Tarea 1 - Evaluador de Chi en Haskell

Teoría de la Computación
Universidad ORT Uruguay

Setiembre 2023

El objetivo de esta tarea es codificar[1] en Haskell el *lenguaje* χ estudiado en el curso como modelo funcional de computabilidad. Ello incluye:

- sintaxis abstracta, y
- reglas de reducción y evaluación,

tales como han sido descriptas en la especificación del lenguaje publicada.

Se pide, concretamente:

1. Declarar un tipo inductivo (`data`) apropiado para representar las expresiones (sintaxis abstracta) de χ.
2. Definir el tipo de las `sustituciones`, así como el efecto de ellas sobre expresiones χ[2].
3. Definir la función (parcial[3]) de `reducción`.
4. Definir la función de `evaluación` de expresiones.
5. Codificar en χ embebido en Haskell las funciones:
   - `and`: la conjunción booleana.
   - `duplicar`: que dado un natural n, retorna el doble n.
   - `unir`: que dadas dos listas l_1 y l_2, retorna l_1 seguido de l_2 (lo que conocemos en haskell como l_1 `++` l_2).
   - `ramaI`: dado un árbol binario, con información en los nodos, y hojas sin información, retorna una lista con todos los elementos de la rama izquierda.

[1] Otro término técnico utilizado es *embeber*. En inglés se usan *to encode* y *to embed*.
[2] Esto requiere definir o usar versiones predefinidas de las operaciones *lookup* y *delete*.
[3] Cuando indicamos *parcial*, nos referimos a que no actúa sobre valores y además falla en los casos así indicados en la especificación.

Probar la función `unir` con una lista que contenga al cero y uno, y otra que tenga al dos y al tres.

Luego probar la función `ramaI` con un árbol que tenga al menos 3 niveles en la rama izquierda.

Adicionalmente se puede aprovechar para explorar algunas condiciones raras, como el uso de variables no declaradas, listas de parámetros y argumentos que no tengan el mismo largo, incluso intentar ejecutar la evaluación de funciones sobre argumentos que no tendrían el tipo esperado por la función.
Ejemplos:

- `and True True False`.
- λ`x -> case y of {True -> False; False -> True}`, notar que la `x` y la `y` no coinciden, que ocurriría al aplicarle `True` o `False`.
- `unir (S(S O)) True`, notar que la función `unir` recibe 2 listas, que ocurre si le paso un natural y un booleano, o incluso, una lista y un booleano, como en el ejemplo siguiente.
- `unir [1,3,0] False`.