Propositional Equivalences, Intro to Predicates

Margaret M. Fleck

27 January 2010

In this lecture, we will see how to show that two propositional formulas are logically equivalent (Section 1.2 of Rosen). We will also see some of the material on predicates and quantifiers from section 1.3 of Rosen.

1 Announcements

Remember that the first homework is due on Friday, 4pm.

2 Recap

Last class, we saw that a proposition is a statement that is either true or false (e.g. not a question, doesn't contain variables). We also saw the meanings of several logical operators, along with shorthand notation: $\vee$ (or), $\wedge$ (and), $\rightarrow$ (implies), $\leftrightarrow$ (implies in both directions), $\neg$ (not), $\oplus$ (exclusive or). These operators can be used to create compound propositions.

3 Logical Equivalence

Two (simple or compound) propositions p and q are *logically equivalent* if they are true for exactly the same input values. The shorthand notation for this is $p \equiv q$. One way to establish logical equivalence is with a truth table.

For example, last class we saw that implication has the truth table:

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

A frequently useful fact is that $p \rightarrow q$ is logically equivalent to $\neg p \vee q$. To show this, built the truth table for $\neg p \vee q$. and compare the output values to those for $p \rightarrow q$.

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

Two very well-known equivalences are *De Morgan's Laws*. These state that $\neg(p \wedge q)$ is equivalent to $\neg p \vee \neg q$. and that $\neg(p \vee q)$ is equivalent to $\neg p \wedge \neg q$. Similar rules in other domains (e.g. set theory) are also called De Morgan's Laws. They are especially helpful, because they tell you how to simplify the negation of a complex statement involving "and" and "or".

We can show this easily with another truth table:

p	q	$\neg p$	$\neg q$	$p \vee q$	$\neg(p \vee q)$	$\neg p \wedge \neg q$
T	T	F	F	T	F	F
T	F	F	T	T	F	F
F	T	T	F	T	F	F
F	F	T	T	F	T	T

Truth tables are a nice way to show equivalence for compound propositions which use only 2-3 variables. For more variables, they are cumbersome because analyzing the effects of k input variables requires 2^k rows.

4 Some useful logical equivalences

Rosen pp. 24-25 (reproduced in handout on-line) lists a number of useful logical equivalences, which you could establish with truth tables (but probably don't want to). Browse through them at your leisure. Some closely (and for good mathematical reasons) resemble rules from algebra and/or set theory, sometimes sharing the same name.

For example, one of the two distributive laws states that

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

which is similar to the familiar rule

$$a(b + c) = ab + ac$$

There are a couple minor differences. First, arithmetic has a clear rule that multiplication is done first, so the righthand side doesn't require parentheses. The order of operations is less clear for the logic, so it's wise to use the parentheses. Second, for arithmetic, you can distribute multiplication over addition, but not vice versa. In logic, either $\vee$ or $\wedge$ can be distributed over the other.

5 Negating propositions

An important use of logical equivalences is to help you correctly state the negation of a complex proposition, i.e. what it means for the complex proposition not to be true. This is important when you are trying to prove a claim false or convert a statement to its contrapositive. Also, looking at the negation of a definition or claim is often helpful for understanding precisely what the definition or claim means. Having clear mechanical rules for negation is important when working with concepts that are new to you, when you have only limited intuitions about what is correct.

For example, suppose we have a claim like "If M is regular, then M is paracompact or M is not Lindelöf." I'm sure you have no idea whether this is even true, because it comes from a math class you are almost certain not to have taken. However, you can figure out its negation.

First, let's convert the claim into shorthand so we can see its structure. Let r be "M is regular", p be "M is paracompact", and l be "M is Lindelöf." Then the claim would be $r \rightarrow (p \vee \neg l)$.

The negation of $r \rightarrow (p \vee \neg l)$ is $\neg(r \rightarrow (p \vee \neg l))$. However, to do anything useful with this negated expression, we normally need to manipulate it into an equivalent expression that has the negations on the individual propositions.

The key equivalences used in doing this manipulation are:

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

- $\neg(p \vee q) \equiv \neg p \wedge \neg q$
- $\neg(p \rightarrow q) \equiv p \wedge \neg q$.

The last of these follows from an equivalence we saw above: $p \rightarrow q \equiv \neg p \vee q$ plus one of DeMorgan's laws.

So we have

$$\neg(r \rightarrow (p \vee \neg l)) \equiv r \wedge \neg(p \vee \neg l) \equiv r \wedge \neg p \wedge \neg\neg l \equiv r \wedge \neg p \wedge l$$

So the negation of our original claim is "M is regular and M is not paracompact and M is Lindelöf." Knowing the mechanical rules helps you handle situations where your logical intuitions aren't fully up to the task of just seeing instinctively what the negation should look like.

6 Some useful terminology

T and F are special constant propositions with no variables that are, respectively, always true and always false.

A compound proposition p that is true for all combinations of input values is called a *tautology*. Or, equivalently, p is logically equivalent to T. A compound proposition p that is false for all combinations of input values (aka logically equivalent to F) is called a *contradiction*. So T and F are especially simple examples of a tautology and an contradiction.

Two compound propositions p and q are logically equivalent exactly when $p \leftrightarrow q$ is a tautology.

7 Establishing new logical equivalences

These established logical equivalences can be used to show that other, novel or more complex, equivalences also hold. You are already familiar with the technique used to do these proofs: long chain of equations.

For example, to show that $\neg(p \vee (\neg p \wedge q))$ is logically equivalent to $\neg p \wedge \neg q$, we show the following sequence of equations.

$$
\begin{aligned}
\neg(p \vee (\neg p \wedge q)) &\equiv \neg p \wedge \neg(\neg p \wedge q) && (1)\\
&\equiv \neg p \wedge (\neg(\neg p) \vee \neg q && (2)\\
&\equiv \neg p \wedge (p \vee \neg q) && (3)\\
&\equiv (\neg p \wedge p) \vee (\neg p \wedge \neg q) && (4)\\
&\equiv (p \wedge \neg p) \vee (\neg p \wedge \neg q) && (5)\\
&\equiv F \vee (\neg p \wedge \neg q) && (6)\\
&\equiv (\neg p \wedge \neg q) \vee F && (7)\\
&\equiv \neg p \wedge \neg q && (8)
\end{aligned}
$$

Steps (1) and (2) use the two De Morgan's laws. Step (3) uses the fact that negations cancel. Step (4) uses a distributive law, and so forth.

In the above derivation, I've been *very* picky about matching the exact form of the rules in our tables. Sometimes it's necessary to be this picky, e.g. when working with operations (such as matrix multiplication) that may not obey some of the usual arithmetic rules such as associativity or commutativity. To be very picky, pretend you are a computer and apply rules literally, without using commonsense simplifications. When you are doing elementary problems using logical equivalences, check instructions carefully to see how picky you need to be.

Extreme pickiness would become oppressive as we get into more complex material. For example, step (6) uses a commutative law to put the formula in the exact right form to apply the identity law for F. In most situations, one would just proceed directly from (4) to (6) without bothering to name the formula justifying the transformation. Similarly, it's common to simplify double-negations right away, without an extra step or a justification.

8 Predicates and Variables

Propositions are a helpful beginning but too rigid to represent most of the interesting bits of mathematics. To do this, we need predicate logic, which allows variables and predicates that take variables as input. We'll get started with predicate logic now, but delay covering some of the details (e.g. sections 1.3 and 1.4) until they become relevant to the proofs we're looking at.

A predicate is a statement that becomes true or false if you substitute in values for its variables. For example, "$x^2 \geq 10$" or "y is winter hardy."

Suppose we call these $P(x)$ and $Q(x)$. Then $Q(x)$ is true if x is "mint" but not if x is "tomato".[1]

If we substitute concrete values for all the variables in a predicate, we're back to having a proposition. That wasn't much use, was it?

The main use of predicates is to make general statements about what happens when you substitute a variety of values for the variables. For example:

P(x) is true for every x

For example, "For every integer x, $x^2 \geq 10$" (false).

Consider "For all x, $2x \geq x$." Is this true or false? This depends on what values x can have. Is x any integer? In that case the claim is false. But if x is supposed to be a natural number, then the claim is true.

In order to decide whether a statement involving quantifiers is true, you need to know what types of values are allowed for each variable. Good style requires that you state the type of the variable explicitly when you introduce it, e.g. "For all natural numbers x, $2x \geq x$." Exceptions involve cases where the type is very, very clear from the context, e.g. when a whole long discussion is all about (say) the integers. If you aren't sure, a redundant type statement is a minor problem whereas a missing type statement is sometimes a big problem.

9 Synchronization notes

The following sections were covered in 1pm but not 9am, so they will be repeated in the notes for lecture 5. We'll even out the synchronization in due course.

10 More on quantifiers

The operation "for all x" is formally known as a *universal quantifier*. The set of values that can be substituted in for a variable is called its *domain* or its *replacement set*.

[1] A winter hardy plant is a plant that can survive the winter in a cold climate, e.g. here in central Illinois.

The other heavily-used quantifier in formal mathematics is the *existential quantifier*, written "there exists an x" For example, "there exists an integer x such that $5 < x < 100$." This means that at least one integer x (and possibly a whole bunch of integers) satisfies the equation.

Notice that the existential quantifer is followed by "such that" in fluent mathematical English, which isn't quite parallel to what you see with the universal quantifier. This isn't for any good reason. It's just how mathematical English happens to have evolved, and you should do likewise so that your written mathematics looks professional. "Such that" is sometimes abbreviated "s.t."

The general idea of a quantifier is that it expresses express how many of the values in the domain make the claim true. Normal English has a wide range of quantifiers which tell you roughly how many of the values work, e.g. "some", "a couple", "a few", "many", "most." Mathematics largely uses just the two we've seen, though you occasionally see a third quantifier "there exists a unique x." This means that one and only one x satisfies the predicate. For example, "there is a unique integer x such that $x^2 = 0$." Mathematicians use the adjective "unique" to mean that there's only one such object (similar to the normal usage but not quite the same).

The universal quantifier has the shorthand notation $\forall$. For example, $\forall x \in \mathbb{N}, x \leq 2x$. The existential quantifier is written $\exists$, e.g. $\exists y \in \mathbb{R}, y = \sqrt{2}$. Notice that we don't write "such that" when the quantifier is in shorthand. The unique existence quantifier is written $\exists!$ as in $\exists! x \in \mathbb{R}, x^2 = 0$.