Proposición

```
((e==y) || elem e (filter p xs)) && p e
```

$\equiv$

```
elem e (filter p (y:xs)
```

Veamos por cassos:

Caso `p e = True`

Queremos ver que

```
((e==y) || elem e (filter p xs)) && True =_Bool (e==y) || elem e (filter p xs)
```

$\equiv$

```
elem e (filter p (y:xs)
```

Caso `(e==y) = True`

Por izquierda:

```
           (e==y) || elem e (filter p xs)
2*{Bool} = True
```

Por derecha:

```
Primero llamamos f = (\x acc -> if p x then x : acc else acc)
Luego:

            elem e (filter p (y:xs)
{Filter1} = elem e (foldr f [] (y:xs)
{Foldr1}  = elem e (\x acc -> if p x then x : acc else acc) y (foldr f [] xs)
Beta      = elem e (\acc -> if p y then y : acc else acc) (foldr f [] xs)
-- Como e == y:
          = elem e (\acc -> if p e then e : acc else acc) (foldr f [] xs)
Beta      = elem e (if p e then e : (foldr f [] xs) else (foldr f [] xs))
-- Como p e = True:
Defino primero g = (\x rec->(e==x)||rec)

          = elem e (e:(foldr f [] xs))
{E0}      = foldr g False (e:(foldr f [] xs))
{Foldr1}  = g e (foldr g False (foldr f [] xs))
2*Beta    = True || (foldr g False (foldr f [] xs))
          = True
```

Caso `(e==y) = False`

Por izquierda

```
  (e==y) || elem e (filter p xs)
= False || elem e (filter p xs)
= elem e (filter p xs)
```

Por derecha:

```
Defino h = (\x acc -> if p x then x : acc else acc)

            elem e (filter p (y:xs))
{Filter1} = elem e (foldr h [] (y:xs))
```

```
{Foldr1}  = elem e (h y foldr h [] xs)
          = elem e ((\x acc -> if p x then x: acc else acc) y foldr h [] xs)
Beta      = elem e ((\acc -> if p y then y: acc else acc) foldr h [] xs)
Beta      = elem e (if p y then y:(foldr h [] xs) else (foldr h [] xs))

-- Caso p y = False:
            elem e (foldr h [] xs)
{Filter1} = elem e (filter p xs) -- IGUALDAD!!!

-- Caso p y = True:
            elem e y:(foldr h [] xs)
{Filter1} = elem e y:(filter p xs)
{E0}      = foldr (\x rec -> (e==x) || rec) y:(filter p xs)
{Foldr1}  = (\x rec->(e==x) || rec) y (foldr (\x rec->(e==x) || rec) (filter p xs))
Beta      = (\rec -> (e==y) || rec) (foldr (\x rec->(e==x) || rec) (filter p xs))
-- Y sabemos e==y = False!
Bool      = (\rec -> False || rec) (foldr (\x rec->(e==x) || rec) (filter p xs))
Bool      = (\rec -> rec) (foldr (\x rec->(e==x) || rec) (filter p xs))
Etta      = foldr (\x rec->(e==x) || rec) (filter p xs)
{E0}      = elem e (filter p xs) -- IGUALDAD!!!
```

Caso `p e = False`

```
 ((e==y) || elem e (filter p xs)) && False
= False
```

≡

```
elem e (filter p (y:xs)
```

Vemos que:

```
Defino m = (\x acc -> if p x then x : acc else acc)

            elem e (filter p (y:xs))
{Filter1} = elem e (foldr (\x acc -> if p x then x : acc else acc) [] (y:xs))
{Foldr1}  = elem e ((\x acc -> if p x then x : acc else acc) y (foldr m [] xs))
Beta      = elem e ((\acc -> if p y then y : acc else acc) (foldr m [] xs))
Beta      = elem e (if p y then y:(foldr m [] xs) else (foldr m [] xs))
{Filter1} = elem e (if p y then y:(filter p xs) else (filter p xs))

-- Recordemos que la HI es : elem e xs && p e = elem e (filter p xs)

-- Veamos el caso p y = False:
       elem e (filter p xs))
{HI} = elem e xs && p e
-- Y sabemos que p e = False, luego:
     = elem e xs && False
     = False

-- Veamos el caso p y = True:
Defino j = (\x rec -> e == x || rec)

           elem e y:(filter p xs)
{E0}     = foldr j False y:(filter p xs)
{Foldr1} = (\x rec -> e == x || rec) y (foldr j False (filter p xs))
Beta     = (\rec -> e==y || rec) (foldr j False (filter p xs))
{E0}     = (\rec -> e==y || rec) (elem e (filter p xs))
```

```
Beta     = (e==y) || (elem e (filter p xs))
HI       = (e==y) || (elem e xs && p e)
-- Y tenemos p e = False, luego:
         = (e==y) || (elem e xs && False)
Bool     = (e==y) || False
Bool     = (e==y)
-- PASO IMPORTANTE!!!:
(p y = True) y (p e = False) => (e==y) = False
-- Esto es porque dan resultados distintos, entonces no pueden ser el mismo elemento!

-- Entonces:

(e==y) = False
```