1 map concat

===

```
(.) :: (b -> c) -> (a -> b) -> a -> c
f . g = \x -> f (g x)

concat :: [[a]] -> [a]
concat []     = []
concat (x:xs) = x++(concat xs)

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

Claim 1.1.

```
map f.concat = concat.map (map f)
```

Proof. structural induction on xs
`xs=[]`

```
(map f.concat) xs = (map f.concat) []
                  = map f (concat [])
                  = map f []
                  = []
                  = concat []
                  = concat (map (map f) [])
                  = (concat.map (map f)) []
                  = (concat.map (map f)) xs
```

`xs=(x:xs')` structural induction on x
  `x=[]`

```
(map f.concat) xs = (map f.concat) (x:xs')
                  = map f (concat ([]:xs'))
                  = map f ([]++(concat xs'))
                  = map f (concat xs')
                  =IH (concat.map (map f)) xs'
                  = concat (map (map f) xs')
                  = []++(concat (map (map f) xs'))
                  = concat [[]]++(map (map f) xs')
                  = concat [map f []]++(map (map f) xs')
                  = concat (map (map f) ([]:xs'))
                  = concat (map (map f) xs)
                  = (concat.map (map f)) xs
```

  `x=(y:ys)`

```
(map f.concat) xs = (map f.concat) (x:xs')
                  = map f (concat ((y:ys):xs'))
                  = map f ((y:ys)++(concat xs'))
                  = [f y]++(map f (ys++(concat xs')))
                  = [f y]++(map f (concat (ys:xs')))
                 IH
                  = [f y]++((concat.map (map f)) (ys:xs'))
                  = [f y]++(concat (map (map f) (ys:xs')))
                  = [f y]++(concat [map f ys]++(map (map f) xs'))
                  = [f y]++(map f ys)++(concat (map (map f) xs'))
                  = (map f (y:ys))++(concat (map (map f) xs'))
                  = concat [map f (y:ys)]++(map (map f) xs')
                  = concat (map (map f) ((y:ys):xs'))
                  = concat (map (map f) (x:xs'))
                  = concat (map (map f) xs)
                  = (concat.map (map f)) xs
```

□

alternative proof

Proof. we show the equivalent claim
`map f (concat xs) = concat (map (map f) xs)`
structural induction on xs
`xs=[]`

```
map f (concat xs) = map f (concat [])
                  = map f []
                  = []
                  = concat []
                  = concat (map (map f) [])
                  = concat (map (map f) xs)
```

`xs=(x:xs')` structural induction on x
  `x=[]`

```
map f (concat xs) = map f (concat ([]:xs'))
                  = map f ([]++(concat xs'))
                  = map f (concat xs')
                 IH
                  = concat (map (map f) xs')
                  = []++(concat (map (map f) xs'))
                  = concat [[]]++(map (map f) xs')
                  = concat [map f []]++(map (map f) xs')
                  = concat (map (map f) ([]:xs'))
                  = concat (map (map f) xs)
```

  `x=(y:ys)`

```
map f (concat xs) = map f (concat ((y:ys):xs'))
                  = map f ((y:ys)++(concat xs'))
                  = [f y]++(map f (ys++(concat xs')))
                  = [f y]++(map f (concat (ys:xs')))
                 IH
                  = [f y]++(concat (map (map f) (ys:xs')))
                  = [f y]++(concat [map f ys]++(map (map f) xs'))
                  = [f y]++(map f ys)++(concat (map (map f) xs'))
                  = (map f (y:ys))++(concat (map (map f) xs'))
                  = concat [map f (y:ys)]++(map (map f) xs')
                  = concat (map (map f) ((y:ys):xs'))
                  = concat (map (map f) xs)
```

□