Propositional calculus (or logic) is the study of the logical relationship between objects called propositions and forms the basis of all mathematical reasoning and all automated reasoning.

A proposition is a statement that is either true or false, but not both (we usually denote a proposition by letters; \( p, q, r, s, \ldots \)).

The value of a proposition is called its truth value; denoted by \( T \) or 1 if it is true and \( F \) or 0 if it is false.

Opinions, interrogative and imperative sentences are not propositions.

Truth table:

<table>
<thead>
<tr>
<th>( p )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
</table>

Example (Propositions)

- Today is Monday.
- The derivative of \( \sin x \) is \( \cos x \).
- Every even number has at least two factors.

Example (Not Propositions)

- C++ is the best language.
- When is the pretest?
- Do your homework.

Connectives are used to create a compound proposition from two or more other propositions.

- Negation (denoted \( \neg \) or !)
- And (denoted \( \land \) or Logical Conjunction
- Or (denoted \( \lor \) or Logical Disjunction
- Exclusive Or (XOR, denoted \( \oplus \))
- Implication (denoted \( \rightarrow \))
- Biconditional; “if and only if” (denoted \( \leftrightarrow \))
Negation

A proposition can be negated. This is also a proposition. We usually denote the negation of a proposition \( p \) by \( \neg p \).

Example (Negated Propositions)

- Today is not Monday.
- It is not the case that today is Monday.
- It is not the case that the derivative of \( \sin x \) is \( \cos x \).

Truth table:

<table>
<thead>
<tr>
<th>( p )</th>
<th>( \neg p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Logical And

The logical connective AND is true only if both of the propositions are true. It is also referred to as a conjunction.

Example (Logical Connective: AND)

- It is raining and it is warm.
- \((2 + 3 = 5) \land (\sqrt{2} < 2)\)
- Schrödinger’s cat is dead and Schrödinger’s cat is not dead.

Truth table:

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( p \land q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Logical Or

The logical disjunction (or logical or) is true if one or both of the propositions are true.

Example (Logical Connective: OR)

- It is raining or it is the second day of lecture.
- \((2 + 2 = 5) \lor (\sqrt{2} < 2)\)
- You may have cake or ice cream.\(^\dagger\)

Truth table:

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( p \lor q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\(^\dagger\)Can I have both?

Implications I

Definition

Let \( p \) and \( q \) be propositions. The implication

\[ p \rightarrow q \]

is the proposition that is false when \( p \) is true and \( q \) is false and true otherwise.

Here, \( p \) is called the “hypothesis” (or “antecedent” or “premise”) and \( q \) is called the “conclusion” or “consequence”.

Truth table:

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( p \rightarrow q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Implications II

The implication \( p \rightarrow q \) can be equivalently read as

- if \( p \) then \( q \)
- \( p \) implies \( q \)
- if \( p, q \)
- \( p \) only if \( q \)
- \( q \) if \( p \)
- \( q \) when \( p \)
- \( q \) whenever \( p \)
- \( p \) is a sufficient condition for \( q \) (\( p \) is sufficient for \( q \))
- \( q \) is a necessary condition for \( p \) (\( q \) is necessary for \( p \))
- \( q \) follows from \( p \)
Examples

Example

▶ If you buy your air ticket in advance, it is cheaper.
▶ If $x$ is a real number, then $x^2 \geq 0$.
▶ If it rains, the grass gets wet.
▶ If the sprinklers operate, the grass gets wet.
▶ If $2 + 2 = 5$ then all unicorns are pink.

Biconditional

Definition

The biconditional $p \leftrightarrow q$ is the proposition that is true when $p$ and $q$ have the same truth values. It is false otherwise.

Note that it is equivalent to $(p \rightarrow q) \land (q \rightarrow p)$

Truth table:

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \rightarrow q$</th>
<th>$q \rightarrow p$</th>
<th>$p \leftrightarrow q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Exercise

Which of the following biconditionals is true?

▶ $x^2 + y^2 = 0$ if and only if $x = 0$ and $y = 0$
  true: both implications hold.
▶ $2 + 2 = 4$ if and only if $\sqrt{2} < 2$
  true: for the same reason above.
▶ $x^2 \geq 0$ if and only if $x \geq 0$.
  false: The converse holds. That is, "if $x \geq 0$ then $x^2 \geq 0$". However, the implication is false; consider $x = -1$. Then the hypothesis is true, $(−1)^2 = 1^2 \geq 0$ but the conclusion fails.

Examples

$p \leftrightarrow q$ can be equivalently read as

▶ $p$ if and only if $q$
▶ $p$ is necessary and sufficient for $q$
▶ if $p$ then $q$, ... $x^2$ is positive.
▶ The alarm goes off iff a burglar breaks in.
▶ You may have pudding if and only if you eat your meat.

Exercise

Which of the following implications is true?

▶ If $-1$ is a positive number, then $2 + 2 = 5$.
  true: the hypothesis is obviously false, thus no matter what the conclusion, the implication holds.
▶ If $-1$ is a positive number, then $2 + 2 = 4$.
  true: for the same reason as above.
▶ If $\sin x = 0$ then $x = 0$.
  false: $x$ can be any multiple of $\pi$; i.e. if we let $x = 2\pi$ then clearly $\sin x = 0$, but $x \neq 0$. The implication "if $\sin x = 0$ then $x = k\pi$ for some integer $k$" is true.

Converse, Contrapositive, Inverse

Consider the proposition $p \rightarrow q$:

▶ Its converse is the proposition $q \rightarrow p$.
▶ Its inverse is the proposition $\neg p \rightarrow \neg q$.
▶ Its contrapositive is the proposition $\neg q \rightarrow \neg p$. 
Truth Tables I

Truth Tables are used to show the relationship between the truth values of individual propositions and the compound propositions based on them.

<p>|</p>
<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>p ∧ q</th>
<th>p ∨ q</th>
<th>p ⊕ q</th>
<th>p → q</th>
<th>p ↔ q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table: Truth Table for Logical Conjunction, Disjunction, Exclusive Or, and Implication

Constructing Truth Tables

Construct the Truth Table for the following compound proposition.

\[(p ∧ q) ∨ ¬q\]

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>p ∧ q</th>
<th>¬q</th>
<th>(p ∧ q) ∨ ¬q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Precedence of Logical Operators

Just as in arithmetic, an ordering must be imposed on the use of logical operators in compound propositions.

Of course, parentheses can be used to make operators disambiguos:

\[¬p \lor q \land ¬r ≡ (¬p) \lor (q \land (¬r))\]

But to avoid using unnecessary parentheses, we define the following precedences:

1. (¬) Negation
2. (∧) Conjunction
3. (∨) Disjunction
4. (→) Implication
5. (↔) Biconditional

Usefulness of Logic

Logic is more precise than natural language:

- You may have cake or ice cream.
  - Can I have both?
- If you buy your air ticket in advance, it is cheaper.
  - Are there or not cheap last-minute tickets?

For this reason, logic is used for hardware and software specification.

Given a set of logic statements, one can decide whether or not they are satisfiable (i.e., consistent), although this is a costly process...

Bitwise Operations

Computers represent information as bits (binary digits).

A bit string is a sequence of bits, the length of the string is the number of bits in the string.

Logical connectives can be applied to bit strings (of equal length). To do this, we simply apply the connective rules to each bit of the string:

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>0110 1010 1101</td>
</tr>
<tr>
<td>0101 0010 1111</td>
</tr>
<tr>
<td>0111 1010 1111 bitwise OR</td>
</tr>
<tr>
<td>0100 0010 1101 bitwise AND</td>
</tr>
<tr>
<td>0011 1000 0010 bitwise XOR</td>
</tr>
</tbody>
</table>

A Boolean variable is a variable that can have value 0 or 1.

Logic in Theoretical Computer Science

SAT

What is SAT? SAT is the problem of determining whether or not a sentence in propositional logic (PL) is satisfiable. Characterizing SAT as an NP-complete problem is at the foundation of Theoretical Computer Science.

Defining SAT

- **Given**: a PL sentence.
- **Question**: Determine whether it is satisfiable or not.

What is a PL sentence? What does satisfiable mean?
Logic in Theoretical Computer Science

A sentence in PL

- A sentence in PL is a conjunction of clauses
- A clause is a disjunction of literals
- A literal is a term or its negation
- A term is a Boolean variable (or proposition)

Example: \((a \lor b \lor \neg c \lor \neg d) \land (\neg b \lor c) \land (\neg a \lor c \lor d)\)

A sentence in PL is a satisfiable iff we can assign truth values to the Boolean variables such that the sentence evaluates to true (i.e., holds).

Logic in Programming

Programming Example I

Say you need to define a conditional statement as follows: “Increment \(x\) if all of the following conditions hold: \(x > 0\), \(x < 10\) and \(x = 10\).”

You may try:

\[\text{if}(0 < x < 10 \text{ OR } x = 10) \ x++;\]

But is not valid in C++ or Java. How can you modify this statement by using a logical equivalence?

Answer:

\[\text{if}(x>0 \text{ AND } x<=10) \ x++;\]

Propositional Equivalences

Introduction

To manipulate a set of statements (here, logical propositions) for the sake of mathematical argumentation, an important step is to replace one statement with another equivalent statement (i.e., with the same truth value).

Below, we discuss:

- Terminology
- Establishing logical equivalences using truth tables
- Establishing logical equivalences using known laws (of logical equivalences)

Terminology

Tautologies, Contradictions, Contingencies

**Definition**

- A compound proposition that is always true, no matter what the truth values of the propositions that occur in it is called a tautology.
- A compound proposition that is always false is called a contradiction.
- Finally, a proposition that is neither a tautology nor a contradiction is called a contingency.

**Example**

A simple tautology is \(p \lor \neg p\)
A simple contradiction is \(p \land \neg p\)

Logical Equivalences

**Definition**

Propositions \(p\) and \(q\) are logically equivalent if \(p \leftrightarrow q\) is a tautology.

Informally, \(p\) and \(q\) are logically equivalent if whenever \(p\) is true, \(q\) is true, and vice versa.

Notation \(p \equiv q\) (“\(p\) is equivalent to \(q\)”), \(p \leftrightarrow q\), \(p \Leftrightarrow q\), \(p \iff q\).

Alert: \(\equiv\) is not a logical connective.
Example

Are and \( p \rightarrow q \) and \( \neg p \lor q \) logically equivalent?

To find out, we construct the truth tables for each:

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( p \rightarrow q )</th>
<th>( \neg p )</th>
<th>( \neg p \lor q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

The two columns in the truth table are identical, thus we conclude that

\( p \rightarrow q \equiv \neg p \lor q \)
Using Logical Equivalences
Example 3

Show that

\[ \neg(q \rightarrow p) \lor (p \land q) \iff q \]

\[ \neg(q \rightarrow p) \lor (p \land q) \]
\[ \iff (\neg q \lor p) \land (p \land q) \quad \text{Implication Law} \]
\[ \iff (q \land \neg p) \lor (q \land p) \quad \text{De Morgan’s & Double Negation} \]
\[ \iff q \land (\neg p \lor p) \quad \text{Commutative Law} \]
\[ \iff q \land 1 \quad \text{Distributive Law} \]
\[ \iff q \quad \text{Identity Law} \]

Logic In Programming
Programming Example II Revisited

Recall the loop:

\[ \text{while}((i < \text{size}) \land (A[i] > 10 \lor A[i] < 0)) \]

Now, using logical equivalences, simplify it.

Logic In Programming
Programming Example II Revisited
Answer: Use De Morgan’s Law and Distributivity.

\[ \text{while}(i < \text{size} \land (A[i] > 10 \lor A[i] < 0)) \]

Notice the ranges of all four conditions on A[i]; they can be merged and we can further simplify it to:

\[ \text{while}(i < \text{size} \land (A[i] \geq 10 \lor A[i] \leq 0)) \]

Programming Pitfall Note

In C, C++ and Java, applying the commutative law is not such a good idea. These languages (compiler dependent) sometimes use “short-circuiting” for efficiency (at the machine level). For example, consider accessing an integer array A of size n.

\[ \text{if}(i < n \land A[i] == 0) \quad \text{i++; } \]

is not equivalent to

\[ \text{if}(A[i] == 0 \land i < n) \quad \text{i++;} \]