

Lenguaje Aplicativo, eager y normal

2017

Estructura de la materia a grandes rasgos:

Primera Parte: Lenguaje imperativo

Segunda Parte: Lenguaje aplicativo puro, y lenguaje aplicativo con referencias y asignación

Ejes de contenidos de la segunda parte

- 1 Cálculo Lambda
- 2 Lenguajes Aplicativos puros

Lenguajes Aplicativos

$$\begin{aligned}
 \langle exp \rangle & ::= \langle var \rangle \mid \langle exp \rangle \langle exp \rangle \mid \lambda \langle var \rangle . \langle exp \rangle \\
 & \mid \langle natconst \rangle \mid \langle boolconst \rangle \\
 & \mid - \langle exp \rangle \mid \langle exp \rangle + \langle exp \rangle \mid \langle exp \rangle * \langle exp \rangle \mid \dots \\
 & \mid \langle exp \rangle \geq \langle exp \rangle \mid \langle exp \rangle \leq \langle exp \rangle \mid \langle exp \rangle < \langle exp \rangle \mid \dots \\
 & \mid \langle exp \rangle \wedge \langle exp \rangle \mid \langle exp \rangle \vee \langle exp \rangle \mid \neg \langle exp \rangle \\
 & \mid \mathbf{if} \langle exp \rangle \mathbf{then} \langle exp \rangle \mathbf{else} \langle exp \rangle \\
 & \mid \mathbf{error} \mid \mathbf{typeerror}
 \end{aligned}$$

$$\langle natconst \rangle ::= 0 \mid 1 \mid 2 \mid \dots$$

$$\langle boolconst \rangle ::= \mathbf{true} \mid \mathbf{false}$$

Evaluación Eager

Formas canónicas

$$\langle cnf \rangle ::= \langle intcnf \rangle \mid \langle boolcnf \rangle \mid \langle funcnf \rangle$$
$$\langle intcnf \rangle ::= \dots \mid -2 \mid -1 \mid 0 \mid 1 \mid 2 \mid \dots$$
$$\langle boolcnf \rangle ::= \langle boolconst \rangle$$
$$\langle funcnf \rangle ::= \lambda \langle var \rangle . \langle exp \rangle$$

No hay formas canónicas para los errores.

Reglas para \Rightarrow_E

Notación: z es una metavariable para $\langle cnf \rangle$, c para $\langle intcnf \rangle$ ó para $\langle boolcnf \rangle$, e i para $\langle intcnf \rangle$.

Regla para las formas canónicas

$$\overline{z \Rightarrow_E z}$$

Regla para la aplicación

$$\frac{e \Rightarrow_E \lambda v. e_0 \quad e' \Rightarrow_E z' \quad (e_0/v \mapsto z') \Rightarrow_E z}{ee' \Rightarrow_E z}$$

Reglas para \Rightarrow_E

Regla para las operaciones enteras y booleanas:

$$\frac{e \Rightarrow_E [i] \quad e' \Rightarrow_E [i']}{e + e' \Rightarrow_E [i + i']}$$

Generalizando:

$$\frac{e \Rightarrow_E [c] \quad e' \Rightarrow_E [c']}{e \oplus e' \Rightarrow_E [c \oplus c']}$$

donde $\oplus \in \{+, -, *, =, \neq, \leq, \geq, \dots\}$

Reglas para \Rightarrow_E

Regla para las operaciones enteras y booleanas:

$$\frac{e \Rightarrow_E [c]}{\sim e \Rightarrow_E [\sim c]} \quad \text{donde } \sim \in \{-, \neg\}$$

$$\frac{e \Rightarrow_E [i] \quad e' \Rightarrow_E [i']}{e \oplus e' \Rightarrow_E [i \oplus i']} \quad (i' \neq 0)$$

donde $\oplus \in \{/, \div\}$

Reglas para \Rightarrow_E

$$\frac{e \Rightarrow_E \text{ true} \quad e_0 \Rightarrow_E z}{\text{if } e \text{ then } e_0 \text{ else } e_1 \Rightarrow_E z}$$

$$\frac{e \Rightarrow_E \text{ false} \quad e_1 \Rightarrow_E z}{\text{if } e \text{ then } e_0 \text{ else } e_1 \Rightarrow_E z}$$

Evaluación Normal

Formas Canónicas: Las mismas (por ahora).

Regla para las formas canónicas

$$\overline{z} \Rightarrow_N z$$

Regla para la aplicación

$$\frac{e \Rightarrow_N \lambda v.e_0 \quad (e_0/v \mapsto e') \Rightarrow_N z}{ee' \Rightarrow_N z}$$

Más reglas para \Rightarrow_N

Regla “lazy” para las operaciones booleanas:

$$\frac{e \Rightarrow_N \mathbf{false}}{e \wedge e' \Rightarrow_N \mathbf{false}}$$

Alternativa: uso de abreviaturas

$$\begin{aligned} e \wedge e' &=_{def} \mathbf{if } e \mathbf{ then } e' \mathbf{ else false} \\ e \vee e' &=_{def} \mathbf{if } e \mathbf{ then true else } e' \\ &\vdots \end{aligned}$$

Semántica Denotacional Eager

Para el CLE:

$$D = V_{\perp}$$

$$V \simeq V_{fun} \quad \text{con } V_{fun} = [V \rightarrow D]$$

Para el Lenguaje Aplicativo Eager:

$$D = (V + \{\mathbf{error}\} + \{\mathbf{typeerror}\})_{\perp}$$

$$V \simeq V_{int} + V_{bool} + V_{fun}$$

Notar que V es un predominio: no hay un elemento mínimo!

¿Cómo son las cadenas en V ?

Semántica Denotacional Eager: Valores

$$V \simeq V_{int} + V_{bool} + V_{fun} \quad D = (V + \{\mathbf{error}\} + \{\mathbf{typeerror}\})_{\perp}$$

$$V_{int} = \mathbb{Z}$$

$$V_{bool} = \{T, F\}$$

$$V_{fun} = [V \rightarrow D]$$

Ahora el isomorfismo ϕ y su inversa ψ tienen tipos:

$$\phi \in V \rightarrow V_{int} + V_{bool} + V_{fun}$$

$$\psi \in V_{int} + V_{bool} + V_{fun} \rightarrow V$$

Funciones auxiliares

$\iota_{int} \in V_{int} \rightarrow V$ se define:

$$\iota_{int} = \psi \circ \iota_0$$

donde

$$\iota_0 \in V_{int} \rightarrow V_{int} + V_{bool} + V_{fun}$$

Lo mismo para booleanos y funciones:

$$\iota_{int} \in V_{int} \rightarrow V$$

$$\iota_{bool} \in V_{bool} \rightarrow V$$

$$\iota_{fun} \in V_{fun} \rightarrow V$$

Constructores de D

$$D = (V + \{\mathbf{error}\} + \{\mathbf{typeerror}\})_{\perp}$$

$$err = (\iota_{\perp} \circ \iota_1)(\mathbf{error}) = \iota_{\perp}(\iota_1(\mathbf{error}))$$

$$tyerr = (\iota_{\perp} \circ \iota_2)(\mathbf{typeerror})$$

$$\iota_{norm} \in V \rightarrow D \qquad \iota_{norm} = \iota_{\perp} \circ \iota_0$$

Uso habitual: $\iota_{norm} \circ \iota_{int} \in V_{int} \rightarrow D$

Abreviatura: $\iota_{\underline{int}} = \iota_{norm} \circ \iota_{int} \in V_{int} \rightarrow D$

En general: $\iota_{\underline{\theta}} = \iota_{norm} \circ \iota_{\theta} \in V_{\theta} \rightarrow D$ con $\theta \in \{int, bool, fun\}$.

Más funciones auxiliares

Si $f \in V \rightarrow D$ entonces $f_* \in D \rightarrow D$ se define:

$$f_*(\iota_{norm} z) = f z$$

$$f_*(err) = err$$

$$f_*(tyerr) = tyerr$$

$$f_*(\perp) = \perp$$

Más funciones auxiliares

Si $f \in V_{int} \rightarrow D$ entonces $f_{int} \in V \rightarrow D$ se define:

$$f_{int}(\iota_{int} i) = f i$$

$$f_{int}(\iota_{bool} b) = tyerr$$

$$f_{int}(\iota_{fun} f) = tyerr$$

Si $f \in V_{int} \rightarrow D$, entonces podemos componer las dos transformaciones

$$(f_{int})_* \in D \rightarrow D$$

que lo escribimos

$$f_{int*} \in D \rightarrow D$$

Más funciones auxiliares

Si $f \in V_\theta \rightarrow D$ entonces $f_\theta \in V \rightarrow D$ se define:

$$\begin{aligned} f_\theta(\iota_\theta x) &= f x \\ f_\theta(\iota_{\theta'} y) &= \text{tyerr} \quad \text{si } \theta \neq \theta' \end{aligned}$$

En general: si $f \in V_\theta \rightarrow D$, entonces

$$\begin{aligned} f_{\theta*} &\in D \rightarrow D && \text{satisfaciendo} \\ f_{\theta*}(\iota_\theta x) &= f x \\ f_{\theta*}(\iota_{\theta'} y) &= \text{tyerr} && \text{si } \theta \neq \theta' \\ f_{\theta*}(\text{err}) &= \text{err} \\ f_{\theta*}(\text{tyerr}) &= \text{tyerr} \\ f_{\theta*}(\perp) &= \perp \end{aligned}$$

Ecuaciones semánticas

$$Env = \langle var \rangle \rightarrow V$$

$$\llbracket - \rrbracket \in \langle exp \rangle \rightarrow Env \rightarrow D$$

$$\llbracket 0 \rrbracket \eta = \underline{\iota}_{int} 0$$

$$\llbracket \mathbf{true} \rrbracket \eta = \underline{\iota}_{bool} T$$

$$\llbracket -e \rrbracket \eta = (\lambda i \in V_{int}. \underline{\iota}_{int} [-i])_{int*} (\llbracket e \rrbracket \eta)$$

$$\llbracket e + e' \rrbracket \eta = (\lambda i \in V_{int}. (\lambda i' \in V_{int}. \underline{\iota}_{int} [i + i']))_{int*} (\llbracket e' \rrbracket \eta)_{int*} (\llbracket e \rrbracket \eta)$$

$$\vdots$$

$$\llbracket e/e' \rrbracket \eta = (\lambda i \in V_{int}. (\lambda i' \in V_{int}. \mathbf{if } i' = 0 \mathbf{ then } err \mathbf{ else } \underline{\iota}_{int} [i/i']))_{int*} (\llbracket e' \rrbracket \eta)_{int*} (\llbracket e \rrbracket \eta)$$

Ecuaciones semánticas

$$Env = \langle var \rangle \rightarrow V$$

$$\llbracket _ \rrbracket \in \langle exp \rangle \rightarrow Env \rightarrow D$$

$$\llbracket v \rrbracket \eta = \iota_{norm}(\eta v)$$

$$\begin{aligned} \llbracket ee' \rrbracket \eta &= (\lambda f \in V_{fun}. (\lambda z \in V. f z)_*(\llbracket e' \rrbracket \eta))_{fun*}(\llbracket e \rrbracket \eta) \\ &= (\lambda f \in V_{fun}. f_*(\llbracket e' \rrbracket \eta))_{fun*}(\llbracket e \rrbracket \eta) \end{aligned}$$

$$\llbracket \lambda v. e \rrbracket \eta = \iota_{fun}(\lambda z \in V. \llbracket e \rrbracket [\eta | v : z])$$

$$\llbracket \mathbf{error} \rrbracket \eta = err$$

$$\llbracket \mathbf{typeerror} \rrbracket \eta = tyerr$$

Semántica Denotacional Normal

Para el CLN:

$$D = V_{\perp}$$

$$V \simeq V_{fun} \quad \text{con } V_{fun} = [D \rightarrow D]$$

Para el Lenguaje Aplicativo Normal (lo mismo que el eager):

$$D = (V + \{\mathbf{error}\} + \{\mathbf{typeerror}\})_{\perp}$$

$$V \simeq V_{int} + V_{bool} + V_{fun}$$

Semántica Denotacional Normal: Valores

$$V \cong V_{int} + V_{bool} + V_{fun}$$

$$V_{int} = \mathbf{Z}$$

$$V_{bool} = \{V, F\}$$

$$V_{fun} = [D \rightarrow D]$$

Los isomorfismos ϕ y ψ , lo mismo que las funciones auxiliares $\iota_\theta \in V_\theta \rightarrow V$, los resultados *err* y *tyerr*, la función $\iota_{norm} \in V \rightarrow D$, y las extensiones $f_* \in D \rightarrow D$ y $f_\theta \in V \rightarrow D$ se definen de la misma manera.

Funciones semánticas

Entornos normales:

$$Env = \langle var \rangle \rightarrow D$$

Función semántica:

$$\llbracket _ \rrbracket^{exp} \in \langle exp \rangle \rightarrow Env \rightarrow D$$

Ecuaciones semánticas

La mayoría de las ecuaciones son las mismas. Vamos a señalar aquellas que presentan algún cambio.

Evaluación lazy de las expresiones booleanas:

¿Cuál es el problema con :

$$\llbracket e \vee e' \rrbracket \eta =$$

$$\begin{aligned}
 & (\lambda b \in V_{bool}. \mathbf{if} \ b \ \mathbf{then} \ \underline{\iota_{bool}} T \\
 & \qquad \qquad \qquad \mathbf{else} \ (\lambda b' \in V_{bool}. \underline{\iota_{bool}} b')_{bool*} (\llbracket e' \rrbracket \eta) \\
 &)_{bool*} (\llbracket e \rrbracket \eta) \ ?
 \end{aligned}$$

$$\llbracket e \vee e' \rrbracket \eta = (\lambda b \in V_{bool}. \mathbf{if} \ b \ \mathbf{then} \ \underline{\iota_{bool}} T \ \mathbf{else} \ \llbracket e' \rrbracket \eta)_{bool*} (\llbracket e \rrbracket \eta) \ ?$$

Ecuaciones semánticas

Cálculo lambda:

$$\llbracket v \rrbracket \eta = \eta v$$

$$\llbracket ee' \rrbracket \eta = (\lambda f \in V_{fun}. f \llbracket e' \rrbracket \eta)_{fun*}(\llbracket e \rrbracket \eta)$$

$$\llbracket \lambda v.e \rrbracket \eta = \underline{\iota_{fun}}(\lambda d \in D. \llbracket e \rrbracket [\eta|v : d])$$

Tuplas

$$\langle exp \rangle ::= \langle \langle exp \rangle, \dots, \langle exp \rangle \rangle$$

$$\langle exp \rangle . \langle tag \rangle$$

$$\langle tag \rangle ::= \langle natconst \rangle$$

Los componentes de la tupla pueden ser de distintos tipos:

Ejemplo $M : \langle 2, \lambda x. \Delta \Delta, \lambda f x. f (f x) \rangle$

y las usamos accediendo a sus campos:

$(M, 0 M, 1) M, 2$

Evaluación de las Tuplas

Formas canónicas:

$$\langle cnf \rangle ::= \langle intcnf \rangle \mid \langle boolcnf \rangle \mid \langle funcnf \rangle \mid \langle tuplecnf \rangle$$

Formas canónicas de las tuplas para la evaluación Eager:

$$\langle tuplecnf \rangle ::= \langle \langle cnf \rangle, \dots, \langle cnf \rangle \rangle$$

Formas canónicas de tuplas para la evaluación Normal:

$$\langle tuplecnf \rangle ::= \langle \langle exp \rangle, \dots, \langle exp \rangle \rangle$$

Reglas para la evaluación Eager (Tuplas)

$$\frac{e_0 \Rightarrow_E z_0 \quad \dots \quad e_{n-1} \Rightarrow_E z_{n-1}}{\langle e_0, \dots, e_{n-1} \rangle \Rightarrow_E \langle z_0, \dots, z_{n-1} \rangle}$$

$$\frac{e \Rightarrow_E \langle z_0, \dots, z_{n-1} \rangle}{e.[k] \Rightarrow_E z_k} \quad (k < n)$$

Reglas para la evaluación Normal (Tuplas)

No hay regla para la tupla $\langle e_0, \dots, e_{n-1} \rangle$ porque es una forma canónica.

$$\frac{e \Rightarrow_N \langle e_0, \dots, e_{n-1} \rangle \quad e_k \Rightarrow_N z}{e.[k] \Rightarrow_N z} \quad (k < n)$$

Semántica denotacional de las tuplas

$$V \simeq V_{int} + V_{bool} + V_{fun} + V_{tuple}$$

Semántica denotacional Eager:

$$\begin{aligned} V_{tuple} = V^* &= \bigcup_{n \in \mathbb{N}} V^n \\ &= \{ \langle \rangle \} \cup \{ \langle v \rangle \mid v \in V \} \cup \dots \cup \{ \langle v_1, \dots, v_n \rangle \mid v_i \in V \} \end{aligned}$$

Notar que $\langle \iota_{tuple} \langle \iota_{int} 2, \iota_{bool} T \rangle \rangle \in V_{tuple}$

Semántica denotacional Normal:

$$V_{tuple} = D^*$$

Ecuaciones semánticas Eager

$$\llbracket \langle e_0, \dots, e_{n-1} \rangle \rrbracket \eta =$$

$$(\lambda z_0 \in V.$$

$$(\lambda z_1 \in V.$$

$$\dots (\lambda z_{n-1} \in V. \underline{\iota_{tuple}} \langle z_0, \dots, z_{n-1} \rangle)_* (\llbracket e_{n-1} \rrbracket) \dots$$

$$)_* (\llbracket e_1 \rrbracket \eta)$$

$$)_* (\llbracket e_0 \rrbracket \eta)$$

Ecuaciones semánticas Eager

$$\llbracket e.[k] \rrbracket \eta =$$

$$(\lambda t \in V_{tuple}.$$

$$\text{if } k \geq |t| \text{ then } tyerr \text{ else } \iota_{norm} t_k$$

$$)_{tuple*}(\llbracket e \rrbracket \eta)$$

Ecuaciones semánticas Normales

$$\llbracket \langle e_0, \dots, e_{n-1} \rangle \rrbracket \eta = \langle \llbracket e_0 \rrbracket \eta, \dots, \llbracket e_{n-1} \rrbracket \eta \rangle$$

$$\llbracket e.[k] \rrbracket \eta =$$

$$(\lambda t \in V_{tuple}.$$

$$\text{if } k \geq |t| \text{ then } tyerr \text{ else } t_k$$

$$)_{tuple*}(\llbracket e \rrbracket \eta)$$

Definiciones locales y patrones

$$\langle exp \rangle ::= \mathbf{let} \langle pat \rangle \equiv \langle exp \rangle, \dots, \langle pat \rangle \equiv \langle exp \rangle \mathbf{in} \langle exp \rangle \\ \lambda \langle pat \rangle . \langle exp \rangle$$

$$\langle pat \rangle ::= \langle var \rangle | \langle \langle pat \rangle, \dots, \langle pat \rangle \rangle$$

Por ejemplo, podemos escribir

$$\lambda \langle u, \langle v, w \rangle \rangle . u w v$$

en vez de

$$\lambda t . (t, 0)(t, 1, 1)(t, 1, 0)$$

$$(\lambda \langle u, \langle v, w \rangle \rangle . u w v) \langle M, \langle N, P \rangle \rangle \Rightarrow M P N$$

Se definen como abreviaturas (azúcar sintáctico):

$$\lambda\langle p_1, \dots, p_n \rangle.e = \\ \lambda v. \mathbf{let} \ p_1 \equiv v, 0, \dots, p_n \equiv v.[n - 1] \ \mathbf{in} \ e$$

donde v es una variable nueva (no ocurre libre en e ni en ninguno de los patrones).

$$\mathbf{let} \ p_1 \equiv e_1, \dots, p_n \equiv e_n \ \mathbf{in} \ e = \\ (\lambda p_1 \dots \lambda p_n.e)e_1 \dots e_n$$

Aplicando repetidamente estas dos transformaciones podemos eliminar los patrones que no sean variables y las definiciones locales (let) obteniendo una expresión cuya semántica ya está definida.

Tener en cuenta que cuando $n = 0$, **let** $p_1 \equiv e_1, \dots, p_n \equiv e_n$ **in** e quedará **let in** e , esto en realidad es directamente la expresión e

Recursión

$\langle exp \rangle ::= \mathbf{letrec} \langle var \rangle \equiv \lambda \langle var \rangle . \langle exp \rangle \mathbf{in} \langle exp \rangle$ (Eval. Eager)
 $\mathbf{rec} \langle exp \rangle$ (Eval. Normal)

El **letrec** permite hacer definiciones recursivas como

letrec *fact* $\equiv \lambda n. \mathbf{if} \ n = 0 \ \mathbf{then} \ 1 \ \mathbf{else} \ n * \mathit{fact}(n - 1) \ \mathbf{in} \ \mathit{fact} \ 2$

Regla para la Evaluación Eager

$$\frac{(e/v \mapsto \lambda u.e_0^*) \Rightarrow_E z}{\mathbf{letrec} \ v \equiv \lambda u.e_0 \ \mathbf{in} \ e \Rightarrow_E z}$$

donde

$$e_0^* = \mathbf{letrec} \ v \equiv \lambda u.e_0 \ \mathbf{in} \ e_0$$

Regla para la Evaluación Normal

$$\frac{e (\mathbf{rec} e) \Rightarrow_N z}{\mathbf{rec} e \Rightarrow_N z}$$

Semántica Denotacional Eager (Recursión)

$$\llbracket \mathbf{letrec} \ v \equiv \lambda u. e_0 \ \mathbf{in} \ e \rrbracket \eta = \llbracket e \rrbracket [\eta | v : \iota_{\mathit{func}}]$$

Semántica Denotacional Eager (Recursión)

$$\llbracket \mathbf{letrec} \ v \equiv \lambda u. e_0 \ \mathbf{in} \ e \rrbracket \eta = \llbracket e \rrbracket [\eta | v : \iota_{fun} g]$$

donde g es el menor punto fijo de

$$F \ f \ z = \llbracket e \rrbracket [\eta | v : \iota_{fun} f, \ u : z]$$

Semántica Denotacional Eager (Recursión)

$$\llbracket \mathbf{letrec} \ v \equiv \lambda u. e_0 \ \mathbf{in} \ e \rrbracket \eta = \llbracket e \rrbracket [\eta | v : \iota_{fun} g]$$

donde g es el menor punto fijo de

$$F \ f \ z = \llbracket e \rrbracket [\eta | v : \iota_{fun} f, \ u : z]$$

o sea

$$g = \mathbf{Y}_{V_{fun}} F \qquad \mathbf{Y}_{V_{fun}} F = \sqcup_i F^i \perp$$

Semántica Denotacional Eager (Recursión)

$$\llbracket \mathbf{letrec} \ v \equiv \lambda u.e_0 \ \mathbf{in} \ e \rrbracket \eta = \llbracket e \rrbracket [\eta | v : \iota_{fun} g]$$

donde g es el menor punto fijo de

$$F \ f \ z = \llbracket e \rrbracket [\eta | v : \iota_{fun} f, \ u : z]$$

o sea

$$g = \mathbf{Y}_{V_{fun}} F \qquad \mathbf{Y}_{V_{fun}} F = \sqcup_i F^i \perp$$

$$g = \mathbf{Y}_{V_{fun}} (\lambda f \in V_{fun} . \lambda z \in V . \llbracket e \rrbracket [\eta | v : \iota_{fun} f, \ u : z])$$

Semántica Denotacional Normal (Recursión)

$$\llbracket \mathbf{rec} \ e \rrbracket \eta = (\lambda f \in V_{fun}. \mathbf{Y}_D f)_{fun*}(\llbracket e \rrbracket \eta)$$

donde \mathbf{Y}_D es el operador de menor punto fijo

$$\mathbf{Y}_D f = \sqcup_i f^i \perp$$