**Notes on Propositions**

* **Definition**: A **proposition** is a declarative sentence that is either **true** or **false**.

**Characteristics of Propositions**

1. A proposition always has a **definite truth value** (True or False).
2. **Examples of Propositions**:
   * "In a pandemic situation, using a face mask is mandatory." (Can be True or False)
   * "Even semester classes normally start from January." (Can be True or False)

**Non-Propositional Statements**

* Some sentences **do not qualify** as propositions because they **lack a definite truth value**:
  + "1 USD = INR?" (Question, no definite answer)
  + "Please submit your project synopsis by the end of this week." (Command, not True or False)
  + "When do you have the Mid-Sem exam?" (Question, no truth value)

Thus, **only declarative statements with a definite truth value** are considered propositions.

**Notes on Tautology, Contradiction, and Logical Equivalence**

**Tautology & Contradiction**

* A **tautology** is a proposition that is **always true**, regardless of conditions.
* A **contradiction** is a proposition that is **always false**, under all conditions.

**Symbolic Representation of Propositions**

* Propositions are often represented by symbols like **p, q, r**, etc.
* Example:
  + Let **p** = "Every student in the class passed the final examination."
  + **p** can be **true** or **false** depending on reality.

**Logical Equivalence**

* Two propositions **p and q** are **equivalent** if:
  + When **p is true**, **q is also true**.
  + When **p is false**, **q is also false**.

**Example of Equivalent Propositions**:

* **p** = "Water froze this morning."
* **q** = "The temperature was below 0°C this morning."
* Since water freezes only when the temperature is below 0°C, these two statements are **logically equivalent**.

**More Examples of Logical Equivalence**

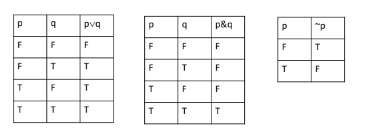
* **Equivalent Propositions**:
  + **p** = "He was born in 2002."
  + **q** = "He will be completing 20 years of age in 2022."
  + Since both statements imply the same fact, **p and q are equivalent**.
* **Non-Equivalent Propositions**:
  + **p** = "x is a prime number."
  + **q** = "x is not divisible by 2."
  + These are **not equivalent** because a number not divisible by 2 is not necessarily a prime (e.g., 9 is not divisible by 2 but is not prime).

**Combining Propositions with Logical Operators**

1. **Disjunction (p ∨ q) – "OR" Operation**
   * Denoted as **p ∨ q** ("p OR q").
   * **True if at least one** of the propositions is **true**.
   * **False only if both** p and q are **false**.
2. **Conjunction (p ∧ q) – "AND" Operation**
   * Denoted as **p ∧ q** ("p AND q").
   * **True only if both** p and q are **true**.
   * **False if at least one** of them is **false**.
3. **Negation(~p)- “NOT” Operation**

Let p be a proposition. We define the negation of p by ~p. ~p is

true when p is false and is false when p is true.



**Notes on Compound Propositions & Logical Implications**

**Compound Proposition**

* A **compound proposition** is formed by combining multiple propositions using logical operators.

**Conditional Statement (Implication: p → q)**

* Example:
  + **p** = "The temperature exceeds 70°C."
  + **q** = "The alarm will be sounded."
  + **r** = "If the temperature exceeds 70°C, then the alarm will be sounded." (**p → q**)

**Truth Table for p → q:**

|  |  |  |
| --- | --- | --- |
| **p** | **q** | **p → q** |
| T | T | T |
| T | F | F |
| F | T | T |
| F | F | T |

* The implication **p → q** is:
  + **True** when both **p and q are true**.
  + **False** when **p is true but q is false**.
  + **True** when **p is false**, regardless of q.

**Biconditional Statement (p ↔ q)**

* **p ↔ q** means **"p if and only if q"**, meaning both must have the same truth value for the statement to be true.
* Example:
  + **p** = "A new computer will be acquired."
  + **q** = "Additional funding is available."
  + **r** = "A new computer will be acquired if and only if additional funding is available." (**p ↔ q**)

**Truth Table for p ↔ q:**

|  |  |  |
| --- | --- | --- |
| **p** | **Q** | **p ↔ q** |
| F | F | T |
| F | T | F |
| T | F | F |
| T | T | T |

* **p ↔ q is true if both p and q are either true or false**.

**Logical Equivalence (p ≡ q)**

* **p ≡ q** means p and q always have the same truth value (i.e., p ↔ q is a tautology).
* Notation:
  + **p ≡ q** (Logical equivalence)
  + Sometimes written as **p ⇔ q**
* **Important Note**: **≡ is not a logical connective** because p ≡ q is not a compound proposition but rather a statement about equivalence.

**Notes on Truth Tables and Logical Operations**

**Table-1: Truth Table of a Tautology**

A **tautology** is a statement that is **always true**, regardless of the truth values of its components.  
Example: p∨¬p (Law of Excluded Middle)

|  |  |  |  |
| --- | --- | --- | --- |
| **p** | **¬p** | **p&¬p** | **p∨¬p** |
| T | F | F | T |
| F | T | F | T |

**Table-2: Truth Table for ¬(p∨q) and ¬p∧¬q**

* **De Morgan's Law** states that: ¬(p∨q) ≡ ¬p ∧ ¬q.
* This means **the negation of a disjunction** is equivalent to **the conjunction of the negations**.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **p** | **q** | **p∨q** | **¬(p∨q)** | **¬p** | **¬q** | **¬p∧¬q** |
| T | T | T | F | F | F | F |
| T | F | T | F | F | T | F |
| F | T | T | F | T | F | F |
| F | F | F | T | T | T | T |

**Table-3: Truth Table for ¬p∨q and p→q**

* The **implication** p→q is **false only if** p is **true** and q is **false**.
* ¬p∨q is **logically equivalent** to p→q.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **p** | **q** | **¬p** | **¬p∨q** | **p→q** |
| T | T | F | T | T |
| T | F | F | F | F |
| F | T | T | T | T |
| F | F | T | T | T |

**C/W-1: Truth Table for p∨(q∧r)p) and (p∨q)∧(p∨r).**

* **Distributive Law** states: p∨(q∧r)≡(p∨q)∧(p∨r)
* This means **distributing OR over AND** gives an equivalent proposition.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **p** | **q** | **r** | **q∧r** | **p∨(q∧r)** | **p∨q** | **p∨r** | **(p∨q)∧(p∨r)** |
| T | T | T | T | T | T | T | T |
| T | T | F | F | T | T | T | T |
| T | F | T | F | T | T | T | T |
| F | T | T | T | T | T | T | T |
| T | F | F | F | T | T | T | T |
| F | T | F | F | F | T | F | F |
| F | F | T | F | F | F | T | F |
| F | F | F | F | F | F | F | F |

Since **p∨(q∧r)≡(p∨q)∧(p∨r) always have the same truth values**, they are **logically equivalent**.

**Equivalence Laws in Logic**

**1. Identity Laws**

* **p∧T≡p**: The AND operation with **true** results in the same proposition.
* **p∨F≡p**: The OR operation with **false** results in the same proposition.

**2. Domination Laws**

* **p∨T≡T**: The OR operation with **true** always results in true.
* **p∧F≡F**: The AND operation with **false** always results in false.

**3. Idempotent Laws**

* **p∨p≡p**: ORing a proposition with itself results in the same proposition.
* **p∧p≡p**: ANDing a proposition with itself results in the same proposition.

**4. Commutative Laws**

* **p∨q≡q∨p**: The OR operation is commutative.
* **p∧q≡q∧p**: The AND operation is commutative.

**5. Associative Laws**

* **(p∨q)∨r≡p∨(q∨r)**: The OR operation is associative.
* **(p∧q)∧r≡p∧(q∧r)**: The AND operation is associative.

**6. Distributive Laws**

* **p∨(q∧r)≡(p∨q)∧(p∨r)** Distributes OR over AND.
* **p∧(q∨r)≡(p∧q)∨(p∧r)**: Distributes AND over OR.

**7. De Morgan’s Laws**

* **¬(p∧q)≡¬p∨¬q**: Negation of AND becomes OR of negations.
* **¬(p∨q)≡¬p∧¬q**: Negation of OR becomes AND of negations.

**8. Absorption Laws**

* **p∨(p∧q)≡p**: ORing a proposition with the AND of itself and another proposition results in the same proposition.
* **p∧(p∨q)≡p**: ANDing a proposition with the OR of itself and another proposition results in the same proposition.

**9. Negation Laws**

* **p∨¬p≡T:** A proposition ORed with its negation is always true.
* **p∧¬p≡F**: A proposition ANDed with its negation is always false.

**C/W-2: Truth Table for p∨q, ¬p∨¬q, and (p∨q)→(¬p∨¬q).**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **p** | **q** | **p∨q** | **¬p∨¬q** | **(p∨q)→(¬p∨¬q)** |
| T | T | T | F | F |
| T | F | T | T | T |
| F | T | T | T | T |
| F | F | F | T | T |

* The implication (p∨q)→(¬p∨¬q)(p \lor q) \rightarrow (\neg p \lor \neg q) is **false** when pp is true and qq is true, since both sides of the implication don't match.

**C/W-3: Truth Table for ¬(¬p∨¬q)\neg (\neg p \lor \neg q), and p∧(q∧r)p \land (q \land r)**

| **p** | **q** | **r** | **¬p\neg p** | **¬q\neg q** | **¬p∨¬q\neg p \lor \neg q** | **¬(¬p∨¬q)\neg (\neg p \lor \neg q)** | **p∧(q∧r)p \land (q \land r)** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T | T | T | F | F | F | T | T |
| T | T | F | F | F | F | T | F |
| T | F | T | F | T | T | F | T |
| T | F | F | F | T | T | F | F |
| F | T | T | T | F | T | F | F |
| F | T | F | T | F | T | F | F |
| F | F | T | T | T | T | F | F |
| F | F | F | T | T | T | F | F |

* **¬(¬p∨¬q)\neg (\neg p \lor \neg q)** is the negation of **De Morgan's Law**.
* **p∧(q∧r)p \land (q \land r)** evaluates as true only when **p**, **q**, and **r** are true.

**C/W-4: Truth Table for (p∧q)∨(q∧r)(p \land q) \lor (q \land r) and (p∧q)∨(q∧r)∨(r∧p)(p \land q) \lor (q \land r) \lor (r \land p)**

| **p** | **q** | **r** | **p∧qp \land q** | **q∧rq \land r** | **(p∧q)∨(q∧r)(p \land q) \lor (q \land r)** | **(p∧q)∨(q∧r)∨(r∧p)(p \land q) \lor (q \land r) \lor (r \land p)** |
| --- | --- | --- | --- | --- | --- | --- |
| T | T | T | T | T | T | T |
| T | T | F | T | F | T | T |
| T | F | T | F | F | F | T |
| T | F | F | F | F | F | T |
| F | T | T | F | T | T | T |
| F | T | F | F | F | F | F |
| F | F | T | F | F | F | F |
| F | F | F | F | F | F | F |

* The expressions are equivalent. They evaluate the same under all truth values of **p**, **q**, and **r**.

**C/W-5: Show that ¬(p∨(¬p∧q))\neg (p \lor (\neg p \land q)) and ¬p∧¬q\neg p \land \neg q are logically equivalent.**

LHS=~(p  (~p & q))

=~p& ~(~p & q)

=~p & [~(~p)  ~q]

=(~p & p)  (~p & ~q)

=F  (~p & ~q)

=(~p & ~q) F

=(~p & ~q) =RHS (Proved).

**C/W-6: Show that (p∧q)→(p∨q)(p \land q) \rightarrow (p \lor q) is a tautology.**

**Step 1: Create the truth table for (p∧q)→(p∨q)(p \land q) \rightarrow (p \lor q)**

| **p** | **q** | **p∧qp \land q** | **p∨qp \lor q** | **(p∧q)→(p∨q)(p \land q) \rightarrow (p \lor q)** |
| --- | --- | --- | --- | --- |
| T | T | T | T | T |
| T | F | F | T | T |
| F | T | F | T | T |
| F | F | F | F | T |

Since **(p∧q)→(p∨q)(p \land q) \rightarrow (p \lor q)** is true in all cases, it is a **tautology**.

**First Order Predicate Logic (FOPL) and its Role in AI**

**Introduction to AI Knowledge Representation**

Artificial Intelligence (AI) involves several methodologies and tools to represent and manipulate knowledge. One of the most fundamental and oldest methods used for knowledge representation in AI is **First Order Predicate Logic** (FOPL), also known as **Predicate Calculus**. This technique plays a critical role in AI, as it allows for the logical representation of knowledge that can be used by AI systems for reasoning and inference.

**The Evolution of Symbolic Logic**

The concept of using symbolic logic to represent knowledge isn't new. However, its practical application for representing and manipulating knowledge in computers became prominent only in the early 1960s when **Gilmore** demonstrated its potential. Since then, **FOPL** has been widely adopted in AI research and development, providing a formal framework for knowledge representation.

**Components of First Order Predicate Logic (FOPL)**

FOPL allows natural language statements to be translated into logical expressions. It uses a structured framework of **predicates**, **functions**, **variables**, **constants**, **quantifiers**, and **logical connectives** to form the building blocks of knowledge representation. These elements help in constructing valid logical statements that a computer can process and reason about.

* **Predicates**: These are symbols that represent properties or relationships between objects. For example, in the statement "John is a programmer," "is-programmer" would be the predicate.
* **Functions**: These map an object to another object (e.g., "father-of(x)" could represent the father of a person x).
* **Variables**: Represent placeholders for objects or entities (e.g., x in “All employees are programmers” can be any person).
* **Constants**: These refer to specific, named objects (e.g., "John" or "Raj").
* **Quantifiers**: These are symbols that express the scope of the statement (e.g., ∀ for "for all," ∃ for "there exists").
* **Logical Connectives**: Used to link predicates (e.g., AND (∧), OR (∨), NOT (¬), IMPLIES (→)).

**Translating English Sentences into FOPL**

In FOPL, natural language statements are converted into symbolic logic expressions. For instance, consider the sentence:

"All employees of the AI Software Company are programmers."

This could be represented in FOPL as:

(∀x)(AI-SOFTWARE-CO-EMPLOYEE(x)→PROGRAMMER(x))(\forall x) \left( \text{AI-SOFTWARE-CO-EMPLOYEE}(x) \rightarrow \text{PROGRAMMER}(x) \right)

Here’s the breakdown of the components:

* **∀x**: "For all x" (indicating that the statement applies to all entities x).
* **AI-SOFTWARE-CO-EMPLOYEE(x)**: Predicate that indicates "x is an employee of the AI Software Company."
* **PROGRAMMER(x)**: Predicate indicating "x is a programmer."
* **→**: Logical connective "implies" (if x is an employee of AI Software Company, then x is a programmer).

In this case, **x** represents a variable, which can be any employee in the company. The statement is a conditional one, asserting that if someone is an employee of the AI Software Company, they must also be a programmer.

**Using FOPL for Knowledge Representation and Inference**

Once knowledge is translated into FOPL, it can be stored in a **knowledge base**. From this knowledge base, AI systems can perform **inference** to deduce new facts or conclusions based on the existing knowledge.

For example, consider the following:

* We know that **Raj** is an employee of the AI Software Company:

AI-SOFTWARE-CO-EMPLOYEE(Raj)\text{AI-SOFTWARE-CO-EMPLOYEE}(Raj)

* Using the earlier FOPL statement about employees and programmers, we can now infer that:

PROGRAMMER(Raj)\text{PROGRAMMER}(Raj)

This means that since Raj is an employee of the AI Software Company, FOPL allows us to conclude that Raj must also be a programmer, based on the given logical rule.

**Propositional Logic (PL) and Valid Statements**

**Introduction to Propositional Logic (PL)**

Propositional Logic (PL), also known as **Sentential Logic**, is a formal system in logic where statements (propositions) are represented using symbols and evaluated based on logical rules. The validity of statements in PL is determined by **propositional syntax** rules, which ensure that each statement follows a structured logical form.

**Understanding Propositions in PL**

A **proposition** is a simple, atomic sentence that expresses a **definite** statement. In Propositional Logic:

* Each proposition can only have **two possible truth values**: **True (T) or False (F)**.
* It **cannot** have uncertain, vague, or in-between values.
* The terms **formula** or **well-formed formula (WFF)** may be used interchangeably with "sentences" in PL.

**Examples of Simple Propositions**

The following are examples of valid propositions because they express definite statements that are either **true or false**:

1. **"It is raining."**
   * This is a proposition because it asserts a fact that can be either true or false at a given time.
2. **"Raymond’s car is painted white."**
   * This statement can be **verified** as either true or false, making it a valid proposition.
3. **"Sanjay and Sharmila have two children."**
   * This is a definite claim that can be confirmed or disproven, making it a valid proposition.
4. **"In winter, sometimes the temperature at Montreal becomes -20°C."**
   * Although it includes the word "sometimes," this still qualifies as a proposition because it describes an objective fact that can be tested.
5. **"Times Square in Manhattan is a famous place in New York."**
   * This is a factual statement and can be **evaluated as true or false** based on real-world knowledge.
6. **"People live on the moon."**
   * Even though this statement is false, it is still a valid proposition in PL because it makes a clear claim that can be **proven false**.

**Properties of Propositions in PL**

* **Atomic Propositions**: A **single** statement that expresses a basic fact (e.g., "It is raining.").
* **Compound Propositions**: Statements formed using **logical connectives** such as AND (∧), OR (∨), NOT (¬), and IMPLIES (→) (e.g., "It is raining AND it is cold.").

**Compound Propositions in Propositional Logic (PL)**

**Introduction to Compound Propositions**

In **Propositional Logic (PL)**, compound propositions are formed by **combining atomic formulas** using **logical connectives**. These connectives allow the construction of **more complex statements** that express logical relationships between individual propositions.

**Logical Connectives in PL**

The fundamental logical connectives used to form compound propositions are:

* **NOT (¬ or ~)** → Negation
* **AND (∧ or &)** → Conjunction
* **OR (∨)** → Disjunction (inclusive OR)
* **IMPLICATION (→)** → Conditional (If… then…)
* **BICONDITIONAL (↔)** → If and only if (IFF)

**Examples of Compound Propositions**

Below are examples of compound statements formed using logical connectives:

1. **"It is raining and the wind is blowing."**
   * Represented as: **(R ∧ B)**
   * Where:
     + **R = "It is raining"**
     + **B = "The wind is blowing"**
   * The **AND (∧)** operator ensures that the entire statement is **true only if both R and B are true**.
2. **"If you study hard, you will do well in the semester exam."**
   * Represented as: **(S → D)**
   * Where:
     + **S = "You study hard"**
     + **D = "You do well in the semester exam"**
   * The **IMPLICATION (→)** operator means that if **S is true, then D must also be true**.
3. **"The sum of 10 and 20 is not 50."**
   * Represented as: **¬P**
   * Where:
     + **P = "The sum of 10 and 20 is 50"**
   * The **NOT (¬)** operator negates the truth value of P, meaning that if P is true, ¬P is false and vice versa.
4. **"The color of the moon is black or it is not."**
   * Represented as: **(M ∨ ¬M)**
   * Where:
     + **M = "The color of the moon is black"**
   * This is an example of the **Law of the Excluded Middle**, stating that **either a statement is true, or its negation is true**.

**Representation of Propositions Using Symbols**

In **Propositional Logic**, capital letters (e.g., P, Q, R) **represent atomic propositions**, and **special symbols (T and F)** are used to denote **true and false** values.

**Example 1: Translating a Sentence to a Logical Formula**

**Sentence:** *"It is raining and the wind is blowing."*

* **Symbolic representation:** **(R ∧ B)**
* Where:
  + **R = "It is raining"**
  + **B = "The wind is blowing"**
  + **∧ (AND) ensures both must be true for the statement to be true.**

**Variation using OR (Inclusive Disjunction)**  
**Sentence:** *"It is raining or the wind is blowing or both."*

* **Symbolic representation:** **(R ∨ B)**
* The **OR (∨) operator represents inclusive disjunction**, meaning **either R, B, or both can be true for the statement to hold true.**

**Syntax of Propositional Logic (PL)**

The syntax of **Propositional Logic** is **recursively defined**, meaning that formulas are built from smaller valid formulas using predefined rules.

**Basic Rules of PL Syntax:**

1. **T (true) and F (false) are always valid formulas.**
2. If **P** and **Q** are valid formulas, then the following are also valid formulas:
   * **(P)** → Parentheses indicate grouping.
   * **(P ∧ Q)** → Conjunction (AND).
   * **(P ∨ Q)** → Disjunction (OR).
   * **(P → Q)** → Implication (If P, then Q).
   * **(P ↔ Q)** → Biconditional (P if and only if Q).

**Example of a Complex Compound Formula**

((P∧(¬Q∨R))→(Q→S))((P \wedge (\neg Q \vee R)) \rightarrow (Q \rightarrow S))

**Breakdown of the formula:**

* **P ∧ (¬Q ∨ R)** → A conjunction where P is ANDed with a disjunction (¬Q OR R).
* **(Q → S)** → An implication stating that if Q is true, then S must also be true.
* **The entire formula represents an implication**: If **P AND (¬Q OR R) holds, then (Q → S) must also hold**.