I217E: Functional Programming

Nao Hirokawa

JAIST

Term 2-1, 2022

`http://www.jaist.ac.jp/~hirokawa/lectures/fp/`

Schedule

10/12	introduction	11/9	interpreters
10/14	algebraic data types I	11/11	compilers
10/19	algebraic data types II	11/16	termination
10/21	program reasoning	11/18	confluence
10/26	applications	11/25	verification
10/28	data structures I	11/30	review
11/2	data structures II	12/5	exam
11/4	computational models		

Evaluation

exam (60) + reports (40)

Lambda Expressions and Infix Operators

Lambda Expressions

$$(\backslash x \text{ -> } x+1)\ 2 \;=\; 2+1 \;=\; 3$$
$$(\backslash x\ y \text{ -> } x*y)\ 2\ 3 \;=\; 2*3 \;=\; 6$$

Infix Operators and Partial Applications

$$(+)\ 10\ 2 \;=\; 10+2 \;=\; 12$$
$$(/)\ 10\ 2 \;=\; 10/2 \;=\; 5.0$$
$$(/\ 2)\ 10 \;=\; 10/2 \;=\; 5.0$$
$$(10\ /)\ 2 \;=\; 10/2 \;=\; 5.0$$

Higher-Order Functions

Definition

higher-order functions take/return functions

$\texttt{twice}\ f\ x = f\ (f\ x)$

$$\texttt{twice}\ (*2)\ 1 = (*\ 2)\ ((*\ 2)\ 1)$$

Map

`map` $f\ [x_1, \ldots, x_n] = [f\ x_1, \ldots, f\ x_n]$

$$
\begin{aligned}
\texttt{map}\ (*\ 10)\ [] &= [] \\
\texttt{map}\ (*\ 10)\ [-4] &= [-40] \\
\texttt{map}\ (*\ 10)\ [3, -4] &= [30, -40] \\
\texttt{map}\ (*\ 10)\ [-2, 3, -4] &= [-20, 30, -40] \\
\texttt{map}\ (*\ 10)\ [1, -2, 3, -4] &= [10, -20, 30, -40]
\end{aligned}
$$

```
myMap :: (a -> b) -> [a] -> [b]
myMap f []        = ...
myMap f (x : xs) = ...
```


Filter

`filter` $p\ [x_1, \ldots, x_n]$ only keeps x_is with $p\ x_i$ == True

$$
\begin{aligned}
\texttt{filter}\ (> 0)\ [] &= [] \\
\texttt{filter}\ (> 0)\ [-4] &= [] \\
\texttt{filter}\ (> 0)\ [3, -4] &= [3] \\
\texttt{filter}\ (> 0)\ [-2, 3, -4] &= [3] \\
\texttt{filter}\ (> 0)\ [1, -2, 3, -4] &= [1, 3]
\end{aligned}
$$

```
myFilter :: (a -> Bool) -> [a] -> [a]
myFilter p []              = ...
myFilter p (x : xs) | ... = ...
                    | ... = ...
```


Partition

$$
\begin{aligned}
\texttt{partition}\ (> 0)\ [] &= ([], []) \\
\texttt{partition}\ (> 0)\ [-4] &= ([], [-4]) \\
\texttt{partition}\ (> 0)\ [3, -4] &= ([3], [-4]) \\
\texttt{partition}\ (> 0)\ [-2, 3, -4] &= ([3], [-2, -4]) \\
\texttt{partition}\ (> 0)\ [1, -2, 3, -4] &= ([1, 3], [-2, -4])
\end{aligned}
$$

```
partition p []        = ...
partition p (x : xs)
  | ...               = ...
  | ...               = ...
  where (ys, zs) = partition p ...
```


Fold Functions

$$
\begin{aligned}
\texttt{foldl}\ (\oplus)\ e\ [x_1, \ldots, x_n] &= ((e \oplus x_1) \oplus x_2) \oplus \cdots \oplus x_n \\
\texttt{foldr}\ (\oplus)\ e\ [x_1, \ldots, x_n] &= x_1 \oplus \cdots \oplus (x_{n-1} \oplus (x_n \oplus e))
\end{aligned}
$$

$$
\begin{aligned}
\texttt{foldl}\ (-)\ 10\ [1, 2, 3] &= ((10 - 1) - 2) - 3 = 4 \\
\texttt{foldr}\ (-)\ 10\ [1, 2, 3] &= 1 - (2 - (3 - 10)) = -8
\end{aligned}
$$

```
myFoldl :: (a -> b -> a) -> a -> [b] -> a
myFoldl f e []        = ...
myFoldl f e (x : xs) = ...
```

List Comprehension

Set Comprehension (in Mathematics)

$$\{x \times 10 \mid x \in \{1, -2, 3, -4\}\} = \{10, -20, 30, -40\}$$
$$\{x + 1 \mid x \in \{1, -2, 3, -4\} \text{ and } x > 0\} = \{2, 4\}$$
$$\{x + y \mid x \in \{10, 20\} \text{ and } y \in \{1, 2\}\} = \{11, 12, 21, 22\}$$

List Comprehension (in Haskell)

$$[\, x * 10 \mid x \text{ <- } [1, -2, 3, -4]\,] = [10, -20, 30, -40]$$
$$[\, x \mid x \text{ <- } [1, -2, 3, -4], x > 0\,] = [1, 3]$$
$$[\, x + y \mid x \text{ <- } [10, 20], y \text{ <- } [1, 2]\,] = [11, 12, 21, 22]$$

Quick Sort (Hoare 1960)

$$\begin{aligned}
\texttt{qsort}\ [\,] &= [\,] \\
\texttt{qsort}\ (x : xs) &= \texttt{qsort}\ [\, y \mid y \text{ <- } xs, y < x\,] \mathbin{+\!+} [x] \mathbin{+\!+} \\
&\quad\ \texttt{qsort}\ [\, y \mid y \text{ <- } xs, y >= x\,]
\end{aligned}$$

Note

- beautiful and practically efficient algorithm
- $O(n^2)$ in worst case (why?)

Divide and Conquer

$$\begin{gathered}
\text{qsort } [3, 4, 2, 5, 1] \\
= \\
\text{qsort } [2, 1] \mathbin{+\!+} [3] \mathbin{+\!+} \text{qsort } [4, 5] \\
= \\
(\text{qsort } [1] \mathbin{+\!+} [2] \mathbin{+\!+} \text{qsort } [\,]) \mathbin{+\!+} [3] \mathbin{+\!+} (\text{qsort } [\,] \mathbin{+\!+} [4] \mathbin{+\!+} \text{qsort } [5]) \\
= \\
([1] \mathbin{+\!+} [2] \mathbin{+\!+} [\,]) \mathbin{+\!+} [3] \mathbin{+\!+} ([\,] \mathbin{+\!+} [4] \mathbin{+\!+} [5]) \\
= \\
[1, 2] \mathbin{+\!+} [3] \mathbin{+\!+} [4, 5] \\
= \\
[1, 2, 3, 4, 5]
\end{gathered}$$

Merge Sort (van Neumann 1945)

$$\texttt{msort}\ [x_1, \ldots, x_n] = \begin{cases} [\,] & \text{if } n = 0 \\ [x_1] & \text{if } n = 1 \\ \texttt{merge}\ (\texttt{msort}\ ys)\ (\texttt{msort}\ zs) & \text{otherwise} \end{cases}$$

where

- $(ys, zs) = ([x_1, x_3, x_5, \ldots], [x_2, x_4, x_6, \ldots])$
- `merge` xs ys merges two **sorted** lists

Note

- problem is divided into subproblems of same size
- $O(n \log n)$

Divide and Conquer

let $xs \otimes ys =$ `merge` $xs\ ys$

$$\begin{array}{c} \text{msort } [3,4,2,5,1] \\ = \\ \text{msort } [3,2,1] \otimes \text{msort } [4,5] \\ = \\ (\text{msort } [3,1] \otimes \text{msort } [2]) \otimes (\text{msort } [4] \otimes \text{msort } [5]) \\ = \\ (([3] \otimes [1]) \otimes [2]) \otimes ([4] \otimes [5]) \\ = \\ ([1,3] \otimes [2]) \otimes [4,5] \\ = \\ [1,2,3] \otimes [4,5] = [1,2,3,4,5] \end{array}$$

Homework

1. Use `foldr` to implement `sumList`:

$$\texttt{sumList}\ [x_1, \ldots, x_n] = x_1 + \cdots + x_n$$

2. Use list comprehension to re-implement `oddplus1`:

```
oddplus1 xs =
  map (+ 1) (filter (\x -> mod x 2 == 1) xs)
```

3. Implement `merge` :: $[\mathsf{Int}] \to [\mathsf{Int}] \to [\mathsf{Int}]$.
4. Implement `split` :: $[\mathsf{Int}] \to ([\mathsf{Int}], [\mathsf{Int}])$:

$$\texttt{split}\ [x_1, \ldots, x_n] = ([x_1, x_3, \ldots], [x_2, x_4, \ldots])$$

5. Implement `msort` :: $[\mathsf{Int}] \to [\mathsf{Int}]$.