Equivalence and Implication

Alice E. Fischer

CSCI 1166 Discrete Mathematics for Computing
February 14, 2018
1. Properties of Propositions
   - Logical Equivalence
   - Contradictions and Tautologies

2. Logical Equivalences

3. Practice with Boolean Operators and Algebra

4. Implication
   - Necessary and Sufficient Conditions
   - Inside out and Backwards

5. Summary
Properties of Propositions

Logical Equivalence

Contradictions and Tautologies
Logical Equivalence: \( \equiv \)

Two propositions are **logically equivalent** \( (\equiv) \) if they have the same outcomes under all input conditions.

We can determine whether two propositions are equivalent \( (\equiv) \) by comparing the last column in the truth tables for the two statements.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>( P \lor Q )</th>
<th>( Q \lor P )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

For example, The column for \( P \lor Q \) is the same as the column for \( Q \lor P \), so the two propositions are equivalent.
Example: The Supervisor’s Critique

Do you remember John and his elevator design from last time? John’s supervisor looked at his logic and told him “It’s too complex, O is just the opposite of M”. Simplify! So John thought a bit and added a column with a simpler formula to his truth table.

<table>
<thead>
<tr>
<th>$O$</th>
<th>$M$</th>
<th>$\sim O$</th>
<th>$\sim M$</th>
<th>$O \land \sim M$</th>
<th>$\sim O \land M$</th>
<th>$(O \land \sim M) \lor (\sim O \land M)$</th>
<th>$O \oplus M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
</tr>
</tbody>
</table>

Indeed, the boss was right. They are equivalent!
A contradiction is a proposition that is always false, no matter what the input truth values are.

For example, this is a contradiction:

An odd number, $X$, equals $2 \times$ some even number, $Y$.

And this is a contradiction: $P \land \sim P$

Let $P$ be All dogs are black.

All dogs are black and not (all dogs are black).

A contradiction can be symbolized by $\mathbf{c}$. 
A **tautology** is a proposition that **is always true**, no matter what the input truth values are.

For example:

Let \( Q \) be the proposition *Today is Tuesday*.

Then this is a tautology: \( Q \lor \sim Q \)

*Today is Tuesday or today is not Tuesday.*

A **tautology** can be symbolized by \( t \).
A set of rules for logical calculation can be proven from the definitions using truth tables. They will be presented in the following slides.

These laws are analogs of the laws for algebra and for sets that you already know. They are essential in making logical proofs and several are also important in programming.

The following slides define the laws for symbolic logic. The corresponding set-theory law(s) will be given at the bottom of each slide.
1. Commutative Laws

Both and and or are commutative, that is, the left and right sides can be reversed without changing the meaning of the proposition.

\[ P \land Q \equiv Q \land P \]
\[ P \lor Q \equiv Q \lor P \]

This is important in programming.

Set theory: \( \cap \) and \( \cup \) are commutative. Given sets \( A \) and \( B \),
\[ A \cap B \equiv B \cap A \]
\[ A \cup B \equiv B \cup A \]
2. Associative Laws

Both \textbf{and} and \textbf{or} are associative operators, that is, you may parenthesize the expression any way you wish without changing its meaning.

\[(P \land Q) \land R \equiv P \land (Q \land R)\]
\[(P \lor Q) \lor R \equiv P \lor (Q \lor R)\]

This is important in programming.

Set theory: \texttt{\&} and \texttt{\lor} are associative. Given sets \(P\), \(Q\), and \(R\),
\[(P \& Q) \& R \equiv P \& (Q \& R)\]
\[(P \lor Q) \lor R \equiv P \lor (Q \lor R)\]
3. Distributive Laws

Both \textit{and} and \textit{or} can be distributed over the other operator.

\[ P \land (Q \lor R) \equiv (P \land Q) \lor (P \land R) \]
\[ P \lor (Q \land R) \equiv (P \lor Q) \land (P \lor R) \]

This is important in programming.

Set theory: \texttt{\&} and \texttt{\lor} can be distributed over each other.

Given sets \textit{P}, \textit{Q}, and \textit{R},
\[ P \cap (Q \cup R) \equiv (P \cap Q) \cup (P \cap R) \]
\[ P \cup (Q \cap R) \equiv (P \cup Q) \cap (P \cup R) \]
4. Identity Laws

The identity laws are based on the definitions of and and or.

\[ P \land t \equiv P \]
\[ P \lor c \equiv P \]

Set Theory: The set identity laws are based on definitions of the Universal set and the empty set.

Given set \( A \),

\[ A \cap U \equiv A \]
\[ A \cup \emptyset \equiv A \]
5. Negation Laws

The negation laws are based on the definitions of \textit{and}, \textit{or}, and \textit{not}.

\[ P \lor \sim P \equiv t \]
\[ P \land \sim P \equiv c \]

Set Theory: Complement Laws
The set complement laws are based on definitions of the Universal set, empty set, and complement.
Given set \( A \),
\[ A \cup A^c \equiv U \]
\[ A \cap A^c \equiv \emptyset \]
6. Double Negative Law

The double-negative law is based on the definitions of not.

\[ \sim (\sim P) \equiv P \]

Set Theory: Double Complement Law
This is based on the definitions of complement.
Given set \( A \),
\[ (A^c)^c \equiv A \]
7. Idempotent Laws

and, or are idempotent, meaning that the trivial computations have no effect and can be repeated without changing the meaning of the expression.

\[
P \land P \equiv P \\
P \lor P \equiv P
\]

Set Theory: intersection and union are idempotent.

Given set \( A \),
\[
A \cap A \equiv A \\
A \cup A \equiv A
\]
8. Universal Bound Laws

A contradiction \textbf{and} anything is a contradiction. A tautology \textbf{or} anything is a tautology.

\begin{align*}
P \land c & \equiv c \\
P \lor t & \equiv t
\end{align*}

The shortcut evaluation of logical operators in C is based on these.

Set Theory: The empty set is as small as you can get and the universal set is as large as you can get. Given set $A$,

\begin{align*}
A \cap \emptyset & \equiv \emptyset \\
A \cup U & \equiv U
\end{align*}
9. DeMorgan’s Laws

To distribute not over and, change the and to or.
To distribute not over or, change the or to and.

\[
\sim (P \land Q) \equiv \sim P \lor \sim Q \\
\sim (P \lor Q) \equiv \sim P \land \sim Q
\]

These laws are very important in programming and in databases.

Set Theory:
To distribute complement over \( \cap \), change the \( \cap \) to \( \cup \).
To distribute complement over \( \cup \), change the \( \cup \) to \( \cap \).
Given sets \( A, B \),
\[
(A \cap B)^c \equiv A^c \cup B^c \\
(A \cup B)^c \equiv A^c \cap B^c
\]
10. Absorption Laws

A truth table can be used to verify these absorptions.

\[ P \lor (P \land Q) \equiv P \]
\[ P \land (P \lor Q) \equiv P \]

These laws are important in proofs.

Set Theory:
Given sets \( A, B \),
\[ A \cup (A \cap B) \equiv A \quad \text{That is, } A \cup \text{ a subset of } A \text{ is still } A. \]
\[ A \cap (A \cup B) \equiv A \quad \text{That is, } A \cap \text{ a superset of } A \text{ is still } A. \]
11. Negations of $t$ and $c$

The negation of a tautology is a contradiction and vice versa.

$$\sim t \equiv c$$
$$\sim c \equiv t$$

Set Theory:
The complement of the Universal set is the empty set, and vice versa.

$$U^c \equiv \emptyset$$
$$\emptyset^c \equiv U$$
Practice

This section contains three logical expressions to simplified by applying the rules, and one law to prove using truth tables.
1. Proving DeMorgan’s Law

Construct a truth table to prove that $\sim(P \land Q) \equiv \sim P \lor \sim Q$

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$P \land Q$</th>
<th>$\sim(P \land Q)$</th>
<th>$\sim P$</th>
<th>$\sim Q$</th>
<th>$(\sim P \lor \sim Q)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
</tbody>
</table>

Compare the contents of the fourth and seventh columns. They are the same, so the two expressions are equivalent.
2. Two small messes to simplify.

\[
(P \land \sim Q) \lor (P \land Q)
\]
Initial expression.

\[
P \land (\sim Q \lor Q)
\]
Use the distributive law, backwards.

\[
P \land (t)
\]
Use negation law.

\[
P
\]
Use identity law.

\[
(P \land \sim Q) \lor (P \land P)
\]
Distributive law.

\[
(P \land \sim Q) \lor P
\]
Idempotent law

\[
P \lor (P \land \sim Q)
\]
Commutative law

\[
P
\]
Absorption law.
3. A big mess.

\[
\begin{align*}
(P \land (\sim(\sim P \lor Q))) & \lor (P \land Q) & \text{Initial expression} \\
(P \land (\sim\sim P) \land \sim Q) & \lor (P \land Q) & \text{DeMorgan’s law.} \\
(P \land (P \land \sim Q)) & \lor (P \land Q) & \text{Double negative law.} \\
((P \land P) \land \sim Q) & \lor (P \land Q) & \text{Associative law.} \\
(P \land \sim Q) & \lor (P \land Q) & \text{Idempotent law.} \\
(P \land (\sim Q \lor Q)) & \text{Distributive law, backwards.} \\
(P \land (t)) & \text{Negation law.} \\
P & \text{Identity law.}
\end{align*}
\]
Implication

→ and its Truth Table

Necessary and Sufficient Conditions
Inside out and Backwards
The symbol \( \rightarrow \) is called if...then in the textbook. It is more traditional to call it implies.

The proposition \( P \rightarrow Q \) is read \( P \) implies \( Q \) or if \( P \), then \( Q \).

We say that \( P \) is the premise and \( Q \) is the conclusion.

The truth table for \( P \rightarrow Q \) is:

<table>
<thead>
<tr>
<th>( P )</th>
<th>( Q )</th>
<th>( P \rightarrow Q )</th>
<th>( \sim P )</th>
<th>( \sim P \lor Q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td>( T )</td>
<td>( T )</td>
<td>( F )</td>
<td>( T )</td>
</tr>
<tr>
<td>( T )</td>
<td>( F )</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>( F )</td>
<td>( F )</td>
<td>( T )</td>
<td>( T )</td>
<td>( T )</td>
</tr>
</tbody>
</table>

It is also true that \( P \rightarrow Q \equiv \sim P \lor Q \).
Implication Can be Confusing.

- Sometimes, but not always, $P \rightarrow Q$ means that $P$ causes $Q$: 
  *I didn’t increment my loop variable so my program had an infinite loop.*

- Sometimes $P \rightarrow Q$ is used to define a term: 
  *If a creature is bipedal, it has two feet.*

- The conclusion can be true independently of the premise: 
  *If the moon is made of green cheese then UNH is a university.* This is often called a **vacuous truth**.

- The implication is true when both the premise and the conclusion are false. *if $0 = 1$ then $1 = 2$.*

A truth table is the most reliable way to figure out the correct interpretation of an implication.
If and Only If \( \iff \)

\( P \implies Q \) is a **conditional statement** that can be read *if \( P \) then \( Q \).*

Now suppose that \( P \implies Q \) and also \( Q \implies P \).

Then we write \( P \iff Q \), which can be read *\( P \) if and only if \( Q \).*

This statement is called a **biconditional**.

\[
\begin{array}{c|c|c|c|c}
P & Q & P \implies Q & Q \implies P & P \iff Q \\
T & T & T & T & T \\
T & F & F & T & F \\
F & T & T & F & F \\
F & F & T & T & T \\
\end{array}
\]
A **sufficient condition** is a premise or set of premises that are enough, without any more premises, to prove the conclusion.

If \( C \) is a sufficient condition for \( S \), then \( C \rightarrow S \).

A **necessary condition** is a premise that *must* be true in order for the conclusion to be true.

If \( C \) is a necessary condition for \( S \), then \( \sim C \rightarrow \sim S \).

If a condition \( C \) is both necessary and sufficient for \( S \), then \( C \leftrightarrow S \).
There are four ways to “reverse” the proposition $P \rightarrow Q$:

- The **contrapositive** is $\sim Q \rightarrow \sim P$.
- The **converse** is $Q \rightarrow P$.
- The **inverse** is $\sim P \rightarrow \sim Q$.
- The **negation** is $\sim(P \rightarrow Q)$, which is equivalent to $P \land \sim Q$.

A proposition and its contrapositive are equivalent.

A proposition is *not* equivalent to its converse, inverse, or negation. For example, “If it is a bee, it can fly” is *not* $\equiv$ to “If it can fly, it is a bee”.
The four “opposites” of $P \rightarrow Q$:

- A conditional is equivalent to its contrapositive.
- The inverse is equivalent to the converse and has the opposite truth value of the proposition.

<table>
<thead>
<tr>
<th>Conditional</th>
<th>Converse</th>
<th>Inverse</th>
<th>Contrapositive</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \rightarrow Q$</td>
<td>$Q \rightarrow P$</td>
<td>$\sim P \rightarrow \sim Q$</td>
<td>$\sim Q \rightarrow \sim P$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$\sim P$</th>
<th>$\sim Q$</th>
<th>$P \rightarrow Q$</th>
<th>$\sim Q \rightarrow \sim P$</th>
<th>$Q \rightarrow P$</th>
<th>$\sim P \rightarrow \sim Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<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>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
When you need to negate an implication, it is usually best to work with the form on the right.

\[ \sim(P \rightarrow Q) \equiv \sim Q \land P \]

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>\sim Q</th>
<th>P \rightarrow Q</th>
<th>\sim(P \rightarrow Q)</th>
<th>\sim Q \land P</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Summary
Summary of Terminology

- A conjunction is two propositions joined by $\land$.
- A disjunction is two propositions joined by $\lor$.
- A negation is $\sim$ applied to a proposition.
- Two propositions are equivalent ($\equiv$) if the results in the last column of their truth tables are identical.
- An implication is two propositions joined by $\rightarrow$.
- A biconditional is two propositions joined by $\leftrightarrow$. 
A proposition that is always true is called a **tautology**.

A proposition that is always false is called a **contradiction**.

Given a proposition $P \rightarrow Q$,

- The **contrapositive** is $\sim Q \rightarrow \sim P$ (equivalent to proposition)
- The **converse** is $Q \rightarrow P$ (equivalent to inverse)
- The **inverse** is $\sim P \rightarrow \sim Q$ (equivalent to converse)
- The **negation** is $\sim(P \rightarrow Q)$ (equivalent to $P \land \sim Q$)