Functional Programming
Practicals 1

Shin-Cheng Mu

FLOLAC 2022

Folds and Fold-Fusion

1. Express the following functions by *foldr*:

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

    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*.

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

3. Express the following functions by *foldr*:

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

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

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

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

7. The *map* fusion theorem is an instance of the *foldr*-*map* fusion theorem: $foldr\ f\ e \cdot map\ g = foldr\ (f \cdot g)\ e$.
   (a) Prove the theorem.
   (b) Express $sum \cdot map\ (2\times)$ as a *foldr*.
   (c) Show that $(2\times) \cdot sum$ reduces to the same *foldr* as the one above.

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

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

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.

11. Recall the function $binary :: \mathrm{Nat} \rightarrow [\mathrm{Nat}]$ that returns the *reversed* binary representation of a natural number, for example $binary\ 4 = [0, 0, 1]$. Also, we talked about a function $decimal :: [\mathrm{Nat}] \rightarrow \mathrm{Nat}$ that converts the representation back to a natural number.
    (a) This time, express *decimal* using a *foldr*.
    (b) Recall the function $exp\ m\ n = m^n$. Use *foldr*-fusion to construct *step* and *base* such that

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

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

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

12. Express $reverse :: \mathrm{List}\ a \rightarrow \mathrm{List}\ a$ by a *foldr*. Let $revcat = (+\!\!+) \cdot reverse$. Express *revcat* as a *foldr*.

13. Fold on natural numbers.
    (a) The predicate $even :: \mathrm{Nat} \rightarrow \mathrm{Bool}$ yields True iff. the input is an even number. Define *even* in terms of *foldN*.
    (b) Express the identity function on natural numbers $id\ n = n$ in terms of *foldN*.

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

15. The famous Fibonacci number is defined by:

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

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

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