

# Lenguajes y Compiladores

2015

## 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
- 3 Un Lenguaje Aplicativo con referencias y asignación

# Lenguajes Aplicativos

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

$$\langle \text{natconst} \rangle \mid \langle \text{boolconst} \rangle$$

$$- \langle \text{exp} \rangle \mid \langle \text{exp} \rangle + \langle \text{exp} \rangle \mid \langle \text{exp} \rangle * \langle \text{exp} \rangle \mid \dots$$

$$\langle \text{exp} \rangle \geq \langle \text{exp} \rangle \mid \langle \text{exp} \rangle \leq \langle \text{exp} \rangle \mid \langle \text{exp} \rangle < \langle \text{exp} \rangle \mid \dots$$

$$\langle \text{exp} \rangle \wedge \langle \text{exp} \rangle \mid \langle \text{exp} \rangle \vee \langle \text{exp} \rangle \mid \neg \langle \text{exp} \rangle$$

**if**  $\langle \text{exp} \rangle$  **then**  $\langle \text{exp} \rangle$  **else**  $\langle \text{exp} \rangle$

**error** | **typeerror**

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

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

# Evaluación Eager

## Formas canónicas

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

# 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 el error.

Reglas para  $\Rightarrow_E$ 

**Notación:**  $z$  es una metavariable para  $\langle cnf \rangle$ ,  $i$  para  $\langle intcnf \rangle$  y  $b$  para  $\langle boolcnf \rangle$ .

Regla para las formas canónicas

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

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:

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

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:

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

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

Reglas para  $\Rightarrow_E$ 

Regla para las operaciones enteras y booleanas:

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

Reglas para  $\Rightarrow_E$ 

Regla para las operaciones enteras y booleanas:

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

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

(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

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

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 enteras:

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

# Más reglas para $\Rightarrow_N$

Regla "lazy" para las operaciones enteras:

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

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:

$$D = V_{\perp}$$

$$V \simeq V_{fun} \quad (\text{aquí } V_{fun} = V \rightarrow D)$$

# Semántica Denotacional Eager

Para el CLE:

$$D = V_{\perp}$$

$$V \simeq V_{fun} \quad (\text{aquí } V_{fun} = V \rightarrow D)$$

Para el Lenguaje Aplicativo Eager:

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

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

# Semántica Denotacional Eager: Valores

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

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

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

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

# Semántica Denotacional Eager: Valores

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

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

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

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

Ahora el isomorfismo  $\phi$  y su inversa  $\psi$  satisfacen:

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

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

# Funciones auxiliares

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

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

donde

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

# 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:

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

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

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

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

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

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

# Constructores de $D$

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

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

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

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

## Más funciones auxiliares

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

$$f_*(\text{norm } a) = f a$$

$$f_*(\text{err}) = \text{err}$$

$$f_*(\text{tyerr}) = \text{tyerr}$$

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

## Más funciones auxiliares

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

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

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

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

## Más funciones auxiliares

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

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

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

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

Por ejemplo, si  $f \in V_{int} \rightarrow D$  entonces

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

## Más funciones auxiliares

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

$$f_\theta(\iota_\theta x) = f x \quad f_\theta a = \text{tyerr} \text{ si } a \text{ no es de la forma } \iota_\theta x$$

para  $\theta = \text{int}, \text{bool}, \text{fun}$

# Funciones semánticas

Entornos:

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

# Funciones semánticas

Entornos:

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

Función semántica:

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

# Ecuaciones semánticas

**Notación:**  $\iota_{\underline{\theta}} = \iota_{norm} \circ \iota_{\theta}$

$$\llbracket 0 \rrbracket_{\eta} = \iota_{\underline{int}} 0$$

$$\llbracket \text{true} \rrbracket_{\eta} = \iota_{\underline{bool}} T$$

# Ecuaciones semánticas

**Notación:**  $\underline{\iota}_\theta = \iota_{norm} \circ \iota_\theta$

$$[[0]]_\eta = \underline{\iota}_{int} 0$$

$$[[\text{true}]]_\eta = \underline{\iota}_{bool} T$$

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

# Ecuaciones semánticas

**Notación:**  $\iota_{\theta} = \iota_{norm} \circ \iota_{\theta}$

$$\llbracket 0 \rrbracket_{\eta} = \iota_{\underline{int}} 0$$

$$\llbracket \text{true} \rrbracket_{\eta} = \iota_{\underline{bool}} T$$

$$\llbracket -e \rrbracket_{\eta} = (\lambda i \in V_{int}. \iota_{\underline{int}} \lfloor -i \rfloor)_{int*} (\llbracket e \rrbracket_{\eta})$$

$$\llbracket e + e' \rrbracket_{\eta} = (\lambda i \in V_{int}. (\lambda i' \in V_{int}. \iota_{\underline{int}} \lfloor i + i' \rfloor)_{int*} (\llbracket e' \rrbracket_{\eta}))_{int*} (\llbracket e \rrbracket_{\eta})$$

$$\vdots$$

# Ecuaciones semánticas

**Notación:**  $\underline{\iota}_\theta = \iota_{norm} \circ \iota_\theta$

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

$$\llbracket \text{true} \rrbracket_\eta = \underline{\iota}_{bool} T$$

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

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

$$\vdots$$

$$\llbracket e/e' \rrbracket_\eta = (\lambda i \in V_{int}. (\lambda i' \in V_{int}.$$

$$\text{if } i' = 0 \text{ then err else } \underline{\iota}_{int}[i/i'])_{int^*}(\llbracket e' \rrbracket_\eta)_{int^*}(\llbracket e \rrbracket_\eta)$$

# Ecuaciones semánticas

$$\llbracket v \rrbracket \eta = \iota_{norm} V$$

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

# Ecuaciones semánticas

$$\llbracket v \rrbracket \eta = \iota_{norm} V$$

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

## Ecuaciones semánticas

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

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

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

# Ecuaciones semánticas

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

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

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

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

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

# Semántica Denotacional Normal

Para el CLN:

$$D = V_{\perp}$$

$$V \simeq V_{fun} \quad (\text{aquí } V_{fun} = D \rightarrow D)$$

# Semántica Denotacional Eager

Para el CLN:

$$D = V_{\perp}$$

$$V \simeq V_{fun} \quad (\text{aquí } V_{fun} = D \rightarrow D)$$

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

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

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

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

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

# Semántica Denotacional Eager: Valores

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

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

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

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

Los isomorfismos  $\phi$  y  $\psi$ , 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:

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

# 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 de presentan algún cambio.

Evaluación lazy de las expresiones booleanas:

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

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

# Ecuaciones semánticas

Cálculo lambda:

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

# Ecuaciones semánticas

Cálculo lambda:

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

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

# Ecuaciones semánticas

Cálculo lambda:

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

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

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

# Tuplas

$$\langle exp \rangle ::= \langle \langle exp \rangle, \dots, \langle exp \rangle \rangle$$
$$\langle exp \rangle . \langle tag \rangle$$
$$\langle tag \rangle ::= \langle natconst \rangle$$

# Evaluación de las Tuplas

## Formas canónicas:

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

# 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 para la evaluación Eager:

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

# 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 para la evaluación Eager:

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

## Formas canónicas para la evaluación Normal:

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

# Reglas para la evaluación Eager (Tuplas)

$$\frac{e_0 \Rightarrow_E Z_0 \quad \dots \quad e_{n-1} \Rightarrow_E Z_{n-1}}{\langle e_0, \dots, e_{n-1} \rangle \Rightarrow_E \langle Z_0, \dots, Z_{n-1} \rangle}$$

## 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.

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

# Semántica denotacional de las tuplas

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

# Semántica denotacional de las tuplas

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

Semántica denotacional Eager:

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

# Semántica denotacional de las tuplas

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

Semántica denotacional Eager:

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

Semántica denotacional Normal:

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

## Ecuaciones semánticas Eager

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

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

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

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

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

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

## 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

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

# 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

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

# Definiciones locales y patrones

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

$$\langle \text{pat} \rangle ::= \langle \text{var} \rangle \mid \langle \langle \text{pat} \rangle, \dots, \langle \text{pat} \rangle \rangle$$

Por ejemplo, podemos escribir

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

en vez de

$$\lambda t . (t.0)(t.1.0)(t.1.1)$$

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).

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.

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

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

$$\mathbf{rec} \langle exp \rangle \quad (\text{Eval. Normal})$$

El **letrec** permite hacer definiciones recursivas como

**letrec** *fact*  $\equiv \lambda n. \mathbf{if} \ n = 0 \ \mathbf{then} \ 1 \ \mathbf{else} \ n * \mathit{fact}(n - 1) \ \mathbf{in} \ \mathit{fact} \ 10$

# 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

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

# Semántica Denotacional Eager (Recursión)

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

# Semántica Denotacional Eager (Recursión)

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

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)

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

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)

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

donde  $g$  es el menor punto fijo de

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

o sea

$$g = \mathbf{Y}_{V_{fun}} F \qquad \mathbf{Y}_{V_{fun}} F = \sqcup_j F^j \perp$$

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

# 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

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

