

1 foldl scanl

```
inits :: [a] -> [[a]]
inits []      = [[]]
inits (x:xs) = []:(map (x:) (inits xs))
```

```
scanl :: (a -> b -> a) -> a -> [b] -> [a]
scanl f a []      = [a]
scanl f a (x:xs) = a:(scanl f (f a x) xs)
```

```
foldl :: (a -> b -> a) -> a -> [b] -> a
foldl f a []      = a
foldl f a (x:xs) = foldl f (f a x) xs
```

Claim 1.1.

`map (foldl f a).inits = scanl f a`

Lemma 1.2.

`map (foldl f a) (map (x:) ys) = map (foldl f (f a x)) ys`

Proof. of Lemma 1.2 by structural induction on `ys`
`ys=[]`

```
map (foldl f a) (map (x:) ys)
  = map (foldl f a) (map (x:) [])
  = map (foldl f a) []
  = []
  = map (foldl f (f a x)) []
  = map (foldl f (f a x)) ys
```

`ys=(y:ys')`

```
map (foldl f a) (map (x:) ys)
  = map (foldl f a) (map (x:) (y:ys'))
  = map (foldl f a) ((x:y):(map (x:) ys'))
  = (foldl f a (x:y)):(map (foldl f a) (map (x:) ys'))
   $\stackrel{\text{IH}}{=}$  (foldl f a (x:y)):(map (foldl f (f a x)) ys')
  = (foldl f (f a x) y):(map (foldl f (f a x)) ys')
  = map (foldl f (f a x)) (y:ys')
  = map (foldl f (f a x)) ys
```

□

Proof. of Claim 1.1

we show the equivalent claim

`map (foldl f a) (inits xs) = scanl f a xs`

by structural induction on `xs`

`xs=[]`

```
map (foldl f a) (inits xs)
  = map (foldl f a) (inits [])
  = map (foldl f a) [[]]
  = [foldl f a []]
  = [a]
  = scanl f a []
  = scanl f a xs
```

`xs=(x:xs')`

```
map (foldl f a) (inits xs)
  = map (foldl f a) (inits (x:xs'))
  = map (foldl f a) ([]:(map (x:) (inits xs')))
  = (foldl f a []):(map (foldl f a) (map (x:) (inits xs')))
  = a:(map (foldl f a) (map (x:) (inits xs')))
1,2 = a:(map (foldl f (f a x)) (inits xs'))
IH = a:(scanl f (f a x) xs')
  = scanl f a (x:xs')
  = scanl f a xs
```

□