Lenguajes y Compiladores

Miguel Pagano

2024

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

Lenguajes Aplicativos puros

Lenguajes Aplicativos puros

El lenguaje.

$$
\begin{array}{rl}
\langle exp \rangle ::= & \langle var \rangle \mid \langle exp \rangle \langle exp \rangle \mid \lambda\, \langle var \rangle . \langle exp \rangle \\
\mid & \langle natconst \rangle \mid \langle boolconst \rangle \\
\mid & -\langle exp \rangle \mid \langle exp \rangle + \langle exp \rangle \mid \langle exp \rangle * \langle exp \rangle \mid \ldots \\
\mid & \langle exp \rangle \geq \langle exp \rangle \mid \langle exp \rangle \leq \langle exp \rangle \mid \langle exp \rangle < \langle exp \rangle \mid \ldots \\
\mid & \langle exp \rangle \wedge \langle exp \rangle \mid \langle exp \rangle \vee \langle exp \rangle \mid \neg \langle exp \rangle \\
\mid & \mathbf{if}\ \langle exp \rangle\ \mathbf{then}\ \langle exp \rangle\ \mathbf{else}\ \langle exp \rangle
\end{array}
$$

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

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

Evaluación Eager

Formas canónicas

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

$$\langle intcnf \rangle ::= \ldots \mid -2 \mid -1 \mid 0 \mid 1 \mid 2 \mid \ldots$$

$$\langle boolcnf \rangle ::= \langle boolconst \rangle$$

$$\langle funcnf \rangle ::= \lambda \langle var \rangle . \langle exp \rangle$$

Reglas para $\Rightarrow_E$

Notación

- z como una metavariable para $\langle \mathit{cnf} \rangle$,
- c para $\langle \mathit{intcnf} \rangle$ ó para $\langle \mathit{boolcnf} \rangle$,
- e i para $\langle \mathit{intcnf} \rangle$.

Regla para las formas canónicas

$$\frac{}{z \;\Rightarrow_E\; z}$$

Regla para la aplicación

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

Reglas para $\Rightarrow_E$

Regla para las operaciones enteras y booleanas:

$$\frac{e \ \Rightarrow_E \ \lfloor i \rfloor \qquad e' \ \Rightarrow_E \ \lfloor i' \rfloor}{e + e' \ \Rightarrow_E \ \lfloor i + i' \rfloor}$$

Generalizando:

$$\frac{e \ \Rightarrow_E \ \lfloor c \rfloor \qquad e' \ \Rightarrow_E \ \lfloor c' \rfloor}{e \oplus e' \ \Rightarrow_E \ \lfloor c \oplus c' \rfloor}$$

donde $\oplus \in \{+, -, *, =, \neq, \leq, \geq, ...\}$ pero $\oplus \notin \{/, \ \div\}$.

¿Por qué excluimos a la división?

Reglas para $\Rightarrow_E$

Regla para las operaciones enteras y booleanas:

$$\frac{e \ \Rightarrow_E \ \lfloor c \rfloor}{\sim e \ \Rightarrow_E \ \lfloor \sim c \rfloor} \text{ con } \sim \in \{-, \neg\}$$

$$\frac{e \ \Rightarrow_E \ \lfloor i \rfloor \qquad e' \ \Rightarrow_E \ \lfloor i' \rfloor}{e \oplus e' \ \Rightarrow_E \ \lfloor i \oplus i' \rfloor} \ i' \neq 0, \text{ con } \oplus \in \{/, \ \div\}$$

Reglas para $\Rightarrow_E$

$$\frac{e \;\Rightarrow_E\; \textbf{true} \qquad e_0 \;\Rightarrow_E\; z}{\textbf{if}\; e \;\textbf{then}\; e_0 \;\textbf{else}\; e_1 \;\Rightarrow_E\; z}$$

$$\frac{e \;\Rightarrow_E\; \textbf{false} \qquad e_1 \;\Rightarrow_E\; z}{\textbf{if}\; e \;\textbf{then}\; e_0 \;\textbf{else}\; e_1 \;\Rightarrow_E\; z}$$

Evaluación Normal

Formas Canónicas: Las mismas (por ahora).

Regla para las formas canónicas

$$\frac{}{z \Rightarrow_N z}$$

Regla para la aplicación

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

Más reglas para $\Rightarrow_N$

Regla "lazy" para las operaciones booleanas:

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

Alternativa: uso de abreviaturas

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

Tuplas

$\langle exp \rangle$ ::= ... | $\langle\langle exp \rangle, \ldots, \langle exp \rangle\rangle$
| $\langle exp \rangle.\langle tagexp \rangle$

$\langle tagexp \rangle$ ::= $\langle natconst \rangle$

Los componentes de la tupla pueden ser de distintos tipos:

Ejemplo si $M = \langle 2, \lambda\, n.1, \lambda\, f\, x.f\, (f\, x)\rangle$

entonces

$$\langle M.0, M.2\, M.1\rangle \Rightarrow_E \langle 2, \lambda x.(\lambda n.1)\, ((\lambda n.1)\, x)\rangle$$

$$\langle M.0, M.2\, M.1\rangle \Rightarrow_N \langle 2, (\lambda f x.f\, (f\, x))(\lambda n.1)\rangle$$

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 de las tuplas para la evaluación Eager:

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

Formas canónicas de tuplas para la evaluación Normal:

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

Evaluación de las Tuplas (Reglas para Eager)

$$\frac{e_0 \Rightarrow_E z_0 \qquad \cdots \qquad e_{n-1} \Rightarrow_E z_{n-1}}{\langle e_0, ..., e_{n-1} \rangle \Rightarrow_E \langle z_0, ..., z_{n-1} \rangle}$$

$$\frac{e \Rightarrow_E \langle z_0, ..., z_{n-1} \rangle}{e.\lfloor k \rfloor \Rightarrow_E z_k} \; k < n$$

Evaluación de las Tuplas (Reglas para Normal)

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

Dicho de otro modo, la regla es $z \Rightarrow_N z$.

$$\frac{e \Rightarrow_N \langle e_0, ..., e_{n-1} \rangle \qquad e_k \Rightarrow_N z}{e.\lfloor k \rfloor \Rightarrow_N z} \; k < n$$

Semántica Denotacional Eager

Para el Cálculo Lambda Eager teníamos:

$$D = V_{\perp}$$

$$V \;\simeq\; V_{fun} \qquad \text{con } V_{fun} = [V \to D]$$

Semántica Denotacional Eager: Dominios

Para el Lenguaje Aplicativo Eager necesitamos más valores:

$$V \underset{\overleftarrow{\psi}}{\overset{\overrightarrow{\phi}}{\cong}} V_{int} + V_{bool} + V_{fun} \qquad \text{y} \qquad D = (V + \{\textbf{error}\} + \{\textbf{typeerror}\})_{\perp}$$

Los pre-dominios para valores son:

$$V_{int} = \mathbb{Z}$$
$$V_{bool} = \{T, F\}$$
$$V_{fun} = [V \to D]$$

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

¿Cómo son las cadenas en V?

Funciones auxiliares

$\iota_{int} \in V_{int} \to V$ se define como

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

donde

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

Lo mismo para booleanos y funciones:

$$\psi \circ \iota_0 = \iota_{int} \in V_{int} \to V$$

$$\psi \circ \iota_1 = \iota_{bool} \in V_{bool} \to V$$

$$\psi \circ \iota_2 = \iota_{fun} \in V_{fun} \to V$$

“Constructores” de D

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

$$\begin{aligned} err, tyerr &\in D \\ err &= \iota_\perp(\iota_1\,(\textbf{error})) \\ tyerr &= \iota_\perp(\iota_2(\textbf{typeerror})) \\ \iota_{norm} &\in V \to D \\ \iota_{norm} &= \iota_\perp \circ \iota_0 \end{aligned}$$

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

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

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

Más funciones auxiliares

Si $f : V \to D$, entonces $f_* : D \to D$ se define:

$$\begin{aligned} f_*(\iota_{norm}\, z) &= f\ z \\ f_*(err) &= err \\ f_*(tyerr) &= tyerr \\ f_*(\perp) &= \perp \end{aligned}$$

Más funciones auxiliares

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

$$f_{int}(\iota_{int}\, i) = f\; i$$
$$f_{int}(\iota_{bool}\, b) = tyerr$$
$$f_{int}(\iota_{fun}\, f) = tyerr$$

Si $f : V_{int} \to D$, podemos componer las dos transformaciones

$$(f_{int})_* : D \to D$$

que lo escribimos

$$f_{int*} : D \to D$$

Más funciones auxiliares

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

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

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

$$\begin{array}{rcll} f_{\theta*} & \in & D \to D & \quad \text{satisfaciendo} \\ f_{\theta*}(\iota_{\underline{\theta}}\, x) & = & f\, x & \\ f_{\theta*}(\iota_{\underline{\theta'}}\, y) & = & tyerr & \text{si } \theta \neq \theta' \\ f_{\theta*}(err) & = & err & \\ f_{\theta*}(tyerr) & = & tyerr & \\ f_{\theta*}(\perp) & = & \perp & \end{array}$$

Ecuaciones semánticas

$$Env = \langle \mathit{var} \rangle \to V$$
$$[\![\_]\!] \in \langle \mathit{exp} \rangle \to Env \to D$$

$$\begin{aligned}
[\![0]\!]\eta &= \iota_{\underline{int}} 0 \\
[\![\mathbf{true}]\!]\eta &= \iota_{\underline{bool}} T \\
[\![-e]\!]\eta &= (\lambda i \in V_{int}.\ \iota_{\underline{int}}(-i))_{int*}([\![e]\!]\eta) \\
[\![e + e']\!]\eta &= (\lambda i \in V_{int}.\ (\lambda i' \in V_{int}.\ \iota_{\underline{int}}(i + i'))_{int*}([\![e']\!]\eta))_{int*}([\![e]\!]\eta) \\
&\ \vdots \\
[\![e/e']\!]\eta &= (\lambda i \in V_{int}.\ (\lambda i' \in V_{int}. \\
&\qquad \mathbf{if}\ i' = 0\ \mathbf{then}\ err\ \mathbf{else}\ \iota_{\underline{int}}(i/i'))_{int*}([\![e']\!]\eta))_{int*}([\![e]\!]\eta)
\end{aligned}$$

Ecuaciones semánticas

$$Env = \langle \mathit{var} \rangle \to V$$
$$[\![\_]\!] \in \langle \mathit{exp} \rangle \to Env \to D$$

$$\begin{array}{lcl}
[\![v]\!]\eta & = & \iota_{norm}(\eta\, v) \\
[\![ee']\!]\eta & = & (\lambda f \in V_{fun}.\, (\lambda z \in V.\, f\ z)_*([\![e']\!]\eta))_{fun*}([\![e]\!]\eta) \\
 & = & (\lambda f \in V_{fun}.\, f_*([\![e']\!]\eta))_{fun*}([\![e]\!]\eta) \\
[\![\lambda v.e]\!]\eta & = & \iota_{\underline{fun}}(\lambda z \in V.\, [\![e]\!][\eta|v : z])
\end{array}$$

Semántica Denotacional Normal

Para el Cálculo Lambda Normal:

$$D = V_\perp$$

$$V \ \simeq \ V_{fun} \qquad \text{con } V_{fun} = [D \to D]$$

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

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

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

Semántica Denotacional Normal: Valores

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

$$V_{int} = \mathbb{Z}$$
$$V_{bool} = \{T, F\}$$
$$V_{fun} = [D \to D]$$

$$\phi \in V \to V_{int} + V_{bool} + V_{fun} \qquad err, tyerr \in D$$
$$\psi \in V_{int} + V_{bool} + V_{fun} \to V$$
$$\iota_{\theta} \in V_{\theta} \to V \qquad f_{\theta} \in V \to D$$
$$\iota_{norm} \in V \to D \qquad f_{*} \in D \to D$$

Ecuaciones semánticas

Entornos: $Env = \langle var \rangle \to D$

Función semántica: $[\![\_]\!] \in \langle exp \rangle \to Env \to D$

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 :

$$[\![e \vee e']\!]\eta =$$

$$(\lambda b \in V_{bool}.\ \textbf{if}\ b\ \textbf{then}\ \iota_{\underline{bool}} T$$

$$\textbf{else}\ (\lambda b' \in V_{bool}.\iota_{\underline{bool}} b')_{bool*}([\![e']\!]\eta)$$

$$)_{bool*}([\![e]\!]\eta)\ ?$$

Ecuaciones semánticas

Cálculo lambda:

$$Env = \langle \mathit{var} \rangle \to V$$
$$[\![\_]\!] \in \langle \mathit{exp} \rangle \to Env \to D$$

$$\begin{array}{lcl}
[\![v]\!]\eta & = & \eta\, v \\
[\![ee']\!]\eta & = & (\lambda f \in V_{fun}.\, f\, [\![e']\!]\eta)_{fun*}([\![e]\!]\eta) \\
[\![\lambda v.e]\!]\eta & = & \iota_{\underline{fun}}(\lambda d \in D.\, [\![e]\!][\eta|v : d])
\end{array}$$

Semántica denotacional de las tuplas

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

Semántica denotacional Eager:

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

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

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

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

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

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

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

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

Ecuaciones semánticas Eager

$$[\![e.\lfloor k \rfloor]\!] \eta =$$

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

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

$$)_{tuple*}([\![e]\!] \eta)$$

Ecuaciones semánticas Normales

$$[\![\langle e_0, ..., e_{n-1}\rangle]\!]\eta = \iota_{\underline{tuple}}\langle [\![e_0]\!]\eta, ..., [\![e_{n-1}]\!]\eta\rangle$$

$$[\![e.\lfloor k \rfloor]\!]\eta =$$

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

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

$$)_{tuple*}([\![e]\!]\eta)$$

Definiciones locales y patrones

$$\begin{array}{rcl} \langle exp \rangle & ::= & \mathbf{let}\ \langle pat \rangle \equiv \langle exp \rangle, \ldots, \langle pat \rangle \equiv \langle exp \rangle\ \mathbf{in}\ \langle exp \rangle \\ & | & \lambda \langle pat \rangle . \langle exp \rangle \end{array}$$

$$\langle pat \rangle ::= \langle var \rangle \mid \langle \langle pat \rangle, \ldots, \langle pat \rangle \rangle$$

Por ejemplo, podemos escribir

$$\mathbf{let}\ \langle m, \langle f, G \rangle \rangle \equiv \langle 2, \langle \lambda n.1, \lambda fx.f(fx) \rangle \rangle\ \mathbf{in}\ G\ f$$

en vez de

$$(\lambda t.(t.1.1)\ (t.1.0))\ \langle 2, \langle \lambda n.1, \lambda fx.f(fx) \rangle \rangle$$

Definiciones locales y patrones (sin azúcar)

Se definen como abreviaturas (azucar sintactico):

$$\lambda\langle p_1, \ldots, p_n\rangle \,.\, e \;\stackrel{def}{=}\; \lambda v.\, \textbf{let}\, p_1 \equiv v.0, \ldots, p_n \equiv v.[n-1]\, \textbf{in}\, e$$

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

$$\textbf{let}\, p_1 \equiv e_1, \ldots, p_n \equiv e_n\, \textbf{in}\, e \;\stackrel{def}{=}\; (\lambda\, p_1 \,\ldots\, \lambda p_n \,.\, e)\, e_1 \,\ldots\, e_n$$

Recursión

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

$$\langle exp \rangle ::= \quad \ldots \mid \textbf{rec}\ \langle exp \rangle \qquad \text{(Eval. Normal)}$$

El letrec permite hacer definiciones recursivas como

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

Regla para la Evaluación Eager

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

donde $e_0^* \;=\; \mathbf{letrec}\ f \equiv \lambda u.e_0\ \mathbf{in}\ e_0$

Regla para la Evaluación Normal

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

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

donde g es el menor punto fijo de

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

o sea

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

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

$$[\![\textbf{rec}\, e]\!]\eta \;=\; (\lambda f \in V_{fun}.\, \mathbf{Y}_D f)_{fun*}([\![e]\!]\eta)$$

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

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