Lecture 2

\[ NX = 1 + X \]
\[ LA X = 1 + A \times X \]
\[ TA X = A + X \times X \]

Least fixpoint of functor \( F \) is 
\((X, \text{in} : FX \rightarrow X)\) s.t.
\[ \forall (A, f : FA \rightarrow A) \]
\[ \exists! h : X \rightarrow A \quad h \circ \text{in} = f \circ F h \]

\[ \text{ MF } \]
\[ \text{ Fh } \]
\[ \text{ FA } \]
\[ \text{ in } \downarrow \]
\[ \text{ f } \]
\[ \text{ MF } \rightarrow \rightarrow A \]
\[ \text{ cataf } \]

Write \( MF \) for \( X \)
cataf for \( h \)

\[ X = M(\text{Fa}) \]
\[ \text{data } \{ a : b \} \]  
\[ \text{data } \text{List } a = \text{Mu } \mathbb{L} \]  
\[ \text{type } u = \text{Mu } a \]  

\[ \text{f} : \text{Mu } a \rightarrow \text{Mu } a \]  
\[ \text{in} \circ \text{f} = \text{id} \]  
\[ \text{in} ^\circ \text{f} = \text{id} \]
class bifunctor f where
  bimap :: (a -> c) -> (b -> d) -> f a b -> f c d

instance bifunctor L where
  bimap f g Nil = Nil
  bimap f g (Cons x y) = Cons (f x) (g y)

class phi (f :: H x y) where
  phi :: bifunctor id bifunctor phi

f :: (a -> b) -> f a a -> f (f a) a
f a = phi (f . a)
class Functor f where

  fmap :: (a -> b) -> fa -> fb

instance Bifunctor f => Functor (Mu f) where

  fmap f (In xc) = In (bimap f (fmap f) xc)

  a \otimes b \mapsto Mu fa

  fmap f = cata (\mu f b \mapsto b (Mu f b))
analyze - functor \( F \)

least fp ("initial algebra") is
\[
\begin{array}{c}
x \quad \text{in: } FX \to X \\
\forall (A, \phi_i : FA \to A), M \quad \text{NF}
\end{array}
\]

\[\exists h : X \to A, \quad \text{cata}_{\phi_i} h\]
\[h \circ \text{in} = \phi_i \circ F h\]

\[FX \xrightarrow{Fh} FA\]

\[\downarrow \phi_i\]

\[x \quad \text{h} \to A\]

\[\text{greatest fp ("final coalgebra") is}
\]

\[
\begin{array}{c}
x \quad \text{out: } X \to FX \\
\forall (A, \phi_i : A \to FA), M \quad \text{NF}
\end{array}
\]

\[\exists ! h : A \to X, \quad \text{ana}_{\phi_i} h\]
\[\text{out} \circ h = F h \circ \phi_i\]

\[FX \xrightarrow{F h} FA\]

\[\uparrow \phi_i\]

\[x \quad \text{h} \to A\]
```haskell
data L a b = Nil | Cons a b

type CoList a = Nu L a

range :: (Int, Int) -> CoList Int
range = ana next where

next :: (Int, Int) -> L Int (Int, Int)
next (m, n)
  | m == n = Nil
  | otherwise = Cons m (m+1, n)

range (0,3) = [0,1,2] ⌊ ⌋
range (0,0) = []
range (3,2) = [3,4,5...]
```
data U a b = Empty | Fork b a a b

type Ctree a = Nu U a

build :: [a] → Ctree a
build = ana next where
  next :: [a] → U a [a]
  next [] = Empty
  next (x:xs) = Fork ys x zs where
    ys = [y | y ← xs, y < x]
    zs = [z | z ← xs, z ≥ x]
$\text{Out}^0 \circ \text{fmap} \circ \text{length} \circ \phi = \phi$ (phi z)

$\text{Conat} = \text{Nil} \circ \text{Maybe}

\text{type}

\text{Conat} :: (\text{b} + \text{Maybe} \circ \text{b}) \rightarrow \text{b} \rightarrow \text{Conat}

\text{ana} \ circ \phi \circ z = \text{Out}^0 \circ \text{fmap} \circ \phi \circ z$

$\text{Unfold} \circ \text{Conat} :: \text{b} + \text{Maybe} \circ \text{b} \rightarrow \text{b} + \text{Conat}$

$\text{data} \ \text{Nu} \ f = \text{Out}^0 \circ \text{fmap} \circ \phi \circ f \circ \text{Nu} \ f$

$\text{Nu} \ f \circ \text{fmap} \circ f \circ \text{Nu} \ f$