Programación Orientada a Objetos

Departamento de Computación, FCEyN, UBA

Cálculo de Objetos no tipado (ς cálculo) [Abadi&Cardelli,98]

Cálculo de Objetos no tipado (ς cálculo) [Abadi&Cardelli,98]

Ingredientes

- Objetos como única estructura computacional.
- Los objetos son una colección de atributos nombrados (registros).
- Todos los atributos son métodos.
- Cada método tiene una única variable ligada que representa a `self` (`this`) y un cuerpo que produce un resultado.
- Los objetos proveen dos operaciones:
  - envío de mensaje (invocación de un método)
  - redefinición de un método

Sintaxis

$$
\begin{array}{llll}
o, b & ::= & x & \textit{variable} \\
 & | & [l_i = \varsigma(x_i)b_i{}^{i\in 1..n}] & \textit{objeto} \\
 & | & o.l & \textit{selección / envío de mensaje} \\
 & | & o.l \Leftarrow \varsigma(x)b & \textit{redefinición de método}
\end{array}
$$

Sintaxis

o, b	$::=$	x	*variable*
	$\mid$	$[l_i = \varsigma(x_i)b_i{}^{i \in 1..n}]$	*objeto*
	$\mid$	$o.l$	*selección / envío de mensaje*
	$\mid$	$o.l \Leftarrow \varsigma(x)b$	*redefinición de método*

Un objeto

$$o \stackrel{\text{def}}{=} [l_1 = \varsigma(x_1)[\ ],\ l_2 = \varsigma(x_2)x_2.l_1]$$

- o tiene dos métodos:
  - l_1 retorna un objeto vacío $[\ ]$.
  - l_2 es un método que envía el mensaje l_1 a `self` (representado por el parámetro x_2).

Atributos vs métodos

- el cálculo ς no incluye explícitamente atributos (campos/fields).
- los atributos se representan como métodos que no utilizan al parámetro `self`. Por ejemplo, l_1 en

$$o \stackrel{\text{def}}{=} [l_1 = \varsigma(x_1)[\ ],\ l_2 = \varsigma(x_2)x_2.l_1]$$

- De esta manera,
  - el envío de un mensaje representa también a la selección de un atributo
  - la redefinición de un método representa también a la asignación de un atributo

Notación

- Atributo: $[\ldots, l = b, \ldots]$ es una abreviatura de $[\ldots, l = \varsigma(x)b, \ldots]$ cuando x no se usa en b.
- Asignación de atributo: $o.l := b$ denota $o.l \Leftarrow \varsigma(x)b$ cuando x no se usa en b

Notación

- Atributo: $[\ldots, l = b, \ldots]$ es una abreviatura de $[\ldots, l = \varsigma(x)b, \ldots]$ cuando x no se usa en b.
- Asignación de atributo: $o.l := b$ denota $o.l \Leftarrow \varsigma(x)b$ cuando x no se usa en b

Escribimos

$$o \stackrel{\text{def}}{=} [l_1 = [\ ],\ l_2 = \varsigma(x_2)x_2.l_1]$$

en lugar de

$$o \stackrel{\text{def}}{=} [l_1 = \varsigma(x_1)[\ ],\ l_2 = \varsigma(x_2)x_2.l_1]$$

Sintaxis - Variables libres

ς es un ligador para el parámetro `self` x_i en el cuerpo b_i de la expresión $\varsigma(x_i)b_i$

Definición

$$\mathsf{fv}(\varsigma(x)b) \quad = $$

Sintaxis - Variables libres

ς es un ligador para el parámetro `self` x_i en el cuerpo b_i de la expresión $\varsigma(x_i)b_i$

Definición

$$
\begin{aligned}
\mathsf{fv}(\varsigma(x)b) &= \mathsf{fv}(b) \setminus \{x\} \\
\mathsf{fv}(x) &=
\end{aligned}
$$

Sintaxis - Variables libres

ς es un ligador para el parámetro `self` x_i en el cuerpo b_i de la expresión $\varsigma(x_i)b_i$

Definición

$$
\begin{array}{ll}
\mathsf{fv}(\varsigma(x)b) & = \mathsf{fv}(b) \setminus \{x\} \\
\mathsf{fv}(x) & = \{x\} \\
\mathsf{fv}([l_i = \varsigma(x_i)b_i{}^{i \in 1..n}]) & =
\end{array}
$$

Sintaxis - Variables libres

ς es un ligador para el parámetro `self` x_i en el cuerpo b_i de la expresión $\varsigma(x_i)b_i$

Definición

$$
\begin{array}{lcl}
\mathsf{fv}(\varsigma(x)b) & = & \mathsf{fv}(b) \setminus \{x\} \\
\mathsf{fv}(x) & = & \{x\} \\
\mathsf{fv}([l_i = \varsigma(x_i)b_i{}^{i \in 1..n}]) & = & \bigcup^{i \in 1..n} \mathsf{fv}(\varsigma(x_i)b_i) \\
\mathsf{fv}(o.l) & = &
\end{array}
$$

Sintaxis - Variables libres

ς es un ligador para el parámetro `self` x_i en el cuerpo b_i de la expresión $\varsigma(x_i)b_i$

Definición

$$
\begin{aligned}
\mathsf{fv}(\varsigma(x)b) &= \mathsf{fv}(b) \setminus \{x\} \\
\mathsf{fv}(x) &= \{x\} \\
\mathsf{fv}([l_i = \varsigma(x_i)b_i{}^{i \in 1..n}]) &= \bigcup^{i \in 1..n} \mathsf{fv}(\varsigma(x_i)b_i) \\
\mathsf{fv}(o.l) &= \mathsf{fv}(o) \\
\mathsf{fv}(o.l \Leftarrow \varsigma(x)b) &=
\end{aligned}
$$

Sintaxis - Variables libres

ς es un ligador para el parámetro `self` x_i en el cuerpo b_i de la expresión $\varsigma(x_i)b_i$

Definición

$$
\begin{array}{ll}
\mathsf{fv}(\varsigma(x)b) & = \mathsf{fv}(b) \setminus \{x\} \\
\mathsf{fv}(x) & = \{x\} \\
\mathsf{fv}([l_i = \varsigma(x_i)b_i{}^{i\in 1..n}]) & = \bigcup^{i\in 1..n} \mathsf{fv}(\varsigma(x_i)b_i) \\
\mathsf{fv}(o.l) & = \mathsf{fv}(o) \\
\mathsf{fv}(o.l \Leftarrow \varsigma(x)b) & = \mathsf{fv}(o.l) \cup \mathsf{fv}(\varsigma(x)b)
\end{array}
$$

Un término o es cerrado si $\mathsf{fv}(o) = \emptyset$.

Sustitución

$$x\{c/x\} \quad = $$

Sustitución

$$x\{c/x\} = c$$

$$y\{c/x\} =$$

Sustitución

$$
\begin{aligned}
&x\{c/x\} &&= c \\
&y\{c/x\} &&= y && \mathit{if}\ x \neq y \\
&([l_i = \varsigma(x_i)b_i^{i \in 1..n}])\{c/x\} &&=
\end{aligned}
$$

Sustitución

$$
\begin{aligned}
x\{c/x\} &= c \\
y\{c/x\} &= y && if\ x \neq y \\
([l_i = \varsigma(x_i)b_i^{i\in 1..n}])\{c/x\} &= [l_i = (\varsigma(x_i)b_i)\{c/x\}^{i\in 1..n}] \\
(o.l)\{c/x\} &=
\end{aligned}
$$

Sustitución

$$
\begin{array}{lll}
x\{c/x\} & = c & \\
y\{c/x\} & = y & if\ \ x \neq y \\
([l_i = \varsigma(x_i)b_i^{i\in 1..n}])\{c/x\} & = [l_i = (\varsigma(x_i)b_i)\{c/x\}^{i\in 1..n}] & \\
(o.l)\{c/x\} & = (o\{c/x\}).l & \\
(o.l \Leftarrow \varsigma(y)b)\{c/x\} & = &
\end{array}
$$

Sustitución

$$
\begin{aligned}
x\{c/x\} &= c \\
y\{c/x\} &= y && \mathit{if}\ \ x \neq y \\
([l_i = \varsigma(x_i)b_i^{i\in 1..n}])\{c/x\} &= [l_i = (\varsigma(x_i)b_i)\{c/x\}^{i\in 1..n}] \\
(o.l)\{c/x\} &= (o\{c/x\}).l \\
(o.l \Leftarrow \varsigma(y)b)\{c/x\} &= (o\{c/x\}).l \Leftarrow ((\varsigma(y)b)\{c/x\}) \\
(\varsigma(y)b)\{c/x\} &=
\end{aligned}
$$

Sustitución

$$
\begin{array}{lll}
x\{c/x\} & = c & \\
y\{c/x\} & = y & \mathit{if}\ x \neq y \\
([l_i = \varsigma(x_i)b_i^{i\in 1..n}])\{c/x\} & = [l_i = (\varsigma(x_i)b_i)\{c/x\}^{i\in 1..n}] & \\
(o.l)\{c/x\} & = (o\{c/x\}).l & \\
(o.l \Leftarrow \varsigma(y)b)\{c/x\} & = (o\{c/x\}).l \Leftarrow ((\varsigma(y)b)\{c/x\}) & \\
(\varsigma(y)b)\{c/x\} & = \varsigma(y')(b\{y'/y\}\{c/x\}) & \\
 & \qquad \mathit{if}\ y' \notin \mathsf{fv}(\varsigma(y)b) \cup \mathsf{fv}(c) \cup \{x\} &
\end{array}
$$

Equivalencia de términos ($\equiv$)

- Los términos $\varsigma(x)b$ y $\varsigma(y)(b\{y/x\})$ con $y \notin \mathsf{fv}(b)$ se consideran equivalentes (α-conversión).
- Dos objetos que difieren en el orden de sus componentes son considerados equivalentes.
- Por ejemplo,

$$
\begin{array}{l}
o_1 \stackrel{\text{def}}{=} [l_1 = \varsigma(x_1)[\ ],\ l_2 = \varsigma(x_2)x_2.l_1] \\
o_2 \stackrel{\text{def}}{=} [l_2 = \varsigma(x_3)x_3.l_1,\ l_1 = \varsigma(x_1)[\ ]]
\end{array}
$$

son equivalentes ($o_1 \equiv o_2$).

Semántica operacional

Valores

$$v ::= [l_i = \varsigma(x_i)b_i{}^{i\in 1..n}]$$

Semántica operacional

Valores

$$v ::= [l_i = \varsigma(x_i)b_i{}^{i\in 1..n}]$$

Reducción *big-step* $\longrightarrow$

Semántica operacional

Valores

$$v ::= [l_i = \varsigma(x_i)b_i{}^{i\in 1..n}]$$

Reducción *big-step* $\longrightarrow$

$$\frac{o \longrightarrow v' \quad v' \equiv [l_i = \varsigma(x_i)b_i{}^{i\in 1..n}] \quad b_j\{v'/x_j\} \longrightarrow v \quad j \in 1..n}{o.l_j \longrightarrow v}\ [\textsc{Sel}]$$

Semántica operacional

Valores

$$v ::= [l_i = \varsigma(x_i)b_i{}^{i\in 1..n}]$$

Reducción *big-step* $\longrightarrow$

$$\frac{}{v \longrightarrow v}\ [\textsc{Obj}]$$

$$\frac{o \longrightarrow v' \quad v' \equiv [l_i = \varsigma(x_i)b_i{}^{i\in 1..n}] \quad b_j\{v'/x_j\} \longrightarrow v \quad j \in 1..n}{o.l_j \longrightarrow v}\ [\textsc{Sel}]$$

Semántica operacional

Valores

$$v ::= [l_i = \varsigma(x_i)b_i{}^{i \in 1..n}]$$

Reducción *big-step* $\longrightarrow$

$$\frac{}{v \longrightarrow v}\ [\textsc{Obj}]$$

$$\frac{o \longrightarrow v' \quad v' \equiv [l_i = \varsigma(x_i)b_i{}^{i \in 1..n}] \quad b_j\{v'/x_j\} \longrightarrow v \quad j \in 1..n}{o.l_j \longrightarrow v}\ [\textsc{Sel}]$$

$$\frac{o \longrightarrow [l_i = \varsigma(x_i)b_i{}^{i \in 1..n}] \qquad j \in 1..n}{o.l_j \Leftarrow \varsigma(x)b \longrightarrow [l_j = \varsigma(x)b,\ l_i = \varsigma(x_i)b_i{}^{i \in 1..n - \{j\}}]}\ [\textsc{Upd}]$$

Ejemplos $[a = [\ ], l = \varsigma(x)x.a].l \longrightarrow$ ...

Ejemplos $[a = [\,], l = \varsigma(x)x.a].l \longrightarrow \ldots$

$$
\dfrac{o \longrightarrow o \qquad \dfrac{o \longrightarrow o \qquad \dfrac{}{\overbrace{[\,]\{o/x\}}^{=[\,]} \longrightarrow [\,]}\,[\textsc{Obj}]}{\overbrace{(x.a)\{o/x\}}^{=o.a} \longrightarrow [\,]}\,[\textsc{Sel}]}{\underbrace{[a = [\,], l = \varsigma(x)x.a]}_{o}.l \longrightarrow []}\,[\textsc{Sel}]
$$

Ejemplos $([a = [\ ], l = \varsigma(x)x.a].l \Leftarrow \varsigma(y)[\ ]).l \longrightarrow$ ...

Ejemplos $([a=[\,],l=\varsigma(x)x.a].l \Leftarrow \varsigma(y)[\,]).l \longrightarrow \ldots$

$$\dfrac{\dfrac{\dfrac{}{o \longrightarrow o}\,[\textsc{Obj}]}{u \longrightarrow [a=[\,],l=[\,]]}\,[\textsc{Upd}] \qquad \dfrac{}{\overbrace{[\,]\{u/x\}}^{=[\,]} \longrightarrow [\,]}\,[\textsc{Obj}]}{\underbrace{(\underbrace{[a=[\,],l=\varsigma(x)x.a]}_{o}.l \Leftarrow \varsigma(y)[\,])}_{u}.l \longrightarrow [\,]}\,[\textsc{Sel}]$$

Ejemplos $[a = \varsigma(x)x.a].a \longrightarrow$ ...

Ejemplos $[a = \varsigma(x)x.a].a \longrightarrow \ldots$

$$
\dfrac{\dfrac{}{o \longrightarrow o}\,[\textsc{Obj}] \qquad \dfrac{\dfrac{\vdots}{}\,[\textsc{Sel}]}{\overbrace{x.a\{o/x\}}^{=o.a} \longrightarrow}\,[\textsc{Sel}]}{\underbrace{[a = \varsigma(x)x.a]}_{o}.a \longrightarrow}
$$

Ejemplos $[a = \varsigma(x)x.a].a \longrightarrow$ ...

$$
\dfrac{
  \dfrac{}{o \longrightarrow o}\,[\textsc{Obj}]
  \qquad
  \dfrac{\vdots}{\overbrace{x.a\{o/x\}}^{=o.a} \longrightarrow}\,[\textsc{Sel}]
}{
  \underbrace{[a = \varsigma(x)x.a]}_{o}.a \longrightarrow
}\,[\textsc{Sel}]
$$

La evaluación de esta expresión se indefine (análogo a `fix` $\lambda x : \sigma.x$).

Ejemplo: Los naturales

- Asumir que existen los objetos `true` y `false` que corresponden a los valores booleanos
- Luego

$$\texttt{zero} \stackrel{\text{def}}{=}$$

Ejemplo: Los naturales

- Asumir que existen los objetos `true` y `false` que corresponden a los valores booleanos
- Luego

$$
\begin{aligned}
\texttt{zero} \stackrel{\text{def}}{=} [\ & \mathit{iszero} = \texttt{true}, \\
& \mathit{pred} \quad = \varsigma(x)x, \\
& \mathit{succ} \quad = \varsigma(x)(x.\mathit{iszero} := \texttt{false}).\mathit{pred} := x]
\end{aligned}
$$

Ejemplo: Los naturales

- Asumir que existen los objetos `true` y `false` que corresponden a los valores booleanos
- Luego

$$
\begin{aligned}
\texttt{zero} \stackrel{\text{def}}{=} [\ & \mathit{iszero} = \texttt{true}, \\
& \mathit{pred} \quad = \varsigma(x)x, \\
& \mathit{succ} \quad = \varsigma(x)(x.\mathit{iszero} := \texttt{false}).\mathit{pred} := x]
\end{aligned}
$$

$\texttt{uno}$

Ejemplo: Los naturales

- Asumir que existen los objetos `true` y `false` que corresponden a los valores booleanos
- Luego

$$
\begin{array}{lcl}
\texttt{zero} & \stackrel{\text{def}}{=} & [\ \mathit{iszero} = \texttt{true}, \\
& & \quad \mathit{pred} = \varsigma(x)x, \\
& & \quad \mathit{succ} = \varsigma(x)(x.\mathit{iszero} := \texttt{false}).\mathit{pred} := x] \\
\texttt{uno} & \stackrel{\text{def}}{=} & \texttt{zero}.\mathit{succ}
\end{array}
$$

$$\texttt{uno} \longrightarrow$$

Ejemplo: Los naturales

- Asumir que existen los objetos `true` y `false` que corresponden a los valores booleanos
- Luego

$$
\begin{array}{lll}
\texttt{zero} & \stackrel{\text{def}}{=} & [\; \begin{array}{ll} \mathit{iszero} & = \texttt{true}, \\ \mathit{pred} & = \varsigma(x)x, \\ \mathit{succ} & = \varsigma(x)(x.\mathit{iszero} := \texttt{false}).\mathit{pred} := x] \end{array} \\
\texttt{uno} & \stackrel{\text{def}}{=} & \texttt{zero}.\mathit{succ}
\end{array}
$$

$$
\texttt{uno} \longrightarrow \underbrace{[\mathit{iszero} = \texttt{false}, \mathit{pred} = \texttt{zero}, \mathit{succ} = \ldots]}_{\texttt{uno}'}
$$

`dos`

Ejemplo: Los naturales

- Asumir que existen los objetos `true` y `false` que corresponden a los valores booleanos
- Luego

$$
\begin{array}{lll}
\texttt{zero} & \stackrel{\text{def}}{=} & [\; \mathit{iszero} = \texttt{true}, \\
 & & \quad \mathit{pred} = \varsigma(x)x, \\
 & & \quad \mathit{succ} = \varsigma(x)(x.\mathit{iszero} := \texttt{false}).\mathit{pred} := x] \\
\texttt{uno} & \stackrel{\text{def}}{=} & \texttt{zero}.\mathit{succ}
\end{array}
$$

$$\texttt{uno} \longrightarrow \underbrace{[\mathit{iszero} = \texttt{false}, \mathit{pred} = \texttt{zero}, \mathit{succ} = \ldots]}_{\texttt{uno}'}$$

$$\texttt{dos} \stackrel{\text{def}}{=} \texttt{zero}.\mathit{succ}.\mathit{succ}$$

$$\texttt{dos} \longrightarrow [\mathit{iszero} = \texttt{false}, \mathit{pred} = \texttt{uno}', \mathit{succ} = \ldots]$$

$$\ldots$$

Las funciones

- Es posible codificar los términos del cálculo λ (no tipado).

$$M ::= MN \mid \lambda x.M \mid x$$

- Idea:
  - Representar a una función como un objeto $[arg = \ldots,\ val = \ldots]$.
  - Al aplicar una función, primero se asigna el valor del argumento al atributo arg y luego se envía el mensaje val que evalúa el cuerpo de la función.
  - (fv) se traduce en $(\mathsf{o}_f.arg := \mathsf{o}_v).val$

Codificación del cálculo lambda $[\![\_]\!] : M \to a$

Codificación del cálculo lambda $[\![\_]\!] : M \to a$

$$[\![x]\!] \quad \stackrel{\text{def}}{=} x$$

Codificación del cálculo lambda $[\![\_]\!] : M \to a$

$$
\begin{aligned}
[\![x]\!] &\stackrel{\text{def}}{=} x \\
[\![MN]\!] &\stackrel{\text{def}}{=} ([\![M]\!].arg := [\![N]\!]).val
\end{aligned}
$$

Codificación del cálculo lambda $[\![\_]\!] : M \to a$

$$
\begin{aligned}
[\![x]\!] &\stackrel{\text{def}}{=} x \\
[\![MN]\!] &\stackrel{\text{def}}{=} ([\![M]\!].arg := [\![N]\!]).val \\
[\![\lambda x.M]\!] &\stackrel{\text{def}}{=} [\ val = \varsigma(y)[\![M]\!]\{y.arg/x\}, \\
&\qquad arg = \qquad\qquad\qquad\qquad] \quad y \notin \mathsf{fv}(M)
\end{aligned}
$$

Codificación del cálculo lambda $[\![\_]\!] : M \to a$

$$
\begin{aligned}
[\![x]\!] &\stackrel{\text{def}}{=} x \\
[\![MN]\!] &\stackrel{\text{def}}{=} ([\![M]\!].arg := [\![N]\!]).val \\
[\![\lambda x.M]\!] &\stackrel{\text{def}}{=} [\, val = \varsigma(y)[\![M]\!]\{y.arg/x\}, \\
&\qquad arg = \varsigma(y)y.arg \,] \quad y \notin \mathsf{fv}(M)
\end{aligned}
$$

Codificación del cálculo lambda $[\![\_]\!] : M \to a$

$$[\![x]\!] \stackrel{\mathsf{def}}{=} x$$

$$[\![MN]\!] \stackrel{\mathsf{def}}{=} ([\![M]\!].arg := [\![N]\!]).val$$

$$[\![\lambda x.M]\!] \stackrel{\mathsf{def}}{=} [\, val = \varsigma(y)[\![M]\!]\{y.arg/x\}, \quad arg = \varsigma(y)y.arg \,] \quad y \notin \mathsf{fv}(M)$$

$(\lambda x.x)v$

Codificación del cálculo lambda $[\![\_]\!] : M \to a$

$$[\![x]\!] \stackrel{\text{def}}{=} x$$

$$[\![MN]\!] \stackrel{\text{def}}{=} ([\![M]\!].arg := [\![N]\!]).val$$

$$[\![\lambda x.M]\!] \stackrel{\text{def}}{=} [\, val = \varsigma(y)[\![M]\!]\{y.arg/x\}, \quad arg = \varsigma(y)y.arg \,] \quad y \notin \mathsf{fv}(M)$$

$(\lambda x.x)v$

$$[\![\lambda x.x]\!] \stackrel{\text{def}}{=}$$

Codificación del cálculo lambda $[\![\_]\!] : M \to a$

$$[\![x]\!] \stackrel{\text{def}}{=} x$$

$$[\![MN]\!] \stackrel{\text{def}}{=} ([\![M]\!].arg := [\![N]\!]).val$$

$$[\![\lambda x.M]\!] \stackrel{\text{def}}{=} [\, val = \varsigma(y)[\![M]\!]\{y.arg/x\}, \; arg = \varsigma(y)y.arg \,] \quad y \notin \mathsf{fv}(M)$$

$(\lambda x.x)v$

$$[\![\lambda x.x]\!] \stackrel{\text{def}}{=} [val = \varsigma(y)[\![x]\!]\{y.arg/x\}, arg = \varsigma(y)y.arg]$$

$$=$$

Codificación del cálculo lambda $[\![\_]\!] : M \to a$

$$[\![x]\!] \stackrel{\text{def}}{=} x$$

$$[\![MN]\!] \stackrel{\text{def}}{=} ([\![M]\!].arg := [\![N]\!]).val$$

$$[\![\lambda x.M]\!] \stackrel{\text{def}}{=} [\ val = \varsigma(y)[\![M]\!]\{y.arg/x\},\ arg = \varsigma(y)y.arg\ ] \quad y \notin \mathsf{fv}(M)$$

$(\lambda x.x)v$

$$\begin{aligned}
[\![\lambda x.x]\!] &\stackrel{\text{def}}{=} [val = \varsigma(y)[\![x]\!]\{y.arg/x\}, arg = \varsigma(y)y.arg] \\
&= [val = \varsigma(y)x\{y.arg/x\}, arg = \varsigma(y)y.arg] \\
&=
\end{aligned}$$

Codificación del cálculo lambda $[\![\_]\!] : M \to a$

$$[\![x]\!] \stackrel{\text{def}}{=} x$$

$$[\![MN]\!] \stackrel{\text{def}}{=} ([\![M]\!].arg := [\![N]\!]).val$$

$$[\![\lambda x.M]\!] \stackrel{\text{def}}{=} [\, val = \varsigma(y)[\![M]\!]\{y.arg/x\}, \; arg = \varsigma(y)y.arg \,] \quad y \notin \mathsf{fv}(M)$$

$(\lambda x.x)v$

$$\begin{aligned}
[\![\lambda x.x]\!] &\stackrel{\text{def}}{=} [val = \varsigma(y)[\![x]\!]\{y.arg/x\}, arg = \varsigma(y)y.arg] \\
&= [val = \varsigma(y)x\{y.arg/x\}, arg = \varsigma(y)y.arg] \\
&= [val = \varsigma(y)y.arg, arg = \varsigma(y)y.arg]
\end{aligned}$$

Codificación del cálculo lambda $[\![\_]\!] : M \to a$

$$[\![x]\!] \stackrel{\text{def}}{=} x$$

$$[\![MN]\!] \stackrel{\text{def}}{=} ([\![M]\!].arg := [\![N]\!]).val$$

$$[\![\lambda x.M]\!] \stackrel{\text{def}}{=} [\, val = \varsigma(y)[\![M]\!]\{y.arg/x\}, \; arg = \varsigma(y)y.arg \,] \quad y \notin \mathsf{fv}(M)$$

$(\lambda x.x)v$

$$\begin{aligned} [\![\lambda x.x]\!] &\stackrel{\text{def}}{=} [val = \varsigma(y)[\![x]\!]\{y.arg/x\}, arg = \varsigma(y)y.arg] \\ &= [val = \varsigma(y)x\{y.arg/x\}, arg = \varsigma(y)y.arg] \\ &= [val = \varsigma(y)y.arg, arg = \varsigma(y)y.arg] \end{aligned}$$

$$[\![(\lambda x.x)\ M]\!] \stackrel{\text{def}}{=}$$

Codificación del cálculo lambda $[\![\_]\!] : M \to a$

$$[\![x]\!] \stackrel{\text{def}}{=} x$$

$$[\![MN]\!] \stackrel{\text{def}}{=} ([\![M]\!].arg := [\![N]\!]).val$$

$$[\![\lambda x.M]\!] \stackrel{\text{def}}{=} [\, val = \varsigma(y)[\![M]\!]\{y.arg/x\}, \quad arg = \varsigma(y)y.arg \,] \quad y \notin \mathsf{fv}(M)$$

$(\lambda x.x)v$

$$\begin{aligned}
[\![\lambda x.x]\!] &\stackrel{\text{def}}{=} [val = \varsigma(y)[\![x]\!]\{y.arg/x\}, arg = \varsigma(y)y.arg] \\
&= [val = \varsigma(y)x\{y.arg/x\}, arg = \varsigma(y)y.arg] \\
&= [val = \varsigma(y)y.arg, arg = \varsigma(y)y.arg]
\end{aligned}$$

$$\begin{aligned}
[\![(\lambda x.x)\ M]\!] &\stackrel{\text{def}}{=} ([\![\lambda x.x]\!].arg := [\![M]\!]).val \\
&=
\end{aligned}$$

Codificación del cálculo lambda $[\![\_]\!] : M \to a$

$$
\begin{array}{lcl}
[\![x]\!] & \stackrel{\text{def}}{=} & x \\
[\![M N]\!] & \stackrel{\text{def}}{=} & ([\![M]\!].arg := [\![N]\!]).val \\
[\![\lambda x.M]\!] & \stackrel{\text{def}}{=} & [\ val = \varsigma(y)[\![M]\!]\{y.arg/x\}, \\
 & & \quad arg = \varsigma(y)y.arg \qquad] \quad y \notin \mathsf{fv}(M)
\end{array}
$$

$(\lambda x.x)v$

$$
\begin{array}{lcl}
[\![\lambda x.x]\!] & \stackrel{\text{def}}{=} & [val = \varsigma(y)[\![x]\!]\{y.arg/x\}, arg = \varsigma(y)y.arg] \\
 & = & [val = \varsigma(y)x\{y.arg/x\}, arg = \varsigma(y)y.arg] \\
 & = & [val = \varsigma(y)y.arg, arg = \varsigma(y)y.arg]
\end{array}
$$

$$
\begin{array}{lcl}
[\![(\lambda x.x)\ M]\!] & \stackrel{\text{def}}{=} & ([\![\lambda x.x]\!].arg := [\![M]\!]).val \\
 & = & ([val = \varsigma(x)x.arg, arg = \varsigma(x)x.arg]. \\
 & & \qquad\qquad arg := [\![M]\!]).val \\
 & \longrightarrow & [\![M]\!]
\end{array}
$$

siempre que $[\![M]\!]$ sea un objeto.

Codificación del cálculo lambda $[\![\_]\!] : M \to a$

$$
\begin{aligned}
[\![x]\!] &\stackrel{\text{def}}{=} x \\
[\![MN]\!] &\stackrel{\text{def}}{=} ([\![M]\!].arg := [\![N]\!]).val \\
[\![\lambda x.M]\!] &\stackrel{\text{def}}{=} [\, val = \varsigma(y)[\![M]\!]\{y.arg/x\}, \\
&\qquad arg = \varsigma(y)y.arg \,] \quad y \notin \mathsf{fv}(M)
\end{aligned}
$$

$(\lambda x.x)v$

$$
\begin{aligned}
[\![\lambda x.x]\!] &\stackrel{\text{def}}{=} [val = \varsigma(y)[\![x]\!]\{y.arg/x\}, arg = \varsigma(y)y.arg] \\
&= [val = \varsigma(y)x\{y.arg/x\}, arg = \varsigma(y)y.arg] \\
&= [val = \varsigma(y)y.arg, arg = \varsigma(y)y.arg]
\end{aligned}
$$

$$
\begin{aligned}
[\![(\lambda x.x)\ M]\!] &\stackrel{\text{def}}{=} ([\![\lambda x.x]\!].arg := [\![M]\!]).val \\
&= ([val = \varsigma(x)x.arg, arg = \varsigma(x)x.arg]. \\
&\qquad\qquad arg := [\![M]\!]).val \\
&\longrightarrow [\![M]\!]
\end{aligned}
$$

siempre que $[\![M]\!]$ sea un objeto.

Qué sucede al evaluar $[\![\lambda x.x]\!].val$?

Métodos con parámetros

Un método que espera un parámetro es un método cuya definición es (un objeto que codifica a) una función.

$$\varsigma(y)[\![\lambda x.M]\!]$$

Métodos con parámetros

Un método que espera un parámetro es un método cuya definición es (un objeto que codifica a) una función.

$$\varsigma(y)[\![\lambda x.M]\!]$$

Notación

- $\lambda(x)M$ en lugar de $[\![\lambda x.M]\!]$
- $M(N)$ en lugar de $[\![MN]\!]$

Ejemplo: Punto en el plano

- Un punto en el plano que puede ser desplazado y se encuentra inicialmente en el origen de coordenadas.

$$\texttt{origen} \stackrel{\text{def}}{=} [\begin{array}{ll} x & = 0, \\ y & = 0, \\ mv\_x & = \varsigma(p)\lambda(d_x)p.x := p.x + d_x, \\ mv\_y & = \varsigma(p)\lambda(d_y)p.y := p.y + d_y \quad] \end{array}$$

Ejemplo: Punto en el plano

- Un punto en el plano que puede ser desplazado y se encuentra inicialmente en el origen de coordenadas.

$$\texttt{origen} \stackrel{\text{def}}{=} [\begin{array}{ll} x & = 0, \\ y & = 0, \\ mv\_x & = \varsigma(p)\lambda(d_x)p.x := p.x + d_x, \\ mv\_y & = \varsigma(p)\lambda(d_y)p.y := p.y + d_y \end{array}]$$

- Luego,

$$\begin{array}{lcl} \texttt{unidad} & \stackrel{\text{def}}{=} & \texttt{origen}.mv\_x(1).mv\_y(1) \\ & \longrightarrow & \end{array}$$

Ejemplo: Punto en el plano

- Un punto en el plano que puede ser desplazado y se encuentra inicialmente en el origen de coordenadas.

$$\texttt{origen} \stackrel{\text{def}}{=} \left[\begin{array}{ll} x & = 0, \\ y & = 0, \\ mv\_x & = \varsigma(p)\lambda(d_x)p.x := p.x + d_x, \\ mv\_y & = \varsigma(p)\lambda(d_y)p.y := p.y + d_y \end{array} \right]$$

- Luego,

$$\begin{array}{rcl} \texttt{unidad} & \stackrel{\text{def}}{=} & \texttt{origen}.mv\_x(1).mv\_y(1) \\ & \longrightarrow & [x = 1, y = 1, mv\_x = \ldots, mv\_y = \ldots] \end{array}$$

(Stateless) Traits

- Un trait es una colección de ciertos métodos.
- (Stateless) *Traits* no especifican variables de estado ni acceden al estado.

(Stateless) Traits

- Un trait es una colección de ciertos métodos.
- (Stateless) *Traits* no especifican variables de estado ni acceden al estado.

$$\texttt{CompT} \stackrel{\text{def}}{=} [\ \begin{array}{ll} eq & = \varsigma(t)\lambda(x)\lambda(y)(x.comp(y)) == 0, \\ lt & = \varsigma(t)\lambda(x)\lambda(y)(x.comp(y)) < 0 \end{array}\ ]$$

(Stateless) Traits

- Los podemos pensar como una colección de pre-métodos:
  - pre-método: $\varsigma(\mathtt{t})\lambda(y)b$ con $\mathtt{t} \notin \mathsf{fv}(\lambda(y)b)$ (no usan el parámetro `self` $\mathtt{t}$).
  - Recordar que en este caso podemos omitir $\varsigma(\mathtt{t})$ y escribir $\lambda(y)b$.
  - Luego, $\mathtt{t} = [l_i = \lambda(y_i)b_i{}^{i \in 1..n}]$ es un trait.

(Stateless) Traits

- Los podemos pensar como una colección de pre-métodos:
  - pre-método: $\varsigma(\mathtt{t})\lambda(y)b$ con $\mathtt{t} \notin \mathsf{fv}(\lambda(y)b)$ (no usan el parámetro `self` $\mathtt{t}$).
  - Recordar que en este caso podemos omitir $\varsigma(\mathtt{t})$ y escribir $\lambda(y)b$.
  - Luego, $\mathtt{t} = [l_i = \lambda(y_i)b_i{}^{i \in 1..n}]$ es un trait.
- A partir de un trait $\mathtt{t} = [l_i = \lambda(y_i)b_i{}^{i \in 1..n}]$ podemos definir un constructor de objetos (cuando $\mathtt{t}$ es completo).

(Stateless) Traits

- Los podemos pensar como una colección de pre-métodos:
  - pre-método: $\varsigma(\texttt{t})\lambda(y)b$ con $\texttt{t} \notin \mathsf{fv}(\lambda(y)b)$ (no usan el parámetro `self` `t`).
  - Recordar que en este caso podemos omitir $\varsigma(\texttt{t})$ y escribir $\lambda(y)b$.
  - Luego, $\texttt{t} = [l_i = \lambda(y_i)b_i{}^{i\in 1..n}]$ es un trait.
- A partir de un trait $\texttt{t} = [l_i = \lambda(y_i)b_i{}^{i\in 1..n}]$ podemos definir un constructor de objetos (cuando `t` es completo).

$$new \stackrel{\mathsf{def}}{=} \lambda(z)[l_i = \varsigma(s)z.l_i(s)^{i\in 1..n}]$$

(Stateless) Traits

- Los podemos pensar como una colección de pre-métodos:
  - pre-método: $\varsigma(\mathtt{t})\lambda(y)b$ con $\mathtt{t} \notin \mathsf{fv}(\lambda(y)b)$ (no usan el parámetro `self` $\mathtt{t}$).
  - Recordar que en este caso podemos omitir $\varsigma(\mathtt{t})$ y escribir $\lambda(y)b$.
  - Luego, $\mathtt{t} = [l_i = \lambda(y_i)b_i{}^{i \in 1..n}]$ es un trait.
- A partir de un trait $\mathtt{t} = [l_i = \lambda(y_i)b_i{}^{i \in 1..n}]$ podemos definir un constructor de objetos (cuando $\mathtt{t}$ es completo).

$$new \stackrel{\mathsf{def}}{=} \lambda(z)[l_i = \varsigma(s)z.l_i(s)^{i \in 1..n}]$$

$$\begin{aligned} o &\stackrel{\mathsf{def}}{=} new\ (\mathtt{t}) \\ &\approx [l_i = \varsigma(s)\mathtt{t}.l_i(s)^{i \in 1..n}] \\ &\approx [l_i = \varsigma(y_i)b_i{}^{i \in 1..n}] \end{aligned}$$

(Stateless) Traits

$$\mathtt{CompT} \stackrel{\mathsf{def}}{=} [\ eq = \varsigma(t)\lambda(x)\lambda(y)(x.comp(y)) == 0,\quad lt = \varsigma(t)\lambda(x)\lambda(y)(x.comp(y)) < 0\ ]$$

$$new \stackrel{\mathsf{def}}{=} \lambda(z)[l_i = \varsigma(s)z.l_i(s)^{i \in 1..n}]$$

$$new\ (\mathtt{CompT}) \approx$$

(Stateless) Traits

$$
\begin{array}{lll}
\texttt{CompT} & \stackrel{\mathsf{def}}{=} & [\ eq = \varsigma(t)\lambda(x)\lambda(y)(x.comp(y)) == 0, \\
 & & \quad lt = \varsigma(t)\lambda(x)\lambda(y)(x.comp(y)) < 0 \quad] \\
new & \stackrel{\mathsf{def}}{=} & \lambda(z)[l_i = \varsigma(s)z.l_i(s)^{i\in 1..n}] \\
new\ (\texttt{CompT}) & \approx & [\ eq = \varsigma(s)\texttt{CompT}.eq(s) \\
 & & \quad lt = \varsigma(s)\texttt{CompT}.lt(s) \quad] \\
 & \approx & [\ eq = \varsigma(s)\lambda(y)\ (s.comp(y)) == 0, \\
 & & \quad lt = \varsigma(s)\lambda(y)\ (s.comp(y)) < 0 \quad]
\end{array}
$$

(Stateless) Traits

$$\texttt{CompT} \stackrel{\text{def}}{=} [\; eq = \varsigma(t)\lambda(x)\lambda(y)(x.comp(y)) == 0,\; lt = \varsigma(t)\lambda(x)\lambda(y)(x.comp(y)) < 0 \;]$$

$$new \stackrel{\text{def}}{=} \lambda(z)[l_i = \varsigma(s)z.l_i(s)^{i \in 1..n}]$$

$$\begin{aligned} new\ (\texttt{CompT}) &\approx [\; eq = \varsigma(s)\texttt{CompT}.eq(s) \\ &\qquad lt = \varsigma(s)\texttt{CompT}.lt(s) \;] \\ &\approx [\; eq = \varsigma(s)\lambda(y)\ (s.comp(y)) == 0, \\ &\qquad lt = \varsigma(s)\lambda(y)\ (s.comp(y)) < 0 \;] \end{aligned}$$

- Este objeto es inutilizable (porque `CompT` no es completo ya que *comp* no es un método de `CompT`).

Clases

- Una clase es un *trait* (completo) que además provee un método new.

$$\mathsf{c} \stackrel{\text{def}}{=} \begin{array}{l} [\ new = \varsigma(z)[l_i = \varsigma(s)z.l_i(s)^{i \in 1..n}], \\ \ \ l_i \quad\ \ = \lambda(s)b_i{}^{i \in 1..n}] \end{array}$$

- Luego,

$$\begin{array}{rcl} o & \stackrel{\text{def}}{=} & \mathsf{c}.new \\ & \longrightarrow & [l_i = \varsigma(s)\mathsf{c}.l_i(s)^{i \in 1..n}] \\ & \approx & [l_i = \varsigma(s)b_i{}^{i \in 1..n}] \end{array}$$

Clase Contador

$$
\begin{aligned}
Contador \stackrel{\mathsf{def}}{=} [\ & new = \varsigma(z)[\ v = \varsigma(s) z.v(s), \\
& \qquad\qquad\quad inc = \varsigma(s) z.inc(s), \\
& \qquad\qquad\quad get = \varsigma(s) z.get(s)], \\
& v = \lambda(s) 0, \\
& inc = \lambda(s) s.v := s.v + 1, \\
& get = \lambda(s) s.v \qquad\qquad\qquad\qquad]
\end{aligned}
$$

Representando Herencia

- Sea la clase

$$\mathsf{c} \stackrel{\mathsf{def}}{=} \begin{array}{l} [\ new = \varsigma(z)[l_i = \varsigma(s)z.l_i(s)^{i\in 1..n}], \\ \quad l_i \quad\ \ = \lambda(s)b_i{}^{i\in 1..n}] \end{array}$$

Representando Herencia

- Sea la clase

$$\mathsf{c} \stackrel{\text{def}}{=} \begin{array}{l} [\; new = \varsigma(z)[l_i = \varsigma(s)z.l_i(s)^{i\in 1..n}], \\ \;\; l_i \quad = \lambda(s)b_i{}^{i\in 1..n}] \end{array}$$

- Se desea definir c' como subclase de c que agrega los pre-métodos $\lambda(s)b_k{}^{k\in n+1..n+m}$

Representando Herencia

- Sea la clase

$$\mathsf{c} \stackrel{\mathsf{def}}{=} \begin{array}{l} [\; new = \varsigma(z)[l_i = \varsigma(s)z.l_i(s)^{i\in 1..n}], \\ \quad l_i \quad\;\; = \lambda(s){b_i}^{i\in 1..n}] \end{array}$$

- Se desea definir c' como subclase de c que agrega los pre-métodos $\lambda(s){b_k}^{k\in n+1..n+m}$

$$\mathsf{c}' \stackrel{\mathsf{def}}{=} \begin{array}{lll} [\; new & = \varsigma(z)[l_i = \varsigma(s)z.l_i(s)^{i\in 1..n+m}], & \\ \quad l_j & = \mathsf{c}.{l_j}^{j\in 1..n} & \\ \quad l_k & = \lambda(s){b_k}^{k\in n+1..n+m} &] \end{array}$$

Representando Herencia

- Sea la clase

$$\mathsf{c} \stackrel{\mathsf{def}}{=} \begin{array}{l} [\; new = \varsigma(z)[l_i = \varsigma(s)z.l_i(s)^{i\in 1..n}], \\ \;\; l_i \quad\; = \lambda(s)b_i{}^{i\in 1..n}] \end{array}$$

- Se desea definir c' como subclase de c que agrega los pre-métodos $\lambda(s)b_k{}^{k\in n+1..n+m}$

$$\mathsf{c}' \stackrel{\mathsf{def}}{=} \begin{array}{l} [\; new = \varsigma(z)[l_i = \varsigma(s)z.l_i(s)^{i\in 1..n+m}], \\ \;\; l_j \quad\; = \mathsf{c}.l_j{}^{j\in 1..n} \\ \;\; l_k \quad\; = \lambda(s)b_k{}^{k\in n+1..n+m} \qquad] \end{array}$$

- En una subclase también se puede redefinir pre-métodos.