

Recommended problems 8 — CS3512 — Fall '11

Our topic this time is structural induction, which the textbook does not teach. Nonetheless, you will find in Section 4.4 and the associated exercises many claims provable by structural induction. Look them over, and try some.

Pay particular attention, as usual, to the structure of the proof — you can't write a good proof if you don't understand your obligations. Follow the structural induction proof format specified in the lecture notes.

Additional problems, with solutions

1 Recall the function

$$\text{map} : (A \rightarrow B) \times \text{lists}(A) \rightarrow \text{lists}(B)$$

defined recursively as follows.

$$\begin{aligned} \text{map}(f, \langle \rangle) &= \langle \rangle \\ \text{map}(f, x :: L) &= f(x) :: \text{map}(f, L) \end{aligned}$$

Let f be a bijection from A to A . Prove that, for all $L \in \text{lists}(A)$,

$$\text{map}(f, \text{map}(f^{-1}, L)) = \text{map}(f^{-1}, \text{map}(f, L)).$$

Proof by structural induction on L .

$$\begin{aligned} \text{Basis: } \text{map}(f, \text{map}(f^{-1}, \langle \rangle)) &= \text{map}(f, \langle \rangle) && \text{(defn map)} \\ &= \langle \rangle && \text{(defn map)} \\ &= \text{map}(f^{-1}, \langle \rangle) && \text{(defn map)} \\ &= \text{map}(f^{-1}, \text{map}(f, \langle \rangle)) && \text{(defn map)} \end{aligned}$$

Induction: $L \in \text{lists}(A)$, $x \in A$.

$$\text{IH: } \text{map}(f, \text{map}(f^{-1}, L)) = \text{map}(f^{-1}, \text{map}(f, L)).$$

$$\text{NTS: } \text{map}(f, \text{map}(f^{-1}, x :: L)) = \text{map}(f^{-1}, \text{map}(f, x :: L)).$$

$$\begin{aligned} \text{map}(f, \text{map}(f^{-1}, x :: L)) &= \text{map}(f, f^{-1}(x) :: \text{map}(f^{-1}, L)) && \text{(defn map)} \\ &= f(f^{-1}(x)) :: \text{map}(f, \text{map}(f^{-1}, L)) && \text{(defn map)} \\ &= f(f^{-1}(x)) :: \text{map}(f^{-1}, \text{map}(f, L)) && \text{(IH)} \\ &= f^{-1}(f(x)) :: \text{map}(f^{-1}, \text{map}(f, L)) && \text{(choice of } f\text{)} \\ &= \text{map}(f^{-1}, f(x) :: \text{map}(f, L)) && \text{(defn map)} \\ &= \text{map}(f^{-1}, \text{map}(f, x :: L)) && \text{(defn map)} \end{aligned}$$

2 Consider the function $\text{cat} : \text{lists}(A) \times \text{lists}(A) \rightarrow \text{lists}(A)$ defined recursively as follows.

$$\begin{aligned}\text{cat}(\langle \rangle, M) &= M \\ \text{cat}(x :: L, M) &= x :: \text{cat}(L, M)\end{aligned}$$

Prove that for all $L, M, N \in \text{lists}(A)$,

$$\text{cat}(L, \text{cat}(M, N)) = \text{cat}(\text{cat}(L, M), N).$$

Take arbitrary $M, N \in \text{lists}(A)$. We'll prove by structural induction on L that

$$\text{cat}(L, \text{cat}(M, N)) = \text{cat}(\text{cat}(L, M), N)$$

for all $L \in \text{lists}(A)$.

$$\begin{aligned}\textit{Basis:} \quad \text{cat}(\langle \rangle, \text{cat}(M, N)) &= \text{cat}(M, N) && \text{(defn cat)} \\ &= \text{cat}(\text{cat}(\langle \rangle, M), N) && \text{(defn cat)}\end{aligned}$$

Induction: $L \in \text{lists}(A)$, $x \in A$

$$\text{IH: } \text{cat}(L, \text{cat}(M, N)) = \text{cat}(\text{cat}(L, M), N)$$

$$\text{NTS: } \text{cat}(x :: L, \text{cat}(M, N)) = \text{cat}(\text{cat}(x :: L, M), N)$$

$$\begin{aligned}\text{cat}(x :: L, \text{cat}(M, N)) &= x :: \text{cat}(L, \text{cat}(M, N)) && \text{(defn cat)} \\ &= x :: \text{cat}(\text{cat}(L, M), N) && \text{(IH)} \\ &= \text{cat}(x :: \text{cat}(L, M), N) && \text{(defn cat)} \\ &= \text{cat}(\text{cat}(x :: L, M), N) && \text{(defn cat)}\end{aligned}$$

3 Let flatten be the function from the binary trees over A to $\text{lists}(A)$ defined recursively as follows.

$$\begin{aligned}\text{flatten}(\langle \rangle) &= \langle \rangle \\ \text{flatten}(\langle L, x, R \rangle) &= \text{cat}(\text{flatten}(L), x :: \text{flatten}(R))\end{aligned}$$

Notice that this definition uses the list concatenation function (cat) from the previous problem.

Define $\text{rev}_L : \text{lists}(A) \rightarrow \text{lists}(A)$ as follows.

$$\begin{aligned}\text{rev}_L(\langle \rangle) &= \langle \rangle \\ \text{rev}_L(x :: L) &= \text{cat}(\text{rev}_L(L), \langle x \rangle)\end{aligned}$$

This is essentially the same as the reverse function for strings, previously discussed in lecture.

We will state without proof a useful lemma (similar to one we proved in class for reverse on strings). You are free to use this lemma in your solution to this problem.

Lemma: For all $L, M \in \text{lists}(A)$,

$$\text{rev}_L(\text{cat}(L, M)) = \text{cat}(\text{rev}_L(M), \text{rev}_L(L)).$$

(You'll also need to use the result from the previous problem.)

And finally, we define a similar reverse function, called rev_T , from the binary trees over A to the binary trees over A .

$$\begin{aligned}\text{rev}_T(\langle \rangle) &= \langle \rangle \\ \text{rev}_T(L, x, R) &= \langle \text{rev}_T(R), x, \text{rev}_T(L) \rangle\end{aligned}$$

Prove that

$$\text{rev}_L(\text{flatten}(T)) = \text{flatten}(\text{rev}_T(T))$$

for all binary trees T over A , by structural induction on T .

Claim: For all binary trees T over A ,

$$\text{rev}_L(\text{flatten}(T)) = \text{flatten}(\text{rev}_T(T)).$$

Proof by structural induction on T .

$$\begin{aligned} \text{Basis: } \quad \text{rev}_L(\text{flatten}(\langle \rangle)) &= \text{rev}_L(\langle \rangle) && \text{(defn flatten)} \\ &= \langle \rangle && \text{(defn rev}_L\text{)} \\ &= \text{flatten}(\langle \rangle) && \text{(defn flatten)} \\ &= \text{flatten}(\text{rev}_T(\langle \rangle)) && \text{(defn rev}_T\text{)} \end{aligned}$$

Induction: L, R are binary trees over A , and $x \in A$.

$$\begin{aligned} \text{IH: } \quad &\text{rev}_L(\text{flatten}(L)) = \text{flatten}(\text{rev}_T(L)) \\ &\text{and } \text{rev}_L(\text{flatten}(R)) = \text{flatten}(\text{rev}_T(R)). \end{aligned}$$

$$\text{NTS: } \text{rev}_L(\text{flatten}(\langle L, x, R \rangle)) = \text{flatten}(\text{rev}_T(\langle L, x, R \rangle)).$$

$$\begin{aligned} &\text{rev}_L(\text{flatten}(\langle L, x, R \rangle)) \\ &= \text{rev}_L(\text{cat}(\text{flatten}(L), x :: \text{flatten}(R))) && \text{(defn flatten)} \\ &= \text{cat}(\text{rev}_L(x :: \text{flatten}(R)), \text{rev}_L(\text{flatten}(L))) && \text{(Lemma)} \\ &= \text{cat}(\text{cat}(\text{rev}_L(\text{flatten}(R)), \langle x \rangle), \text{rev}_L(\text{flatten}(L))) && \text{(defn rev}_L\text{)} \\ &= \text{cat}(\text{cat}(\text{flatten}(\text{rev}_T(R)), \langle x \rangle), \text{flatten}(\text{rev}_T(L))) && \text{(IH)} \\ &= \text{cat}(\text{flatten}(\text{rev}_T(R)), \text{cat}(\langle x \rangle, \text{flatten}(\text{rev}_T(L)))) && \text{(previous problem)} \\ &= \text{cat}(\text{flatten}(\text{rev}_T(R)), x :: \text{cat}(\langle \rangle, \text{flatten}(\text{rev}_T(L)))) && \text{(defn cat)} \\ &= \text{cat}(\text{flatten}(\text{rev}_T(R)), x :: \text{flatten}(\text{rev}_T(L))) && \text{(defn cat)} \\ &= \text{flatten}(\langle \text{rev}_T(R), x, \text{rev}_T(L) \rangle) && \text{(defn flatten)} \\ &= \text{flatten}(\text{rev}_T(\langle L, x, R \rangle)) && \text{(defn rev}_T\text{)} \end{aligned}$$