# Programming Languages: Functional Programming
# Practicals 6. Folds, and Fold-Fusion

Shin-Cheng Mu

Spring 2022

1. Express the following functions by $foldr$:

    1. $all\ p$ :: List $a \rightarrow$ Bool, where $p :: a \rightarrow$ Bool.
    2. $elem\ z$ :: List $a \rightarrow$ Bool, where $z :: a$.
    3. $concat$ :: List (List $a$) $\rightarrow$ List $a$.
    4. $\mathit{filter}\ p$ :: List $a \rightarrow$ List $a$, where $p :: a \rightarrow$ Bool.
    5. $takeWhile\ p$ :: List $a \rightarrow$ List $a$, where $p :: a \rightarrow$ Bool.
    6. $id$ :: List $a \rightarrow$ List $a$.

    In case you haven't seen them, $all\ p\ xs$ is True iff. all elements in $xs$ satisfy $p$, and $elem\ z\ xs$ is True iff. $x$ is a member of $xs$.

    **Solution:**

    1. $all\ p = foldr\ (\lambda x\ b \rightarrow p\ x \wedge b)\ \mathsf{True}$ .
    2. $elem\ x = foldr\ (\lambda y\ b \rightarrow x == y \vee b)\ \mathsf{False}$ ,
    3. $concat = foldr\ (+\!\!+)\ [\,]$ .
    4. $\mathit{filter}\ p = foldr\ (\lambda x\ xs \rightarrow \mathbf{if}\ p\ x\ \mathbf{then}\ x : xs\ \mathbf{else}\ xs)\ [\,]$ ,
    5. $takeWhile\ p = foldr\ (\lambda x\ xs \rightarrow \mathbf{if}\ p\ x\ \mathbf{then}\ x : xs\ \mathbf{else}\ [\,])\ [\,]$ ,
    6. $id = foldr\ (:)\ [\,]$ .

2. Given $p :: a \rightarrow$ Bool, can $dropWhile\ p$ :: List $a \rightarrow$ List $a$ be written as a $foldr$?

**Solution:** No. Consider $dropWhile\ even\ [5,4,2,1]$, which ought to be $[5,4,1,1]$. Meanwhile, $dropWhile\ even\ [4,2,1] = [1]$, and the lost elements cannot be recovered.

3. Express the following functions by $foldr$:

    1. $inits :: \mathsf{List}\ a \to \mathsf{List}\ (\mathsf{List}\ a)$.
    2. $tails :: \mathsf{List}\ a \to \mathsf{List}\ (\mathsf{List}\ a)$.
    3. $perms :: \mathsf{List}\ a \to \mathsf{List}\ (\mathsf{List}\ a)$.
    4. $sublists :: \mathsf{List}\ a \to \mathsf{List}\ (\mathsf{List}\ a)$.
    5. $splits :: \mathsf{List}\ a \to \mathsf{List}\ (\mathsf{List}\ a, \mathsf{List}\ a)$.

**Solution:**

1. $inits = foldr\ (\lambda x\ xss \to [\,] : map\ (x{:})\ xss)\ [[\,]]$ .
2. $tails = foldr\ (\lambda x\ xss \to (x : head\ xss) : xss)\ [[\,]]$ ,
3. $perms = foldr\ (\lambda x\ xss \to concat\ (map\ (fan\ x)\ xss))\ [[\,]]$
4. $sublists = foldr\ (\lambda x\ xss \to xss \mathbin{+\!\!+} map\ (x{:})\ xss)\ [[\,]]$
5. $splits$ can be defined by:

$$
\begin{aligned}
&splits = foldr\ spl\ [([\,],[\,])]\ , \\
&\quad \mathbf{where}\ spl\ x\ ((xs, ys) : zss) = \\
&\qquad\qquad ([\,], x : xs \mathbin{+\!\!+} ys) : map\ ((x{:}) \times id)\ ((xs, ys) : zss)\ .
\end{aligned}
$$

where $(f \times g)\ (x, y) = (f\ x, g\ y)$.

4. Prove the $foldr$-fusion theorem. To recite the theorem: given $f :: a \to b \to b$, $e :: b$, $h :: b \to c$ and $g :: a \to c \to c$, we have

$$h \cdot foldr\ f\ e = foldr\ g\ (h\ e)\ ,$$

if $h\ (f\ x\ y) = g\ x\ (h\ y)$ for all $x$ and $y$.

**Solution:** The aim is to prove that $h\ (foldr\ f\ e\ xs) = foldr\ g\ (h\ e)\ xs$ for all $xs$, assuming that $h\ (f\ x\ y) = g\ x\ (h\ y)$.

**Case** $xs := [\,]$:

$$
\begin{aligned}
& h\ (\mathit{foldr}\ f\ e\ [\,]) \\
={} & h\ e \\
={} & \mathit{foldr}\ g\ (h\ e)\ [\,] \quad .
\end{aligned}
$$

**Case** $xs := x : xs$:

$$
\begin{aligned}
& h\ (\mathit{foldr}\ f\ e\ (x : xs)) \\
={} & \quad \{ \text{ definition of } \mathit{foldr} \ \} \\
& h\ (f\ x\ (\mathit{foldr}\ f\ e\ xs)) \\
={} & \quad \{ \text{ fusion condition: } h\ (f\ x\ y) = g\ x\ (h\ y) \ \} \\
& g\ x\ (h\ (\mathit{foldr}\ f\ e\ xs)) \\
={} & \quad \{ \text{ induction } \} \\
& g\ x\ (\mathit{foldr}\ g\ (h\ e)\ xs) \\
={} & \quad \{ \text{ definition of } \mathit{foldr} \ \} \\
& \mathit{foldr}\ g\ (h\ e)\ (x : xs) \quad .
\end{aligned}
$$

5. Prove the $\mathit{map}$-fusion rule $\mathit{map}\ f \cdot \mathit{map}\ g = \mathit{map}\ (f \cdot g)$ by $\mathit{foldr}$-fusion.

**Solution:** Since $\mathit{map}\ g$ is a $\mathit{foldr}$, we proceed as follows:

$$
\begin{aligned}
& \mathit{map}\ f \cdot \mathit{map}\ g \\
={} & \quad \{ \ \mathit{map}\ g \text{ is a } \mathit{foldr} \ \} \\
& \mathit{map}\ f \cdot \mathit{foldr}\ (\lambda x\ ys \rightarrow g\ x : ys)\ [\,] \\
={} & \quad \{ \ \mathit{foldr}\text{-fusion } \} \\
& \mathit{foldr}\ (\lambda x\ ys \rightarrow f\ (g\ x) : ys)\ [\,] \\
={} & \quad \{ \text{ definition of } \mathit{map} \text{ as a } \mathit{foldr} \ \} \\
& \mathit{map}\ (f \cdot g) \quad .
\end{aligned}
$$

The fusion condition is proved below:

$$
\begin{aligned}
& \mathit{map}\ f\ (g\ x : ys) \\
={} & \quad \{ \text{ definition } \mathit{map} \ \} \\
& f\ (g\ x) : \mathit{map}\ f\ ys \quad .
\end{aligned}
$$

6. Prove that $\mathit{sum} \cdot \mathit{concat} = \mathit{sum} \cdot \mathit{map}\ \mathit{sum}$ by $\mathit{foldr}$-fusion, twice. Compare the proof with you previous proof in earlier parts of this course.

**Solution:**

$$
\begin{aligned}
& \mathit{sum} \cdot \mathit{concat} \\
= {} & \mathit{sum} \cdot \mathit{foldr}\ (+\!\!+)\ [\,] \\
= {} & \quad \{\ \mathit{foldr}\text{-fusion}\ \} \\
& \mathit{foldr}\ (\lambda xs\ n \rightarrow \mathit{sum}\ xs + n)\ 0 \\
= {} & \quad \{\ \mathit{foldr}\text{-}\mathit{map}\ \text{fusion, see Exercise 7}\ \} \\
& \mathit{foldr}\ (+)\ 0 \cdot \mathit{map}\ \mathit{sum} \\
= {} & \mathit{sum} \cdot \mathit{map}\ \mathit{sum} \quad .
\end{aligned}
$$

Fusion conditions for the $\mathit{foldr}$-fusion is

$$\mathit{sum}\ (xs +\!\!+ ys) = \mathit{sum}\ xs + \mathit{sum}\ ys \quad ,$$

which is the key property we needed in the early part of this term to prove the same property. We have proved the property before, by induction on $xs$. We omit the proof here. (Note that we can also prove it by two more $\mathit{foldr}$-fusion, noting that $(+\!\!+\ ys)$ is a $\mathit{foldr}$, and so is $\mathit{sum}$.)

See Exercise 7 for $\mathit{foldr}\text{-}\mathit{map}$ fusion. The penultimate equality holds because $(+) \cdot \mathit{sum} = (\lambda xs\ n \rightarrow \mathit{sum}\ xs + n)$. Instead of $\mathit{foldr}\text{-}\mathit{map}$ fusion we cal also use $\mathit{foldr}$ fusion alone. The fusion condition is $\mathit{sum}\ (\mathit{sum}\ xs : xss) = \mathit{sum}\ xs + \mathit{sum}\ xss$.

The $\mathit{foldr}$-fusion theorem captures the common pattern in these proofs. We only need to fill in the problem-dependent proofs.

7. The $\mathit{map}$ fusion theorem is an instance of the $\mathit{foldr}\text{-}\mathit{map}$ fusion theorem: $\mathit{foldr}\ f\ e \cdot \mathit{map}\ g = \mathit{foldr}\ (f \cdot g)\ e$.

   (a) Prove the theorem.

**Solution:** Since $\mathit{map}\ g$ is a $\mathit{foldr}$, we proceed as follows:

$$
\begin{aligned}
& \mathit{foldr}\ f\ e \cdot \mathit{map}\ g \\
= {} & \quad \{\ \mathit{map}\ g\ \text{is a}\ \mathit{foldr}\ \} \\
& \mathit{foldr}\ f\ e \cdot \mathit{foldr}\ (\lambda x\ ys \rightarrow g\ x : ys)\ [\,] \\
= {} & \quad \{\ \mathit{foldr}\text{-fusion}\ \} \\
& \mathit{foldr}\ (f \cdot g)\ (\mathit{foldr}\ f\ e\ [\,]) \\
= {} & \quad \{\ \text{definition of}\ \mathit{foldr}\ \} \\
& \mathit{foldr}\ (f \cdot g)\ e \quad .
\end{aligned}
$$

The fusion condition is proved below:

$$
\begin{aligned}
& \mathit{foldr}\ f\ e\ (g\ x : ys) \\
= {} & \quad \{\ \text{definition}\ \mathit{foldr}\ \} \\
& f\ (g\ x)\ (\mathit{foldr}\ f\ e\ ys) \quad .
\end{aligned}
$$

(b) Express $sum \cdot map\ (2\times)$ as a $foldr$.

**Solution:**

$$\begin{aligned}
& \quad sum \cdot map\ (2\times) \\
&= foldr\ (+)\ 0 \cdot map\ (2\times) \\
&= \quad \{\ foldr\text{-}map \text{ fusion } \} \\
& \quad foldr\ ((+) \cdot (2\times))\ 0 \quad .
\end{aligned}$$

(c) Show that $(2\times) \cdot sum$ reduces to the same $foldr$ as the one above.

**Solution:**

$$\begin{aligned}
& \quad (2\times) \cdot sum \\
&= (2\times) \cdot foldr\ (+)\ 0 \\
&= \quad \{\ foldr \text{ fusion } \} \\
& \quad foldr\ ((+) \cdot (2\times))\ 0 \quad .
\end{aligned}$$

The fusion condition is

$$\begin{aligned}
& \quad 2 \times (x + y) \\
&= \quad \{ \text{ distributivity } \} \\
& \quad 2 \times x + 2 \times y \\
&= \quad \{ \text{ definition of } (\cdot) \} \\
& \quad ((+) \cdot (2\times))\ x\ (2 \times y) \quad .
\end{aligned}$$

8. Prove that $map\ f\ (xs \mathbin{+\!\!+} ys) = map\ f\ xs \mathbin{+\!\!+} map\ f\ ys$ by $foldr$-fusion. **Hint**: this is equivalent to $map\ f \cdot (\mathbin{+\!\!+}\ ys) = (\mathbin{+\!\!+}\ map\ f\ ys) \cdot map\ f$. You may need to do (any kinds of) fusion twice.

**Solution:** Recall that $(\mathbin{+\!\!+}\ ys)$ is a $foldr$. Use $foldr$ fusion and $foldr$-$map$ fusion:

$$\begin{aligned}
& \quad (\mathbin{+\!\!+}\ map\ f\ ys) \cdot map\ f \\
&= \quad \{\ foldr\text{-}map \text{ fusion } \} \\
& \quad foldr\ ((:) \cdot f)\ (map\ f\ ys) \\
&= \quad \{\ foldr \text{ fusion } \} \\
& \quad map\ f \cdot (\mathbin{+\!\!+}\ ys) \quad .
\end{aligned}$$

The fusion condition of the last step is:

$$\begin{aligned}
& \quad map\ f\ (x : zs) \\
&= \quad \{ \text{ definition of } map\ \}
\end{aligned}$$

$$
\begin{aligned}
&\quad f\ x : map\ f\ zs \\
= &\quad \{\ \text{definition of } (\cdot)\ \} \\
&\quad ((:) \cdot f)\ x\ (map\ f\ zs)\ \ .
\end{aligned}
$$

9. Prove that $length \cdot concat = sum \cdot map\ length$ by fusion.

**Solution:** We caculate

$$
\begin{aligned}
&\quad length \cdot concat \\
= &\ length \cdot foldr\ (+\!\!+)\ [\,] \\
= &\quad \{\ foldr\text{-fusion}\ \} \\
&\quad foldr\ ((+) \cdot length)\ 0 \\
= &\ \{ \mid sum = foldr\ (+)\ 0 \mid, \mid foldr \mid - \mid map \mid fusion \} \\
&\quad sum \cdot map\ length\ \ .
\end{aligned}
$$

The fusion condition is proved below:

$$
\begin{aligned}
&\quad length\ (xs +\!\!+ ys) \\
= &\quad \{\ (+\!\!+) \text{ and } (+) \text{ homorphic}\ \} \\
&\quad length\ xs + length\ ys \\
= &\quad \{\ \text{definition of } (\cdot)\ \} \\
&\quad ((+) \cdot length)\ xs\ (length\ ys)\ \ .
\end{aligned}
$$

10. Let $scanr\ f\ \ e = map\ (foldr\ f\ \ e) \cdot tails$. Construct, by $foldr$-fusion, an implementation of $scanr$ whose number of calls to $f$ is proportional to the length of the input list.

**Solution:** Recall that $tails$ is a $foldr$:

$$
tails = foldr\ (\lambda x\ xss \to (x : head\ xss) : xss)\ [[\,]]\ \ ,
$$

We try to fuse $map\ (foldr\ f\ \ e)$ into $tails$. For the base value, notice that

$$
map\ (foldr\ f\ \ e)\ [[\,]] = [\,e\,]\ \ .
$$

To construct the step function, we work on the fusion condition:

$$
\begin{aligned}
&\quad map\ (foldr\ f\ \ e)\ ((x : head\ xss) : xss) \\
= &\quad \{\ \text{definition of } map\ \} \\
&\quad foldr\ f\ \ e\ (x : head\ xss) : map\ (foldr\ f\ \ e)\ xss
\end{aligned}
$$

$$
\begin{aligned}
&= \quad \{ \text{definition of } foldr \} \\
&\quad f\ x\ (foldr\ f\ e\ (head\ xss)) : map\ (foldr\ f\ e)\ xss \\
&= \quad \{ foldr\ f\ e\ (head\ xss) = head\ (map\ (foldr\ f\ e)\ xss) \} \\
&\quad \mathbf{let}\ ys = map\ (foldr\ f\ e)\ xss \\
&\quad \mathbf{in}\ f\ x\ (head\ ys) : ys \quad .
\end{aligned}
$$

We have therefore constructed:

$$scanr\ f\ e = foldr\ (\lambda x\ ys \to f\ x\ (head\ ys) : ys)\ [e] \quad .$$

You may find the inductive definition easier to comprehend:

$$
\begin{aligned}
&scanr\ f\ e\ [\,] &&= [e] \\
&scanr\ f\ e\ (x : xs) &&= f\ x\ (head\ ys) : ys \quad , \\
&\quad \mathbf{where}\ ys = scanr\ f\ e\ xs \quad .
\end{aligned}
$$

11. Recall the function $binary :: \mathsf{Nat} \to [\mathsf{Nat}]$ that returns the *reversed* binary representation of a natural number, for example $binary\ 4 = [0, 0, 1]$. Also, we talked about a function $decimal :: [\mathsf{Nat}] \to \mathsf{Nat}$ that converts the representation back to a natural number.

(a) This time, express $decimal$ using a $foldr$.

**Solution:**

$$decimal = foldr\ (\lambda d\ n \to d + 2 \times n)\ 0 \quad .$$

(b) Recall the function $exp\ m\ n = m^n$. Use $foldr$-fusion to construct $step$ and $base$ such that

$$exp\ m \cdot decimal = foldr\ step\ base \quad .$$

If the fusion succeeds, we have derived a hylomorphism computing $m^n$:

$$fastexp\ m = foldr\ step\ base \cdot binary \quad .$$

**Solution:** For the base value, we have $base = exp\ m\ 0 = 1$.
For the step function, we calculate

$$
\begin{aligned}
&\quad exp\ m\ (d + 2 \times n) \\
&= \quad \{ \text{since } m^{x+y} = m^x \times m^y \}
\end{aligned}
$$

$$
\begin{aligned}
&\quad \mathit{exp}\ m\ d \times \mathit{exp}\ m\ (2 \times n) \\
= &\quad \{ \text{ since } m^{2n} = (m^n)^2\text{, let } \mathit{square}\ x = x \times x \ \} \\
&\quad \mathit{exp}\ m\ d \times \mathit{square}\ (\mathit{exp}\ m\ n) \\
= &\quad \{ \ d \text{ is either } 0 \text{ or } 1\text{. Expand the definition } \} \\
&\quad \mathbf{if}\ d == 0\ \mathbf{then}\ \mathit{square}\ (\mathit{exp}\ m\ n)\ \mathbf{else}\ m \times \mathit{square}\ (\mathit{exp}\ m\ n) \quad .
\end{aligned}
$$

Therefore we conclude

$$
\begin{aligned}
\mathit{exp}\ m \cdot \mathit{decimal} = \mathit{foldr}\ (\lambda d\ x \to \mathbf{if}\ d == 0\ &\mathbf{then}\ \mathit{square}\ x \\
&\mathbf{else}\ m \times \mathit{square}\ x)\ 1 \quad .
\end{aligned}
$$

12. Express $\mathit{reverse} :: \mathsf{List}\ a \to \mathsf{List}\ a$ by a $\mathit{foldr}$. Let $\mathit{revcat} = (+\!\!+) \cdot \mathit{reverse}$. Express $\mathit{revcat}$ as a $\mathit{foldr}$.

**Solution:** $\mathit{reverse} = \mathit{foldr}\ (\lambda x\ xs \to xs +\!\!+ [x])\ []$.

To fuse $(+\!\!+)$ into $\mathit{reverse}$, the base value is $(+\!\!+)\ [] = \mathit{id}$. To construct the step function, we try to meet the fusion condition:

$$
(+\!\!+)\ ((\lambda x\ xs \to xs +\!\!+ [x])\ x\ xs) = \mathit{step}\ x\ ((+\!\!+)\ xs) \quad .
$$

If we calculate:

$$
\begin{aligned}
&(+\!\!+)\ ((\lambda x\ xs \to xs +\!\!+ [x])\ x\ xs) \\
&\ = (+\!\!+)\ (xs +\!\!+ [x]) \quad ,
\end{aligned}
$$

it is hard to figure out how to proceed, since $(+\!\!+)$ expects another argument. It is easier to calculate if we supply it another argument $ys$. We restart and calculate:

$$
\begin{aligned}
&(+\!\!+)\ ((\lambda x\ xs \to xs +\!\!+ [x])\ x\ xs)\ ys \\
&\ = (+\!\!+)\ (xs +\!\!+ [x])\ ys \\
&\ = (xs +\!\!+ [x]) +\!\!+ ys \\
&\ = \quad \{ \ (+\!\!+) \text{ associative } \} \\
&\quad\ xs +\!\!+ ([x] +\!\!+ ys) \\
&\ = \quad \{ \text{ definition of } (\cdot) \ \} \\
&\quad\ (((+\!\!+)\ xs) \cdot (x{:}))\ ys \\
&\ = \quad \{ \text{ factor out } x\text{, } ((+\!\!+)\ xs)\text{, and } ys \ \} \\
&\quad\ (\lambda x\ f \to f \cdot (x{:}))\ x\ ((+\!\!+)\ xs)\ ys \quad .
\end{aligned}
$$

We conclude that

$$
\mathit{revcat} = \mathit{foldr}\ (\lambda x\ f \to f \cdot (x{:}))\ \mathit{id} \quad .
$$

13. Fold on natural numbers.

    (a) The predicate $even :: \mathsf{Nat} \to \mathsf{Bool}$ yields True iff. the input is an even number. Define $even$ in terms of $foldN$.

    **Solution:**

    $$even = foldN\ not\ \mathsf{True}\ \ .$$

    (b) Express the identity function on natural numbers $id\ n = n$ in terms of $foldN$.

    **Solution:**

    $$id = foldN\ \mathbf{1}_+\ 0\ \ .$$

14. Fuse $even$ into $(+n)$. This way we get a function that checks whether a natural number is even after adding $n$.

    **Solution:** Recall that $(+n) = foldN\ \mathbf{1}_+\ n$. To fuse $even \cdot (+n)$ into one $foldN$, the base value is $even\ n$. To find out the step function, recall that $even\ (\mathbf{1}_+\ n) = not\ (even\ n)$. We may then conclude:

    $$even \cdot (+n) = foldN\ not\ (even\ n)\ \ .$$

15. The famous Fibonacci number is defined by:

    $$\begin{aligned} fib\ 0 &= 0 \\ fib\ 1 &= 1 \\ fib\ (2+n) &= fib\ (1+n) + fib\ n\ \ . \end{aligned}$$

    The definition above, when taken directly as an algorithm, is rather slow. Define $fib2\ n = (fib\ (1+n), fib\ n)$. Derive an $O(n)$ implementation of $fib2$ by fusing it with $id :: \mathsf{Nat} \to \mathsf{Nat}$.

    **Solution:** Recall that $id = foldN\ (\mathbf{1}_+)\ 0$. Fusing $fib2$ into $id$, the base value is $fib2\ 0 = (1, 0)$. To construct the step function we calculate

    $$\begin{aligned} & fib2\ (\mathbf{1}_+\ n) \\ = {} & (fib\ (\mathbf{1}_+\ (\mathbf{1}_+\ n)), fib\ (\mathbf{1}_+\ n)) \end{aligned}$$

$$
\begin{aligned}
&= \quad \{ \text{definition of } \mathit{fib} \} \\
&\quad (\mathit{fib}\ (\mathbf{1}_+\ n) + \mathit{fib}\ n, \mathit{fib}\ (\mathbf{1}_+\ n)) \\
&= (\lambda(x,y) \to (x+y, x))\ (\mathit{fib2}\ n) \ .
\end{aligned}
$$

Therefore we conclude that

$$\mathit{fib2} = \mathit{foldN}\ (\lambda(x,y) \to (x+y,x))\ (1,0) \ .$$

16. What are the fold fusion theorems for ETree and ITree?

**Solution:**

$$
\begin{aligned}
h \cdot \mathit{foldIT}\ f\ e = \mathit{foldIT}\ g\ (h\ e) \ &\Leftarrow\ h\ (f\ x\ y\ z) = g\ x\ (h\ y)\ (h\ z) \ , \\
h \cdot \mathit{foldET}\ f\ k = \mathit{foldET}\ g\ (h \cdot k) \ &\Leftarrow\ h\ (f\ x\ y) = g\ (h\ x)\ (h\ y) \ .
\end{aligned}
$$