Programming Languages
Practicals 3. Definition and Proof by Induction

Shin-Cheng Mu

Spring 2022

1. Prove that length distributes into $(+\!\!+)$:

$$\mathit{length}\ (\mathit{xs} +\!\!+ \mathit{ys}) = \mathit{length}\ \mathit{xs} + \mathit{length}\ \mathit{ys} \quad .$$

Solution: Prove by induction on the structure of xs.

Case $\mathit{xs} := [\,]$:

$$\begin{aligned}
& \mathit{length}\ ([\,] +\!\!+ \mathit{ys}) \\
= & \quad \{ \text{ definition of } (+\!\!+) \ \} \\
& \mathit{length}\ \mathit{ys} \\
= & \quad \{ \text{ definition of } (+) \ \} \\
& 0 + \mathit{length}\ \mathit{ys} \\
= & \quad \{ \text{ definition of } \mathit{length} \ \} \\
& \mathit{length}\ [\,] + \mathit{length}\ \mathit{ys}
\end{aligned}$$

Case $\mathit{xs} := x : \mathit{xs}$:

$$\begin{aligned}
& \mathit{length}\ ((x : \mathit{xs}) +\!\!+ \mathit{ys}) \\
= & \quad \{ \text{ definition of } (+\!\!+) \ \} \\
& \mathit{length}\ (x : (\mathit{xs} +\!\!+ \mathit{ys})) \\
= & \quad \{ \text{ definition of } \mathit{length} \ \} \\
& 1 + \mathit{length}\ (\mathit{xs} +\!\!+ \mathit{ys}) \\
= & \quad \{ \text{ by induction } \} \\
& 1 + \mathit{length}\ \mathit{xs} + \mathit{length}\ \mathit{ys} \\
= & \quad \{ \text{ definition of } \mathit{length} \ \} \\
& \mathit{length}\ (x : \mathit{xs}) + \mathit{length}\ \mathit{ys}
\end{aligned}$$

Note that we in fact omitted one step using the associativity of $(+)$.

2. Prove: $sum \cdot concat = sum \cdot map\ sum$.

Solution: By extensional equality, $sum \cdot concat = sum \cdot map\ sum$ if and only if

$$(sum \cdot concat)\ xss = (sum \cdot map\ sum)\ xss,$$

for all xss, which, by definition of $(\cdot)$, is equivalent to

$$sum\ (concat\ xss) = sum\ (map\ sum\ xss),$$

which we will prove by induction on xss.

Case $xss := [\,]$:

$$\begin{aligned}
& sum\ (concat\ [\,])) \\
= \ & \{ \text{ definition of } concat \ \} \\
& sum\ [\,] \\
= \ & \{ \text{ definition of } map \ \} \\
& sum\ (map\ sum\ [\,])
\end{aligned}$$

Case $xss := xs : xss$:

$$\begin{aligned}
& sum\ (concat\ (xs : xss)) \\
= \ & \{ \text{ definition of } concat \ \} \\
& sum\ (xs +\!\!+ (concat\ xss)) \\
= \ & \{ \text{ lemma: } sum \text{ distributes over } +\!\!+ \ \} \\
& sum\ xs + sum\ (concat\ xss) \\
= \ & \{ \text{ by induction } \} \\
& sum\ xs + sum\ (map\ sum\ xss) \\
= \ & \{ \text{ definition of } sum \ \} \\
& sum\ (sum\ xs : map\ sum\ xss) \\
= \ & \{ \text{ definition of } map \ \} \\
& sum\ (map\ sum\ (xs : xss)).
\end{aligned}$$

The lemma that sum distributes over $+\!\!+$, that is,

$$sum\ (xs +\!\!+ ys) = sum\ xs + sum\ ys,$$

needs a separate proof by induction. Here it goes:

Case $xs := [\,]$:

$$sum\ ([\,] +\!\!+ ys)$$

$$
\begin{aligned}
&= \{ \text{ definition of } (+\!\!+) \ \} \\
&\quad \mathit{sum}\ \mathit{ys} \\
&= \{ \text{ definition of } (+) \ \} \\
&\quad 0 + \mathit{sum}\ \mathit{ys} \\
&= \{ \text{ definition of } \mathit{sum} \ \} \\
&\quad \mathit{sum}\ [\,] + \mathit{sum}\ \mathit{ys}.
\end{aligned}
$$

Case $xs := x : xs$:

$$
\begin{aligned}
&\quad \mathit{sum}\ ((x : xs) +\!\!+ ys) \\
&= \{ \text{ definition of } (+\!\!+) \ \} \\
&\quad \mathit{sum}\ (x : (xs +\!\!+ ys)) \\
&= \{ \text{ definition of } \mathit{sum} \ \} \\
&\quad x + \mathit{sum}\ (xs +\!\!+ ys) \\
&= \{ \text{ induction } \} \\
&\quad x + (\mathit{sum}\ xs + \mathit{sum}\ ys) \\
&= \{ \text{ since } (+) \text{ is associative } \} \\
&\quad (x + \mathit{sum}\ xs) + \mathit{sum}\ ys \\
&= \{ \text{ definition of } \mathit{sum} \ \} \\
&\quad \mathit{sum}\ (x : xs) + \mathit{sum}\ ys.
\end{aligned}
$$

3. Prove: $\mathit{filter}\ p \cdot \mathit{map}\ f = \mathit{map}\ f \cdot \mathit{filter}\ (p \cdot f)$.
Hint: for calculation, it might be easier to use this definition of *filter*:

$$
\begin{aligned}
&\mathit{filter}\ p\ [\,] &&= [\,] \\
&\mathit{filter}\ p\ (x : xs) &&= \mathbf{if}\ p\ x\ \mathbf{then}\ x : \mathit{filter}\ p\ xs \\
& && \quad \mathbf{else}\ \mathit{filter}\ p\ xs
\end{aligned}
$$

and use the law that in the world of total functions we have:

$$
f\ (\mathbf{if}\ q\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2) = \mathbf{if}\ q\ \mathbf{then}\ f\ e_1\ \mathbf{else}\ f\ e_2
$$

You may also carry out the proof using the definition of *filter* using guards:

$$
\begin{aligned}
&\ldots \\
&\mathit{filter}\ p\ (x : xs) &&\mid p\ x = \ldots \\
& &&\mid \mathbf{otherwise} = \ldots
\end{aligned}
$$

You will then have to distinguish between the two cases: $p\ x$ and $\neg\ (p\ x)$, which makes the proof more fragmented. Both proofs are okay, however.

Solution:

$$
\begin{aligned}
& \mathit{filter}\ p \cdot \mathit{map}\ f = \mathit{map}\ f \cdot \mathit{filter}\ (p \cdot f) \\
\equiv\ & \{\ \text{extensional equality}\ \} \\
& (\forall xs :: (\mathit{filter}\ p \cdot \mathit{map}\ f)\ xs = (\mathit{map}\ f \cdot \mathit{filter}\ (p \cdot f))\ xs) \\
\equiv\ & \{\ \text{definition of } (\cdot)\ \} \\
& (\forall xs :: \mathit{filter}\ p\ (\mathit{map}\ f\ xs) = \mathit{map}\ f\ (\mathit{filter}\ (p \cdot f)\ xs)).
\end{aligned}
$$

We proceed by induction on xs.

Case $xs := [\,]$:

$$
\begin{aligned}
& \mathit{filter}\ p\ (\mathit{map}\ f\ [\,]) \\
=\ & \{\ \text{definition of } \mathit{map}\ \} \\
& \mathit{filter}\ p\ [\,] \\
=\ & \{\ \text{definition of } \mathit{filter}\ \} \\
& [\,] \\
=\ & \{\ \text{definition of } \mathit{map}\ \} \\
& \mathit{map}\ f\ [\,] \\
=\ & \{\ \text{definition of } \mathit{filter}\ \} \\
& \mathit{map}\ f\ (\mathit{filter}\ (p \cdot f)\ [\,])
\end{aligned}
$$

Case $xs := x : xs$:

$$
\begin{aligned}
& \mathit{filter}\ p\ (\mathit{map}\ f\ (x : xs)) \\
=\ & \{\ \text{definition of } \mathit{map}\ \} \\
& \mathit{filter}\ p\ (f\ x : \mathit{map}\ f\ xs) \\
=\ & \{\ \text{definition of } \mathit{filter}\ \} \\
& \mathbf{if}\ p\ (f\ x)\ \mathbf{then}\ f\ x : \mathit{filter}\ p\ (\mathit{map}\ f\ xs)\ \mathbf{else}\ \mathit{filter}\ p\ (\mathit{map}\ f\ xs) \\
=\ & \{\ \text{induction hypothesis}\ \} \\
& \mathbf{if}\ p\ (f\ x)\ \mathbf{then}\ f\ x : \mathit{map}\ f\ (filter(p \cdot f)\ xs)\ \mathbf{else}\ \mathit{map}\ f\ (filter\ (p \cdot f)\ xs) \\
=\ & \{\ \text{defintion of } \mathit{map}\ \} \\
& \mathbf{if}\ p\ (f\ x)\ \mathbf{then}\ \mathit{map}\ f\ (x : \mathit{filter}\ (p \cdot f)\ xs)\ \mathbf{else}\ \mathit{map}\ f\ (\mathit{filter}\ (p \cdot f)\ xs) \\
=\ & \{\ \text{since } f\ (\mathbf{if}\ q\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2) = \mathbf{if}\ q\ \mathbf{then}\ f\ e_1\ \mathbf{else}\ f\ e_2\ \} \\
& \mathit{map}\ f\ (\mathbf{if}\ p\ (f\ x)\ \mathbf{then}\ x : \mathit{filter}\ (p \cdot f)\ xs\ \mathbf{else}\ \mathit{filter}\ (p \cdot f)\ xs) \\
=\ & \{\ \text{definition of } (\cdot)\ \} \\
& \mathit{map}\ f\ (\mathbf{if}\ (p \cdot f)\ x\ \mathbf{then}\ x : \mathit{filter}\ (p \cdot f)\ xs\ \mathbf{else}\ \mathit{filter}\ (p \cdot f)\ xs) \\
=\ & \{\ \text{definition of } \mathit{filter}\ \} \\
& \mathit{map}\ f\ (\mathit{filter}\ (p \cdot f)\ (x : xs))
\end{aligned}
$$

4. Reflecting on the law we used in the previous exercise:

$$f\ (\mathbf{if}\ q\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2) = \mathbf{if}\ q\ \mathbf{then}\ f\ e_1\ \mathbf{else}\ f\ e_2$$

Can you think of a counterexample to the law above, when we allow the presence of $\perp$? What additional constraint shall we impose on f to make the law true?

Solution: Let $f = \mathit{const}\ 1$ (where $\mathit{const}\ x\ y = x$), and $q = \perp$. We have:

$$\begin{aligned}
& \mathit{const}\ 1\ (\mathbf{if}\ \perp\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2) \\
= & \quad \{\ \text{definition of } \mathit{const}\ \} \\
& 1 \\
\neq & \perp \\
= & \quad \{\ \mathbf{if} \text{ is strict on the conditional expression}\ \} \\
& \mathbf{if}\ \perp\ \mathbf{then}\ f\ e_1\ \mathbf{else}\ f\ e_2
\end{aligned}$$

The rule is restored if f is strict, that is, $f\ \perp = \perp$.

5. Prove: $\mathit{take}\ n\ xs \mathbin{+\!\!+} \mathit{drop}\ n\ xs = xs$, for all n and xs.

Solution: By induction on n, then induction on xs.

Case $n := 0$

$$\begin{aligned}
& \mathit{take}\ 0\ xs \mathbin{+\!\!+} \mathit{drop}\ 0\ xs \\
= & \quad \{\ \text{definitions of } \mathit{take} \text{ and } \mathit{drop}\ \} \\
& [\,] \mathbin{+\!\!+} xs \\
= & \quad \{\ \text{definition of } (\mathbin{+\!\!+})\ \} \\
& xs.
\end{aligned}$$

Case $n := \mathbf{1}_+\ n$ and $xs := [\,]$

$$\begin{aligned}
& \mathit{take}\ (\mathbf{1}_+\ n)\ [\,] \mathbin{+\!\!+} \mathit{drop}\ (\mathbf{1}_+\ n)\ [\,] \\
= & \quad \{\ \text{definitions of } \mathit{take} \text{ and } \mathit{drop}\ \} \\
& [\,] \mathbin{+\!\!+} [\,] \\
= & \quad \{\ \text{definition of } (\mathbin{+\!\!+})\ \} \\
& [\,].
\end{aligned}$$

Case $n := \mathbf{1}_+\ n$ and $xs := x : xs$

$$\mathit{take}\ (\mathbf{1}_+\ n)\ (x : xs) \mathbin{+\!\!+} \mathit{drop}\ (\mathbf{1}_+\ n)\ (x : xs)$$

$$
\begin{aligned}
&= \quad \{ \text{ definitions of } \mathit{take} \text{ and } \mathit{drop} \ \} \\
&\quad (x : \mathit{take}\ n\ \mathit{xs}) +\!\!+ \mathit{drop}\ n\ \mathit{xs} \\
&= \quad \{ \text{ definition of } (+\!\!+) \ \} \\
&\quad x : \mathit{take}\ n\ \mathit{xs} +\!\!+ \mathit{drop}\ n\ \mathit{xs} \\
&= \quad \{ \text{ induction} \} \\
&\quad x : \mathit{xs}.
\end{aligned}
$$

6. Define a function $\mathit{fan} :: a \to \mathit{List}\ a \to \mathit{List}\ (\mathit{List}\ a)$ such that $\mathit{fan}\ x\ \mathit{xs}$ inserts x into the 0th, 1st...nth positions of xs, where n is the length of xs. For example:

$$\mathit{fan}\ 5\ [1,2,3,4] = [[5,1,2,3,4],[1,5,2,3,4],[1,2,5,3,4],[1,2,3,5,4],[1,2,3,4,5]] \ .$$

Solution:

$$
\begin{aligned}
&\mathit{fan} && :: a \to \mathit{List}\ a \to \mathit{List}\ (\mathit{List}\ a) \\
&\mathit{fan}\ x\ [\,] && = [[x]] \\
&\mathit{fan}\ x\ (y : \mathit{ys}) && = (x : y : \mathit{ys}) : \mathit{map}\ (y :)\ (\mathit{fan}\ x \mathit{ys})
\end{aligned}
$$

7. Prove: $\mathit{map}\ (\mathit{map}\ f) \cdot \mathit{fan}\ x = \mathit{fan}\ (f\ x) \cdot \mathit{map}\ f$, for all f and x. **Hint**: you will need the map-fusion law, and to spot that $\mathit{map}\ f \cdot (y :) = (f\ y :) \cdot \mathit{map}\ f$ (why?).

Solution: This is equivalent to proving that, for all f, x, and xs:

$$\mathit{map}\ (\mathit{map}\ f)\ (\mathit{fan}\ x\ \mathit{xs}) = \mathit{fan}\ (f\ x)\ (\mathit{map}\ f\ \mathit{xs}) \ .$$

Induction on xs.
Case $\mathit{xs} := [\,]$:

$$
\begin{aligned}
&\quad \mathit{map}\ (\mathit{map}\ f)\ (\mathit{fan}\ x\ [\,]) \\
&= \quad \{ \text{ definition of } \mathit{fan} \ \} \\
&\quad \mathit{map}\ (\mathit{map}\ f)\ [[x]] \\
&= \quad \{ \text{ definition of } \mathit{map} \ \} \\
&\quad [[f\ x]] \\
&= \quad \{ \text{ definition of } \mathit{fan} \ \} \\
&\quad \mathit{fan}(f\ x)\ [\,] \\
&= \quad \{ \text{ definition of } \mathit{fan} \ \} \\
&\quad \mathit{fan}\ (f\ x)\ (\mathit{map}\ f\ [\,]) \ .
\end{aligned}
$$

Case $xs := y : ys$:

$$\begin{aligned}
& \mathit{map}\ (\mathit{map}\ f)\ (\mathit{fan}\ x\ (y : ys)) \\
=\ & \quad \{\ \text{definition of } \mathit{fan}\ \} \\
& \mathit{map}\ (\mathit{map}\ f)\ ((x : y : ys) : \mathit{map}\ (y :)\ (\mathit{fan}\ x\ ys)) \\
=\ & \quad \{\ \text{definition of } \mathit{map}\ \} \\
& \mathit{map}\ f\ (x : y : ys) : \mathit{map}\ (\mathit{map}\ f)\ (\mathit{map}\ (y :)\ (\mathit{fan}\ x\ ys))) \\
=\ & \quad \{\ \mathit{map}\text{-fusion}\ \} \\
& \mathit{map}\ f\ (x : y : ys) : \mathit{map}\ (\mathit{map}\ f \cdot (y :))\ (\mathit{fan}\ x\ ys) \\
=\ & \quad \{\ \text{definition of } \mathit{map}\ \} \\
& \mathit{map}\ f\ (x : y : ys) : \mathit{map}\ ((f y :) \cdot \mathit{map}\ f)\ (\mathit{fan}\ x\ ys) \\
=\ & \quad \{\ \mathit{map}\text{-fusion}\ \} \\
& \mathit{map}\ f\ (x : y : ys) : \mathit{map}\ (f y :)\ (\mathit{map}\ (\mathit{map}\ f)\ (\mathit{fan}\ x\ ys)) \\
=\ & \quad \{\ \text{induction}\ \} \\
& \mathit{map}\ f\ (x : y : ys) : \mathit{map}\ (f y :)\ (\mathit{fan}\ (f\ x)\ (\mathit{map}\ f\ ys)) \\
=\ & \quad \{\ \text{definition of } \mathit{map}\ \} \\
& (f\ x : f\ y : \mathit{map}\ f\ ys) : \mathit{map}\ (f y :)\ (\mathit{fan}\ (f\ x)\ (\mathit{map}\ f\ ys)) \\
=\ & \quad \{\ \text{definition of } \mathit{fan}\ \} \\
& \mathit{fan}\ (f\ x)\ (f\ y : \mathit{map}\ f\ ys) \\
=\ & \quad \{\ \text{definition of } \mathit{map}\ \} \\
& \mathit{fan}\ (f\ x)\ (\mathit{map}\ f\ (y : ys))\ \ .
\end{aligned}$$

8. Define $\mathit{perms} :: \mathit{List}\ a \to \mathit{List}\ (\mathit{List}\ a)$ that returns all permutations of the input list. For example:

$$\mathit{perms}\ [1,2,3] = [[1,2,3],[2,1,3],[2,3,1],[1,3,2],[3,1,2],[3,2,1]]\ \ .$$

You will need several auxiliary functions defined in the lectures and in the exercises.

Solution:

$$\begin{aligned}
& \mathit{perms} && :: \mathit{List}\ a \to \mathit{List}\ (\mathit{List}\ a) \\
& \mathit{perms}\ [\,] && = [[\,]] \\
& \mathit{perms}\ (x : xs) && = \mathit{concat}\ (\mathit{map}\ (\mathit{fan}\ x)\ (\mathit{perms}\ xs))
\end{aligned}$$

9. Prove: $\mathit{map}\ (\mathit{map}\ f) \cdot \mathit{perm} = \mathit{perm} \cdot \mathit{map}\ f$. You may need previously proved results, as well as a property about concat and map: for all g, we have $\mathit{map}\ g \cdot \mathit{concat} = \mathit{concat} \cdot \mathit{map}\ (\mathit{map}\ g)$.

Solution: This is equivalent to proving that, for all f and xs:

$$map\ (map\ f)\ (perm\ xs) = perm\ (map\ f\ xs)\ \ .$$

Induction on xs.
Case $xs := [\,]$:

$$\begin{array}{ll} & map\ (map\ f)\ (perm\ [\,]) \\ = & \{\ \text{definition of } perm\ \} \\ & map\ (map\ f)\ [[\,]] \\ = & \{\ \text{definition of } map\ \} \\ & [[\,]] \\ = & \{\ \text{definition of } perm\ \} \\ & perm\ [\,] \\ = & \{\ \text{definition of } map\ \} \\ & perm\ (map\ f\ [\,])\ \ . \end{array}$$

Case $xs := x : xs$:

$$\begin{array}{ll} & map\ (map\ f)\ (perm\ (x : xs)) \\ = & \{\ \text{definition of } perm\ \} \\ & map\ (map\ f)\ (concat\ (map\ (fan\ x)\ (perm\ xs))) \\ = & \{\ \text{since } map\ g \cdot concat = concat \cdot map\ (map\ g)\ \} \\ & concat\ (map\ (map\ (map\ f))(map\ (fan\ x)\ (perm\ xs))) \\ = & \{\ map\text{-fusion}\ \} \\ & concat\ (map\ (map\ (map\ f) \cdot fan\ x)\ (perm\ xs)) \\ = & \{\ \text{previous exercise}\ \} \\ & concat\ (map\ (fan\ (f\ x) \cdot map\ f)\ (perm\ xs)) \\ = & \{\ map\text{-fusion}\ \} \\ & concat\ (map\ (fan\ (f\ x))\ (map\ (map\ f)\ (perm\ xs))) \\ = & \{\ \text{induction}\ \} \\ & concat\ (map\ (fan\ (f\ x))\ (perm\ (map\ f\ xs))) \\ = & \{\ \text{definition of } perm\ \} \\ & perm\ (f\ x : map\ f\ xs) \\ = & \{\ \text{definition of } map\ \} \\ & perm\ (map\ f\ (x : xs))\ \ . \end{array}$$

10. Define $inits :: List\ a \rightarrow List\ (List\ a)$ that returns all prefixes of the input list.

$$inits\ \texttt{"abcde"} = [\texttt{""}, \texttt{"a"}, \texttt{"ab"}, \texttt{"abc"}, \texttt{"abcd"}, \texttt{"abcde"}].$$

Hint: the empty list has *one* prefix: the empty list. The solution has been given in the lecture. Please try it again yourself.

Solution:

$$
\begin{aligned}
&\mathit{inits} && :: \mathit{List}\ a \to \mathit{List}\ (\mathit{List}\ a) \\
&\mathit{inits}\ [\,] && = [[\,]] \\
&\mathit{inits}\ (x : xs) && = [\,] : \mathit{map}\ (x :)\ (\mathit{inits}\ xs)\ \ .
\end{aligned}
$$

11. Define $\mathit{tails} :: \mathit{List}\ a \to \mathit{List}\ (\mathit{List}\ a)$ that returns all suffixes of the input list.

    tails "abcde" $=$ ["abcde", "bcde", "cde", "de", "e", ""].

    Hint: the empty list has *one* suffix: the empty list. The solution has been given in the lecture. Please try it again yourself.

Solution:

$$
\begin{aligned}
&\mathit{tails} && :: \mathit{List}\ a \to \mathit{List}\ (\mathit{List}\ a) \\
&\mathit{tails}\ [\,] && = [[\,]] \\
&\mathit{tails}\ (x : xs) && = (x : xs) : \mathit{tails}\ xs\ \ .
\end{aligned}
$$

12. The function $\mathit{splits} :: \mathit{List}\ a \to \mathit{List}\ (\mathit{List}\ a, \mathit{List}\ a)$ returns all the ways a list can be split into two. For example,

$$
\begin{aligned}
\mathit{splits}\ [1,2,3,4]\ \ =\ \ & [([\,],[1,2,3,4]),([1],[2,3,4]),([1,2],[3,4]), \\
& ([1,2,3],[4]),([1,2,3,4],[\,])]\ \ .
\end{aligned}
$$

Define splits inductively on the input list. **Hint**: you may find it useful to define, in a **where**-clause, an auxiliary function $f\ (ys, zs) = \ldots$ that matches pairs. Or you may simply use $(\lambda\ (ys, zs) \to \ldots)$.

Solution:

$$
\begin{aligned}
&\mathit{splits} && ::\ \ \mathit{List}\ a \to \mathit{List}\ (\mathit{List}\ a, \mathit{List}\ a) \\
&\mathit{splits}\ [\,] && =\ \ [([\,],[\,])] \\
&\mathit{splits}\ (x : xs) && =\ \ ([\,], x : xs) : \mathit{map}\ \mathit{cons1}\ (\mathit{splits}\ xs)\ \ , \\
&\quad\ \mathbf{where}\ \mathit{cons1}\ (ys, zs) = (x : ys, zs)\ \ .
\end{aligned}
$$

If you know how to use λ expressions, you may:

$$
\begin{aligned}
&\mathit{splits} && ::\ \ \mathit{List}\ a \to \mathit{List}\ (\mathit{List}\ a, \mathit{List}\ a) \\
&\mathit{splits}\ [\,] && =\ \ [([\,],[\,])] \\
&\mathit{splits}\ (x : xs) && =\ \ ([\,], x : xs) : \mathit{map}\ (\lambda\ (ys, zs) \to (x : ys, zs))\ (\mathit{splits}\ xs)\ \ .
\end{aligned}
$$

13. An *interleaving* of two lists xs and ys is a permutation of the elements of both lists such that the members of xs appear in their original order, and so does the members of ys. Define $interleave :: List\ a \rightarrow List\ a \rightarrow List\ (List\ a)$ such that $interleave\ xs\ ys$ is the list of interleaving of xs and ys. For example, $interleave\ [1,2,3]\ [4,5]$ yields:

$$[[1,2,3,4,5],[1,2,4,3,5],[1,2,4,5,3],[1,4,2,3,5],[1,4,2,5,3],\\ [1,4,5,2,3],[4,1,2,3,5],[4,1,2,5,3],[4,1,5,2,3],[4,5,1,2,3]].$$

Solution:

$$\begin{array}{lcl} interleave & :: & List\ a \rightarrow List\ a \rightarrow List\ (List\ a) \\ interleave\ [\,]\ ys & = & [ys] \\ interleave\ xs\ [\,] & = & [xs] \\ interleave\ (x : xs)\ (y : ys) & = & map\ (x :)\ (interleave\ xs\ (y : ys)) \mathbin{+\!\!+} \\ & & \quad map\ (y :)\ (interleave\ (x : xs)\ ys)\ \ . \end{array}$$

14. A list ys is a *sublist* of xs if we can obtain ys by removing zero or more elements from xs. For example, $[2,4]$ is a sublist of $[1,2,3,4]$, while $[3,2]$ is *not*. The list of all sublists of $[1,2,3]$ is:

$$[[],[3],[2],[2,3],[1],[1,3],[1,2],[1,2,3]].$$

Define a function $sublist :: List\ a \rightarrow List\ (List\ a)$ that computes the list of all sublists of the given list. **Hint**: to form a sublist of xs, each element of xs could either be kept or dropped.

Solution:

$$\begin{array}{lcl} sublist & :: & List\ a \rightarrow List\ (List\ a) \\ sublist\ [\,] & = & [[\,]] \\ sublist\ (x : xs) & = & xss \mathbin{+\!\!+} map\ (x :)\ xss\ \ , \\ & & \mathbf{where}\ xss = sublist\ xs\ \ . \end{array}$$

The righthand side could be $sublist\ xs \mathbin{+\!\!+} map\ (x :)\ (sublist\ xs)$ (but it could be much slower).

15. Consider the following datatype for internally labelled binary trees:

$$\mathbf{data}\ Tree\ a\ =\ \mathsf{Null} \mid \mathsf{Node}\ a\ (Tree\ a)\ (Tree\ a)\ \ .$$

(a) Given $(\downarrow) :: Nat \rightarrow Nat \rightarrow Nat$, which yields the smaller one of its arguments, define $minT :: Tree\ Nat \rightarrow Nat$, which computes the minimal element in a tree. (Note: $(\downarrow)$ is actually called min in the standard library. In the lecture we use the symbol $(\downarrow)$ to be brief.)

Solution:

$$
\begin{aligned}
&\mathit{minT} && :: \mathit{Tree}\ \mathit{Nat} \to \mathit{Nat}\\
&\mathit{minT}\ \mathsf{Null} && = \mathit{maxBound}\\
&\mathit{minT}\ (\mathsf{Node}\ x\ t\ u) && = x \downarrow \mathit{minT}\ t \downarrow \mathit{minT}\ u\ \ .
\end{aligned}
$$

(b) Define $\mathit{mapT} :: (a \to b) \to \mathit{Tree}\ a \to \mathit{Tree}\ b$, which applies the functional argument to each element in a tree.

Solution:

$$
\begin{aligned}
&\mathit{mapT} && :: (a \to b) \to \mathit{Tree}\ a \to \mathit{Tree}\ b\\
&\mathit{mapT}\ f\ \mathsf{Null} && = \mathsf{Null}\\
&\mathit{mapT}\ f\ (\mathsf{Node}\ x\ t\ u) && = \mathsf{Node}\ (f\ x)\ (\mathit{mapT}\ f\ t)\ (\mathit{mapT}\ f\ u)\ \ .
\end{aligned}
$$

(c) Can you define $(\downarrow)$ inductively on Nat?

Solution:

$$
\begin{aligned}
&(\downarrow) && :: \mathit{Nat} \to \mathit{Nat} \to \mathit{Nat}\\
&0 \downarrow n && = 0\\
&(\mathbf{1}_{+} m) \downarrow 0 && = 0\\
&(\mathbf{1}_{+} m) \downarrow (\mathbf{1}_{+} n) && = \mathbf{1}_{+}\ (m \downarrow n)\ \ .
\end{aligned}
$$

(d) Prove that for all n and t, $\mathit{minT}\ (\mathit{mapT}\ (n+)\ t) = n + \mathit{minT}\ t$. That is, $\mathit{minT} \cdot \mathit{mapT}\ (n+) = (n+) \cdot \mathit{minT}$.

Solution: Induction on t.

Case $t := \mathsf{Null}$. Omitted.

Case $t := \mathsf{Node}\ x\ t\ u$.

$$
\begin{aligned}
& \mathit{minT}\ (\mathit{mapT}\ (n+)\ (\mathsf{Node}\ x\ t\ u))\\
=\ & \{\ \text{definition of } \mathit{mapT}\ \}\\
& \mathit{minT}\ (\mathsf{Node}\ (n+x)\ (\mathit{mapT}\ (n+)\ t)\ (\mathit{mapT}\ (n+)\ u))\\
=\ & \{\ \text{definition of } \mathit{minT}\ \}\\
& (n+x) \downarrow \mathit{minT}\ (\mathit{mapT}\ (n+)\ t)) \downarrow \mathit{minT}\ (\mathit{mapT}\ (n+)\ u)\\
=\ & \{\ \text{by induction}\ \}\\
& (n+x) \downarrow (n + \mathit{minT}\ t) \downarrow (n + \mathit{minT}\ u)\\
=\ & \{\ \text{lemma: } (n+x) \downarrow (n+y) = n + (x \downarrow y)\ \}
\end{aligned}
$$

$$
\begin{aligned}
& \quad n + (x \downarrow minT\ t \downarrow minT\ u) \\
= & \quad \{ \text{ definition of } minT \ \} \\
& \quad n + minT\ (\mathsf{Node}\ x\ t\ u) \ \ .
\end{aligned}
$$

The lemma $(n + x) \downarrow (n + y) = n + (x \downarrow y)$ can be proved by induction on n, using inductive definitions of $(+)$ and $(\downarrow)$.