiation
- Existential generalization

Universal instantiation

- Meaning: If a statement is true for every element in a domain, it's also true for any particular element.

$$\frac{\forall x\ P(x)}{P(c)}$$

where 'c' is a specific constant.

- Example: If "All humans are mortal" ($\forall x\ (IsHuman(x) \rightarrow IsMortal(x))$)
- Then "Socrates is mortal" (IsMortal(Socrates)) can be inferred using a specific name for x.

Universal Generalization (UG)

If a statement can be shown to be true for an arbitrary element (not specific to it), then it must be true for all elements in the domain.

$$\frac{P(c)\ for\ an\ arbitrary\ c}{\forall x\ P(x)}$$

Explanation: If you can prove that a property (P(c)) holds for a constant (c) that has no special properties and was chosen arbitrarily, you can generalize that the property holds for all (x).

Example: If "Any student who gets an 'A' will pass" (P(c)) is true for an arbitrary student 'c', then it can be concluded that "All students will pass" (∀x P(x)).

Existential Instantiation (EI)

Meaning: If we know that at least one object exists with a certain property, we can assign a temporary, new name (a constant) to that object to reason about it.

$$\frac{\exists x\ P(x)}{P(c)\ for\ a\ new\ constant\ c}$$

where 'c' is a new, unassigned constant.

Example: If we know "There exists a student who loves AI" (∃x LovesAI(x)), we can name that student 'John' (LovesAI(John)) to infer further properties about that specific, currently unidentified student.

Existential Generalization (EG)

Meaning: If a statement is true for a specific, named object, we can conclude that there exists at least one object for which the statement is true.

$$\frac{P(c)}{\exists x\, P(x).}$$

Example: If "John teaches a course" (Teaches(John, x)), then it can be generalized that "There exists at least one course that John teaches" ($\exists x$ Teaches(John, x)).

Translating English to FOL

1. All boys like cricket

$\Rightarrow \forall x : boys(x) \rightarrow like(x, Cricket)$

2. Some boys like football

$\Rightarrow \exists x : boys(x) \wedge like(x, football)$

3. Some girls hate football

$\Rightarrow \exists x : girls(x) \wedge hate(x, football)$

4. All girls love pink

$\Rightarrow \forall x : girls(x) \rightarrow love(x, Pink)$

5. Every Person who buys a Policy is smart

$\Rightarrow \forall x \forall y : Person(x) \wedge Policy(y) \wedge buys(x, y) \rightarrow Smart(x)$

6. No Person buys expensive Policy

$\forall x \forall y : Person(x) \wedge Policy(y) \wedge expensive(y) \rightarrow \neg buys(x, y)$

Quantifiers

i) Universal Quantifiers
   $\forall$ for all such that ($\rightarrow$)

ii) Existential Quantifiers
   $\exists$ for some such that ($\wedge$)

$\Rightarrow$ Symbol	Word
$\neg$	not
$\wedge$	and
$\vee$	or
$\rightarrow$	Implies (If Then)
	iff

Translating English to FOL

Every gardener likes the sun.

$\forall x\ \text{gardener}(x) \rightarrow \text{likes}(x,\text{Sun})$

You can fool some of the people all of the time.

$\exists x\ \forall t\ \ \text{person}(x) \wedge \text{time}(t) \rightarrow \text{can-fool}(x,t)$

You can fool all of the people some of the time.

$\forall x\ \exists t\ (\text{person}(x) \rightarrow \text{time}(t) \wedge \text{can-fool}(x,t))$

$\forall x\ (\text{person}(x) \rightarrow \exists t\ (\text{time}(t) \wedge \text{can-fool}(x,t))$

Equivalent

All purple mushrooms are poisonous.

$\forall x\ (\text{mushroom}(x) \wedge \text{purple}(x)) \rightarrow \text{poisonous}(x)$

No purple mushroom is poisonous.

$\neg\exists x\ \text{purple}(x) \wedge \text{mushroom}(x) \wedge \text{poisonous}(x)$

$\forall x\ \ (\text{mushroom}(x) \wedge \text{purple}(x)) \rightarrow \neg\text{poisonous}(x)$

Equivalent

There are exactly two purple mushrooms.

$\exists x\ \exists y\ \text{mushroom}(x) \wedge \text{purple}(x) \wedge \text{mushroom}(y) \wedge \text{purple}(y) \wedge \neg(x=y) \wedge \forall z\ (\text{mushroom}(z) \wedge \text{purple}(z)) \rightarrow ((x=z) \vee (y=z))$

Clinton is not tall.

$\neg\text{tall}(\text{Clinton})$

X is above Y iff X is on directly on top of Y or there is a pile of one or more other objects directly on top of one another starting with X and ending with Y.

$\forall x\ \forall y\ \text{above}(x,y) \leftrightarrow (\text{on}(x,y) \vee\ \exists z\ (\text{on}(x,z) \wedge \text{above}(z,y)))$

Unification

- Unification is a process in which two terms, which may include variables, constants, and functions, are made identical by finding a substitution for the variables.
- A substitution is a mapping of variables to terms. For instance, if we have a variable (X) and we substitute it with a constant (a), the variable (X) is unified with (a).

Basic Terminology -

Term: A term can be a constant, a variable, or a compound term (a function with arguments).
Substitution: A mapping from variables to terms.
Unifier: A substitution that, when applied to two terms, makes them identical.
Most General Unifier (MGU): The simplest unifier that can be used to unify two terms, meaning that any other unifier can be derived from the MGU by further substitution

Unification Algorithm

The unification algorithm attempts to find the MGU for two terms. The algorithm involves recursively applying substitutions until the terms become identical or a conflict is found that prevents unification.

Steps in the Unification Algorithm:

1. **Initialization**: Start with the two terms you want to unify.
2. **Decompose Compound Terms**: If the terms are compound (i.e., functions with arguments), break them down into their constituent parts.
3. **Check for Conflicts**: If a variable is being unified with a term that contains that variable (occurs check), unification fails.
4. **Apply Substitutions**: Continuously apply the substitutions and simplify the terms until they are identical or no further simplification is possible.

Example

Let's consider a example involving predicates and quantifiers in first-order logic:

Expression 1: ∀x P(x, f(Y))
Expression 2: P(Z, f(a))

To unify these expressions, we need to find a substitution that makes them equal. The unification process involves matching the quantifiers and predicates and finding a set of variable assignments:

Match ∀x P(x, f(Y)) with P(Z, f(a)):
x unifies with Z (x/Z)
Y unifies with a (Y/a)

The resulting substitution is: **{x/Z, Y/a}.** Applying this substitution to both expressions gives us:

∀Z P(Z, f(a)) = P(Z, f(a))
The expressions are now unified.

Applications

1. **Logic Programming (Prolog):**
- Unification is the core mechanism for pattern matching in Prolog. When a query is made, Prolog uses unification to match the query with facts and rules in the database to infer new information or find solutions.

2. **Theorem Proving:**
- Automated theorem provers use unification to match premises with conclusions of rules, thereby deriving new statements and ultimately proving or disproving theorems.

3. **Natural Language Processing (NLP):**
- In NLP, unification is used for parsing and understanding sentences. Feature structures representing grammatical properties are unified to check for agreement and syntactic correctness.

4. **Type Inference in Programming Languages**:
- Unification is used in type inference algorithms to determine the type of expressions in statically typed languages like Haskell and ML.

Forward Chaining-Backward 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.

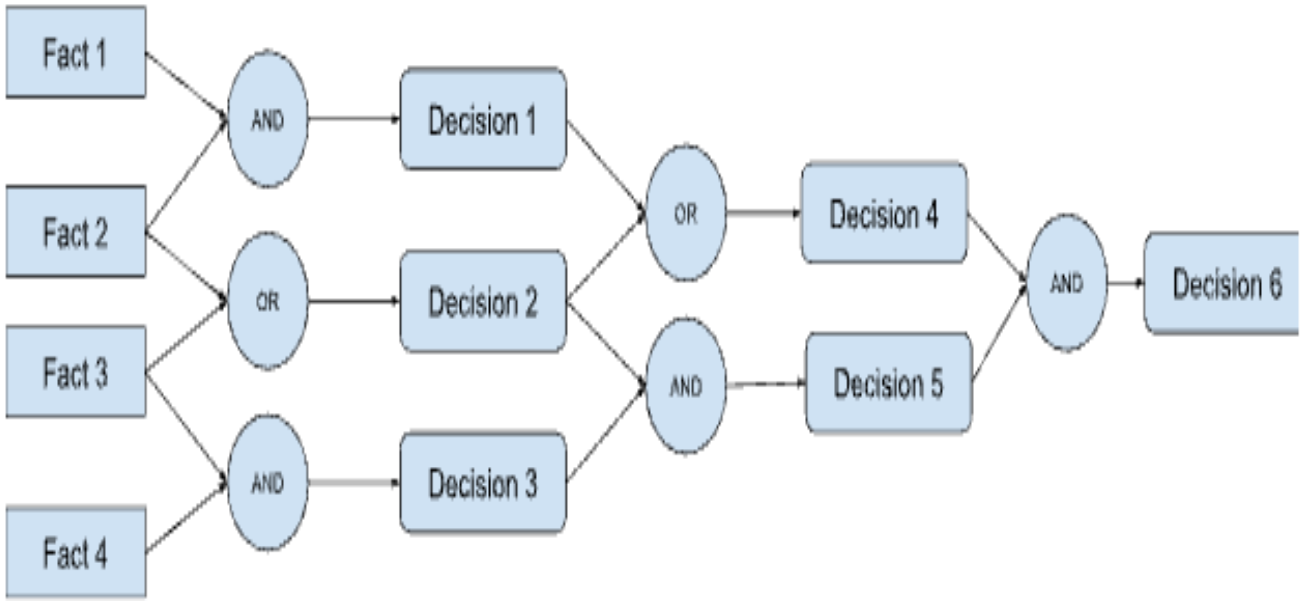
- The Forward-chaining algorithm starts from known facts, triggers all rules whose premises are satisfied, and add their conclusion to the known facts. This process repeats until the problem is solved

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 datadriven 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.



Example :As per the law, it is a crime for an American to sell weapon to hostile nations. Country A, an enemy of America has some missiles, and all the missiles were sold to it by Robert, who is an American Citizen." Prove that "Robert is Criminal"

Solution:

First, identify the facts in the problem and list it out. Second, convert the facts into First Order Logic.

1. It is a crime for an American to sell weapons to hostile nations :**American(P) ^ Weapon(Q) ^ Hostile(R) ^ Sells(P,Q,R) → Criminals(P)**

2. Country A has some missiles : **Owns(A,T1) , Missile(T1)**

3. All of the missiles were Sold to country A by Robert. :**Missiles(P) ^ Own(A,P) → Sells(Robert , P,A)**

4. Missiles are Weapons :**Missile(P) → Weapons(P)**

5) Enemy of America is known as hostile :**Enemy(P, America) → Hostile(P)**

6) Country A is an enemy of America :**Enemy(A, America)**

7) Robert is an American :**American(Robert)**

The above steps gave us the available facts and its first order logic representation, lets us now proceed in proving the statement “Robert is Criminal” through forward chaining.

Facts in First order Logic – Axioms

1. American(P) ^ Weapon(Q) ^ Hostile(R) ^ Sells(P,Q,R) → Criminals(P)
2. Owns(A,T1)
3. Missile(T1)
4. Missiles(P) ^ Own(A,P) → Sells(Robert ,P , A)
5. Missile(P) → Weapons(P)
6. Enemy(P , America) → Hostile(P)
7. Enemy(A , America)
8. American(Robert)

Proof: Step 1 : Start with Known Facts

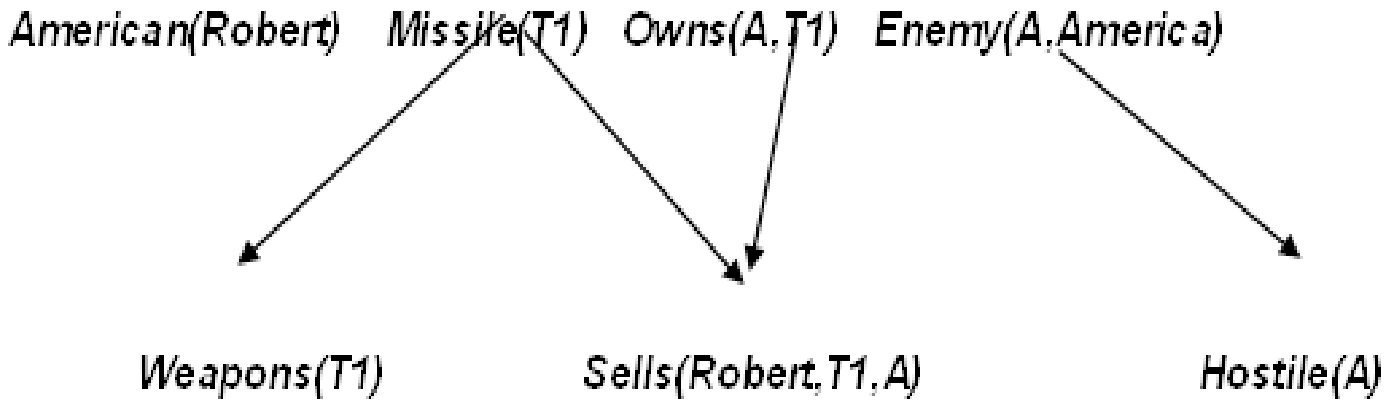
Facts number 2,3,7 and 8 are already known, so lets write it down first.

American(Robert) Missile(T1) Owns(A,T1) Enemy(A, America)

Step 2: Establish the connections,

By looking into the various facts in step 1 try to match it with the LHS or premise of our facts in first order logic and take its consequent or RHS.

For instance Missile(T1) matches with Fact number 5 so if T1 is a missile then from that axiom we can conclude T1 is a Weapon. Similarly do it form other facts too



Step 3 : The previous step gave us a way to apply axiom 1 and we can now conclude that

Robert is a criminal.