

TP PLP

Javaneta

September 15, 2024

1 Ejercicio 9

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

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

Preorder y Postorder son dos maneras diferentes de recorrer un árbol. En este caso en puntual, recorren un árbol ternario.

La lista resultante de ordenar el árbol ternario que devuelve preorder t y postorder t son diferentes, pero lo que ambos tienen en común es que contienen los mismos elementos, es decir, $(\forall x :: a)(x \in listaResPreOrder \iff x \in listaResPostOrder)$

Tenemos un caso que podemos mencionar, donde en el ejercicio 4 del TP: $elem\ n\ (preorder\ at) = elem\ n\ (postorder\ at)$ es válido $\forall n :: a$

Por lo tanto, probemos esto utilizando los principios de extensionalidad e inducción estructural sobre árboles para concluir que esto es verdadero para cualquier árbol ternario.

Recordemos la definición del tipo AT y cuáles son los constructores del tipo correspondientes: $data\ AT\ a = Nil \mid Tern\ a\ (AT\ a)\ (AT\ a)\ (AT\ a)\ deriving\ Eq$

Luego, recordemos qué ecuaciones representan a las operaciones de elem, preorder y postorder.

$$\begin{aligned} elem &:: Eq\ a \Rightarrow a \rightarrow [a] \rightarrow Bool \\ \{E0\} \quad elem\ e\ [] &= False \\ \{E1\} \quad elem\ e\ (x:xs) &= (e==x) \quad || \quad elem\ e\ xs \end{aligned}$$

$$\begin{aligned} foldAT &:: (a \rightarrow b \rightarrow b \rightarrow b \rightarrow b) \rightarrow b \rightarrow AT\ a \rightarrow b \\ \{F0\} \quad foldAT\ _\ b\ Nil &= b \\ \{F1\} \quad foldAT\ f\ b\ (Tern\ r\ i\ c\ d) &= \\ f\ r\ (foldAT\ f\ b\ i) \quad (foldAT\ f\ b\ c) \quad (foldAT\ f\ b\ d) \end{aligned}$$

```

preorder :: AT a -> [a]
{PR0} preorder = foldAT(\r i c d -> r : (i ++ c ++ d)) []

postorder :: AT a -> [a]
{PS0} postorder = foldAT(\r i c d -> reverse (r : (d ++ c ++ i))) []

```

Por inducción estructural en t tenemos:

- $P(t) = (\forall x :: a)(\text{elem } x (\text{preorder } t) = \text{elem } x (\text{postorder } t))$

Probemos el caso base, es decir, el constructor del tipo AT a no recursivo: este caso es Nil .

Caso Base: $P(Nil) = (\forall x :: a)(\text{elem } x (\text{preorder } Nil) = \text{elem } x (\text{postorder } Nil))$
Resolvamos ambos lados por separado y deberíamos llegar a una equivalencia.

- $\text{elem } x (\text{preorder } Nil) \stackrel{PR0}{=} \text{elem } x (\text{foldAT}(\overset{\circ}{i} \text{ c d} \rightarrow r : (i ++ c ++ d)) [])$
 $Nil) \stackrel{F0}{=} \text{elem } x [] \stackrel{E0}{=} \text{False}$
- $\text{elem } x (\text{postorder } Nil) \stackrel{PS0}{=} \text{elem } x (\text{foldAT}(\overset{\circ}{i} \text{ c d} \rightarrow \text{reverse } (r : (d ++ c ++ i)))) [] Nil) \stackrel{F0}{=} \text{elem } x [] \stackrel{E0}{=} \text{False}$

Por lo tanto, el caso base es verdadero.

Recordemos que para poder hacer inducción sobre listas necesitamos predicar acerca de $(\forall xs :: [a])(\forall x :: a)(P(xs) \implies P(x : xs))$. En árboles, la inducción lo hacemos sobre cada constructor recursivo, en este caso serían 3.

Paso Inductivo:

- $(\forall i, c, d :: AT \ a)(\forall r :: a)(P(i) \wedge P(c) \wedge P(d) \implies P(\text{Tern } a \ i \ c \ d))$
 - HI: $P(i) \wedge P(c) \wedge P(d)$ donde:
 - * $P(i)$: $\text{elem } x (\text{preorder } i) = \text{elem } x (\text{postorder } i)$
 - * $P(c)$: $\text{elem } x (\text{preorder } c) = \text{elem } x (\text{postorder } c)$
 - * $P(d)$: $\text{elem } x (\text{preorder } d) = \text{elem } x (\text{postorder } d)$
 - TI: $P(\text{Tern } a \ i \ c \ d)$