Peter Dillinger

Given definitional axioms:

```
(equal (true-listp x)
       (if (endp x)
         (equal x nil)
         (true-listp (cdr x))))

(equal (add-to-end e l)
       (append l (cons e nil)))

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

Let us first prove

```
  (true-listp (add-to-end e l))
```

using the lemma True-listp-append:

```
  (implies (true-listp y)
           (true-listp (append x y)))
```

This isn't an equality proof, so we need to make a series of reductions:

```
  (true-listp (add-to-end e l))
<= { Defn add-to-end }
  (true-listp (append l (cons e nil)))
<= { Lemma True-listp-append }
  (true-listp (cons e nil))
<= { Defn true-listp, IF axiom, not endp cons }
  (true-listp (cdr (cons e nil)))
<= { cdr-cons axiom }
  (true-listp nil)
<= { Evaluation }
  t
```

Basically, using a propositional deduction on (true-listp (cons e nil)) and an instantiation of the lemma,

```
   (implies (true-listp (cons e nil))
            (true-listp (append l (cons e nil))))
```

allowed me to conclude (true-listp (append l (cons e nil))). In fact, the particular propositional deduction is Modus Ponens.

===

What if I said to prove (consp 42) using the lemma (integerp nil)?
Believe it or not, I can construct such a proof:

```
  (consp 42)
<= { Propositional deduction }
  nil
<= { Evaluation }
  (integerp nil)
<= { Lemma }
  t
```

What is interesting is that the assumption that (integerp nil) is a lemma stipulates that it is a theorem. In this case, the lemma given was not a theorem; in fact, it always evaluates to nil! This allowed us--through legal deductions on flawed assumptions--to conclude nil is a theorem. Once we have concluded nil, or "false", we can use a propositional deduction to conclude anything. (false -> p is a tautology.)

Is this really a proof? Yes it is, but not with regard to any theory ACL2 will allow. If an extension of ACL2's base theory is able to prove (integerp nil) then that extension is UNSOUND, because it allows us to conclude two contradictary propositions: (not (integerp nil)) and (integerp nil).

ACL2 does not allow unsound extensions of its theory, so this will not happen if the Lemmas you are given are actually theorems in a proper extension of ACL2's base theory.

(It's not important that you completely understand soundness yet.)

===

Prove

```
  (implies (and (integer-listp l)
                (integerp e))
           (integer-listp (add-to-end e l)))
```

using the lemma Integer-listp-append:

```
  (implies (and (integer-listp x)
                (integer-listp y))
           (integer-listp (append x y)))
```

Once again, we are not proving an equality, but this time we have assumptions we can use:

```
Assumptions:  (integerp-listp l)
              (integerp e)

  (integer-listp (add-to-end e l))
<= { Defn add-to-end }
  (integer-listp (append l (cons e nil)))
<= { Lemma Integer-listp-append }
  (and (integer-listp l)
       (integer-listp (cons e nil)))
<= { Prop. deduction, assumption (integer-listp l) }
  (integer-listp (cons e nil))
<= { Defn integer-listp, IF axiom, not endp cons }
  (and (integerp (car (cons e nil)))
       (integer-listp (cdr (cons e nil))))
<= { car-cons and cdr-cons }
  (and (integerp e)
       (integer-listp nil))
<= { Prop. deduction with Assumption (integerp e) }
  (integer-listp nil)
<= { Evaluation }
  t
```