# Trabajo Práctico 1 - Programación Funcional

Ellos foldearon a un personaje motorizado

20/09/2024

## Demostración ejercicio 9

De acuerdo a las definiciones de las funciones para árboles ternarios de más arriba, se pide demostrar lo siguiente:

`∀t :: AT a . ∀x :: a. (elem x (preorder t) = elem x (postorder t))`

Tenemos:

```
preorder :: AT a -> [a]
{pr0} preorder = foldAT [] (\r recI recM recD -> [r] ++ recI ++ recM ++ recD)

postorder :: AT a -> [a]
{po0} postorder = foldAT [] (\r recI recM recD -> recI ++ recM ++ recD ++ [r])

foldAT :: b -> (a -> b -> b -> b -> b) -> AT a -> b
{f0} foldAT cNil cNodo Nil = cNIl
{f1} foldAT cNil cNodo (Tern r i m d) -> cNodo r (rec i) (rec m) (rec d)
            where rec = foldAT cNil cNodo

elem :: (Eq a) => a -> [a] -> Bool
{e0} elem _ [] = False
{e1} elem x (y:ys) = (x == y) || elem x ys

(++) :: [a] -> [a] -> [a]
{++0} xs ++ ys = foldr (:) ys xs

foldr :: (a -> b -> b) -> b -> [a] -> b
{fr0} foldr f z []     = z
{fr1} foldr f z (x:xs) = f x (foldr f z xs)
```

Para demostrar lo siguiente:

$\forall$`t :: AT a .` $\forall$`x :: a. (elem x (preorder t) = elem x (postorder t))`

Haremos inducción sobre árbol ternario:

```
Tomo P(t): elem x (preorder t) = elem x (postorder t)

Caso base. Pruebo P(Nil):
elem x (preorder Nil) = elem x (postorder Nil)
{pr_0} elem x (foldAt [] (\r recI recM recD -> [r] ++ recI ++ recM ++ recD) Nil) =
elem x (postorder Nil)
{f_0} elem x [] = elem x (postorder Nil)
{po_0} elem x [] = elem x (foldAT [] (\r recI recM recD ->
recI ++ recM ++ recD ++ [r]) Nil)
{f_0} elem x [] = elem x []
{e_0} False = elem x []
{e_0} False = False ✓

Caso inductivo. Sup ∀ i,m,d :: AT a . ∀ r :: a . P(i) ∧ P(m) ∧ P(d)
qvq P(Tern r i m d)

elem x (preorder Tern r i m d) = elem x (postorder Tern r i m d)

{pr_0} elem x (foldAT [] (\r recI recM recD -> [r] ++ recI ++ recM ++ recD)
(Tern r i m d)) = elem x (postorder Tern r i m d)

{f_1} elem x ([r] ++
(foldAt [] (\r recI recM recD -> [r] ++ recI ++ recM ++ recD) i)++
(foldAt [] (\r recI recM recD -> [r] ++ recI ++ recM ++ recD) m)++
(foldAt [] (\r recI recM recD -> [r] ++ recI ++ recM ++ recD) d))
= elem x (postorder Tern r i m d)

{pr_0} elem x ([r] ++ preorder i ++ preorder m ++ preorder d)
= elem x (postorder Tern r i m d)

{po_0} elem x ([r] ++ preorder i ++ preorder m ++ preorder d) =
elem x (foldAT [] (\r recI recM recD -> recI ++ recM ++ recD ++ [r]) Tern r i m d)

{f_1} elem x ([r] ++ preorder i ++ preorder m ++ preorder d) =
elem x ((foldAT [] (\r recI recM recD -> recI ++ recM ++ recD ++ [r]) i)++
(foldAT [] (\r recI recM recD -> recI ++ recM ++ recD ++ [r]) m)++
(foldAT [] (\r recI recM recD -> recI ++ recM ++ recD ++ [r]) d)++
[r])

{po_0} elem x ([r] ++ preorder i ++ preorder m ++ preorder d) =
elem x (postorder i ++ postorder m ++ postorder d ++ [r])
```

Lema auxiliar a utilizar, demostrado al final:

$\boxed{\forall x :: a.\ \forall xs :: [a].\ \{iE\}\ \text{elem z (xs++ys) = elem z (xs) || elem z (ys)}}$

Siguiendo. Tenemos:

```
elem x ([r] ++ preorder i ++ preorder m ++ preorder d) =
elem x (postorder i ++ postorder m ++ postorder d ++ [r])

elem x ([r] ++ preorder i ++ preorder m ++ preorder d) =
elem x (postorder i ++ postorder m ++ postorder d ++ [r])

elem x ((([r] ++ preorder i) ++ preorder m) ++ preorder d) =
elem x (((postorder i ++ postorder m) ++ postorder d) ++ [r])

{iE x2} elem x (([r] ++ preorder i) ++ preorder m) || elem x preorder d =
elem x ((postorder i ++ postorder m) ++ postorder d) || elem x [r]

{iE x2} elem x ([r] ++ preorder i) || elem x preorder m || elem x preorder d =
elem x (postorder i ++ postorder m) || elem x postorder d || elem x [r]

{iE x2} elem x [r] || elem x preorder i || elem x preorder m || elem x preorder d =
elem x postorder i || elem x postorder m || elem x postorder d || elem x [r]

{e1 x2} (x == r) || elem x [] || elem x preorder i || elem x preorder m ||
elem x preorder d = elem x postorder i || elem x postorder m || elem x postorder d
|| (x == r) || elem x []
```

Evaluación de casos

Caso x = r:

```
{e1 x2} True || elem x [] || elem x preorder i || elem x preorder m ||
elem x preorder d = elem x postorder i || elem x postorder m ||
elem x postorder d || True || elem x []

True = True ✓
```

Caso x $\neq$ r:

```
Si elem x preorder i = True

elem x [r] || True || elem x preorder m || elem x preorder d =
elem x postorder i || elem x postorder m || elem x postorder d || elem x [r]

Por HI sabemos que P(i): elem x (preorder i) = elem x (postorder i). Entonces:

elem x [r] || True || elem x preorder m || elem x preorder d =
True || elem x postorder m || elem x postorder d || elem x [r]
```

True = True ✓

Equivalente para m y d.

En caso de que no este en ninguno de preorder entonces tenemos:

False || False || False || False =
elem x postorder i || elem x postorder m || elem x postorder d || False

y por HI que dice que P(i) $\wedge$ P(m) $\wedge$ P(d), entonces:

False || False || False || False = False || False || False || False

False = False ✓

Probamos entonces P(Tern r i m d) ✓

---

Demostración elem x (xs++ys) == elem x (xs) || elem x (ys) por inducción en listas. Busco demostrar:

$\forall$x :: a . $\forall$xs :: [a]. (P(xs) $\Longrightarrow$ P(x:xs))

con P(xs) = elem z (xs++ys) = elem z (xs) || elem z (ys)

Pruebo P([])
elem z ([]++ys) = elem z ([]) || elem z (ys)
{++$_0$} elem z (foldr (:) ys []) = elem z ([]) || elem z (ys)
{fr$_0$} elem z (ys) = elem z ([]) || elem z (ys)
{e$_0$} elem z (ys) = False || elem z (ys)
{||$_0$} elem z (ys) = elem z (ys) ✓

Pruebo P(x : xs)
elem z ((x:xs) ++ ys) = elem z (x:xs) || elem z (ys)
{++$_0$} elem z (foldr (:) ys (x:xs)) = elem z (x:xs) || elem z (ys)
{fr$_1$} elem z (x : (foldr (:) ys xs)) = elem z (x:xs) || elem z (ys)
elem z (x : (foldr (:) ys xs)) = elem z (x:xs) || elem z (ys)
{e$_1$} (x == z) || elem z (foldr (:) ys xs) = elem z (x:xs) || elem z (ys)
{e$_1$} (x == z) || elem z (foldr (:) ys xs) = (x == z) || elem z (xs) || elem z (ys)
{++$_0$} (x == z) || elem z (xs++ys) = (x == z) || elem z (xs) || elem z (ys)

Evalúo casos
Si x = z:

True || elem z (xs++ys) = True || elem z (xs) || elem z (ys)
True == True ✓

Si x $\neq$ z:

False || elem z (xs++ys) = False || elem z (xs) || elem z (ys)
elem z (xs++ys) = elem z (xs) || elem z (ys) ✓ (Ya que P(xs) es verdadero por HI)