Outline

1 Propositional Logic
   • Definition
   • Operators
   • Examples
...Propositions

Defn: A *proposition* is a statement that is either *true* or *false* but not both.

- “Joseph Stalin is a Catholic saint.” and “Barrack Obama is President of the United States.” are propositions.
- “Why buy a parrot?” and “x is an even number.” are *not* propositions. Why not?
Defn: A *proposition* is a statement that is either *true* or *false* but not both.

- We use variables such as $p, q, r$ to represent entire propositions.
...Propositions

Defn: A proposition is a statement that is either true or false but not both.

- We use variables such as $p$, $q$, $r$ to represent entire propositions.
- What types of values do $p$, $q$, $r$ take on?
Defn: A *proposition* is a statement that is either *true* or *false* but not both.

- We use variables such as $p, q, r$ to represent entire propositions.
- What types of values do $p, q, r$ take on?
- What *truth values* do $p, q, r$ take on?
Logical Operators

Definition

Just as there are operators on integer values:

- **unary** \(-17, +4\)
- **binary** \(11 \times 2, 2 + 4, -5/2\)
Logical Operators

Definition

Just as there are operators on integer values:

unary $-17, +4$

binary $11 \times 2, 2 + 4, -5/2$

There are also operators on boolean (truth) values:

unary $\neg r$

binary $a \land b, p \lor q, r \implies s, y \oplus z$
Logical Operators

Negation

Defn: The *negation* of a proposition is also a proposition.

\[ p: \text{It is raining.} \]

\[ \neg p: \text{It is not the case that it is raining.} \]

\[ \neg p: \text{It is not raining.} \]

Note: \( \neg p \) also can be written \( \overline{p} \).

**BE CAREFUL WRITING NEGATIONS OF ENGLISH SENTENCES.** The truth value of \( \neg p \) depends on the truth value of \( p \):

<table>
<thead>
<tr>
<th>( p )</th>
<th>( \neg p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>
Logical Operators

Conjunction

Defn: The *conjunction* of two propositions is also a proposition.

\[ p: \text{ It is raining.} \]
\[ q: \text{ It is cold.} \]
\[ p \land q: \text{ It is raining and it is cold.} \]

The truth value of \( p \land q \) depends on the truth values of \( p \) and \( q \):

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( p \land q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Logical Operators

Disjunction

Defn: The *disjunction* of two propositions is also a proposition.

\[ p: \text{It is raining.} \]
\[ q: \text{It is cold.} \]
\[ p \lor q: \text{It is raining or it is cold.} \]

The truth value of \( p \lor q \) depends on the truth values of \( p \) or \( q \):

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( p \lor q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Logical Operators

Exclusive Or

The disjunction or $\lor$ operator is the “inclusive or” meaning one, the other, or both are true.

Defn: The exclusive or of two propositions is a proposition.

$p$: It is raining.
$q$: It is cold.

$p \oplus q$: It is raining or it is cold but not both.

The truth value of $p \oplus q$ depends on the truth values of $p$ or $q$:

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \oplus q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>
Defn: A conditional (or implication) combining two propositions is also a proposition.

\[ p \rightarrow q : \text{If it has gone viral then it is famous.} \]

The truth value of \( p \rightarrow q \) depends on the truth values of \( p \) and \( q \):

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( p \rightarrow q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

\( p \rightarrow q \): \( p \) is the antecedent, \( q \) is the consequent.
Logical Operators

Conditional

A conditional is a rule. It is not an if...then statement in a programming language. We are interested in the truth of the rule. There are many ways to express implication in English:

"if $p$ then $q$"  
"if $p$, $q$"  
"$p$ is sufficient for $q$"  
"$q$ if $p$"  
"$q$ when $p$"  
"a necessary condition for $p$ is $q$"  
"$q$ unless $\neg p$"

"$p$ implies $q$"  
"$p$ only if $q$"  
"a sufficient condition for $q$ is $p$"  
"$q$ whenever $p$"  
"$q$ is necessary for $p$"  
"$q$ follows from $p$"

**NOTE:** The antecedent being true is sufficient for the consequent to be necessarily true.
Defn: A *biconditional* combining two propositions is also a proposition.

\[ p: \text{It has gone viral.} \]

\[ q: \text{It is famous} \]

\[ p \iff q: \text{It has gone viral if and only if it is famous.} \]

The truth value of \( p \iff q \) depends on the truth values of \( p \) and \( q \):

\[
\begin{array}{c|c|c}
 p & q & p \iff q \\
 F & F & T \\
 F & T & F \\
 T & F & F \\
 T & T & T \\
\end{array}
\]

\( p \iff q \) is the same as \( p \rightarrow q \land q \rightarrow p \).
Logical Operators

Related Conditionals

Defn: The *converse* of $p \rightarrow q$ is $q \rightarrow p$.
Defn: The *inverse* of $p \rightarrow q$ is $\neg p \rightarrow \neg q$.
Defn: The *contrapositive* of $p \rightarrow q$ is $\neg q \rightarrow \neg p$. 
Logical Operators

Summary

- Unary Boolean operator: \( \neg \)
- Binary Boolean operators: \( \land, \lor, \oplus, \rightarrow, \leftrightarrow \)
- Simple proposition: \( p, q, r, \ldots \)
- Compound propositions: \( p \land q, (p \lor \neg q) \rightarrow (p \land q) \)
Truth Tables

- A truth table can be drawn for any proposition, simple or compound.
- There must be a line in the table for every combination of truth values for the simple propositions in the proposition.
Truth Tables

- A truth table can be drawn for any proposition, simple or compound.
- There must be a line in the table for every combination of truth values for the simple propositions in the proposition.

\[ p \lor (q \rightarrow r) \]
Truth Tables

- A truth table can be drawn for any proposition, simple or compound.
- There must be a line in the table for every combination of truth values for the simple propositions in the proposition.

\[ p \lor (q \rightarrow r) \]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>q</td>
<td>r</td>
<td>q → r</td>
<td>p \lor (q → r)</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>-------</td>
<td>----------------</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Truth Tables

- A truth table can be drawn for any proposition, simple or compound.
- There must be a line in the table for every combination of truth values for the simple propositions in the proposition.

\[ p \lor (q \rightarrow r) \]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>q</td>
<td>r</td>
<td>q → r</td>
<td>p ∨ (q → r)</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Can you rewrite the expression as an equivalent negated three-way conjunction?