Más demostraciones

Federico Sawady

May 2019

1 Simples

```
1) (\x -> maybe x id Nothing) = head . (:[])
```

2 Sobre listas

```
1) factorial x = product (countFrom x)
countFrom :: Int -> [Int]
countFrom 0 = []
countFrom n = n : countFrom (n-1)

2) map f . concat = concat . map (map f)

3) replicate n x = applyN n (x:) []

4) snoc xs y = xs ++ [y]
snoc xs y = xs ++ [y]
snoc :: [a] -> a -> [a]
snoc [] y = [y]
snoc (x:xs) y = x : snoc xs y

5) last xs = head (reverse xs)
```

3 Sobre árboles binarios

```
1) sumT . mapT (const 1) = sizeT

2) sizeT = sizeT . mirrorT

3) allT f = andT . (mapT f)
```

4 Más sobre listas

```
1) sum (xs ++ ys) =
       sum (zipWith (+) xs ys)
(es falsa, dar contraejemplo)

2) filter p (xs ++ ys) = filter p xs ++ filter p ys

3) filter p (filter q xs) = filter (\y -> p y && q y) xs

4) filter p . map f = map f . filter (p . f)

5) takewhile p xs ++ dropwhile p xs = xs

6) applyN n f . applyN m f = applyN (n+m) f

7) applyN n (applyN m f) = applyN (n*m) f

8) applyN n (applyN m) = applyN (m^n)

9) applyN n f x = iterate f x !! n

10) (!!) n = head . drop n
```