

Structured Traversals for (Multiply) Recursive Algebraic Datatypes

G. Cassian Alexandru

January 11, 2021

Presentation generated from .lhs sources using lhs2TeX

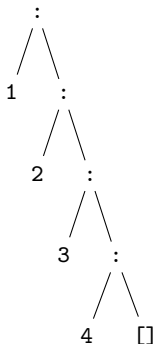
```
length :: [a] → Int
length [] = 0
length (x:xs) = 1 + length xs
```

```
filter :: (a → Bool) → [a] → [a]
filter p = go where
  go [] = []
  go (x:xs) = if p x then [x] else [] ++ go xs
```

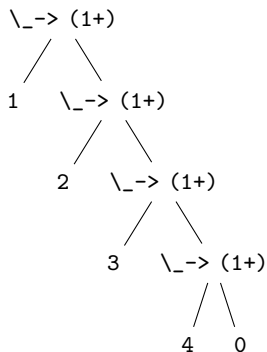
- List Design pattern?
- Design Patterns are a poor man's abstraction
- Recognize common structure & find correct abstract notion

Traversals

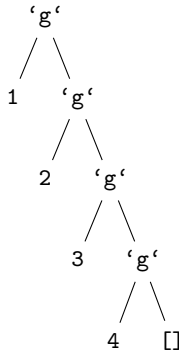
List



length



filter p



g x xs =

bool [] [x] (p x) ++ xs

```
data List a = Nil | Cons a (List a)
```

```
data List a where
```

```
  Nil :: List a
```

```
  Cons :: a → (List a) → (List a)
```

$$F : \mathcal{C} \rightarrow \mathcal{C}, A, B, A_0 \in \mathcal{C}_0$$

Algebra

$$\begin{array}{c} FA \\ \downarrow \phi \\ A \end{array}$$

Algebra-Hom:

$$(A, \phi) \rightarrow (B, \psi)$$

$$\begin{array}{ccc} FA & \xrightarrow{Ff} & FB \\ \downarrow \phi & & \downarrow \psi \\ A & \xrightarrow{f} & B \end{array}$$

Initial Algebra:

$$(A, \kappa)$$

$$\begin{array}{ccc} FA & \xrightarrow{Fh} & FB \\ \kappa^{-1} \uparrow \downarrow \kappa & & \downarrow \psi \\ A & \xrightarrow{h} & B \end{array}$$

Initiality requirement: $h = \kappa^{-1}; Fh; \psi$

As Program

```
newtype Fix f = In { out :: f (Fix f) }  
  
type Algebra f c = f c → c  
  
cata :: Functor f => Algebra f a → Fix f → a  
cata alg = alg · fmap (cata alg) · out
```