Functional Programming: Homework

Shin-Cheng Mu

Due: 9:10am, Monday, 16th July 2018

You may either write the proofs on paper and hand it in on Monday, or type the solutions in plain text ($(\wedge)$ can be written as `&&`, $(\uparrow)$ can be written as `max`, etc) and email me the solutions (`scm[AT]iis.sinica.edu.tw`). Either way, please clearly note your name, department and year, student id, etc.

1. (5 points) Recall the *steep list* problem.

$$
\begin{aligned}
&\mathit{steep} :: \mathsf{List\ Int} \rightarrow \mathsf{Bool}\\
&\mathit{steep}\ [\,] = \mathsf{True}\\
&\mathit{steep}\ (x : xs) = \mathit{steep}\ xs \wedge x > \mathit{sum}\ xs \quad ,\\
&\mathit{steepsum} :: \mathsf{List\ Int} \rightarrow (\mathsf{Bool}, \mathsf{Int})\\
&\mathit{steepsum}\ xs = (\mathit{steep}\ xs, \mathit{sum}\ xs) \quad .
\end{aligned}
$$

Derive a faster version of *steepsum* — by *foldr*-fusion! We need the fact that $\mathit{id} = \mathit{foldr}\ (:)\ [\,]$. The derivation may start from:

$$
\begin{aligned}
&\quad \mathit{steepsum}\\
= &\quad \{\ f = f \cdot \mathit{id} \text{ for all } f\ \}\\
&\quad \mathit{steepsum} \cdot \mathit{id}\\
= &\quad \{\ \mathit{id} \text{ is a } \mathit{foldr}\ \}\\
&\quad \mathit{steepsum} \cdot \mathit{foldr}\ (:)\ [\,]\\
= &\quad \{\ \text{hmmm.. } \mathit{foldr} \text{ fusion? }\ \}\\
&\quad \ldots
\end{aligned}
$$

2. (5 points) Recall the *longest positive segment* problem from Practical 3. The function *lpp* computes the length of the longest prefix that is all positive. It has an inductive definition:

$$
\begin{aligned}
&\mathit{lpp} :: \mathsf{List\ Int} \rightarrow \mathsf{Nat}\\
&\mathit{lpp}\ [\,] = 0\\
&\mathit{lpp}\ (x : xs) = \mathbf{if}\ x > 0\ \mathbf{then}\ \mathbf{1}_{+}\ (\mathit{lpp}\ xs)\ \mathbf{else}\ 0 \quad .
\end{aligned}
$$

Another, one-liner specification of *lpp* is:

$$
\mathit{lpp} = \mathit{maximum} \cdot \mathit{map\ length} \cdot \mathit{filter}\ (\mathit{all}\ (>0)) \cdot \mathit{inits} \quad .
$$

where *all* and *maximum* can be defined by:

$$\begin{aligned}
&all\ p\ [\,] = \mathsf{True} \\
&all\ p\ (x:xs) = p\ x \wedge all\ xs \ \ ,
\end{aligned}$$

$$\begin{aligned}
&maximum\ [\,] = -\infty \\
&maximum\ (x:xs) = x \uparrow maximum \ \ .
\end{aligned}$$

Derive the inductive definition from the one-liner specification (that is, show that they are the same function). You may need the *map* fusion law:

$$map\ f\ (map\ g\ xs) = map\ (f \cdot g)\ xs \ \ , \qquad (1)$$

the fact that functions around **if** can be distributed into the branches — in the world of total functions:

$$f\ (\mathbf{if}\ p\ \mathbf{then}\ x\ \mathbf{else}\ y) = \mathbf{if}\ p\ \mathbf{then}\ f\ x\ \mathbf{else}\ f\ y \ \ , \qquad (2)$$

some more specific properties of the functions we used (which can all be proved as separate lemmas):

$$filter\ (all\ p)\ (map\ (x{:})\ xs) = \mathbf{if}\ p\ x\ \mathbf{then}\ map\ (x{:})\ (filter\ (all\ p)\ xs)\ \mathbf{else}\ [\,] \ \ , \qquad (3)$$

$$length \cdot (x{:}) = (\mathbf{1}_{+}) \cdot length \ \ , \qquad (4)$$

$$maximum\ (map\ (\mathbf{1}_{+})\ xs) = \mathbf{1}_{+}\ (maximum\ xs) \ \ , \qquad (5)$$

and some basic arithmetic laws, such as that $0 \uparrow \mathbf{1}_{+}\ n = \mathbf{1}_{+}\ n$ for all n.

Hint: This may be a long, lengthy derivation. Do not be afraid! During the derivation, you may temporary focus on one of the branches of **if** if necessary, before going back to the main proof.