Artificial Intelligence

Parul® University

CHAPTER-3

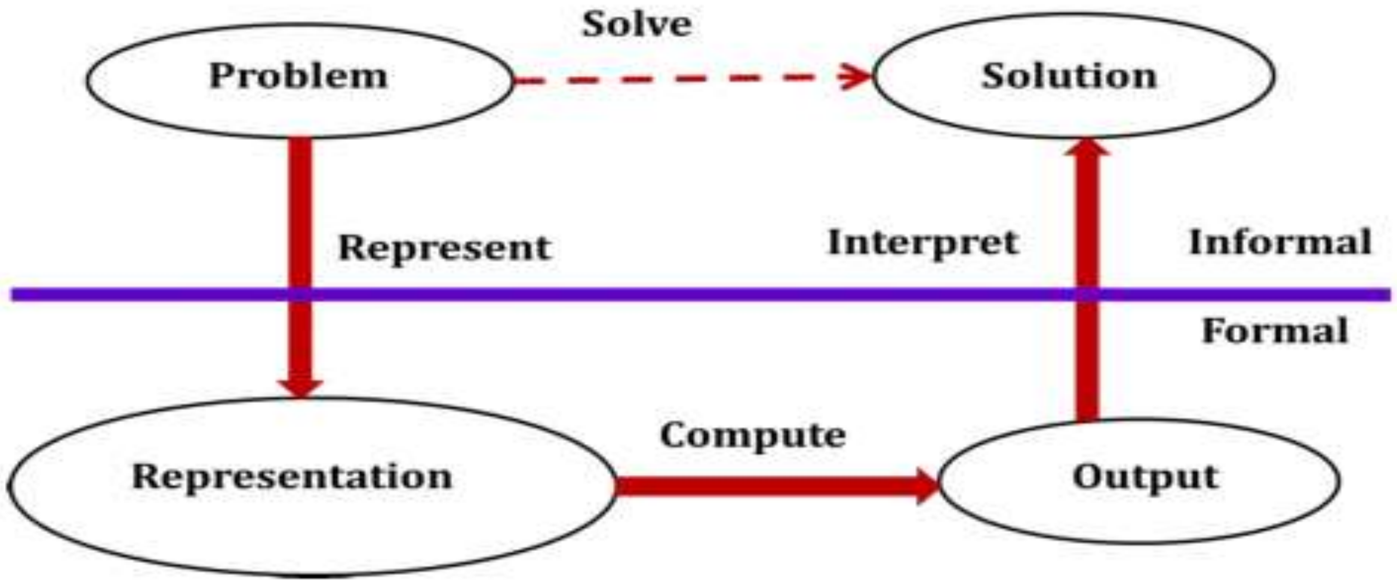
Knowledge representation

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

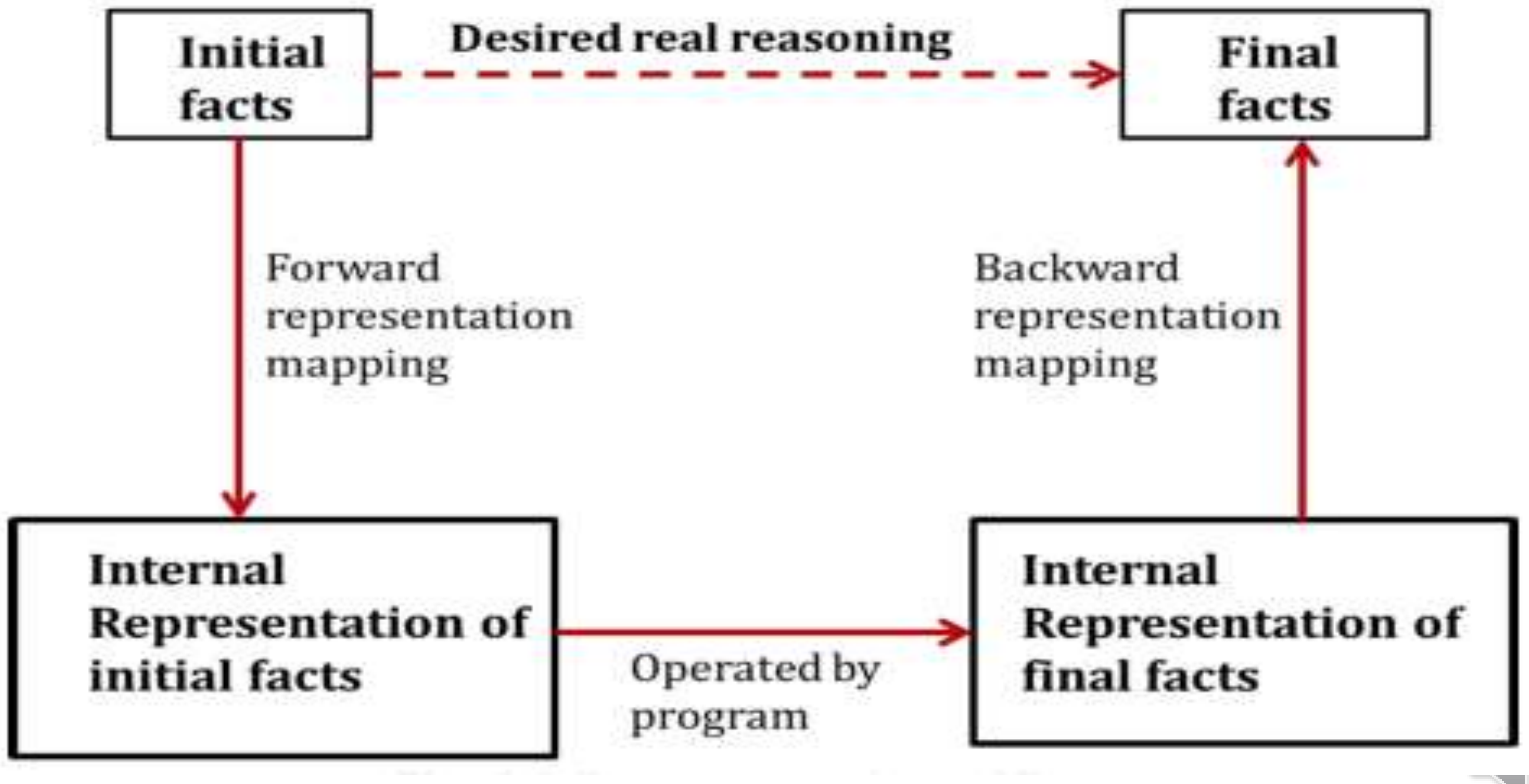
List of each guest and their dietary restriction

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

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


Initial facts
Desired real reasoning
Final facts
Forward representation mapping
Backward representation mapping
Internal Representation of initial facts
Operated by program
Internal Representation of final 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

  dog(Spot)

- Every dog has a tail

  $\forall x$: dog(x) $\rightarrow$ hastail(x)

- Spot has a tail

  hastail(Spot)

  [it is new knowledge]

Using backward mapping function to generate English sentence

Approaches(Properties) 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

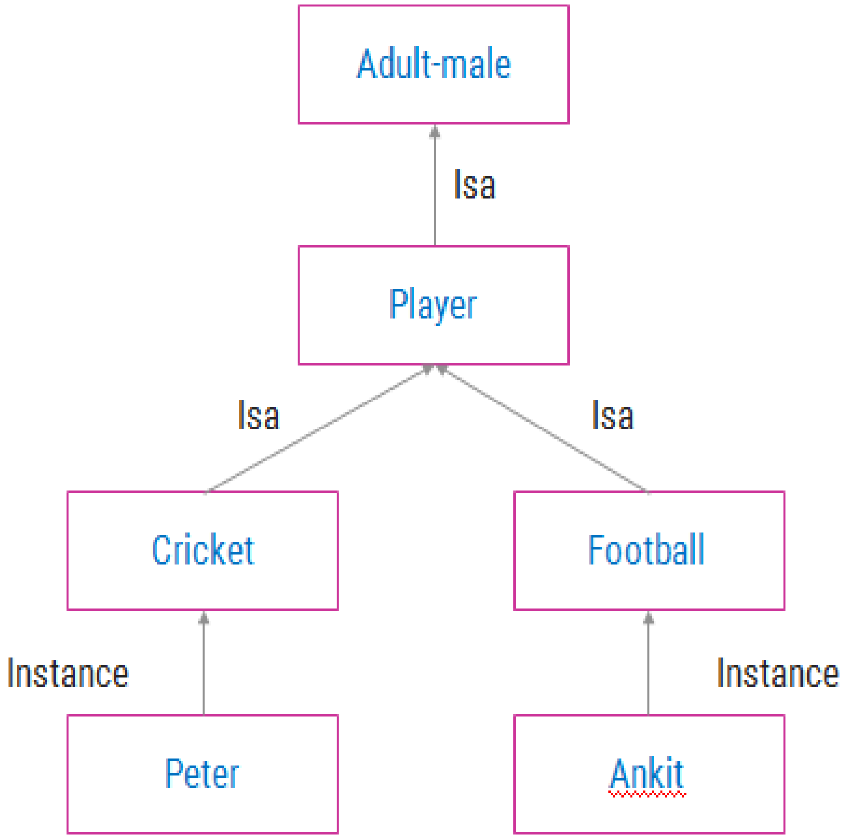
•The ability of a knowledge representation system or AI model to effectively and efficiently acquire new knowledge or information.
• In other words, an acquisitionally efficient system should be able to rapidly and accurately learn from new data or experience.
•A system with high acquisitional efficiency 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 form 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 :: $\forall x: dog(x) \rightarrow hastail(x)$

- 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.

Issues in knowledge representation

1. Scope and Granularity: The real world is vast and complex. Deciding which facts, objects, relationships, and processes are relevant to a given problem is a significant challenge.
   Should we represent knowledge at a high, abstract level (e.g., "Birds can fly") or at a very detailed, low level (e.g., "A robin's wing muscles contract to generate lift")?
2. **Expressiveness vs. Tractability (Computational Efficiency):**

Expressiveness: A KR language should be expressive enough to capture all the necessary nuances of the domain. First-Order Logic (FOL), for example, is highly expressive.

Tractability: However, the more expressive a representation, the more computationally expensive it often becomes to perform inference and reasoning over it. Finding a balance between what can be expressed and what can be efficiently reasoned about is a core dilemma

Issues in knowledge representation

3. Handling Uncertainty and Incompleteness:
Uncertainty: The real world is rarely black and white. Knowledge is often uncertain, probabilistic, or fuzzy. Traditional logical representations struggle with this (e.g., "It will probably rain today").
Incompleteness: AI systems rarely have complete knowledge. They must be able to reason and make decisions even with missing information.
Approaches: This has led to the development of probabilistic graphical models (Bayesian networks), fuzzy logic, and other methods for handling uncertainty.

Issues in knowledge representation

4. Common Sense Knowledge:
Humans possess a vast amount of intuitive common sense knowledge about how the world works (e.g., "objects fall down," "people have one head"). This knowledge is incredibly difficult to formalize and represent for AI systems.
Brittleness: AI systems often lack common sense, making them "brittle" – they perform well within their narrow domain of programmed knowledge but fail dramatically when faced with situations outside that scope.

Issues in knowledge representation

5. Knowledge Acquisition and Learning:

Knowledge Acquisition Bottleneck: Manually encoding large amounts of knowledge is extremely time-consuming, expensive, and prone to errors. This is often called the "knowledge acquisition bottleneck."

Integration with Learning: How can AI systems automatically acquire, refine, and update their knowledge from data and experience? Integrating KR with machine learning is a major research direction, moving from static, hand-coded knowledge bases to dynamic, adaptable ones.

Issues in knowledge representation

6. Representing Relationships and Context:
Relationships among attributes: How do attributes of objects relate to each other? For example, the height attribute of a person relates to their age and gender.
Context: The meaning of knowledge often depends on the context. Representing and reasoning about context is crucial for understanding natural language, interpreting sensory data, and making appropriate decisions.
7. **Dealing with Time and Change:**
The world is dynamic. Events happen, states change, and knowledge evolves over time. Representing temporal information, causality, and concurrency accurately is complex.

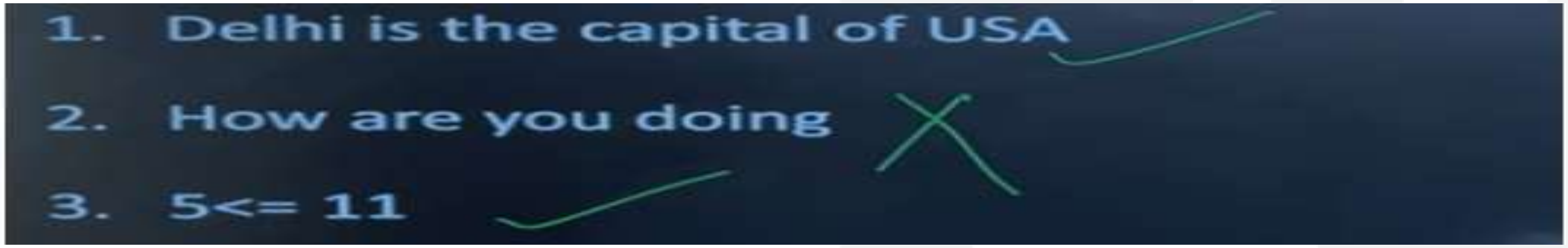
Issues in knowledge representation

8. Scalability:

As the amount of knowledge increases, efficient storage, retrieval, and reasoning become critical. Large-scale knowledge bases require sophisticated indexing, distributed systems, and optimized inference engines.

Propositional Logic

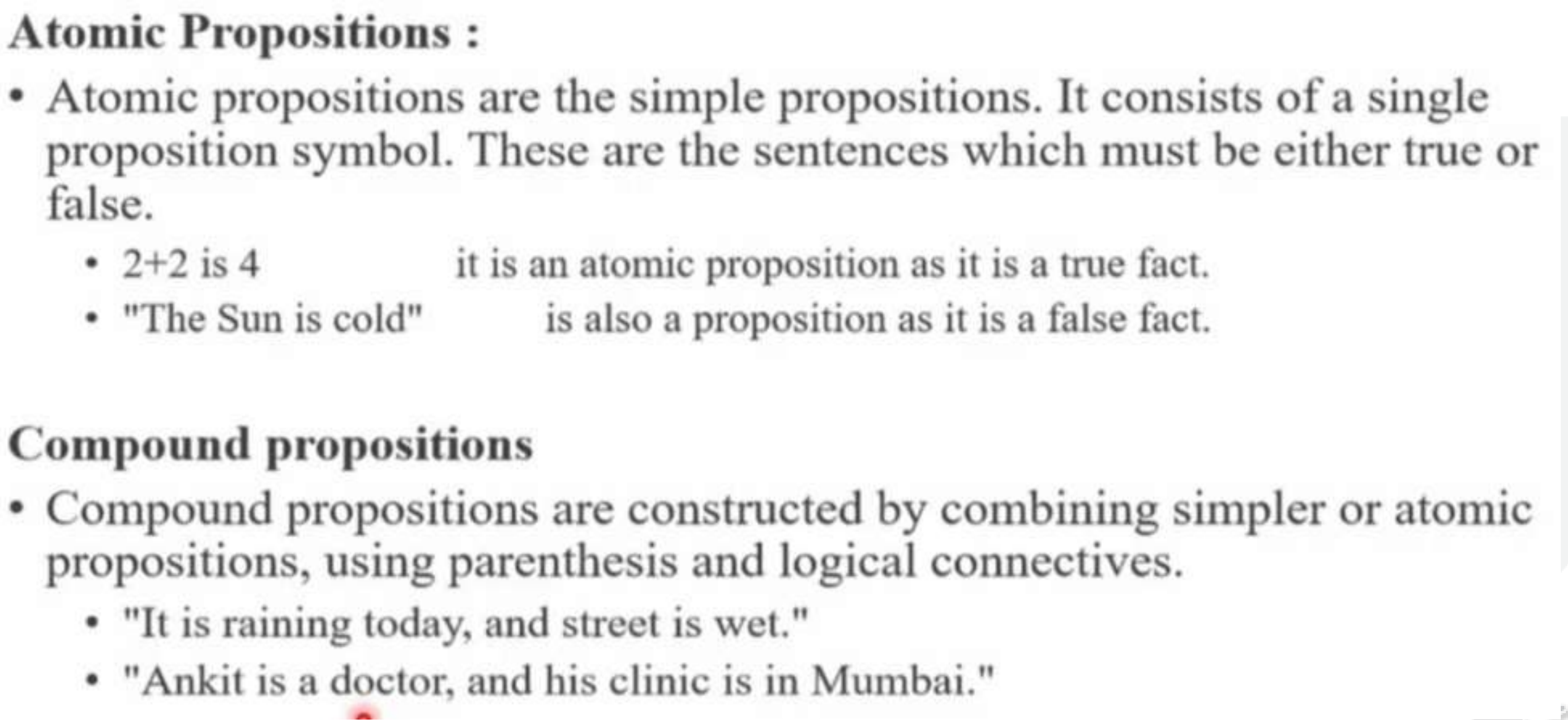
Propositional logic deals with **propositions**, which are declarative statements that are either definitively **true** or definitively **false**, but not both. They are the basic building blocks of knowledge.



1. Delhi is the capital of USA ✓
2. How are you doing ✗
3. 5<= 11 ✓

DIGITAL LEARNING CONTENT

ey Components for Representation



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 $\neg B$

- **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∧ Q**.

- **Disjunction**
  - **Example: "Ritika is a doctor or Engineer“**
  - P= Ritika is Doctor.
  - Q= Ritika is Doctor,
  - so we can write it as **P ∨ Q**.

- **Negation**
  - **Geeta is not a engineer**
  - A sentence such as ¬ 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 $P \Leftrightarrow Q$.

Reasoning Pattern/Inference in Propositional Logic

Reasoninig/Inference is the process of deriving new, valid conclusions from existing knowledge (a set of propositions or a knowledge base). It allows an AI system to "reason" and extend its understanding of the world.

Cont..

Inference Rules /reasoning patterns: These are patterns of logical reasoning.

•**Modus Ponens:**
If you know P and you know P → Q, then you can infer Q.
Example: "It is raining" (P), "If it is raining, then the ground is wet" (P → Q). Infer: "The ground is wet" (Q).

•**Modus Tollens:**
If you know P → Q and you know ¬Q, then you can infer ¬P.
Example: "If it is raining, then the ground is wet" (P → Q), "The ground is NOT wet" (¬Q). Infer: "It is NOT raining" (¬P).

•**And-Introduction:**
If you know P and you know Q, then you can infer P ∧ Q.

Cont..

•**And-Elimination:**
If you know P ∧ Q, then you can infer P (or Q).

•**Hypothetical Syllogism:**
If we know two conditional statements where the consequent of the first is the antecedent of the second (P → Q and Q → R), then we can infer a new conditional statement (P → R)

Cont..

•**Resolution:** A powerful inference rule that forms the basis of many automated theorem provers. It works by converting sentences into Conjunctive Normal Form (CNF) and then applying a specific rule to derive new clauses (disjunctions of literals). It's a complete inference procedure for propositional logic (and first-order logic too).
If you have a clause A ∨ B and another clause ¬B ∨ C, you can resolve them to get A ∨ C.
Often used in **Resolution Refutation:** To prove KB ->α, you show that KB ∧ ¬ α is unsatisfiable (leads to a contradiction, the empty clause []).

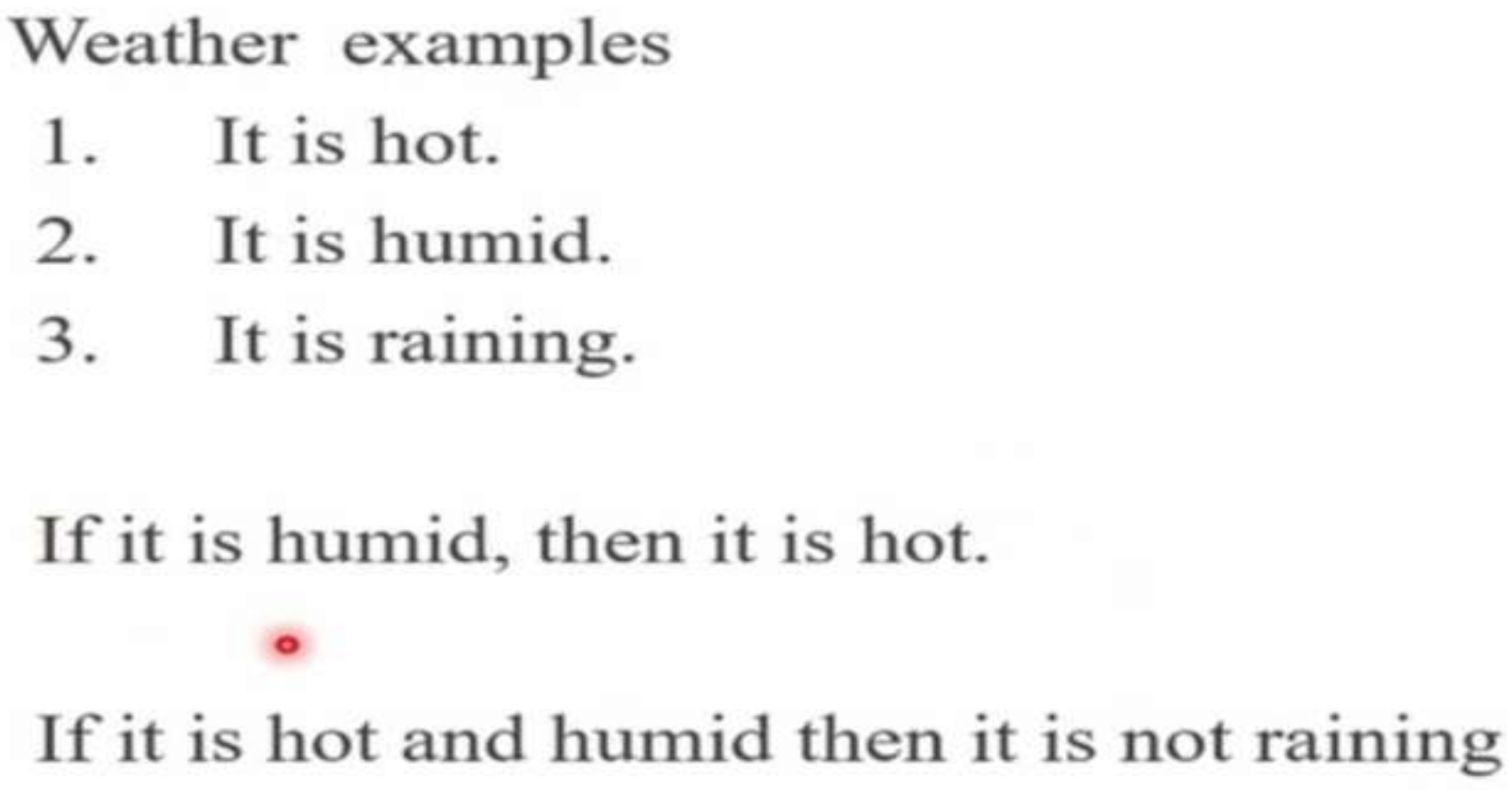
Parul® University NAAC A++ ACCREDITED UNIVERSITY

DIGITAL LEARNING CONTENT

Resolution in Propositional Logic

From Unit 2 part2

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.

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

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

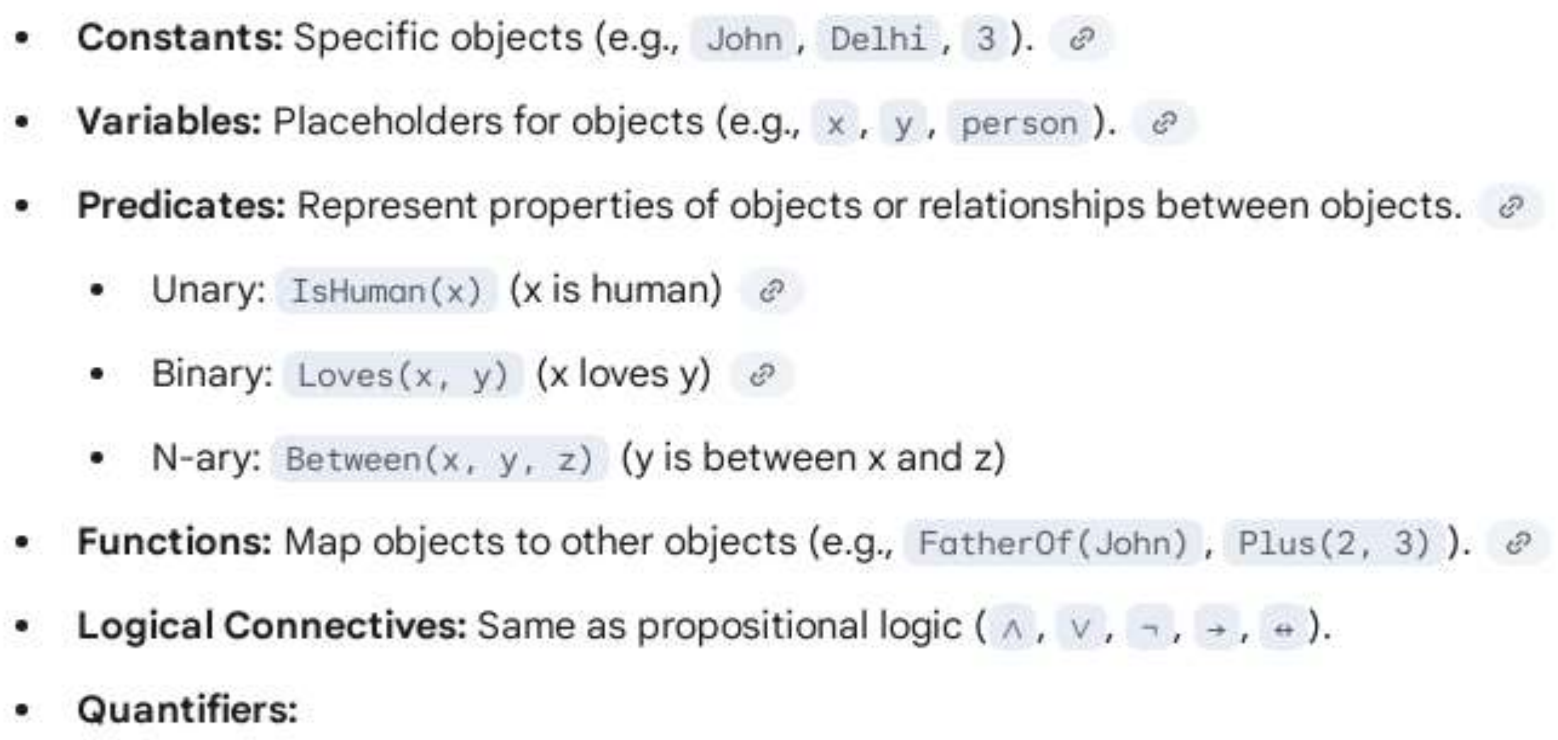
  E.g., $(\forall x)\ \text{dolphin}(x) \rightarrow \text{mammal}(x)$

- Existential quantification

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

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

Knowledge Representation in Predicate Logic(First Order Logic)



- **Constants:** Specific objects (e.g., `John`, `Delhi`, `3`).
- **Variables:** Placeholders for objects (e.g., `x`, `y`, `person`).
- **Predicates:** Represent properties of objects or relationships between objects.
  - Unary: `IsHuman(x)` (x is human)
  - Binary: `Loves(x, y)` (x loves y)
  - N-ary: `Between(x, y, z)` (y is between x and z)
- **Functions:** Map objects to other objects (e.g., `FatherOf(John)`, `Plus(2, 3)`).
- **Logical Connectives:** Same as propositional logic (`∧`, `∨`, `¬`, `→`, `↔`).
- **Quantifiers:**

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 ≠ 10

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

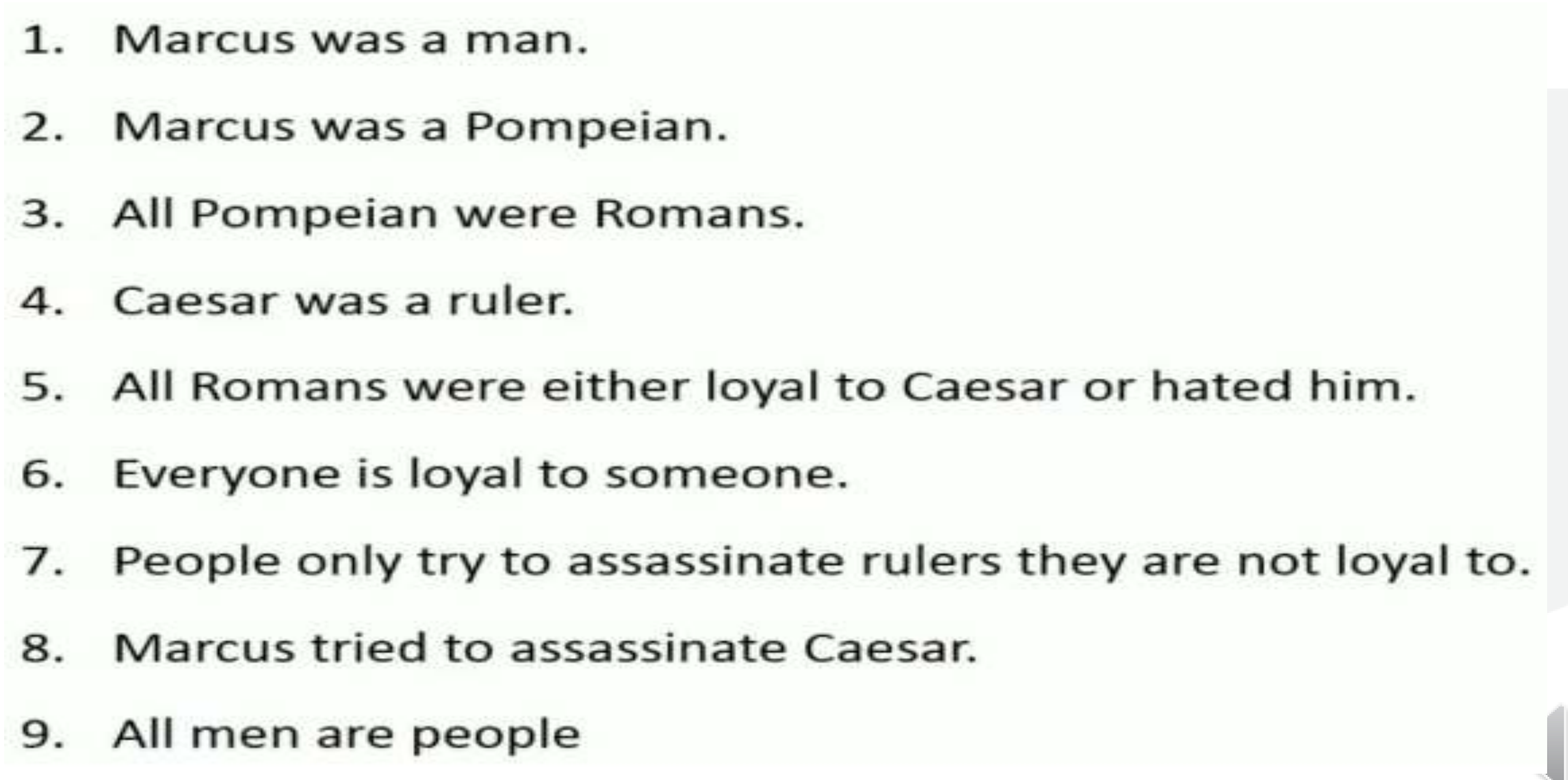
The statement ∀x ∃y P(x,y) is true because for any integer x, we can always find an integer y such that their sum is 0 (y would be -x).

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

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

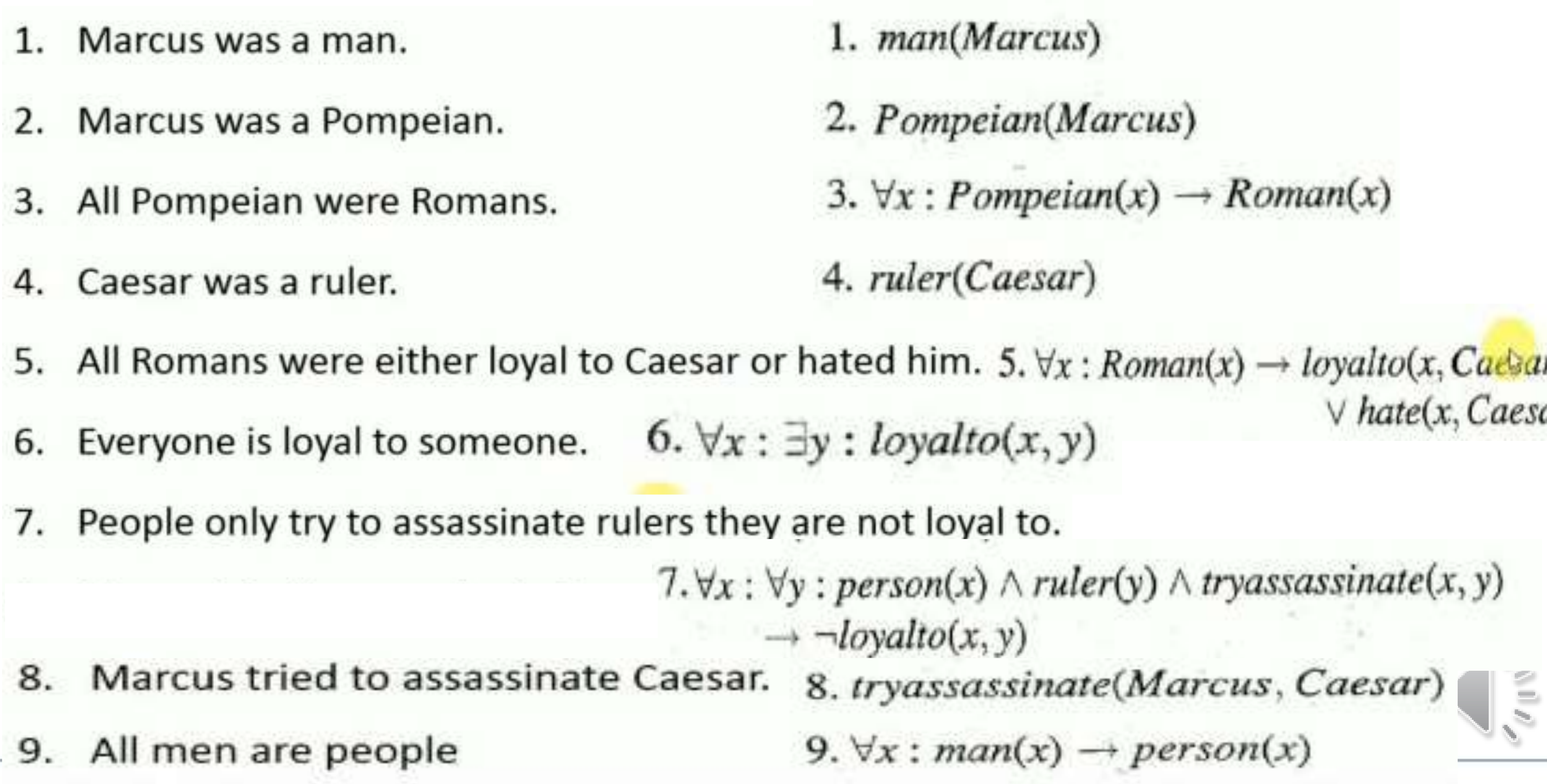
The statement ∃x Q(x) is true because there exist prime numbers (e.g., 2, 3, 5, 7, etc.).

Representing simple facts 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

Representing simple facts 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. $\forall x : man(x) \rightarrow person(x)$

Representing instance and ISA relationship

Representing "instance" and "ISA" (Is-A) relationships in predicate logic is fundamental for building knowledge bases

1. Instance Relationship (Individual to Class)

This relationship connects a specific individual to a general class or category it belongs to.

In Natural Language:

"Socrates is a man."

"My car is a red vehicle."

"The Eiffel Tower is a monument.“

Cont..

In Predicate Logic: We use a **unary predicate** to represent the class, and the individual is the argument of that predicate.

ClassName(individual_name)

Examples:

•"Socrates is a man."

Man(Socrates)

•"My car is a red vehicle." (Assuming 'RedVehicle' is a class)

RedVehicle(MyCar)

Alternatively, using multiple predicates: Vehicle(MyCar) and Color(MyCar, Red)

•"The Eiffel Tower is a monument."

Monument(EiffelTower)

Cont..

2. ISA Relationship (Class to Class / Subclass to Superclass)
This relationship expresses that one class is a subcategory of another more general class. It implies inheritance of properties. If something is a member of the subclass, it is also a member of the superclass.

In Natural Language:
"All men are mammals."
"A dog is a canine."
"Vehicles are modes of transport."

Cont..

In Predicate Logic: We typically use **universal quantification (∀)** and **implication (→)** to express that if something is a member of the subclass, then it is also a member of the superclass.
$\forall x(\mathrm{SubClass}(x) \rightarrow \mathrm{SuperClass}(x))$

Examples:
•"All men are mammals."
$\forall x(\mathrm{Man}(x) \rightarrow \mathrm{Mammal}(x))$
•This reads: "For all things x, if x is a Man, then x is a Mammal."
"A dog is a canine."
$\forall x(\mathrm{Dog}(x) \rightarrow \mathrm{Canine}(x))$
•"Vehicles are modes of transport."
$\forall x(\mathrm{Vehicle}(x) \rightarrow \mathrm{ModeOfTransport}(x))$

Cont..

❑Instance: This is a direct assertion that a specific entity belongs to a particular category.

❑ISA:This represents a hierarchical relationship between categories, where the properties of the superclass are inherited by the subclass.

Combining Instance and ISA Relationships for Reasoning

These two types of relationships work together to enable logical inference.

Example Scenario:

Given the following knowledge base:

1. Man(Socrates) (Socrates is a man)
2. ∀x(Man(x)→Mammal(x)) (All men are mammals)
3. ∀x(Mammal(x)→WarmBlooded(x)) (All mammals are warm- blooded)

Cont..

Can we infer that Socrates is Warm-Blooded?

Yes, using rules of inference:
From (1) Man(Socrates) and (2) ∀x(Man(x)→Mammal(x)), by Universal Instantiation we can infer:
Mammal(Socrates)   (Socrates is a mammal)

From Mammal(Socrates) and (3) ∀x(Mammal(x)→WarmBlooded(x)), by Universal Instantiation we can infer:
WarmBlooded(Socrates)   (Socrates is warm-blooded)

Computable Functions and Predicates

Computable Functions

A function is **computable** if there exists an **algorithm** (a finite, unambiguous set of step-by-step instructions) that can, for any valid input in the function's domain, produce the correct output in a finite amount of time.

•**Turing Machine Connection:** The most widely accepted formal definition states that a function is computable if and only if it can be computed by a **Turing machine**.

Examples:

•Addition(x, y) = x + y is computable.

•Factorial(n) = n! is computable.

•FindShortestPath(graph, start, end) (like Dijkstra's algorithm) is computable.

•SortList(list) is computable.

Cont..

Computable Predicates

A predicate is a statement that can be either **true** or **false** depending on its arguments. A predicate is **computable** (or **decidable**) if there exists an algorithm that can, for any valid input, determine whether the predicate is true or false in a finite amount of time.

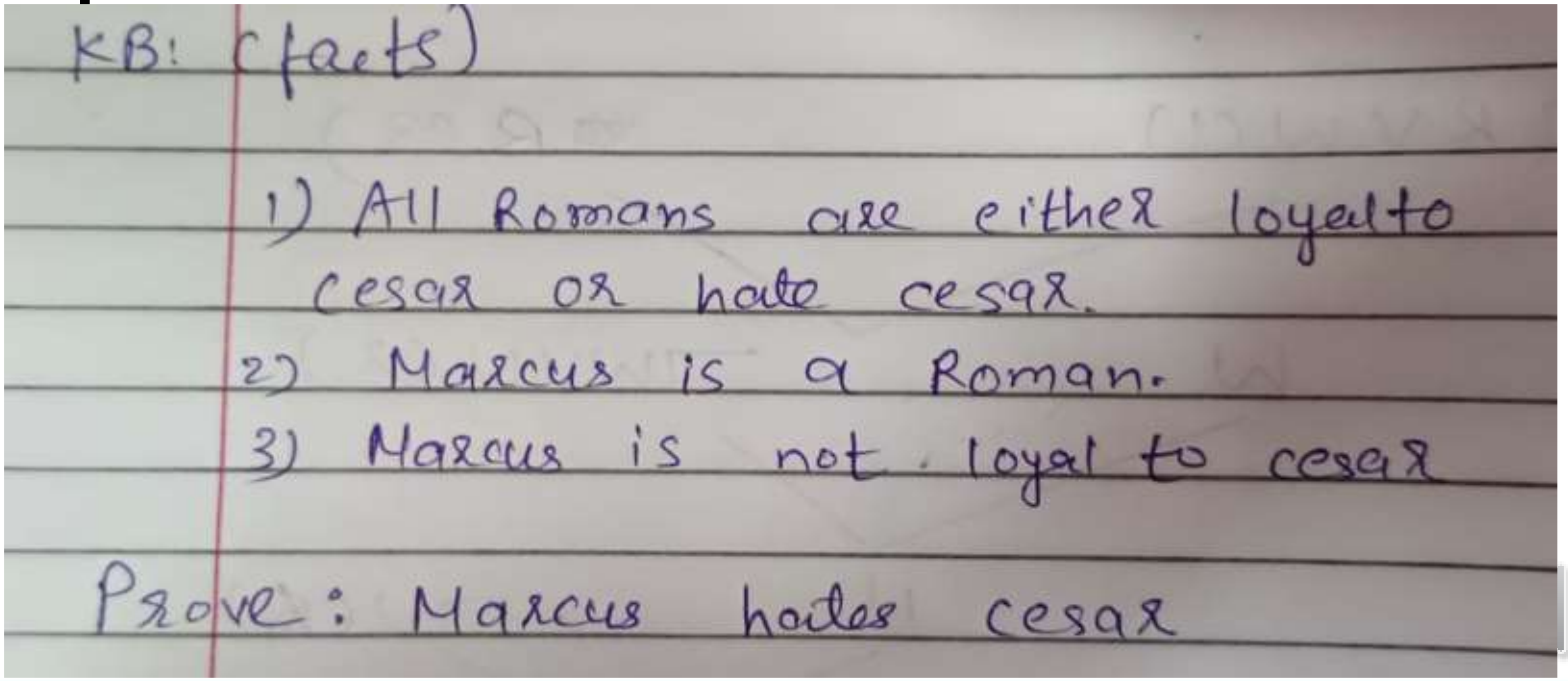
Boolean Functions: Computable predicates are essentially computable functions that always return a Boolean (True/False) value.

Decidability: The term "decidable" is often used interchangeably with "computable" for predicates. A problem whose answer is a "yes" or "no" (or True/False) is decidable if there's an algorithm to solve it.

DIGITAL LEARNING CONTENT

Resolution in predicate logic

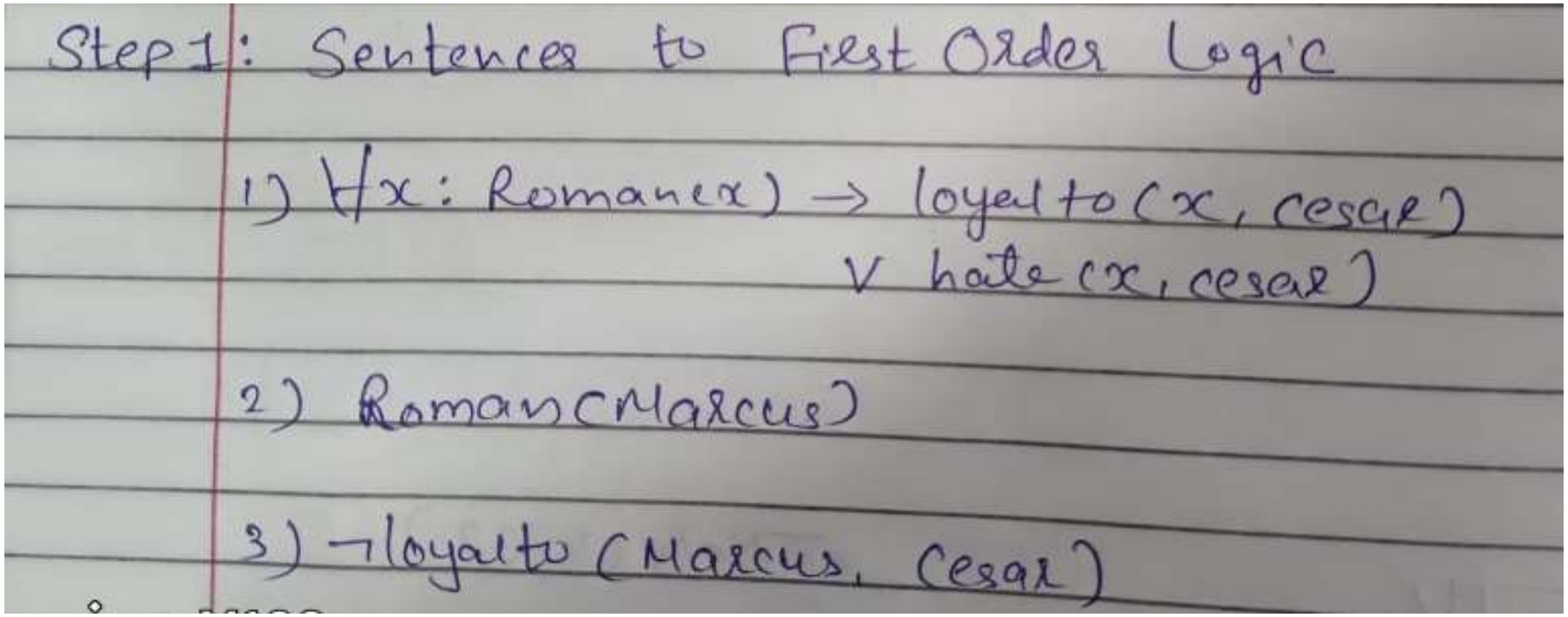
Theory From Unit 2 Part 2
Example:


KB: (facts)

1) All Romans are either loyalto cesar or hate cesar.
2) Marcus is a Roman.
3) Marcus is not loyal to cesar

Prove : Marcus hates cesar

Cont..

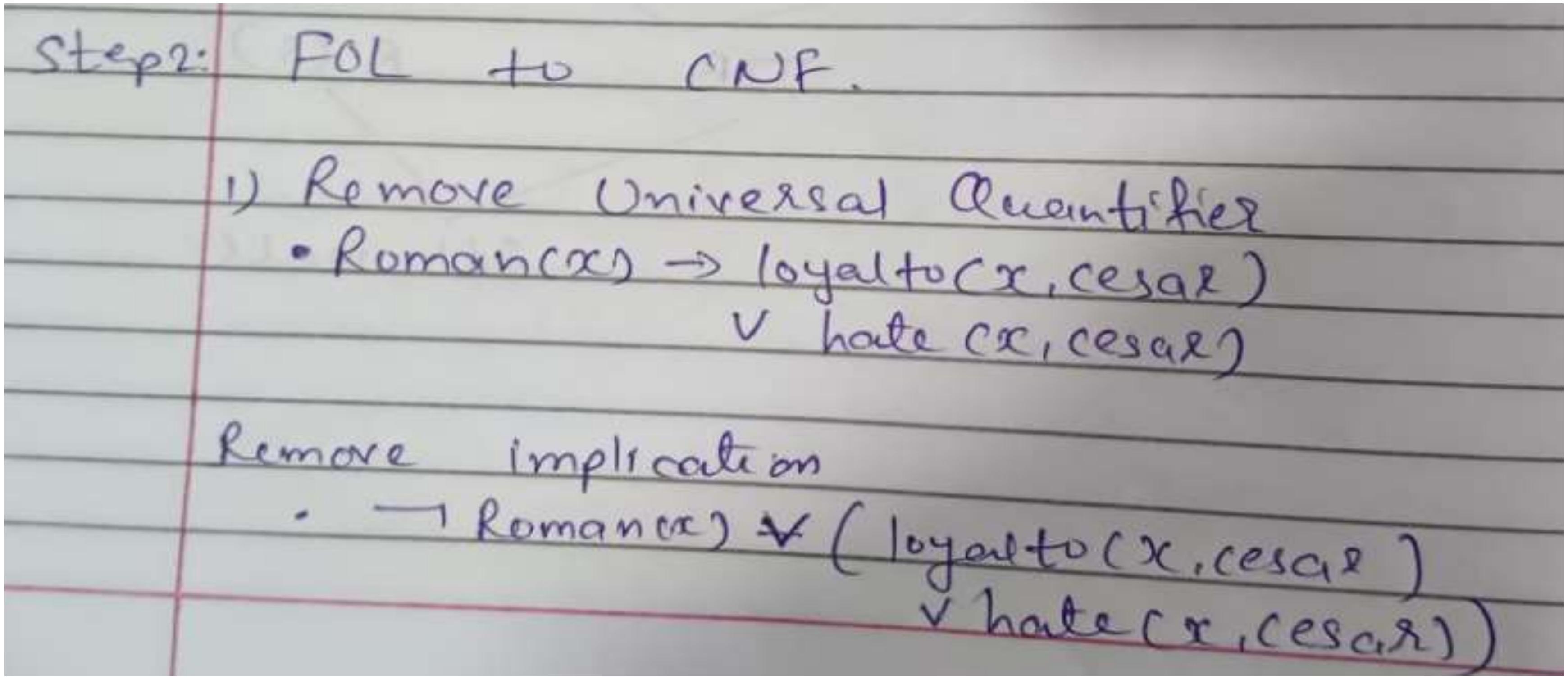

Step 1: Sentences to First Order Logic

1) ∀x: Roman(x) → loyalto(x, cesar) ∨ hate(x, cesar)

2) Roman(Marcus)

3) ¬loyalto(Marcus, Cesar)

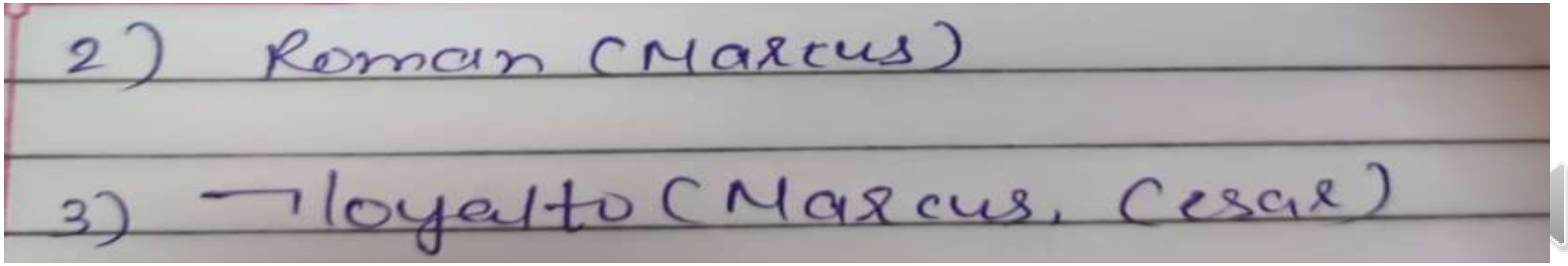
Prove: Hate(Marcus,Cesar)



Step2: FOL to CNF.

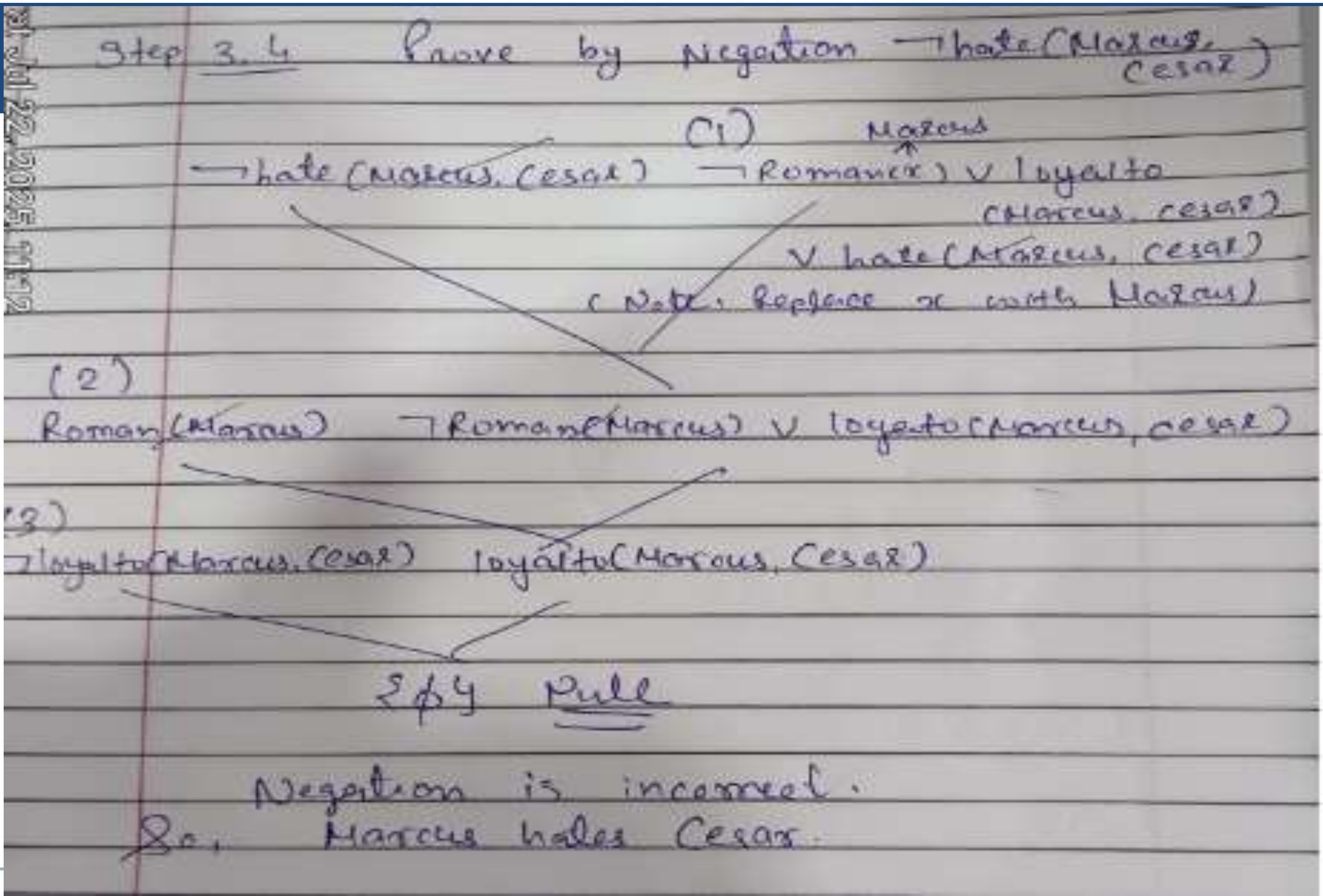
1) Remove Universal Quantifier
- $Roman(x) \rightarrow loyalto(x, cesar) \lor hate(x, cesar)$

Remove implication
- $\neg Roman(x) \lor (loyalto(x, cesar) \lor hate(x, cesar))$



2) Roman(Marcus)

3) $\neg loyalto(Marcus, Cesar)$



Step 3.4 Prove by Negation $\neg$hate(Marcus, Cesar)

(1)

$\neg$hate(Marcus, Cesar) $\quad$ $\neg$Roman($\overset{\text{Marcus}}{x}$) $\vee$ loyalto(Marcus, cesar) $\vee$ hate(Marcus, cesar)

(Note: Replace x with Marcus)

(2)

Roman(Marcus) $\quad$ $\neg$Roman(Marcus) $\vee$ loyalto(Marcus, cesar)

(3)

$\neg$loyalto(Marcus, Cesar) $\quad$ loyalto(Marcus, Cesar)

{ϕ} Null

Negation is incorrect.

So, Marcus hates Cesar.

Inference Rules for FOL/Predicate Logic

1.Universal Instantiation (UI) / Universal Elimination:

Pattern: If a statement is true for *all* objects (universally quantified), then it's true for any *specific* object.

Example:
KB: ∀x (Human(x)→Mortal(x)) (All humans are mortal)
KB: Human(Socrates)   (Socrates is human)
Apply UI :
Human(Socrates) → Mortal(Socrates)

Cont..

2. Existential Instantiation (EI) / Existential Elimination:

Pattern: If a statement is true for *at least one* object (existentially quantified), then we can introduce a new, unique constant (a **Skolem constant**) to represent that object.

Example:
KB: $\exists x\,(King(x) \wedge Greedy(x))$ (There exists a king who is greedy)
Apply EI: new constant k=King1: $King(King1) \wedge Greedy(King1)$

Cont..

3.Universal Generalization (UG) / Universal Introduction:

Pattern: If a property P(c) holds true for an *arbitrary* (arbitrarily chosen, not special) element c, then we can conclude that ∀xP(x) is true.

Example: c=2k
P(c) : is divisible by 2
P(x)= even number
∀x P(x) P(c) is true.

Cont..

4. Existential Introduction (EI) / Existential Generalization:

Pattern: If a property P(c) holds true for a *specific* element c, then we can infer that there exists at least one object for which that property holds.

∃xP(x)P(c)

Example:

Fact: Loves(John, Mary)

Inference: ∃x Loves(x,Mary) (Someone loves Mary)

Reasoning Strategies or pattern in Predicate Logic/FOL

1.Forward Chaining (Data-Driven):
Starts with known facts and applies inference rules (like Generalized Modus Ponens) to derive new facts until the query is proven or no new facts can be derived.

Analogy: "What else can I know from what I already know?"

Applications: Situations where new data arrives frequently and you want to deduce all possible consequences (e.g., monitoring systems, some types of expert systems).

Cont..

2. Backward Chaining (Goal-Driven):
Starts with the query (goal) and works backward, looking for rules that could prove the goal. For each such rule, it tries to prove its premises as sub-goals.

Analogy: "How can I prove this goal? What must be true for this goal to be true?"

Applications: Question-answering systems, diagnostic systems, logic programming (Prolog primarily uses backward chaining)

Parul® University NAAC A++ ACCREDITED UNIVERSITY

Inference Rules for FOL/Predicate Logic