Inducción Estructural sobre Árboles Ternarios

1 Introducción

En este trabajo se abordará una propiedad de los árboles ternarios utilizando el principio de inducción estructural. Los árboles ternarios están definidos de la siguiente manera:

- **Definición del Generador de Árboles Ternarios (AT):**

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

  El tipo de dato `AT a` representa un árbol ternario, donde cada nodo contiene un valor de tipo `a` y tres subárboles. El constructor `Nil` representa un árbol vacío.

- **Definición de la función `foldAT`:**

  ```
  foldAT :: b -> (a -> b -> b -> b -> b) -> AT a -> b
  foldAT atNil atBranch Nil = atNil
  foldAT atNil atBranch (Tern raiz left right center) =
      atBranch raiz (rec left) (rec center) (rec right)
      where rec = foldAT atNil atBranch
  ```

  La función `foldAT` es un fold sobre el árbol ternario que nos permite procesar el árbol con base en un valor para el caso base (`atNil`) y una función cuaternaria (`atBranch`) que combina el nodo raíz y con el resultado recursivo de sus tres subárboles.

- **Preorden y postorden:**

  – `preorder :: AT a -> [a]`

    ```
    preorder = foldAT [] (\x left middle right -> x : (left ++ right ++ middle))
    ```

  – `postorder :: AT a -> [a]`

    ```
    postorder = foldAT [] (\x left middle right -> left ++ right ++ middle ++ [x])
    ```

2 Objetivo

El objetivo es demostrar la siguiente propiedad:

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

Es decir, queremos probar que para cualquier árbol ternario t y cualquier elemento x, x está en el recorrido en preorden del árbol si y solo si está en el recorrido en postorden del mismo árbol:

$$(\forall x), x \in \texttt{preorder t} \iff x \in \texttt{postorder t}$$

3 Demostración por inducción estructural

Para probar esta propiedad, aplicaremos el principio de inducción estructural sobre la estructura del árbol t. Esto implica probar:

- Caso base: la propiedad es verdadera para el árbol vacío (`Nil`).
- Paso inductivo: asumiendo que la propiedad es verdadera para los subárboles, mostrar que también es verdadera para un árbol no vacío (único generador es `Tern`).

3.1 Caso base

Consideremos el árbol vacío $t = \texttt{Nil}$. El recorrido en preorden y postorden del árbol vacío es la lista vacía:

$$\begin{aligned} \texttt{preorder Nil} &= [] \\ \texttt{postorder Nil} &= [] \end{aligned}$$

Claramente, para cualquier x, se cumple que:

$$\texttt{elem}\ x\ [] = \texttt{False}$$

Por lo tanto, la propiedad se cumple para el caso base.

3.2 Paso inductivo

Partiendo de que aquello que queremos probar es válido para los subárboles de `t`, se intenta probar que la propiedad vale para `t`.

3.2.1 Hipótesis inductiva

Se asume que la propiedad se cumple para los subárboles de

$$t = \texttt{Tern r lT mT rT}$$
$$(\texttt{Tern root leftTree middleTree rightTree })$$

Es decir, se asume que:

$$\forall x\colon a,\ \texttt{elem}\ x\ (\texttt{preorder lT}) = \texttt{elem}\ x\ (\texttt{postorder lT})$$
$$\forall x\colon a,\ \texttt{elem}\ x\ (\texttt{preorder mT}) = \texttt{elem}\ x\ (\texttt{postorder mT})$$
$$\forall x\colon a,\ \texttt{elem}\ x\ (\texttt{preorder rT}) = \texttt{elem}\ x\ (\texttt{postorder rT})$$

Estas asunciones son las hipótesis inductivas que van a ser utilizadas para probar que:

$$\begin{aligned}&\forall x\colon a,\\ &\quad \texttt{elem}\ x\ (\texttt{preorder (Tern r lT mT rT)})\\ &\quad = \texttt{elem}\ x\ (\texttt{postorder (Tern r lT mT rT)})\end{aligned}$$

3.2.2 Simplificaciones Útiles

Para la claridad del paso inductivo se consideró útil contar con definciones de `preorder` y `postorder` equivalentes que no estén definidas por `foldAT`, sino con recursión explícita. Por definición, tenemos:

```
preorder (Tern r lT mT rT) =
    foldAT [] (\ x left middle right ->
        x : (left ++ right ++ middle)) (Tern r lT mT rT )
```

Aplicando la definición de foldAT :

```
foldAT [] (\ x left middle right -> x : (left ++ right ++ middle)) (Tern lT mT rT ) =
(\ x left middle right -> x : (left ++ right ++ middle))
    raiz (rec left) (rec center) (rec right) (Tern lT mT rT )

      where rec = foldAT [] (\ x left middle right -> x : (left ++ right ++ middle))
```

Notar que la definición de `rec` es igual a la de `preorder`. Por tanto:

$\texttt{preorder (Tern r lT mT rT)} = \texttt{x : (preorder lT) ++ (preorder mT) ++ (preorder rT)}$

De la misma manera, queremos una definición de `postorder` que cumpla el esquema de recursión explícita. Aplicando la recusión explícita de foldAT en la definición de `postorder` queda:

```
foldAT [] (\ x left middle right -> (left ++ right ++ middle ++ [x]))
    (Tern r lT mT rT ) = (\ x left middle right ->
    (left ++ right ++ middle ++ [x]))
    r (rec left) (rec center) (rec right)
      (Tern r lT mT rT )

    where rec = foldAT [] (\ x left middle right -> (left ++ right ++ middle++[x]))
```

Otra vez, la definición de `rec` es igual a la de `postorder`. Por tanto:

$$\begin{aligned}\texttt{postorder (Tern r lT mT rT)} = \ & \texttt{(preorder lT)} ++ \\ & \texttt{(postorder mT)} ++ \\ & \texttt{(postorder rT)} \ ++\texttt{[r]}\end{aligned}$$

3.2.3 Inducción

Podemos ver que `x` puede estar en la raíz, en el subárbol izquierdo, en el subárbol derecho o en el subárbol central o no estar en en `t`. Por un lado tenemos que

`elem x (postorder (Tern r lT mT rT =`

`elem x ((postorder lT) ++ (postorder mT) ++ (postorder rT) ++ [r]) =`

`elem x ((postorder lT )++ (postorder mT) ++ (postorder mT))` $\vee$ `( x == r)`

Otra forma más redudundante (por propiedades básicas de las listas) de decir lo mismo es:

$$\begin{aligned}\texttt{elem x (postorder (Tern r lT mT rT))} \ = \ & \texttt{(elem x (postorder lT))} \\ & \vee\texttt{(elem x (postorder mT))} \\ & \vee\texttt{(elem x (postorder rT))} \ \vee\texttt{(x == r)}\end{aligned}$$

Por hipótesis inductiva, se puede reemplazar los {`elem x postorder`} aplicados a los subárboles de `t` por {`elem x preorder`}, pues por asumimos que son equivalentes.

$$\begin{aligned}\texttt{elem x (postorder (Tern left center right))} \ = \ & \texttt{(elem x (preorder lT))} \\ & \vee\texttt{(elem x (preorder mT))} \\ & \vee\texttt{(elem x (preorder rT))} \ \vee\texttt{(x == raiz)}\end{aligned}$$

Eso es lo mismo que decir:

`elem x (postorder Tern r lT mT rT )  = elem x (preorder Tern r lT mT rT )`

Demostrados tanto el caso base como el paso inductivo, queda demostrado que se cumple la propiedad.

3.3 Conclusión

Mediante inducción estructural queda probado que para todo árbol ternario t y todo elemento x, x pertenece al recorrido en preorden de t si y solo si pertenece al recorrido en postorden de t.