# Ejercicio 9

***Enunciado:***

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

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

**Para ello, por el principio de extensionalidad de funciones basta demostrar con inducción en la estructura de Árboles ternarios que:**

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

## Inducción

```
DEFINICIONES

data AT a = Nil | Tern a (AT a) (AT a) (AT a) deriving Eq

preorder :: Procesador (AT a) a
{pre} preorder = foldAT (\rr ri rm rd -> rr : (ri ++ rm ++ rd)) []

postorder :: Procesador (AT a) a
{post} postorder = foldAT (\rr ri rm rd -> (ri ++ rm ++ rd) ++ [rr]) []

foldAT :: (a -> b -> b -> b -> b) -> b -> AT a -> b
{f0} foldAT _ z Nil = z
{f1} foldAT f z (Tern x i m d) = f x (foldAT f z i) (foldAT f z m) (foldAT f z d)

{e0} elem e []     = False
{e1} elem e (x:xs) = (e == x) || elem e xs
```

### Caso Base

$P(Nil) : \forall x :: a\ (elem\ x\ (preorder\ Nil) = elem\ x\ (postorder\ Nil))$

*Demostración:*

```
(1) elem x (preorder Nil)
(2) elem x (postorder Nil)

(1)
= elem x (foldAT (\rr ri rm rd -> rr : (ri ++ rm ++ rd)) [] Nil)     {pre}
= elem x []                                                          {f0}
= False                                                              {e0}

(2)
= elem x (foldAT (\rr ri rm rd -> (ri ++ rm ++ rd) ++ [rr]) [] Nil)  {post}
= elem x []                                                          {f0}
= False                                                              {e0}

(1) = (2)
```

### Caso Inductivo:

$(P(i) \wedge P(m) \wedge P(d) \Rightarrow P(\text{tern r i m d}))$

#### Hipótesis Inductiva :

$\forall i :: AT\ a.\ \forall m :: AT\ a.\ \forall d :: AT\ a.\ (P(i) \wedge P(m) \wedge P(d))$

**Lo que queremos demostrar entonces es lo siguiente:**

$$\forall i :: AT\ a.\ \forall m :: AT\ a.\ \forall d :: AT\ a. \forall r :: a.\ (P(i) \wedge P(m) \wedge P(d) \Rightarrow P(\text{tern r i m d}))$$

*Demostración*:

```
P(tern r i m d):
elem x (preorder (tern r i m d)) = elem x (postorder tern r i m d)

(1) elem x (preorder (tern r i m d))
(2) elem x (postorder(tern r i m d))

(1)
= elem x (foldAT (\rr ri rm rd -> rr : (ri ++ rm ++ rd)) [] (tern r i m d))                {pre}
(renombramos f = (\rr ri rm rd -> rr : (ri ++ rm ++ rd)))
= elem x (f r (foldAT f [] i) (foldAT f [] m) (foldAT f [] d))
= elem x (r : ((foldAT f [] i) ++ (foldAT f [] m) ++ (foldAT f [] d)))                     {=Beta}
= elem x (r : ((preorder i) ++ (preorder m) ++ (preorder d))                               {pre}
= x == r || elem x ((preorder i) ++ (preorder m) ++ (preorder d)) {e1}                     {e1}
= x == r || elem x (preorder i) || elem x (preorder m) || elem x (preorder d)              {Lema 1}

(2)
= elem x (foldAT (\rr ri rm rd -> (ri ++ rm ++ rd) ++ [rr]) [] (tern r i m d))             {post}
(renombre de g = (\rr ri rm rd -> (ri ++ rm ++ rd) ++ [rr])
= elem x (g r (foldAT g [] i) (foldAT g [] m) (foldAT g [] d))
= elem x ((foldAT g [] i) ++ (foldAT g [] m) ++ (foldAT g [] d) ++ [r])                    {=Beta}
= elem x ((postorder i) ++ (postorder m) ++ (postorder d) ++ [r])                          {post}
= elem x (postorder i) || elem x (postorder m) || elem x (postorder d) || elem x [r]       {Lema 1}
= elem x (postorder i) || elem x (postorder m) || elem x (postorder d) || x == r           {e1 + e0}
= elem x (preorder i) || elem x (preorder m) || elem x (preorder d) || x == r              {HI x3}
= x == r || elem x (preorder i) || elem x (preorder m) || elem x (preorder d)              {||}

(1) = (2)
```

☐ **Demostramos entonces tanto el caso base como el paso inductivo, por lo que es cierto por inducción en la estructura de Árboles Ternarios que $\forall t :: AT\ a.\ P(t)$ y, por tanto, vale la propiedad pedida:**

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

## *Lema 1:*

$$\forall xs :: [a]\ \ \forall ys :: [a]\ \ \forall e :: a\ (\text{elem e (xs} ++ \text{ys)} = \text{elem e xs} \mid\mid \text{elem e ys})$$

**Por el principio de extensionalidad de funciones, basta con probar por inducción en listas que :**

$$\forall xs :: [a]\ P(xs) : \forall ys :: [a].\ \forall e :: a.\ (\text{elem e (xs} ++ \text{ys)} = \text{elem e xs} \mid\mid \text{elem e ys})$$

### Inducción

```
DEFINICIONES

{e0} elem e []     = False
{e1} elem e (x:xs) = (e == x) || elem e xs

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

foldr :: (a -> b -> b) -> b -> [a] ->b
{f0} foldr f z []     = z
{f1} foldr f z (x:xs) = f x (foldr f z xs)
```

### Caso Base

$$P([]) : \forall ys :: [a].\ \forall e :: a.\ (\text{elem e ([]} ++ \text{ys)} = \text{elem e [] } \mid\mid \text{elem e ys})$$

*Demostración:*

```
(1) elem e ([] ++ ys)
(2) elem e [] || elem e ys

(1)
= elem x (foldr (:) ys [])                                    {++}
= elem x ys                                                   {f0}


(2)
= False || elem e ys                                          {e0}
= elem x ys                                                   {||}

(1) = (2)
```

## Caso Inductivo

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

Donde $P(x : xs) : \forall ys :: [a].\ \forall e :: a.\ (\text{elem e ((x:xs) ++ ys)} = \text{elem e (x:xs)} \mid\mid \text{elem e ys})$

### Hipótesis inductiva:

$\forall xs :: [a]\ P(xs) : \forall ys :: [a].\ \forall e :: a.\ (\text{elem e (xs ++ ys)} = \text{elem e xs} \mid\mid \text{elem e ys})$

```
elem e ((x:xs) ++ ys) = elem e (x:xs) || elem e ys
(1) elem e ((x:xs) ++ ys))
(2) elem e (x:xs) || elem e ys

(1)
= elem e (foldr (:) ys (x:xs))       {++}
= elem e (x : (foldr (:) ys xs))     {f1}
= elem e (x : (xs ++ ys))            {++}
= e == x || elem e (xs ++ ys)        {e1}
= e == x || elem e xs || elem e ys   {HI}

(2)
= e == x || elem e xs || elem e ys   {e1}

(1) = (2)
```

☐ **Demostramos entonces el caso base y el paso inductivo luego, por inducción en listas, es válido el Lema 1.**