Peter Dillinger

Given the function definitions:

```
(defun integer-listp (x)
  (if (endp x)
    (equal x nil)
    (and (integerp (car x))
         (integer-listp (cdr x)))))

(defun keep-ints (l)
  (if (endp l)
    nil
    (if (integerp (car l))
      (cons (car l)
            (keep-ints (cdr l)))
      (keep-ints (cdr l)))))
```

Let us prove some cases of

```
  (integer-listp (keep-ints x))
```

Let us start with

```
  (implies (endp x)
           (integer-listp (keep-ints x)))
```

Assumption: (endp x)

```
  (integer-listp (keep-ints x))
<= { Def keep-ints, assumption (endp x), if-true }
  (integer-listp nil)
<= { Evaluation }
  t
```

===

Let us next prove

```
  (implies (and (consp x)
                (integer-listp (keep-ints (cdr x))))
           (integer-listp (keep-ints x)))
```

Assumptions: (consp x)
(integer-listp (keep-ints (cdr x)))

```
  (integer-listp (keep-ints x))
<= { Def keep-ints, assumption (consp x), if-false }
  (integer-listp (if (integerp (car x))
                   (cons (car x)
                         (keep-ints (cdr x)))
                   (keep-ints (cdr x))))
```

Now there does not seem to be much we can do here, because (car x) could be an integer or it could be something else. How do we prove it holds in both cases?

One of our rules of inference can help us out here. A kind of propositional deduction is CASE ANALYSIS, which comes from the tautology

```
  ((p -> q) /\ (~p -> q)) -> q
```

That says that if q is true when we assume p and also true when we assume ~p, then q is always true. That means we can reduce proving any proposition q to proving p -> q and ~p -> q for any formula p. This is a case of propositional deduction.

For the problem we are working on, let us first assume (integerp (car x)) and prove the conjecture, and then we will assume (not (integerp (car x))) and prove the conjecture. That will complete the proof of the original conjecture.

Case: (integerp (car x))

```
  (integer-listp (if (integerp (car x))
                     (cons (car x)
                           (keep-ints (cdr x)))
                     (keep-ints (cdr x))))
<= { Assumption (integerp (car x)), if-true }
  (integer-listp (cons (car x)
                       (keep-ints (cdr x))))
<= { Def integer-listp, not endp cons, if-false }
  (and (integerp (car (cons (car x)
                            (keep-ints (cdr x)))))
       (integer-listp (cdr (cons (car x)
                                 (keep-ints (cdr x))))))
<= { car-cons, cdr-cons }
  (and (integerp (car x))
       (integer-listp (keep-ints (cdr x))))
<= { both are assumptions, propositional deduction }
  t
```

Other case: (not (integerp (car x)))

```
  (integer-listp (if (integerp (car x))
                     (cons (car x)
                           (keep-ints (cdr x)))
                     (keep-ints (cdr x))))
<= { Assumption (not (integerp (car x))), if-false }
  (integer-listp (keep-ints (cdr x)))
<= { Assumption }
  t
```

Having proven both cases, the proof is complete.