Logic and Computation	**Lecture 19**
CSU 290 Spring 2009 (Pucella)	**Monday, Mar 9, 2009**

Review via Reverse

Between the midterm on Wednesday and Spring Break last week, it pays to review some of what we've done relating to induction. We'll focus on `rev`, because it leads to interesting examples.

First, let's prove the following result that we'll need later. It tells you that the result of an append is a true list exactly when the second element of the append is a true list.

Theorem 1. `(= (true-listp (app x y)) (true-listp y))`

Proof. We prove this by induction on `x`. Here are the two proof obligations:

P1: `(endp x)` $\Longrightarrow$ `(= (true-listp (app x y)) (true-listp y))`

P2: $\neg$`(endp x)` $\wedge$ `(= (true-listp (app (cdr x) y)) (true-listp y))`
$\Longrightarrow$ `(= (true-listp (app x y)) (true-listp y))`

Let's prove P1. It's an implication, so let's identify the context and the assumptions in the context:

A1: `(endp x)`

Let's prove the consequent of the implication under that assumption

```
(= (true-listp (app x y)) (true-listp y))
```

*by def of **app**, A1, and if axiom*

```
(= (true-listp y) (true-listp y))
```

by reflexivity of `=`

Now, let's prove P2. It's also an implication, so let's identify the context and the assumptions in the context:

A1: $\neg$`(endp x)`

A2: `(= (true-listp (app (cdr x) y)) (true-listp y))`

Let's prove the consequent of the implication under those assumptions:

```
(= (true-listp (app x y)) (true-listp y))
```

$\boxed{\textit{by def of } \texttt{app}\textit{, A1, if axiom}}$

```
(= (true-listp (cons (car x) (app (cdr x) y))) (true-listp y))
```

$\boxed{\textit{by def of } \texttt{true-listp}\textit{, endp if and cdr axioms}}$

```
(= (true-listp (app (cdr x) y)) (true-listp y))
```

$\boxed{\textit{by A2}}$

```
(= (true-listp y) (true-listp y))
```

$\boxed{\textit{by reflexivity of } \texttt{=}}$

□

Recall the definition of `rev`:

```
(defun rev (L)
  (if (endp L)
      NIL
      (app (rev (cdr L)) (list (car L)))))
```

First off, consider `(true-listp (rev x))`. Let's look at what this says: no matter what `x` is, then the reverse of `x` is a true list. First question, do you think this is valid? A few examples seem to point out that yes. So let's try to prove it.

Theorem 2. `(true-listp (rev x))`

Proof. We prove this by induction on `x`. Here are the two proof obligations:

P1: `(endp x)` $\Longrightarrow$ `(true-listp (rev x))`

P2: $\neg$`(endp x)` $\wedge$ `(true-listp (rev (cdr x)))` $\Longrightarrow$ `(true-listp (rev x))`

Let's prove P1. It's an implication, so let's identify the context and the assumptions in the context:

A1: `(endp x)`

Let's prove the consequent of the implication under that assumption

```
(true-listp (rev x))
```

$\boxed{\textit{by def of } \texttt{rev}\textit{, A1, if axiom}}$

```
(true-listp NIL)
```

$\boxed{\textit{by def of } \texttt{true-listp}\textit{, endp and if axioms}}$

```
(= NIL NIL)
```

$\boxed{\textit{by reflexivity of } \texttt{=}}$

Now, let's prove P2. It's also an implication, so let's identify the context and the assumptions in the context:

A1: ¬`(endp x)`

A2: `(true-listp (rev (cdr x)))`

Let's prove the consequent of the implication under those assumptions:

```
(true-listp (rev x))
```

$\boxed{\textit{by def of } \texttt{rev}\textit{, A1, if axiom}}$

```
(true-listp (app (rev (cdr x)) (list (car x))))
```

$\boxed{\textit{by Theorem 1}}$

```
(true-listp (list (car x)))
```

$\boxed{\textit{by def of } \texttt{list}}$

```
(true-listp (cons (car x) NIL))
```

$\boxed{\textit{by def of } \texttt{true-listp}\textit{, endp if and cdr axioms}}$

```
(true-listp NIL)
```

$\boxed{\textit{by def of } \texttt{true-listp}\textit{, endp and if axioms}}$

```
(= NIL NIL)
```

$\boxed{\textit{by reflexivity of } \texttt{=}}$

□

Okay, so those were easy, partly because they were simple equality formulas.

The next one is more interesting. Consider the formula `(= (rev (rev x)) x)`, which says that reversing a list twice leads to the original list. Is it a valid formula?

The answer is no, and it requires you to think a bit. The easiest counter-example is to take `x` to be an integer, such as 5. `(rev 5)` is `NIL`, and so `(rev (rev 5))` is `NIL`, which is not equal to `5`.

So how can we restrict the formula so that it is valid. Intuitively, if we restrict `x` to be a true list, then it seems (by looking at examples) that the formula is actually valid. Thus, let's try to prove `(true-listp x)` $\Longrightarrow$ `(= (rev (rev x)) x)`.

Theorem 3. `(true-listp x)` $\Longrightarrow$ `(= (rev (rev x)) x)`.

Proof. We prove this by induction on `x`. Coming up with the proof obligations is a bit more subtle, because the formula to prove is an implication. Look at the lecture notes from last lecture. Intuitively, we add the new bits to the *existing* context of the formula. Here are the two proof obligations:

P1: `(endp x)` $\wedge$ `(true-listp x)` $\Longrightarrow$ `(= (rev (rev x)) x)`

P2: $\neg$`(endp x)`
$\wedge$ (`(true-listp (cdr x))` $\Longrightarrow$ `(= (rev (rev (cdr x))) (cdr x))`)
$\wedge$ `(true-listp x)`
$\Longrightarrow$ `(= (true-listp (app x y)) (true-listp y))`

Let's prove P1. It's an implication, so let's identify the context and the assumptions in the context:

A1: `(endp x)`

A2: `(true-listp x)`

Let's first work on the context a little bit. First off, note that we're assuming that both `x` is an atom, and `x` is a true list. Now, the only atom that is also a true list is `NIL`, so intuitively, the context gives us that `x` is `NIL`. There are a few ways to convince ourselves formally of that. The easiest is to see that `(endp x)` $\Longrightarrow$ `(true-listp x)` $\equiv$ `(= x NIL)` is valid. Here's a direct proof:

`(endp x)` $\Longrightarrow$ `(true-listp x)` $\equiv$ `(= x NIL)`

$\boxed{\textit{by def of } \texttt{true-list}\textit{, assumption, and if axiom}}$

`(endp x)` $\Longrightarrow$ `(= x NIL)` $\equiv$ `(= x NIL)`

$\boxed{\textit{by reflexivity of } \texttt{=} \textit{ and propositional reasoning}}$

So, because we know A1, we know that `(true-listp x)` and `(= x NIL)` are equivalent. And from A2, we know that `(true-listp x)` is true, meaning that `(= x NIL)` is true, so we can just add it to the context:

A3: `(= x NIL)`

Let's prove the consequent of the implication under those assumptions:

```
(= (rev (rev x)) x)
```
$\boxed{\textit{by def of } \texttt{rev}\textit{, A1, if axiom}}$

```
(= (rev NIL) x)
```
$\boxed{\textit{by def of } \texttt{rev}\textit{, endp and if axioms}}$

```
(= NIL x)
```
$\boxed{\textit{by A3}}$

Now, let's prove P2. It's also an implication, so let's identify the context and the assumptions in the context:

A1: ¬`(endp x)`

A2: `(true-listp (cdr x))` ⟹ `(= (rev (rev (cdr x))) (cdr x))`

A3: `(true-listp x)`

Again, let's work on the context a bit. First off, note that intuitively, because `x` is not an atom, and because `x` is a true list, then it must be the case that the `cdr` of `x` is a true list. We can establish formally as follows:

¬`(endp x)` ⟹ `(true-listp x)` ≡ `(true-listp (cdr x))`

$\boxed{\textit{by def of } \texttt{true-listp}\textit{, assumption, and if axiom}}$

¬`(endp x)` ⟹ `(true-listp (cdr x))` ≡ `(true-listp (cdr x))`

$\boxed{\textit{by reflexivity of } \texttt{=} \textit{ and propositional reasoning}}$

So, because we know A1, we know that `(true-listp x)` and `(true-listp (cdr x))` are equivalent. And from A3 we know that `(true-listp x)` is true, meaning that `(true-listp (cdr x))` is also true, so let's add it to the context:

A4: `(true-listp (cdr x))`

Now, because we know A4, the antecedent of A2 is true, meaning that its consequent is true, so we can just add the consequent of A2 to the context:

A5: `(= (rev (rev (cdr x))) (cdr x))`

Now, let's prove the consequent of the implication P2 under those assumptions:

`(= (rev (rev x)) x)`

by def of `rev`*, A1, and if axiom*

`(= (rev (app (rev (cdr x)) (list (car x)))) x)`

Now, it's not clear what to do. So let's look at what we have. We're taking the reverse of an append of two lists. We've proved something in the past about the reverse of appends, we get the append of the reverses (in the opposite order): `(= (rev (app A B)) (app (rev B) (rev A)))`. That's the key here, because it brings two `rev`s together and lets us use A5:

`(= (rev (app (rev (cdr x)) (list (car x)))) x)`

by Theorem `(= (rev (app A B)) (app (rev B) (rev A)))`

`(= (app (rev (list (car x))) (rev (rev (cdr x)))) x)`

by A5

`(= (app (rev (list (car x))) (cdr x)) x)`

by def of `list`

`(= (app (rev (cons (car x) NIL)) (cdr x)) x)`

by def of `rev`*, endp if cdr and car axioms*

`(= (app (app (rev NIL) (list (car x))) (cdr x)) x)`

by def of `rev`*, endp and if axioms*

`(= (app (app NIL (list (car x))) (cdr x)) x)`

by def of `app`*, endp and if axioms*

`(= (app (list (car x)) (cdr x)) x)`

by def of `list`

`(= (app (cons (car x) NIL) (cdr x)) x)`

by def of `app`*, endp if car and cdr axioms*

`(= (cons (car x) (app NIL (cdr x))) x)`

by def of `app`*, endp and if axioms*

`(= (cons (car x) (cdr x)) x)`

by `cons` *axioms, because we know A1*

□

I've spelled out most of the details in the above proof, but the only "tricky" bit really is to notice that we can use the rev-app theorem.

What about the following formula? `(= (rev (rev (rev x))) (rev x))`: reversing a list three times is the same as reversing it once. Is that valid? Actually, if we try with, say, integer 5, the result is indeed true, since `(rev (rev (rev 5)))` is `NIL`, and `(rev 5)` is also `NIL`. So this may just be valid. And we can in fact prove it very easily:

Theorem 4. `(= (rev (rev (rev y))) (rev y))`

Proof. There is a much more direct way to prove this than by proving it by induction—although we could. We could apply the previous Theorem 3, with `x` instantiated to `(rev y)`. Except that we can only apply Theorem 3 if we know that `(true-listp (rev y))` is true. But we do—that's just the content of Theorem 2. So here's the argument:

```
(= (rev (rev (rev y))) (rev y))
```

by Theorem 3, because we know ***(true-listp (rev y))*** *by Theorem 2*

□