**Propositional Logic, Truth Tables**  
(Rosen, Sections 1.1, 1.2, 1.3)

**TOPICS**

- Propositional Logic
- Logical Operations
- Equivalences

---

**What is logic?**

Logic is a **truth-preserving system of inference**.

*Truth-preserving:* If the initial statements are true, the inferred statements will be true.

*Inference:* the process of deriving (inferring) new statements from old statements.

---

**Why logic?** To reason about programs, e.g. Are the two snippets “the same”?

```java
float x1=0, x2=0, y1=0, y2=0;
// Some code that assigns values to these
// variables (don't count on them all being
// zero by the time the next line is executed
if ((x1 > x2) || !((y1 > y2) || (x1 >= y2)))
    System.out.println("Call the paintBlue method");
else
    System.out.println("Call the paintRed method");

float x1=0, x2=0, y1=0, y2=0;
// Another conditional, same as above?
if (((x1 > x2) && (y1 <= y2)) &&
    ((x1 > x2) && (x1 < y2)))
    System.out.println("Call the paintBlue method");
else
    System.out.println("Call the paintRed method");
```
What are the options to answer
Are the two snippets “the same”?

Options:
A: They are both the same (we’re not guessing and can come by after class and show you)
B: They are both different (we’re not guessing and can come by after class and show you)
C: We have no clue but want to figure this out
D: We will write the program, run it for many values and report back to you
E: This is not an interesting problem

Let’s build towards the math

- We want to reason about Boolean expressions
- They are built out of numbers (and also strings) and variables and operators
- Operators as functions:
  - The comparison operator \( > \) maps two numbers to a Boolean value:
  
    \[
    > : \mathbb{R} \times \mathbb{R} \to \mathbb{B}
    \]
  - So do all comparison/relation operators
  - Boolean operators (\(||\), \(&&\))? They map two Booleans to a Boolean value:
    
    \[
    && : \mathbb{B} \times \mathbb{B} \to \mathbb{B}
    \]

Towards the math: let’s simplify

- If we reason over all possible numeric values life will be hard (and it will take a long time).
- For now, focus only on Boolean expressions:
  - variables
  - values
  - expressions built using Boolean operators
- How?
  - Let’s modify the program

Only Boolean variables/operators inside the condition

```java
float x1=0, x2=0, y1=0, y2=0;
boolean b1,b2,b3,b4,b5;
// Replace
// if ((x1 > x2) || ! ((y1 > y2) || (x1 >= y2)))
// by
b1 = (x1 > x2); b2 = (y1 > y2); b3 = (x1 >= y2);
if (b1 || ! (b2 || b3))
// and in the other one
// replace
// if (((x1 > x2) || (y1 <= y2)) &&
// ((x1 > x2) || (x1 < y2)))
// by
b4 = (y1 <= y2); b5 = (x1 < y2);
if ((b1 || b4) && (b1 || b5))
```
Welcome to Propositional Logic

- Also known as:
  - Propositional calculus
  - Boolean algebra
- Propositional logic allows us to prove or disprove equalities that appear in programs:
  - For example, is \((b1 || !(b2 || b3))\) the same thing as \((b1 || !b2) || (b1 || !b3))\)?
  - Yes, they are (always) equivalent by De Morgan's Law and Distributive Law.

Propositional Logic

- A *proposition* is a statement that is either true or false
- Examples:
  - This class is CS160 (true)
  - Today is Sunday (false)
  - It is currently raining in Singapore (???)
- The value may be unknown (i.e., a *variable*)
- Similar to numerical expressions, propositional logic also defines *operators*.

Propositional Logic (II)

- A propositional statement is one of:
  - A simple proposition
    - denoted by a capital letter, e.g. ‘A’.
  - A *negation* of a propositional statement
    - e.g. \(\neg A\) : “not A”
  - Two propositional statements joined by an *operator*
    - e.g. \(A \land B\) : “A and B”
    - e.g. \(A \lor B\) : “A or B”
  - Use *parentheses* as needed (precedence is a convention)
    - e.g. \(A \land (B \lor C)\)

Truth Tables

- The truth value of a compound propositional statement is determined by its truth table
- Truth tables define the truth value of a connective for every possible truth value of its terms
Logical negation

- Negation of proposition A is ¬A
  - A: It is snowing.
  - ¬A: It is not snowing
  - A: Newton knew Einstein.
  - ¬A: Newton did not know Einstein.
  - A: I am not registered for CS195.
  - ¬A: I am registered for CS195.

Negation Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>¬A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Logical and (conjunction)

- Conjunction of A and B is A ∧ B
  - A: CS160 teaches logic.
  - B: CS160 teaches Java.
  - A ∧ B: CS160 teaches logic and Java.

- Combining conjunction and negation
  - A: I like fish.
  - B: I like sushi.
  - I like fish but not sushi: A ∧ ¬B

Truth Table for Conjunction

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A∧B</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 (disjunction)

- Disjunction of A and B is \( A \lor B \)
  - A: Today is Friday.
  - B: It is snowing.
  - \( A \lor B \): Today is Friday or it is snowing.

- This statement is true if any of the following hold:
  - Today is Friday
  - It is snowing
  - Both
  - Otherwise it is false

Truth Table for Disjunction

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( A \lor B )</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>

Exclusive Or

- The “or” connective \( \lor \) is inclusive: it is true if either or both arguments are true
- There is also an exclusive or \( \oplus \)

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( A \oplus B )</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>0</td>
</tr>
</tbody>
</table>

Confusion over Inclusive OR and Exclusive OR

- Restaurants typically let you pick one (either soup or salad, not both) when they say “The entrée comes with a soup or salad”.
  - Use exclusive OR to write as a logic proposition
- Give two interpretations of the sentence using inclusive OR and exclusive OR:
  - Students who have taken calculus or intro to programming can take this class
**Conditional & Biconditional Implication**

- The conditional implication connective is \( \rightarrow \)
- The biconditional implication connective is \( \leftrightarrow \)
- These, too, are defined by truth tables

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></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>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</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>

**Conditional implication**

- A: A programming homework is due.
- B: It is Tuesday.
- \( A \rightarrow B: \)
  - If a programming homework is due, then it must be Tuesday.
  - A programming homework is due only if it is Tuesday.
  - Is this the same?
  - If it is Tuesday, then a programming homework is due.

**Bi-conditional**

- A: You can drive a car.
- B: You have a driver’s license.
- \( A \leftrightarrow B \)
  - You can drive a car if and only if you have a driver’s license (and vice versa).
- What if we said “if”? 
- What if we said “only if”?

**Compound Truth Tables**

- Truth tables can also be used to determine the truth values of compound statements, such as \((A \lor B) \land (\neg A)\) (fill this as an exercise)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</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>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Tautology and Contradiction

- A **tautology** is a compound proposition that is always true.
- A **contradiction** is a compound proposition that is always false.
- A **contingency** is neither a tautology nor a contradiction.
- A compound proposition is **satisfiable** if there is at least one assignment of truth values to the variables that makes the statement true.

Examples

<table>
<thead>
<tr>
<th></th>
<th>¬A</th>
<th>A ∧ ¬A</th>
<th>A ∧ A</th>
<th>A ¬A</th>
</tr>
</thead>
<tbody>
<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>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Logical Equivalence

- Two compound propositions, p and q, are logically equivalent if p → q is a tautology.
- Notation: p ≡ q
- De Morgan’s Laws:
  - ¬ (p ∧ q) ≡ ¬ p ∨ ¬ q
  - ¬ (p ∨ q) ≡ ¬ p ∧ ¬ q
- How so? Let’s build a truth table!

Proof ¬(p ∧ q) ≡ ¬ p ∨ ¬ q

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>¬p</th>
<th>¬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>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Show \( \neg(p \lor q) \equiv \neg p \land \neg q \)

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

Other Equivalences

- Show \( p \rightarrow q \equiv \neg p \lor q \)
- Show Distributive Law:
  - \( p \lor (q \land r) \equiv (p \lor q) \land (p \lor r) \)

Show \( p \rightarrow q \equiv \neg p \lor q \)

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

Show \( p \lor (q \land r) \equiv (p \lor q) \land (p \lor r) \)

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

<table>
<thead>
<tr>
<th>Equivalence</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p \land T \equiv p )</td>
<td>Identity</td>
</tr>
<tr>
<td>( p \lor F \equiv p )</td>
<td></td>
</tr>
<tr>
<td>( p \land q \equiv q \land p )</td>
<td>Commutative</td>
</tr>
<tr>
<td>( p \lor q \equiv q \lor p )</td>
<td></td>
</tr>
<tr>
<td>( p \lor (p \land q) \equiv p )</td>
<td>Absorption</td>
</tr>
<tr>
<td>( p \land (p \lor q) \equiv p )</td>
<td></td>
</tr>
</tbody>
</table>

See Rosen for more.

Equivalences with Conditionals and Biconditionals

- **Conditionals**
  - \( p \to q \equiv \neg p \lor q \)
  - \( p \to q \equiv \neg q \to \neg p \)
  - \( \neg(p \to q) \equiv p \land \neg q \)

- **Biconditionals**
  - \( p \leftrightarrow q \equiv (p \to q) \land (q \to p) \)
  - \( p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q \)
  - \( \neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q \)

Prove Biconditional Equivalence

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

Converse, Contrapositive, Inverse

- The **converse** of an implication \( p \to q \) reverses the propositions: \( q \to p \)
- The **inverse** of an implication \( p \to q \) inverts both propositions: \( \neg p \to \neg q \)
- The **contrapositive** of an implication \( p \to q \) reverses and inverts: \( \neg q \to \neg p \)

The converse and inverse are not logically equivalent to the original implication, but the contrapositive is, and may be easier to prove.