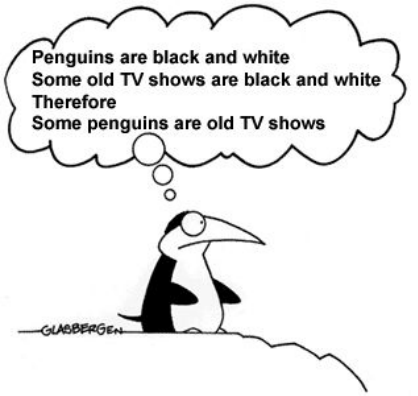
Propositional Logic, Truth Tables (Rosen, Sections 1.1, 1.2, 1.3)

TOPICS

- Propositional Logic
- Logical Operations
- Equivalences

Logic?



What is logic?

Logic is a <u>truth-preserving</u> <u>system</u> of <u>inference</u>

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

System: a set of mechanistic transformations, based on syntax alone

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"?

```
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: $> : R \times R \rightarrow B$
  - So do all comparison/relational operators
  - Boolean operators (||, &&)? They map two Booleans to a Boolean value: $\&\& : B \times B \rightarrow B$

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

```
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 <u>De Morgan's Law</u> and <u>Distributive Law</u>.

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. ¬A : "not A"
  - Two propositional statements joined by an *operator*
    - e.g. A ∧ B : "A and B"
    - e.g. A ∨ B : "A or B"
  - Use parentheses as needed (precedence is a convention)
    - e.g. A ∧ (B∨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

A	$\neg A$
0	1
1	0

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

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Logical or (*disjunction*)

- Disjunction of A and B is A ∨ B
  - A: Today is Friday.
  - B: It is snowing.
  - A ∨ 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

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Exclusive Or

- The "or" connective ∨ is inclusive: it is true if either *or both* arguments are true
- There is also an exclusive or ⊕

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

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 →
- The biconditional implication connective is ↔
- These, too, are defined by truth tables

A	B	$A \rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

A	B	$A \leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

Conditional implication

- A: A programming homework is due.
- B: It is Tuesday.
- A → 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 ↔ 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∨B)∧(¬A) (fill this as an exercise)

A	B	$\neg A$	$A \vee B$	$(A \square B) \square (\square A)$
0	0	1	0	0
0	1	1	1	1
1	0	0	1	0
1	1	0	1	0

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

A	¬A	A∨¬A	A∧¬A
0	1	1	0
1	0	1	0

Result is always true, no matter what A is

Therefore, it is a ***tautology***

Result is always false, no matter what A is

Therefore, it is a ***contradiction***

Logical Equivalence

- Two compound propositions, p and q, are logically equivalent if $p \leftrightarrow q$ is a tautology.
- Notation: $p \equiv q$
- De Morgan's Laws:
- $\neg (p \wedge q) \equiv \neg p \vee \neg q$
- $\neg (p \vee q) \equiv \neg p \wedge \neg q$
- How so? Let's build a truth table!

Prove $\neg(p \wedge q) \equiv \neg p \vee \neg q$

p	*q*	¬p	¬q	(p ∧ q)	¬(p ∧ q)	¬p ∨ ¬q
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

Show $\neg(p \vee q) \equiv \neg p \wedge \neg q$

p	q	$\neg p$	$\neg q$	$(p \vee q)$	$\neg(p \vee q)$	$\neg p \wedge \neg q$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

Other Equivalences

- Show $p \rightarrow q \equiv \neg p \vee q$
- Show Distributive Law:
  - $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$

Show $p \rightarrow q \equiv \neg p \vee q$

p	q	$\neg p$	$p \rightarrow q$	$\neg p \vee q$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	0	1	1

Show $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$

p	q	r	$q \wedge r$	$p \vee q$	$p \vee r$	$p \vee (q \wedge r)$	$(p \vee q) \wedge (p \vee r)$
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

More Equivalences

Equivalence	Name
$p \wedge T \equiv p$ $p \vee F \equiv p$	Identity
$p \wedge q \equiv q \wedge p$ $p \vee q \equiv q \vee p$	Commutative
$p \vee (p \wedge q) \equiv p$ $p \wedge (p \vee q) \equiv p$	Absorption

See Rosen for more.

Equivalences with Conditionals and Biconditionals

- Conditionals
  - $p \rightarrow q \equiv \neg p \vee q$
  - $p \rightarrow q \equiv \neg q \rightarrow \neg p$
  - $\neg(p \rightarrow q) \equiv p \wedge \neg q$
- Biconditionals
  - $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
  - $p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$
  - $\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$

Prove Biconditional Equivalence

p	q	$\neg q$	$p \leftrightarrow q$	$\neg(p \leftrightarrow q)$	$p \leftrightarrow \neg q$
0	0	1	1	0	0
0	1	0	0	1	1
1	0	1	0	1	1
1	1	0	1	0	0

Converse, Contrapositive, Inverse

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

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