UNIT IV

Knowledge

Contents

Unit IV	Knowledge	07 Hours
Logical Agents, Knowledge-Based Agents, The Wumpus World, Logic, Propositional Logic: A Very Simple Logic, Propositional Theorem Proving, Effective Propositional Model Checking, Agents Based on Propositional Logic, First-Order Logic, Representation Revisited, Syntax and Semantics of First-Order Logic, Using First-Order Logic, Knowledge Engineering in First-Order Logic.		
#Exemplar/Case Studies	BBC To Launch AI - Enabled Interactive Radio Show For Amazon Echo And Google Home Chatbots	
***Mapping of Course Outcomes for Unit IV**	CO3, CO4	

Propositional logic

- **Propositional Logic (PL)** is a branch of logic that focuses on **statements (propositions)** that can be **either true or false**. It is also known as **Boolean logic** since the truth values are binary—either **True (1)** or **False (0)**.
- In AI, propositional logic forms the **foundation for logical reasoning**, allowing systems to represent **facts** and **rules** about a problem domain. These rules help the system infer new information or make decisions based on the given inputs.
- Propositional logic simplifies **knowledge representation** by breaking down reasoning into **atomic statements** or **propositions**. For example, an AI system used in home automation might have propositions such as:

  P: “The light is on.”

  Q: “The window is open.”
- Using **logical connectives**, the system can combine these propositions to represent more complex statements like:
  “If the light is on and the window is open, turn off the light.”
- By using propositional logic, AI systems can **reason** effectively and perform tasks like **automated decision-making, knowledge representation**, and **game playing**.

Basic Facts About Propositional Logic

1. Propositions are Declarative Statements:

- In **propositional logic**, each statement, known as a **proposition**, is either **True** or **False**.
  Example:

  P: “It is raining.” (True or False)
  Q: “The heater is on.” (True or False)

2. Atomic Propositions:

- These are **simple, indivisible statements** that cannot be broken down further. Each atomic proposition represents a basic fact or condition.
  Example: “The door is closed.”

3. Compound Propositions:

- **Multiple atomic propositions** can be combined using **logical connectives** (like AND, OR, NOT) to create **compound propositions**.
  Example: “The door is closed AND the heater is on.”

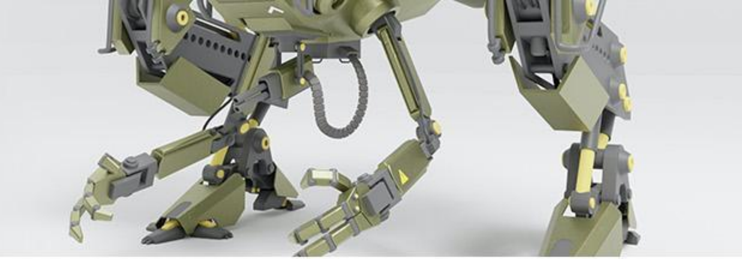
Basic Facts About Propositional Logic

4. Binary Truth Values:

- Every proposition has a **binary truth value**: it can only be **True (1)** or **False (0)**. There are no intermediate states. This simplicity makes propositional logic ideal for **clear-cut decisions**.

5. Logical Connectives Combine Propositions:

- Logical connectives such as **AND, OR, NOT, IF-THEN,** and **IF AND ONLY IF** allow us to create more complex propositions from simple ones.



Syntax of Propositional Logic

- The **syntax** of propositional logic defines the **rules for creating valid propositions**.
- In propositional logic, we combine **atomic propositions** using **logical connectives** to form more complex statements, known as **compound propositions**.
- **Building Blocks of Propositional Logic Syntax**

1. **Atomic Propositions:**

- These are basic statements that represent individual facts or conditions.
  Example:
- **P**: “It is raining.”
- **Q**: “The heater is on.”

Syntax of Propositional Logic

2. Logical Connectives:

- Connectives are used to combine atomic propositions to form **compound propositions**.
- **AND (∧)**: True if both propositions are true.
- **OR (∨)**: True if at least one proposition is true.
- **NOT (¬)**: Negates the truth value of a proposition.
- **IF-THEN (→)**: True unless the first proposition is true and the second is false.
- **IF AND ONLY IF (↔)**: True if both propositions have the same truth value.

3. Compound Propositions:

- These are **more complex statements** formed by connecting atomic propositions using logical connectives.

Syntax of Propositional Logic

- Example:

  “If it is raining and the heater is on, then the room will be warm.”
  This can be written in **propositional logic syntax** as: $(P \land Q) \rightarrow R$

Where:

- **P**: “It is raining.”
- **Q**: “The heater is on.”
- **R**: “The room is warm.”

Syntax of Propositional Logic

- **Example of Propositional Logic**

Let's explore a **real-world scenario** where propositional logic is applied in AI. Consider a **home automation system** that needs to decide whether to **turn on the air conditioner** based on the weather conditions and indoor temperature.

Scenario:

- **P**: "It is hot outside." **Q**: "The windows are open." **R**: "Turn on the air conditioner."

Using **propositional logic**, we can represent the system's decision-making with the following compound proposition:

$(P \land \neg Q) \rightarrow R$

This logic reads as:
"If it is hot outside **AND** the windows are not open, then **turn on the air conditioner**."

Logical Connectives in Propositional Logic

Logical connectives are essential operators that combine **atomic propositions** to form **compound propositions**. These connectives allow AI systems to build more complex rules and perform logical reasoning.

- **AND (∧):**
  - The result is **True** only if both propositions are true.
    Example: If **P** is “It is hot” and **Q** is “The fan is on”, then $(P \land Q)$ means both conditions are satisfied.
- **OR (∨):**
  - The result is **True** if at least one of the propositions is true.
    Example: $(P \lor Q)$ will be true if either it is hot or the fan is on.

Logical Connectives in Propositional Logic

- **NOT (¬):**
  - This **inverts** the truth value of the proposition.
    Example: If **P** is true, ¬P will be false.
- **IF-THEN (→):**
  - This implies that if the first proposition is true, the second must also be true for the compound statement to be true.
    Example: “If it rains, then the ground will be wet” (P → Q).
- **IF AND ONLY IF (↔):**
  - This is true only when both propositions have the **same truth value** (either both true or both false).
    Example: “It is cloudy if and only if it will rain” (P ↔ Q).

Precedence of Connectives in Propositional Logic

P	Q	$\neg$ P	P $\wedge$ Q	P $\vee$ Q	P $\rightarrow$ Q	P $\Leftrightarrow$ Q
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

Precedence of Connectives in Propositional Logic

- When evaluating **compound propositions** with multiple logical connectives, it's important to follow a specific **order of precedence** to ensure accurate results. Similar to arithmetic operations, **logical operators** are evaluated in a defined sequence, from highest to lowest precedence.

Order of Precedence

1. **NOT (¬)** – Negation has the **highest precedence** and is evaluated first.
2. **AND (∧)** – Conjunction is evaluated next, after negations are resolved.
3. **OR (∨)** – Disjunction comes after AND operations.
4. **IF-THEN (→)** – Implication is evaluated after OR.
5. **IF AND ONLY IF (↔)** – Biconditional has the **lowest precedence**.

Logical Equivalence in Propositional Logic

- **Logical equivalence** occurs when two or more logical expressions produce the **same truth values** for all possible combinations of their propositions.
- In other words, two statements are logically equivalent if they always have the **same result**, regardless of the truth values of the individual propositions.
- Two propositions P and Q are logically equivalent if:

  $$P \equiv Q$$

Logical Equivalence in Propositional Logic

Example of Logical Equivalence

1. **De Morgan's Laws:**
   - These laws show how **negations of conjunctions** and **disjunctions** behave:

$$\neg(P \land Q) \equiv (\neg P \lor \neg Q)$$

$$\neg(P \lor Q) \equiv (\neg P \land \neg Q)$$

2. **Double Negation:**
   - Negating a negation gives the original proposition:

$$\neg(\neg P) \equiv P$$

3. **Implication and Disjunction:**
   - An implication can be rewritten as:

$$P \to Q \equiv \neg P \lor Q$$

Logical Equivalence in Propositional Logic

Tautologies and Contradictions:

- **Tautology:**A tautology is a statement that is **always true**, no matter the truth values of its individual propositions.
  - **Example:** $P \lor \neg P \equiv \text{True}$
- **Contradiction:** A contradiction is a statement that is **always false**.
  - **Example:** $P \land \neg P \equiv \text{False}$

Applications of Propositional Logic in AI

1. Knowledge Representation in Expert Systems:

- Represents **rules and facts** to solve domain-specific problems (e.g., medical diagnosis systems).

2. Reasoning and Decision-Making:

- AI agents use logical rules to make **decisions** (e.g., robot vacuum cleaners deciding when to start cleaning).

3. Natural Language Processing (NLP):

- Helps analyze text and respond logically (e.g., chatbots understanding weather-related queries).

4. Game-Playing AI:

- Uses logic to **make strategic moves** (e.g., deciding checkmate in chess).

Limitations of Propositional Logic

1. Inability to Handle Complex Relationships

- Propositional logic cannot represent **relationships between multiple objects** or deal with hierarchies of information.

2. No Handling of Uncertainty

- It works only with **true or false** values and cannot deal with **probabilities or uncertain outcomes**, limiting its use in real-world applications involving incomplete data.

3. Limited Expressiveness

- It cannot represent **time-based sequences** or dynamic events, which are crucial in some AI systems like speech recognition and robotics.

4. Scalability Issues

- As the number of propositions grows, the **complexity of expressions** increases, making reasoning slower and harder to manage.

First-Order Logic in Artificial intelligence

- In propositional logic, we can only represent the facts, which are either true or false.
- PL is not sufficient to represent the complex sentences or natural language statements.
- The propositional logic has very limited expressive power.
- Consider the following sentence, which we cannot represent using PL logic.

"Some humans are intelligent", or

"Sachin likes cricket."

To represent the above statements, PL logic is not sufficient, so we required some more powerful logic, such as first-order logic.

First-Order Logic in Artificial intelligence

- 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 in Artificial intelligence

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, ...

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

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

First-Order Logic in Artificial intelligence

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.

Basic Elements of First-order logic:

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 in Artificial intelligence

Atomic sentences:

- 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 in Artificial intelligence

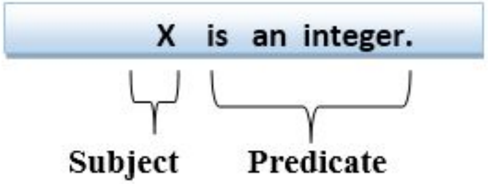
Complex Sentences:

- 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 in Artificial intelligence

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 in Artificial intelligence

Universal Quantifier:

- 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.
- **Note: In universal quantifier we use implication "→".**

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

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

First-Order Logic in Artificial intelligence

Example:

All man drink coffee.

Let a variable x which refers to a man

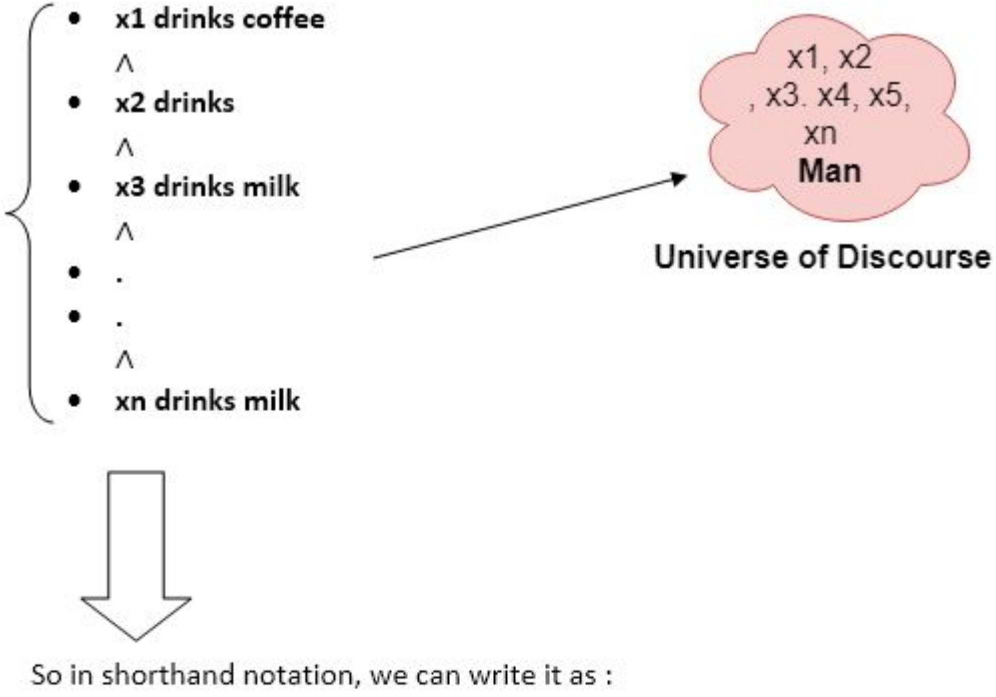
so all x can be represented in

UOD as below:

∀x man(x) → drink (x, coffee).

It will be read as: There are all x

where x is a man who drink coffee.



First-Order Logic in Artificial intelligence

Existential Quantifier:

- 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.

Note: In Existential quantifier we always use AND or Conjunction symbol (∧).

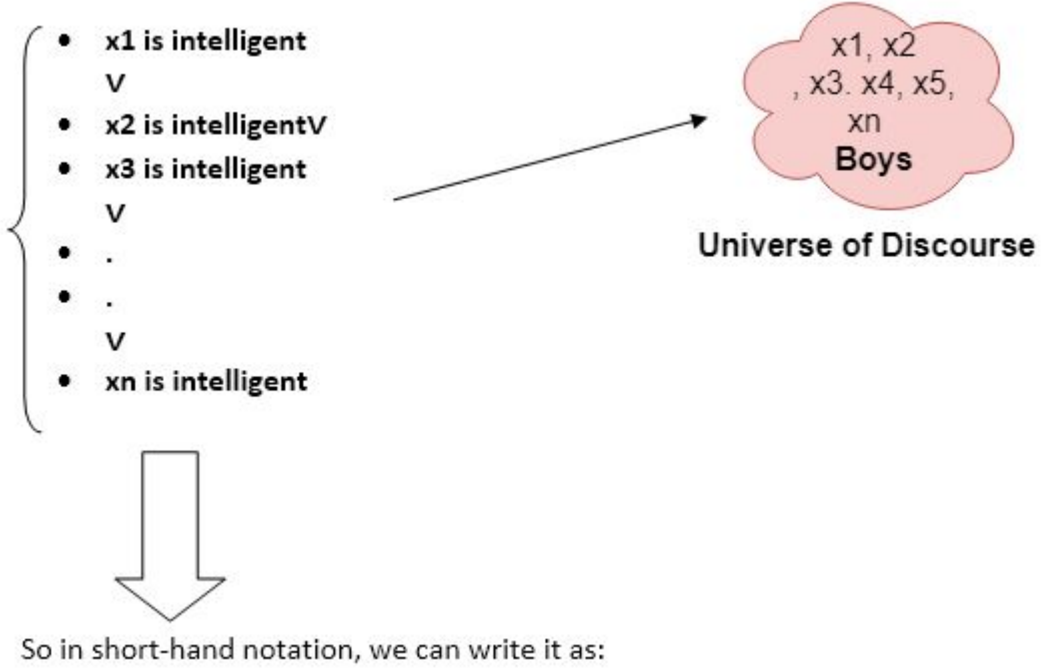
If x is a variable, then existential quantifier will be ∃x or ∃(x). And it will be read as:

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

First-Order Logic in Artificial intelligence

Example:

Some boys are intelligent.



∃x: boys(x) ∧ intelligent(x)

It will be read as: There are some

x where x is a boy who is intelligent.

First-Order Logic in Artificial intelligence

Points to remember:

- The main connective for universal quantifier $\forall$ is implication $\rightarrow$.
- The main connective for existential quantifier $\exists$ is and $\wedge$.

Properties of Quantifiers:

- In universal quantifier, $\forall x \forall y$ is similar to $\forall y \forall x$.
- In Existential quantifier, $\exists x \exists y$ is similar to $\exists y \exists x$.
- $\exists x \forall y$ is not similar to $\forall y \exists x$.

First-Order Logic in Artificial intelligence

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.

$$\forall x \; \mathbf{bird(x) \rightarrow fly(x)}.$$

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:

$$\forall x \; \mathbf{man(x) \rightarrow respects\ (x, parent)}.$$

First-Order Logic in Artificial intelligence

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 $\exists$ **, and it will be represented as**:

$$\exists x \; \mathbf{boys}(x) \rightarrow \mathbf{play}(x, \mathbf{cricket}).$$

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 $\forall$ **with negation, so** following representation for this:

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

First-Order Logic in Artificial intelligence

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 in Artificial intelligence

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: $\forall x\ \exists(y)[P(x, y, z)]$, 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: $\forall x\ [A(x)\ B(y)]$, here x and y are the bound variables.

Resolution in FOL

- Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions.
- Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule which can efficiently operate on the **conjunctive normal form or clausal form**.
- Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule which can efficiently operate on the **conjunctive normal form or clausal form**.

Resolution in FOL

Steps for Resolution:

1. Conversion of facts into first-order logic.
2. Convert FOL statements into CNF
3. Negate the statement which needs to prove (proof by contradiction)
4. Draw resolution graph (unification).

Resolution in FOL

Convert FOL statements into CNF

1. Eliminate biconditionals and implications:

- Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$.
- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \lor \beta$.

2. Move $\neg$ inwards:

- $\neg(\forall\ x\ p) \equiv \exists\ x\ \neg p$,
- $\neg(\exists\ x\ p) \equiv \forall\ x\ \neg p$,
- $\neg(\alpha \lor \beta) \equiv \neg\alpha \land \neg\beta$,
- $\neg(\alpha \land \beta) \equiv \neg\alpha \lor \neg\beta$,
- $\neg\neg\alpha \equiv \alpha$

Resolution in FOL

3. Standardize variables apart by renaming them: each quantifier should use a different variable.

4. Skolemize: each existential variable is replaced by a Skolem constant or Skolem function of the
enclosing universally quantified variables.

• For instance, ∃x Rich(x) becomes Rich(G1) where G1 is a new Skolem constant.

5. Drop universal quantifiers

• For instance, ∀ x P erson(x) becomes Person(x).

Resolution in FOL

Example:

- John likes all kind of food.
- Apple and vegetable are food.
- Anything anyone eats and not killed is food.
- Anil eats peanuts and still alive.
- Harry eats everything that Anil eats.
- **John likes peanuts.**

Resolution in FOL

Step-1: Conversion of Facts into FOL

In the first step we will convert all the given statements into its first order logic.

a. $\forall x: food(x) \rightarrow likes(John, x)$

b. $food(Apple) \land food(vegetables)$

c. $\forall x \forall y: eats(x, y) \land \neg killed(x) \rightarrow food(y)$

d. $eats(Anil, Peanuts) \land alive(Anil)$.

e. $\forall x : eats(Anil, x) \rightarrow eats(Harry, x)$

f. $\forall x: \neg killed(x) \rightarrow alive(x)$

g. $\forall x: alive(x) \rightarrow \neg killed(x)$

added predicates.

h. $likes(John, Peanuts)$

Resolution in FOL

Step-2: Conversion of FOL into CNF

In First order logic resolution, it is required to convert the FOL into CNF as CNF form makes easier for resolution proofs.

2.1 Eliminate all implication (→) and rewrite

1. $\forall x \neg \text{food}(x) \lor \text{likes}(\text{John}, x)$
2. $\text{food}(\text{Apple}) \land \text{food}(\text{vegetables})$
3. $\forall x \; \forall y \neg [\text{eats}(x, y) \land \neg \text{killed}(x)] \lor \text{food}(y)$
4. $\text{eats}(\text{Anil}, \text{Peanuts}) \land \text{alive}(\text{Anil})$
5. $\forall x \neg \text{eats}(\text{Anil}, x) \lor \text{eats}(\text{Harry}, x)$
6. $\forall x \neg [\neg \text{killed}(x)] \lor \text{alive}(x)$
7. $\forall x \neg \text{alive}(x) \lor \neg \text{killed}(x)$

Resolution in FOL

Step-2: Conversion of FOL into CNF

2.1 Eliminate all implication (→) and rewrite

1. ∀x ¬ food(x) V likes(John, x)
2. food(Apple) Λ food(vegetables)
3. **∀x ∀y ¬ [eats(x, y) Λ ¬ killed(x)] V food(y)**
4. eats (Anil, Peanuts) Λ alive(Anil)
5. ∀x ¬ eats(Anil, x) V eats(Harry, x)
6. **∀x¬ [¬ killed(x)] V alive(x)**
7. ∀x ¬ alive(x) V ¬ killed(x)
8. likes(John, Peanuts).

2.2 Move negation (¬)inwards and rewrite

1. ∀x ¬ food(x) V likes(John, x)
2. food(Apple) Λ food(vegetables)
3. ∀x ∀y ¬ eats(x, y) V killed(x) V food(y)
4. eats (Anil, Peanuts) Λ alive(Anil)
5. ∀x ¬ eats(Anil, x) V eats(Harry, x)
6. ∀x killed(x)] V alive(x)
7. ∀x ¬ alive(x) V ¬ killed(x)
8. likes(John, Peanuts).

Resolution in FOL

Step-2: Conversion of FOL into CNF

2.2 Move negation (¬)inwards and rewrite

1. ∀x ¬ food(x) V likes(John, x)
2. food(Apple) Λ food(vegetables)
3. ∀x ∀y ¬ eats(x, y) V killed(x) V food(y)
4. eats (Anil, Peanuts) Λ alive(Anil)
5. ∀x ¬ eats(Anil, x) V eats(Harry, x)
6. ∀x killed(x)] V alive(x)
7. ∀x ¬ alive(x) V ¬ killed(x)
8. likes(John, Peanuts).

2.3 Rename variables or standardize variables

1. ∀x ¬ food(x) V likes(John, x)
2. food(Apple) Λ food(vegetables)
3. ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)
4. eats (Anil, Peanuts) Λ alive(Anil)
5. ∀w¬ eats(Anil, w) V eats(Harry, w)
6. ∀g killed(g) V alive(g)
7. ∀k ¬ alive(k) V ¬ killed(k)
8. likes(John, Peanuts).

Resolution in FOL

Step-2: Conversion of FOL into CNF
2.3 Rename variables or standardize variables

1. ∀x ¬ food(x) V likes(John, x)
2. food(Apple) Λ food(vegetables)
3. ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)
4. eats (Anil, Peanuts) Λ alive(Anil)
5. ∀w¬ eats(Anil, w) V eats(Harry, w)
6. ∀g killed(g)] V alive(g)
7. ∀k ¬ alive(k) V ¬ killed(k)
8. likes(John, Peanuts).

2.4 Eliminate existential instantiation quantifier by elimination.

In this step, we will eliminate existential quantifier ∃, and this process is known as **Skolemization**. But in this example problem since there is no existential quantifier so all the statements will remain same in this step.

Resolution in FOL

Step-2: Conversion of FOL into CNF
2.3 Rename variables or standardize variables

1. **∀x ¬ food(x) V likes(John, x)**
2. food(Apple) Λ food(vegetables)
3. **∀y ∀z ¬ eats(y, z) V killed(y) V food(z)**
4. eats (Anil, Peanuts) Λ alive(Anil)
5. **∀w¬ eats(Anil, w) V eats(Harry, w)**
6. **∀g killed(g)] V alive(g)**
7. **∀k ¬ alive(k) V ¬ killed(k)**
8. likes(John, Peanuts).

2.5 Drop Universal quantifiers.

In this step we will drop all universal quantifier since all the statements are not implicitly quantified so we don't need it.

1. ¬ food(x) V likes(John, x)
2. food(Apple)
3. food(vegetables)
4. ¬ eats(y, z) V killed(y) V food(z)
5. eats (Anil, Peanuts)
6. alive(Anil)
7. ¬ eats(Anil, w) V eats(Harry, w)
8. killed(g) V alive(g)
9. ¬ alive(k) V ¬ killed(k)
10. likes(John, Peanuts).

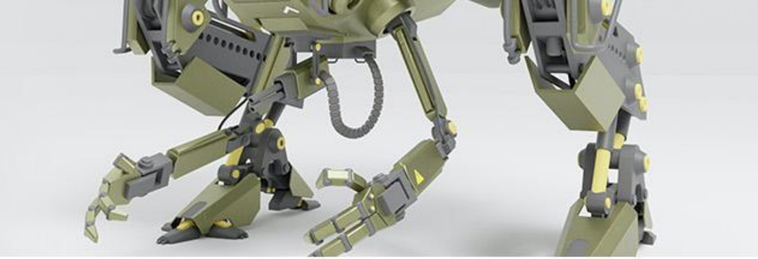
Resolution in FOL

Step-3: Negate the statement to be proved

In this statement, we will apply negation to the conclusion statements, which will be written as ¬likes(John, Peanuts)

Step-4: Draw Resolution graph:

Now in this step, we will solve the problem by resolution tree using substitution. For the above problem, it will be given as follows:

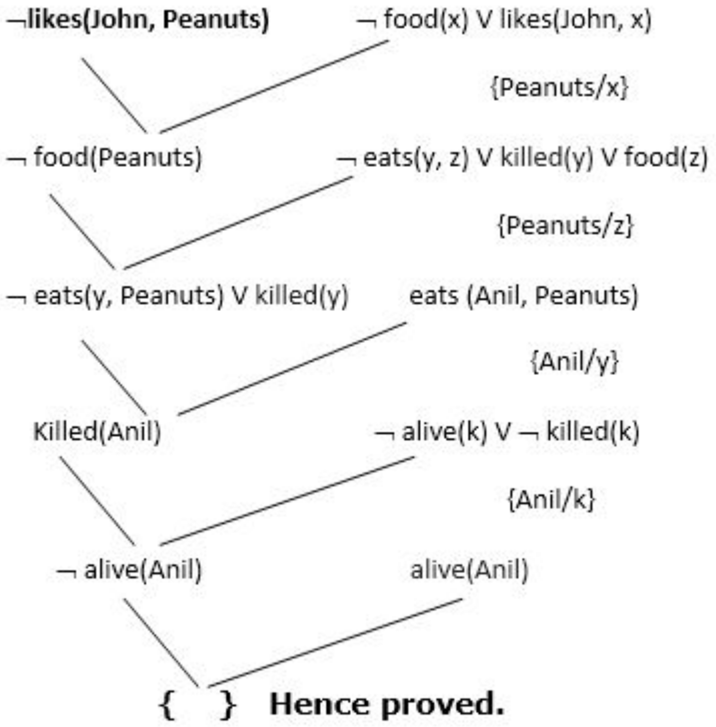


Resolution in FOL

Step-4: Draw Resolution graph:

Now in this step, we will solve the problem by resolution tree using substitution. For the above problem, it will be given as follows:

1. ¬ food(x) V likes(John, x)
2. food(Apple)
3. food(vegetables)
4. ¬ eats(y, z) V killed(y) V food(z)
5. eats (Anil, Peanuts)
6. alive(Anil)
7. ¬ eats(Anil, w) V eats(Harry, w)
8. killed(g) V alive(g)
9. ¬ alive(k) V ¬ killed(k)
10. likes(John, Peanuts).



Resolution in FOL

Hence the negation of the conclusion has been proved as a complete contradiction with the given set of statements.