Some example proofs

In the proof of the Church-Rosser Theorem, we saw one example of structural induction: we used it to show that the "walk" relation on lambda expressions has the diamond property. The base case involved non-recursive *var*expressions, and the inductive cases involved expressions built using *apply* or *lambda*.

We can also use structural induction to prove properties of **code** that operates on a recursive data structure. For example, here is the definition of the *concat*function again (which operates on lists):

```
let rec concat(L1, L2) =
   case(L1) {
      nil:  L2
      x::L: x::concat(L,L2)
   }
```

Example 1: Prove that for all lists L: concat(L, nil) = L

For this kind of proof (of the form *left-hand-side = right-hand-side*), we must transform each side of the equation (using the definition of the function itself and the induction hypothesis) so that we arrive at the same final term for both. Here we go...

Base case: L = nil

LHS: concat(nil, nil)
= nil // def of concat
RHS: nil

Induction step:

assume: ∀ L of length ≤ n, concat(L, nil) = L
show: true for L of length n+1, i.e., L is x::Tail
LHS: concat(x::Tail, nil)
= x::concat(Tail, nil) // def of concat
= x::Tail // ind. hyp.
RHS: x::Tail

Example 2: Prove that for all lists L1, L2, L3: concat(L1, concat(L2, L3))= concat(concat(L1, L2), L3)

Now let's try a slightly harder one: we'll show that *concat* is associative, using structural induction on L1.

Base case: L1 = nil

LHS: concat(nil, concat(L2, L3))
= concat(L2, L3) // def of concat
RHS: concat(concat(nil, L2), L3)
= concat(L2, L3) // def of concat

Induction step:

assume: ∀ L1 of length ≤ n, concat(L1, concat(L2, L3)) = concat(concat(L1, L2), L3)
show: true for L1 of length n+1, i.e., L1 is x::Tail
LHS: concat(x::Tail, concat(L2, L3))
= x::concat(Tail, concat(L2, L3)) // def of concat
= x::concat(concat(Tail, L2), L3) // ind. hyp.
RHS: concat(concat(x::Tail, L2), L3)
= concat(x::concat(Tail, L2), L3) // def of concat