Peter Dillinger

A Review of Proving by Induction

Let us look at some more examples and examine how to prove them by induction. (These are the theorems that led to the proof obligations in Homework 4. Refer to it for definitions of functions.)

What are the proof obligations for proving the following by induction?

```
(implies (true-listp b)
         (true-listp (app a b)))
```

First we need to consider what induction scheme to use and over what variables. Remember that induction schemes come from functions, and the best induction scheme for a conjecture will almost always come from one of the functions in it. This one has TRUE-LISTP and APP. Let's examine the induction schemes suggested by the definitions of these functions:

```
Induction Scheme for (true-listp x)
  Base test: (endp x)
  Replacements:  x <- (cdr x)

Induction Scheme for (app x y)
  Base test: (endp x)
  Replacements:  x <- (cdr x)
                 y <- y
```

(These came from the function definitions. See Lecture 21 for how.) Remember that each variable involved in the induction, each parameter to the function, must be replaced by something in the Induction Hypothesis, and those replacements are given by the parameters to the recursive call.

For example, APP calls itself with (cdr x) for x and y for y. So effectively, there is no replacing of y at all. That means the induction schemes given by (true-listp x) and by (app x y) are equivalent.

Getting back to what induction scheme to use for our conjecture... We have just one scheme, but we need to know which variable to induct over. The fact that we see (true-listp b) and (app a b) suggests using effectively the same scheme for two different variables. One would have (endp a) and the other (endp b). Because the base case (endp a) seems trivial, inducting based on (true-listp a), which is equivalent to inducting based on (app a b), seems best... and turns out to work.

So what are the proof obligations for proving

```
(implies (true-listp b)
         (true-listp (app a b)))
```

by induction based on (true-listp a)?

The conventional approach to an inductive proof would give us two obligations:

```
Base case:
(implies (endp a)
         (implies (true-listp b)
                  (true-listp (app a b))))
Inductive Step:
(implies (and (not (endp a))
              (implies (true-listp b)
                       (true-listp (app (cdr a) b))))
         (implies (true-listp b)
                  (true-listp (app a b))))
```

Now this is a correct answer, but there is another answer with three proof obligations, that is based on what I presented for proving implications by induction:

For this problem,

```
  B = (endp a)
  H = (true-listp b)
  H'= (true-listp b)
  C = (true-listp (app a b))
  C'= (true-listp (app (cdr a) b))
```

which gives us

```
  Modified Base Case:
  (implies (and (endp a)
                (true-listp b))
           (true-listp (app a b)))

  Ind. Hyp. Chaining:          (this one is trivial to prove; it's a tautology)
  (implies (and (not (endp a))
                (true-listp b))
           (true-listp b))

  Modified Inductive Step:
  (implies (and (not (endp a))
                (true-listp b)
                (true-listp b) ; if you omit repeated hypotheses, that's ok too
                (true-listp (app (cdr a) b)))
           (true-listp (app a b)))
```

And that is another answer, which I find easier to work with if we actually have to go prove it.

What are the proof obligations for proving the following by induction?

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

The schemes we get from TRUE-LISTP and from REV are identical, so we will just induct according to (rev x). Note that this formula is *not* an implication, so there is no need for the B, H, H'... business, and we have only two proof obligations:

```
  Base Case:
  (implies (endp x)
           (true-listp (rev x)))

  Inductive Step:
  (implies (and (not (endp x))
                (true-listp (rev (cdr x))))
           (true-listp (rev x)))
```

This problem is interesting however, because we do not need induction to prove it. You'll notice that in Homework 4, the induction hypothesis, (true-listp (rev (cdr x))) was not among the hypotheses. It turns out we can prove this just by cases on (endp x) and (not (endp x)). It's not incorrect or invalid to prove this by induction, but it's not necessary.

Finally, what are the obligations for proving the following by induction?

```
  (equal (rev-append x y)
         (app (rev x) y))
```

In this case we get different schemes depending on whether we use (rev x), which is the same as (app x y), or (rev-append x y). The correct choice is often the more complicated induction scheme, (rev-append x y), and in this case that will be the right choice:

```
 Induction Scheme for (rev-append x y)
   Base test:  (endp x)
   Replacements:  x <- (cdr x)
                  y <- (cons (car x) y)

 Proof obligations:
  Base Case:
  (implies (endp x)
           (equal (rev-append x y)
                  (app (rev x) y)))
  Inductive Step:
  (implies (and (not (endp x))
                (equal (rev-append (cdr x) (cons (car x) y))
                       (app (rev (cdr x)) (cons (car x) y))))
           (equal (rev-append x y)
                  (app (rev x) y)))
```

If we try inducting based on (rev x), then the inductive step is different and too hard to prove:

```
  (implies (and (not (endp x))
                (equal (rev-append (cdr x) y)
                       (app (rev (cdr x)) y)))
           (equal (rev-append x y)
                  (app (rev x) y)))
```

If we start with (rev-append x y) and open up the definition, we get

```
  (rev-append (cdr x) (cons (car x) y))
```

and what we know about (rev-append (cdr x) y) doesn’t seem to be helpful.