Abordaje Funcional a EDSLs

Alberto Pardo (pardo@fing.edu.uy)
Marcos Viera (mviera@fing.edu.uy)

ECI 2024

Se desea implementar un pequeño EDSL para representar proposiciones basadas en comparaciones entre números naturales, el cual tiene las siguientes construcciones:

- `val`: valor natural
- `eq`: igualdad entre naturales
- `lt`: relación de menor entre naturales
- `not`: negación
- `and`: conjunción
- `or`: disyunción

Asociado al EDSL se tiene un sistema de tipos que está formado por las reglas que se presentan a continación. Como es habitual, el juicio $\vdash p : \tau$ significa que la expresión p tiene tipo τ. El lenguaje maneja dos tipos (*integer* y *boolean*).

$$\frac{}{\vdash \texttt{val}\ \ n : \mathit{integer}} \qquad \frac{\vdash p_1 : \mathit{integer} \qquad \vdash p_2 : \mathit{integer}}{\vdash \texttt{eq}\ \ p_1\ p_2 : \mathit{boolean}} \qquad \frac{\vdash p_1 : \mathit{integer} \qquad \vdash p_2 : \mathit{integer}}{\vdash \texttt{lt}\ \ p_1\ p_2 : \mathit{boolean}}$$

$$\frac{\vdash p : \mathit{boolean}}{\vdash \texttt{not}\ \ p : \mathit{boolean}} \qquad \frac{\vdash p_1 : \mathit{boolean} \qquad \vdash p_2 : \mathit{boolean}}{\vdash \texttt{and}\ \ p_1\ p_2 : \mathit{boolean}} \qquad \frac{\vdash p_1 : \mathit{boolean} \qquad \vdash p_2 : \mathit{boolean}}{\vdash \texttt{or}\ \ p_1\ p_2 : \mathit{boolean}}$$

Por ejemplo, las siguientes son expresiones bien tipadas del EDSL:

$\vdash$ `var` 4 : *integer*
$\vdash$ `lt` (`var` 4) (`var` 5) : *boolean*
$\vdash$ `or` (`not` (`lt` (`var` 4) (`var` 3)) (`eq` (`var` 4) (`var` 3))) : *boolean*

En cambio, las siguientes no lo son:

`or` (`var` 4) (`var` 5)
`not` (`var` 4)
`eq` (`lt` (`var` 4) (`var` 3))(`var` 5)

Considerando que la interpretación estándar de las proposiciones es su valor de verdad o el propio natural n en el caso de expresiones de la forma `val` n, se pide:

1. Implementar el EDSL como un **shallow embedding** bien tipado en Haskell siguiendo el enfoque *tagless-final*. Interprete los tipos *integer* y *boolean* como tipos de Haskell.

2. Implementar el EDSL como un **deep embedding** bien tipado en Haskell utilizando GADTs. Definir la función *eval* que evalúa una expresión bien tipada de tipo t y retorna un valor de ese tipo.

3. Dada la siguiente gramática que describe una sintaxis concreta para el lenguaje:

   $prop$ ::= $term$ `"\/"` $prop$ | $term$
   $term$ ::= $factor$ `"/\"` $term$ | $factor$
   $factor$ ::= `'~'` $prop$ | `'('` $prop$ `')'` | `'('` $prop$ `'='` $prop$ `')'` | `'('` $prop$ `'<'` $prop$ `')'` | $\mathbb{N}$

   (a) Definir una nueva interpretación para 1. y 2. que retorne un *String* que represente el programa en esa sintaxis.

   (b) Escribir un parser del lenguaje utilizando los combinadores vistos en el curso (puede optar por usar los combinadores aplicativos o monádicos).

4. (OPCIONAL) Considere la siguiente extensión al lenguaje, en la que se agrega:

   - `var`: variable proposicional

   con la siguiente regla de tipado:

   $$\frac{}{\vdash \texttt{var}\ x : \mathit{boolean}}$$

   Ahora la interpretación depende de un *ambiente de variables* en el que se le asocian valores de verdad a las variables.

   (a) Extienda el EDSL definido en 1.

   (b) Extienda el EDSL definido en 2.

   (c) Extienda el parser definido en 3.b.

Sobre la entrega:

- Enviar un archivo `NombreApellido.zip` conteniendo los archivos .hs con la implementación de sus soluciones a `pardo@fing.edu.uy` y `mviera@fing.edu.uy`. Incluya sus datos (nombre, apellido, universidad, email) en todos los archivos que entregue.

- La prueba consiste de las partes 1 a 3. La parte 4. es simplemente opcional para quienes deseen extender el EDSL con variables proposicionales. No cuenta en la nota final.