COURSE NAME: CS558

COURSE DATE: 9/4/18

1. LECTURE
   1. Homework is due tomorrow night.
   2. Last time we talked about List Functionals

map (a -> b) -> [a] -> [b]

map f [ ] : [ []

map f (x: xs) = (fx) : (map fxs)

filter (a=> Bool) -> [a] -> [a]

filter p [ ] = [ ]

filter p (x:xs) = if p x then x (filter x xs)

else filter x xs

* + 1. We looked at a folding example, with summing and multiplying the list:

foldr (a -> b -> b) -> b -> [a] -> b

foldr f v [ ] = v

foldr f v ( x xs) = f x (fold fv xs)

foldr (+) 0 [1, 2, 3] = 1 + (2 + (3+ 0))

Opposite side:

foldl.. (b-> a -> a) -> b -> [a] -> b

foldl f v [ ] = v

foldl f v (x xs) = foldl f (f v x) xs

foldl (+) 0 [1, 2, 3] = ((0 + 1) +2) +3

* + - 1. Remember that plus is associative.
         1. But you get different results for non-associative operators.
      2. Then we looked at how to implement map and filter using foldr.
    1. Formal proofs of connections 🡪 we can do this for all lists.
       1. More general way: we have some properly to prove for all lists L.
          1. Remember, we have to prove the base case and then we have to prove the inductive step.
    2. We can do this for all lists. To prove all lists, then first prove the property holds for the empty lists 🡪 All Lists P(l) so P ([ ] )
       1. Then prove that it holds for all x and xs 🡪 All x and All xs P(xs) 🡪 P (x xs)
    3. Example:

Let sum [ ] = 0

sum (x xs) = x + sum (xs)

sum all L foldr (+) 0 L = sum L

* + - * 1. Prove the base case:

foldr (+) 0 [ ] = sum [ ]

foldr (+) 0 [ ] = 0

sum [ ] = 0

* + - * 1. Prove inducive case

All x all xs (foldr (+) 0 xs = sum xs)

(foldr (+) 0 (x : xs) = sum (x: xs)

Assume inductive hypothesis 🡪 foldr (+) 0 xs = sum xs

Must show foldr (+) 0 (x: xs) = sum (x: xs)

Start on one side and rewrite to the other:

foldr (+) 0 (x: xs) = x + (foldr (+) 0 xs) 🡪 by definition of foldr!

= x + (sum xs) -> by induction hypo.

* + - 1. Implementing map and filter using foldr:

All f All L map f L = foldr (\x -> \acc -> (f x) acc ) [ ] L

Prove base case:

map f [ ] = fpldr (\x -> \ acc -> (f x ) acc) [ ] [ ]

map f [ ] = [ ]

foldr (\x -> 1acc -> (fx) acc) [ ] [ ] = [ ]

Inductive step:

Assume map f xs = foldr (\x -> \acc -> (f x) : acc) [ ] xs

Show that map f (x : xs) = fold (\x -> \acc -> (fx) acc) [ ] (x: xs)

Start with foldr side:

foldr (\x -> \acc -> (fx ) : acc) [ ] (x: xs)

=( \x -> \acc -> 9f x) acc) x (foldr (\x -> \acc -> (fx \:acc) [ ] xs)

By IH:

= ( \x -> \acc -> (fx) acc) x (map f xs)

= (fx ): (map f xs)

= map f (x: xs)

* + - 1. Example:

All p all L filter p L = foldr (\x -> \acc -> if p x then x: acc else acc)

Base case:

filtr p [ ] = foldr ( \x -> \acc -> if p x then x: acc else acc) [ ] [ ]

filter p [ ] = [ ]

foldr (\x -> \acc -> if p x then x: acc else acc) [ ] [ ] = [ ]

Inductive:

(L = x: xs for same x, same Xs)

We assume; filter p xs – foldr (\x -> \acc -> if p x then x: acc else acc) [ ] xs

Rewrite as:

foldr (\x -> \acc -> if p x then x: acc else acc) [ ] (x: xs)

= (\x -> \acc -> if p x then x: acc else acc) x (foldr q [ ] xs

(q is the above expression!)

By IH:

= (\x -> \acc -> if p x then x: acc else acc) x (filter p xs)

By function application, substitute:

= If p x then x: (filter p xs) else (filter p xs)

= filter p (x: xs)

##end notes##