Our topic this time is mathematical induction.

“Weak” mathematical induction is a special form of structural induction. In the lecture notes, I specify a format for weak induction proofs and provide quite a few examples.

We also studied “strong” mathematical induction. Again, in the lecture notes I specify a format for such proofs and give an example.

Finally, we treated structural induction, which the textbook barely covers. 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 induction proof formats for each of the three kinds of induction as specified in the lecture notes.

In Section 4.4 and the associated exercises, you can find many claims provable by one or the other form of mathematical induction. Look them over, and try some. Make sure to understand and follow the recommended format.

Additional problems, with solutions

1 Recall the recursively-defined sequence of Fibonacci numbers.

\[
\begin{align*}
F_0 &= 0 \\
F_1 &= 1 \\
F_{n+2} &= F_n + F_{n+1}
\end{align*}
\]

Here is a similar recursively-defined sequence, the so-called Lucas numbers.

\[
\begin{align*}
L_0 &= 2 \\
L_1 &= 1 \\
L_{n+2} &= L_n + L_{n+1}
\end{align*}
\]

Prove the following claim by strong mathematical induction.

Claim: For all \( n \in \mathbb{N} \), \( F_n = L_{n+1} - F_{n+2} \).

Proof by strong mathematical induction on \( n \) (\( n \geq 0 \)), with \( n_0 = 0 \) and \( n_1 = 1 \).
IH: For all $m \in \mathbb{N}$ s.t. $0 \leq m < n$, $F_m = L_{m+1} - F_{m+2}$.

NTS: $F_n = L_{n+1} - F_{n+2}$.

Consider three cases.

**Basis Case 1:** $n = 0$. $F_0 = 0 = 1 - 1 = L_1 - F_2$.

**Basis Case 2:** $n = 1$. $F_1 = 1 = 3 - 2 = L_2 - F_3$.

**Induction:** $n \geq 2$.

\[
F_n = F_{n-2} + F_{n-1} \quad \text{(defn } F_n, n \geq 2) \\
= (L_{n-1} - F_n) + (L_n - F_{n+1}) \quad \text{(IH, twice)} \\
= (L_{n-1} + L_n) - (F_n + F_{n+1}) \\
= L_{n+1} - F_{n+2} \quad \text{(defn } L_{n+1}, \text{ defn } F_{n+2}, n \geq 1)
\]

2 Assume that for all $n \in \mathbb{Z}$, if $P(m)$ for all $m < n$, then $P(n)$. Can we conclude that for all $n \in \mathbb{Z}$, $P(n)$? Prove your answer correct.

No, the conclusion does not follow. For example, let $P(n)$ stand for the claim that $n$ is odd. Take an arbitrary odd integer $n$. It is not the case that all integers less than $n$ are odd. So the assumption holds for this choice of $P(n)$, but the conclusion does not.

3 Assume that for all $n \in \mathbb{Z}$, if $P(m)$ for all $m < n$, then $P(n)$. Also assume that for some $n \in \mathbb{Z}$, $P(m)$ for all $m < n$. Can we conclude that for all $n \in \mathbb{Z}$, $P(n)$? Prove your answer correct.

Yes, the conclusion follows. Indeed, from the second assumption, we know that for some $n_0 \in \mathbb{Z}$ we have $P(m)$ for all $m < n_0$. So it will suffice to prove $P(n)$ for all $n \geq n_0$, which is easily done, by strong induction on $n$, given the first assumption. [You might want to try this.]
Structural induction problems

4 (Not in text) Recall the function

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

defined recursively as follows (page 164):

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

Let $f$ be a function from $A$ to $B$, and $g$ be a function from $B$ to $C$.

Claim 1: Prove that, for all $L \in \text{lists}(A)$,

$$\text{map}(g \circ f, L) = \text{map}(g, \text{map}(f, L)).$$

Proof by structural induction on $L$.

Basis: $L = \langle \rangle$

$$\text{map}(g \circ f, \langle \rangle) = \langle \rangle$$
$$\quad = \text{map}(g, \langle \rangle) \quad \text{(defn map, basis)}$$
$$\quad = \text{map}(g, \text{map}(f, \langle \rangle)) \quad \text{(defn map)}$$

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

IH: $\text{map}(g \circ f, L) = \text{map}(g, \text{map}(f, L)).$

NTS: $\text{map}(g \circ f, x :: L)) = \text{map}(g, \text{map}(f, x :: L)).$

$$\text{map}(g \circ f, x :: L)) = (g \circ f)(x) :: \text{map}(g \circ f, L)) \quad \text{(defn map)}$$
$$\quad = g(f(x)) :: \text{map}(g \circ f, L)) \quad \text{(defn } \circ)$$
$$\quad = g(f(x)) :: \text{map}(g, \text{map}(f, L)) \quad \text{(IH)}$$
$$\quad = \text{map}(g, f(x) :: \text{map}(f, L)) \quad \text{(defn map)}$$
$$\quad = \text{map}(g, \text{map}(f, x :: L)) \quad \text{(defn map)}$$
5 (Not in text) Recall the definition of the identity function $\text{id}_A$ from a set $A$ to itself (page 98):

$$\text{id}_A(x) = x \text{ for each } x \in A$$

**Claim 2:** Prove that, for all $L \in \text{lists}(A)$,

$$\text{map}(\text{id}_A, L) = L.$$ 

Proof by structural induction on $L$.

*Basis:* $L = \langle \rangle$

$$\text{map}(\text{id}_A, \langle \rangle) = \langle \rangle \quad \text{(defn map, basis)}$$

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

IH: $\text{map}(\text{id}_A, L) = L$.

NTS: $\text{map}(\text{id}_A, x :: L) = x :: L$.

$$\text{map}(\text{id}_A, x :: L) = \text{id}_A(x) :: \text{map}(\text{id}_A, L) \quad \text{(defn map)}$$

$$= x :: \text{map}(\text{id}_A, L) \quad \text{(defn id)}$$

$$= x :: L \quad \text{(IH)}$$

6 (Not in text) Let $f$ be a bijection from $A$ to $B$. Prove that, for all $L \in \text{lists}(A)$ and all $M \in \text{lists}(B)$,

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

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

Proof by Claims 1 and 2 above.

$$\text{map}(f^{-1}, \text{map}(f, L)) = \text{map}(f^{-1} \circ f, L) \quad \text{(Claim 1)}$$

$$= \text{map}(\text{id}_A, L) \quad \text{(defn } f^{-1})$$

$$= L \quad \text{(Claim 2)}$$

$$\text{map}(f, \text{map}(f^{-1}, M)) = \text{map}(f \circ f^{-1}, M) \quad \text{(Claim 1)}$$

$$= \text{map}(\text{id}_B, M) \quad \text{(defn } f^{-1})$$

$$= M \quad \text{(Claim 2)}$$
Consider the function \( \text{cat} : \text{lists}(A) \times \text{lists}(A) \to \text{lists}(A) \) defined recursively as follows (page 163):

\[
\begin{align*}
\text{cat}(\langle \rangle, M) &= M \\
\text{cat}(x :: L, M) &= x :: \text{cat}(L, M)
\end{align*}
\]

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 \( L \in \text{lists}(A) \).

**Basis:** \( \text{cat}(\langle \rangle, \text{cat}(M, N)) = \text{cat}(M, N) \) \hspace{1cm} (defn cat)

\[
= \text{cat}(\text{cat}(\langle \rangle, M), N) \hspace{1cm} \text{(defn cat)}
\]

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

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

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

\[
\begin{align*}
\text{cat}(x :: L, \text{cat}(M, N)) &= x :: \text{cat}(L, \text{cat}(M, N)) \hspace{1cm} \text{(defn cat)} \\
&= x :: \text{cat}(L, \text{cat}(M, N)) \hspace{1cm} \text{(IH)} \\
&= \text{cat}(x :: \text{cat}(L, M), N) \hspace{1cm} \text{(defn cat)} \\
&= \text{cat}(\text{cat}(x :: L, M), N) \hspace{1cm} \text{(defn cat)}
\end{align*}
\]
Let flatten be the function from the binary trees over $A$ to \( \text{lists}(A) \) defined recursively as follows.

\[
\begin{align*}
\text{flatten}(\langle \rangle) &= \langle \rangle \\
\text{flatten}(\langle L, x, R \rangle) &= \text{cat}(\text{flatten}(L), x :: \text{flatten}(R))
\end{align*}
\]

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

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

\[
\begin{align*}
\text{rev}_L(\langle \rangle) &= \langle \rangle \\
\text{rev}_L(x :: L) &= \text{cat}(\text{rev}_L(L), \langle x \rangle)
\end{align*}
\]

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 \( \text{rev}_T \), from the binary trees over \( A \) to the binary trees over \( A \).

\[
\begin{align*}
\text{rev}_T(\langle \rangle) &= \langle \rangle \\
\text{rev}_T(\langle L, x, R \rangle) &= \langle \text{rev}_T(R), x, \text{rev}_T(L) \rangle
\end{align*}
\]

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 $T$.

**Basis:**

$$\text{rev}_L(\text{flatten}(\langle \rangle)) = \text{rev}_L(\langle \rangle) \quad \text{(defn flatten)}$$
$$= \langle \rangle \quad \text{(defn rev)}$$
$$= \text{flatten}(\langle \rangle) \quad \text{(defn flatten)}$$
$$= \text{flatten}(\text{rev}_T(\langle \rangle)) \quad \text{(defn rev)}$$

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

IH: $\text{rev}_L(\text{flatten}(L)) = \text{flatten}(\text{rev}_T(L))$
and $\text{rev}_L(\text{flatten}(R)) = \text{flatten}(\text{rev}_T(R))$.

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

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