

Definiciones

miércoles, 29 de mayo de 2024 11:54

Las cosas con \Box_N son solo para normal y las con \Box_E solo para eager

Lenguaje aplicativo:

$$\begin{aligned}\langle \text{exp} \rangle &= \langle \text{var} \rangle \mid \langle \text{exp} \rangle \langle \text{exp} \rangle \mid \lambda \langle \text{var} \rangle . \langle \text{exp} \rangle \\&\mid \langle \text{natconst} \rangle \mid \langle \text{boolconst} \rangle \\&\mid - \langle \text{exp} \rangle \mid \langle \text{exp} \rangle + \langle \text{exp} \rangle \mid \langle \text{exp} \rangle * \langle \text{exp} \rangle \mid \langle \text{exp} \rangle - \langle \text{exp} \rangle \\&\mid \langle \text{exp} \rangle \% \langle \text{exp} \rangle \mid \langle \text{exp} \rangle / \langle \text{exp} \rangle \\&\mid \langle \text{exp} \rangle \leq \langle \text{exp} \rangle \mid \langle \text{exp} \rangle < \langle \text{exp} \rangle \mid \langle \text{exp} \rangle = \langle \text{exp} \rangle \\&\mid \langle \text{exp} \rangle \neq \langle \text{exp} \rangle \mid \langle \text{exp} \rangle \geq \langle \text{exp} \rangle \mid \langle \text{exp} \rangle > \langle \text{exp} \rangle \\&\mid \neg \langle \text{exp} \rangle \mid \langle \text{exp} \rangle \wedge \langle \text{exp} \rangle \mid \langle \text{exp} \rangle \vee \langle \text{exp} \rangle \\&\mid \text{if } \langle \text{exp} \rangle \text{ then } \langle \text{exp} \rangle \text{ else } \langle \text{exp} \rangle\end{aligned}$$

$$\langle \text{natconst} \rangle = 0 \mid 1 \mid 2 \mid \dots$$

$$\langle \text{boolconst} \rangle = \text{true} \mid \text{false}$$

Formas canónicas:

$$\langle \text{cnf} \rangle = \langle \text{intcnf} \rangle \mid \langle \text{boolcnf} \rangle \mid \langle \text{funcnf} \rangle$$

$$\langle \text{intcnf} \rangle = \dots \mid -2 \mid -1 \mid 0 \mid 1 \mid 2 \mid \dots$$

$$\langle \text{boolcnf} \rangle = \langle \text{boolconst} \rangle$$

$$\langle \text{funcnf} \rangle = \lambda \langle \text{var} \rangle . \langle \text{exp} \rangle$$

Con $z \in \langle \text{cnf} \rangle, c \in \mathbb{B}, i \in \mathbb{Z}$

Definición de \Rightarrow :

$$z \Rightarrow z$$

Si $e_0 \Rightarrow_E \lambda v. e'_0, e_1 \Rightarrow_E z \text{ y } e'_0 / [v : z] \Rightarrow_E z'$ entonces $e_0 e_1 \Rightarrow_E z'$

Si $e \Rightarrow [i]$ entonces $\neg e \Rightarrow [-i]$

Si $\oplus \in \{+, -, *, <, \leq, \neq, >, \geq\}$, $e_0 \Rightarrow [i_0] \text{ y } e_1 \Rightarrow [i_1]$ entonces $e_0 \oplus e_1 \Rightarrow [i_0 \oplus i_1]$

Si $\oplus \in \{/, \% \}$, $e_0 \Rightarrow [i_0], e_1 \Rightarrow [i_1] \text{ y } i_1 \neq 0$ entonces $e_0 \oplus e_1 \Rightarrow [i_0 \oplus i_1]$

Si $e \Rightarrow [b]$ entonces $\neg e \Rightarrow [\neg b]$

Si $\oplus \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, $e_0 \Rightarrow_E [b_0] \text{ y } e_1 \Rightarrow_E [b_1]$ entonces $e_0 \oplus e_1 \Rightarrow_E [b_0 \oplus b_1]$

Si $e \Rightarrow \text{true}$ y $e_0 \Rightarrow z$ entonces if e then e_0 else $e_1 \Rightarrow z$

Si $e \Rightarrow \text{false}$ y $e_1 \Rightarrow z$ entonces if e then e_0 else $e_1 \Rightarrow z$

Si $e_0 \Rightarrow_N \lambda v. e'_0$ y $e'_0/[v : e_1] \Rightarrow_N z$ entonces $e_0 e_1 \Rightarrow_N z$

Si $e_0 \Rightarrow_N \text{true}$ entonces $e_0 \vee e_1 \Rightarrow_N \text{true}$

Si $e_0 \Rightarrow_N \text{false}$ y $e_1 \Rightarrow_N z$ entonces $e_0 \vee e_1 \Rightarrow_N z$

Si $e_0 \Rightarrow_N \text{false}$ y $e_1 \Rightarrow_N b$ entonces $e_0 \vee e_1 \Rightarrow_N b$

Tuplas:

$\langle \text{exp} \rangle = \dots \mid \langle \langle \text{exp} \rangle, \dots, \langle \text{exp} \rangle \rangle \mid \langle \text{exp} \rangle. \langle \text{natconst} \rangle$

$\langle \text{cnf} \rangle = \dots \mid \langle \text{tuplecnf} \rangle$

$\langle \text{tuplecnf} \rangle_E = \langle \langle \text{cnf} \rangle, \dots, \langle \text{cnf} \rangle \rangle$

$\langle \text{tuplecnf} \rangle_N = \langle \langle \text{exp} \rangle, \dots, \langle \text{exp} \rangle \rangle$

Si $\forall i < n . e_i \Rightarrow_E z_i$ entonces $\langle e_0, \dots, e_{n-1} \rangle \Rightarrow_E \langle z_0, \dots, z_{n-1} \rangle$

Si $\forall i < n . e_i \Rightarrow_E z_i$ y $k < n$ entonces $\langle e_0, \dots, e_{n-1} \rangle. k \Rightarrow_E z_k$

Si $e \Rightarrow_N \langle e_0, \dots, e_{n-1} \rangle$, $k < n$ y $e_k \Rightarrow_N z$ entonces $e \Rightarrow z$

$\langle \text{exp} \rangle_E = \dots \mid \text{letrec } \langle \text{var} \rangle \equiv \lambda \langle \text{var} \rangle . \langle \text{exp} \rangle \text{ in } \langle \text{exp} \rangle$

$\langle \text{exp} \rangle_N = \dots \mid \text{rec } \langle \text{exp} \rangle$

Si $e' / [f : \lambda u . \text{letrec } f \equiv \lambda u . e \text{ in } e] \Rightarrow_E z$ y $f \neq u$

entonces $\text{letrec } f \equiv \lambda u . e \text{ in } e' \Rightarrow_E z$

Si $e (\text{rec } e) \Rightarrow_N z$ entonces $\text{rec } e \Rightarrow_N z$

Azúcar sintáctico:

$\text{let } p_0 \equiv e_0, \dots, p_{n-1} \equiv e_{n-1} \text{ in } e \stackrel{\text{def}}{=} (\lambda p_0. \dots. \lambda p_{n-1}. e) e_0 \dots e_n$

1)

viernes, 17 de mayo de 2024 10:53

1. Evalúe de modo eager y normal las expresiones $\mathbf{True} \vee 0$, $\mathbf{True} \vee \Delta\Delta$, donde $\Delta = (\lambda x. xx)$.

Eager:

$\mathbf{True} \vee 0$

$\mathbf{True} \Rightarrow_E \mathbf{True}$

$0 \Rightarrow_E 0$

No evalúa a nada

Normal:

$\mathbf{True} \vee 0$

$\mathbf{True} \Rightarrow_N \mathbf{True}$

$\Rightarrow_N \mathbf{True}$

Eager:

$\mathbf{True} \vee \Delta\Delta$

$\mathbf{True} \Rightarrow_E \mathbf{True}$

$\Delta\Delta \Rightarrow_E \dots$

Normal:

$\mathbf{True} \vee \Delta\Delta$

$\mathbf{True} \Rightarrow_E \mathbf{True}$

$\Rightarrow_E \mathbf{True}$

2)

jueves, 30 de mayo de 2024 17:01

2. Considere la expresión $\text{let } f \equiv \lambda x. \text{True} \text{ in } f(\text{True} + 0)$. Recuerde que $\text{let } f \equiv e \text{ in } e' \doteq (\lambda f. e')e$
- Explique, sin hacer ninguna evaluación, si ese término tiene o no forma canónica en evaluación eager y en evaluación normal.
 - Construya el árbol de la evaluación para cada uno de esos órdenes.

$$\lambda f . f(\text{True} + 0) \lambda x . \text{True}$$

Normal:

$$\lambda f . f(\text{True} + 0) \lambda x . \text{True}$$

$$\lambda f . f(\text{True} + 0) \Rightarrow_N \lambda f . f(\text{True} + 0)$$

$$(\lambda x . \text{True})(\text{True} + 0)$$

$$\lambda x . \text{True} \Rightarrow_N \lambda x . \text{True}$$

$$\text{True} \Rightarrow_N \text{True}$$

$$\Rightarrow_N \text{True}$$

$$\Rightarrow_N \text{True}$$

Eager:

$$\lambda f . f(\text{True} + 0) \lambda x . \text{True}$$

$$\lambda f . f(\text{True} + 0) \Rightarrow_E \lambda f . f(\text{True} + 0)$$

$$\lambda x . \text{True} \Rightarrow_E \lambda x . \text{True}$$

$$(\lambda x . \text{True})(\text{True} + 0)$$

$$\lambda x . \text{True} \Rightarrow_E \lambda x . \text{True}$$

$$\text{True} + 0$$

$$\text{True} \Rightarrow_E \text{True}$$

$$0 \Rightarrow_E 0$$

No se puede seguir

3)

jueves, 30 de mayo de 2024 17:02

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:

$$\frac{e \Rightarrow [i] \quad e' \Rightarrow [0]}{e \div e' \Rightarrow \text{error}} \quad \frac{e \Rightarrow [b] \quad e' \Rightarrow z' \quad (z' \notin \langle \text{boolcnf} \rangle)}{e \vee e' \Rightarrow \text{typeerror}}$$

$z \Rightarrow_E z$

Si $e_0 \Rightarrow_E \lambda v. e'_0$, $e_1 \Rightarrow_E z$ y $e'_0 / [v : z] \Rightarrow_E z'$ entonces $e_0 e_1 \Rightarrow_E z'$

Si $e \Rightarrow_E [i]$ entonces $\neg e \Rightarrow [-i]$

Si $\oplus \in \{+, -, *, /\}$, $e_0 \Rightarrow_E [i_0]$ y $e_1 \Rightarrow_E [i_1]$ entonces $e_0 \oplus e_1 \Rightarrow_E [i_0 \oplus i_1]$

Si $\oplus \in \{/, \%$ }, $e_0 \Rightarrow_E [i_0]$, $e_1 \Rightarrow_E [i_1]$ y $i_1 \neq 0$ entonces $e_0 \oplus e_1 \Rightarrow_E [i_0 \oplus i_1]$

Si $e \Rightarrow_E [b]$ entonces $\neg e \Rightarrow [\neg b]$

Si $\oplus \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, $e_0 \Rightarrow_E [b_0]$ y $e_1 \Rightarrow_E [b_1]$ entonces $e_0 \oplus e_1 \Rightarrow_E [b_0 \oplus b_1]$

Si $e \Rightarrow_E \text{true}$ y $e_0 \Rightarrow_E z$ entonces if e then e_0 else $e_1 \Rightarrow_E z$

Si $e \Rightarrow_E \text{false}$ y $e_1 \Rightarrow_E z$ entonces if e then e_0 else $e_1 \Rightarrow_E z$

Si $\oplus \in \{/, \%$ }, $e_0 \Rightarrow_E [i]$, $e_1 \Rightarrow_E [0]$ entonces $e_0 \oplus e_1 \Rightarrow_E \text{error}$

Si $e_0 \Rightarrow_E z$ y $z \notin \langle \text{funcnf} \rangle$ entonces $e_0 e_1 \Rightarrow_E \text{typeerror}$

Si $e \Rightarrow_E z$ y $z \notin \langle \text{intcnf} \rangle$ entonces $\neg e \Rightarrow \text{typeerror}$

Si $\oplus \in \{+, -, *, /, \%$ }, $e_0 \Rightarrow_E z$ y $z \notin \langle \text{intcnf} \rangle$ entonces $e_0 \oplus e_1 \Rightarrow_E \text{typeerror}$

Si $\oplus \in \{+, -, *, /, \%$ }, $e_0 \Rightarrow_E [i]$, $e_1 \Rightarrow_E z$ y $z \notin \langle \text{intcnf} \rangle$ entonces $e_0 \oplus e_1 \Rightarrow_E \text{typeerror}$

Si $e \Rightarrow_E z$ y $z \notin \langle \text{boolcnf} \rangle$ entonces $\neg e \Rightarrow \text{typeerror}$

Si $\oplus \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, $e_0 \Rightarrow_E z$ y $z \notin \langle \text{boolcnf} \rangle$ entonces $e_0 \oplus e_1 \Rightarrow_E \text{typeerror}$

Si $\oplus \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, $e_0 \Rightarrow_E [b]$, $e_1 \Rightarrow_E z$ y $z \notin \langle \text{boolcnf} \rangle$ entonces $e_0 \oplus e_1 \Rightarrow_E \text{typeerror}$

Si $e \Rightarrow_E z$ y $z \notin \langle \text{boolcnf} \rangle$ entonces if e then e_0 else $e_1 \Rightarrow_E \text{typeerror}$

4) 

viernes, 31 de mayo de 2024 11:45

4. Analice qué régimen 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)

viernes, 31 de mayo de 2024 11:52

5. Evalúe de modo eager y normal las expresiones $\langle \text{True} + 0, \Delta\Delta \rangle$ y $\langle \Delta\Delta, \text{True} + 0 \rangle$.

Normal:

$$\langle \text{True} + 0, \Delta\Delta \rangle \Rightarrow_N \langle \text{True} + 0, \Delta\Delta \rangle$$

$$\langle \Delta\Delta, \text{True} + 0 \rangle \Rightarrow_N \langle \Delta\Delta, \text{True} + 0 \rangle$$

Eager:

$$\langle \text{True} + 0, \Delta\Delta \rangle$$

$$\text{True} + 0$$

$$\text{True} \Rightarrow_E \text{True}$$

$$0 \Rightarrow_E 0$$

$$\Rightarrow_E \text{typeerror}$$

$$\Rightarrow_E \text{typeerror}$$

$$\langle \Delta\Delta, \text{True} + 0 \rangle$$

$$\Delta\Delta$$

No llega a nada

6)

viernes, 31 de mayo de 2024 11:52

6. Para el lenguaje aplicativo normal, reescribir utilizando patrones y **rec**, el término

letrec *par* $\equiv \lambda x. \text{if } x = 0 \text{ then true else } \text{impar}(x - 1)$
 impar $\equiv \lambda x. \text{if } x = 0 \text{ then false else } \text{par}(x - 1)$
in *e*

let *g* $\equiv \text{rec}(\lambda f. \lambda x. \text{if } x = 0 \text{ then } \langle \text{true}, \text{false} \rangle \text{ else } f((x - 1))f((x - 1)0))$

in let *par* $\equiv \lambda x. (g\ x).0$

in let *impar* $\equiv \lambda x. (g\ x).1$

in *e*

Donde $g \notin FV(e)$

Otra opción:

let *g* $\equiv \text{rec}(\lambda f. \langle \lambda x. \text{if } x = 0 \text{ then true else } (f.1)(x - 1), \lambda x. \text{if } x = 0 \text{ then false else } (f.0)(x - 1) \rangle)$

in let *par* $\equiv g.0$

in let *impar* $\equiv g.1$

in *e*

7)

viernes, 31 de mayo de 2024 11:53

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.

Asumo que indexa de 1

$$e = \text{rec}(\lambda f . \langle 0, f \rangle)$$

$e.1$

e

$$(\lambda f . \langle 0, f \rangle)e$$

$$\lambda f . \langle 0, f \rangle \Rightarrow_N \lambda f . \langle 0, f \rangle$$

$$\langle 0, f \rangle / [f : e] = \langle 0, e \rangle \Rightarrow_N \langle 0, e \rangle$$

$$\Rightarrow_N \langle 0, e \rangle$$

$$\Rightarrow_N \langle 0, e \rangle$$

$$0 \Rightarrow_N 0$$

0

8) 

viernes, 31 de mayo de 2024 11:54

8. Suponga que e es una expresión cerrada. Considere las siguientes expresiones:

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

Evaluar del modo eager y normal estos programas, considerando por separado los casos $e \Rightarrow \text{true}$ y $e \Rightarrow \text{false}$.

Eager:

$$\begin{aligned} \text{letrec } f \equiv \lambda x. \text{ if } e \text{ then } 1 \text{ else } f x &\text{ in } f 0 \\ f 0 / [f : \lambda x. \text{ letrec } f \equiv \lambda x. \text{ if } e \text{ then } 1 \text{ else } f x \text{ in } (\text{if } e \text{ then } 1 \text{ else } f x)] \\ &= \lambda x. \text{ letrec } f \equiv \lambda x. \text{ if } e \text{ then } 1 \text{ else } f x \text{ in } (\text{if } e \text{ then } 1 \text{ else } f x) \\ \lambda x. \dots &\Rightarrow_E \lambda x. \dots \\ 0 &\Rightarrow_E 0 \\ \text{letrec } f \equiv \lambda x. \text{ if } e \text{ then } 1 \text{ else } f x &\text{ in } (\text{if } e \text{ then } 1 \text{ else } f 0) \\ \text{if } e \text{ then } 1 \text{ else } f 0 / [f : \dots] \\ &\quad e \Rightarrow_E \text{true} \\ &\quad 1 \Rightarrow_E 1 \\ &\Rightarrow_E 1 \\ \Rightarrow_E 1 & \end{aligned}$$

9)

viernes, 31 de mayo de 2024 12:12

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”.

Falso:

Por ejemplo:

$$e = \lambda x. \Delta\Delta$$

$$z = \lambda x. \Delta\Delta$$

$$e' = \Delta\Delta$$