CS 558 5 September 2024

Note Title 2024-09-05

```
(filter (\n -> n `mod` 2 == 0)) . (map (\n -> n * 3))
```

[Int] -> [Int] Int -> Bool Int -> Int [Int] -> [Int]

Review of function composition (.):

$(f \cdot g)\ x = f\ (g\ x)$

[Int] -> [Int]

```
map (filter (\n -> n `mod` 2 == 0)) [[1,2,3],[4],[5]]
```

(a->b) -> [a] -> [b] Int -> Bool [Int] -> [Int] [[Int]]

evaluates to

```
[[2],[4],[]]
```

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr f v []     = v
foldr f v (x:xs) = f x (foldr f v xs)
```

Example

```
foldr f v [x0, x1]
= foldr f v (x0 : (x1 : []))
= f x0 (foldr f v (x1 : []))
= f x0 (f x1 (foldr f v []))
= f x0 (f x1 v)
```

Can be written as

```
= x0 `f` (x1 `f` v)
```

What is

```
foldr (:) [] = id

foldr (+) 0 = sum

foldr (\x -> \acc -> f x : acc) [] = map f
```

```
f = \n -> n `mod` 2 == 0
foldr (\x -> \acc -> f x : acc) [] [1,2]
= (\x -> \acc -> f x : acc) 1 (foldr (\x -> \acc -> f x : acc) [] [2])
= (\x -> \acc -> f x : acc) 1 ((\x -> \acc -> f x : acc) 2 (foldr (\x -> \acc -> f x : acc) [] []))
= (\x -> \acc -> f x : acc) 1 ((\x -> \acc -> f x : acc) 2 [])
= (\x -> \acc -> f x : acc) 1 [True]
= [False, True]
```

```
map f = foldr ((:) . f) []
```

```
filter p = foldr f []
  where
    f x acc = if p x then x : acc else acc
```

---

foldl $[x_0, x_1, x_2]$

$x_0 \oplus (x_1 \oplus (x_2 \oplus e))$

```
foldl :: (b -> a -> b) -> b -> [a] -> b
foldl f v [] = v
foldl f v (x:xs) = foldl f (f v x) xs
```