Lecture 3

```
cata :: Bifunctor f => (f a b -> b) -> Mu f a -> b
cata phi (In x) = phi (bimap id (cata phi) x)

data Mu f a = In (f a (Mu f a))

para :: Bifunctor f => (f a (b, Mu f a) -> b) -> Mu f a -> b
para phi (In x) = phi (bimap id (~~cata~~ para phi Δ id) x)
(para phi) Δ id
```

```
dropWhile :: (a -> Bool) -> [a] -> [a]
dropWhile p []     = []
dropWhile p (x:xs) = if p x then dropWhile p xs
                            else x:xs

dropWhile even [2,4,5,6,8]

dropWhile p (x:xs)    = ... x ... dropWhile p xs ... ?
dropWhile even [5,6,8] = ... 5 ... dropWhile even [6,8] ...
                                   []

h (x:xs) = ... x ... h xs ... xs ...
```

$$\begin{tikzcd}
F(\mu F) \arrow[r, "F(\mathrm{cata}\ phi)"] \arrow[d, "in"'] & FA \arrow[d, "phi"] \\
\mu F \arrow[r, "\mathrm{cata}\ phi"'] & A
\end{tikzcd}$$

$$\begin{tikzcd}
F(\mu F) \arrow[rr, "F\langle h, id\rangle"] \arrow[dr, "Fh"'] \arrow[rrr, bend left, "F\,id"] \arrow[dd, "in"'] & & F(A \times \mu F) \arrow[r, "F\,snd"] \arrow[dl, "F\,fst"] \arrow[dd, "phi"] & F(\mu F) \\
& FA & & \\
\mu F \arrow[rr, "h = \mathrm{para}\ phi"'] & & A &
\end{tikzcd}$$

every cata is a para

cata phi = para psi <u>where</u> psi = ...phi...

every para is a cata with some post-processing

para phi = post ∘ cata psi <u>where</u> psi = ...phi...

post = fst

<u>Zygomorphism</u>

for para, $h' = id$

$h(x:xs) = \ldots x \ldots h\ xs \ldots h'\ xs \ldots$

<u>data</u> Tree a = Tip a | Bin (Tree a) (Tree a)

```
perfect :: Tree a → Bool
perfect (Tip x) = True
perfect (Bin t u) = perfect t ∧ perfect u ∧ ht t == ht u

ph :: Tree a → (Bool, Int)
ph (Tip x) = (True, 1)
ph (Bin t u) = let (b,m) = ph t; (c,n) = ph u in (b∧c∧m==n,
                                                  1+max m n)
```

```
zygo :: Bifunctor f => (f a (b,c) -> b) -> (f a c -> c) -> Mu f a -> b
zygo phi psi = fst . zygo' phi psi

zygo' :: Bifunctor f => (f a (b,c) -> b) -> (f a c -> c) -> Mu f a -> (b,c)
zygo' phi psi (In x) = let x' = bimap id (zygo' phi psi) x in
                        (phi x', psi (bimap id snd x'))
```

every para is a zygo ...

every zygo is a post-processed cata ...

mutumorphism

```
mutu :: Bifunctor f ⇒ (f a (b,c) → b) → (f a (b,c) → c) →
                    Mu f a → b
```

eg minimax

histomorphism "course-of-values iteration"

$h\ (x : xs) = \ldots x \ldots h\ xs \ldots \quad \leftarrow$ cata

$wvs :: [(Int, Rat)]$

$knapsack :: \overset{Nat}{\cancel{Int}} \to Rat$

$knapsack\ c = maximum_\phi\ [v + knapsack\ (c - w)$
$\quad \mid (w, v) \leftarrow wvs,\ 0 \leq w \leq c]$

$histo :: Bifunctor\ f \Rightarrow (f\ a\ (Cofree\ (f\ a)\ b) \to b) \to$
$\quad Mu\ f\ a \to b$

cofree comonad on f a

$histo\ phi = root \circ cata\ (\lambda x \to Node\ (phi\ x)\ x)$

Cofree comonad

```
data Cofree f b = Node b (f (Cofree f b))
data Mu f = In (f (Mu f))

root :: Cofree f b -> b
root (Node x t) = x
subs :: Cofree f b -> f (Cofree f b)
subs (Node x t) = t
```

data L a b = Nil | Cons a b

what is Cofree (L a) c? effectively,

data LabList a c = LabNil c
    | LabCons c a (LabList a c)

Duality

para	$f\,a\,(b, x\,\mathrm{Mu}\,f\,a)$	apo	$f\,a\,(b + \mathrm{Nu}\,f\,a)$
zygo		cozygo	
muto		comutu	
histo		futu	