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:
∀ t :: AT a . ∀ 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: $data\ AT\ a\ =\ Nil\ |\ Tern\ a\ (AT\ a)\ (AT\ a)\ (AT\ a)\ deriving\ Eq$

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

```
elem :: Eq a => a -> [a] -> Bool
{E0} elem e [] = False
{E1} elem e (x:xs) = (e==x) || elem e xs
```

```
foldAT :: (a -> b -> b -> b -> b) -> b -> AT a -> b
{F0} foldAT _ b Nil = b
{F1} foldAT f b (Tern r i c d) = f r (foldAT f b i) (foldAT f b c) (foldAT f b
```

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

Probemos el caso base, es decir, el constructor del tipo AT a no recursivo: este caso es Nil.
Caso Base: P(Nil) = (∀ x::a)(elem x (preorder Nil) = elem x (postorder Nil))
Resolvamos ambos lados por separado y deberíamos llegar a una equivalencia.

- elem x (preorder Nil) $\overset{PR0}{=}$ elem x (foldAT(i̊ c d → r : (i ++ c ++ d)) [] Nil) $\overset{F0}{=}$ elem x [] $\overset{E0}{=}$ False

- elem x (postorderNil) $\overset{PS0}{=}$ elem x (foldAT(i̊ c d -¿ reverse (r : (d ++ c ++ i))) [] Nil) $\overset{F0}{=}$ elem x [] = $\overset{E0}{=}$ False

Por lo tanto, el caso base es verdadero.
Paso Inductivo: