Verificación formal de arboles rojinegros en Haskell con Coq

David Felipe Hernández Chiapa

Facultad de Ciencias
Universidad Nacional Autónoma de México

29 de noviembre de 2018

Verificación formal en Haskell

Haskell es un lenguaje con una base muy grande de desarrolladores que constantemente están generando mas programas escritos en el.
Una de las cosas que se dice de Haskell es que la verificación de su código es bastante sencilla.

Verificación formal en Haskell

¿Pero que tan cierto y escalable es esto?

Verificación formal en Coq.

A diferencia de Haskell, Coq es un asistente de pruebas, con el cual tu puedes escribir un programa y a la par crear la especificación formal.

Verificación formal en Coq.

Las diferencias mas grandes entre la escritura de programas entre Haskell y Coq, es que Coq solo acepta funciones totales y que estas terminen e todos sus casos.

Problema.

Nos gustaría una manera de traducir módulos de Haskell con funciones totales a Coq para poder verificarlas formalmente de una manera mas sencilla, ordenada y escalable.

hs-to-coq

Es una herramienta en desarrollo por un equipo de la Universidad de Pensilvania.
En esta herramienta ya existen bibliotecas de Haskell traducidas a Coq y también te da la facilidad de traducir tus propios programas de Haskell.

hs-to-coq

Esta herramienta es creada para facilitar la verificación, siguiendo los siguientes pasos:

1. Escribir un modulo de Haskell, digamos un modulo de Arboles Rojinegros.
2. Probar ese código en Haskell, generar ejemplos.
3. Utilizar hs-to-coq para traducir el codigo a Coq.
4. ¡A verificar!

hs-to-coq

Esto simplifica mucho la verificación en varios frentes:

- La traducción no se hace a mano.
- La cooperación en un equipo de trabajo se hace mas sencilla.

Ejemplos

Código de balance de Arboles Rojinegros en Haskell.

```
balance :: RB a → a → RB a → RB a
balance (T R a x b) y (T R c z d) = T R (T B a x b) y (T B c z d)
balance (T R (T R a x b) y c) z d = T R (T B a x b) y (T B c z d)
balance (T R a x (T R b y c)) z d = T R (T B a x b) y (T B c z d)
balance a x (T R b y (T R c z d)) = T R (T B a x b) y (T B c z d)
balance a x (T R (T R b y c) z d) = T R (T B a x b) y (T B c z d)
balance a x b = T B a x b
```

Ejemplos

Código de balance de Arboles Rojinegros en Coq, traducido con hs-to-coq.

```
Definition balance {a} : RB a → a → RB a → RB a :=
 fun arg_0__ arg_1__ arg_2__ ⇒
   match arg_0__, arg_1__, arg_2__ with
   | T R a x b, y, T R c z d ⇒ T R (T B a x b) y (T B c z d)
   | T R (T R a x b) y c, z, d ⇒ T R (T B a x b) y (T B c z d)
   | T R a x (T R b y c), z, d ⇒ T R (T B a x b) y (T B c z d)
   | a, x, T R b y (T R c z d) ⇒ T R (T B a x b) y (T B c z d)
   | a, x, T R (T R b y c) z d ⇒ T R (T B a x b) y (T B c z d)
   | a, x, b ⇒ T B a x b
 end.
```

Ejemplos

Código de inserción de Arboles Rojinegros en Haskell.

```
ins :: Ord a ⇒ a → RB a → RB a
ins x E = T R E x E
ins x s@(T B a y b)
  | x<y = balance (ins x a) y b
  | x>y = balance a y (ins x b)
  | otherwise = s
ins x s@(T R a y b)
  | x<y = T R (ins x a) y b
  | x>y = T R a y (ins x b)
  | otherwise = s
```

Ejemplos

Código de inserción de Arboles Rojinegros en Coq, traducido con hs-to-coq.

```
Definition ins {a} `{GHC.Base.Ord a} : a → RB a → RB a :=
  fix ins arg_0__ arg_1__
        := match arg_0__, arg_1__ with
           | x, E ⇒ T R E x E
           | x, (T B a y b as s) ⇒
             if x GHC.Base.< y : bool then balance (ins x a) y b else
             if x GHC.Base.> y : bool then balance a y (ins x b) else
             s
           | x, (T R a y b as s) ⇒
             if x GHC.Base.< y : bool then T R (ins x a) y b else
             if x GHC.Base.> y : bool then T R a y (ins x b) else
             s
end.
```

Observaciones

Este es un trabajo en curso, se busca poder traducir y verificar las operaciones de los arboles rojinegros.