Objectives

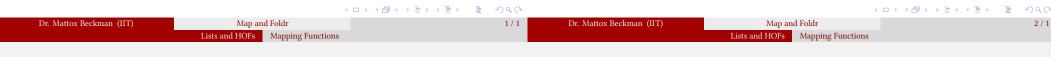
Objectives

Map and Foldr

Dr. Mattox Beckman

Illinois Institute of Technology Department of Computer Science

- Define *higher order function* and give some examples.
- Define the foldr and map functions.
- Use foldr and map to implement recursion patters we saw earlier.
- Understand the lambda form and how to use eta-expansion.



Mapping functions the hard way

Dr. Mattox Beckman (IIT)

Mattox's Law of Computing

What do the following definitions have in common?

```
Example 1

1 incl [] = []

2 incl (x:xs) = x+1 : incl xs

Example 2

1 doublel [] = []

2 doublel (x:xs) = x*2 : doublel xs
```

The computer exists to work for us; not us for the computer. If you are doing something repetitive for the computer, you are doing something wrong.

Stop what you're doing and find out how to do it right.

Map and Foldr

Lists and HOFs Mapping Functions Lists and HOFs Folding functions

Folding functions

Mapping functions the easy way

Map Definition

$$map f[x_0, x_1, \dots, x_n] = [f x_0, f x_1, \dots, f x_n]$$

```
_{1} map :: (a->b) -> [a] -> [b]
2 map f [] = []
_3 map f (x:xs) = f x : map f xs
5 incL = map inc
7 doubleL = map double
```

- inc and double have been transformed into recursive functions.
- I dare you to try this in Java.

```
Example 1
                                       Example 2
1 sumL [] = 0
                                      1 prodL [] = 1
                                      _2 prodL (x:xs) = x * prodL xs
_2 sumL (x:xs) = x + sumL xs
```

What do the following definitions have in common?

```
◆□▶◆□▶◆壹▶◆壹▶ 壹 め९℃
                                  Map and Foldr
                                                                                           Dr. Mattox Beckman (IIT)
                                                                                                                             Map and Foldr
Dr. Mattox Beckman (IIT)
                            Lists and HOFs Folding functions
                                                                                                                       Lists and HOFs Folding functions
```

foldr

Fold Right Definition

$$foldr f z [x_0, x_1, \ldots, x_n] = f x_0 (f x_1 \cdots (f x_n z) \cdots)$$

• To use foldr, we specify the function and the base case.

```
1 foldr :: (a -> b -> b) -> b -> [a] -> [b]
2 foldr f z [] = z
_3 foldr f z (x:xs) = f x (foldr f z xs)
5 sumlist = foldr (+) 0
6 prodlist = foldr (*) 1
```

Encoding Recursion using fold

```
Recursive Style
1 flatten [] = []
2 flatten (x:xs) = x ++ flatten xs
```

- Notice the pattern between the recursive version and the higher order function version.
- Note well: the second parameter of the function argument to foldr represents the *result* of the recursive call.

```
Higher Order Style
1 flatten = foldr (++) []
```

Lists and HOFs Folding functions Lists and HOFs Folding functions

Other Functions

Something to think about...

- You can write map using fold. Try it!
- Note, the reverse direction will not work. Why not?

Other Examples

- foldl
- zipWith
- exists
- filter
- fix
- There are many many other HOFs defined.
- Learn them; they will help you.





Dr. Mattox Beckman (IIT) Map and Foldr 9 / 1 Dr. Mattox Beckman (IIT) Map and Foldr 10 / 2