Discussion Section: Folds

2021/11/05
Higher-Order Functions

- Takes one or more functions as an argument, or
- Returns a function
- Examples:
  - map
  - filter
  - foldr
  - foldl
Higher-Order Functions

\textbf{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]

- \textit{Maps} each element to a new value
- Polymorphic: maps a list of \textit{as} to a list of \textit{bs}.
- Returns a list of the same \textit{length}. 
Quiz

map (\x \to \ x `mod` \textcolor{red}{10}) [1, 2, 100, 85]

A. [1,2,100,85]
B. [0,0,10,8]
C. [1,2,0,5]
D. [5,0,2,1]
E. Type Error
Quiz

map (\x \rightarrow x \ `mod` 10) [1, 2, 100, 85]

A. [1, 2, 100, 85]
B. [0, 0, 10, 8]
C. [1, 2, 0, 5]
D. [5, 0, 2, 1]
E. Type Error
Higher-Order Functions

\[\text{filter} :: (\text{a} \rightarrow \text{Bool}) \rightarrow \text{[a]} \rightarrow \text{[a]}\]

- *Filters* the elements of the list.
- Doesn't change the elements, but
- May return a list of a different length
Quiz

filter (not . even) \([1,2,3,4,5,6]\)

A. \([1,2,3,4,5,6]\)
B. \([2,4,6]\)
C. \([1,3,5]\)
D. \([6,4,2]\)
E. None of the above
Quiz

filter (not even) [1,2,3,4,5,6]

A. [1,2,3,4,5,6]
B. [2,4,6]
C. [1,3,5]  **(Correct Answer)**
D. [6,4,2]
E. None of the above
Quiz

filter (not . even) [1,2,3,4,5,6]

A. [1,2,3,4,5,6]
B. [2,4,6]
C. [1,3,5]
D. [6,4,2]
E. None of the above

Function composition:

( . ) :: (b -> c) -> (a -> b) -> a -> c
Higher-Order Functions

foldl :: (b → a → b) → b → [a] → b
foldl f b xs = helper b xs
  where
    helper b [] = b
    helper b (x:xs) = helper (f b x) xs
Higher-Order Functions

\[
\text{foldl} :: (b \to a \to b) \to b \to [a] \to b
\]
\[
\text{foldl } f \ b \ xs = \text{helper } b \ xs
\]
where

\[
\text{helper } b \ [\ ] = b
\]
\[
\text{helper } b \ (x:xs) = \text{helper } (f \ b \ x) \ xs
\]
Higher-Order Functions

foldl :: (b \( \rightarrow a \rightarrow b \)) \( \rightarrow b \rightarrow [a] \rightarrow b \)

cat :: [String] \( \rightarrow \) String

cat xs = foldl (++) "" xs
Higher-Order Functions

foldl :: (b → a → b) → b → [a] → b

cat :: [String] → String
cat xs = foldl (++) "" xs
Higher-Order Functions

foldl :: (b → a → b) → b → [a] → b

cat :: [String] → String
cat xs = foldl (++) "" xs
Higher-Order Functions

foldl :: (b → a → b) → b → [a] → b

cat :: [String] → String

cat xs = foldl (++) "" xs
Higher-Order Functions

foldl :: (b -> a -> b) -> b -> [a] -> b

```haskell
cat :: [String] -> String
cat xs = foldl (++) "" xs
```

![Diagram of a and b connected, followed by c]
Higher-Order Functions

foldl :: (b → a → b) → b → [a] → b

cat :: [String] → String

\[
\text{cat } \text{xs} = \text{foldl } (\text{++}) \ " \ \text{xs}
\]
Higher-Order Functions

foldl :: (b → a → b) → b → [a] → b

cat :: [String] → String

cat xs = foldl (++) "" xs
Higher-Order Functions

foldl :: (b → a → b) → b → [a] → b

cat :: [String] → String
cat xs = foldl (++) "" xs
Higher-Order Functions

foldl :: (b → a → b) → b → [a] → b

\[ \text{cat :: [String]} → \text{String} \]

\[ \text{cat} \text{ } \text{xs} = \text{foldl} \ (++) \ "" \ \text{xs} \]
Higher-Order Functions

foldl :: (b \rightarrow a \rightarrow b) \rightarrow b \rightarrow [a] \rightarrow b

cat :: [String] \rightarrow String

cat xs = foldl (++) "" xs
Higher-Order Functions

```haskell
foldl :: (b -> a -> b) -> b -> [a] -> b
cat :: [String] -> String
cat xs = foldl (++) "" xs
```

```
"a"
"b"
"c"
```

"abc"
Higher-Order Functions

foldr :: (a → b → b) → b → [a] → b
Higher-Order Functions

foldr :: (a → b → b) → b → [a] → b

cat :: [String] → String

cat xs = foldr (++) "" xs
Higher-Order Functions

foldr :: \( (a \rightarrow b \rightarrow b) \rightarrow b \rightarrow [a] \rightarrow b \)

cat :: [String] \rightarrow String

cat xs = foldr (++) "" xs
Higher-Order Functions

foldr :: (a → b → b) → b → [a] → b

cat :: [String] → String
cat xs = foldr (++) "" xs
Higher-Order Functions

foldr :: (a -> b -> b) -> b -> [a] -> b

cat :: [String] -> String
cat xs = foldr (++) "" xs
Higher-Order Functions

foldr :: (a → b → b) → b → [a] → b

cat :: [String] → String
cat xs = foldr (++) "" xs
Higher-Order Functions

foldr :: (a → b → b) → b → [a] → b

cat :: [String] → String
cat xs = foldr (++) "" xs
Higher-Order Functions

foldr :: (a → b → b) → b → [a] → b

`cat :: [String] → String`
cat xs = foldr (++) "" xs
Higher-Order Functions

\[
\text{foldr} :: (a \rightarrow b \rightarrow b) \rightarrow b \rightarrow [a] \rightarrow b
\]

\[
\text{cat} :: [\text{String}] \rightarrow \text{String}
\]

\[
\text{cat} \; \text{xs} = \text{foldr} \; (\text{+++}) \; "\" \; \text{xs}
\]
Higher-Order Functions

foldr :: (a → b → b) → b → [a] → b

cat :: [String] → String

cat xs = foldr (++) "" xs
Higher-Order Functions

\[\text{foldr} :: (a \to b \to b) \to b \to [a] \to b\]

\[\text{cat} :: [\text{String}] \to \text{String}\]

\[\text{cat } xs = \text{foldr } (++) \ "\ " \ xs\]
Higher-Order Functions

foldr :: (a → b → b) → b → [a] → b

cat :: [String] → String

\[ \text{cat } xs = \text{foldr } (\text{++}) \ " \ xs \]

"abc"

"a" -- "b" -- "c"
foldl vs foldr

-- Left:
foldl (++) "" ["a", "b", "c"] \( \Rightarrow (("" \oplus "a") \oplus "b") \oplus "c"\)

-- Right:
foldr (++) "" ["a", "b", "c"] \( \Rightarrow "a" \oplus ("b" \oplus ("c" \oplus ""))\)
Quiz: foldl vs foldr

foldr (-) 0 [1,2,3,4]

A. [1,2,3,4]
B. -10
C. 0
D. -2
E. None of the above
Quiz: foldl vs foldr

foldr (-) 0 [1,2,3,4]

A. [1,2,3,4]
B. -10
C. 0
D. -2
E. None of the above
Quiz: foldl vs foldr

\[
\text{foldr} \ (-) \ 0 \ [1,2,3,4] \quad \quad \quad \quad \quad 1 - (2 - (3 - (4 - 0)))
\]

A. \ [1,2,3,4] \quad \quad \quad = 1 - (2 - (3 - 4))
B. \ -10 \quad \quad \quad = 1 - (2 - (-1))
C. \ 0 \quad \quad \quad = 1 - 3
D. \ -2 \quad \quad \quad = -2
E. \ None \ of \ the \ above
Quiz: foldl vs foldr

foldl (-) 0 [1,2,3,4]

A. [1,2,3,4]
B. -10
C. 0
D. -2
E. None of the above
Quiz: foldl vs foldr

foldl (-) 0 [1,2,3,4]

A. [1,2,3,4]
B. -10
C. 0
D. -2
E. None of the above
Quiz: foldl vs foldr

foldl (\(\neg\)) 0 [1, 2, 3, 4]

A. [1, 2, 3, 4]
B. -10
C. 0
D. -2
E. None of the above
Quiz: foldl vs foldr

foldl (-) 0 [1,2,3,4]

A.  [1,2,3,4]
B.  **-10**
C.  0
D.  -2
E.  None of the above

\[
\begin{align*}
((0 - 1) - 2) - 3) - 4 \\
= (-1 - 2) - 3) - 4 \\
= (-3 - 3) - 4 \\
= -6 - 4 \\
= -10
\end{align*}
\]
Practice

reverse :: [a] -> [a]
reverse xs = foldl f base xs

where
  f a x =

base =
Practice

last :: [a] -> a

last [] = error "last: empty list"

last (x:xs) = foldl f base xs

  where

    f a x =

    base =
Practice

append :: [a] -> [a] -> [a]
append xs ys = foldr f base l
  where
    f x a =
    base =
    l =
Practice

map :: (a -> b) -> [a] -> [b]

map f xs = foldr fold_fun base xs
Practice

filter :: (a -> Bool) -> [a] -> [a]
filter p xs = foldr f base xs