Artificial Intelligence-BSCE-306L

Module 4:

Logic and Reasoning

Dr . Saurabh Agrawal

Faculty Id: 20165

School of Computer Science and Engineering

VIT, Vellore-632014

Tamil Nadu, India

Outline

- ❑Introduction to Logic and Reasoning
- ❑Propositional Logic
- ❑First Order Logic
- ❑Inference in First Order Logic
- ❑Unification
- ❑Forward Chaining
- ❑Backward Chaining
- ❑Resolution

Introduction to Logic

- **Logic:"the reasoning conducted or assessed according to strict principles and validity"**.
- Logic can be defined as the proof or validation behind any reason provided.
- It is simply the ‘dialectics behind reasoning’.
- It was important to include logic in Artificial Intelligence because we want our agent (system) to think and act humanly, and for doing so, it should be capable of taking any decision based on the current situation.
- If we talk about normal human behavior, then a decision is made by choosing an option from the various available options.
- There are reasons behind selecting or rejecting an option.
- So, our artificial agent should also work in this manner.
- While taking any decision, the agent must provide specific reasons based on which the decision was taken.
- And this reasoning can be done by the agent only if the agent has the capability of understanding the logic.

Introduction to Logic

❑In artificial Intelligence, we deal with two **types of logics**:

1. **Deductive logic**

- In deductive logic, the complete evidence is provided about the truth of the conclusion made.
- Here, the agent uses specific and accurate premises that lead to a specific conclusion.
- An example of this logic can be seen in an expert system designed to suggest medicines to the patient.
- The agent gives the complete proof about the medicines suggested by it, like the particular medicines are suggested to a person because the person has so and so symptoms.

2. **Inductive logic**

- In Inductive logic, the reasoning is done through a ‘bottom-up’ approach.
- What this means is that the agent here takes specific information and then generalizes it for the sake of complete understanding.
- An example of this can be seen in the natural language processing by an agent in which it sums up the words according to their category, i.e. verb, noun article, etc., and then infers the meaning of that sentence.

Introduction to Reasoning

- The reasoning is the mental process of deriving logical conclusion and making predictions from available knowledge, facts, and beliefs.
- Or we can say, "**Reasoning is a way to infer facts from existing data**." It is a general process of thinking rationally, to find valid conclusions.
- In artificial intelligence, the reasoning is essential so that the machine can also think rationally as a human brain, and can perform like a human.
- In artificial intelligence, reasoning can be divided into the following categories:
  1. **Deductive reasoning**
  2. **Inductive reasoning**
  3. **Abductive reasoning**
  4. **Common Sense Reasoning**
  5. **Monotonic Reasoning**
  6. **Non-monotonic Reasoning**

Introduction to Reasoning

1. **Deductive reasoning:** Deductive reasoning is deducing new information from logically related known information.

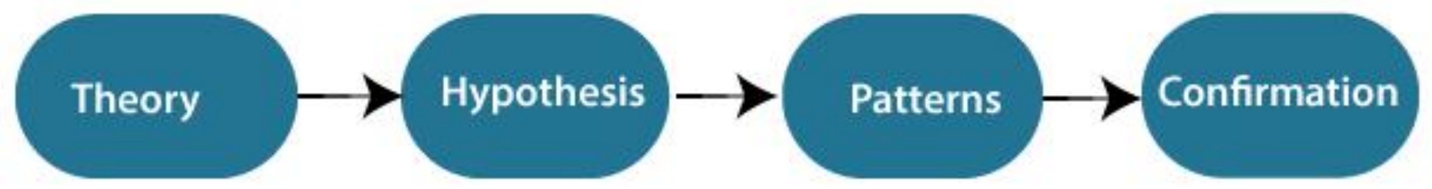
- It is the form of valid reasoning, which means the argument's conclusion must be true when the premises are true.
- Deductive reasoning is a type of propositional logic in AI, and it requires various rules and facts.
- It is sometimes referred to as top-down reasoning, and contradictory to inductive reasoning.
- In deductive reasoning, the truth of the premises guarantees the truth of the conclusion.
- Deductive reasoning mostly starts from the general premises to the specific conclusion, which can be explained as below example.

Example:

Premise-1: All the human eats veggies

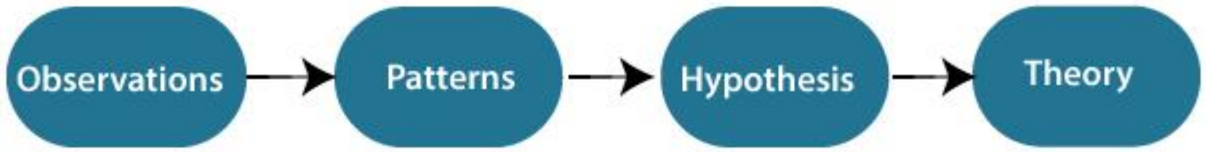
Premise-2: Suresh is human.

Conclusion: Suresh eats veggies.



2. Inductive reasoning:

- Inductive reasoning is a form of reasoning to arrive at a conclusion using limited sets of facts by the process of generalization.
- It starts with the series of specific facts or data and reaches to a general statement or conclusion.
- Inductive reasoning is a type of propositional logic, which is also known as cause-effect reasoning or bottom-up reasoning.
- In inductive reasoning, we use historical data or various premises to generate a generic rule, for which premises support the conclusion.
- In inductive reasoning, premises provide probable supports to the conclusion, so the truth of premises does not guarantee the truth of the conclusion.



Example:

Premise: All of the pigeons we have seen in the zoo are white.

Conclusion: Therefore, we can expect all the pigeons to be white.

Introduction to Reasoning

3. Abductive reasoning:

- Abductive reasoning is a form of logical reasoning which starts with single or multiple observations then seeks to find the most likely explanation or conclusion for the observation.
- Abductive reasoning is an extension of deductive reasoning, but in abductive reasoning, the premises do not guarantee the conclusion.

Example:

Implication: Cricket ground is wet if it is raining

Axiom: Cricket ground is wet.

Conclusion It is raining.

4. Common Sense Reasoning:

- Common sense reasoning is an informal form of reasoning, which can be gained through experiences.
- Common Sense reasoning simulates the human ability to make presumptions about events which occurs on every day.
- It relies on good judgment rather than exact logic and operates on heuristic knowledge and heuristic rules.

Example:

One person can be at one place at a time.

If I put my hand in a fire, then it will burn.

The above two statements are the examples of common sense reasoning which a human mind can easily understand and assume.

5. Monotonic Reasoning

- In monotonic reasoning, once the conclusion is taken, then it will remain the same even if we add some other information to existing information in our knowledge base.
- In monotonic reasoning, adding knowledge does not decrease the set of prepositions that can be derived.
- To solve monotonic problems, we can derive the valid conclusion from the available facts only, and it will not be affected by new facts.
- Monotonic reasoning is not useful for the real-time systems, as in real time, facts get changed, so we cannot use monotonic reasoning.
- Monotonic reasoning is used in conventional reasoning systems, and a logic-based system is monotonic.

Example:

Earth revolves around the Sun.

It is a true fact, and it cannot be changed even if we add another sentence in knowledge base like, "The moon revolves around the earth" Or "Earth is not round," etc.

6. Non-monotonic Reasoning:

- In Non-monotonic reasoning, some conclusions may be invalidated if we add some more information to our knowledge base.
- Logic will be said as non-monotonic if some conclusions can be invalidated by adding more knowledge into our knowledge base.
- Non-monotonic reasoning deals with incomplete and uncertain models.
- "Human perceptions for various things in daily life, "is a general example of non-monotonic reasoning.

Example: Let suppose the knowledge base contains the following knowledge:

Birds can fly

Penguins cannot fly

Pitty is a bird

So from the above sentences, we can conclude that **Pitty can fly**.

However, if we add one another sentence into knowledge base "**Pitty is a penguin**", which concludes "**Pitty cannot fly**", so it invalidates the above conclusion.

Propositional Logic

- Propositional logic (PL) is the simplest form of logic where all the statements are made by propositions.
- A proposition is a declarative statement which is either true or false.
- It is a technique of knowledge representation in logical and mathematical form.
- **Example:**

a) It is Sunday.

b) The Sun rises from West (False proposition)

c) 3+3= 7(False proposition)

d) 5 is a prime number.

Propositional Logic

❑**Following are some basic facts about propositional logic:**

▪Propositional logic is also called Boolean logic as it works on 0 and 1.

▪In propositional logic, we use symbolic variables to represent the logic, and we can use any symbol for a representing a proposition, such A, B, C, P, Q, R, etc.

▪Propositions can be either true or false, but it cannot be both.

▪Propositional logic consists of an object, relations or function, and **logical connectives**.

▪These connectives are also called logical operators.

▪The propositions and connectives are the basic elements of the propositional logic.

▪Connectives can be said as a logical operator which connects two sentences.

▪A proposition formula which is always true is called **tautology**, and it is also called a valid sentence.

▪A proposition formula which is always false is called **Contradiction**.

▪Statements which are questions, commands, or opinions are not propositions such as "**Where is Anthony**", "**How are you**", "**What is your name**", are not propositions.

Propositional Logic

❑**Syntax of propositional logic:** The syntax of propositional logic defines the allowable sentences for the knowledge representation.

❑**There are two types of Propositions:**

1. Atomic Proposition: Atomic propositions are the simple propositions. It consists of a single proposition symbol. These are the sentences which must be either true or false.

Example:

a) 2+2 is 4, it is an atomic proposition as it is a **true** fact.

b) "The Sun is cold" is also a proposition as it is a **false** fact.

2. Compound propositions: Compound propositions are constructed by combining simpler or atomic propositions, using parenthesis and logical connectives.

Example:

a) "It is raining today, and street is wet."

b) "Ankit is a doctor, and his clinic is in Mumbai."

Propositional Logic

❑**Logical Connectives:** 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:

1. Negation: A sentence such as $\neg$ P is called negation of P. A literal can be either Positive literal or negative literal.

2. Conjunction: A sentence which has $\wedge$ connective such as, $P \wedge Q$ is called a conjunction.

Example: Rohan is intelligent and hardworking. It can be written as,

P= Rohan is intelligent,

Q= Rohan is hardworking. $\rightarrow P \wedge Q$.

Propositional Logic

❑Logical Connectives:

3. Disjunction: A sentence which has ∨ connective, such as $P \lor Q$. is called disjunction, where P and Q are the propositions.

Example: "Ritika is a doctor or Engineer",

Here P= Ritika is Doctor. Q= Ritika is Doctor, so we can write it as $P \lor Q$.

4. Implication: A sentence such as $P \rightarrow Q$, is called an implication. Implications are also known as if-then rules. It can be represented as

If it is raining, then the street is wet.

Let P= It is raining, and Q= Street is wet, so it is represented as $P \rightarrow Q$

5. Biconditional: A sentence such as $P \Leftrightarrow Q$ is a Biconditional sentence, example If I am breathing, then I am alive

P= I am breathing, Q= I am alive, it can be represented as $P \Leftrightarrow Q$.

Propositional Logic

❑**Logical Connectives:** Following is the summarized table for Propositional Logic Connectives:

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 $\neg B$

Propositional Logic

- **Truth Table:** In propositional logic, we need to know the truth values of propositions in all possible scenarios. We can combine all the possible combination with logical connectives, and the representation of these combinations in a tabular format is called **Truth table**. Following are the truth table for all logical connectives:

For Negation:

P	¬ P
True	**False**
False	**True**

For Conjunction:

P	Q	P∧ Q
True	True	**True**
True	False	**False**
False	True	**False**
False	False	**False**

❑Truth Table:

For disjunction:

P	Q	P V Q.
True	True	**True**
False	True	**True**
True	False	**True**
False	False	**False**

For Implication:

P	Q	P→ Q
True	True	**True**
True	False	**False**
False	True	**True**
False	False	**True**

For Biconditional:

P	Q	P⇔ Q
True	True	**True**
True	False	**False**
False	True	**False**
False	False	**True**

Propositional Logic

❑**Truth table with three propositions:** We can build a proposition composing three propositions P, Q, and R. This truth table is made-up of 8n Tuples as we have taken three proposition symbols.

P	Q	R	¬R	Pv Q	PvQ→¬R
True	True	True	False	True	False
True	True	False	True	True	True
True	False	True	False	True	False
True	False	False	True	True	True
False	True	True	False	True	False
False	True	False	True	True	True
False	False	True	False	False	True
False	False	False	True	False	True

Propositional Logic

❑**Precedence of connectives:** Just like arithmetic operators, there is a precedence order for propositional connectors or logical operators. This order should be followed while evaluating a propositional problem. Following is the list of the precedence order for operators:

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

First Order Logic (FOL)

- First-order logic is another way of knowledge representation in artificial intelligence. It is an extension to propositional logic.
- FOL is sufficiently expressive to represent the natural language statements in a concise way.
- First-order logic is also known as **Predicate logic or First-order predicate logic**.
- First-order logic is a powerful language that develops information about the objects in a more easy way and can also express the relationship between those objects.
- First-order logic (like natural language) does not only assume that the world contains facts like propositional logic but also assumes the following things in the world:
  - **Objects:** A, B, people, numbers, colors, wars, theories, squares, pits, wumpus, ...
  - **Relations:** It can be unary relation such as: red, round, is adjacent, or n-any relation such as: the sister of, brother of, has color, comes between
  - **Function:** Father of, best friend, third inning of, end of, ...

First Order Logic (FOL)

- As a natural language, first-order logic also has two main parts:

  1. **Syntax**
  2. **Semantics**

First Order Logic (FOL)

→Syntax of First-Order logic:

- The syntax of FOL determines which collection of symbols is a logical expression in first-order logic.
- The basic syntactic elements of first-order logic are symbols.
- We write statements in short-hand notation in FOL.
- Following are the basic elements of FOL syntax:

Constant	**1, 2, A, John, Mumbai, cat,....**
Variables	**x, y, z, a, b,....**
Predicates	**Brother, Father, >,....**
Function	**sqrt, LeftLegOf, ...**
Connectives	**∧, ∨, ¬, ⇒, ⇔**
Equality	**==**
Quantifier	**∀, ∃**

First Order Logic (FOL)

→Syntax of First-Order logic:

- ❑**Atomic sentences:** are the most basic sentences of first-order logic.
- ❑These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms.
- ❑We can represent atomic sentences as **Predicate (term1, term2, ..., term n)**.

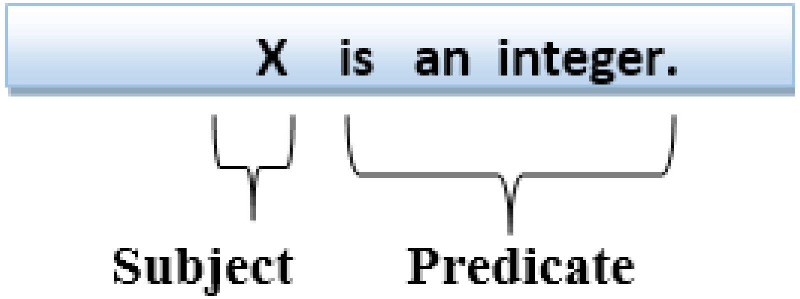
Example: Ravi and Ajay are brothers: => Brothers(Ravi, Ajay).

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

First Order Logic (FOL)

→Syntax of First-Order logic:

- **Complex Sentences:** are made by combining atomic sentences using connectives.
- First-order logic statements can be divided into two parts:
    - Subject: Subject is the main part of the statement.
    - Predicate: A predicate can be defined as a relation, which binds two atoms together in a statement.
- Consider the statement: "x is an integer.", it consists of two parts, the first part x is the subject of the statement and second part "is an integer," is known as a predicate.



First Order Logic (FOL)

→Quantifiers in First-order logic:

- ❑A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- ❑These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression.
- ❑There are two types of quantifier:
    1. **Universal Quantifier, (for all, everyone, everything)**
    2. **Existential quantifier, (for some, at least one).**

First Order Logic (FOL)

→Quantifiers in First-order logic:

1. Universal Quantifier, (for all, everyone, everything)

❑Universal Quantifier: is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.

❑The Universal quantifier is represented by a symbol ∀, which resembles an inverted A.

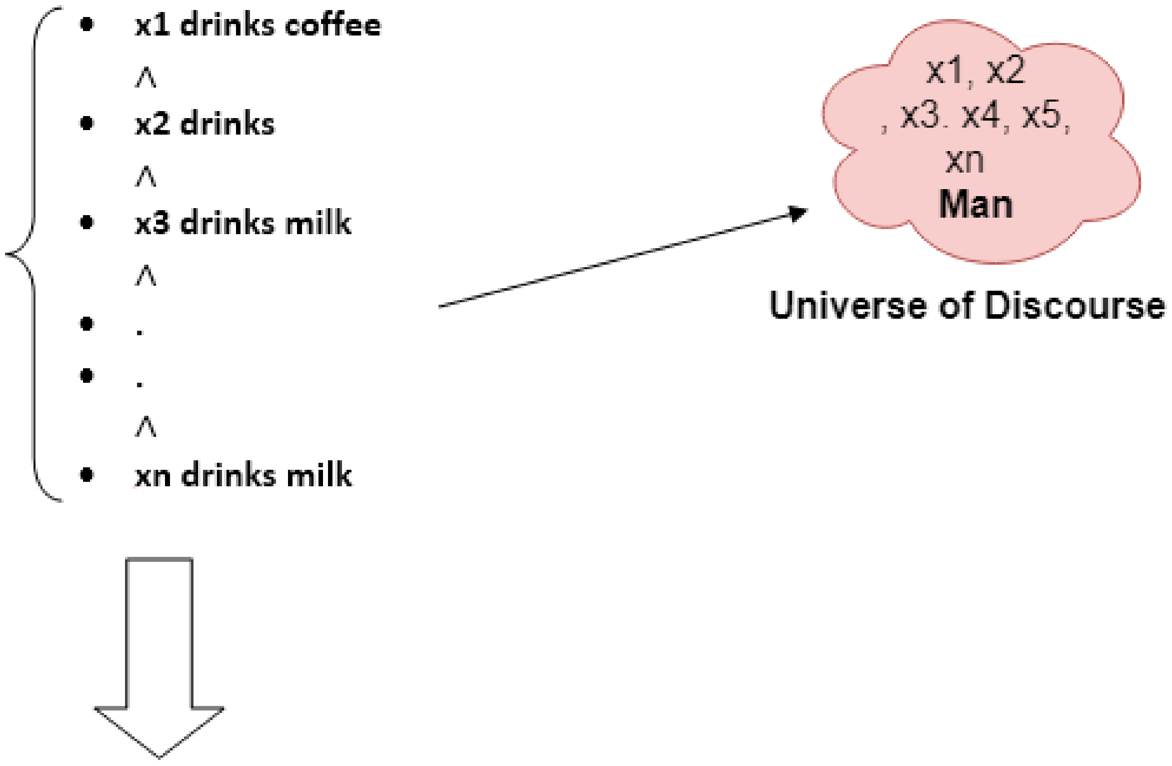
❑If x is a variable, then ∀x is read as:

- **For all x**
- **For each x**
- **For every x.**

→Quantifiers in First-order logic:

1. **Universal Quantifier, (for all, everyone, everything)**

- ❑Example: **All man drink coffee.**
- ❑Let a variable x which refers to a cat so all x can be represented in UOD as below:
- ❑**$\forall$x man(x) → drink (x, coffee).**
- ❑It will be read as: There are all x where x is a man who drink coffee.



So in shorthand notation, we can write it as :

→Quantifiers in First-order logic:

2. Existential quantifier, (for some, at least one).

❑Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.

❑It is denoted by the logical operator ∃, which resembles as inverted E. When it is used with a predicate variable then it is called as an existential quantifier.

❑If x is a variable, then existential quantifier will be ∃x or ∃(x). And it will be read as:

- **There exists a 'x.'**
- **For some 'x.'**
- **For at least one 'x.'**

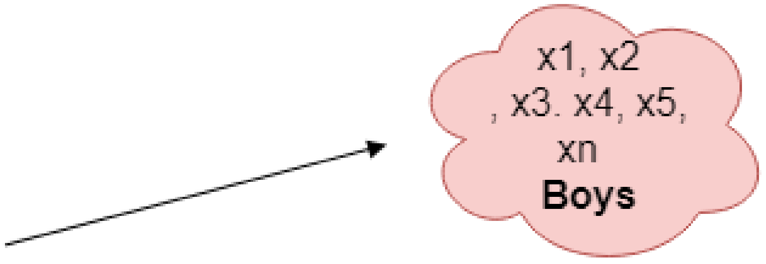
→Quantifiers in First-order logic:

2. Existential quantifier, (for some, at least one).

- ❑Example: **Some boys are intelligent.**
- ❑It will be read as: There are some x where x is a boy who is intelligent.

Some boys are intelligent.

- **x1 is intelligent**
  V
- **x2 is intelligentV**
- **x3 is intelligent**
  V
- .
- .
  V
- **xn is intelligent**



Universe of Discourse

So in short-hand notation, we can write it as:

∃x: boys(x) ∧ intelligent(x)

First Order Logic (FOL)

→Quantifiers in First-order logic:

❑Points to remember:

- The main connective for universal quantifier ∀ is implication →.
- The main connective for existential quantifier ∃ is and ∧.

❑Properties of Quantifiers:

- In universal quantifier, ∀x∀y is similar to ∀y∀x.
- In Existential quantifier, ∃x∃y is similar to ∃y∃x.
- ∃x∀y is not similar to ∀y∃x.

First Order Logic (FOL)

→**Quantifiers in First-order logic:** Some Examples of FOL using quantifier:

1. **All birds fly.**

   In this question the predicate is "**fly(bird)**."

   And since there are all birds who fly so it will be represented as follows.

   ∀x bird(x) →fly(x).

2. **Every man respects his parent.**

In this question, the predicate is "**respect(x, y)," where x=man, and y= parent**.

Since there is every man so will use ∀, and it will be represented as follows:

∀x man(x) → respects (x, parent).

3. **Some boys play cricket.**

In this question, the predicate is "**play(x, y)**," where x= boys, and y= game. Since there are some boys so we will use **∃, and it will be represented as**:

∃x boys(x) → play(x, cricket).

First Order Logic (FOL)

→**Quantifiers in First-order logic:** Some Examples of FOL using quantifier:

4. Not all students like both Mathematics and Science.

In this question, the predicate is "**like(x, y)," where x= student, and y= subject**.

Since there are not all students, so we will use ∀ **with negation, so** following representation for this:

$$\neg\forall (x)\ [\ \text{student}(x) \rightarrow \text{like}(x, \text{Mathematics}) \land \text{like}(x, \text{Science})].$$

5. Only one student failed in Mathematics.

In this question, the predicate is "**failed(x, y)," where x= student, and y= subject**.

Since there is only one student who failed in Mathematics, so we will use following representation for this:

$$\exists(x)\ [\ \text{student}(x) \rightarrow \text{failed}\ (x, \text{Mathematics}) \land\forall\ (y)\ [\neg(x==y) \land \text{student}(y) \rightarrow \neg\text{failed}\ (x, \text{Mathematics})].$$

First Order Logic (FOL)

→Quantifiers in First-order logic:

❑Free and Bound Variables:

❑The quantifiers interact with variables which appear in a suitable way. There are two types of variables in First-order logic which are given below:

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

Example: ∀x ∃(y)[P (x, y, z)], where z is a free variable.

❑**Bound Variable:** A variable is said to be a bound variable in a formula if it occurs within the scope of the quantifier.

Example: ∀x [A (x) B(y)], here x and y are the bound variables.

Inference in First Order Logic

- Inference in First-Order Logic is used to deduce new facts or sentences from existing sentences. Before understanding the FOL inference rule, let's understand some basic terminologies used in FOL.
- **Substitution:** is a fundamental operation performed on terms and formulas. It occurs in all inference systems in first-order logic. The substitution is complex in the presence of quantifiers in FOL. If we write **F[a/x]**, so it refers to substitute a constant "**a**" in place of variable "**x**".
- **Equality:** First-Order logic does not only use predicate and terms for making atomic sentences but also uses another way, which is equality in FOL. For this, we can use **equality symbols** which specify that the two terms refer to the same object.
- **Example:** **Brother (John) = Smith.**
- As in the above example, the object referred by the **Brother (John)** is similar to the object referred by **Smith**. The equality symbol can also be used with negation to represent that two terms are not the same objects.
- **Example:** **$\neg$(x=y) which is equivalent to x $\neq$y.**

Inference in First Order Logic

- FOL inference rules for quantifier: As propositional logic we also have inference rules in first-order logic, so following are some basic inference rules in FOL:

1. Universal Generalization
2. Universal Instantiation
3. Existential Instantiation
4. Existential Introduction

Inference in First Order Logic

1. Universal Generalization

- Universal generalization is a valid inference rule which states that if premise P(c) is true for any arbitrary element c in the universe of discourse, then we can have a conclusion as ∀ x P(x).
- It can be represented as: $\frac{P(c)}{\forall x\, P(x)}$
- This rule can be used if we want to show that every element has a similar property.
- In this rule, x must not appear as a free variable.
- **Example:** Let's represent, P(c): "**A byte contains 8 bits**", so for **∀ x P(x)** "**All bytes contain 8 bits**.", it will also be true.

2. Universal Instantiation

- Universal instantiation is also called as universal elimination or UI is a valid inference rule. It can be applied multiple times to add new sentences.
- The new KB is logically equivalent to the previous KB.
- As per UI, **we can infer any sentence obtained by substituting a ground term for the variable**.
- The UI rule state that we can infer any sentence P(c) by substituting a ground term c (a constant within domain x) from **∀ x P(x) for any object in the universe of discourse**.
- It can be represented as: $\dfrac{\forall x\, P(x)}{P(c)}$

2. Universal Instantiation

Example:1: IF "Every person like ice-cream"=> $\forall x\ P(x)$ so we can infer that

"John likes ice-cream" => $P(c)$

Example: 2: Let's take a famous example,

"All kings who are greedy are Evil." So let our knowledge base contains this detail as in the form of FOL:

$\forall x\ \mathbf{king(x) \land greedy\ (x) \rightarrow Evil\ (x)},$

So from this information, we can infer any of the following statements using Universal Instantiation:

King(John) ∧ Greedy (John) → Evil (John),

King(Richard) ∧ Greedy (Richard) → Evil (Richard),

King(Father(John)) ∧ Greedy (Father(John)) → Evil (Father(John)),

3. Existential Instantiation

- Existential instantiation is also called as Existential Elimination, which is a valid inference rule in first-order logic.
- It can be applied only once to replace the existential sentence.
- The new KB is not logically equivalent to old KB, but it will be satisfiable if old KB was satisfiable.
- This rule states that one can infer P(c) from the formula given in the form of ∃x P(x) for a new constant symbol c.
- The restriction with this rule is that c used in the rule must be a new term for which P(c) is true.
- It can be represented as: $\frac{\exists x\, P(x)}{P(c)}$

3. Existential Instantiation

❑**Example:** From the given sentence: **∃x Crown(x) ∧ OnHead(x, John),**

❑So we can infer: **Crown(K) ∧ OnHead(K, John),** as long as K does not appear in the knowledge base.

- The above used K is a constant symbol, which is called **Skolem constant**.
- The Existential instantiation is a special case of **Skolemization process**.

4. Existential Introduction

- An existential introduction is also known as an existential generalization, which is a valid inference rule in first-order logic.
- This rule states that if there is some element c in the universe of discourse which has a property P, then we can infer that there exists something in the universe which has the property P.
- It can be represented as: $\frac{P(c)}{\exists x P(x)}$
- **Example: Let's say that,**

"Priyanka got good marks in English."

"Therefore, someone got good marks in English."

Unification

❑What is Unification?

❑Unification is a process of making two different logical atomic expressions identical by finding a substitution. Unification depends on the substitution process.

❑It takes two literals as input and makes them identical using substitution.

❑Let Ψ_1 and Ψ_2 be two atomic sentences and σ be a unifier such that, $\boldsymbol{\Psi_1\sigma = \Psi_2\sigma}$, then it can be expressed as **UNIFY(Ψ_1, Ψ_2)**.

❑**Example:** Find the MGU for Unify{King(x), King(John)}

Let Ψ_1 = King(x), Ψ_2 = King(John),

Unification

- **Substitution θ = {John/x}** is a unifier for these atoms and applying this substitution, and both expressions will be identical.
- The UNIFY algorithm is used for unification, which takes two atomic sentences and returns a unifier for those sentences (If any exist).
- Unification is a key component of all first-order inference algorithms.
- It returns fail if the expressions do not match with each other.
- The substitution variables are called Most General Unifier or MGU.

Unification

- **E.g.** Let's say there are two different expressions, **P(x, y), and P(a, f(z))**.
- In this example, we need to make both above statements identical to each other.
- For this, we will perform the substitution.

  P(x, y)......... (i)

  P(a, f(z))......... (ii)

- Substitute x with a, and y with f(z) in the first expression, and it will be represented as **a/x** and f(z)/y.
- With both the substitutions, the first expression will be identical to the second expression and the substitution set will be: **[a/x, f(z)/y]**.

Unification

❑Following are some basic conditions for unification:

1. Predicate symbol must be same, atoms or expression with different predicate symbol can never be unified.
2. Number of Arguments in both expressions must be identical.
3. Unification will fail if there are two similar variables present in the same expression.

❑ Unification Algorithm:

Step. 1: If Ψ_1 or Ψ_2 is a variable or constant, then:

a) If Ψ_1 or Ψ_2 are identical, then return NIL.

b) Else if $\Psi_1$is a variable,

a. then if Ψ_1 occurs in Ψ_2, then return FAILURE

b. Else return { (Ψ_2/ Ψ_1)}.

c) Else if Ψ_2 is a variable,

a. If Ψ_2 occurs in Ψ_1 then return FAILURE,

b. Else return {(Ψ_1/ Ψ_2)}.

d) Else return FAILURE.

Unification

❑ Unification Algorithm:

Step.2: If the initial Predicate symbol in Ψ_1 and Ψ_2 are not same, then return FAILURE.

Step. 3: IF Ψ_1 and Ψ_2 have a different number of arguments, then return FAILURE.

Step. 4: Set Substitution set(SUBST) to NIL.

Step. 5: For i=1 to the number of elements in Ψ_1.

a) Call Unify function with the ith element of Ψ_1 and ith element of Ψ_2, and put the result into S.

b) If S = failure then returns Failure

c) If S ≠ NIL then do,

a. Apply S to the remainder of both L1 and L2.

b. SUBST= APPEND(S, SUBST).

Step.6: Return SUBST.

Unification

❑ Implementation of the Algorithm:

Step.1: Initialize the substitution set to be empty.

Step.2: Recursively unify atomic sentences:

a) Check for Identical expression match.

b) If one expression is a variable v_i, and the other is a term t_i which does not contain variable v_i, then:

   a. Substitute t_i / v_i in the existing substitutions

   b. Add t_i /v_i to the substitution setlist.

   c. If both the expressions are functions, then function name must be similar, and the number of arguments must be the same in both the expression.

Unification

❑**For each pair of the following atomic sentences find the most general unifier (If exist).**

1. Find the MGU of {p(f(a), g(Y)) and p(X, X)}

Sol: S_0 => Here, $\Psi_1 = p(f(a), g(Y))$, and $\Psi_2 = p(X, X)$

SUBST θ= {f(a) / X}

S1 => $\Psi_1 = p(f(a), g(Y))$, and $\Psi_2 = p(f(a), f(a))$

SUBST θ= {f(a) / g(y)}, **Unification failed**.

Unification is not possible for these expressions.

Unification

❑**For each pair of the following atomic sentences find the most general unifier (If exist).**

2. Find the MGU of {p(b, X, f(g(Z))) and p(Z, f(Y), f(Y))}

Here, Ψ_1 = p(b, X, f(g(Z))) , and Ψ_2 = p(Z, f(Y), f(Y))

S_0 => { p(b, X, f(g(Z))); p(Z, f(Y), f(Y))}

SUBST θ={b/Z}

S_1 => { p(b, X, f(g(b))); p(b, f(Y), f(Y))}

SUBST θ={f(Y) /X}

S_2 => { p(b, f(Y), f(g(b))); p(b, f(Y), f(Y))}

SUBST θ= {g(b) /Y}

S_2 => { p(b, f(g(b)), f(g(b)); p(b, f(g(b)), f(g(b))} **Unified Successfully.**

And Unifier = { b/Z, f(Y) /X , g(b) /Y}.

❑**For each pair of the following atomic sentences find the most general unifier (If exist).**

3. Find the MGU of {p (X, X), and p (Z, f(Z))}

Here, Ψ_1 = {p (X, X), and Ψ_2 = p (Z, f(Z))

S_0 => {p (X, X), p (Z, f(Z))}

SUBST θ= {X/Z}

S1 => {p (Z, Z), p (Z, f(Z))}

SUBST θ= {f(Z) / Z}, **Unification Failed**.

Hence, unification is not possible for these expressions.

Unification

❑**For each pair of the following atomic sentences find the most general unifier (If exist).**

4. Find the MGU of UNIFY(prime (11), prime(y))

Here, Ψ_1 = {prime(11) , and Ψ_2 = prime(y)}

S_0 => {prime(11) , prime(y)}

SUBST θ= {11/y}

S_1 => {prime(11) , prime(11)} , **Successfully unified.**

Unifier: {11/y}.

Unification

❑**For each pair of the following atomic sentences find the most general unifier (If exist).**

5. Find the MGU of Q(a, g(x, a), f(y)), Q(a, g(f(b), a), x)}

Here, Ψ_1 = Q(a, g(x, a), f(y)), and Ψ_2 = Q(a, g(f(b), a), x)

S_0 => {Q(a, g(x, a), f(y)); Q(a, g(f(b), a), x)}

SUBST θ= {f(b)/x}

S_1 => {Q(a, g(f(b), a), f(y)); Q(a, g(f(b), a), f(b))}

SUBST θ= {b/y}

S_1 => {Q(a, g(f(b), a), f(b)); Q(a, g(f(b), a), f(b))}, **Successfully Unified.**

Unifier: [a/a, f(b)/x, b/y].

Forward Chaining and Backward Chaining

- In artificial intelligence, forward and backward chaining is very important topics.
- Before understanding forward and backward chaining lets first understand that from where these two terms came.
- **Inference engine:**
  - The inference engine is the component of the intelligent system in artificial intelligence, which applies logical rules to the knowledge base to infer new information from known facts.
  - The first inference engine was part of the expert system. Inference engine commonly proceeds in two modes, which are:
    A. **Forward chaining**
    B. **Backward chaining**

Forward Chaining and Backward Chaining

- **Horn Clause and Definite clause:** are the forms of sentences, which enables knowledge base to use a more restricted and efficient inference algorithm.
- Logical inference algorithms use forward and backward chaining approaches, which require KB in the form of the **first-order definite clause**.
- **Definite Clause:** A clause which is a disjunction of literals with **exactly one positive literal** is known as a definite clause or strict horn clause.
- **Horn Clause:** A clause which is a disjunction of literals with **at most one positive literal** is known as horn clause. Hence all the definite clauses are horn clauses.
- **Example: ($\neg p \vee \neg q \vee k$)**. It has only one positive literal k. It is equivalent to $p \wedge q \rightarrow k$.

Forward Chaining and 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.

Forward Chaining and Backward Chaining

❑Properties of Forward-Chaining:

❑It is a down-up approach, as it moves from bottom to top.

❑It is a process of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.

❑Forward-chaining approach is also called as data-driven as we reach to the goal using available data.

❑Forward -chaining approach is commonly used in the expert system, such as CLIPS, business, and production rule systems.

Forward Chaining and Backward Chaining

- **Example:"As per the law, it is a crime for an American to sell weapons 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."**
- To solve the above problem, first, we will convert all the above facts into first-order definite clauses, and then we will use a forward-chaining algorithm to reach the goal.

Forward Chaining and Backward Chaining

- Facts Conversion into FOL:

- It is a crime for an American to sell weapons to hostile nations. (Let's say p, q, and r are variables)

American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p) ...(1)

- Country A has some missiles. **?p Owns(A, p) ∧ Missile(p)**. It can be written in two definite clauses by using Existential Instantiation, introducing new Constant T1.

Owns(A, T1) ......(2)

Missile(T1) ....(3)

- All of the missiles were sold to country A by Robert.

?p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A) ......(4)

- Missiles are weapons.

Missile(p) → Weapons (p) ....(5)

Forward Chaining and Backward Chaining

❑Facts Conversion into FOL:

▪Enemy of America is known as hostile.

Enemy(p, America) →Hostile(p) ....(6)

▪Country A is an enemy of America.

Enemy (A, America) ....(7)

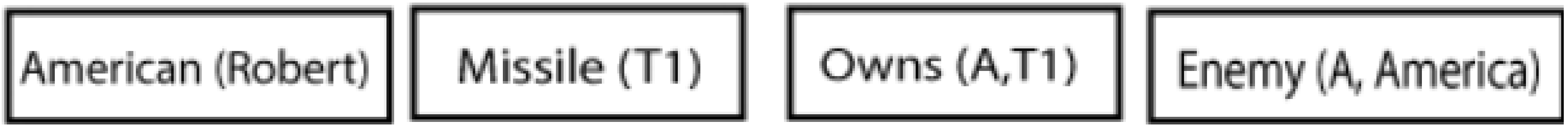
▪Robert is American

American(Robert). ....(8)

Forward Chaining and Backward Chaining

❑Forward Chaining Proof:

❑**Step-1:** In the first step we will start with the known facts and will choose the sentences which do not have implications, such as: **American(Robert), Enemy(A, America), Owns(A, T1), and Missile(T1)**. All these facts will be represented as below.



Forward Chaining and Backward Chaining

❑Forward Chaining Proof:

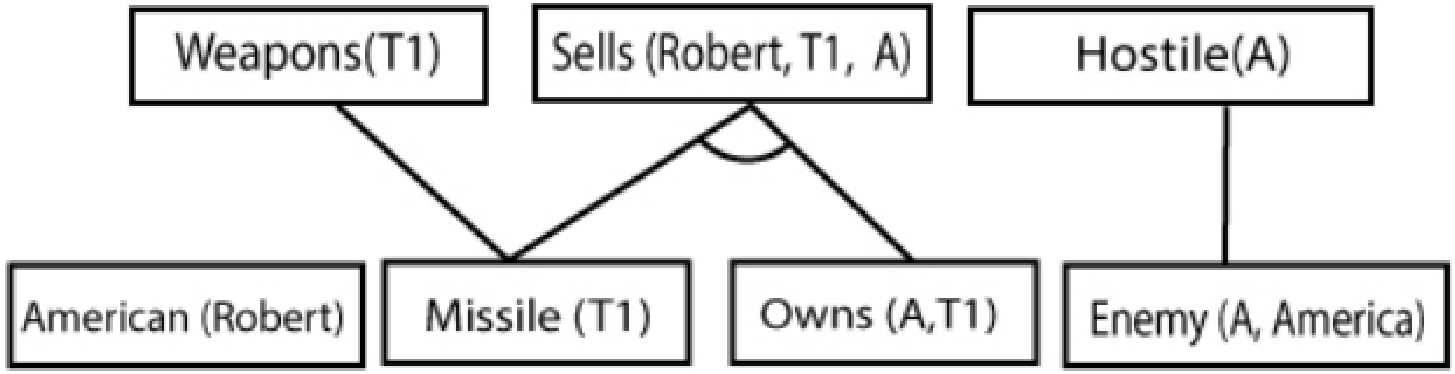
❑**Step-2:** At the second step, we will see those facts which infer from available facts and with satisfied premises.

▪Rule-(1) does not satisfy premises, so it will not be added in the first iteration.

▪Rule-(2) and (3) are already added.

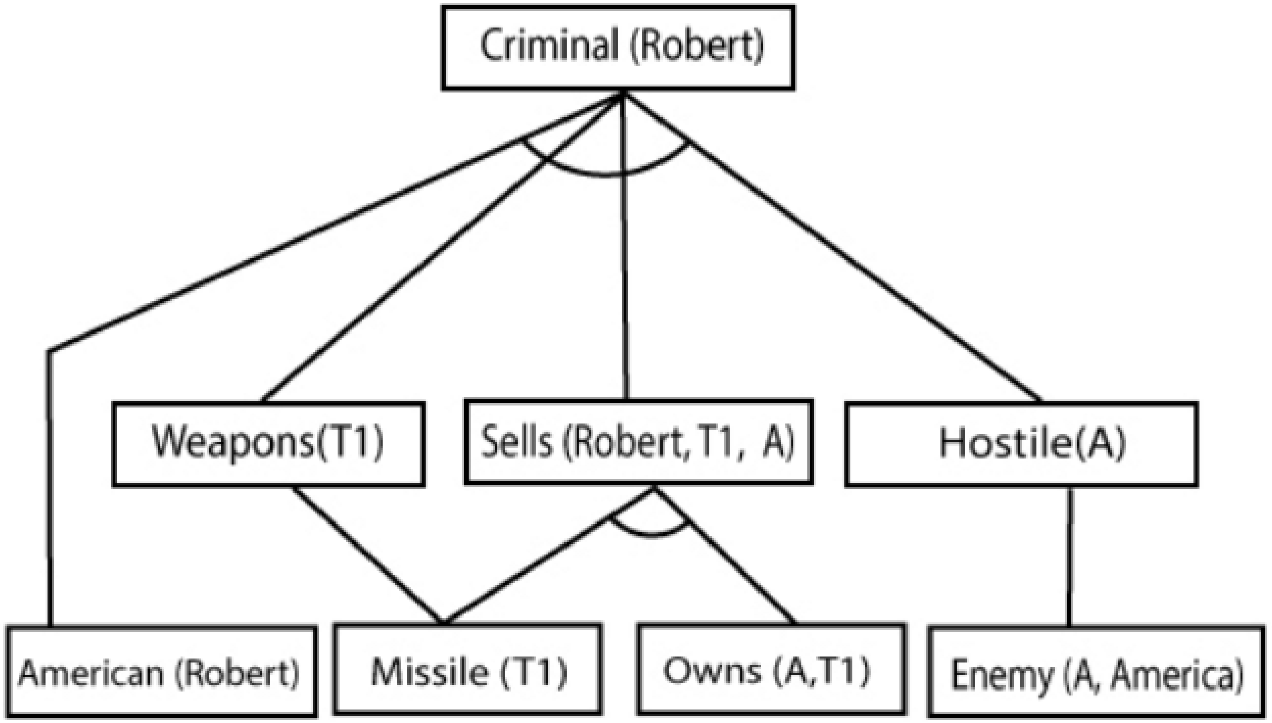
▪Rule-(4) satisfy with the substitution {p/T1}, **so Sells (Robert, T1, A)** is added, which infers from the conjunction of Rule (2) and (3).

▪Rule-(6) is satisfied with the substitution(p/A), so Hostile(A) is added and which infers from Rule-(7).



Forward Chaining and Backward Chaining

- Forward Chaining Proof:
- **Step-3:** At the third step, as we can check Rule-(1) is satisfied with the substitution **{p/Robert, q/T1, r/A}, so we can add Criminal(Robert)** which infers all the available facts. And hence we reached our goal statement.



- Hence it is proved that Robert is Criminal using forward chaining approach.

Forward Chaining and Backward Chaining

❑Backward Chaining:

❑Backward-chaining is also known as a backward deduction or backward reasoning method when using an inference engine.

❑A backward chaining algorithm is a form of reasoning, which starts with the goal and works backward, chaining through rules to find known facts that support the goal.

Forward Chaining and Backward Chaining

❑Properties of the Backward Chaining:

❑It is known as a top-down approach.

❑Backward-chaining is based on modus ponens inference rule.

❑In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true.

❑It is called a goal-driven approach, as a list of goals decides which rules are selected and used.

❑Backward -chaining algorithm is used in game theory, automated theorem proving tools, inference engines, proof assistants, and various AI applications.

❑The backward-chaining method mostly used a **depth-first search** strategy for proof.

Forward Chaining and Backward Chaining

- Example:
- In backward-chaining, we will use the same above example, and will rewrite all the rules.

1. **American (p) ∧ weapon(q) ∧ sells (p, q, r) ∧ hostile(r) → Criminal(p) ...(1)**

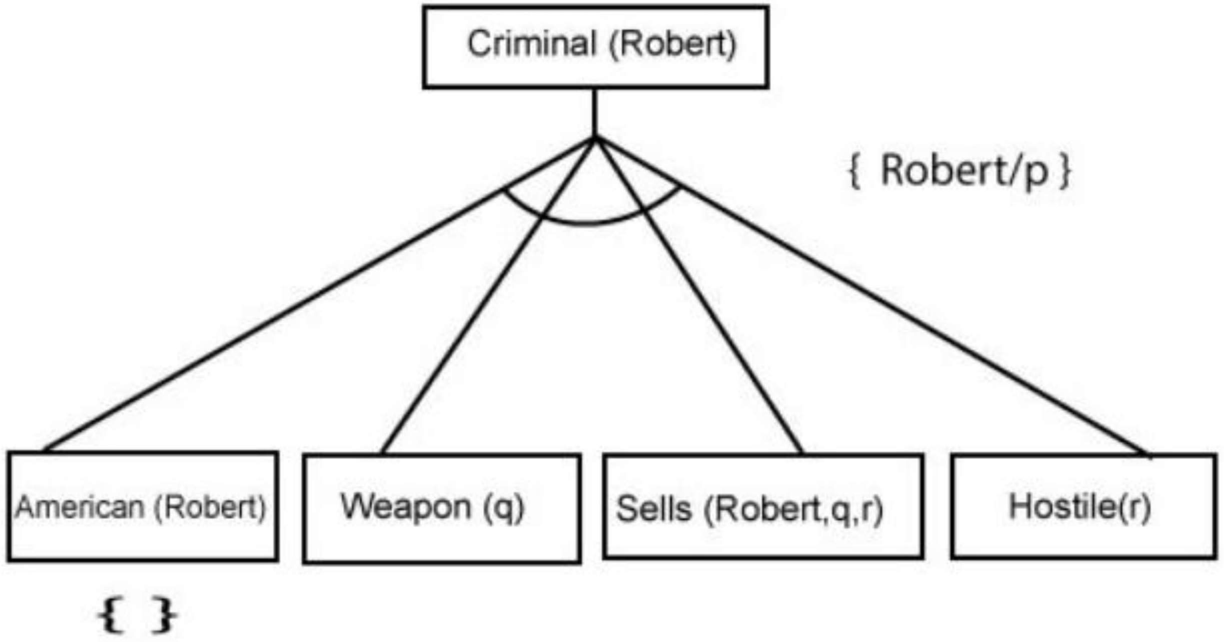
   Owns(A, T1) ....(2)
2. **Missile(T1)**
3. **?p Missiles(p) ∧ Owns (A, p) → Sells (Robert, p, A) ......(4)**
4. **Missile(p) → Weapons (p) ....(5)**
5. **Enemy(p, America) →Hostile(p) ....(6)**
6. **Enemy (A, America) ....(7)**
7. **American(Robert). ....(8)**

Forward Chaining and Backward Chaining

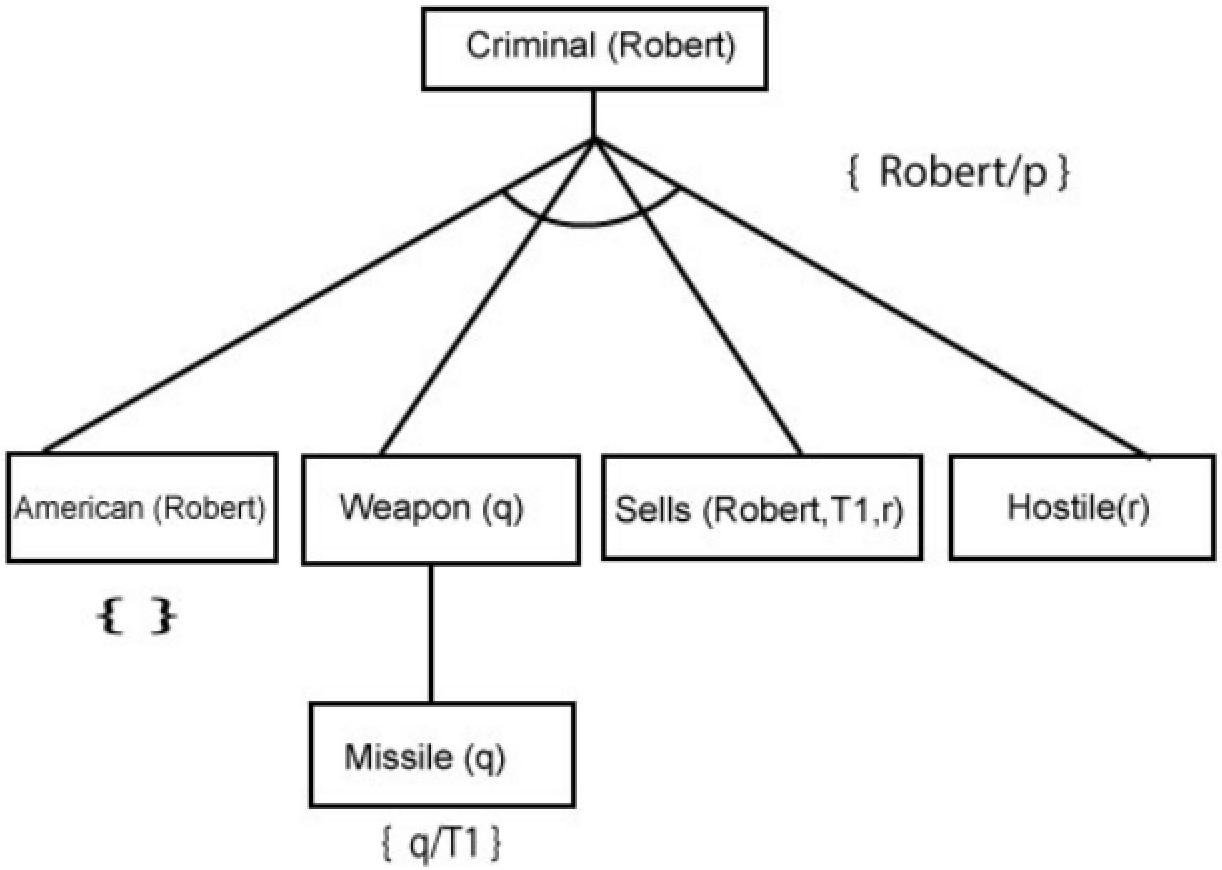
- Backward-Chaining proof: In Backward chaining, we will start with our goal predicate, which is **Criminal(Robert)**, and then infer further rules.
- **Step-1:** At the first step, we will take the goal fact. And from the goal fact, we will infer other facts, and at last, we will prove those facts true. So our goal fact is "Robert is Criminal," so following is the predicate of it.

Forward Chaining and Backward Chaining

- Backward-Chaining proof: In Backward chaining, we will start with our goal predicate, which is **Criminal(Robert)**, and then infer further rules.
- **Step-2:** At the second step, we will infer other facts form goal fact which satisfies the rules. So as we can see in Rule-1, the goal predicate Criminal (Robert) is present with substitution {Robert/P}. So we will add all the conjunctive facts below the first level and will replace p with Robert.
- **Here we can see American (Robert) is a fact, so it is proved here.**

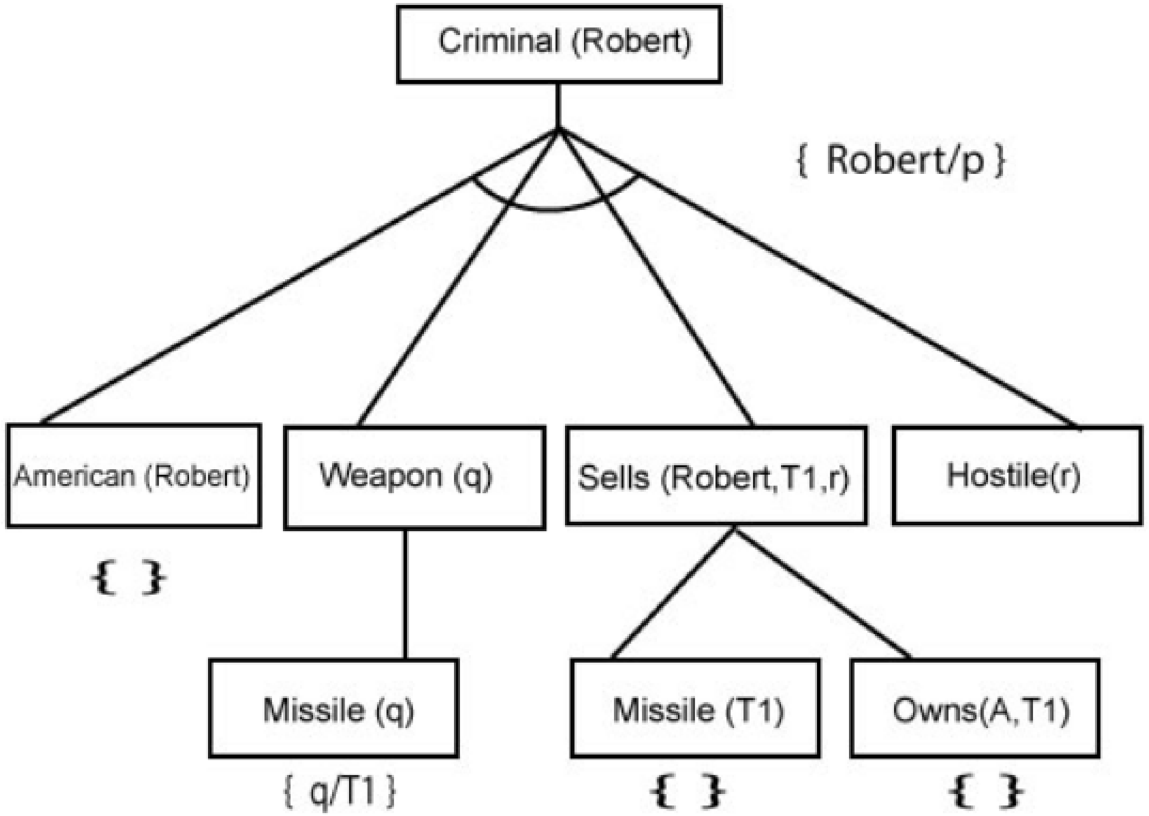


- ❑Backward-Chaining proof: In Backward chaining, we will start with our goal predicate, which is **Criminal(Robert)**, and then infer further rules.
- ❑**Step-3:** At step-3, we will extract further fact Missile(q) which infer from Weapon(q), as it satisfies Rule-(5). Weapon (q) is also true with the substitution of a constant T1 at q.



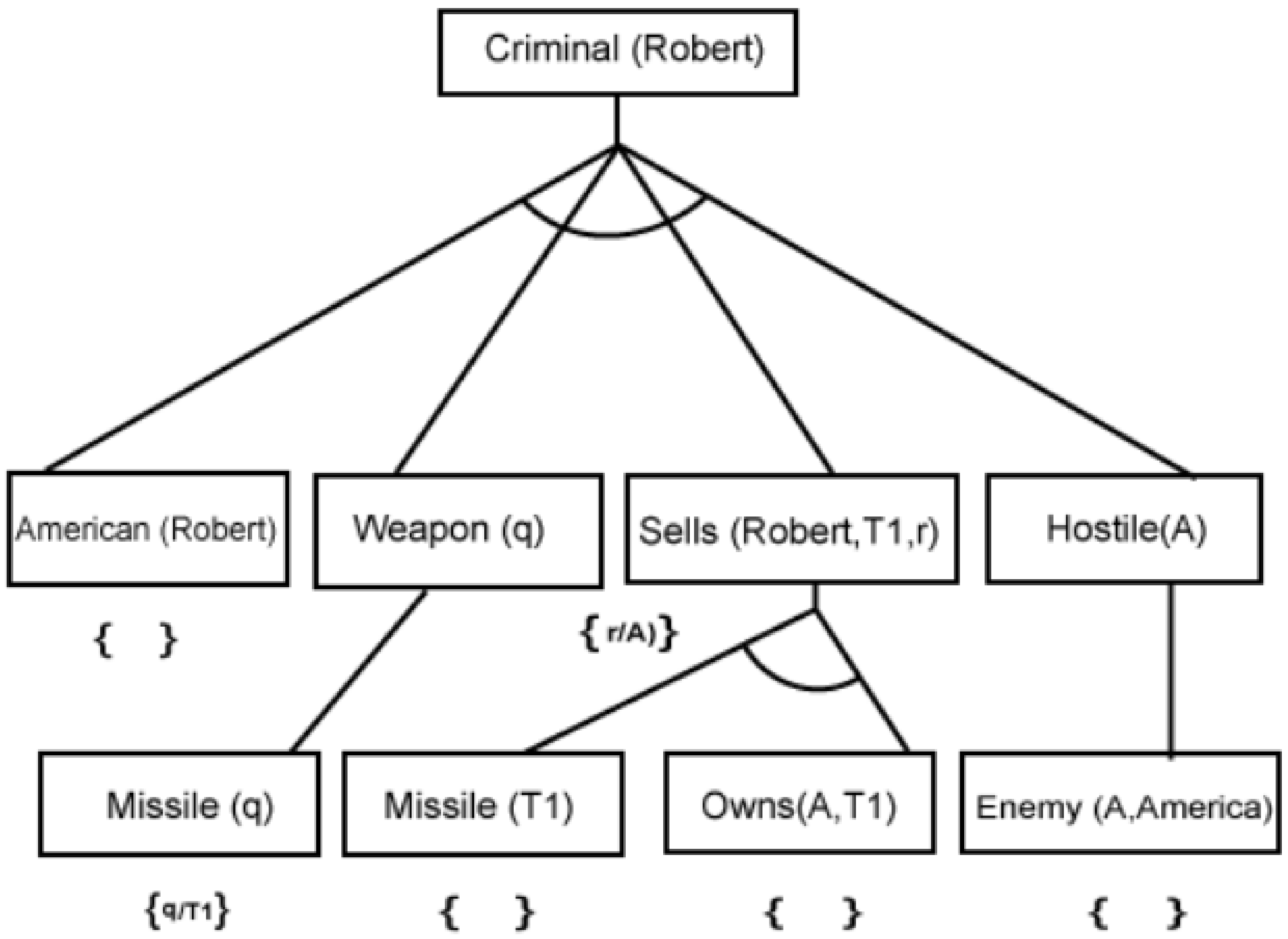
Forward Chaining and Backward Chaining

- Backward-Chaining proof: In Backward chaining, we will start with our goal predicate, which is **Criminal(Robert)**, and then infer further rules.
- **Step-4:** At step-4, we can infer facts Missile(T1) and Owns(A, T1) form Sells(Robert, T1, r) which satisfies the **Rule- 4**, with the substitution of A in place of r. So these two statements are proved here.



Forward Chaining and Backward Chaining

- Backward-Chaining proof: In Backward chaining, we will start with our goal predicate, which is **Criminal(Robert)**, and then infer further rules.
- **Step-5:** At step-5, we can infer the fact **Enemy(A, America)** from **Hostile(A)** which satisfies Rule- 6. And hence all the statements are proved true using backward chaining.



Note for Students

- ❑This power point presentation is for lecture, therefore it is suggested that also utilize the text books and lecture notes.
- ❑Also Refer the solved and unsolved examples of Text and Reference Books.