Logic and Computation **Lectures 11**
CSU 290 Spring 2009 (Pucella) **Monday, Feb 2, 2009**

Before we start this lecture, let me talk about the quiz I gave, since it is good practice for the midterm—the midterm will involve establishing the validity/satisfiability of propositional formulas using truth tables, finding counterexamples of validity, as well as coming up with equational proofs of validity.

The quiz consisted giving an equational proof of the validity of $\neg a \Rightarrow \neg(a \wedge b)$. Two proofs at least can be given, a long one that requires no insight, and a shorter one that does require some thinking.

As we saw, proving the validity of $\neg a \Rightarrow \neg(a \wedge b)$ is the same as proving the validity of $(\neg a \Rightarrow \neg(a \wedge b)) \equiv \mathit{true}$. Here is the first proof, then. You should refer to the bank of basic propositional validities I handed out.

Equational proof of: $(\neg a \Rightarrow \neg(a \wedge b)) \equiv \mathit{true}$

$$\begin{aligned}
\neg a \Rightarrow \neg(a \wedge b) &\equiv \mathit{true} \\
&\quad \text{Substitution } \neg(a \wedge b) \equiv (\neg a \vee \neg b) \\
\neg a \Rightarrow (\neg a \vee \neg b) &\equiv \mathit{true} \\
&\quad \text{Substitution } \neg a \Rightarrow (\neg a \vee \neg b) \equiv \neg\neg a \vee (\neg a \vee \neg b) \\
\neg\neg a \vee (\neg a \vee \neg b) &\equiv \mathit{true} \\
&\quad \text{Substitution } \neg\neg a \equiv a \\
a \vee (\neg a \vee \neg b) &\equiv \mathit{true} \\
&\quad \text{Substitution } a \vee (\neg a \vee \neg b) \equiv (a \vee \neg a) \vee \neg b \\
(a \vee \neg a) \vee \neg b &\equiv \mathit{true} \\
&\quad \text{Substitution } a \vee \neg a \equiv \mathit{true} \\
\mathit{true} \vee \neg b &\equiv \mathit{true} \\
&\quad \text{Substitution } \mathit{true} \vee \neg b \equiv \mathit{true} \\
\mathit{true} &\equiv \mathit{true}
\end{aligned}$$

There, longish, but straightforward.

If you're willing to put in some thought, you can come up with a shorter equational proof.

Equational proof of: $(\neg a \Rightarrow \neg(a \wedge b)) \equiv \mathit{true}$

$$\begin{aligned}
\neg a \Rightarrow \neg(a \wedge b) &\equiv \mathit{true} \\
&\quad \text{Substitution } \neg a \Rightarrow \neg(a \wedge b) \equiv (a \wedge b) \Rightarrow a
\end{aligned}$$

$$\begin{aligned} (a \wedge b) \Rightarrow a &\equiv \mathit{true} \\ & \quad \text{Substitution } (a \wedge b) \Rightarrow a \equiv \mathit{true} \\ \mathit{true} &\equiv \mathit{true} \end{aligned}$$

For the midterm, I will ask you to put in all the steps, all the substitutions, in gory explicit detail. Just for this one time, to make sure that we get enough data to be able to judge whether you understand the idea of equational proofs, or not.

Reasoning about Equality

Last week, we looked at standard propositional logic, and gave three ways of proving that a propositional logic formula was valid:

(1) a direct proof using truth tables;

(2) an indirect proof using an equational proof;

(3) using ACL2 as an oracle.

Our goal in the next few weeks is to take what we learned about propositional logic, and apply it to the problem of reasoning about ACL2 programs.

To do so, we will not to more towards a more expressive logic, one that does not just talk about truth values of propositional variables, but more generally can talk about the truth of arbitrary ACL2 expressions. We'll be getting there in a few steps.

Today, we show how to extend our logic to reason about equality of values in general (and in ACL2 in particular).

I fumbled this a little bit in lecture, so let me try again, trying to do things right.

First, let me define the basis of what will be *(quantifier-free) first-order logic*, a generalization of propositional logic. I will use $F, G, H, \ldots$ to range over formulas of quantifier-free first-order logic, to distinguish them from propositiional logic formulas.

A formula F, G of quantifier-free first-order logic is one of:

- $\mathit{true}, \mathit{false}$ (constants)
- `(= x y)`, where `x` and `y` are variables
- $\neg F$ (negation)
- $F \wedge G$ (conjunction)
- $F \vee G$ (disjunction)

- $F \Rightarrow G$ (implication)
- $F \equiv G$ (equivalence, also written $F \Leftrightarrow G$).

Thus, a formula F looks just like a propositional formula, except that it has expressions such as `(= x y)` instead of propositional variables.

We will care about the "validity" of such formulas. We will define this term by generalizing the notion of validity for propositional logic.

In propositional logic, the fact that, say, $\neg a \Rightarrow \neg(a \wedge b)$ is valid can be understood (if you squint just right at the definition of validity) to mean that "for all possible values of a and b, $\neg a \Rightarrow \neg(a \wedge b)$ is true", where truth is given by the truth table semantics of the various logical connectives. Here, the possible values of a and b, because we are in propositional logic and because they are propositional variables, are true and false.

For our formulas, we will say something similar: a formula F is valid if for any choice of values for the variables in F, the resulting formula is true. Thus, for example,

$$\neg\texttt{(= x y)} \Rightarrow \neg(\texttt{(= x y)} \wedge \texttt{(= y z)}) \tag{1}$$

is valid if for all choices of values for `x`, `y`, `z`, the formula is true. Now, the values for `x`, `y`, `z` are not limited to true and false, but rather *to all possible values in the universe of ACL2.* It is easy to evaluate the truth of the formula for any value assignment, using what we know of `=` as it is defined in ACL2: if we take `x` to have value `1`, `y` to have value `2`, `z` to have value `3`, then formula (1) evaluates to $\mathsf{F} \Rightarrow \neg(\mathsf{F} \wedge \mathsf{F})$ because both `(= 1 2)` and `(= 2 3)` are false, which evaluates to T because of the truth table for $\Rightarrow$ (it is true if the antecedent is false). What we want to check is that the formula is true *no matter what values we choose for the variables.* The problem is that we can't exhaustively look at all possible values—there are infinitely many of them.

But thankfully, we still have our indirect proofs.

In propositional logic, we saw two principles, or rules: specialization, and substitution. We have something similar in our new logic.

First, we need a way to come up with valid formulas of quantifier-free first-order logic. The easiest way is to use a principle, or rule, that tells us that a valid formula of propositional can be "instantiated" to a valid formula of quantifier-free first-order logic.

If φ is a valid formula of propositional logic, then replacing every propositional variable in φ by a formula of quantifier-free first-order logic (where we replace the same propositional variable by the same formula) yields a valid formula of quantifier-free first-order logic.

As an example of this rule, consider the example above. We showed (in the quiz), that $\neg a \Rightarrow \neg(a \wedge b)$ is a valid propositional formula. Great. Now, replace every a by `(= x y)` and every b by `(= y z)`, and the result, which is just formula (1), is a valid formula of quantifier-free first-order logic.[1]

Good, so that immediately gives us a bank of valid quantifier-free first-order formulas, by instantiating any of the valid propositional formulas we know. Great.

The interesting thing is that substitution can also be used in quantifier-free first-order logic.

If $G_1 \equiv G_2$ is a valid formula of quantifier-free first-order logic, then F is valid if and only if F where we replace an occurrence of G_1 by G_2 is valid.

Because our equational proofs all rely on the rule of substitution, we can derive equational proofs for quantifier-free first-order logic like we did for first-order logic, and obtain proofs of validity.

Here is another equational proof of the validity of formula (1).

$$\underline{\textbf{Equational proof of:}}\ \ (\neg\texttt{(= x y)} \Rightarrow \neg(\texttt{(= x y)} \wedge \texttt{(= y z)})) \equiv \mathit{true}$$

$$\begin{array}{rcl}
\neg\texttt{(= x y)} \Rightarrow \neg(\texttt{(= x y)} \wedge \texttt{(= y z)}) & \equiv & \mathit{true} \\
\text{Substitution} & & \neg\texttt{(= x y)} \Rightarrow \neg(\texttt{(= x y)} \wedge \texttt{(= y z)}) \\
 & & \quad \equiv (\texttt{(= x y)} \wedge \texttt{(= y z)}) \Rightarrow \texttt{(= x y)} \\
(\texttt{(= x y)} \wedge \texttt{(= y z)}) \Rightarrow \texttt{(= x y)} & \equiv & \mathit{true} \\
\text{Substitution} & & (\texttt{(= x y)} \wedge \texttt{(= y z)}) \Rightarrow \texttt{(= x y)} \equiv \mathit{true} \\
\mathit{true} & \equiv & \mathit{true}
\end{array}$$

This is of course the same proofs as the second proof in the quiz solution, except that we use "instantiation" to get a valid formula to substitute at every step.

Good, so now we can prove a bunch of validities of quantifier-free first-order logic. Except, we can't prove everything that's valid. Right now, the only things we can prove are "propositional" things. We do not treat the `(= x y)` expressions in any special way. But we know that `(= x y)` actually means something. Equality has some properties, that are not captured by the rules we have above, the rules coming from propositional logic. These are *axioms*, that capture the properties of equality. There are the well-known properties that say that equality is an equivalence relation:

Axioms for Equality: The following formulas are valid for all choices of `x`, `y`, and `z`

- reflexivity: $\texttt{(= x x)} \equiv \mathit{true}$
- symmetry: $\texttt{(= x y)} \equiv \texttt{(= y x)}$

[1] You can prove formally that this is always the case—we won't do it here though. It's a good exercise if you're mathematically inclined.

- transitivity: $\texttt{(= x y)} \wedge \texttt{(= y z)} \equiv \texttt{(= x y)} \wedge \texttt{(= x z)}$

As an example, consider proving the validity of the following formula, which is a form of transitivity: $[\texttt{(= x y)} \wedge (\texttt{(= x y)} \Rightarrow \texttt{(= y z)})] \Rightarrow \texttt{(= x z)}$. Note what this is saying: if we know that `(= x y)` holds, and that `(= x y)` implies `(= y z)`, then `(= x z)` must hold as well.

Here is a proof, where I refer to substitutions by the name of the rule instantiated or axiom used.

To keep this proof somewhat short, let me use the following valid propositional formula, which we proved valid a few lectures ago. (Redo the proof, it's a good exercise.)

$$p \wedge (p \Rightarrow q) \equiv p \wedge q \tag{2}$$

(We often refer to such helper formulas as *lemmas*.)

Equational proof of: $[\texttt{(= x y)} \wedge (\texttt{(= x y)} \Rightarrow \texttt{(= y z)})] \Rightarrow \texttt{(= x z)}$

$$[\texttt{(= x y)} \wedge (\texttt{(= x y)} \Rightarrow \texttt{(= y z)})] \Rightarrow \texttt{(= x z)} \equiv \textit{true}$$

Substitution by (2)

$$[\texttt{(= x y)} \wedge \texttt{(= y z)}] \Rightarrow \texttt{(= x z)} \equiv \textit{true}$$

Substitution by transitivity of `=`

$$[\texttt{(= x y)} \wedge \texttt{(= x z)}] \Rightarrow \texttt{(= x z)} \equiv \textit{true}$$

Substitution by ∧ implication

$$\textit{true} \equiv \textit{true}$$