2008 Formosan Summer School on Logic, Language, and Computation

Program Construction and Reasoning
Exercises for Day 2

Shin-Cheng Mu

July 7th, 2008

1 In-Class Exercises

1.1 Folds and Fold-Fusion

1. Given functions $f :: \alpha \rightarrow \beta$ and $g :: \alpha \rightarrow \gamma$, $\mathit{split}\, f\; g :: \alpha \rightarrow (\beta, \gamma)$ is a function defined by:

$$\mathit{split}\, f\; g\; a = (f\; a, g\; a).$$

Recall the definition of *steep* and *sum*. The definition of *steepsum* can be re-written as:

$$\mathit{steepsum} \quad = \quad \mathit{split}\; \mathit{steep}\; \mathit{sum}.$$

Also recall that the identity function *id* on lists is a fold: $\mathit{id} = \mathit{foldr}\; (:)\; [\,]$. Use the fold-fusion theorem to fuse $\mathit{steepsum} \cdot \mathit{id}$ into one fold.

Ans:

We reason:

$$\begin{array}{cl}
 & \mathit{steepsum} \\
= & \quad \{ \text{ since } f = f \cdot \mathit{id} \text{ } \} \\
 & \mathit{steepsum} \cdot \mathit{id} \\
= & \quad \{ \text{ since } \mathit{id} = \mathit{foldr}\; (:)\; [\,] \text{ } \} \\
 & \mathit{steepsum} \cdot \mathit{foldr}\; (:)\; [\,] \\
= & \quad \{ \text{ } \mathit{foldr}\text{-fusion, see below } \} \\
 & \mathit{foldr}\; \mathit{step}\; (\mathit{true}, 0).
\end{array}$$

To perform *foldr*-fusion, we construct a function *step* such that:

$$\mathit{steepsum}\; ((:)\; x\; xs) \quad = \quad \mathit{step}\; x\; (\mathit{steepsum}\; xs).$$

We reason:

$$
\begin{aligned}
& \mathit{steepsum}\,(x{:}xs) \\
= \quad & \{\ \text{def. of } \mathit{steepsum} \text{ and } \mathit{split}\ \} \\
& (\mathit{steep}\,(x{:}xs), \mathit{sum}\,(x{:}xs)) \\
= \quad & \{\ \text{def. of } \mathit{steep} \text{ and } \mathit{sum}\ \} \\
& (\mathit{steep}\,xs\ \wedge\ x > \mathit{sum}\,xs, x + \mathit{sum}\,xs) \\
= \quad & \{\ \text{introducing local identifiers}\ \} \\
& \mathbf{let}\,(st, ss) = (\mathit{steep}\,xs, \mathit{sum}\,xs) \\
& \mathbf{in}\,(st\ \wedge\ x > ss, x + ss) \\
= \quad & \{\ \text{let } \mathit{step}\,x\,(st, ss) = (st\ \wedge\ x > ss, x + ss)\ \} \\
& \mathit{step}\,x\,(\mathit{steep}\,xs, \mathit{sum}\,xs) \\
= \quad & \{\ \text{def. of } \mathit{steepsum}\ \} \\
& \mathit{step}\,x\,(\mathit{steepsum}\,xs).
\end{aligned}
$$

We have thus derived:

$$
\begin{aligned}
& \mathit{steepsum} = \mathit{foldr}\ \mathit{step}\,(\mathit{true}, 0) \\
& \qquad \mathbf{where}\ \mathit{step}\,x\,(st, ss) = (st\ \wedge\ x > ss, x + ss).
\end{aligned}
$$

2. Recall the definition of *scanr* from the lecture:

$$\mathit{scanr}\,f\ e \quad = \quad \mathit{map}\,(\mathit{foldr}\,f\ e) \cdot \mathit{tails}$$

and its implementation as a fold:

$$
\begin{aligned}
& \mathit{scanr}\,f\ e = \mathit{foldr}\,(\mathit{sc}\,f)\,[e] \\
& \qquad\qquad \mathbf{where}\ \mathit{sc}\,f\ x\,(y{:}ys) = f\ x\ y : y : ys
\end{aligned}
$$

(a) Expand $\mathit{scanr}\,(+)\,0\,[1,2,3]$ step by step:

$$
\begin{aligned}
& \mathit{scanr}\,(+)\,0\,[1,2,3] \\
= \quad & \mathit{foldr}\,(\mathit{sc}\,(+))\,[0]\,[1,2,3] \\
= \quad & \ldots
\end{aligned}
$$

Ans:

$$
\begin{aligned}
= \quad & \mathit{sc}\,(+)\,1\,(\mathit{foldr}\,(\mathit{sc}\,(+))\,[0]\,[2,3]) \\
= \quad & \mathit{sc}\,(+)\,1\,(\mathit{sc}\,(+)\,2\,(\mathit{foldr}\,(\mathit{sc}\,(+))\,[0]\,[3])) \\
= \quad & \mathit{sc}\,(+)\,1\,(\mathit{sc}\,(+)\,2\,(\mathit{sc}\,(+)\,3\,(\mathit{foldr}\,(\mathit{sc}\,(+))\,[0]\,[\,]))) \\
= \quad & \mathit{sc}\,(+)\,1\,(\mathit{sc}\,(+)\,2\,(\mathit{sc}\,(+)\,3\,[0])) \\
= \quad & \mathit{sc}\,(+)\,1\,(\mathit{sc}\,(+)\,2\,[3,0]) \\
= \quad & \mathit{sc}\,(+)\,1\,[5,3,0] \\
= \quad & [6,5,3,0]
\end{aligned}
$$

(b) Derive the implementation of $scanr\, f\ e$ by fusing $map\,(foldr\, f\ e) \cdot tails$ into one fold.

Ans:

$$
\begin{aligned}
& map\,(foldr\, f\ e) \cdot tails \\
= & \quad \{ \text{ since } tails \text{ is a fold } \} \\
& map\,(foldr\, f\ e) \cdot foldr\ til\,[[\,]] \\
= & \quad \{ \ foldr\text{-fusion, see below } \} \\
& foldr\,(sc\, f)\,[[e]].
\end{aligned}
$$

Recall the definition of til:

$$til\, x\,(ys{:}yss) = (x : ys) : ys : yss.$$

This fusion condition is proved below:

$$
\begin{aligned}
& map\,(foldr\, f\ e)\,(til\, x\,(ys : yss)) \\
= & \quad \{ \text{ def. of } til \ \} \\
& map\,(foldr\, f\ e)\,((x : ys) : ys : yss) \\
= & \quad \{ \text{ def. of } map \ \} \\
& foldr\, f\ e\,(x : ys) : foldr\, f\ e\ ys : map\,(foldr\, f\ e)\ yss \\
= & \quad \{ \text{ def. of } foldr \ \} \\
& f\ x\,(foldr\, f\ e\ ys) : foldr\, f\ e\ ys : map\,(foldr\, f\ e)\ yss \\
= & \quad \{ \text{ introducing local identifiers } \} \\
& \mathbf{let}\,(ys, yss) = (foldr\, f\ e\ ys, map\,(foldr\, f\ e)\ yss) \\
& \mathbf{in}\, f\ x\ ys{:}\ ys{:}\ yss \\
= & \quad \{ \text{ let } sc\, f\ x\,(ys{:}yss) = f\ x\ ys : ys : yss \ \} \\
& sc\, f\ x\,(fold\, f\ e\ ys : map\,(foldr\, f\ e)\ yss) \\
= & \quad \{ \text{ def. of } map \ \} \\
& sc\, f\ x\,(map\,(foldr\, f\ e)\,(ys : yss)).
\end{aligned}
$$

We have therefore derived:

$$scanr\, f\ e = foldr\,(sc\, f)\,[[e]],$$

where $sc\, f\ x\,(ys{:}yss) = f\ x\ ys : ys : yss$.

2 Take-Home Exercise (Due Date: July 10th)

You do not have to do the exercises below if you have completed any of the exercises from Day 1. Exercise 1 is worth 40 points while exercise 2 is worth 50 points.

1. The function $filter\ p$ selects from a list all elements satisfying a predicate p. For example, $filter\ even\,[1, 2, 3, 4] = [2, 4]$.

(a) Give a recursive definition of *filter*:

$$\begin{array}{lcl} \mathit{filter}\ p\ [\,] & = & \ldots \\ \mathit{filter}\ p\ (x{:}xs) & = & \ldots \end{array}$$

(b) Define *filter* p in terms of *foldr*.

(c) Prove, by fold-fusion, that

$$\mathit{filter}\ p \cdot \mathit{map}\ f \quad = \quad \mathit{map}\ f \cdot \mathit{filter}\ (p \cdot f).$$

Hint: apply fold-fusion on both sides, and show that they are equal to the same fold.

2. Given two functions h_1 and h_2, the function $\mathit{split}\ h_1\ h_2$ computes the pair of their results:

$$\mathit{split}\ h_1\ h_2\ xs \quad = \quad (h_1\ xs, h_2\ xs).$$

In the special case when both h_1 and h_2 are defined by *foldr*:

$$\begin{array}{lcl} h_1 & = & \mathit{foldr}\ f_1\ e_1, \\ h_2 & = & \mathit{foldr}\ f_2\ e_2, \end{array}$$

the following "banana-split" rule allows us to express $\mathit{split}\ h_1\ h_2$ using one single *foldr*:

$$\begin{array}{l} \mathit{split}\ h_1\ h_2 = \mathit{foldr}\ g\ (e_1, e_2), \\ \qquad\qquad \mathbf{where}\ g\ x\ (y, z) = (f_1\ x\ y, f_2\ x\ z). \end{array}$$

It optimises two traversal through the list to only one traversal. It is called "banana-split" because folds used to be written using a notation called "banana brackets".

(a) The function *split sum length* return the pair of sum and length of the input list. Use the banana-split rule to express *split sum length* by a fold.

(b) Prove the banana-split rule by fold fusion. Hint: recall that $\mathit{split}\ h_1\ h_2 = \mathit{split}\ h_1\ h_2 \cdot \mathit{id}$, and *id* is a fold.