Programming Languages: Functional Programming Practicals 4. Program Calculation

Shin-Cheng Mu

Spring 2022

1. Let $descend$ be defined by:

$$\begin{array}{l} descend :: \mathsf{Nat} \rightarrow \mathsf{List\ Nat} \\ descend\ 0 = [\,] \\ descend\ (\mathbf{1}_+\ n) = \mathbf{1}_+\ n : descend\ n \quad . \end{array}$$

(a) Let $sumseries = sum \cdot descend$. Synthesise an inductive definition of $sumseries$.

Solution: It is immediate that $sum\ (descend\ 0) = 0$. For the inductive case we calculate:

$$\begin{array}{ll} & sum\ (descend\ (\mathbf{1}_+\ n)) \\ = & \quad \{ \text{ definition of } descend \ \} \\ & sum\ ((\mathbf{1}_+\ n) : descend\ n) \\ = & \quad \{ \text{ definition of } sum \ \} \\ & (\mathbf{1}_+\ n) + sum\ (descend\ n) \\ = & \quad \{ \text{ definition of } sumseries \ \} \\ & (\mathbf{1}_+\ n) + sumseries\ n \quad . \end{array}$$

Thus we have

$$\begin{array}{ll} sumseries\ 0 & = 0 \\ sumseries\ (\mathbf{1}_+\ n) & = (\mathbf{1}_+\ n) + sumseries\ n \quad . \end{array}$$

(b) The function $repeatN :: (\mathsf{Nat}, a) \rightarrow \mathsf{List}\ a$ is defined by

$$repeatN\ (n, x) = map\ (const\ x)\ (descend\ n) \quad .$$

Thus $repeatN\ (n, x)$ produces n copies of x in a list. E.g. $repeatN\ (3, \texttt{'a'}) = \texttt{"aaa"}$. Calculate an inductive definition of $repeatN$.

Solution: It is immediate that $\mathit{repeatN}\ (0, x) = [\,]$. For the inductive case we calculate

$$
\begin{aligned}
& \mathit{repeatN}\ (\mathbf{1}_{+}\ n, x) \\
= \quad & \{\ \text{definition of } \mathit{repeatN}\ \} \\
& \mathit{map}\ (\mathit{const}\ x)\ (\mathit{descend}\ (\mathbf{1}_{+}\ n)) \\
= \quad & \{\ \text{definition of } \mathit{descend}\ \} \\
& \mathit{map}\ (\mathit{const}\ x)\ (\mathbf{1}_{+}\ n : \mathit{descend}\ n) \\
= \quad & \{\ \text{definition of } \mathit{map} \text{ and } \mathit{const}\ \} \\
& x : \mathit{map}\ (\mathit{const}\ x)\ (\mathit{descend}\ n) \\
= \quad & \{\ \text{definition of } \mathit{repeatN}\ \} \\
& x : \mathit{repeatN}\ (n, x)\ \ .
\end{aligned}
$$

Thus we have

$$
\begin{aligned}
\mathit{repeatN}\ (0, \quad\ \ x) &= [\,] \\
\mathit{repeatN}\ (\mathbf{1}_{+}\ n, x) &= x : \mathit{repeatN}\ (n, x)\ \ .
\end{aligned}
$$

(c) The function $\mathit{rld} :: \mathsf{List}\ (\mathsf{Nat}, a) \rightarrow \mathsf{List}\ a$ performs run-length decoding:

$$\mathit{rld} = \mathit{concat} \cdot \mathit{map}\ \mathit{repeatN}\ \ .$$

For example, $\mathit{rld}\ [(2, \texttt{'a'}), (3, \texttt{'b'}), (1, \texttt{'c'})] =$ `"aabbbc"`. Come up with an inductive defintion of rld.

Solution: For the base case:

$$
\begin{aligned}
& \mathit{rld}\ [\,] \\
= \quad & \{\ \text{definition of } \mathit{rld}\ \} \\
& \mathit{concat}\ (\mathit{map}\ \mathit{repeatN}\ [\,]) \\
= \quad & \{\ \text{definitions of } \mathit{map} \text{ and } \mathit{concat}\ \} \\
& [\,]
\end{aligned}
$$

For the inductive case:

$$
\begin{aligned}
& \mathit{rld}\ ((n, x) : \mathit{xs}) \\
= \quad & \{\ \text{definition of } \mathit{rld}\ \} \\
& \mathit{concat}\ (\mathit{map}\ \mathit{repeatN}\ ((n, x) : \mathit{xs})) \\
= \quad & \{\ \text{definitions of } \mathit{map}\ \} \\
& \mathit{concat}\ (\mathit{repeatN}\ (n, x) : \mathit{map}\ \mathit{repeatN}\ \mathit{xs}) \\
= \quad & \{\ \text{definitions of } \mathit{concat}\ \} \\
& \mathit{repeatN}\ (n, x) +\!\!+ \mathit{concat}\ (\mathit{map}\ \mathit{repeatN}\ \mathit{xs}) \\
= \quad & \{\ \text{definition of } \mathit{rld}\ \} \\
& \mathit{repeatN}\ (n, x) +\!\!+ \mathit{rld}\ \mathit{xs}\ \ .
\end{aligned}
$$

We have thus derived:

$$
\begin{aligned}
&\mathit{rld}\ [\,] &&= [\,]\\
&\mathit{rld}\ ((n,x):\mathit{xs}) &&= \mathit{repeatN}\ (n,x) \mathbin{+\!\!+} \mathit{rld}\ \mathit{xs}\ \ .
\end{aligned}
$$

We can in fact further construct a definition of rld that does not use $(+\!\!+)$, by pattern matching on n. It is immediate that $\mathit{rld}\ ((0,x):\mathit{xs}) = \mathit{rld}\ \mathit{xs}$. By a routine calculation we get:

$$
\begin{aligned}
&\mathit{rld}\ [\,] &&= [\,]\\
&\mathit{rld}\ ((0, \quad x):\mathit{xs}) &&= \mathit{rld}\ \mathit{xs}\ \ .\\
&\mathit{rld}\ ((\mathbf{1}_{+}\ n,x):\mathit{xs}) &&= x : \mathit{rld}\ ((n,x):\mathit{xs})\ \ .
\end{aligned}
$$

2. There is another way to define pos such that $\mathit{pos}\ x\ \mathit{xs}$ yields the index of the first occurrence of x in xs:

$$
\begin{aligned}
&\mathit{pos} :: \mathsf{Eq}\ a \Rightarrow a \rightarrow \mathsf{List}\ a \rightarrow \mathsf{Int}\\
&\mathit{pos}\ x = \mathit{length} \cdot \mathit{takeWhile}\ (x \not=)
\end{aligned}
$$

(This pos behaves differently from the one in the lecture when x does not occur in xs.) Construct an inductive definition of pos.

Solution: It is immediate that $\mathit{pos}\ x\ [\,] = 0$. For the inductive case we calculate:

$$
\begin{aligned}
&\quad \mathit{pos}\ x\ (y:\mathit{ys})\\
= &\quad \{\ \text{definition of } \mathit{pos}\ \}\\
&\quad \mathit{length}\ (\mathit{takeWhile}\ (x \not=)\ (y:\mathit{ys}))\\
= &\quad \{\ \text{definition of } \mathit{takeWhile}\ \}\\
&\quad \mathit{length}\ (\mathbf{if}\ x \not= y\ \mathbf{then}\ y : \mathit{takeWhile}\ (x \not=)\ \mathit{ys}\ \mathbf{else}\ [\,])\\
= &\quad \{\ \text{function application distributes into } \mathbf{if}\text{, defn. of } \mathit{length}\ \}\\
&\quad \mathbf{if}\ x \not= y\ \mathbf{then}\ \mathbf{1}_{+}\ (\mathit{length}\ (\mathit{takeWhile}\ (x \not=)\ \mathit{ys}))\ \mathbf{else}\ 0\\
= &\quad \{\ \text{definition of } \mathit{pos}\ \}\\
&\quad \mathbf{if}\ x \not= y\ \mathbf{then}\ \mathbf{1}_{+}\ (\mathit{pos}\ x\ \mathit{ys})\ \mathbf{else}\ 0\ \ .
\end{aligned}
$$

Thus we have constructed:

$$
\begin{aligned}
&\mathit{pos}\ x\ [\,] &&= 0\\
&\mathit{pos}\ x\ (y:\mathit{xs}) &&= \mathbf{if}\ x \not= y\ \mathbf{then}\ \mathbf{1}_{+}\ (\mathit{pos}\ x\ \mathit{xs})\ \mathbf{else}\ 0\ \ .
\end{aligned}
$$

3. Zipping and mapping.

(a) Let $second\ f\ (x, y) = (x, f\ y)$. Prove that $zip\ xs\ (map\ f\ ys) = map\ (second\ f)\ (zip\ xs\ ys)$.

Solution: Recall one of the possible definitions of zip:

$$
\begin{aligned}
&zip\ [\,]\ \ ys &&= [\,]\\
&zip\ (x : xs)\ [\,] &&= [\,]\\
&zip\ (x : xs)\ (y : ys) &&= (x, y) : zip\ xs\ ys \quad .
\end{aligned}
$$

Following the structure, we prove the proposition by induction on xs and ys. A tip for equational reasoning: it is usually easier to go from the more complex side to the simpler side, from the side with more structure to the side with less structure. Thus we start from the left-hand side.

Case $xs := [\,]$.

$$
\begin{aligned}
&\ map\ (second\ f)\ (zip\ [\,]\ ys)\\
=&\quad \{\ \text{definition of } zip\ \}\\
&\ map\ (second\ f)\ [\,]\\
=&\quad \{\ \text{definition of } map\ \}\\
&\ [\,]\\
=&\quad \{\ \text{definition of } zip\ \}\\
&\ zip\ [\,]\ (map\ f\ ys) \quad .
\end{aligned}
$$

Case $xs := x : xs$, $ys := [\,]$:

$$
\begin{aligned}
&\ map\ (second\ f)\ (zip\ (x : xs)\ [\,])\\
=&\quad \{\ \text{definiton of } zip\ \}\\
&\ map\ (second\ f)\ [\,]\\
=&\quad \{\ \text{definition of } map\ \}\\
&\ [\,]\\
=&\quad \{\ \text{definition of } zip\ \}\\
&\ zip\ (x : xs)\ [\,]\\
=&\quad \{\ \text{definition of } map\ \}\\
&\ zip\ (x : xs)\ (map\ f\ [\,]) \quad .
\end{aligned}
$$

Case $xs := x : xs$, $ys := y : ys$:

$$
\begin{aligned}
&\ map\ (second\ f)\ (zip\ (x : xs)\ (y : ys))\\
=&\quad \{\ \text{definition of } zip\ \}\\
&\ map\ (second\ f)\ ((x, y) : zip\ xs\ ys)\\
=&\quad \{\ \text{definition of } map\ \}\\
&\ second\ f\ (x, y) : map\ (second\ f)\ (zip\ xs\ ys)\\
=&\quad \{\ \text{definition of } second\ \}\\
&\ (x, f\ y) : map\ (second\ f)\ (zip\ xs\ ys)
\end{aligned}
$$

$$
\begin{aligned}
&= \quad \{ \text{ induction } \} \\
&\quad (x, f\ y) : \mathit{zip}\ \mathit{xs}\ (\mathit{map}\ f\ \mathit{ys}) \\
&= \quad \{ \text{ definiton of } \mathit{zip} \} \\
&\quad \mathit{zip}\ (x : \mathit{xs})\ (f\ y : \mathit{map}\ f\ \mathit{ys}) \\
&= \quad \{ \text{ definition of } \mathit{map} \} \\
&\quad \mathit{zip}\ (x : \mathit{xs})\ (\mathit{map}\ f\ (y : \mathit{ys}))\ \ .
\end{aligned}
$$

(b) Consider the following definition

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

such that

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

That is, each element in the input list is deleted in turns. Let $\mathit{select} :: \mathsf{List}\ a \to \mathsf{List}\ (a, \mathsf{List}\ a)$ be defined by $\mathit{select}\ \mathit{xs} = \mathit{zip}\ \mathit{xs}\ (\mathit{delete}\ \mathit{xs})$. Come up with an inductive definition of select. **Hint**: you may find second useful.

Solution: The base case $[\,]$ is immediate. For the inductive case:

$$
\begin{aligned}
&\quad \mathit{select}\ (x : \mathit{xs}) \\
&= \quad \{ \text{ definition of } \mathit{select} \} \\
&\quad \mathit{zip}\ (x : \mathit{xs})\ (\mathit{delete}\ (x : \mathit{xs})) \\
&= \quad \{ \text{ definition of } \mathit{delete} \} \\
&\quad \mathit{zip}\ (x : \mathit{xs})\ (\mathit{xs} : \mathit{map}\ (x{:})\ (\mathit{delete}\ \mathit{xs})) \\
&= \quad \{ \text{ definition of } \mathit{zip} \} \\
&\quad (x, \mathit{xs}) : \mathit{zip}\ \mathit{xs}\ (\mathit{map}\ (x{:})\ (\mathit{delete}\ \mathit{xs})) \\
&= \quad \{ \text{ property proved above } \} \\
&\quad (x, \mathit{xs}) : \mathit{map}\ (\mathit{second}\ (x{:}))\ (\mathit{zip}\ \mathit{xs}\ (\mathit{delete}\ \mathit{xs})) \\
&= \quad \{ \text{ definition of } \mathit{select} \} \\
&\quad (x, \mathit{xs}) : \mathit{map}\ (\mathit{second}\ (x{:}))\ (\mathit{select}\ \mathit{xs})\ \ .
\end{aligned}
$$

We thus have

$$
\begin{aligned}
&\mathit{select}\ [\,] && = [\,] \\
&\mathit{select}\ (x : \mathit{xs}) && = (x, \mathit{xs}) : \mathit{map}\ (\mathit{second}\ (x{:}))\ (\mathit{select}\ \mathit{xs})\ \ .
\end{aligned}
$$

(c) An alternative specification of delete is

$$
\begin{aligned}
&\mathit{delete}\ \mathit{xs} = \mathit{map}\ (\mathit{del}\ \mathit{xs})\ [0\mathinner{..}\mathit{length}\ \mathit{xs} - 1] \\
&\quad \mathbf{where}\ \mathit{del}\ \mathit{xs}\ i = \mathit{take}\ i\ \mathit{xs} \mathbin{+\!\!\!+} \mathit{drop}\ (1 + i)\ \mathit{xs}\ \ ,
\end{aligned}
$$

(here we take advantage of the fact that $[0\,..\,n]$ returns $[\,]$ when n is negative). From this specification, derive the inductive definition of $delete$ given above. **Hint**: you may need the following property:

$$[0\,..\,n] = 0 : map\ (\mathbf{1}_+)\ [0\,..\,n-1], \quad \text{if } n \geqslant 0, \tag{1}$$

and the *map-fusion* law (2) given below.

Solution:

$$\begin{aligned}
& delete\ (x : xs) \\
= \quad & \{ \text{ definition of } delete \ \} \\
& map\ (del\ (x : xs))\ [0\,..\,length\ (x : xs) - 1] \\
= \quad & \{ \text{ defintion of } length\text{, arithmetics } \} \\
& map\ (del\ (x : xs))\ [0\,..\,length\ xs] \\
= \quad & \{ \ length\ xs \geqslant 0\text{, by (1) } \} \\
& map\ (del\ (x : xs))\ (0 : map\ (\mathbf{1}_+)\ [0\,..\,length\ xs - 1]) \\
= \quad & \{ \text{ definition of } map \ \} \\
& del\ (x : xs)\ 0 : map\ (del\ (x : xs))\ (map\ (\mathbf{1}_+)\ [0\,..\,length\ xs - 1]) \\
= \quad & \{ \text{ map fusion (2) } \} \\
& del\ (x : xs)\ 0 : map\ (del\ (x : xs) \cdot (\mathbf{1}_+))\ [0\,..\,length\ xs - 1]
\end{aligned}$$

Now we pause for a while to inspect $del\ (x : xs)$. Apparently, $del\ (x : xs)\ 0 = xs$. For $del\ (x : xs) \cdot (\mathbf{1}_+)$ we calculate:

$$\begin{aligned}
& (del\ (x : xs) \cdot (\mathbf{1}_+))\ i \\
= \quad & \{ \text{ definition of } (\cdot) \ \} \\
& del\ (x : xs)\ (\mathbf{1}_+\ i) \\
= \quad & \{ \text{ definition of } del \ \} \\
& take\ (\mathbf{1}_+\ i)\ (x : xs) \mathbin{+\!\!+} drop\ (\mathbf{1}_+\ (\mathbf{1}_+\ i))\ (x : xs) \\
= \quad & \{ \text{ definitions of } take \text{ and } drop \ \} \\
& x : take\ i\ xs \mathbin{+\!\!+} drop\ (\mathbf{1}_+\ i)\ xs \\
= \quad & \{ \text{ definition of } del \ \} \\
& x : del\ xs\ i \\
= \quad & \{ \text{ definition of } (\cdot) \ \} \\
& ((x{:}) \cdot del\ xs)\ i \ \ .
\end{aligned}$$

We resume the calculation:

$$\begin{aligned}
& del\ (x : xs)\ 0 : map\ (del\ (x : xs) \cdot (\mathbf{1}_+))\ [0\,..\,length\ xs - 1] \\
= \quad & \{ \text{ calculation above } \} \\
& xs : map\ ((x{:}) \cdot del\ xs)\ [0\,..\,length\ xs - 1] \\
= \quad & \{ \text{ map fusion (2) } \} \\
& xs : map\ (x{:})\ (map\ (del\ xs)\ [0\,..\,length\ xs - 1]) \\
= \quad & \{ \text{ definition of } delete \ \} \\
& xs : map\ (x{:})\ (delete\ xs) \ \ .
\end{aligned}$$

We have thus derived the first, inductive definition of $delete$.

4. Prove the following *map-fusion* law:

$$map\ f \cdot map\ g = map\ (f \cdot g) \quad . \tag{2}$$

Solution: To find out how to conduct induction:

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

We prove the proposition by induction on xs.

Case $xs := [\,]$. Omitted.

Case $xs := x : xs$.

$$\begin{aligned}
& map\ f\ (map\ g\ (x : xs)) \\
= \quad & \{ \text{definition of } map\text{, twice} \} \\
& f\ (g\ x) : map\ f\ (map\ g\ xs) \\
= \quad & \{ \text{induction} \} \\
& f\ (g\ x) : map\ (f \cdot g)\ xs \\
= \quad & \{ \text{definition of } (\cdot) \} \\
& (f \cdot g)\ x : map\ (f \cdot g)\ xs \\
= \quad & \{ \text{definition of } map \} \\
& map\ (f \cdot g)\ (x : xs) \quad .
\end{aligned}$$

5. Assume that multiplication $(\times)$ is a constant-time operation. One possible definition for $exp\ m\ n = m^n$ could be:

$$\begin{aligned}
& exp :: \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat} \\
& exp\ m\ 0 \qquad\ \ = 1 \\
& exp\ m\ (\mathbf{1}_+\ n) = m \times exp\ m\ n \quad .
\end{aligned}$$

Therefore, to compute $exp\ m\ n$, multiplication is called n times: $m \times m \ldots m \times 1$. Can we do better? Yet another way to represent a natural number is to use the binary representation.

(a) The function $binary :: \mathsf{Nat} \to \mathsf{List\ Bool}$ returns the *reversed* binary representation of a natural number. For example:

$$\begin{aligned}
& binary\ 0 = [\,] \quad , \\
& binary\ 1 = [\mathsf{T}] \quad ,
\end{aligned}$$

$$binary\ 2 = [\mathsf{F}, \mathsf{T}]\ ,$$
$$binary\ 3 = [\mathsf{T}, \mathsf{T}]\ ,$$
$$binary\ 4 = [\mathsf{F}, \mathsf{F}, \mathsf{T}]\ ,$$

where T and F abbreviates True and False. Given the following functions:

$even :: Nat \to Bool$, returning true iff the input is even,

$odd :: Nat \to Bool$, returning true iff the input is odd, and

$div :: Nat \to Nat \to Nat$, for integral division,

define $binary$. You may just present the code.
Hint One possible implementation discriminates between 3 cases – the input is 0, the input is odd, and the input is even.

Solution:

$$\begin{array}{ll} binary\ 0 & = [\,] \\ binary\ n \mid even\ n & = \mathsf{F} : binary\ (n\ \text{`}div\text{`}\ 2) \\ \mid odd\ n & = \mathsf{T} : binary\ (n\ \text{`}div\text{`}\ 2)\ . \end{array}$$

(b) Briefly explain in words whether your implementation of $binary$ terminates for all input in Nat, and why.

Solution: All non-zero natural numbers strictly decreases when being divided by 2, and thus we eventually reaches the base case for 0.

(c) Define a function $decimal ::$ List Bool $\to$ Nat that takes the reversed binary representation and returns the corresponding natural number. E.g. $decimal\ [\mathsf{T}, \mathsf{T}, \mathsf{F}, \mathsf{T}] = 11$. You may just present the code.

Solution:

$$\begin{array}{ll} decimal\ [\,] & = 0 \\ decimal\ (c : cs) & = \mathbf{if}\ c\ \mathbf{then}\ 1 + 2 \times decimal\ cs\ \mathbf{else}\ 2 \times decimal\ cs\ . \end{array}$$

Or equivalently,

$$\begin{array}{ll} decimal\ [\,] & = 0 \\ decimal\ (\mathsf{False} : cs) & = 2 \times decimal\ cs \\ decimal\ (\mathsf{True} : cs) & = 1 + 2 \times decimal\ cs\ . \end{array}$$

(d) Let $roll\ m = exp\ m \cdot decimal$. Assuming we have proved that $exp\ m\ n$ satisfies all arithmetic laws for m^n. Construct (with algebraic calculation) a definition of $roll$ that does not make calls to exp or $decimal$.

Solution: Let's calculate $roll\ m\ xs = exp\ m\ (decimal\ xs)$ by distinguishing between the three cases of xs:

Case $xs := [\,]$:

$$
\begin{aligned}
& roll\ m\ [\,] \\
= {} & exp\ m\ (decimal\ [\,]) \\
= {} & \quad \{\ \text{definition of } decimal\ \} \\
& exp\ m\ 0 \\
= {} & \quad \{\ \text{definition of } exp\ \} \\
& 1\ \ .
\end{aligned}
$$

Case $xs := \mathsf{False} : xs$:

$$
\begin{aligned}
& roll\ m\ (\mathsf{False} : xs) \\
= {} & \quad \{\ \text{definition of } roll\ \} \\
& exp\ m\ (decimal\ (\mathsf{False} : xs)) \\
= {} & \quad \{\ \text{definition of } decimal\ \} \\
& exp\ m\ (2 \times decimal\ xs) \\
= {} & \quad \{\ \text{arithmetic: } m^{2n} = (m^2)^n\ \} \\
& exp\ (m \times m)\ (decimal\ xs) \\
= {} & \quad \{\ \text{definition of } roll\ \} \\
& roll\ (m \times m)\ xs\ \ .
\end{aligned}
$$

Case $xs := \mathsf{True} : xs$:

$$
\begin{aligned}
& roll\ m\ (\mathsf{True} : xs) \\
= {} & \quad \{\ \text{definition of } roll\ \} \\
& exp\ m\ (decimal\ (\mathsf{True} : xs)) \\
= {} & \quad \{\ \text{definition of } decimal\ \} \\
& exp\ m\ (1 + 2 \times decimal\ xs) \\
= {} & \quad \{\ \text{definition of } exp\ \} \\
& m \times exp\ m\ (2 \times decimal\ xs) \\
= {} & \quad \{\ \text{arithmetic: } m^{2n} = (m^2)^n\ \} \\
& m \times exp\ (m \times m)\ (decimal\ xs) \\
= {} & \quad \{\ \text{definition of } roll\ \} \\
& m \times roll\ (m \times m)\ xs\ \ .
\end{aligned}
$$

We have thus constructed:

$$
\begin{aligned}
& roll\ m\ [\,] && = 1 \\
& roll\ m\ (\mathsf{False} : cs) && = roll\ (m \times m)\ xs \\
& roll\ m\ (\mathsf{True}\ : cs) && = m \times roll\ (m \times m)\ xs\ \ .
\end{aligned}
$$

Remark If the fusion succeeds, we have derived a program computing m^n:

$$fastexp\ m = roll\ m \cdot binary.$$

The algorithm runs in time proportional to the length of the list generated by $binary$, which is $O(\log_2 n)$.

6. The following problem concerns calculating the sum $\sum_{i=0}^{n}(x_i \times y^i)$. Let geo be defined by:

$$\begin{aligned}&geo\ y = 1 : map\ (y\times)\ (geo\ y)\ \ ,\\ &horner\ y\ xs = sum\ (map\ mul\ (zip\ xs\ (geo\ y)))\ \ ,\end{aligned}$$

where $mul\ (a, b) = a \times b$. Let $xs = [x_0, x_1, x_2 \ldots x_n]$, $horner\ y\ xs$ computes the sum $x_0 + x_1 \times y + x_2 \times y^2 + \cdots + x_n \times y^n$. (**Remark**: for those who familiar with currying, $mul = uncurry\ (\times)$.)

(a) Show that $mul \cdot second\ (y\times) = (y\times) \cdot mul$.

Solution:

$$\begin{aligned}&\ \ mul\ (second\ (y\times)\ (x, z))\\ =&\quad \{\ \text{definition of } second\ \}\\ &\ \ mul\ (x, y \times z)\\ =&\quad \{\ \text{definition of } mul\ \}\\ &\ \ x \times (y \times z)\\ =&\quad \{\ \text{arithmetics}\ \}\\ &\ \ y \times (x \times z)\\ =&\quad \{\ \text{definition of mul}\ \}\\ &\ \ y \times mul\ (x, z)\ \ .\end{aligned}$$

(b) Let $n = length\ xs$. Asymptotically (that is, in terms of the big-O notation), how many multiplications $(\times)$ one must perform to compute $horner\ y\ xs$?

Solution: We need $O(n^2)$ multiplications.

(c) Prove that $sum \cdot map\ (y\times) = (y\times) \cdot sum$.

Solution: The aim is equivalent to prove that $sum\ (map\ (y\times)\ xs) = y \times sum\ xs$ for all xs. The case for $xs := [\,]$ is immediate. We consider the case for $x := x : xs$.

$$\begin{aligned}&\ \ sum\ (map\ (y\times)\ (x : xs))\\ =&\quad \{\ \text{definition of } map\ \}\\ &\ \ sum\ (y \times x : map\ (y\times)\ xs)\\ =&\quad \{\ \text{definition of } sum\ \}\end{aligned}$$

$$
\begin{aligned}
& y \times x + sum\ (map\ (y\times)\ xs) \\
= & \quad \{\ \text{induction}\ \} \\
& y \times x + y \times sum\ xs \\
= & \quad \{\ \text{arithmetics}\ \} \\
& y \times (x + sum\ xs) \\
= & \quad \{\ \text{definition of } sum\ \} \\
& y \times sum\ (x : xs) \quad .
\end{aligned}
$$

(d) Construct an inductive definition of $horner$ that uses only $O(n)$ multiplications to compute $horner\ y\ xs$. **Hint**: you will need a number of properties proved in the previous problems in this exercise, and perhaps some more properties.

Solution: We construct an inductive definition of $horner$ by case analysis.

Case $xs := [\,]$. It is immediate that $horner\ y\ [\,] = 0$. Details omitted.

Case $xs := x : xs$:

$$
\begin{aligned}
& horner\ y\ (x : xs) \\
= & \quad \{\ \text{definition of } horner\ \} \\
& sum\ (map\ mul\ (zip\ (x : xs)\ (geo\ y))) \\
= & \quad \{\ \text{definition of } geo\ \} \\
& sum\ (map\ mul\ (zip\ (x : xs)\ (1 : map\ (y\times)\ (geo\ y)))) \\
= & \quad \{\ \text{definition of } zip\ \} \\
& sum\ (map\ mul\ ((x, 1) : zip\ xs\ (map\ (y\times)\ (geo\ y)))) \\
= & \quad \{\ \text{definitions of } map, mul, \text{ and } sum\ \} \\
& x + sum\ (map\ mul\ (zip\ xs\ (map\ (y\times)\ (geo\ y)))) \\
= & \quad \{\ \text{since } zip\ xs\ (map\ f\ ys) = map\ (second\ f)\ (zip\ xs\ ys)\ \} \\
& x + sum\ (map\ mul\ (map\ (second\ (y\times))\ (zip\ xs\ (geo\ y)))) \\
= & \quad \{\ map \text{ fusion}\ \} \\
& x + sum\ (map\ (mul \cdot second\ (y\times))\ (zip\ xs\ (geo\ y))) \\
= & \quad \{\ \text{since } mul \cdot second\ (y\times) = (y\times) \cdot mul, map \text{ fusion}\ \} \\
& x + sum\ (map\ (y\times)\ (map\ mul\ (zip\ xs\ (geo\ y)))) \\
= & \quad \{\ \text{since } sum \cdot map\ (y\times) = (y\times) \cdot sum\ \} \\
& x + y \times sum\ (map\ mul\ (zip\ xs\ (geo\ y))) \\
= & \quad \{\ \text{definition of } horner\ \} \\
& x + y \times horner\ y\ xs \quad .
\end{aligned}
$$

Thus we conclude that

$$
\begin{aligned}
& horner\ y\ [\,] = 0 \\
& horner\ y\ (x : xs) = x + y \times horner\ y\ xs \quad .
\end{aligned}
$$