Lenguajes y Compiladores. Práctico 9 del 26/05/2020

Objetivos: Comprender las extensiones del cálculo lambda con constantes y operadores. Lograr definir mécanismos de evaluación alternativos. Identificar el origen de la diferencia en los órdenes de evaluación.

1. Evalue de modo eager y normal las expresiones **True** $\vee$ 0, **True** $\vee$ $\Delta\Delta$, donde $\Delta = (\lambda x.\, xx)$.
2. Considere la expresión **let** f $\equiv \lambda$x.**True in** f (**True**$+0$). Recuerde que **let** f $\equiv e$ **in** $e' \doteq (\lambda \mathsf{f}.e')e$
   a) Explique, sin hacer ninguna evaluación, si ese término tiene o no forma canónica en evaluación eager y en evaluación normal.
   b) Construya el árbol de la evaluación para cada uno de esos órdenes.
3. Extender la semántica de la evaluación para describir el tratamiento de errores. Para esto incorpore las expresiones **error** y **typeerror** como resultados posibles de una evaluación, al mismo nivel que las formas canónicas. Por ejemplo se deberán agregar (entre otras) la reglas:

$$\frac{e \Rightarrow \lfloor i \rfloor \qquad e' \Rightarrow \lfloor 0 \rfloor}{e \div e' \Rightarrow \textbf{error}} \qquad \frac{e \Rightarrow \lfloor b \rfloor \qquad e' \Rightarrow z'}{e \vee e' \Rightarrow \textbf{typeerror}}\,(z' \notin \langle \mathit{boolcnf} \rangle)$$

4. Analice qué regimen de evaluación de subexpresiones ha adoptado en la semántica del ejercicio anterior. Esto es, las subexpresiones (por ejemplo en $e + e'$) se evalúan antes de chequear que los tipos sean correctos, o se obtiene **typeerror** frente a una inconsistencia de tipos, aunque no se hayan terminado de evaluar todas las subexpresiones?
   De reglas que representen la opción no considerada en el ejercicio anterior.
5. Evalue de modo eager y normal las expresiones $\langle \textbf{True} + 0, \Delta\Delta \rangle$ y $\langle \Delta\Delta, \textbf{True} + 0 \rangle$.
6. Para el lenguaje aplicativo normal, reescribir utilizando patrones y **rec**, el término

$$\begin{array}{lrcl} \textbf{letrec} & \mathit{par} & \equiv & \lambda x.\textbf{if } x = 0 \textbf{ then true else } \mathit{impar}(x-1) \\ & \mathit{impar} & \equiv & \lambda x.\textbf{if } x = 0 \textbf{ then false else } \mathit{par}(x-1) \\ \textbf{in } e & & & \end{array}$$

7. De una expresión e tal que esta tenga forma canónica bajo orden normal y que también la tengan las siguientes (infinitas) expresiones: $e.1$, $(e.2).1$, $((e.2).2).1$, etc.
8. Suponga que e es una expresión cerrada. Considere las siguientes expresiones:

$$\begin{array}{ll} \textbf{letrec } \mathsf{f} \equiv \lambda x.\ \textbf{if } e \textbf{ then } 1 \textbf{ else } \mathsf{f}\ x & \textbf{in } \mathsf{f}\ 0 \\ \textbf{letrec } \mathsf{f} \equiv \lambda x.\ \textbf{if } e \textbf{ then True else } \mathsf{f}\ x & \textbf{in } \mathsf{f}\ 0 + 1 \end{array}$$

   Evaluar del modo eager y normal estos programas, considerando por separado los casos $e \Rightarrow$ **true** y $e \Rightarrow$ **false**.
9. Decida si la siguiente afirmación *Mmm*? es cierta o no y justifique su respuesta: "Si $e \Rightarrow_E z$, entonces toda subexpresión e' de e tiene forma canónica".