Trabajo Práctico N°1 - PLP

Demostración del ejercicio 9

Reglas y Funciones

```haskell
preorder :: Procesador (AT a) a
preorder = foldAT (\w x y z -> w : x ++ y ++ z) []      -- P1

postorder :: Procesador (AT a) a
postorder = foldAT (\w x y z -> x ++ y ++ z ++ [w]) []  -- P0

elem :: Int -> [Int] -> Bool
elem x [] = False                                       -- E0
elem x (y:ys) = if x == y then True else elem x ys      -- E1
```

Demostración

Se debe demostrar lo siguiente:

$\forall t :: ATa.\forall x :: a.(elemx(preorder\ t) = elemx(postorder\ t))$

Voy a probar esto con inducción estructural.

Caso Base

Caso Base AT:

- $atVacio = Tern\ Nil\ (Nil)\ (Nil)\ (Nil)$

Reemplazo t con $atVacio$. $elem\ x\ (preorder\ atVacio) = elem\ x\ (postorder\ atVacio)$

Como preorder y postorder devuelven una lista de elementos de un AT, si la lista está vacía el foldAT devuelve [].

- $elem\ x\ [\ ] = elem\ x\ [\ ]$

Esto por $E0$, da falso.

- $False = False$

Como False == False entonces el caso base es verdadero.

Hipótesis Inductiva

La **Hipótesis Inductiva** es:

- $elem\ x\ (preorder\ t) = elem\ x\ (postorder\ t)$

Donde t es un $Tern\ a(AT\ a)\ (AT\ a)\ (AT\ a)$

Paso Inductivo

El paso inductivo es:

$P(izq)$ & $P(med)$ & $P(der) \implies P(t)$

- $elem\ x\ (preorder\ w\ \ izq\ \ med\ \ der) = elemx(postorder\ w\ \ izq\ \ med\ \ der)$

Reemplazo preorder y postorder con $P0$ y $P1$

- $elem\ x(w : (preorder\ izq) + +(preorder\ med) + +(preorder\ der)) = elem\ x(w : (postorder\ izq) + +(postorder\ med) + +(postorder\ der))$

Hay dos casos para elem según $E1$ $(elemx(y : ys))$, voy a pasar por ambos

1. Caso en el cual $x == w$;
   - En ese caso devuelve $True$ para ambos.
   - Por $E1$ $(elem\ x(y : ys) = if\ x == y\ \ then\ \ True\ \ else\ \ elem\ x\ ys)$
   - $True = True$
2. Caso en el cual $x \neq w$:
   - Por $E1(elem\ x(y : ys) = if\ x == y\ \ then\ \ True\ \ else\ \ elem\ x\ ys)$
   - Reemplazo con el termino del else de $E1$
   - $elem\ x((preorder\ izq) + +(preorder\ med) + +(preorder\ der)) = elem\ x((postorder\ izq) + +(postorder\ med) + +(postorder\ der))$
   - Como $elem\ x$ no altera la estructura de las listas puedo decir que esta expresión es igual a la siguiente. Probado con *(1).
   - $elem\ x(preorder\ izq)\ ||\ elem\ x(preorder\ med)\ ||\ elem\ x(preorder\ der) =$
     $elem\ x$
     $(postorder\ izq)\ ||\ elem\ x(postorder\ med)\ ||\ elem\ x(postorder\ der)$
   - Y si aplico **HI**. $(elem\ x(preorder\ t) = elem\ x(postorder\ t))$ en cada uno de los preorder y postorder nos queda:
     - $elem\ x(postorder\ izq)\ ||\ elem\ x(postorder\ med)||elem\ x(postorder\ der) =$
       $elem\ x(postorder\ izq)$
       $||\ elem\ x(postorder\ med)\ ||\ elemx\ (postorder\ der)$
   - Lo cual cumple con la igualdad, probando la demostración.

Por lo tanto, queda demostrado que $\forall t :: AT\ a.\forall x :: a.(elem\ x\ (preorder\ t) = elem\ x\ (postorder\ t))$. ■

Anotaciones

Demo para paso Inductivo en *(1)

elem x (a ++ b) == elem x a || elem x b

```
elem :: Int -> [Int] -> Bool
elem x [] = False                                       -- E0
elem x (y:ys) = if x == y then True else elem x ys      -- E1
```

Caso Base de lista Vacía:

- Caso de a == [ ] :

Reemplazamos a con [ ]:

- elem x ([] ++ b) = elem x [] || elem x b

Como [ ] ++ b == b, y por E0 (elem x [ ])

- elem x b = False || elem x b

Por logica de bool (falso || x == x)

- elem x b = elem x b

La igualdad cumple, así que probamos el caso base.

Hipótesis Inductiva

La hipotesis inductiva es verdadera para una lista 'a'

- elem x (as ++ bs) = elem x (as) || elem x (bs)

Paso Inductivo

- elem x ((a:as) ++ bs) = elem x (a:as) || elem x (bs)

Usamos la definición de (++)

- elem x (a: (as ++ bs)) = elem x (a:as) || elem x (bs)

Y reemplazamos con E1 de elem

- if x == a then True else elem x (as ++ bs) = if x == a then True else elem x (as) || elem x (bs)

Con esto tenemos dos casos por el if, los casos dependen si x es igual a 'a'

– **Caso (x == a), x es igual a 'a'**

- if x == a then True else elem x (as ++ bs) = if x == a then True else elem x (as) || elem x (bs)

Si x == a retorna True

- True = True || elem x (bs)

Por propiedad de logica (True || x) == True

- True

Lo cual nos prueba este caso de la propiedad.

– **Caso (x /= a), x no es igual a 'a'**

  - if x == a then True else elem x (as ++ bs) = if x == a then True else elem x (as) || elem x (bs)

Usamos el caso de False del if.

  - elem x (as ++ bs) = elem x (as) || elem x (bs)

Aplico la hipotesis inductiva sobre el termino de la izquierda

  - elem x (as) || elem x (bs) = elem x (as) || elem x (bs)

Y como ambos terminos son iguales se prueba la propiedad de este otro lado.

Como ambos casos son verdaderos probamos que la demostracion es correcta. ■