Resumen de la materia
Lenguajes y Compiladores

Agustín Curto, agucurto95@gmail.com

2019

Índice

Nota: Este resumen se corresponde con la materia dictada en el año 2019. El autor no se responsabilizan de posibles cambios que pudiesen realizarse en los temas dictados en la misma, así como tampoco de errores involuntarios que pudiesen existir en dicho resumen.

Por favor, mejorá este documento en github
https://github.com/ResumenesFaMAF/resumenCompiladores

1. Semántica

Dirección por sintáxis

Un conjunto de ecuaciones es *dirigido por sintáxis* cuando se satisfacen las siguientes condiciones:

- hay una ecuación por cada producción de la gramática abstracta
- cada ecuación que expresa el significado de una frase compuesta, lo hace puramente en función de los significados de sus subfrases inmediatas

Composicionalidad

Una semántica se dice que es *composicional* cuando el significado de una frase no depende de ninguna propiedad de sus subfrases, salvo de sus significados.

Dirección por sintáxis $\Rightarrow$ composicionalidad

Sustitución

$$\begin{aligned}
\Delta &= \langle var \rangle \to \langle intexp \rangle \\
&\in \langle intexp \rangle \times \Delta \to \langle intexp \rangle \\
(Qv.b)/\delta &= Qv_{new}.(b/[\delta|v : v_{new}]) \\
\\
\text{donde } v_{new} &\notin \bigcup_{\omega \in FV(b-\{v\})} FV(\delta\omega)
\end{aligned}$$

Propiedades

- **Teorema de Coincidencia:** expresa que el significado de una frase no puede depender de variables que no ocurran libres en la misma.

  Enunciado: Si dos estados σ, σ' coinciden en las variables libres de p, entonces da lo mismo evaluar p en σ o σ'.
  $$(\forall \omega \in FV(p).\sigma\omega = \sigma'\omega) \Rightarrow [\![p]\!]\sigma = [\![p]\!]\sigma'$$
- **Teorema de Renombre:** asegura que el significado no depende de las variables ligadas de una frase.

  Enunciado: Los nombres de las variables ligadas no tienen importancia.
  $$u \notin FV(q) - \{v\} \Rightarrow [\![\forall u.q/v \to u]\!] = [\![\forall v.q]\!]$$
- **Teorema de Sustitución:** Si aplico la sustitución δ a p y luego evaluo en el estado σ, puedo obtener el mismo resultado a partir de p sin sustituir si evaluo en un estado que hace el trabajo de δ y de σ (en las variables libres de p).
  $$(\forall \omega \in FV(p).[\![\delta\ \omega]\!]\sigma = \sigma'\omega) \Rightarrow [\![p/\delta]\!]\sigma = [\![p]\!]\sigma'$$

2. Lenguaje Imperativo Simple

<u>Semántica Denotacional</u> (con fallas)

$$\begin{aligned}
[\![_]\!] &\in \langle comm \rangle \to \Sigma \to \Sigma'_\perp \\
[\![\mathbf{skip}]\!]\sigma &= \sigma \\
[\![\mathbf{fail}]\!]\sigma &= \langle \mathbf{abort}, \sigma \rangle \\
[\![v := e]\!]\sigma &= [\sigma|v : [\![e]\!]\sigma]
\end{aligned}$$

$$
\begin{aligned}
f_*, f_+, f_\dagger &\in \Sigma'_\perp \to \Sigma'_\perp \\
f_* x &= \begin{cases} f\sigma & \text{si } x = \sigma \in \Sigma \\ x & \text{c.c} \end{cases} \\
f_+ x &= \begin{cases} f\sigma & \text{si } x = \langle \textbf{abort}, \sigma \rangle \in \{\textbf{abort}\} \times \Sigma \\ x & \text{c.c} \end{cases} \\
f_\dagger x &= \begin{cases} \langle \textbf{abort}, f\sigma \rangle & x = \langle \textbf{abort}, \sigma \rangle \\ fx & x \in \Sigma \\ \perp & x = \perp \end{cases}
\end{aligned}
$$

Variables libres y asignables

$$
\begin{array}{lcllcl}
FV(\textbf{skip}) & = & \emptyset & FA(\textbf{skip}) & = & \emptyset \\
FV(v := e) & = & \{v\} \cup FV(e) & FA(v := e) & = & \{v\} \\
FV(c_0; c_1) & = & FV(c_0) \cup FV(c_1) & FA(c_0; c_1) & = & FA(c_0) \cup FA(c_1) \\
FV(\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1) & = & FV(b) \cup FV(c_0) \cup FV(c_1) & FA(\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1) & = & FA(c_0) \cup FA(c_1) \\
FV(\textbf{while } b \textbf{ do } c) & = & FV(b) \cup FV(c) & FA(\textbf{while } b \textbf{ do } c) & = & FA(c) \\
FV(\textbf{newvar } v := e \textbf{ in } c) & = & FV(e) \cup (FV(c) - \{v\}) & FA(\textbf{newvar } v := e \textbf{ in } c) & = & FA(c) - \{v\}
\end{array}
$$

Teorema de Coincidencia (TC)

Si dos estados σ y σ' coinciden en las variables libres de c, entonces da lo mismo evaluar c en σ o σ'.

1. $\forall \omega \in FV(c).\ \sigma\omega = \sigma'\omega$ entonces
   - $[\![c]\!]\sigma = \perp = [\![c]\!]\sigma'$ o
   - $[\![c]\!]\sigma \neq \perp \neq [\![c]\!]\sigma'$ y $\forall \omega \in FV(c).\ [\![c]\!]\sigma\omega = [\![c]\!]\sigma'\omega$
2. Si $[\![c]\!]\sigma \neq \perp$, entonces $\forall \omega \notin FA(c).\ [\![c]\!]\sigma\omega = \sigma\omega$

Teorema de Renombre (TR)

No importa el nombre de las variables utilizadas en las declaraciones de variables locales, es decir, las ligadas.

$u \notin FV(c) - \{v\} \Rightarrow [\![\textbf{newvar } u := e \textbf{ in } c/v \to u]\!] = [\![\textbf{newvar } v := e \textbf{ in } c]\!]$

Teorema de Sustitución (TS)

Si δ es inyectiva sobre $FV(c)$ y $\forall \omega \in FV(c).\ \sigma(\delta\omega) = \sigma'\omega$ entonces:

- $[\![c/\delta]\!]\sigma = \perp = [\![c]\!]\sigma'$ o
- $[\![c/\delta]\!]\sigma \neq \perp \neq [\![c]\!]\sigma'$ y $\forall \omega \in FV(c).\ [\![c/\delta]\!]\sigma(\delta\omega) = [\![c]\!]\sigma'\omega$

Lema de Sustitución (LS)

Sea $FV(c) \subseteq V \subseteq \langle var \rangle$ tal que δ es inyectiva sobre V y $\forall \omega \in V.\ \sigma(\delta\omega) = \sigma'\omega$ entonces:

- $[\![c/\delta]\!]\sigma = \perp = [\![c]\!]\sigma'$ o
- $[\![c/\delta]\!]\sigma \neq \perp \neq [\![c]\!]\sigma'$ y $\forall \omega \in V. [\![c/\delta]\!]\sigma(\delta\omega) = [\![c]\!]\sigma'\omega$

Semántica Denotacional (Completa)

$$
\begin{aligned}
\iota_{term} &= \psi \cdot \iota_{\perp} \cdot \iota_0 \cdot \iota_{norm} \in \Sigma \to \Omega \\
\iota_{abort} &= \psi \cdot \iota_{\perp} \cdot \iota_0 \cdot \iota_{abnorm} \in \Sigma \to \Omega \\
\iota_{out} &= \psi \cdot \iota_{\perp} \cdot \iota_1 \in \mathbb{Z} \times \Sigma \to \Omega \\
\iota_{in} &= \psi \cdot \iota_{\perp} \cdot \iota_2 \in (\mathbb{Z} \to \Sigma) \to \Omega \\
\perp_{\Omega} &= \psi(\perp) \in \Omega
\end{aligned}
$$

$$
f_*, f_+, f_\dagger \in \Sigma'_{\perp} \to \Sigma'_{\perp}
$$

$$
\begin{aligned}
[\![\mathbf{skip}]\!]\sigma &= \iota_{term}\sigma \\
[\![\mathbf{fail}]\!]\sigma &= \iota_{abort}\sigma \\
[\![v := e]\!]\sigma &= \iota_{term}[\sigma|v : [\![e]\!]\sigma] \\
[\![!e]\!]\sigma &= \iota_{out}([\![e]\!]\sigma, \iota_{term}\sigma) \\
[\![?v]\!]\sigma &= \iota_{in}(\lambda n \in \mathbb{Z}.\iota_{term}[\sigma|v : n]) \\
[\![\mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1]\!]\sigma &= \begin{cases} [\![c_0]\!]\sigma & \text{si } [\![b]\!]\sigma \\ [\![c_1]\!]\sigma & c.c \end{cases} \\
[\![c_0; c_1]\!]\sigma &= [\![c_1]\!]_*([\![c_0]\!]\sigma) \\
[\![\mathbf{catchin}\ c_0\ \mathbf{with}\ c_1]\!]\sigma &= [\![c_1]\!]_+([\![c_0]\!]\sigma) \\
[\![\mathbf{newvar}\ v := e\ \mathbf{in}\ c]\!]\sigma &= (\lambda\sigma' \in \Sigma.[\sigma'|v : \sigma v])_\dagger([\![c]\!][\sigma|c : [\![e]\!]\sigma]) \\
[\![\mathbf{while}\ b\ \mathbf{do}\ c]\!]\sigma &= \bigsqcup_{i=0}^{\infty} F^i \underline{\perp} \\
F\ \omega\ \sigma &= \begin{cases} \omega_*([\![c]\!]\sigma) & \text{si } [\![b]\!]\sigma \\ \iota_{term}\sigma & \text{c.c} \end{cases}
\end{aligned}
$$

$$
f_* x = \begin{cases} \perp_\Omega & x = \perp_\Omega \\ f\sigma & x = \iota_{term}\sigma \\ \iota_{abort}\sigma & x = \iota_{abort}\sigma \\ \iota_{out}(n, f_*\omega) & x = \iota_{out}(n,\omega) \\ \iota_{in}(f_* \cdot g) & x = \iota_{in} g \end{cases}
$$

$$
f_+ x = \begin{cases} \perp_\Omega & x = \perp_\Omega \\ \iota_{term}\sigma & x = \iota_{term}\sigma \\ f\sigma & x = \iota_{abort}\sigma \\ \iota_{out}(n, f_+\omega) & x = \iota_{out}(n,\omega) \\ \iota_{in}(f_+ \cdot g) & x = \iota_{in} g \end{cases}
$$

$$
f_\dagger x = \begin{cases} \perp_\Omega & x = \perp_\Omega \\ \iota_{term}(f\sigma) & x = \iota_{term}\sigma \\ \iota_{abort}(f\sigma) & x = \iota_{abort}\sigma \\ \iota_{out}(n, f_\dagger\omega) & x = \iota_{out}(n,\omega) \\ \iota_{in}(f_\dagger \cdot g) & x = \iota_{in} g \end{cases}
$$

Semántica Operacional

$$
\frac{}{\langle \mathbf{skip}, \sigma\rangle \to \sigma}
$$

$$
\frac{}{\langle v := e, \sigma\rangle \to\ [\sigma|v : [\![e]\!]\sigma]}
$$

$$
\frac{\langle c_0, \sigma\rangle \to \sigma'}{\langle c_0; c_1, \sigma\rangle \to \langle c_1, \sigma'\rangle}
$$

$$
\frac{\langle c_0, \sigma\rangle \to \langle c_0', \sigma'\rangle}{\langle c_0; c_1, \sigma\rangle \to \langle c_0'; c_1, \sigma'\rangle}
$$

$$
\frac{([\![e]\!]\sigma = V)}{\langle \mathbf{if}\ e\ \mathbf{then}\ c\ \mathbf{else}\ c', \sigma\rangle \to \langle c, \sigma\rangle}
$$

$$
\frac{([\![e]\!]\sigma = F)}{\langle \mathbf{if}\ e\ \mathbf{then}\ c\ \mathbf{else}\ c', \sigma\rangle \to \langle c', \sigma\rangle}
$$

$$\frac{(\llbracket e \rrbracket \sigma = F)}{\langle \textbf{while } e \textbf{ do } c, \sigma \rangle \to \sigma}$$

$$\frac{(\llbracket e \rrbracket \sigma = V)}{\langle \textbf{while } e \textbf{ do } c, \sigma \rangle \to \langle c; \textbf{while } e \textbf{ do } c, \sigma \rangle}$$

$$\frac{\langle c, [\sigma | v : \llbracket e \rrbracket \sigma] \rangle \to \sigma'}{\langle \textbf{newvar } v := e \textbf{ in } c, \sigma \rangle \to [\sigma' | v : \sigma v]}$$

$$\frac{\langle c, [\sigma | v : \llbracket e \rrbracket \sigma] \rangle \to \langle c', \sigma' \rangle}{\langle \textbf{newvar } v := e \textbf{ in } c, \sigma \rangle \to \langle \textbf{newvar } v := \sigma' v \textbf{ in } c', [\sigma' | v : \sigma v] \rangle}$$

$$\{\!\!\{ c \}\!\!\}_\sigma = \begin{cases} \perp & \text{si } \langle c, \sigma \rangle \uparrow \\ \sigma' & \text{si existe } \sigma' \text{ tal que } \langle c, \sigma \rangle \to^* \sigma' \end{cases}$$

Con fallas

$$\frac{}{\langle \textbf{fail}, \sigma \rangle \to \langle \textbf{abort}, \sigma \rangle}$$

$$\frac{\langle c_0, \sigma \rangle \to \langle \textbf{abort}, \sigma' \rangle}{\langle c_0; c_1, \sigma \rangle \to \langle \textbf{abort}, \sigma' \rangle}$$

$$\frac{\langle c_0, \sigma \rangle \to \sigma'}{\langle \textbf{catchin } c_0 \textbf{ with } c_1, \sigma \rangle \to \sigma'}$$

$$\frac{\langle c_0, \sigma \rangle \to \langle \textbf{abort}, \sigma' \rangle}{\langle \textbf{catchin } c_0 \textbf{ with } c_1, \sigma \rangle \to \langle c_1, \sigma' \rangle}$$

$$\frac{\langle c_0, \sigma \rangle \to \langle c_0', \sigma' \rangle}{\langle \textbf{catchin } c_0 \textbf{ with } c_1, \sigma \rangle \to \langle \textbf{catchin } c_0' \textbf{ with } c_1, \sigma' \rangle}$$

$$\frac{\langle c, \sigma \rangle \to \langle \textbf{abort}, \sigma' \rangle}{\langle \textbf{newvar } v := e \textbf{ in } c, \sigma \rangle \to \langle \textbf{abort}, [\sigma' | v : \sigma v] \rangle}$$

$$\{\!\!\{ c \}\!\!\}_\sigma = \begin{cases} \perp & \text{si } \langle c, \sigma \rangle \uparrow \ \ (\uparrow\text{: ejecución infinita}) \\ \sigma' & \text{si existe } \sigma' \text{ tal que } \langle c, \sigma \rangle \to^* \sigma' \\ \langle \textbf{abort}, \sigma' \rangle & \text{si existe } \sigma' \text{ tal que } \langle c, \sigma \rangle \to^* \langle \textbf{abort}, \sigma \rangle' \end{cases}$$

<u>PROPIEDADES:</u>

- **Lema 1:**

  1. Si $\langle c_0, \sigma \rangle \to^* \sigma'$ entonces $\langle c_0; c_1, \sigma \rangle \to^* \langle c_1, \sigma' \rangle$
  2. Si $\langle c, [\sigma | v : \llbracket e \rrbracket \sigma] \rangle \to^* \sigma'$ entonces $\langle \textbf{newvar } v := e \textbf{ in } c, \sigma \rangle \to^* [\sigma' \ | v : \sigma v]$

3. Si $\langle c, [\sigma|v : [\![e]\!]\sigma]\rangle \rightarrow^* \langle c', \sigma'\rangle$ entonces $\langle \textbf{newvar } v := e \textbf{ in } c, \sigma\rangle \rightarrow^* \langle \textbf{newvar } v := \sigma' x \textbf{ in } c', [\sigma'|\ v : \sigma v]\rangle$

Prueba:

- **Lema 2:**

1. $\langle c, \sigma\rangle \rightarrow \sigma' \Rightarrow [\![c]\!]\sigma = \sigma'$
2. $\langle c, \sigma\rangle \rightarrow \langle c', \sigma'\rangle \Rightarrow [\![c]\!]\sigma = [\![c']\!]\sigma'$

Prueba:

- **Lema 3:** $[\![c]\!]\sigma = \sigma' \Rightarrow \langle c, \sigma\rangle \rightarrow^* \sigma'$

  Prueba:

- **Teorema:** Para todo comando c se tiene $\{\!\{c\}\!\} = [\![c]\!]$

  Prueba:

3. Recursión

- **Orden parcial:** Relación reflexiva, antisimétrica y transitiva.

- **Espacio ordenado de funciones:** $(Y,\ \leq_Y)$ poset $\Rightarrow (X \rightarrow Y,\ \leq)$ poset donde $f \leq g\ sii\ \forall x \in X. fx \leq gx$

- **Lifting:** $(X,\ \leq_X)$ poset $\Rightarrow (X_\perp,\ \leq)$ poset donde $x \leq y\ sii\ x \leq_X y \vee x = \perp$

  Ejemplo: $\mathbb{Z}_\perp$

$$\begin{array}{ccccccccc} \dots & -3 & -2 & -1 & 0 & 1 & 2 & 3 & \dots \\ & & \dots & \backslash & | & / & \dots & & \\ & & & & \perp & & & & \end{array}$$

- **Infinito:** $(X,\ \leq_X)$ poset $\Rightarrow (X^\infty,\ \leq)$ poset donde $x \leq y\ sii\ x \leq_X y \vee y = \infty$

  Ejemplo: $\mathbb{N}_\perp$

$$\begin{array}{c} \infty \\ \vdots \\ 3 \\ 2 \\ 1 \\ 0 \end{array}$$

- **Supremo:** Sea $Q \subseteq P$ donde $(P,\ \leq)$ poset, el supremo se define:

  - $\forall q \in Q. q \leq sup(Q)$
  - $\forall p \in P. (\forall q \in Q. q \leq p) \Rightarrow sup(Q) \leq p$

- **Cadenas:** $p_0 \leq p_1 \leq p_2 \dots$

  - Interesantes: si $\{p_0, p_1, p_2, \dots\}$ es infinita.
  - No interesantes: si $\{p_0, p_1, p_2, \dots\}$ es finita o repite infinitamente un elemento.

- **Predominios:** es un poset donde todas las cadenas (interesantes) tienen supremo.

  Si Y es predominio entonces $X \rightarrow Y$ también lo es.

- **Dominios:** es un predominio con elemento mínimo.

  Si D es dominio entonces $X \rightarrow D$ también lo es.

- **Monotonía:** Sean $(P, \leq_P)$ poset y $(Q, \leq_Q)$ poset, y $f \in P \to Q$, f es *monótona* si:

$$x \leq_P y \Rightarrow fx \leq_Q fy \qquad \text{(preserva orden)}$$

- **Continuidad:** Sean P, Q con $\leq_P, \leq_Q$ y sup_P, sup_Q predominios y $f \in P \to Q$, se dice que f es *continua* si preserva supremos de cadenas, es decir, si $p_0 \leq_P p_1 \ldots \leq_P p_n$ entonces el supremo $sup_Q(\{fp_i | i \in \mathbb{N}\})$ existe y $sup_Q(\{fp_i | i \in \mathbb{N}\}) = f sup_P(\{p_i | i \in \mathbb{N}\})$

- **Funciones Estrictas:** Sean D, D' dominios con $\perp, \perp'$ respectivamente. Se dice que la función $f \in D \to D'$ es *estricta* si f preserva el elemento mínimo, es decir, $f\perp = \perp'$.

PROPIEDADES:

- **Proposición 1:** Si f es monótona, f aplicada a los elementos de una cadena devuelve una cadena.

- **Proposición 2:** Si f es monótona, entonces f preserva el supremo de cadenas no interesantes.

- **Proposición 3:** Si la función $f \in P \Rightarrow Q$ entre predominios es monótona entonces $sup_Q(\{fp_i | i \in \mathbb{N}\})$ existe y $sup_Q(\{fp_i | i \in \mathbb{N}\}) \leq_Q f\ sup_Q(\{p_i | i \in \mathbb{N}\})$

- **Proposición 4:** Si f es continua, entonces f es monótona.

  La inversa, es decir, f monótona entonces f continua, solo vale para las cadenas no interesantes. Para las interesantes vale $sup_Q(\{fp_i | i \in \mathbb{N}\}) \leq_Q f sup_P(\{p_i | i \in \mathbb{N}\})$

- **Corolario:** Sean P, Q con $\leq_P, \leq_Q$ y sup_P, sup_Q predominios y $f \in P \to Q$ monótona, entonces f es continua sii si para toda cadena interesante $p_0 \leq_P p_1 \ldots \leq_P p_n \leq_P \ldots$, la desigualdad $f\ sup_P(\{p_i | i \in \mathbb{N}\}) \leq sup_Q(\{fp_i | i \in \mathbb{N}\})$ también vale.

TEOREMA DEL MENOR PUNTO FIJO

Teorema: Sea D un dominio, y $F \in D \to D$ continua, entonces $sup(F^i \perp)$ existe y es el menor punto fijo de F.
Prueba: Como $\perp$ es el elemento mínimo, $\perp \leq F \perp$. Como F es continua, F es monótona. Aplicando F a ambos lados obtenemos

$$F \perp \leq F\ (F \perp) = F^2 \perp$$

Iterando esto obtenemos $\perp \leq F \perp \leq F^2 \perp \leq F^3 \perp \leq \ldots$, es decir que $\{F^i \perp | i \in \mathbb{N}\}$ es una cadena y por lo tanto el supremo $x = \sup(\{F^i \perp | i \in \mathbb{N}\})$ existe.
Veamos que es punto fijo de F, es decir, que $F\ x = x$:

$$\begin{aligned} F\ x &= F\ \sup(\{F^i \perp | i \in \mathbb{N}\}) \\ &= \sup(\{F\ (F^i \perp) | i \in \mathbb{N}\}) \\ &= \sup(\{F^{i+1} \perp | i \in \mathbb{N}\}) \\ &= \sup(\{F^i \perp | i \in \mathbb{N}\}) \\ &= x \end{aligned}$$

Veamos que es el menor de ellos. Sea y punto fijo de F, es decir $F\ y = y$. Veamos que $x \leq y$. Claramente $\perp \leq y$ por ser elemento mínimo. Como F es monótona, se obtiene $F \perp \leq F\ y = y$. Iterando, obtenemos $F^i \perp \leq y$ para todo i. Es decir, y es cota superior de la cadena $\{F^i \perp | i \in \mathbb{N}\}$. Como el supremo es la menor de esas cotas,

$$\begin{aligned} x &= \sup(\{F^i \perp | i \in \mathbb{N}\}) \\ &\leq y \end{aligned}$$

4. Cálculo Lambda

$$
\begin{array}{lll}
\langle \mathsf{expr} \rangle ::= & & \text{término lambda, o expresión} \\
\quad \mid & \langle \mathsf{var} \rangle & \text{variable} \\
\quad \mid & \langle \mathsf{expr} \rangle \langle \mathsf{expr} \rangle & \text{aplicación, el primero es el operador y el segundo el operando} \\
\quad \mid & \lambda \langle \mathsf{var} \rangle . \langle \mathsf{expr} \rangle & \text{abstracción o expresión lambda}
\end{array}
$$

$$
\begin{array}{rcl}
FV(v) & = & \{v\} \\
FV(e_0 e_1) & = & FV(e_0) \cup FV(e_1) \\
FV(\lambda v.e) & = & FV(e) - \{v\}
\end{array}
$$

También se define sustitución:

$$
\begin{array}{rcl}
\Delta & = & \langle \mathsf{var} \rangle \to \langle \mathsf{expr} \rangle \\
\_/\_ & \in & \langle \mathsf{expr} \rangle \times \Delta \to \langle \mathsf{expr} \rangle \\
v/\delta & = & \delta v \\
(e_0 e_1)/\delta & = & (e_0/\delta)(e_1/\delta) \\
(\lambda v.e)/\delta & = & \lambda v'.(e/[\delta | v : v']) \\
& & \text{donde } v' \notin \bigcup\limits_{w \in FV(e) - \{v\}} FV(\delta w)
\end{array}
$$

Propiedad 1.

1. *si para todo $w \in FV(e)$, $\delta w = \delta' w$ entonces $(e/\delta) = (e/\delta')$*

2. *sea i la sustitución identidad, entonces $e/i = e$.*

3. $FV(e/\delta) = \bigcup_{w \in FV(e)} FV(\delta w)$

Definiciones:

- Expresión cerrada: sin variables libres

- Forma canónica: abstracciones $(\lambda x.e)$

- *Redex*: expresión de la forma $(\lambda v.e)e'$.

- Formal normal: expresión sin rédices.

  No necesariamente son abstracciones, por ejemplo, $(\lambda x.y)(\lambda z.z)$ β-contrae a y es es formal normal pero no canónica.

Renombre (α) La operación de cambiar una ocurrencia de la expresión lambda $\lambda v.e$ por $\lambda v'.(e/v \to v')$ donde $v' \notin FV(e) - \{v\}$ se llama renombre o cambio de variable ligada.

Contracción (β) Es la aplicación de una función $(\lambda v.e)$ a su argumento e'. Debe calcularse reemplazando las ocurrencias libres de v en e por e', es decir $(e/v \to e')$.

Teorema 1. *Church-Rosser. Si $e \to^* e_0$ y $e \to^* e_1$, entonces existe e' tal que $e_0 \to^* e'$ y $e_1 \to^* e'$.*

Corolario 1. *Salvo renombre, toda expresión tiene a lo sumo una forma normal.*

NO TODA EXPRESIÓN TINE FORMAL NORMAL (contraejemplos).

- $\Delta = (\lambda x.xx)$

- $\Delta' = (\lambda x.xxy)$

Propiedad 2. *Una aplicación cerrada no puede ser forma normal.*

- Si una expresión cerrada, es forma normal entonces es foerma canónica.

- Al revés no vale, contrajemplo $\lambda x.(\lambda y.y)x$.

4.1. Evaluación

En este tipo de semántica operacional se describen la relación entre los términos y sus valores, que también son términos, formaas canónicas. Llamamos $\Rightarrow$ a esta relación. Cuando decimos que $e \Rightarrow z$ se cumple, estamos diciendo que existe un árbol de derivación que prueba $e \Rightarrow z$.

Puede ocurrir que la evaluación eager no termine mientras que la normal si, por ejemplo: $(\lambda x.\lambda y.y)(\Delta\Delta) \Rightarrow_N \lambda y.y$, mientras que en eager diverge.

Regla η Un η-redex es una expresión de la forma $\lambda v.e\, v$ donde $v \notin FV(e)$. Gracias a la regla β, uno obtiene que $(\lambda v.e\, v)e'$ contrae a $e\, e'$ para toda expresión e'. Si uno asume que toda expresión lambda denota una función, $\lambda v.e\, v$ y e parecen comportarse extensionalmente igual: cuando se las aplica a e', ambas dan ee'. Esto motiva la regla η:

$$\frac{}{(\lambda v.ev) \to e} \text{ si } v \notin FV(e)$$

4.2. Semántica denotacional del cálculo lambda

Sea $D_\infty \cong [D_\infty \to D_\infty]$, donde $D_\infty \to D_\infty$ se refiere al espacio de funciones continuas donde todas tiene punto fijo, y sean:

$$\begin{array}{rl} \phi \in & D_\infty \to [D_\infty \to D_\infty] \\ \psi \in & [D_\infty \to D_\infty] \to D_\infty \end{array}$$

los isomorfismos tales que

$$\phi.\psi = id$$
$$\psi.\phi = id$$

$$\begin{array}{rcl} \text{Ambiente: } Env & = & \langle \mathsf{var} \rangle \to D_\infty \\ \eta & \in & Env \\ [\![\_]\!] & \in & \langle \mathsf{expr} \rangle \to Env \to D_\infty \end{array}$$

$$\begin{array}{rcl} [\![v]\!]\eta & = & \eta\, v \\ [\![e_0\, e_1]\!]\eta & = & \phi([\![e_0]\!]\eta)([\![e_1]\!]\eta) \\ [\![\lambda x.e]\!]\eta & = & \psi(\lambda d \in D_\infty.[\![e]\!][\eta|v : d]) \end{array}$$

Teorema 2. *Coincidencia Si $\eta\, w = \eta'\, w$ para todo $w \in FV(e)$, entonces $[\![e]\!]\eta = [\![e]\!]\eta'$.*

Teorema 3. *Sustitución Si $[\![\delta\, w]\!]\eta = \eta' w$ para todo $w \in FV(e)$, entonces $[\![e/\delta]\!]\eta = [\![e]\!]\eta'$.*

Teorema 4. *Sustitución Finita $[\![e/v_1 \to e_1, \ldots, v_n \to e_n]\!]\eta = [\![e]\!][\eta|v_1 : [\![e_1]\!]\eta| \ldots |v_n : [\![e_n]\!]\eta]$.*

Teorema 5. *Renombre Si $v' \notin FV(e) - \{v\}$, entonces $[\![\lambda v'.(e/v \to v')]\!] = [\![\lambda v.e]\!]$.*

Propiedad 3. *(correctitud de la regla β): $[\![(\lambda v.e)\, e']\!] = [\![e/v \to e']\!]$.*

Propiedad 4. *(correctitud de la regla η): Si $v \notin FV(e)$, entonces $[\![\lambda v.ev]\!] = [\![e]\!]$.*

Asumiremos que $[\![\Delta\Delta]\!]\eta = \perp$.

4.2.1. Normal

$$
\begin{aligned}
D &= V_\perp, \text{ donde } V \cong [D \to D] \\
\\
\phi &\in V \to [D \to D] \\
\psi &\in [D \to D] \to V \\
\\
\lambda d &\in D.\perp \text{ bottom de } D \to D \\
\psi(\lambda d &\in D.\perp) \text{ bottom de V pero no de D} \\
\\
\phi_{\perp\!\!\perp} &\in D \to [D \to D] \\
\iota_\perp.\psi &\in [D \to D] \to D \\
\\
Env &= \langle\mathsf{var}\rangle \to D \\
[\![\_]\!] &\in \langle\mathsf{expr}\rangle \to Env \to D \\
\\
[\![v]\!]\eta &= \eta v \\
[\![e_0\, e_1]\!]\eta &= \phi_{\perp\!\!\perp}([\![e_0]\!]\eta)([\![e_1]\!]\eta) \\
[\![\lambda x.e]\!]\eta &= (\iota_\perp.\psi)(\lambda d \in D.[\![e]\!][\eta|v:d])
\end{aligned}
$$

Los teoremas antes vistos junto con la regla β siguen valiendo pero no la regla η, pues $[\![\lambda y.\Delta\Delta]\!] = \psi(\lambda d \in D.\perp)$ mientras que $[\![\Delta\Delta]\!] = \perp$.

4.2.2. Eager

$$
\begin{aligned}
D &= V_\perp, \text{ donde } V \cong [V \to D] \\
\\
\phi &\in V \to [V \to D] \\
\psi &\in [V \to D] \to V \\
\\
\phi_{\perp\!\!\perp} &\in D \to [V \to D] \\
\iota_\perp.\psi &\in [V \to D] \to D \\
\\
Env &= \langle\mathsf{var}\rangle \to V \\
[\![\_]\!] &\in \langle\mathsf{expr}\rangle \to Env \to D \\
\\
[\![v]\!]\eta &= \iota_\perp(\eta v) \\
[\![e_0\, e_1]\!]\eta &= (\phi_{\perp\!\!\perp}([\![e_0]\!]\eta))_{\perp\!\!\perp}\,([\![e_1]\!]\eta) \\
[\![\lambda x.e]\!]\eta &= (\iota_\perp.\psi)(\lambda z \in V.[\![e]\!][\eta|x:z])
\end{aligned}
$$

Los teoremas antes vistos (salvo sustitución) siguen valiendo pero no las reglas β, η.

5. Gramáticas

$$
\begin{array}{rcll}
\langle \textsf{expr} \rangle & ::= & & \text{término lambda, o expresión} \\
& | & \langle \textsf{var} \rangle & \text{variable} \\
& | & \langle \textsf{expr} \rangle \langle \textsf{expr} \rangle & \text{aplicación, el primero es el operador y} \\
& | & \lambda \langle \textsf{var} \rangle . \langle \textsf{expr} \rangle & \text{abstracción o expresión lambda} \\
& | & \langle natconst \rangle \,|\, \langle boolconst \rangle & \\
& | & -\langle \textsf{expr} \rangle \mid \langle \textsf{expr} \rangle + \langle \textsf{expr} \rangle \mid \langle \textsf{expr} \rangle * \langle \textsf{expr} \rangle \mid \ldots & \text{operadores aritméticos} \\
& | & \langle \textsf{expr} \rangle \geq \langle \textsf{expr} \rangle \mid \langle \textsf{expr} \rangle \leq \langle \textsf{expr} \rangle \mid \langle \textsf{expr} \rangle < \langle \textsf{expr} \rangle \mid \ldots & \text{operadores relacionales} \\
& | & \langle \textsf{expr} \rangle \wedge \langle \textsf{expr} \rangle \mid \langle \textsf{expr} \rangle \vee \langle \textsf{expr} \rangle \mid \neg \langle \textsf{expr} \rangle & \text{operadores lógicos} \\
& | & \textbf{if } \langle \textsf{expr} \rangle \textbf{ then } \langle \textsf{expr} \rangle \textbf{ else } \langle \textsf{expr} \rangle & \\
& | & \textbf{error} \mid \textbf{typeerror} & \\
& | & \textbf{letrec} \langle \textsf{var} \rangle \equiv \lambda \langle \textsf{var} \rangle . \langle \textsf{expr} \rangle \textbf{ in } \langle \textsf{expr} \rangle & \text{(Eval. Eager)} \\
& | & \textbf{rec } \langle \textsf{expr} \rangle & \text{(Eval. Normal)} \\
\langle natconst \rangle & ::= & 0 \mid 1 \mid 2 \mid \ldots. & \\
\langle boolconst \rangle & ::= & \textbf{true} \mid \textbf{false} &
\end{array}
$$

Formas canónicas

$$
\begin{array}{lcl}
\langle cnf \rangle & ::= & \langle intcnf \rangle \mid \langle boolcnf \rangle \mid \langle funcnf \rangle \mid \langle tuplecnf \rangle \\
\langle intcnf \rangle & ::= & \ldots \mid -2 \mid -1 \mid 0 \mid 1 \mid 2 \mid \ldots \\
\langle boolcnf \rangle & ::= & \langle boolconst \rangle \\
\langle funcnf \rangle & ::= & \lambda \langle \textsf{var} \rangle . \langle \textsf{expr} \rangle \\
\langle tuplecnf \rangle & ::= & \langle \langle cnf \rangle, \ldots, \langle cnf \rangle \rangle
\end{array}
$$

6. Lenguajes Aplicativos

7. Evaluación

7.1. Compartido entre Normal y Eager

Para todas las formas canónicas vale: $\overline{z \Rightarrow z}$

- **Aritméticos y relacionales:** $\oplus \in \{+, -, *, =, \neq, \leq, \geq, \ldots\}$, $\oslash \in \{/, mod\}$
  - $\dfrac{e \Rightarrow \lfloor i \rfloor}{-e \Rightarrow \lfloor -i \rfloor}$
  - $\dfrac{e \Rightarrow \lfloor i \rfloor \qquad e' \Rightarrow \lfloor i' \rfloor}{e \oplus e' \Rightarrow \lfloor i \oplus i' \rfloor}$
  - $\dfrac{e \Rightarrow \lfloor i \rfloor \qquad e' \Rightarrow \lfloor i' \rfloor}{e \oslash e' \Rightarrow \lfloor i \oslash i' \rfloor}$

    $i' \neq 0$
- **Booleanos:**
  - $\dfrac{e \Rightarrow \lfloor b \rfloor}{\neg e \Rightarrow \lfloor \neg b \rfloor}$
  - $\dfrac{e \Rightarrow \textbf{true} \qquad e_0 \Rightarrow z}{\textbf{if } e \textbf{ then } e_0 \textbf{ else } e_1 \Rightarrow z}$
  - $\dfrac{e \Rightarrow \textbf{false} \qquad e_1 \Rightarrow z}{\textbf{if } e \textbf{ then } e_0 \textbf{ else } e_1 \Rightarrow z}$
- **Tuplas:**
- **Recursión:**

7.2. Normal

- **Funciones:**

$$\frac{e \Rightarrow \lambda v.e'' \qquad (e''/v \to e') \Rightarrow z}{ee' \Rightarrow z} \text{(aplicación)}$$

- **Booleanos:**

- $e \wedge e' = \textbf{if } e \textbf{ then } e' \textbf{ else } false$
- $e \vee e' = \textbf{if } e \textbf{ then } true \textbf{ else } e'$
- $e \Rightarrow e' = \textbf{if } e \textbf{ then } e' \textbf{ else } true$

- **Tuplas:**

$$\frac{}{\langle e_1, ..., e_n \rangle \Rightarrow \langle e_1, ..., e_n \rangle} \qquad \frac{e \Rightarrow \langle e_1, ..., e_n \rangle \qquad e_k \Rightarrow z}{e.\lfloor k \rfloor \Rightarrow \ z} \qquad k \leq n$$

- **Recursión:**

$$\frac{e\ (\textbf{rec}\ e) \Rightarrow z}{\textbf{rec}\ e \Rightarrow \ z}$$

7.3. Eager

- **Funciones:**

$$\frac{e \Rightarrow \lambda v.e'' \qquad e' \Rightarrow z' \qquad (e''/v \to z') \Rightarrow z}{ee' \Rightarrow z} \text{(aplicación)}$$

- **Booleanos:** $\oslash \in \{\wedge, \vee, \Rightarrow, ...\}$

$$\frac{e \ \Rightarrow \ \lfloor b \rfloor \qquad e' \ \Rightarrow \ \lfloor b' \rfloor}{e \oslash e' \ \Rightarrow \ \lfloor b \oslash b' \rfloor}$$

- **Tuplas:**

$$\frac{e_1 \Rightarrow z_1 \quad ... \quad e_n \Rightarrow z_n}{\langle e_1, ..., e_n \rangle \ \Rightarrow \ \langle z_1, ..., z_n \rangle} \qquad \frac{e \Rightarrow \langle z_1, ..., z_n \rangle}{e.\lfloor k \rfloor \Rightarrow \ z_k} \ k \leq n$$

- **Recursión:**

$$\frac{e/(f \mapsto \lambda v.\textbf{letrec}\ f \equiv \lambda v.e'\ \textbf{in}\ e'\ ) \Rightarrow z}{\textbf{letrec}\ f \equiv \lambda v.e'\ \textbf{in}\ e \ \Rightarrow \ z} f \neq v$$

8. Semántica denotacional

8.1. Compartido entre Normal y Eager

V: es el conjunto de valores, de la misma forma que a las formas canónicas las llamamos también valores en su momento.
D: es el conjunto de resultados, que incluyen valores y otras cosas, en este caso, sólo se agrega bottom.

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

$$\begin{aligned} V &\simeq V_{int} + V_{bool} + V_{fun} + V_{tuple} \\ \phi &\in V \to V_{int} + V_{bool} + V_{fun} + V_{tuple} \\ \psi &\in V_{int} + V_{bool} + V_{fun} + V_{tuple} \to V \end{aligned}$$

Donde: $V_{int} = \mathbb{Z}, V_{bool} = \mathbb{B}$ y V_{fun}, V_{tuple} se definen para cada semántica en particular.

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

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

Si $\ell \in \{int, bool, fun, tuple\}, f \in V_\ell \to D$ entonces f_ℓ se define:

$$\begin{aligned} f_\ell(\iota_\ell\ z) &= f\ z \\ f_\ell(\iota_{\ell'}\ z) &= tyerr, \text{ si } \ell \neq \ell' \end{aligned}$$

$$
\begin{aligned}
\iota_{int} &= \psi.\iota_0 \in V_{int} \to V \\
\iota_{bool} &= \psi.\iota_1 \in V_{bool} \to V \\
\iota_{fun} &= \psi.\iota_2 \in V_{fun} \to V \\
\iota_{tuple} &= \psi.\iota_3 \in V_{tuple} \to V
\end{aligned}
$$

$$
\begin{aligned}
[\![0]\!]\eta &= \iota_{norm}(\iota_{int}0) \\
[\![\mathbf{true}]\!]\eta &= \iota_{norm}(\iota_{bool}T) \\
\\
[\![-e]\!]\eta &= (\lambda i \in V_{int}.\iota_{norm}(\iota_{int} - i))_{int*}([\![e]\!]\eta) \\
[\![\neg e]\!]\eta &= (\lambda b \in V_{bool}.\iota_{norm}(\iota_{bool}\neg b))_{bool*}([\![e]\!]\eta) \\
\\
[\![e_0 + e_1]\!]\eta &= (\lambda i \in V_{int}.\ (\lambda j \in V_{int}.\iota_{norm}(\iota_{int} i + j)_{int*}([\![e_1]\!]\eta))_{int*}([\![e_0]\!]\eta) \\
[\![e_0 / e_1']\!]\eta &= (\lambda i \in V_{int}.\ (\lambda j \in V_{int}.\ \left\{ \begin{array}{ll} err & j = 0 \\ \iota_{norm}(\iota_{int} i/j) & c.c \end{array} \right)_{int*} ([\![e_1]\!]\eta))_{int*}([\![e_0]\!]\eta) \\
[\![\mathbf{if}\ e\ \mathbf{then}\ e_0\ \mathbf{else}\ e_1]\!]\eta &= (\lambda b \in V_{bool}.\ \left\{ \begin{array}{ll} [\![e_0]\!] & sib \\ [\![e_1]\!] & c.c \end{array} \right)_{bool*} ([\![e]\!]\eta) \\
\\
[\![error]\!]\eta &= err \\
[\![typeerror]\!]\eta &= tyerr \\
\\
[\![e.[k]]\!]\eta &= (\lambda t \in V_{tuple}.\ \left\{ \begin{array}{ll} \iota_{norm} t.k & si\ k \leq \#t \\ tyerr & c.c \end{array} \right)_{tuple*} ([\![e]\!]\eta)
\end{aligned}
$$

8.2. Normal

$V_{fun} = D \to D \qquad V_{tuple} = D^*$

$$
\begin{aligned}
Env &= \langle \mathsf{var} \rangle \to D \\
[\![\_]\!] &\in \langle \mathsf{expr} \rangle \to Env \to D \\
\\
[\![v]\!]\eta &= \eta v \\
[\![e_0\, e_1]\!]\eta &= (\lambda f \in V_{fun}.f([\![e_1]\!]\eta))_{fun*}([\![e_0]\!]\eta) \\
[\![\lambda x.e]\!]\eta &= \iota_{norm}(\iota_{fun}(\lambda d \in D.[\![e]\!][\eta|x:d])) \\
\\
[\![\langle e_1, ..., e_n \rangle]\!]\eta &= \iota_{norm}(\iota_{tuple} \langle [\![e_1]\!]\eta, ..., [\![e_n]\!]\eta \rangle) \\
[\![\mathbf{rec}\ e]\!]\eta &= (\lambda f \in V_{fun}.\ Y f)_{fun*}([\![e]\!]\eta) \\
\text{donde} & \quad Y \text{ es el operador de menor punto fijo}
\end{aligned}
$$

8.3. Eager

$V_{fun} = V \to D \qquad V_{tuple} = V^*$

$$\begin{array}{rcl} Env & = & \langle \mathsf{var} \rangle \to V \\ [\![\_]\!] & \in & \langle \mathsf{expr} \rangle \to Env \to D \\ \\ [\![v]\!]\eta & = & \iota_{norm}(\eta v) \\ [\![e_0\, e_1]\!]\eta & = & (\lambda f \in V_{fun}. f_*([\![e_1]\!]\eta))_{fun*}([\![e_0]\!]\eta) \\ [\![\lambda x.e]\!]\eta & = & \iota_{norm}(\iota_{fun}(\lambda z \in V.[\![e]\!][\eta|x:z])) \\ \\ [\![\langle e_1, ..., e_n\rangle]\!]\eta & = & (\lambda z_1 \in V....(\lambda z_n \in V.\iota_{norm}(\iota_{tuple}\langle z_1, ..., z_n\rangle))_*([\![e_n]\!]\eta)...)_*([\![e_1]\!]\eta) \\ [\![\textbf{letrec } v \equiv \lambda u.e \textbf{ in } e'\ ]\!]\eta & = & [\![e']\!][\eta|v : \iota_{fun} Y_{V_{fun}} F] \\ \text{donde} & & F\ f\ z = [\![e]\!][\eta|v : \iota_{fun} f|u : z] \\ & & Y_{V_{fun}} \text{ es el operador de menor punto fijo sobre } V_{fun} \end{array}$$

9. El lenguaje Iswin

Componente imperativa a un lenguaje aplicativo eager.

$$\langle \mathsf{expr} \rangle ::= \textbf{skip} \mid \textbf{ref}\ \langle \mathsf{expr} \rangle \mid \textbf{val}\ \langle \mathsf{expr} \rangle \mid \langle \mathsf{expr} \rangle := \langle \mathsf{expr} \rangle \mid \langle \mathsf{expr} \rangle =_{ref} \langle \mathsf{expr} \rangle$$

Definiciones:

- Rf: conjunto infinito de referencias
- $new(\sigma) \in Rf$, devuelve una referencia nueva tal que $new(\sigma) \notin dom(\sigma)$
- V = $V_{int} + V_{bool} + V_{fun} + V_{tuple} + V_{ref}$
- D = $(\Sigma \times V + \{error, typeerror\})_\perp$
- $\iota_{norm}\langle \sigma, z\rangle \in D$, $\sigma \in Sigma, z \in V$

$$\begin{array}{lcllcl} V_{int} & = & \mathbf{Z} & \iota_{int} & \in & V_{int} \to V \\ V_{bool} & = & \mathbf{B} & \iota_{bool} & \in & V_{bool} \to V \\ V_{fun} & = & \Sigma \times V \to D & \iota_{fun} & \in & V_{fun} \to V \\ V_{tuple} & = & V^* & \iota_{tuple} & \in & V_{tuple} \to V \\ V_{ref} & = & Rf & \iota_{ref} & \in & V_{ref} \to V \end{array}$$

- Si $f \in \Sigma \times V \to D$, entonces $f_* \in \Sigma \times D \to D$ se define:

$$\begin{array}{rcl} f_*\ \iota_{norm}\langle \sigma, z\rangle & = & f\langle \sigma, z\rangle \\ f_*\ err & = & err \\ f_*\ tyerr & = & tyerr \\ f_*\ \perp & = & \perp \end{array}$$

- Si $f \in \Sigma \times V_{int} \to D$, entonces $f_{int} \in \Sigma \times V \to D$ se define:

$$\begin{array}{rcll} f_{int}\langle \sigma, \iota_{int} k\rangle & = & f\langle \sigma, k\rangle & \\ f_{int}\langle \sigma, \iota_\theta z\rangle & = & tyerr & \qquad (\theta \neq int) \end{array}$$

SEMÁNTICA

$$
\begin{aligned}
[\![\mathbf{val}\ e]\!]\eta\sigma &= (\lambda \langle\sigma', r\rangle .\ \in \Sigma \times V_{ref}.\ \left\{\begin{array}{ll} \iota_{norm}\langle\sigma', \sigma' r\rangle & r \in dom(\sigma') \\ err & c.c \end{array}\right)_{ref*} ([\![e]\!]\eta\sigma) \\
[\![\mathbf{ref}\ e]\!]\eta\sigma &= (\lambda \langle\sigma', r\rangle \in \Sigma \times V.\ \iota_{norm}\langle[\sigma'|r:z], \iota_{ref} r\rangle)_*([\![e]\!]\eta\sigma) \text{ donde } r = new(\sigma') \\
[\![e := e']\!]\eta\sigma &= (\lambda \langle\sigma', r\rangle \in \Sigma \times V_{ref}.\ (\lambda \langle\sigma'', z\rangle \in \Sigma \times V.\ \iota_{norm}\langle[\sigma''|r:z], z\rangle)_*([\![e']\!]\eta\sigma'))_{ref*}([\![e]\!]\eta\sigma) \\
[\![e =_{ref} e']\!]\eta\sigma &= (\lambda \langle\sigma', r\rangle \in \Sigma \times V_{ref}.\ (\lambda \langle\sigma'', z\rangle \in \Sigma \times V_{ref}.\ \iota_{norm}\langle\sigma'', \iota_{bool} r = r'\rangle)_{ref*}([\![e']\!]\eta\sigma'))_{ref*}([\![e]\!]\eta\sigma)
\end{aligned}
$$

$$
\begin{aligned}
[\![0]\!]\eta\sigma &= \iota_{norm}\langle\sigma, \iota_{int} 0\rangle \\
[\![\mathbf{true}]\!]\eta\sigma &= \iota_{norm}\langle\sigma, \iota_{bool} T\rangle \\
[\![-e]\!]\eta\sigma &= (\lambda \langle\sigma', i\rangle .\ \iota_{norm}\langle\sigma', \iota_{int} - i\rangle)_{int*}([\![e]\!]\eta\sigma) \\
[\![\neg e]\!]\eta\sigma &= (\lambda \langle\sigma', b\rangle .\ \iota_{norm}\langle\sigma', \iota_{bool} \neg b\rangle)_{bool*}([\![e]\!]\eta\sigma) \\
[\![e + e']\!]\eta\sigma &= (\lambda \langle\sigma', i\rangle \in \Sigma \times V_{int}.(\lambda \langle\sigma'', j\rangle \in \Sigma \times V_{int}.\ \iota_{norm}\langle\sigma'', \iota_{int} i + j\rangle)_{int*}([\![e']\!]\eta\sigma'))_{int*}([\![e]\!]\eta\sigma)
\end{aligned}
$$

$$
\begin{aligned}
[\![ee']\!]\eta\sigma &= (\lambda \langle\sigma', f\rangle \in \Sigma \times V_{fun}.\ f_*([\![e']\!]\eta\sigma'))_{fun*}([\![e]\!]\eta\sigma) \\
[\![\lambda v.\ e']\!]\eta\sigma &= \iota_{norm}\langle\sigma, \iota_{fun}(\lambda \langle\sigma', z\rangle \in \Sigma \times V.\ [\![e]\!][\eta|v:z]\sigma')\rangle \\
[\![\mathbf{letrec}\ w &= \lambda v.\ e\ \mathbf{in}\ e']\!]\eta\sigma \;=\; [\![e']\!][\eta|w:\iota_{fun} f]\sigma \\
\text{donde} & \\
f &= \mathbf{Y}_{V_{fun}} F \\
F\ f\ \langle\sigma', z\rangle &= [\![e]\!][\eta|w:\iota_{fun} f|v:z]\sigma'
\end{aligned}
$$

9.1. Semántica operacional de Iswim

$$
\frac{\sigma, e \Rightarrow \lambda v.\ e_0, \sigma' \quad \sigma', e' \Rightarrow z', \sigma'' \quad \sigma'', (e_0/v \to z') \Rightarrow z, \sigma'''}{ee', \sigma \Rightarrow z, \sigma'''}
$$

$$
\frac{\sigma, e \Rightarrow r, \sigma' \quad \sigma', e' \Rightarrow z, \sigma''}{\sigma, e := e' \Rightarrow z, [\sigma''|r:z]}
$$

$$
\frac{\sigma, e \Rightarrow z, \sigma'}{\sigma, \mathbf{ref}\ e \Rightarrow r, [\sigma'|r:z]}
$$

$$
\frac{\sigma, e \Rightarrow r, \sigma'}{\sigma, \mathbf{val}\ e \Rightarrow \sigma' r, \sigma'}
$$

$$
\frac{\sigma, e \Rightarrow r, \sigma' \quad \sigma', e' \Rightarrow r', \sigma''}{\sigma, e =_{ref} e' \Rightarrow \lfloor r = r' \rfloor, \sigma''}
$$

9.2. Algunas propiedades del fragmento imperativo

- $e; e' \ =_{def}$ **let** $v = e$ **in** e' $\qquad (v \notin FV\ e')$

$$
\frac{\sigma, e \Rightarrow z, \sigma' \quad \sigma', e' \Rightarrow z', \sigma''}{e; e', s \Rightarrow z', \sigma''}
$$

$$
[\![e; e']\!]\eta\sigma = (\lambda \langle\sigma', z\rangle .\ [\![e']\!]\eta\sigma')_*([\![e]\!]\eta\sigma)
$$

- **newvar** v **in** $= e$ **in** e' $=_{def}$ **let** $v =$ **ref** e **in** e'

$$
\frac{\sigma, e \Rightarrow z, \sigma' \quad [\sigma'|r:z], (e'/v \mapsto r) \Rightarrow z', \sigma''}{\sigma, \mathbf{newvar}\ v\ \mathbf{in}\ := e\ \mathbf{in}\ e' \Rightarrow z', \sigma''} \quad (r = new(dom\ \sigma'))
$$

$$
[\![\mathbf{newvar}\ v\ \mathbf{in}\ = e\ \mathbf{in}\ e']\!]\eta\sigma = (\lambda \langle\sigma', z\rangle .[\![e']\!][\eta|v:\iota_{ref} r][\sigma'|r:z])_*([\![e]\!]\eta\sigma) \quad \text{donde } r = new(dom\ \sigma')
$$

- $\textbf{while}\ e\ \textbf{do}\ \ \textbf{do}\ e' \ =_{def}\ \textbf{letrec}\ w = \lambda v.\ \textbf{if}\ e\ \textbf{then}\ e'; w\ \textbf{skip}\ \textbf{else}\ \textbf{skip}\ \ \textbf{in}\ w\ \textbf{skip}$

  donde w y v no deben ocurrir en e ni e'

$$\frac{\sigma, e \Rightarrow \textbf{false}, \sigma'}{\textbf{while}\ e\ \textbf{do}\ \ \textbf{do}\ e', s \Rightarrow \langle\rangle, \sigma'} \qquad \frac{\sigma, e \Rightarrow \textbf{true}, \sigma' \quad e'; \textbf{while}\ e\ \textbf{do}\ \ \textbf{do}\ \sigma', e' \Rightarrow z', \sigma''}{\textbf{while}\ e\ \textbf{do}\ \ \textbf{do}\ e', s \Rightarrow z', \sigma''}$$