Catalina Gonzalo Juarros

Hola,

Corregí su TP. Si bien las funciones hacen lo que se pide y los módulos **Util** e **Histograma** están muy bien, la implementación del módulo **Expr** indica que falta reforzar el paradigma funcional, y hay algunas cosas para corregir en la demostración, por lo que van a tener que reentregar.

Cosas que sí o sí tienen que corregir para la reentrega:

- Hay errores conceptuales serios en **recrExpr**.
    - Los casos **Const** y **Rango** sólo deberían tomar los elementos de tipo **Float**, **no la expresión entera**. Dado que la expresión puede reconstruirse trivialmente a partir de los **Floats**, es redundante tomarla como parámetro.
    - Los casos recursivos deben tomar los resultados recursivos y **las subestructuras**, no la expresión entera.
- **mostrar** arrastra algunos problemas de la implementación de **recrExpr** y no sigue las buenas prácticas del paradigma. Como consecuencia de esto, es difícil entender lo que hace el código.
    - Pueden escribir código mucho más claro y sin patrones repetidos con la signatura correcta en **recrExpr**, ya que esto les permite usar **constructor** para saber cómo fue construida una expresión, sin **_cuandoIzq** y **_cuandoDer**.
    - **_operandos** hace exactamente lo mismo en todos los casos, y podría abstraerse en un **where**. De todas

formas, con los parámetros correctos, no debería ser necesario tener una función como esta.
    - **yo no es un buen nombre para una variable**. No estamos trabajando con objetos. No hay un receptor de mensajes, sino un parámetro que se le pasa a una función.
    - Las funciones **_cuando**, **_cuandoIzq** y **_cuandoDer** parecen estar pensadas como métodos de un objeto, y el código resultante en Haskell es más complicado de lo que sería si lo pensaran directamente desde el paradigma funcional. Tampoco son buenos nombres, pero esto también parece ser una consecuencia de estar pensando en el paradigma equivocado. Si se replantean **mostrar** desde el paradigma funcional, también van a solucionar este problema.
- En la demostración, el **∀e :: Expr** no debería estar en el predicado unario; justamente el predicado unario P(e) es algo que vale para cada elemento en particular, y lo que están demostrando es ∀e :: Expr . P(e). Más adelante, en el punto (b), sí dicen correctamente que por inducción van a probar esto último. Si les ayuda a tenerlo más claro, piensen que la idea de la inducción es demostrar que el predicado unario vale para todos los elementos.
- No hace falta plantear las ecuaciones {eCONST}, {eRANGO}, etc. Pueden simplemente escribir cada caso de la inducción con el constructor correspondiente y desarrollar a partir de ahí.

Detalles que no es estrictamente necesario corregir, pero igualmente está bueno tener en cuenta:

- Revisen las tildes y las mayúsculas en la demostración y en algunos comentarios de Expr (algunas mayúsculas están de más).
- En histograma, pueden hacer una eta-reducción del parámetro valores.
- En eval, pueden hacer una eta-reducción del parámetro expr.

Tienen tiempo hasta el 21/10 para reentregar. No olviden consultar por cualquier duda que tengan.

Saludos,

Catalina