Quiz: Given `(defun second (L) (car (cdr L)))`, prove that `(= (second (cons a (cons b c))) b)`.

Proving Conditional Equality Formulas

Last time, we saw proofs for theorems of the form `(= exp exp')`. That's an important class of theorems, and everal of the theorems we care about for proving program correctness take that form.

Another very important class of theorems, which you can think as a generalization of the above, are theorems of the form: `F` $\Longrightarrow$ `(= exp exp')`, that is, that two expressions are equal when some conditions `F` are met. We will develop a bunch of standard techniques for dealing with such formulas that do not rely on expanding out the $\Longrightarrow$ into $\vee$. The formula F in the above is often called the *context*, or the *assumptions*.

To help us with such formulas, we introduce a variant of the Rule of Equality Substitution, which lets us substitute equals for equals in the right-hand side of the implication, when the equality is given in the context.

> **Rule of Equality Substitution 2**: We can rewrite a formula of the form $F \wedge$ `(= exp exp')` $\Longrightarrow G$ by replacing occurrences of `exp` in G by `exp'`.

Thus, for example, we can rewrite $F \wedge$ `(= x y)` $\Longrightarrow$ `(= (foo x) (bar z))` into $F \wedge$ `(= x y)` $\Longrightarrow$ `(= (foo y) (bar z))`.

Without further ado, here is an example, a generalization of Theorem 2 in Lecture 13. I am listing all steps explicitly, just so that you see where everything is used.

Theorem 1. `(endp x)` $\Longrightarrow$ `(= (app x y) y)`

Proof.

`(endp x)` $\Longrightarrow$ `(= (app x y) y)`

by endp axiom

`(= (endp x) T)` $\Longrightarrow$ `(= (app x y) y)`

by definition of ***app***

`(= (endp x) T)` $\Longrightarrow$ `(= (if (endp x) y (cons (car x) (app (cdr x) y))) y)`

by equality substitution with `(= (endp x) T)`

`(= (endp x) T)` $\Longrightarrow$ `(= (if T y (cons (car x) (app (cdr x) y))) y)`

by if axiom

`(= (endp x) T)` $\Longrightarrow$ `(= y y)`

by reflexivity of `=` *and propositional reasoning (since* $p \Longrightarrow$ *true is valid)*

□

Note the use of the Rule of Equality Substitution 2 in the 3rd step. Make sure you understand where it comes from. Note also how we ended the proof. We got something that is an instance of the propositional formula $p \Longrightarrow$ *true*, and you can convince yourself that that's a valid formula. That's how most of our proofs for formulas of the form $F \Longrightarrow$ `(= exp exp')` will end.

Consider two other examples. Those two examples put together, we will see in a week, argue that `app` is associative, that is, `(= (app x (app y z)) (app (app x y) z))`. We can't quite prove that yet, but we can prove those things that, when put together later, will give us that.

First off, we can prove associativity in the special case where `x` is not a cons-structure. This proof is fairly simple, and we will use it to illustrate a slight simplification for our proofs. It is somewhat painful to carry the context along when it does not change, so we will call the context C, and for ease of reference when applying substitutions from the context name all the different formulas that appear in the context. (The context will generally be a conjunction of formulas.) Note also that I skip the "obvious" subsitutions, like the `car`, `cons`, `if` axioms. Stuff that by now you know how to do. I only justify the use of theorems, assumptions from the context, Modus Ponens, or the use of *interesting* axioms.

Theorem 2. `(endp x)` $\Longrightarrow$ `(= (app x (app y z)) (app (app x y) z))`

Proof. There are two proofs. First, the "direct" one that uses good old substitution. Let

$$C = \texttt{(endp x)} \qquad [A_1]$$

be the context.

$C \Longrightarrow$ `(= (app x (app y z))`
`(app (app x y) z))`

by definition of `app`

$C \Longrightarrow$ `(= (app x (app y z))`
`(app (if (endp x)`
`y`
`(cons (car x) (app (cdr x) y))) z))`

by A_1

$C \Longrightarrow$ `(= (app x (app y z))`
`(app y z))`

by definition of ***app***

$C \Longrightarrow$ `(= (if (endp x)`
`(app y z)`
`(cons (car x) (app (cdr x) (app y z))))`
`(app y z))`

by A_1

$C \Longrightarrow$ `(= (app y z)`
`(app y z))`

by propositional reasoning

Here is an alternate proof that uses the already proved Theorem 1, and in which all the action is in the context. It's a bit trickier, and not that much shorter. And the context grows, so it's hard to give it a stable name. I use the Rule of Modus Ponens (MP) to apply Theorem 1, which is of the right form if you'll remember.

`(endp x)` $\Longrightarrow$
`(= (app x (app y z)) (app (app x y) z))`

by MP with appropriate instance of Theorem 1

`(endp x)` $\wedge$ `(= (app x (app y z)) (app y z))` $\Longrightarrow$
`(= (app x (app y z)) (app (app x y) z))`

by MP with appropriate instance of Theorem 1

`(endp x)` $\wedge$ `(= (app x (app y z)) (app y z))` $\wedge$ `(= (app x y) y)` $\Longrightarrow$
`(= (app x (app y z)) (app (app x y) z))`

Okay, at this point let the context be

$$\begin{aligned} C = \ & \texttt{(endp x)} && [A_1] \\ & \wedge \texttt{(= (app x (app y z)) (app y z))} && [A_2] \\ & \wedge \texttt{(= (app x y) y)} && [A_3] \end{aligned}$$

and we can continue the proof, which is now trivial:

$C \Longrightarrow$ `(= (app x (app y z)) (app (app x y) z))`

by A_2

$C \Longrightarrow$ `(= (app y z) (app (app x y) z))`

by A_3

$C \Longrightarrow$ `(= (app y z) (app y z))`

by propositional reasoning

□

The second theorem says that `app` is associative if we assume it is associative for "smaller" lists.

Theorem 3.

```
(consp x) ∧ (= (app (cdr x) (app y z)) (app (app (cdr x) y) z)) ⟹
                                    (= (app x (app y z)) (app (app x y) z))
```

Proof. Let the context

```
C =  (consp x)                                                    [A1]
   ∧ (= (app (cdr x) (app y z)) (app (app (cdr x) y) z))          [A2]
```

$C \Longrightarrow$ `(= (app x (app y z))`
`(app (app x y) z))`

by definition of ***app***

$C \Longrightarrow$ `(= (app x (app y z))`
`(app (if (endp x)`
`y`
`(cons (car x) (app (cdr x) y))) z))`

by A_1 *(note the axiom* ***(consp x)*** $\equiv \neg$ ***(endp x)****)*

$C \Longrightarrow$ `(= (app x (app y z))`
`(app (cons (car x) (app (cdr x) y)) z))`

by Theorem 1 from lecture 13

$C \Longrightarrow$ `(= (app x (app y z))`
`(cons (car x) (app (app (cdr x) y) z)))`

by A_2

$C \Longrightarrow$ `(= (app x (app y z))`
`(cons (car x) (app (cdr x) (app y z))))`

by Theorem 1 from lecture 13

$C \Longrightarrow$ `(= (app x (app y z))`
`(app (cons (car x) (cdr x)) (app y z)))`

by MP with cons axiom — name the new context formula A_3

$C \wedge$ `(= (cons (car x) (cdr x)) x)` $\Longrightarrow$ `(= (app x (app y z))`
`(app (cons (car x) (cdr x)) (app y z)))`

by A_3

$C \wedge$ `(= (cons (car x) (cdr x)) x)` $\Longrightarrow$ `(= (app x (app y z))`
`(app x (app y z)))`

by propositional reasoning

□

Remark 1: A Strategy for Starting Proofs

As a summary, let me point out some of the ways in which you should approach the question of proving a particular formula we give you. It all depends on the form of the formula.

(1) If the formula is simply an equality such as `(= exp exp')`, then the proof will generally consist in a sequence of substitutions.

(2) If the formula is of the form $C \Longrightarrow$ `(= exp exp')`, then identity the context as we have done in this lecture, and prove the equality, possibly using formulas from the context.

(3) If the formula is of the form $F \wedge G$, then try to prove F and G separately. The reason why this work is that there is a result about our logic that says that whenever F is a theorem (i,e,, F has a proof) and G is a theorem (i,e., G has a proof), then $F \wedge G$ is a theorem (i.e, it has a proof, and in fact that proof can be obtained by "merging" together the proof of F and the proof of G)—and this is true no matter what F and G are. Generally, F and G will be in one of the forms (1) or (2), and you can use the techniques there for it.

(4) If the formula is not of the form (1), (2), or (3), it may be possible to use propositional reasoning to convert the formula in that form. For instance, if the formula if of the form $F \Longrightarrow (G \Longrightarrow$ `(= exp exp')`$)$, which is *not* of any of the forms above, then you can use the fact that $p \Longrightarrow (q \Longrightarrow r) \equiv (p \wedge q) \Longrightarrow r$ is a valid propositional formula (check it!) to rewrite $F \Longrightarrow (G \Longrightarrow$ `(= exp exp')`$)$ into $(F \wedge G) \Longrightarrow$ `(= exp exp')`, which is of the form (2).

Remark 2: An Alternate Proof Method

If the formula you want to prove is of form (1) above, then what you have to prove is an equality of the form `(= exp exp')`. Similarly if the formula you want to prove is of the form (2) above, at least after you've identified and isolated the context, by which point you're usually only doing substitutions on the equality formula. (Not always though, note the use of MP in the proof of Theorem 3.)

The way I've advocated you work on that equality is to simply rewrite it until you get to something that is trivially true. (Generally, an instance of the reflexivity of `=` is what you end up with.)

To use the high-school algebra example from lecture 9, here is how we prove equalities.

$$\begin{aligned}(a+b)^2 &= a^2+2ab+b^2\\ (a+b)(a+b) &= a^2+2ab+b^2\\ a^2+ab+ba+b^2 &= a^2+2ab+b^2\\ a^2+ab+ab+b^2 &= a^2+2ab+b^2\\ a^2+2ab+b^2 &= a^2+2ab+b^2\end{aligned}$$

An alternative approach, which is the one used in Section 1, and that you might see in the labs, is to derive an equality $(a+b)^2 = a^2+2ab+b^2$
as follows. Start with one side of the equality, and try to rewrite it into the other side.

$$\begin{aligned}&(a+b)^2\\ &= (a+b)(a+b)\\ &= a^2+ba+ab+b^2\\ &= a^2+ab+ab+b^2\\ &= a^2+2ab+b^2\end{aligned}$$

In other words, a proof that we would write:

`(= exp1 exp)`

justification 1

`(= exp2 exp)`

justification 2

`(= exp3 exp)`

justification 3

`(= exp4 exp)`

...

`(= exp exp)`

could be written

```
    exp1
=     [justification 1]
    exp2
=     [justification 2]
    exp3
=     [justification 3]
    exp3
...
    exp
```

You should play with a few examples (such as redoing the proofs from last lecture and this lecture) to see how this can be used to prove an equality.

I would prefer if you used the first method, the one we used in class, because it is slightly easier to use, even though it requires a bit more writing at times. Why is it easier? Because the second method requires you to: (1) determine which side to start from, and choosing the wrong side may cause you to have restart the proof from the beginning, and (2) some equalities are easier to deal with if you can work on both sides of the equation—for those equalities, the second method calls for showing that both sides of the equality are in fact equal to a third expression, and requires you to be able to identify that third expression when you encounter it, i.e., the proof for `(= exp exp')` may end up looking like:

```
  exp                    exp'
=                      =
  exp1                   exp1'
=                      =
  ...                    ...
=                      =
  expN                   expN
```