$\boxed{\text{Lec 11}}$

aka point-wise

Combinators — func that "combines"/"operates on" funcs

e.g. (op o)

Math: $(f+g)(x) = f(x) + g(x)$ ← Do that!

ML:

```
infix ++
fun (f ++ g)(x) = f(x) + g(x)
infix **
fun (f ** g)(x) = f(x) * g(x)
fun MAX(f,g)(x) = Int.max (f(x), g(x))
```

} : ('a -> int) * ('a -> int) -> ('a -> int)

Applications:

```
fun id x = x
val double = id ++ id
val square = id ** id
val quadratic = double ++ square      ≅   fn x => x*x + 2*x
```

# Staging

Consider:

(int * int) -> int

```
fun f1 (x:int, y:int) : int =
  let
    val z : int = horrible_computation (x)
  in
    z + y
  end
```

takes a year :/

somehow stash the result

How to do better?

```
f1(5,10)  — a year
f1(5,7)   — another year   ...   :C
```

→ try...

```
           int → int → int
fun f2 (x: int) (y: int) : int =
  let
    val z : int = horrible_computation (x)
  in
    z + y
  end
```

```
f2 5 10   ] Umm... uh oh... also takes 2 years
f2 5 7
```

```
val f2' = f2 5      — takes ε        ]
f2' 10              — takes 1 year   ] :c Problem not solved
f2' 7               — takes 1 year   ]
```

But we're trying to compute as soon as first arg given

→ Try:

```
fun f3 ( x: int ): int → int =
  let
    val z = horrible_computation (x)
  in
    fn (y: int) => y + z
  end
```

```
val f3' = f3 5     — a year
f3' 10             — ε
f3' 7              — ε
```

Catamorphism aka natural fold

Recall:

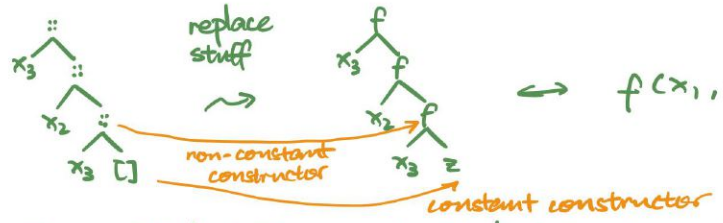
```
(* map : ('a -> 'b) -> 'a list -> 'b list *)

datatype 'a tree = Empty | Node of 'a tree * 'a * 'a tree
```

Now do same to tree

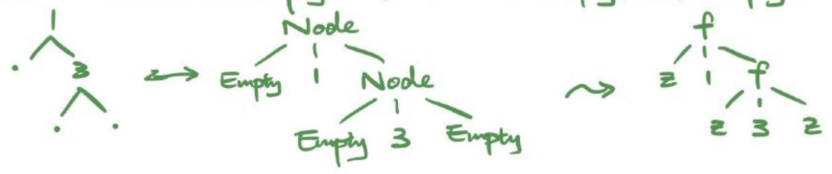
```
(* tmap : ('a -> 'b) -> 'a tree -> 'b tree *)
fun tmap f Empty = Empty
  | tmap f (Node (L, x, R)) = Node (tmap f L, f x, tmap f R)
```

Consider $[x_1, x_2, x_3]$. We can write it as



$\longleftrightarrow \quad f(x_1, f(x_2, f(x_3))) \cong$ foldr f z $[x_1, x_2, x_3]$

Consider Node (Empty, 1, Node (Empty, 2, Empty))



```
(* tfold : ('b * 'a * 'b -> 'b) -> 'b -> 'a tree -> 'b
   fun tfold f z Empty = z
     | tfold f z Node(L, x, R) = f(tfold f z L, x, tfold f z R)
```

Some other tree

```
datatype 'a tree = Leaf of 'a | Node of 'a tree * 'a tree
```

$$(* \text{ tfold } \overbrace{('a \to 'b)}^{\text{do for Leaf}} \to \overbrace{('b * 'b \to 'b)}^{\text{do for node}} \to 'a \text{ tree} \to 'b *)$$

```
fun tfold g f (Leaf x) = g x
  | tfold g f Node(L,R) = f(tfold g f L, tfold g f R)
```

Hmm...

```
datatype 'a option = NONE | SOME of 'a

(* optfold : 'b -> ('a -> 'b) -> 'a option -> 'b )
fun optfold z f NONE = z
  | optfold z f (SOME x) = f x
```

# Bonus

* Connection btwn. structural induction & fold