

Artificial Intelligence





CHAPTER-2

Search techniques – Knowledge representation





Basic Concepts

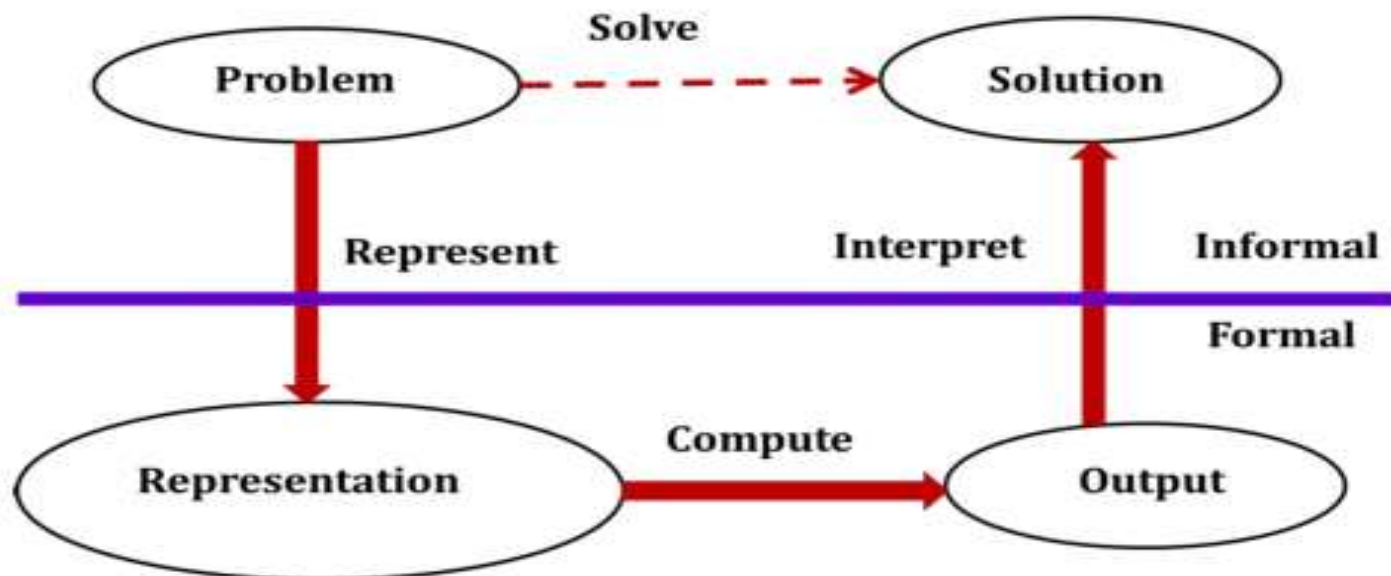
- ▶ In order to solve complex problems encountered in artificial intelligence, one needs both a large amount of **knowledge** and some **mechanism for manipulating that knowledge** to create solutions.
- ▶ Knowledge and Representation are two distinct entities.
- ▶ They play central but distinguishable roles in the **intelligent system**.
- ▶ Knowledge is a description of the world. It determines **a system's competence** by what it knows.
- ▶ Moreover, **Representation** is the way knowledge is encoded. It defines a **system's performance** in doing something.
- ▶ Different types of knowledge require **different kinds** of representation.
- ▶ The Knowledge Representation models/mechanisms are often based on: **Logic, Rules, Frames, Semantic Net**, etc.





Framework For Knowledge Representation

- Computer requires a well-defined problem description to process and provide well-defined acceptable solution.
- To collect fragments of knowledge we need first to formulate a description in our spoken language and then represent it in formal language so that computer can understand.
- The computer can then use an algorithm to compute an answer. This process is illustrated as,





Framework For Knowledge Representation

- The steps are:
 - The informal formalism of the problem takes place first.
 - It is then represented formally and the computer produces an output.
 - This output can then be represented in an informally described solution that user understands or checks for consistency.
- The Problem solving requires,
 - Formal knowledge representation, and
 - Conversion of informal knowledge to formal knowledge that is conversion of implicit knowledge to explicit knowledge.





Knowledge Representation in AI

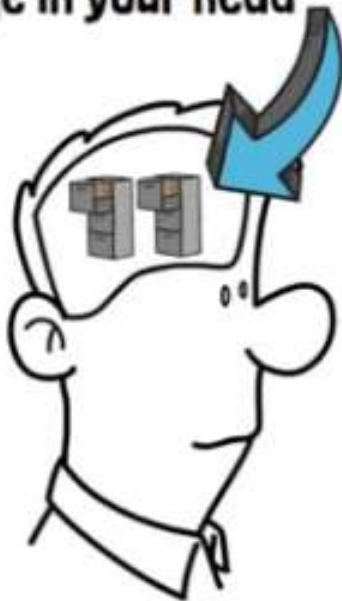
List of each guest and their dietary restriction

Guest Name	Dietary Restriction
John	No restriction
Bob	Vegetarian
Paul	Gluten-free
Alice	No Alcohol



Types of Knowledge

Knowledge in your head



Tacit knowledge

Knowledge written



or recorded

Explicit knowledge



Informal or Implicit

- Exists within a human being
- It is embodied.
- Difficult to articulate formally.
- Difficult to communicate or share.
- Hard to steal or copy.
- Drawn from experience, action, subjective insight

Formal or Explicit

- Exists outside a human being;
- It is embedded.
- Can be articulated formally.
- Can be shared, copied, processed and stored.
- Easy to steal or copy
- Drawn from artifact of some type as principle, procedure, process, concepts.

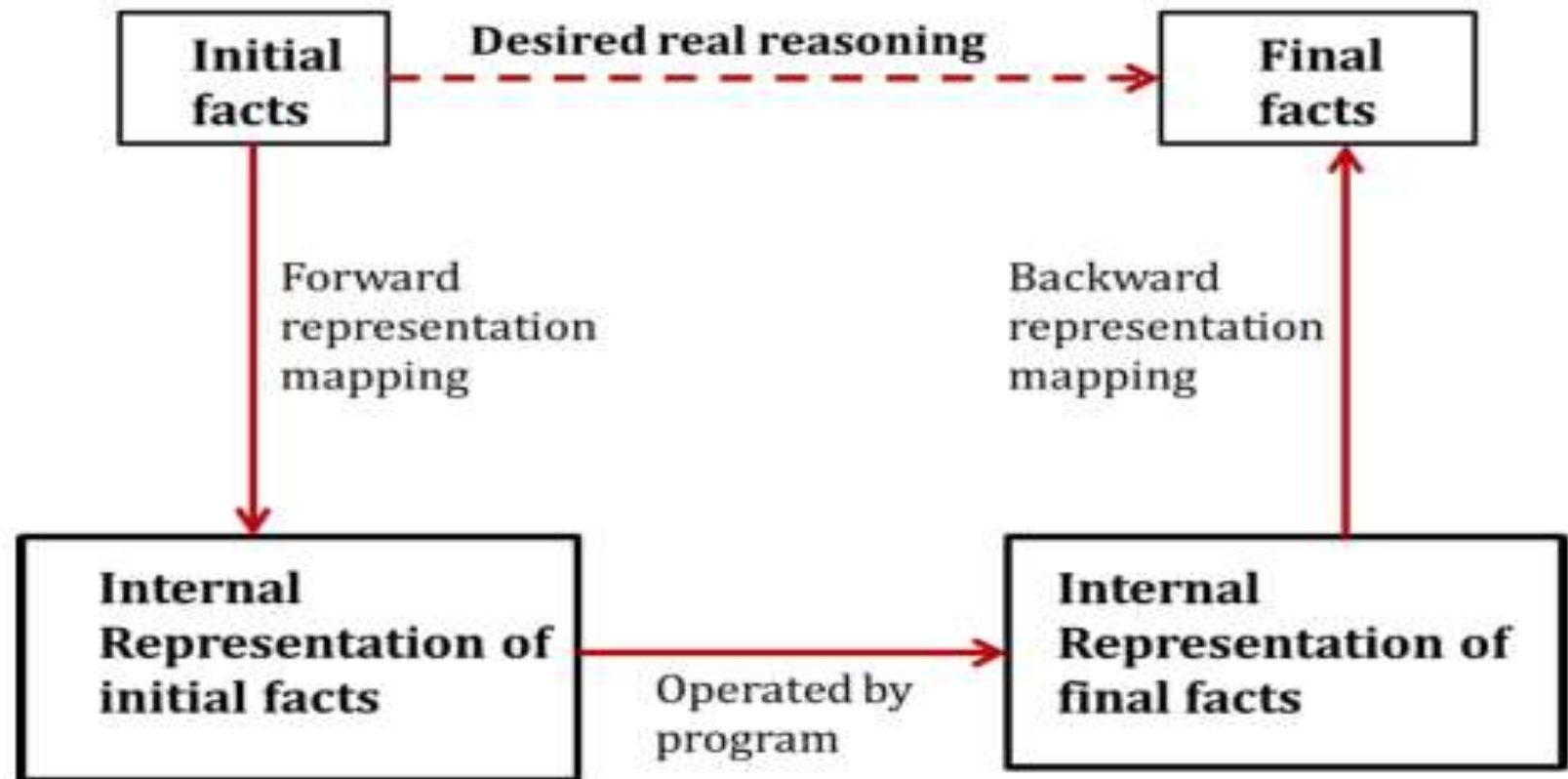
Representation And mapping(facts and representation)

- Knowledge is a collection of facts from some domain.
- We need a representation of "facts" that can be manipulated by a program.
- Normal English is insufficient, too hard currently for a computer program to draw inferences in natural languages.
- Thus some symbolic representation is necessary.





Representation of facts





- **Facts:** things we want to represent.
 - **Representations of facts:** things we can manipulate.
- Eg Sky is blue Sky(BLUE)
- Thus, knowledge representation can be considered at two levels :
 - Knowledge level at which facts are described, and
 - Symbol level at which the representations of the objects, defined in terms of symbols, can be manipulated in the programs.





- Spot is a dog
- Every dog has a tail
- Spot has a tail

[it is new knowledge]

Using backward mapping function to generate English sentence





- Spot is a dog
 $\text{dog}(\text{Spot})$
- Every dog has a tail
 $\forall x: \text{dog}(x) \rightarrow \text{hastail}(x)$
- Spot has a tail
 $\text{hastail}(\text{Spot})$
[it is new knowledge]

Using backward mapping function to generate English sentence





Knowledge Representation Paradigms


- Knowledge representation involves structuring information about the world, which can be understood by the computer and can be used for reasoning, learning, and problem solving.
- Key Paradigms are:
 - Logical Representation
 - Semantic Networks
 - Frame based Representation
 - Production Rules








1. Logical Representation:

- Uses formal languages like propositional logic and first-order logic to represent facts and relationships.
- Enables reasoning using logical inference to derive new knowledge
- For example, an expert system might use logical rules to diagnose a disease based on symptoms. 

2. Semantic Networks:

- Represent knowledge as a graph of nodes (concepts) and edges (relationships).
- Visually appealing and good for representing relationships between objects.
- Example: A network showing "dog" is a "mammal" which is an "animal". 





3. Frame-based Representation:

- Organizes knowledge into frames, which are data structures containing slots with associated values.
- Suitable for representing objects with attributes and default values.
- Example: A "car" frame might have slots for "color," "model," and "engine type". [🔗](#)

4. Production Rules (Rule-based Systems):

- Use "if-then" rules to represent knowledge and guide reasoning.
- Rules trigger actions or inferences when their conditions are met.
- Example: "If a user clicks on a product, then add it to their shopping cart". [🔗](#)





Approaches to knowledge representation

- ▶ A knowledge representation system should have following properties.
- 1. **Representational Adequacy** : The ability to represent all kinds of knowledge that are needed in that domain.
- 2. **Inferential Adequacy** : The ability to manipulate the representational structures to derive new structures corresponding to new knowledge inferred from old.
- 3. **Inferential Efficiency** : The ability to incorporate additional information into the knowledge structure that can be used to focus the attention of the inference mechanisms in the most promising direction.
- 4. **Acquisitional Efficiency** : The ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.





Representational Adequacy

- Representational adequacy refers to the ability of a knowledge representation system to effectively and accurately represent all the necessary knowledge within a specific domain.
- Essentially, it means the system can capture and store all the relevant information required for an AI to understand and reason about a particular subject area.
- Example: imagine trying to build a model of a car. Representational adequacy would mean your model can accurately represent all the parts of the car, their functions, and how they relate to each other. If you leave out the engine, your model lacks representational adequacy for a car that can actually move.





Inferential Adequacy

- A system's ability to derive new knowledge or information from existing knowledge through logical reasoning or inference.
- It essentially means the system can manipulate the stored information to draw conclusions and generate new insights.
- Example: If a system knows "all humans are mortal" and "Socrates is a human", it should be able to infer that "Socrates is mortal". Another example is if a system knows that "if it rains, the ground will be wet" and "it is raining", it should be able to infer that "the ground is wet".
- Inferential adequacy ensures that the knowledge representation is not just a static storage of facts, but a dynamic system capable of producing new knowledge through reasoning.





Inferential Efficiency

- How effectively an AI model utilizes its knowledge to make accurate predictions or decisions from new data, especially in a computationally efficient way.
- It's about the speed and accuracy with which the model can process information and arrive at a conclusion, minimizing both processing time and resource consumption.
- A highly inferentially efficient AI model is one that can quickly and accurately draw conclusions from new data while minimizing resource usage. This is a crucial characteristic for AI systems to be practical and effective in real-world applications.





Acquisitional Efficiency

- The ability of a knowledge representation system or AI model to effectively and efficiently acquire new knowledge or information.
- In other words, an acquisitionally efficient system should be able to rapidly and accurately learn from new data or experience.
- A system with high acquisitional efficiency can automatically learn new information, adapt to changing circumstances, and improve its performance over time.





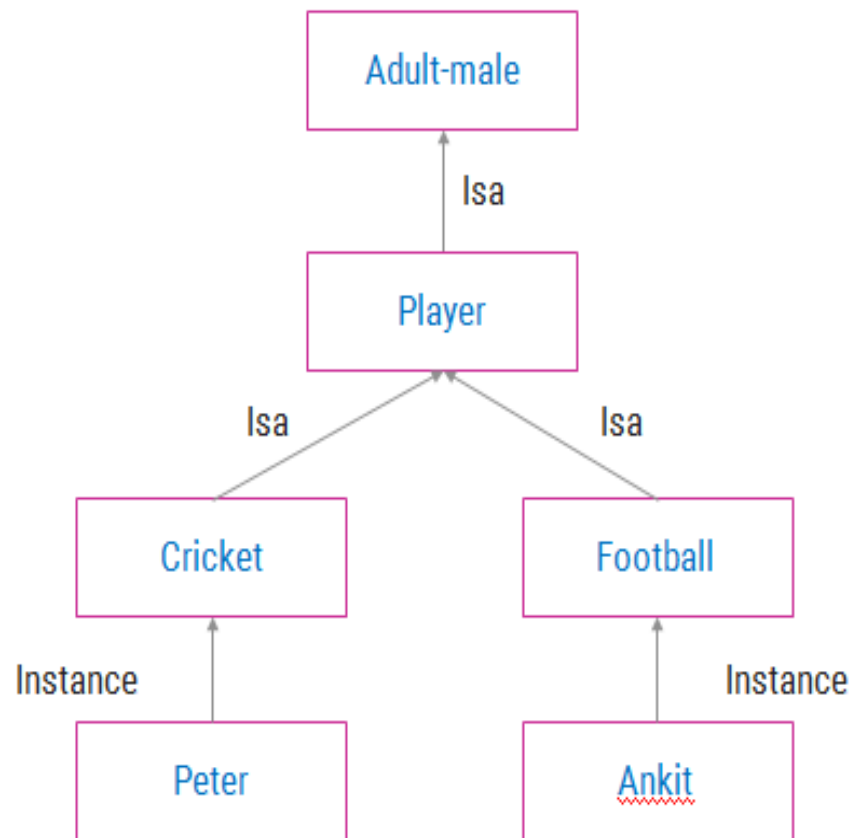
Knowledge representation schemes

- **Relational Knowledge** : The simplest way to represent declarative facts is as a set of relations of the same sort used in the database system.

Player	Height	Weight	Bats - Throws
Aaron	6-0	180	Right - Right
Mays	5-10	170	Right - Right
Ruth	6-2	215	Left - Left
Williams	6-3	205	Left - Right



► **Inheritable Knowledge** : Here the knowledge elements inherit attributes from their parents.



▶ **Inferential Knowledge** : This knowledge generates new information from the given information. This new information does not require further data gathering from source, but does require analysis of the given information to generate new knowledge.

▶ **Example:**

- given a set of relations and values, one may infer other values or relations.
- A predicate logic (a mathematical deduction) is used to infer from a set of attributes.
- Inference through predicate logic uses a set of logical operations to relate individual data.

▶ **Represent knowledge as formal logic:**

All dogs have tails :: $\forall x: \text{dog}(x) \rightarrow \text{hastail}(x)$

○ **Advantages:**

- A set of strict rules.
- Can be used to derive more facts.
- Truths of new statements can be verified.
- Guaranteed correctness.





- ▶ **Procedural Knowledge** : A representation in which the control information, to use the knowledge, is embedded in the knowledge itself.
- ▶ For example, computer programs, directions, and recipes.
- ▶ Knowledge is encoded in some procedures, small programs that know how to do specific things, how to proceed.

Cooking:

Knowing how to bake a cake involves a sequence of steps: preheating the oven, mixing ingredients, baking for a specific time, etc.

Playing a musical instrument:

Learning to play a piano involves understanding how to position your fingers, press keys, and coordinate movements.





Propositional Logic

► Logic

- The logical formalism of a language is useful because it immediately suggests a powerful way of deriving new knowledge from old using mathematical deduction.
- In this formalism, we can conclude that a new statement is true by proving that it follows from the statements that are already known.

► Proposition

- A proposition is a statement, or a simple declarative sentence.
- For example, "the book is expensive" is a proposition.
- A proposition can be either true or false





1. Delhi is the capital of USA
2. How are you doing
3. $5 \leq 11$
4. Temperature is less than 10 C
5. It is cold today
6. Read this carefully
7. $X + y = z$





1. Delhi is the capital of USA ✓
2. How are you doing ✗
3. $5 \leq 11$ ✓
4. Temperature is less than 10 C ✓
5. It is cold today ✗
6. Read this carefully ✗
7. $X + y = z$ ✗





Atomic Propositions :

- Atomic propositions are the simple propositions. It consists of a single proposition symbol. These are the sentences which must be either true or false.
 - $2+2$ is 4 it is an atomic proposition as it is a true fact.
 - "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.
 - "It is raining today, and street is wet."
 - "Ankit is a doctor, and his clinic is in Mumbai."





- Logical connectives are used to connect two simpler propositions or representing a sentence logically.
- We can create compound propositions with the help of logical connectives.
- There are mainly five connectives, which are given as follows

Connective symbols	Word	Technical term	Example
\wedge	AND	Conjunction	$A \wedge B$
\vee	OR	Disjunction	$A \vee B$
\rightarrow	Implies	Implication	$A \rightarrow B$
\Leftrightarrow	If and only if	Biconditional	$A \Leftrightarrow B$
\neg or \sim	Not	Negation	$\neg A$ or $\neg B$



Example

- **Conjunction**

- **Example:** Rohan is intelligent and hardworking. It can be written as,
- **P= Rohan is intelligent,**
Q= Rohan is hardworking.
- so we can write it as **$P \wedge Q$** .

- **Disjunction**

- **Example:** "Ritika is a doctor or Engineer"
- **P= Ritika is Doctor.**
- **Q= Ritika is Doctor,**
- so we can write it as **$P \vee Q$** .



- **Negation**

- **Geeta is not a engineer**
- A sentence such as $\neg P$ is called negation of P

- **Implication**

- **Example: it is raining, then the street is wet.**
- Let $P =$ It is raining
 $Q =$ Street is wet
- so it is represented as $P \rightarrow Q$

- **Biconditional**

- **Example: If I am breathing, then I am alive**
- $P =$ I am breathing, $Q =$ I am alive,
- it can be represented as $P \Leftrightarrow Q$.



Example of Propositional Logic(PL)

Representing simple facts

It is raining
RAINING

It is sunny
SUNNY

It is windy
WINDY

If it is raining, then it is not sunny
 $\text{RAINING} \rightarrow \neg \text{SUNNY}$





Example of Propositional Logic(PL)

Weather examples

1. It is hot.
2. It is humid.
3. It is raining.

If it is humid, then it is hot.



If it is hot and humid then it is not raining





Weather examples

1. It is hot. - A
2. It is humid. - B
3. It is raining. - c

If it is humid, then it is hot.

If it is hot and humid then it is not raining.





Weather examples

1. It is hot. - A
2. It is humid. - B
3. It is raining. - c

If it is humid, then it is hot.

$B \rightarrow A$

If it is hot and humid then it is not raining.

$(A \wedge B) \rightarrow \neg C$





Limitations of Propositional Logic(PL)

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





Predicate Logic(First Order Logic)

- ▶ First-order Predicate logic (FOPL) is a formal language in which propositions are expressed in terms of **predicates, variables and quantifiers**.
- ▶ It is different from propositional logic which lacks quantifiers.
- ▶ It should be viewed as an extension to propositional logic, in which the notions of truth values, logical connectives, etc. still apply but propositional letters will be replaced by a newer notion of proposition involving predicates and quantifiers.
- ▶ **A predicate is an expression of one or more variables defined on some specific domain.**
- ▶ A predicate with variables can be made up of a proposition by either assigning a value to the variable or by quantifying the variable.





In predicate logic, a predicate is a statement or expression that contains variables and becomes a proposition (either true or false) when specific values are assigned to those variables or when quantified. Predicates represent properties or relationships of objects. @

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., $(\forall x) \text{dolphin}(x) \rightarrow \text{mammal}(x)$

- Existential quantification

$(\exists x)P(x)$ means that P holds for some value of x in the domain associated with that variable

E.g., $(\exists x) \text{mammal}(x) \wedge \text{lays-eggs}(x)$





Example 1: Let $P(x)$ be the predicate " $x > 5$ " where x is a real number.

$P(7)$ is true because $7 > 5$

$P(3)$ is false because 3 is not > 5

Example 2: Let $Q(x,y)$ be the predicate " $x + y = 10$ " where x and y are integers.

$Q(3,7)$ is true because $3 + 7 = 10$

$Q(4,5)$ is false because $4 + 5 \neq 10$





Quantifiers:

There are two main types of quantifiers:

Universal Quantifier (\forall): "for all"

Existential Quantifier (\exists): "there exists"

Example 3: Let $R(x)$ be the predicate " $x^2 \geq 0$ " where x is a real number.

The statement $\forall x R(x)$ is true because for all real numbers, their square is always non-negative.

Example 4: Let $S(x)$ be the predicate " $x^2 = 4$ " where x is a real number.

The statement $\exists x S(x)$ is true because there exist real numbers (2 and -2) whose square is 4.





Example 5: Consider the predicate $P(x,y)$: " $x + y = 0$ " where x and y are integers.

The statement $\forall x \exists y P(x,y)$ is true because for any integer x , we can always find an integer y such that their sum is 0 (y would be $-x$).

Example 6: Let $Q(x)$ be the predicate " x is prime" where x is a positive integer.

The statement $\forall x Q(x)$ is false because not all positive integers are prime.

The statement $\exists x Q(x)$ is true because there exist prime numbers (e.g., 2, 3, 5, 7, etc.).





► **Well Formed Formula (wff)** is a predicate holding any of the following -

- All propositional constants and propositional variables are wffs
- If x is a variable and Y is a wff, $\forall Y$ and $\exists x$ are also wff
- Truth value and false values are wffs
- Each atomic formula is a wff
- All connectives connecting wffs are wffs





Sentences to well formed formulas(wff) in Predicate logic

1. Marcus was a man.
2. Marcus was a Pompeian.
3. All Pompeian were Romans.
4. Caesar was a ruler.
5. All Romans were either loyal to Caesar or hated him.
6. Everyone is loyal to someone.
7. People only try to assassinate rulers they are not loyal to.
8. Marcus tried to assassinate Caesar.
9. All men are people





Sentences to well formed formulas(wff) in Predicate logic

1. Marcus was a man.

1. $man(Marcus)$

2. Marcus was a Pompeian.

2. $Pompeian(Marcus)$

3. All Pompeian were Romans.

3. $\forall x : Pompeian(x) \rightarrow Roman(x)$

4. Caesar was a ruler.

4. $ruler(Caesar)$

5. All Romans were either loyal to Caesar or hated him. 5. $\forall x : Roman(x) \rightarrow loyalto(x, Caesar) \vee hate(x, Caesar)$

6. Everyone is loyal to someone. 6. $\forall x : \exists y : loyalto(x, y)$

7. People only try to assassinate rulers they are not loyal to.

7. $\forall x : \forall y : person(x) \wedge ruler(y) \wedge tryassassinate(x, y) \rightarrow \neg loyalto(x, y)$

8. Marcus tried to assassinate Caesar. 8. $tryassassinate(Marcus, Caesar)$

9. All men are people

9. $\forall x : man(x) \rightarrow person(x)$





Predicate Calculus

Predicate calculus, also known as First-Order Logic (FOL). It's an extension of propositional logic that allows for a much richer and more nuanced way of expressing facts, relationships, and rules about the world.

Key Components of Predicate Calculus:

- Predicates:** These represent properties of objects or relationships between objects. They take one or more arguments.

Examples:

IsHuman(John): "John is a human." (property)

Married(John, Mary): "John is married to Mary." (relationship)

GreaterThan(5, 3): "5 is greater than 3." (relationship)





•**Variables:** These are symbols that stand for any object in a given domain. They are typically denoted by lowercase letters (e.g., x, y).

Example: IsStudent(x): "x is a student." (Here, x can represent any student.)

•**Constants:** These are symbols that refer to specific, fixed objects or entities in the domain. They are typically denoted by uppercase letters or specific names (e.g., John, Mary, Table).

•**Functions:** These represent mappings from a number of entities to a single entity. They return a value.

Examples:

FatherOf(John): refers to John's father.

Plus(2, 3): refers to the number 5.





- Quantifiers:** These allow us to make statements about collections of objects.

Universal Quantifier (\forall): Means "for all" or "for every."

Existential Quantifier (\exists): Means "there exists" or "for some."

- Logical Connectives:** These are the same as in propositional logic, used to combine atomic formulas into more complex ones.

\neg (**Negation**): "not"

\wedge (**Conjunction**): "and"

\vee (**Disjunction**): "or"

\Rightarrow (**Implication**): "if...then"

\Leftrightarrow (**Biconditional**): "if and only if"





Predicate And Argument

- In Artificial Intelligence, particularly within **First-Order Logic (FOL)** or **Predicate Calculus**, predicates and arguments are the core building blocks for representing knowledge about the world.
- Just as a sentence in human language has a verb (the action or state) and nouns (the participants), a logical statement in predicate calculus has a predicate and its arguments.





1. Predicates

A **predicate** is a symbol that represents a **property** of an object or a **relationship** between one or more objects. It essentially describes what is true about something or how things relate to each other.

- Role:** Predicates are like verbs or adjectives in a natural language sentence. They assert something about the entities they refer to.

- Truth Value:** A predicate, when applied to its arguments, evaluates to either **true** or **false**. This makes it a foundational element for logical reasoning.





•**Arity:** Predicates have an "arity," which is the number of arguments they take.

□**Unary Predicate (Arity 1):** Describes a property of a single object.

- Example: IsRed(apple) - "The apple is red."
- Example: IsHuman(Socrates) - "Socrates is human."

□**Binary Predicate (Arity 2):** Describes a relationship between two objects.

- Example: Likes(John, Mary) - "John likes Mary."
- Example: GreaterThan(5, 3) - "5 is greater than 3."

□**Ternary Predicate (Arity 3):** Describes a relationship among three objects, and so on.

- Example: Between(Delhi, Agra, Mathura) - "Agra is between Delhi and Mathura."





•Examples of Predicates in AI Knowledge Representation:

- Rainy(Today): "It is rainy today."
- Student(Mary): "Mary is a student."
- Parent(John, Alice): "John is a parent of Alice."
- LivesIn(Person, City): "A person lives in a city."
- Color(Car, Red): "The car's color is red."





2. Arguments

Arguments are the entities, objects, or variables that the predicate is making a statement about. They are the "subjects" or "objects" of the logical statement. Arguments are enclosed in parentheses after the predicate symbol and are separated by commas.

•**Role:** Arguments provide the specific context for the predicate. They tell us *who* or *what* the predicate is referring to.

Types of Arguments:

•**Constants:** These are specific, named individuals or objects in the domain. They are typically denoted by capitalized words or specific names.

Examples: John, Mary, Socrates, Apple, Delhi, 5.





•**Variables:** These are placeholders that can stand for any object in a given domain. They are typically denoted by lowercase letters (e.g., x , y , z). Variables are crucial when making general statements using quantifiers (like "for all" or "there exists").

Example: $\text{IsHuman}(x)$ - Here, x is a variable that can represent any human.

•**Functions:** Functions in predicate calculus map one or more arguments to a single object. The result of a function can also serve as an argument to a predicate.

Example: $\text{FatherOf}(\text{John})$ - This function refers to John's father.

So, $\text{IsHuman}(\text{FatherOf}(\text{John}))$ would mean "John's father is human."





•Examples of Arguments in AI Knowledge Representation:

In the predicate statements below, the bolded terms are the arguments:

IsRed(**apple**)

Likes(**John**, **Mary**)

GreaterThan(5, 3)

LivesIn(**Person**, **City**)

Color(**Car**, **Red**)

Mortal(**Socrates**)





ISA Hierarchy

- An **ISA hierarchy** (pronounced "is-a") is a fundamental way of organizing knowledge about concepts and objects in a structured, hierarchical manner.
- It represents a special type of relationship where one concept or object is a *kind of* another.
- This relationship is crucial because it facilitates a powerful form of reasoning known as **inheritance**.
- The "ISA" relationship signifies a **subclass-superclass** or **subtype-supertype** connection. If "A ISA B" is true, it means that A is a more specific type of B. Every property that is true for B is also true for A.





Examples:

Dog ISA Mammal (A dog is a kind of mammal)

Mammal ISA Animal (A mammal is a kind of animal)

Car ISA Vehicle (A car is a kind of vehicle)

Apple ISA Fruit (An apple is a kind of fruit)

Fruit ISA Food (A fruit is a kind of food)





ISA Hierarchy

•An **ISA hierarchy** (pronounced "is-a") is a fundamental way of organizing knowledge about concepts and objects in a structured, hierarchical manner.

How ISA Hierarchies are Structured:

An ISA hierarchy typically forms a **tree-like** or **graph-like structure** where:

Nodes: Represent concepts or categories (e.g., "Animal," "Mammal," "Dog," "Poodle").

Edges (or Links): Represent the "ISA" relationship, pointing from the more specific concept (subclass) to the more general concept (superclass).





Example - ISA Hierarchy





Frame Notation

- Frame is a record like structure but specifically designed for knowledge representation.
- Collection of attributes and its values to describe an entity
- Each frame represents a particular entity or concept and contains a collection of **slots** and their corresponding **fillers**.





Key Components of a Frame:

•**Frame Name:** This is the identifier for the concept or entity being represented.

Examples: CAR, RESTAURANT, BIRD, MEDICAL_DIAGNOSIS.

•**Slots (Attributes/Properties):** These are the characteristics or properties that describe the entity represented by the frame. Each slot represents a piece of information related to the concept.

Examples for a CAR frame: Number_of_Wheels, Color, Engine_Type, Fuel_Type, Manufacturer, Model.

Examples for a RESTAURANT frame: Cuisine, Menu, Seating_Capacity, Waitstaff, Bill_Payment_Method.





•**Fillers (Values):** These are the specific values or information that fill the slots. Fillers can be:

Specific Values: Concrete data (e.g., Color: Red, Number_of_Wheels: 4).

Default Values: Common or typical values that are assumed if no specific value is provided (e.g., Number_of_Wheels: 4 for a CAR frame). This allows for efficient representation of common knowledge and handling of missing information.

Range Constraints: Specifies the possible types or range of values a slot can take (e.g., Age: (INTEGER, 0-150)).





Lecturer	
Name:	Prof Jones
Tolerance:	Intolerant
If intolerant, then turn off mobile phone	
If intolerant, then pay attention	

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

Slots	Filter
Name	Peter
Profession	Doctor
Age	25
Marital status	Single
Weight	78





Advantages of Frame Notation

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





Disadvantages of Frame Notation

1. In frame system inference mechanism is **not be easily processed**.
2. Inference mechanism cannot be smoothly represented.
3. Frame representation has a **much generalized approach**.





Resolution

- The fundamental idea behind resolution is to find a contradiction (or unsatisfiability) by systematically deriving new clauses until an empty clause (representing falsity) is produced.
- If an empty clause can be derived, then the original set of clauses is unsatisfiable. Conversely, if no empty clause can be derived and the process terminates, the set is satisfiable.
- **Resolvent:** The new clause derived from two parent clauses by resolving away a pair of complementary literals.





Steps for resolution

1. Conversion of facts into first-order logic.
2. Convert FOL statements into CNF (Conjunctive Normal Form) is a way to represent logical statements as a conjunction (AND) of disjunctions (OR).
3. Negate the statement which needs to prove (proof by contradiction)
4. Draw resolution graph (unification).
5. Whenever we get an empty clause, stop and report the original theorem is true.





Propositional Logic-resolution refutation

Example 1:

- Consider the following Knowledge Base:
 1. The humidity is high or the sky is cloudy.
 2. If the sky is cloudy, then it will rain.
 3. If the humidity is high, then it is hot.
 4. It is not hot.**Goal:** It will rain.





Propositional Logic-resolution refutation

Step 1 : Conversion to first order logic

1. The humidity is high or the sky is cloudy.
2. If the sky is cloudy, then it will rain.
3. If the humidity is high, then it is hot.
4. It is not hot.

- Let **P** : The humidity is high
- Let **Q**: sky is cloudy

**The humidity is high or
the sky is cloudy**

P V Q



Propositional Logic-resolution refutation

Conversion to first order logic...

1. The humidity is high or the sky is cloudy.
2. If the sky is cloudy, then it will rain.
3. If the humidity is high, then it is hot.
4. It is not hot.

Goal: It will rain.

- Let **Q**: sky is cloudy
- Let **R**: it will rain

If the sky is cloudy, then it will rain

$$Q \rightarrow R$$



Propositional Logic-resolution refutation

Conversion to first order logic...

1. The humidity is high or the sky is cloudy.
 2. If the sky is cloudy, then it will rain.
 3. If the humidity is high, then it is hot.
 4. It is not hot.
- **Goal:** It will rain.

- Let **P** : The humidity is high
- Let **S**: it is hot

If the humidity is high,
then it is hot

$$P \rightarrow S$$





Propositional Logic-resolution refutation

Conversion to first order logic...

1. The humidity is high or the sky is cloudy.
2. If the sky is cloudy, then it will rain.
3. If the humidity is high, then it is hot.
4. It is not hot.

• **Goal:** It will rain.

• **Let S:** it is hot

It is not hot

$\neg S$





Propositional Logic-resolution refutation

Statements in first order logic

1. The humidity is high or the sky is cloudy.
2. If the sky is cloudy, then it will rain.
3. If the humidity is high, then it is hot.
4. It is not hot.

• **Goal:** It will rain. (R)

• $P \vee Q$

• $Q \rightarrow R$

• $P \rightarrow S$

• $\neg S$



Propositional Logic-resolution refutation

Step 2:

Convert it to CNF

- $P \vee Q$

- $Q \rightarrow R$

- $P \rightarrow S$

- $\neg S$

- $P \vee Q$

- $\neg Q \vee R$

- $\neg P \vee S$

- $\neg S$

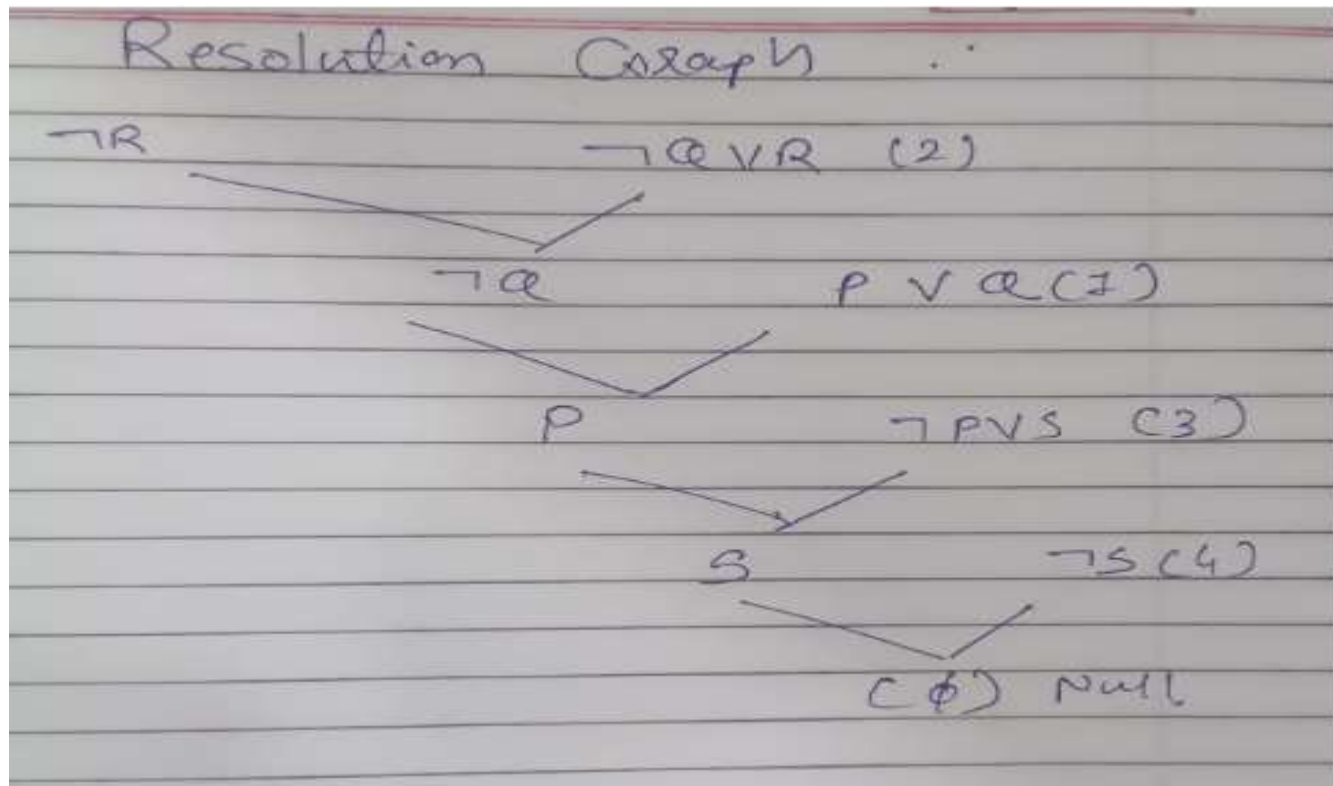
Negation of Goal ($\neg R$): It will not rain.





Propositional Logic-resolution refutation

Step 3,4





Propositional Logic-resolution refutation

Example 2

1) IF it's raining (R), then the ground is wet (W)

~~R → W~~ (W)
2) IF the ground is wet, people use umbrella (U).

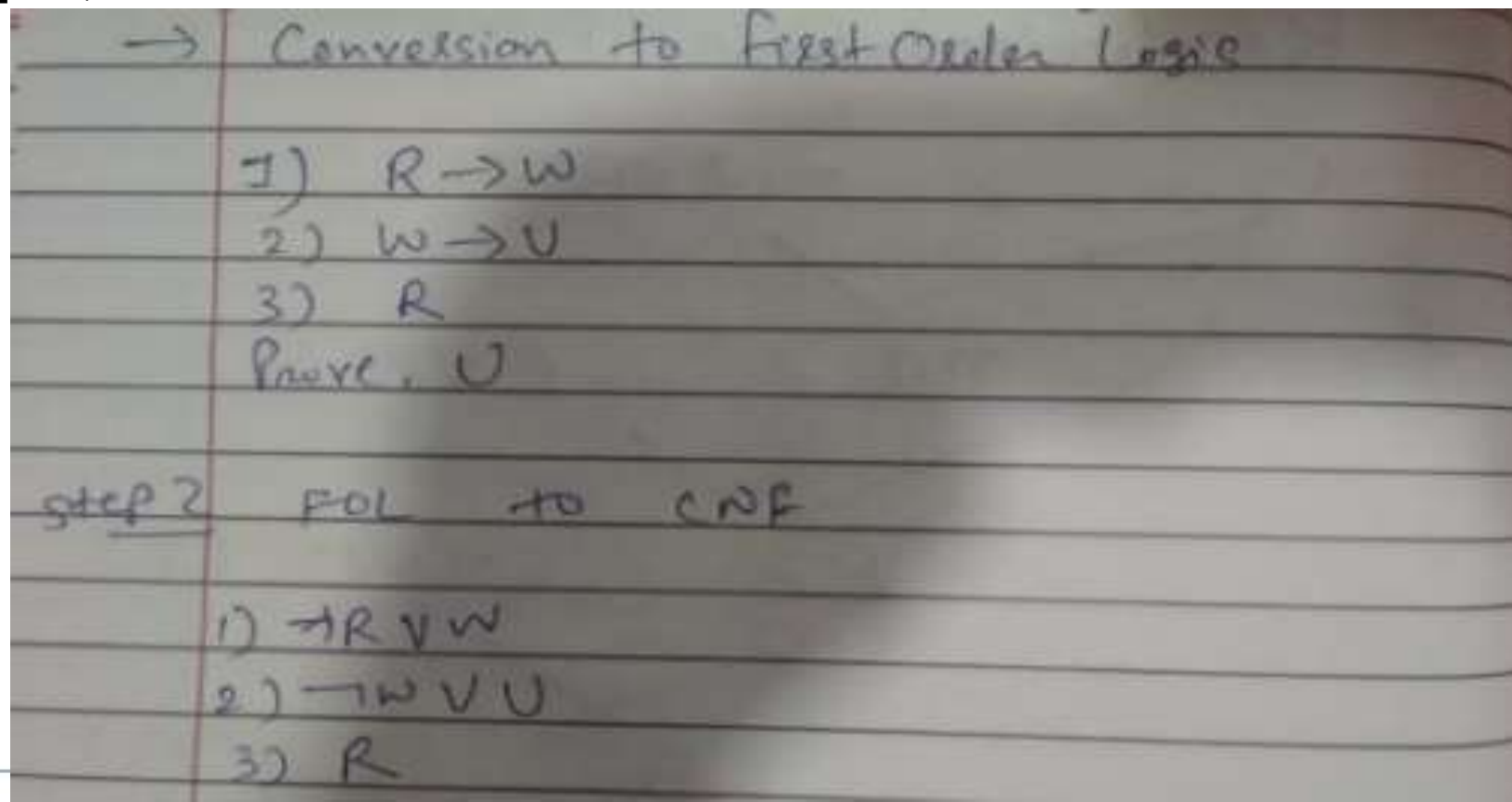
3) It is raining (R)

Prove: - People Use umbrella (U)



Propositional Logic-resolution refutation

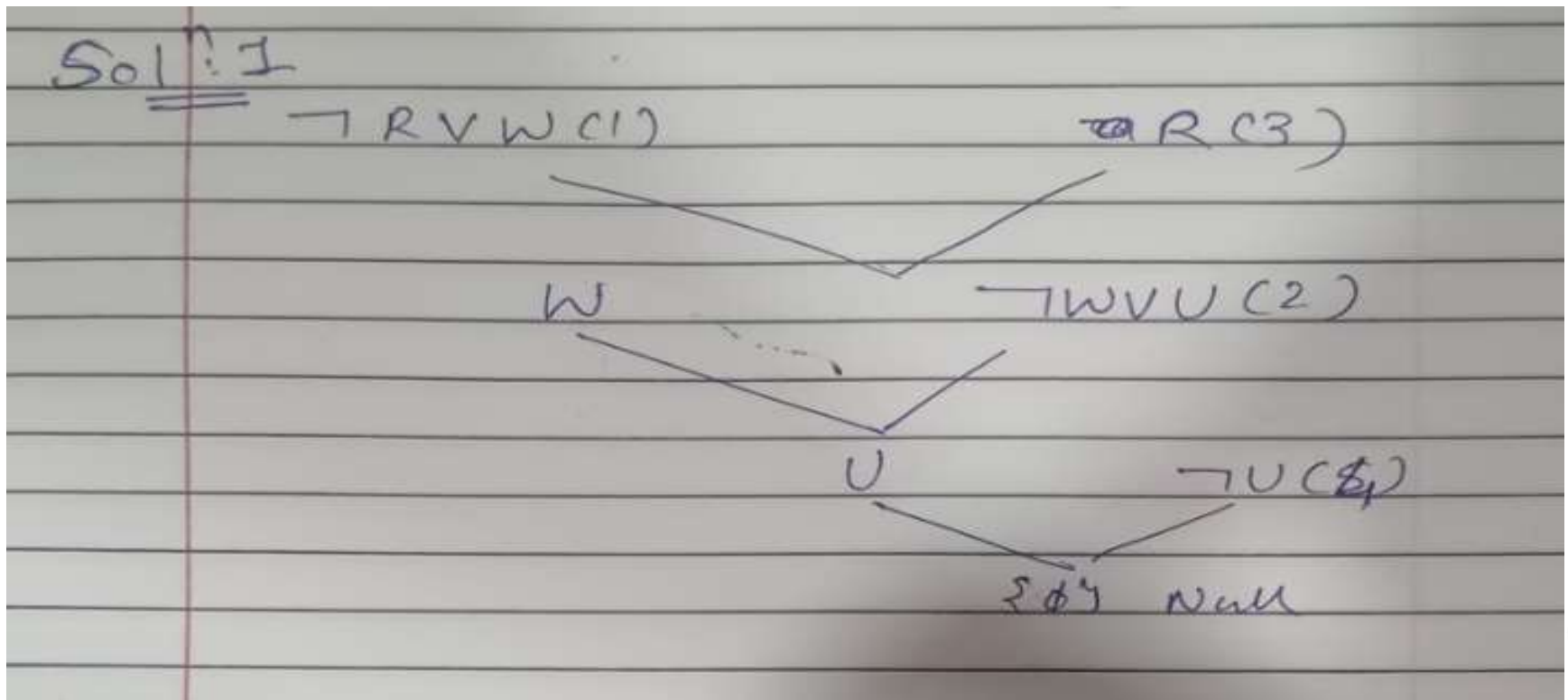
Step 1,2:





Propositional Logic-resolution refutation

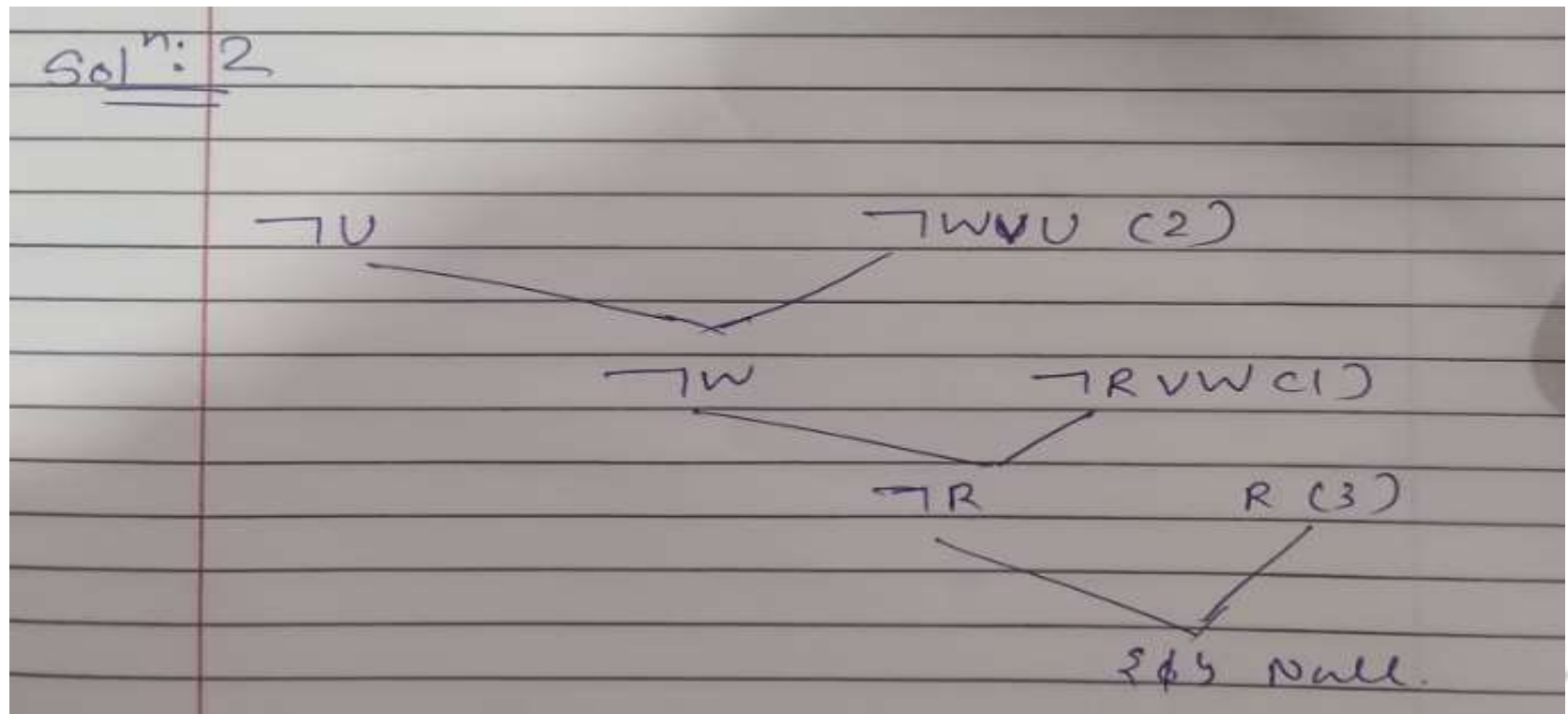
Step 3,4 (Solution 1):





Propositional Logic-resolution refutation

Step 3,4 (Solution 2):





Propositional Logic-resolution refutation

Example 3:

1. If It is sunny and warm day you will enjoy
2. If it is raining you will get wet
3. It is warm day
4. It is raining
5. It is sunny

Goal: You will enjoy

Prove: enjoy





Predicate Logic-resolution refutation

Step 1 : Conversion to first order logic

- If It is sunny and warm day you will enjoy
Sunny \wedge warm \rightarrow enjoy
- If it is raining you will get wet
raining \rightarrow wet
- It is warm day
warm
- It is raining
raining
- It is sunny
Sunny





Predicate Logic-resolution refutation

Step 2 : conversion to CNF

- **Sunny \wedge Warm \rightarrow enjoy**
Eliminate implication:
 $\neg(\text{Sunny} \wedge \text{warm}) \vee \text{enjoy}$
Moving negation inside
 $\neg\text{Sunny} \vee \neg\text{warm} \vee \text{enjoy}$
- **raining \rightarrow wet**
Eliminate implication:
 $\neg\text{raining} \vee \text{wet}$
- **warm**
- **raining**
- **Sunny**

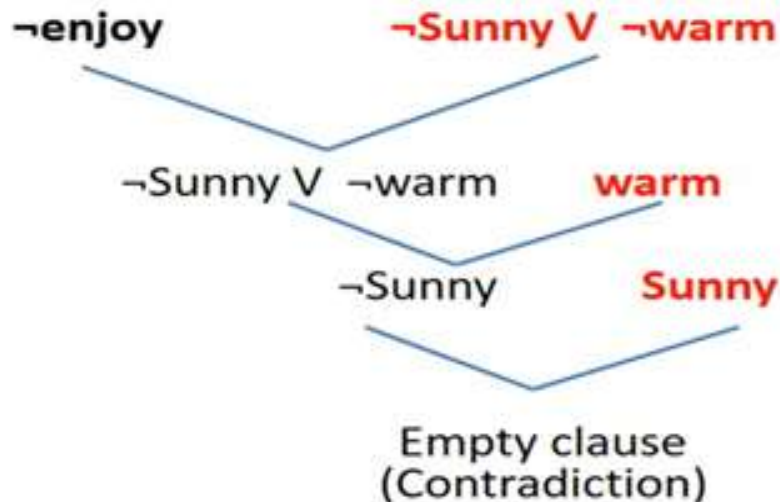




Propositional Logic-resolution refutation

Step: 3 & 4 Resolution graph

- Negate the statement to be proved: $\neg \text{enjoy}$
- Now take the statements one by one and create resolution graph.



After applying Proof by Refutation (Contradiction) on the goal, the problem is solved, and it has terminated with a **Null clause** (\emptyset). Hence, the goal is achieved.





Predicate Logic-resolution refutation

Example 1:

1. All dogs are animals.
2. Fido is a dog.
3. All animals will die.





Predicate Logic-resolution refutation

Step 1

Predicate Form

$\forall X(\text{dog}(X) \rightarrow \text{animal}(X))$

$\text{dog}(\text{fido})$

$\forall Y(\text{animal}(Y) \rightarrow \text{die}(Y))$





Predicate Logic-resolution refutation

Step 2

Now we want to prove that "Fido will die". So to apply resolution we first convert the above statements into clause form, which are given as

1. $\neg \text{dog}(X) \vee \text{animal}(X)$
2. $\text{dog}(\text{fido})$
3. $\neg \text{animal}(Y) \vee \text{die}(Y)$

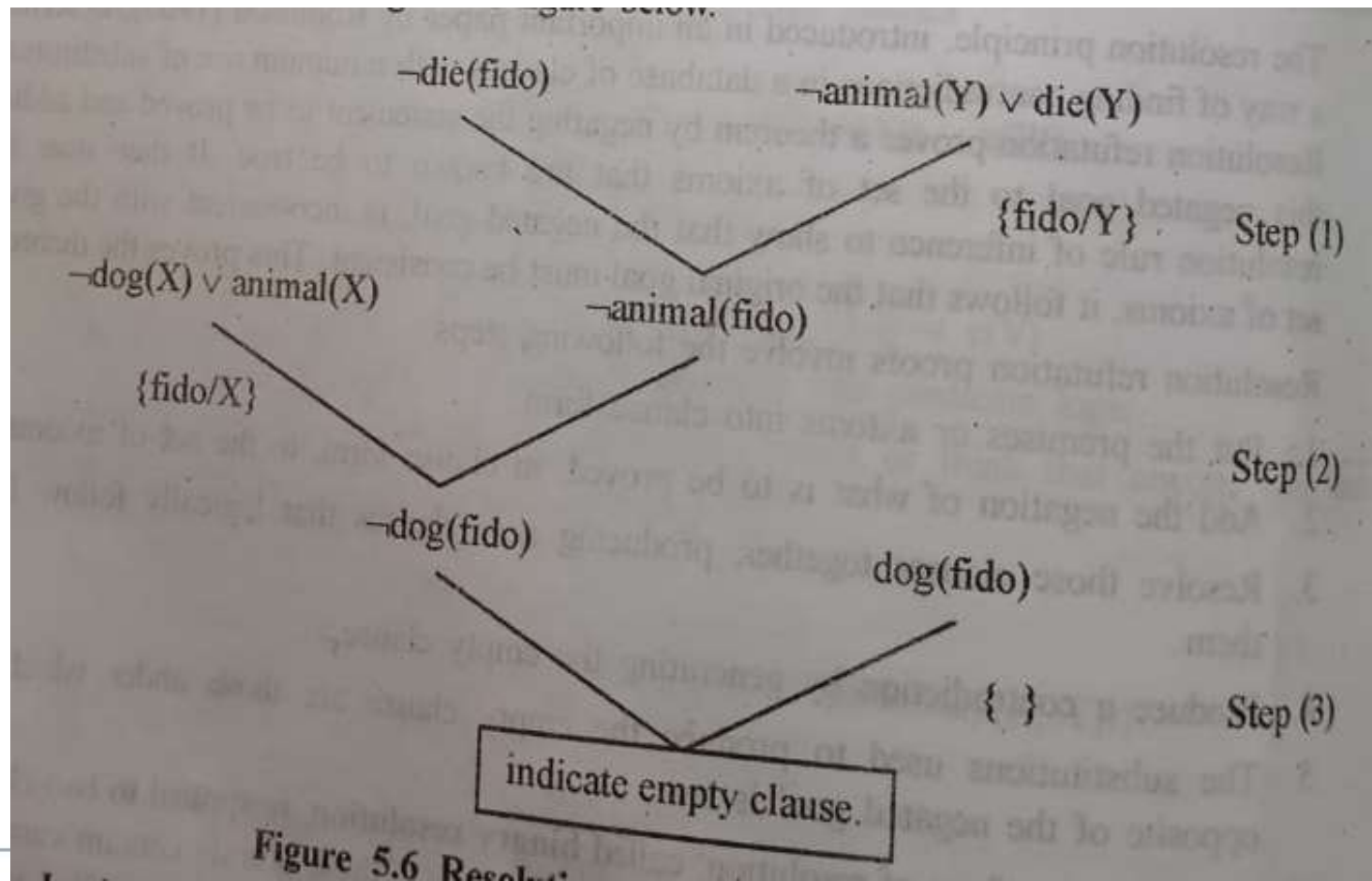
Now to get "Fido will die", we add negation $\neg \text{die}(\text{fido})$ with our knowledge base and apply resolution as given in figure below.





Predicate Logic-resolution refutation

Step 3,4





Predicate Logic-resolution refutation

Example 2

1. Sunil likes all kind of food.
 2. Apple and vegetable are food
 3. Anything anyone eats and not killed is food.
 4. Anil eats peanuts and still alive
 5. Sohan eats everything that Anil eats.
- **Prove by resolution that:**
Sunil likes peanuts





Predicate Logic-resolution refutation

Conversion of Facts/Statements into FOL

1. Sunil likes all kind of food.
2. Apple and vegetable are food
3. Anything anyone eats and not killed is food.
4. Anil eats peanuts and still alive
5. Sohan eats everything that Anil eats
6. Sunil likes peanuts.

1. $\forall x : \text{food}(x) \rightarrow \text{likes}(\text{Sunil}, x)$
2. $\text{food}(\text{apple}) \wedge \text{food}(\text{vegetable})$
3. $\forall x \forall y : \text{eats}(x, y) \wedge \neg \text{killed}(x) \rightarrow \text{food}(y)$
4. $\text{eats}(\text{Anil}, \text{Peanut}) \wedge \text{alive}(\text{Anil})$
5. $\forall x : \text{eats}(\text{Anil}, x) \rightarrow \text{eats}(\text{Sohan}, x)$
6. $\text{Likes}(\text{Sunil}, \text{Peanut})$
- $\forall x : \neg \text{killed}(x) \rightarrow \text{alive}(x)$
- $\forall x : \text{alive}(x) \rightarrow \neg \text{killed}(x)$
(added predicates)



Predicate Logic-resolution refutation

Conversion of FOL into CNF: Elimination of \rightarrow

1. $\forall x : \text{food}(x) \rightarrow \text{likes}(\text{Sunil}, x)$
2. $\text{food}(\text{apple}) \wedge \text{food}(\text{vegetable})$
3. $\forall x \forall y : \text{eats}(x, y) \wedge \neg \text{killed}(x) \rightarrow \text{food}(y)$
4. $\text{eats}(\text{Anil}, \text{Peanut}) \wedge \text{alive}(\text{Anil})$
5. $\forall x : \text{eats}(\text{Anil}, x) \rightarrow \text{eats}(\text{Sohan}, x)$
6. $\forall x : \neg \text{killed}(x) \rightarrow \text{alive}(x)$
7. $\forall x : \text{alive}(x) \rightarrow \neg \text{killed}(x)$
8. $\text{Likes}(\text{Sunil}, \text{Peanut})$

1. $\forall x \neg \text{food}(x) \vee \text{likes}(\text{Sunil}, x)$
2. $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$
3. $\forall x \forall y \neg [\text{eats}(x, y) \wedge \neg \text{killed}(x)] \vee \text{food}(y)$
4. $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$
5. $\forall x \neg \text{eats}(\text{Anil}, x) \vee \text{eats}(\text{Sohan}, x)$
6. $\forall x \neg [\neg \text{killed}(x)] \vee \text{alive}(x)$
7. $\forall x \neg \text{alive}(x) \vee \neg \text{killed}(x)$
8. $\text{likes}(\text{Sunil}, \text{Peanuts})$





Predicate Logic-resolution refutation

Move negation (\neg) inwards and rewrite

1. $\forall x \neg \text{food}(x) \vee \text{likes}(\text{Sunil}, x)$
2. $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$
3. $\forall x \forall y \neg [\text{eats}(x, y) \wedge \neg \text{killed}(x)] \vee \text{food}(y)$
4. $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$
5. $\forall x \neg \text{eats}(\text{Anil}, x) \vee \text{eats}(\text{Sohan}, x)$
6. $\forall x \neg [\neg \text{killed}(x)] \vee \text{alive}(x)$
7. $\forall x \neg \text{alive}(x) \vee \neg \text{killed}(x)$
8. $\text{likes}(\text{Sunil}, \text{Peanuts}).$

1. $\forall x \neg \text{food}(x) \vee \text{likes}(\text{Sunil}, x)$
2. $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$
3. $\forall x \forall y \neg \text{eats}(x, y) \vee \text{killed}(x) \vee \text{food}(y)$
4. $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$
5. $\forall x \neg \text{eats}(\text{Anil}, x) \vee \text{eats}(\text{Sohan}, x)$
6. $\forall x \text{killed}(x) \vee \text{alive}(x)$
7. $\forall x \neg \text{alive}(x) \vee \neg \text{killed}(x)$
8. $\text{likes}(\text{Sunil}, \text{Peanuts}).$





Predicate Logic-resolution refutation

Drop Universal quantifiers

1. $\forall x \neg \text{food}(x) \vee \text{likes}(\text{Sunil}, x)$
2. $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$
3. $\forall y \forall z \neg \text{eats}(y, z) \vee \text{killed}(y) \vee \text{food}(z)$
4. $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$
5. $\forall w \neg \text{eats}(\text{Anil}, w) \vee \text{eats}(\text{Sohan}, w)$
6. $\forall g \text{killed}(g) \vee \text{alive}(g)$
7. $\forall k \neg \text{alive}(k) \vee \neg \text{killed}(k)$
8. $\text{likes}(\text{Sunil}, \text{Peanuts})$.

1. $\neg \text{food}(x) \vee \text{likes}(\text{Sunil}, x)$
2. $\text{food}(\text{Apple})$
3. $\text{food}(\text{vegetables})$
4. $\neg \text{eats}(y, z) \vee \text{killed}(y) \vee \text{food}(z)$
5. $\text{eats}(\text{Anil}, \text{Peanuts})$
6. $\text{alive}(\text{Anil})$
7. $\neg \text{eats}(\text{Anil}, w) \vee \text{eats}(\text{Sohan}, w)$
8. $\text{killed}(g) \vee \text{alive}(g)$
9. $\neg \text{alive}(k) \vee \neg \text{killed}(k)$
10. $\text{likes}(\text{Sunil}, \text{Peanuts})$

Statements " $\text{food}(\text{Apple}) \wedge \text{food}(\text{vegetables})$ " and " $\text{eats}(\text{Anil}, \text{Peanuts}) \wedge \text{alive}(\text{Anil})$ " can be written in two separate statements.





Predicate Logic-resolution refutation

Negate the statement to be proved

- Prove by resolution that: **Sunil likes peanuts.**

likes(Sunil, Peanuts).

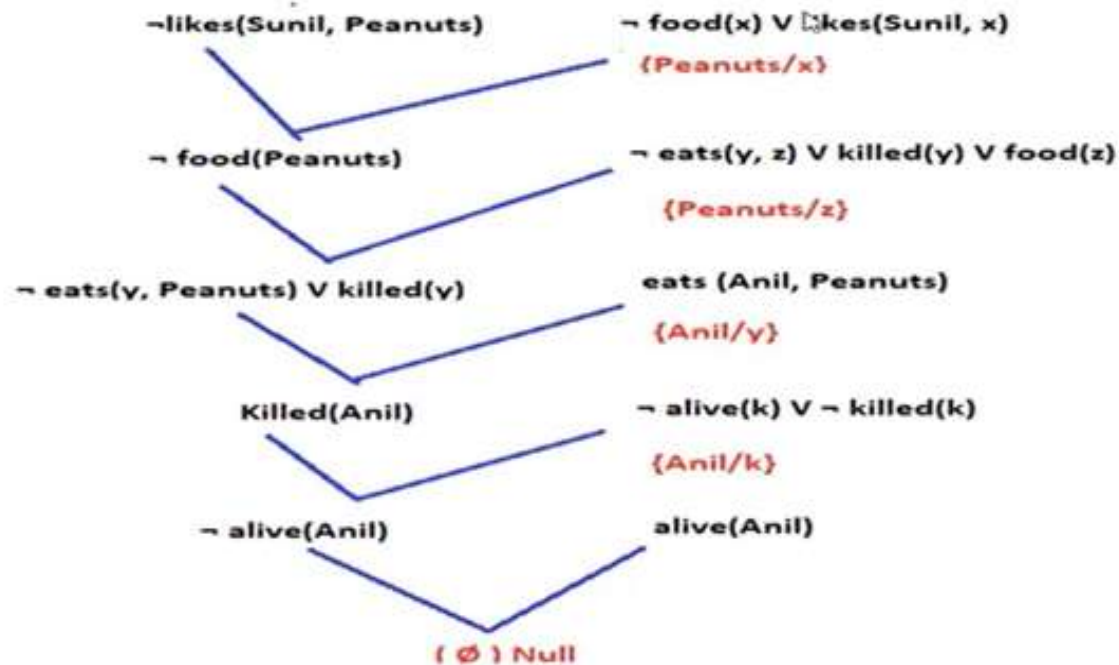
¬likes(Sunil, Peanuts)





Predicate Logic-resolution refutation

Resolution graph





Parul[®] University

