Ejercicios de Repaso - Primer Parcial

Paradigmas de Lenguajes de Programación

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

4 de octubre de 2023

Ejercicio 1 - Razonamiento Ecuacional

Considerar las siguientes definiciones sobre listas y árboles estrictamente binarios:

```
    const :: a -> b -> a
{C} const = (\ x -> \ y -> x)

    head :: [a] -> a
{H} head (x:xs) = x

    tail :: [a] -> [a]
{T} tail (x:xs) = xs
```

```
     length :: [a] -> Int
{L0} length [] = 0
{L1} length (x:xs) = 1 + length xs

     null :: [a] -> Bool
{N0} null [] = True
{N1} null (x:xs) = False
```

```
      (==) :: Eq a => [a] -> [a] -> Bool
{==0} [] == [] = True
{==1} [] == (_:_) = False
{==2} (_:_) == [] = False
{==3} (x:xs) == (y:ys) = (x == y) && (xs == ys)
```

Ejercicio 1 - Razonamiento Ecuacional

```
data AEB a = Hoja a | Bin (AEB a) a (AEB a)

     altura :: AEB a -> Int
{A0} altura (Hoja x) = 1
{A1} altura (Bin i r d) = 1 + max (altura i) (altura d)

     esPreRama :: Eq a => AEB a -> [a] -> Bool
{E0} esPreRama (Hoja x) = \xs -> null xs || (xs == [x])
{E1} esPreRama (Bin i r d) = \xs -> null xs ||
        (r == head xs && (esPreRama i (tail xs) || esPreRama d (tail xs)))
```

Asumiendo `Eq a`, demostrar la siguiente propiedad:

$\forall$ `t::AEB a` . $\forall$ `xs::[a]` . esPreRama t xs $\Rightarrow$ length xs $\leq$ altura t

Se consideran demostradas todas las propiedades conocidas sobre enteros y booleanos, así como también que $\forall$ `t::AEB a` . altura t ≥ 0.

Ejercicio 2 - Cálculo Lambda

Considerar el Cálculo Lambda tipado extendido con listas. Se desea extender el cálculo con el esquema de recursión estructural para listas `foldr` y con el término `from`, que permite construir listas a partir de un primer elemento, una función generadora y una condición de corte.

El conjunto de tipos no se modifica. El conjunto de términos se extiende de la siguiente manera:

$$M ::= \dots \mid \text{foldr } \ M \ \text{ base} \hookrightarrow M; \text{rec}(h, r) \hookrightarrow M \mid \text{from } M \text{ until}_x \ M \text{ by } M$$

- $\text{foldr } \ M \ \text{ base} \hookrightarrow N; \text{rec}(h, r) \hookrightarrow O$ es el operador de recursión estructural. Los nombres de variables indicados entre paréntesis (h y r en este caso) son variables que pueden aparecer libres en O y deberán ser ligadas con la cabeza y el resultado de la recursión respectivamente.
- Una expresión de la forma from M_1 until$_x$ M_2 by M_3, donde M_1 es un elemento cualquiera, M_2 es una expresión booleana que puede tener libre la variable x, y M_3 una función que se aplicará a cada elemento para generar el siguiente, reducirá a una lista cuyo primer elemento - de haberlo - es el valor de M_1, y cada elemento subsiguiente se obtiene aplicando M_3 al elemento anterior hasta que se satisfaga la condición M_2 tomando como x al elemento actual. Si M_2 es verdadera al tomar M_1 como x, entonces la lista resultante será vacía.

Ejercicio 2 - Cálculo Lambda

Ejemplos:

- foldr $\underline{1} :: \underline{2} :: \underline{3} :: (\lambda x\colon [\mathrm{Nat}].\ x)\ [\ ]_{\mathrm{Nat}}$ base $\hookrightarrow \underline{0}$; rec$(r, h) \hookrightarrow h + r \twoheadrightarrow \underline{6}$
- from 0 until$_x$ if isZero(x) then False else Not isZero(pred(x)) by $\lambda n\colon$ Nat. succ$(n) \twoheadrightarrow 0 :: \underline{1} :: [\ ]_{\mathrm{Nat}}$
- from False until$_y$ y by $\lambda b\colon$ Bool.Not $b \twoheadrightarrow$ False $:: [\ ]_{\mathrm{Bool}}$
- from False until$_z$ True by $\lambda b\colon$ Bool.$b \twoheadrightarrow [\ ]_{\mathrm{Bool}}$

Se asume el Cálculo Lambda extendido con la suma de naturales y el término `Not` implementado como macro.

Ejercicio 2 - Cálculo Lambda

a) Dar las regla de tipado para soportar los nuevos términos.

b) Describir el conjunto de valores y dar las reglas de reducción en un paso para los nuevos términos.

c) Mostrar paso a paso cómo reduce la expresión:
from 0 until$_x$ if isZero(x) then False else True by $\lambda n\colon$ Nat. succ(n)

d) Definir como **macro** la función sucesiónAritmética, que dados tres naturales - el primer elemento, el incremento y la cota superior - genera la sucesión aritmética correspondiente. Por ejemplo:
sucesiónAritmética $\underline{1}\ \underline{4}\ \underline{10} \twoheadrightarrow \underline{1} :: \underline{5} :: \underline{9} :: [\ ]_{\mathrm{Nat}}$
sucesiónAritmética $\underline{2}\ \underline{3}\ \underline{5} \twoheadrightarrow \underline{2} :: \underline{5} :: [\ ]_{\mathrm{Nat}}$
Suponer definidos los operadores $<$, $>$, $=$, $\leq$, $\geq$ y $+$ para el tipo Nat.