CMPSCI 250: Introduction to Computation

Lecture #2: Propositions and Boolean Operators
David Mix Barrington
6 September 2013

Propositions and Operators

- What is a Proposition?
- Java Boolean Variables
- Boolean Operators, Compound Propositions
- AND, OR, NOT, and XOR
- Implication and Equivalence
- Tautologies

What is a Proposition?

- A **proposition** is a statement that is either true or false.
- In mathematics we want to reason about statements like "$x = 5$" or "these two triangles are congruent" without knowing whether they are true or false. We could say "if $x = 5$, then $x^2 = 25$", or "if one length and all three angles are the same, then the triangles are congruent".

More about propositions

- In computing we reason with **assertions** about a program, like "if this method terminates, the value of `i` is positive". Ultimately we'd like to say "if the input is as specified, then the output is as specified", meaning "the program is **correct**".
- What isn't a proposition? Questions, commands, statements without meaning, paradoxes like "this statement is false", or incompletely specified statements.

Java Boolean Values

- Java has a primitive `boolean` data type, and every boolean has either the value `true` or the value `false`.
- We use booleans in the conditions for `if` or `while` statements -- if we write "`if (x > 4) y = 5;`", then the statement "`y = 5`" will be executed only if the boolean value "`x > 4`" evaluates to `true` at run time.

Java Boolean Operators

- The operators ==, !=, >, >=, <, and <= create boolean values from values of other types. We often write methods that return `boolean` values, or use existing `boolean` methods like `equals`. We'll soon see operators that make new booleans from old.
- You may think of a "proposition" as any statement that could be modeled by a boolean variable. Of course, propositions may be about anything, not just computer data.

Making Compound Propositions

- A **compound proposition** is a proposition that is made up from other propositions, called **atomic propositions**, using **boolean operators**.
- If I say "you must have MATH 132, and either CMPSCI 187 or ECE 242", we can define three atomic propositions and write this as a compound proposition. We let x represent "you have MATH 132", y be "you have CMPSCI 187, and z be "you have ECE 242".

- Now my statement can be written "x, and either y or z". Symbolically, we write this as "$x \wedge (y \vee z)$".
- If x, y, and z are any three booleans, the truth of $x \wedge (y \vee z)$ depends on which of x, y, and z are true. In Java, if x, y, and z are boolean variables, we can write the expression `x && (y || z)`, and this represents $x \wedge (y \vee z)$.
- This is the **propositional calculus**.

AND and OR

- If x and y are any two propositions, their **conjunction** $x \wedge y$ is the proposition that is true if and only if *both* x and y are true. We read it "x and y". The Java operators & and && both compute the value of a conjunction -- we usually use && which only evaluates the second argument if it is needed.
- The **disjunction** of x and y is written $x \vee y$, read "x or y", and is true if either is true, or both. In Java the disjunction is | or ||.

Practice Clicker Question #1

- Let p be "dogs like beef", q be "cats like tuna", and r be "pigs like mud". Which of the following English statements matches "$(q \wedge p) \vee (r \wedge q)$"?
- (a) Cats like tuna and dogs like beef, or pigs like mud and cats like tuna, or both.
- (b) Dogs like beef or pigs like mud, or both.
- (c) If pigs like mud, then so do both dogs and cats.
- (d) Either cats like tuna or dogs like beef, and either pigs like mud or cats like tuna.

Answer #1

- Let p be "dogs like beef", q be "cats like tuna", and r be "pigs like mud". Which of the following English statements matches "$(q \wedge p) \vee (r \wedge q)$"?
- (a) Cats like tuna and dogs like beef, or pigs like mud and cats like tuna, or both.
- (b) Dogs like beef or pigs like mud, or both.
- (c) If pigs like mud, then so do both dogs and cats.
- (d) Either cats like tuna or dogs like beef, and either pigs like mud or cats like tuna.

NOT and XOR

- The **negation** of x is written ¬x, is read "not x", and is true when x is false and false when x is true. In Java the negation operator is !.
- The **exclusive or** of x and y is written x ⊕ y, read "x exclusive or y" or "x or y, but not both", and is true if one of x and y is true and the other false. In Java we can write x ^ y to compute the exclusive or of x and y.

Implication

- The last two boolean operators we will define are **implication** and **equivalence**. These are important in mathematics because each expresses a relationship between propositions that we often want to prove.
- The implication $x \rightarrow y$ is read "if x, then y" or "x implies y". It is true if *either* x is false or y is true. Equivalently, it is true *unless* x is true and y is false. It's important to learn this formal definition, whatever you think "if" means.

Practice Clicker Question #2

- Let p be "frogs are green" and q be "trout live in trees". Which English sentence does not mean the same as "$\neg p \rightarrow \neg q$"?
- (a) It is not the case that frogs are not green and trout live in trees.
- (b) If frogs are green, then trout live in trees.
- (c) If frogs are not green, then trout do not live in trees.
- (d) Either frogs are green or trout do not live in trees.

Answer #2

- Let p be "frogs are green" and q be "trout live in trees". Which English sentence does not mean the same as "$\neg p \rightarrow \neg q$"?
- (a) It is not the case that frogs are not green and trout live in trees.
- (b) If frogs are green, then trout live in trees.
- (c) If frogs are not green, then trout do not live in trees.
- (d) Either frogs are green or trout do not live in trees.

`false` implies anything

- Normally in mathematics we want to make some **assumptions** and prove that some must be true if the assumptions are true. This is an implication.
- Given our rule, from any false proposition we can prove anything else. Bertrand Russell gave an example of a proof of "I am Elvis" from the premise "0 = 1". ("1 = 2 by arithmetic, Elvis and I are two people, thus Elvis and I are one person".)

Equivalence

- Two boolean values are **equivalent** if they are both true or both false. If x and y are propositions, x ↔ y is the proposition that x

  and y are equivalent. We can write this this in Java as `x == y`.

- We are often interested in the equivalence of two compound propositions with the same atomic propositions. For example, "x → y" and "¬x ∨ y" are equivalent.

More on Equivalence

- How do we know this? They are each true in three of the four possible cases -- they are false only if x is true and y is false. They have the same **truth tables**, as we will soon see.
- As in Java, we have rules for precedence of operations. Negation is first, then the operators $\wedge$, $\vee$, and $\oplus$, then the operators $\rightarrow$ and $\leftrightarrow$. So we can write our equivalence of $x \rightarrow y$ and $\neg x \vee y$ as the single compound proposition $(x \rightarrow y) \leftrightarrow (\neg x \vee y)$.

Tautologies

- This compound proposition $(x \rightarrow y) \leftrightarrow \neg x \vee$

  y is true in all four possible situations of truth values for x and y, so it is *always true*. We call such a compound proposition a **tautology**.

- In the next lecture we will learn a systematic method to show that a compound proposition is a tautology, by checking all the possible combinations of values of its atomic propositions.

The Bigger Picture

- Next week we will see how to use particular tautologies as rules, chaining them together to verify larger tautologies without having to check all the possible cases.
- If there are many atomic propositions, this may be the only feasible way to verify the tautology. Remember that if there are k atomic propositions, there are 2^k possible cases!

The Bigger Picture

- In mathematics, our central task with boolean values turns out to be verifying that particular implications or equivalences *are* tautologies.
- Verifying $x \rightarrow y$ means that if we assume x, we may conclude y.
- Verifying $x \leftrightarrow y$ means that x and y are in effect the same compound proposition.