Revisar, capaz faltan lemas!

(IV) $\forall xs :: [a] . \forall f :: (a \to b) .$ length (map f xs) = length xs

$\Rightarrow$)

Caso base $P([\,]) = \text{length (map f [ ])} \underset{m0}{=} \text{length [ ]} \underset{l0}{=} 0$ (1)

($\Leftarrow$) length [ ] = 0 (2)

(1) $\equiv$ (2)

Caso inductivo

$HI \equiv \forall ys :: [a] . \forall f :: a \to b$ length (map f ys) = length ys

con $|ys| < |(x:xs)|$

Qvq $P(ys) \Rightarrow P((x:xs))$

$P((x:xs))$ = length (map f (x:xs)) $\underset{m1}{=}$ length (f x : map f xs)

$\underset{L1}{=}$ 1 + length (map f xs) $\underset{HI}{=}$ 1 + length xs $\underset{L1\ izq}{=}$ length (x:xs) q.e.d

$\therefore$ Vimos que la propiedad vale para cualquier lista. y función $f :: a \to b$

(V) $\forall xs :: [a] . \forall p :: a \to Bool . \forall e :: a .$ *

* Asumir Eq a.

P = elem e (filter p xs) $\Rightarrow$ (elem e xs)

Esta demo está mal, Hay q' ver todos esos casos.
elem $<^T_F$ filter $<^T_F$

Nota: Estoy usando filter con su def. recursiva explícita

Caso Base: P ([ ])

P ([ ]) = elem e (filter p [ ]) $\underset{f0}{=}$ elem e [ ] $\underset{e0}{=}$ false.

(False → *) Es trivialmente cierto por falsedad del antecedente en la implicación

Caso Inductivo: P((x:xs))

"Si el elemento e estaba en la lista filtrada. Entonces, estaba en la original"

HI ≡ elem e (filter p ys) ⇒ (elem e ys)

$p \to q \equiv \neg p \vee q$ ¿Es más fácil probar eso?

P((x:xs)) = elem e (filter p (x:xs)) =

f1: separo en casos V y F.

P x : True

elem _ [] = F {e1}

elem x (y:ys) = x==y || elem x ys {e2}

= elem e (x : filter p xs) = e == x || elem e (filter p xs)

e2

= x == e || elem e xs = elem e (x:xs) (1)

¿HI?

e2 izq

P x : False

= elem e (filter p xs) ⇒ (elem e xs) (2) ya valía.

HI

∴ Por el caso base, (1) y (2) del paso inductivo la prop. vale

Dudoso cómo apliqué HI.

VI. $\forall$ xs :: [a] . $\forall$ x :: a . ponerAlFinal x xs = xs ++ (x : [])

P ([]) = ponerAlFinal x [] $\underset{\text{P0}}{=}$ foldr (:) (x : []) [] $\underset{\text{f1}}{=}$ (x : []) $\underset{\text{[] elem neutro}}{=}$ [] ++ (x : [])

P (x : xs) = ponerAlFinal x' (x : xs) $\underset{\text{P0}}{=}$ foldr (:) (x' : []) (x : xs) $\underset{\text{f1}}{=}$

= x : foldr (:) (x' : []) (xs) $\underset{\text{P0 1z}}{=}$ x : ponerAlFinal x' xs = x : (xs ++ x')

= (x : xs) : x'

= (x : xs) ++ [x']

No compro. Rehacer.