Propositional Logic

Margaret M. Fleck

26 January 2008

This lecture introduces propositional logic. It closely follows section 1.1 of Rosen, through about p. 12. (We don't cover bitwise operations.)

1 Announcements

Office hours are now posted on the web page and will be adjusted as needed.

Reminder that Homework 0 is due in class on Friday and that you should get yourself onto the class newsgroup. Notice that latex source for the homework is posted for students who want to learn to use latex, but it is **not** required. If you do format your homework, turn it in **as hardcopy** not electronically.

When submitting homework, it is not necessary to include the problem statements. The answers are sufficient.

2 A bit about style

Writing mathematics requires two things. You need to get the logical flow of ideas correct. And you also need to express yourself in standard style. Mathematical style is best taught by example and is similar to what happens in English classes.

Mathematical writing uses a combination of equations and parts that look superficially like English. Mathematical English is almost like normal English, but differs in some crucial ways. You are probably familiar with the fact that physicists use terms like "force" differently from everyone else. Or the fact that people from England think that "paraffin" is a liquid whereas

that word refers to a solid substance in the US. We will try to highlight the places where mathematical English isn't like normal English.

You will also learn how to make the right choice between an equation and an equivalent piece of mathematical English. For example, $\wedge$ is a shorthand symbol for "and." The shorthand equations are used when we want to look at a complex structure all at once, e.g. discuss the logical structure of a proof. When writing the proof itself, it's usually better to use the longer English equivalents, because the result is easier to read. There is no hard-and-fast line here, but we'll help make sure you don't go too far in either direction.

3 Propositions

Two systems of logic are commonly used in mathematics: propositional logic and predicate logic. We'll start by covering propositional logic.

A *proposition* is a statement which is true or false (but never both!). For example, "Urbana is in Illinois" or $2 \leq 15$. It can't be a question. It also can't contain variables, e.g. $x \leq 9$ isn't a proposition.

The lack of variables prevents propositional logic from being useful for very much, though it has some applications in circuit analysis, databases, and artificial intelligence. Predicate logic is an upgrade that adds variables. We will mostly be using predicate logic in this course. We just use propositional logic to get started.

4 Complex propositions

Statements can be joined together to make more complex statements. For example, "Urbana is in Illinois and Margaret was born in Wisconsin." To talk about complex sequences of statements without making everything too long, we represent each simple statement by a variable. E.g. if p is "Urbana is in Illinois" and q is "Margaret was born in Wisconsin", then the whole long statement would be "p and q". Or, using shorthand notation $p \wedge q$.

The statement $p \wedge q$ is true when both p and q are true. We can express this with a "truth table":

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

We use the value T for "true" and F for "false". Please don't use 1 and 0. That makes your math look like a computer program rather than math.

Similarly, $\neg p$ is the shorthand for "not p." In our example, $\neg p$ would be "Urbana is not in Illinois." $\neg p$ is true exactly when p is false.

$p \vee q$ is the shorthand for "p or q", which is true when either p or q is true. Notice that it is also true when both p and q are true, i.e. its true table is:

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

When mathematicians use "or", this is always how they intend to be understood. Notice that this is different from normal English, in which "or" sometimes excludes the possibility that both statements are true. Normal English "or" sometimes matches mathematical "or" and sometimes another operation called **exclusive or** defined by

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

Exclusive or has some important applications in computer science, especially in encoding strings of letters for security reasons. However, we won't see it much in this class.

5 Implication

Two propositions p and q can also be joined into the **conditional** statement. "if p, then q." This has many synonyms in mathematical English, e.g. "p implies q" or "q follows from p". Browse the (long) list in Rosen and look back at it later on, as needed.

The shorthand for this conditional is $p \rightarrow q$ and its truth table is

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

For example, "If Obama is president, then Obama lives in the White House" is true. But "if Obama is president, then $2 > 4$ is false. All the examples tend to be a bit artificial, because we don't have variables yet.

In normal English, we tend not to use conditionals in which the "if" part is false. E.g. "If Bush is president, then Urbana is in Illinois." In mathematical English, such statements occur more often. And, worse, they are always considered true, no matter whether the "then" part is true or false. For example, this statement is true: "If Bush is president, then $2 > 4$."

The easiest way to remember the right output values for this operation is to remember that the value is false in exactly one case: when p is true and q is false.

Normal English requires that conditional sentences have some sort of causal connection between the two propositions, i.e. one proposition is true because the other is true. E.g. "If Helen learns to write C++, she will get a good job." It would seem odd if we said "If Urbana is in Illinois, then Margaret was born in Wisconsin." because there's no reason why one follows from the other. In mathematical English, this statement is just fine: there doesn't have to be any causal connection.

In normal English if/then statements, there is frequently a flow of time involved. Unless we make a special effort to build a model of time, propositional logic is timeless. This makes the English motivating examples slightly awkward. It's not a big problem in mathematics, because mathematical proofs normally discuss a world that is static. It has a cast of characters (e.g. variables, sets, functions) with a fixed set of properties, and we are just reasoning about what those properties are. Only very occasionally do we talk about taking an object and modifying it.

In computer programming, we often see things that look like conditional statements, e.g. "if $x > 0$, then increment y". But these are commands for the computer to do something, changing its little world. whereas the similar-looking mathematical statements are timeless. Formalize what it means for a computer program to "do what it's supposed to" (which you'll see in CS 421) requires modelling how the world changes over time.

6 Converse, contrapositive, biconditional

The converse of $p \to q$ is $q \to p$. The two statements are not equivalent. To see this, compare the previous truth table with this one:

p	q	$q \to p$
T	T	T
T	F	T
F	T	F
F	F	T

The converse mostly occurs in two contexts. First, getting the direction of implication backwards is a common bug in writing proofs. That is, using the converse rather than the original statement. Second, the phrase "p implies q, and conversely" means that p and q are true under exactly the same conditions. The shorthand for this is the biconditional operator $p \leftrightarrow q$.

p	q	$q \leftrightarrow p$
T	T	T
T	F	F
F	T	F
F	F	T

The contrapositive of $p \to q$ is formed by swapping the roles of p and q and negating both of them to get $\neg q \to \neg p$. The contrapositive **is** equivalent to the original statement. Here's a truth table showing why:

p	q	$\neg p$	$\neg q$	$\neg q \to \neg p$
T	T	F	F	T
T	F	F	T	F
F	T	T	F	T
F	F	T	T	T

To figure out the last column of the table, recall that $\neg q \to \neg p$ will be false in only one case: when the if part ($\neg q$) is true and the consequent ($\neg p$) is false.

Let's consider what these variations look like in an English example:

- If it's below zero, my car won't start.
- converse: If my car won't start, it's below zero
- contrapositive: If my car will start, then it's not below zero.

7 Complex statements

Very complex statements can be made using combinations of connectives. E.g. "If it's below zero or my car does not have gas, then my car won't start and I can't go get groceries." The shorthand notation is particularly useful for manipulating complicated statements. For example (as we'll see Wednesday), taking the negative of a statement. So, this example has the form

$$(p \vee \neg q) \rightarrow (\neg r \wedge \neg s)$$

When you try to read a complex set of propositions all glued together with connectives, there is sometimes a question about which parts to group together first. English is a bit vague about the rules. So, for example, in the previous example, you need to use common sense to figure out that "I can't go get groceries" is intended to be part of the conclusion of the if/then statement.

In mathematical shorthand, there are conventions about which parts to group together first. In particular, you apply the "not" operators first, then the "and" and "or". Then you take the results and do the implication operations. Use parentheses if you intend the reader to group things differently.

This is basically similar to the rules in (say) high-school algebra. Look at the examples in Rosen.

Rosen points out that there are conventions about applying "and" before "or." However, in math it is traditional to use parentheses rather than relying the reader remembering this distinction.

If you aren't sure that your reader will group your statement the way you intended, use parentheses.

You can build truth tables for complex statements, e.g.

p	q	r	$q \wedge r$	$(q \wedge r) \rightarrow p$
T	T	T	T	T
T	F	T	F	T
F	T	T	T	F
F	F	T	F	T
T	T	F	F	T
T	F	F	F	T
F	T	F	F	T
F	F	F	F	T

However, if there are k variables, your table needs 2^k lines to cover all

possible combinations of input truth values. This is cumbersome when there are more than 2-3 variables.