Semántica denotacional para un cálculo-λ relacional

Defensa de tesis de licenciatura

Mariana Milicich
Director: Pablo Barenbaum
Jurado: Alejandro Díaz-Caro y Hernán Melgratti

Departamento de Computación - FCEyN - UBA

10 de agosto de 2022

Motivación

Los lenguajes de programación tienen características diversas que pueden agruparse en paradigmas tales como:

Motivación

Los lenguajes de programación tienen características diversas que pueden agruparse en paradigmas tales como:

Funcional
Ej: Haskell, OCaml

Lógico/relacional
Ej: Prolog, Mercury

Motivación

Los lenguajes de programación tienen características diversas que pueden agruparse en paradigmas tales como:

Funcional
Ej: Haskell, OCaml

- Funciones de alto orden

Lógico/relacional
Ej: Prolog, Mercury

Motivación

Los lenguajes de programación tienen características diversas que pueden agruparse en paradigmas tales como:

Funcional
Ej: Haskell, OCaml

- Funciones de alto orden
- Tipos de datos algebraicos

Lógico/relacional
Ej: Prolog, Mercury

Motivación

Los lenguajes de programación tienen características diversas que pueden agruparse en paradigmas tales como:

Funcional
Ej: Haskell, OCaml

- Funciones de alto orden
- Tipos de datos algebraicos
- *Pattern matching*

Lógico/relacional
Ej: Prolog, Mercury

Motivación

Los lenguajes de programación tienen características diversas que pueden agruparse en paradigmas tales como:

Funcional
Ej: Haskell, OCaml

- Funciones de alto orden
- Tipos de datos algebraicos
- *Pattern matching*

Lógico/relacional
Ej: Prolog, Mercury

- Variables instanciables

Motivación

Los lenguajes de programación tienen características diversas que pueden agruparse en paradigmas tales como:

Funcional
Ej: Haskell, OCaml

- Funciones de alto orden
- Tipos de datos algebraicos
- *Pattern matching*

Lógico/relacional
Ej: Prolog, Mercury

- Variables instanciables
- No determinismo

Motivación

Los lenguajes de programación tienen características diversas que pueden agruparse en paradigmas tales como:

Funcional
Ej: Haskell, OCaml

- Funciones de alto orden
- Tipos de datos algebraicos
- *Pattern matching*

Lógico/relacional
Ej: Prolog, Mercury

- Variables instanciables
- No determinismo
- Unificación

Los lenguajes de programación tienen características diversas que pueden agruparse en paradigmas tales como:

Funcional
Ej: Haskell, OCaml

- Funciones de alto orden
- Tipos de datos algebraicos
- *Pattern matching*

Lógico/relacional
Ej: Prolog, Mercury

- Variables instanciables
- No determinismo
- Unificación

Funcional-lógico

Ej: Curry, λ-Prolog

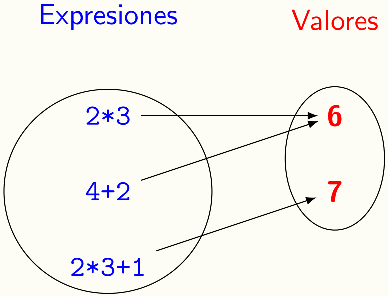
Semántica de lenguajes de programación

Manera de atribuirle significado a las expresiones de un lenguaje.

Semántica de lenguajes de programación

Manera de atribuirle significado a las expresiones de un lenguaje.

Hay varios tipos de semánticas:

Operacional	Denotacional
`2*3+1` $\leadsto$ `6+1` $\leadsto$ `7`	

Semántica operacional

$$\text{Programa} \iff \text{estado en el cómputo.}$$

Semántica operacional

Programa $\iff$ estado en el cómputo.

$$P_1 \to P_2 \to \ldots \to P_n$$

Semántica operacional

$$\text{Programa} \iff \text{estado en el cómputo.}$$

$$P_1 \to P_2 \to \ldots \to P_n$$

Ejemplo — semántica operacional del cálculo-λ

$$\frac{t \to t'}{\lambda x.\, t \to \lambda x.\, t'}\,\xi \qquad \frac{t \to t'}{t\, s \to t'\, s}\,\mu \qquad \frac{s \to s'}{t\, s \to t\, s'}\,\nu$$

$$\frac{}{(\lambda x.\, t)\, s \to t\{x := s\}}\,\beta \qquad \frac{x \notin \mathsf{fv}(t)}{\lambda x.\, t\, x \to t}\,\eta$$

Donde los términos son:

$$t ::= x \mid \lambda x.\, t \mid t\, t$$

Semántica denotacional

Programa $\iff$ valor matemático en algún dominio de interpretación.

Programa $\iff$ valor matemático en algún dominio de interpretación.

Ejemplo — semántica denotacional del cálculo-λ

Dada:

$$
\begin{aligned}
[\![\alpha]\!] &\stackrel{\mathrm{def}}{=} \mathsf{S}_\alpha \\
[\![A \to B]\!] &\stackrel{\mathrm{def}}{=} [\![B]\!]^{[\![A]\!]}
\end{aligned}
$$

Definimos:

$$
\begin{aligned}
[\![x]\!]_\rho &\stackrel{\mathrm{def}}{=} \rho(x) \\
[\![\lambda x.\, t]\!]_\rho &\stackrel{\mathrm{def}}{=} f \qquad \text{donde } f(a) = [\![t]\!]_{\rho[x \mapsto a]} \\
[\![t\, s]\!]_\rho &\stackrel{\mathrm{def}}{=} [\![t]\!]_\rho([\![s]\!]_\rho)
\end{aligned}
$$

Relación entre la semántica operacional y la denotacional

Sea t un término cerrado y tipable. Entonces:

Teorema (Correctitud)

Si $t \to s$ entonces $[\![t]\!] = [\![s]\!]$ vale en toda interpretación.

Teorema (Completitud)

Si $[\![t]\!] = [\![s]\!]$ vale en toda interpretación, entonces $t \leftrightarrow^* s$.

Objetivo de la tesis

Esta tesis se enfoca en el cálculo-$\lambda^{\mathtt{U}}$:

Objetivo de la tesis

Esta tesis se enfoca en el cálculo-$\lambda^{\mathtt{U}}$:

- Cálculo funcional–lógico

Objetivo de la tesis

Esta tesis se enfoca en el cálculo-$\lambda^{\mathtt{U}}$:

- Cálculo funcional–lógico
- Extensión del cálculo-λ con unificación, no determinismo, introducción de variables simbólicas

Objetivo de la tesis

Esta tesis se enfoca en el cálculo-$\lambda^{\mathtt{U}}$:

- Cálculo funcional–lógico
- Extensión del cálculo-λ con unificación, no determinismo, introducción de variables simbólicas

Objetivos

- Proponer un sistema de tipos para el cálculo-$\lambda^{\mathtt{U}}$

Objetivo de la tesis

Esta tesis se enfoca en el cálculo-$\lambda^{\mathtt{U}}$:

- Cálculo funcional–lógico
- Extensión del cálculo-λ con unificación, no determinismo, introducción de variables simbólicas

Objetivos

- Proponer un sistema de tipos para el cálculo-$\lambda^{\mathtt{U}}$
- Dar una semántica denotacional para este cálculo

Objetivo de la tesis

Esta tesis se enfoca en el cálculo-$\lambda^{\mathtt{U}}$:

- Cálculo funcional–lógico
- Extensión del cálculo-λ con unificación, no determinismo, introducción de variables simbólicas

Objetivos

- Proponer un sistema de tipos para el cálculo-$\lambda^{\mathtt{U}}$
- Dar una semántica denotacional para este cálculo
- Estudiar sus propiedades:

Objetivo de la tesis

Esta tesis se enfoca en el cálculo-$\lambda^{\mathtt{U}}$:

- Cálculo funcional–lógico
- Extensión del cálculo-λ con unificación, no determinismo, introducción de variables simbólicas

Objetivos

- Proponer un sistema de tipos para el cálculo-$\lambda^{\mathtt{U}}$
- Dar una semántica denotacional para este cálculo
- Estudiar sus propiedades:
  - Preservación de tipos

Objetivo de la tesis

Esta tesis se enfoca en el cálculo-$\lambda^{\mathtt{U}}$:

- Cálculo funcional–lógico
- Extensión del cálculo-λ con unificación, no determinismo, introducción de variables simbólicas

Objetivos

- Proponer un sistema de tipos para el cálculo-$\lambda^{\mathtt{U}}$
- Dar una semántica denotacional para este cálculo
- Estudiar sus propiedades:
  - Preservación de tipos
  - Correctitud
  - Completitud

El cálculo-$\lambda^{\mathtt{U}}$

Propuesto en 2019 por Pablo Barenbaum y Federico Lochbaum.

El cálculo-$\lambda^{\mathtt{U}}$

Propuesto en 2019 por Pablo Barenbaum y Federico Lochbaum.

Sintaxis

Términos:

$$
\begin{array}{rcll}
t & ::= & x & \text{variable} \\
 & \mid & \mathbf{c} & \text{constructor} \\
 & \mid & \lambda x.\, P & \text{abstracción} \\
 & \mid & \lambda^{\ell} x.\, P & \text{abstracción alojada} \\
 & \mid & t\, s & \text{aplicación} \\
 & \mid & t \stackrel{\bullet}{=} s & \text{unificación} \\
 & \mid & t; s & \text{secuencia} \\
 & \mid & \nu x.\, t & \text{introducción de variable simbólica}
\end{array}
$$

Programas:

$P ::= t_1 \oplus \ldots \oplus t_n$

El programa vacío se nota `fail`.

Valores:

$$
\begin{array}{rcl}
\mathtt{v} & ::= & x \\
 & \mid & \lambda^{\ell} x.\, P \\
 & \mid & \mathbf{c}\, \mathtt{v}_1 \ldots \mathtt{v}_n
\end{array}
$$

El cálculo-$\lambda^{\mathtt{U}}$

Propuesto en 2019 por Pablo Barenbaum y Federico Lochbaum.

Sintaxis

Términos:

$$
\begin{array}{rcll}
t & ::= & x & \text{variable} \\
 & | & \mathbf{c} & \text{constructor} \\
 & | & \lambda x.\, P & \text{abstracción} \\
 & | & \lambda^{\ell} x.\, P & \text{abstracción alojada} \\
 & | & t\, s & \text{aplicación} \\
 & | & t \overset{\bullet}{=} s & \text{unificación} \\
 & | & t; s & \text{secuencia} \\
 & | & \nu x.\, t & \text{introducción de variable simbólica}
\end{array}
$$

Programas:

$P ::= t_1 \oplus \ldots \oplus t_n$

El programa vacío se nota `fail`.

Valores:

$$
\begin{array}{rcl}
\mathtt{v} & ::= & x \\
 & | & \lambda^{\ell} x.\, P \\
 & | & \mathbf{c}\, \mathtt{v}_1 \ldots \mathtt{v}_n
\end{array}
$$

Los programas deben cumplir con un *invariante de coherencia*:

- Para cada par de subtérminos de la forma $\lambda^{\ell} x.\, P$ y $\lambda^{\ell'} y.\, Q$ tales que $\ell = \ell'$, se tiene que $P\{x := y\} = Q$.
- En todo subtérmino de la forma $\lambda^{\ell} x.\, P$, las variables libres de P deben estar globalmente libres o ligadas por λx.

Algoritmo de unificación

Un *problema de unificación* es de la forma:

$$\mathcal{E} = \{\mathtt{v}_1 \stackrel{\bullet}{=} \mathtt{w}_1, \ldots, \mathtt{v}_n \stackrel{\bullet}{=} \mathtt{w}_n\}$$

Algoritmo de unificación

Un *problema de unificación* es de la forma:

$$\mathcal{E} = \{\mathtt{v}_1 \stackrel{\bullet}{=} \mathtt{w}_1, \ldots, \mathtt{v}_n \stackrel{\bullet}{=} \mathtt{w}_n\}$$

Variante del algoritmo de Martelli–Montanari:

$$\begin{array}{rcll}
\{x \stackrel{\bullet}{=} x\} \uplus \mathcal{E} & \rightsquigarrow & \mathcal{E} & \\
\{\mathtt{v} \stackrel{\bullet}{=} x\} \uplus \mathcal{E} & \rightsquigarrow & \{x \stackrel{\bullet}{=} \mathtt{v}\} \uplus \mathcal{E} & \text{si } \mathtt{v} \notin \mathsf{Var} \\
\{\lambda^\ell x.\, P \stackrel{\bullet}{=} \lambda^\ell x.\, P\} \uplus \mathcal{E} & \rightsquigarrow & \mathcal{E} & \\
\{\mathbf{c}\,\mathtt{v}_1 \ldots \mathtt{v}_n \stackrel{\bullet}{=} \mathbf{c}\,\mathtt{w}_1 \ldots \mathtt{w}_n\} \uplus \mathcal{E} & \rightsquigarrow & \{\mathtt{v}_1 \stackrel{\bullet}{=} \mathtt{w}_1, \ldots, \mathtt{v}_n \stackrel{\bullet}{=} \mathtt{w}_n\} \uplus \mathcal{E} & \\
\{\mathtt{v} \stackrel{\bullet}{=} \mathtt{w}\} \uplus \mathcal{E} & \rightsquigarrow & \mathrm{FAIL} & \text{si } \mathtt{v} \text{ y } \mathtt{w} \text{ colisionan} \\
\{x \stackrel{\bullet}{=} \mathtt{v}\} \uplus \mathcal{E} & \rightsquigarrow & \{x \stackrel{\bullet}{=} \mathtt{v}\} \uplus \mathcal{E}\{x := \mathtt{v}\} & \text{si } x \in \mathsf{fv}(\mathcal{E}) \setminus \mathsf{fv}(\mathtt{v}) \\
\{x \stackrel{\bullet}{=} \mathtt{v}\} \uplus \mathcal{E} & \rightsquigarrow & \mathrm{FAIL} & \text{si } x \neq \mathtt{v} \text{ y } x \in \mathsf{fv}(\mathtt{v})
\end{array}$$

Algoritmo de unificación

Un *problema de unificación* es de la forma:

$$\mathcal{E} = \{\mathtt{v}_1 \overset{\bullet}{=} \mathtt{w}_1, \ldots, \mathtt{v}_n \overset{\bullet}{=} \mathtt{w}_n\}$$

Variante del algoritmo de Martelli–Montanari:

$$\begin{array}{rcll}
\{x \overset{\bullet}{=} x\} \uplus \mathcal{E} & \rightsquigarrow & \mathcal{E} & \\
\{\mathtt{v} \overset{\bullet}{=} x\} \uplus \mathcal{E} & \rightsquigarrow & \{x \overset{\bullet}{=} \mathtt{v}\} \uplus \mathcal{E} & \text{si } \mathtt{v} \notin \mathsf{Var} \\
\{\lambda^\ell x.\, P \overset{\bullet}{=} \lambda^\ell x.\, P\} \uplus \mathcal{E} & \rightsquigarrow & \mathcal{E} & \\
\{\mathbf{c}\,\mathtt{v}_1 \ldots \mathtt{v}_n \overset{\bullet}{=} \mathbf{c}\,\mathtt{w}_1 \ldots \mathtt{w}_n\} \uplus \mathcal{E} & \rightsquigarrow & \{\mathtt{v}_1 \overset{\bullet}{=} \mathtt{w}_1, \ldots, \mathtt{v}_n \overset{\bullet}{=} \mathtt{w}_n\} \uplus \mathcal{E} & \\
\{\mathtt{v} \overset{\bullet}{=} \mathtt{w}\} \uplus \mathcal{E} & \rightsquigarrow & \mathrm{FAIL} & \text{si } \mathtt{v} \text{ y } \mathtt{w} \text{ colisionan} \\
\{x \overset{\bullet}{=} \mathtt{v}\} \uplus \mathcal{E} & \rightsquigarrow & \{x \overset{\bullet}{=} \mathtt{v}\} \uplus \mathcal{E}\{x := \mathtt{v}\} & \text{si } x \in \mathsf{fv}(\mathcal{E}) \setminus \mathsf{fv}(\mathtt{v}) \\
\{x \overset{\bullet}{=} \mathtt{v}\} \uplus \mathcal{E} & \rightsquigarrow & \mathrm{FAIL} & \text{si } x \neq \mathtt{v} \text{ y } x \in \mathsf{fv}(\mathtt{v})
\end{array}$$

Si existe solución, el algoritmo devuelve $\sigma = \mathsf{mgu}(\mathcal{E})$.
Si no, el algoritmo falla.

Semántica operacional para el cálculo-$\lambda^{\mathtt{U}}$

Los términos se evalúan bajo contextos de evaluación débiles, W.

$$
\begin{array}{rcll}
\mathsf{W}\langle \lambda x.\, P\rangle & \xrightarrow{\texttt{alloc}} & \mathsf{W}\langle \lambda^{\ell} x.\, P\rangle & \text{con } \ell \text{ fresca} \\
\mathsf{W}\langle (\lambda^{\ell} x.\, t_1 \oplus \ldots \oplus t_n)\, \mathtt{v}\rangle & \xrightarrow{\texttt{beta}} & \mathsf{W}\langle t_1\{x := \mathtt{v}\}\rangle \oplus \ldots \oplus \mathsf{W}\langle t_n\{x := \mathtt{v}\}\rangle & \\
\mathsf{W}\langle \mathtt{v}; t\rangle & \xrightarrow{\texttt{seq}} & \mathsf{W}\langle t\rangle & \\
\mathsf{W}\langle \nu x.\, t\rangle & \xrightarrow{\texttt{fresh}} & \mathsf{W}\langle t\{x := y\}\rangle & \text{con } y \text{ fresca} \\
\mathsf{W}\langle \mathtt{v} \doteq \mathtt{w}\rangle & \xrightarrow{\texttt{unif}} & \mathsf{W}\langle \mathbf{ok}\rangle^{\sigma} & \text{si } \exists\ \sigma = \mathrm{mgu}(\{\mathtt{v} \doteq \mathtt{w}\}) \\
\mathsf{W}\langle \mathtt{v} \doteq \mathtt{w}\rangle & \xrightarrow{\texttt{fail}} & \texttt{fail} & \text{si } \mathrm{mgu}(\{\mathtt{v} \doteq \mathtt{w}\}) \text{ falla}
\end{array}
$$

Se incluye la siguiente regla de congruencia:

$$
\frac{t \to t_1 \oplus \ldots \oplus t_n}{P \oplus t \oplus Q \to P \oplus t_1 \oplus \ldots \oplus t_n \oplus Q}
$$

Ejemplos

$$(\lambda x.\, \mathbf{a}\, x \oplus \mathbf{b}\, x)\, \mathbf{c} \quad \to^* \quad \mathbf{a}\, \mathbf{c} \oplus \mathbf{b}\, \mathbf{c}$$

Ejemplos

$$
\begin{aligned}
(\lambda x.\, \mathbf{a}\, x \oplus \mathbf{b}\, x)\, \mathbf{c} &\quad \to^* \quad \mathbf{a}\, \mathbf{c} \oplus \mathbf{b}\, \mathbf{c} \\
(\lambda f.\, f\, (f\, \mathbf{c}))\, (\lambda x.\, \mathbf{a}\, x \oplus \mathbf{b}\, x) &\quad \to^* \quad \mathbf{a}\, (\mathbf{a}\, \mathbf{c}) \oplus \mathbf{a}\, (\mathbf{b}\, \mathbf{c}) \oplus \mathbf{b}\, (\mathbf{a}\, \mathbf{c}) \oplus \mathbf{b}\, (\mathbf{b}\, \mathbf{c})
\end{aligned}
$$

Ejemplos

$$\begin{array}{rcl}
(\lambda x.\, \mathbf{a}\, x \oplus \mathbf{b}\, x)\, \mathbf{c} & \to^* & \mathbf{a}\, \mathbf{c} \oplus \mathbf{b}\, \mathbf{c} \\
(\lambda f.\, f\, (f\, \mathbf{c}))\, (\lambda x.\, \mathbf{a}\, x \oplus \mathbf{b}\, x) & \to^* & \mathbf{a}\, (\mathbf{a}\, \mathbf{c}) \oplus \mathbf{a}\, (\mathbf{b}\, \mathbf{c}) \oplus \mathbf{b}\, (\mathbf{a}\, \mathbf{c}) \oplus \mathbf{b}\, (\mathbf{b}\, \mathbf{c}) \\
(\lambda x.\, (x \stackrel{\bullet}{=} \mathbf{c}\, \mathbf{b}\, y); \mathbf{d}\, y\, z)\, (\mathbf{c}\, z\, \mathbf{a}) & \to^* & \mathbf{d}\, \mathbf{a}\, \mathbf{b}
\end{array}$$

Ejemplos

$$
\begin{array}{rcl}
(\lambda x.\, \mathbf{a}\, x \oplus \mathbf{b}\, x)\, \mathbf{c} & \to^* & \mathbf{a}\, \mathbf{c} \oplus \mathbf{b}\, \mathbf{c} \\
(\lambda f.\, f\, (f\, \mathbf{c}))\, (\lambda x.\, \mathbf{a}\, x \oplus \mathbf{b}\, x) & \to^* & \mathbf{a}\, (\mathbf{a}\, \mathbf{c}) \oplus \mathbf{a}\, (\mathbf{b}\, \mathbf{c}) \oplus \mathbf{b}\, (\mathbf{a}\, \mathbf{c}) \oplus \mathbf{b}\, (\mathbf{b}\, \mathbf{c}) \\
(\lambda x.\, (x \stackrel{\bullet}{=} \mathbf{c}\, \mathbf{b}\, y); \mathbf{d}\, y\, z)\, (\mathbf{c}\, z\, \mathbf{a}) & \to^* & \mathbf{d}\, \mathbf{a}\, \mathbf{b} \\
(\lambda x.\, (x \stackrel{\bullet}{=} \mathbf{c}\, \mathbf{b}\, y); \mathbf{d}\, y\, z)\, (\mathbf{c}\, \mathbf{a}\, z) & \to^* & \mathtt{fail}
\end{array}
$$

Ejemplos

$$(\lambda x.\,((x \doteq \mathbf{0}); \mathbf{none}) \oplus \nu y.\,\mathbf{just}\,((x \doteq \mathbf{s}\,y); y))\,(\mathbf{s}\,(\mathbf{s}\,\mathbf{0}))$$

Ejemplos

$$
\begin{aligned}
& (\lambda x. ((x \doteq \mathbf{0}); \mathbf{none}) \oplus \nu y. \mathbf{just}\,((x \doteq \mathbf{s}\,y); y))\,(\mathbf{s}\,(\mathbf{s}\,\mathbf{0})) \\
\xrightarrow{\texttt{alloc}} \; & (\lambda^{\ell} x. ((x \doteq \mathbf{0}); \mathbf{none}) \oplus \nu y. \mathbf{just}\,((x \doteq \mathbf{s}\,y); y))\,(\mathbf{s}\,(\mathbf{s}\,\mathbf{0}))
\end{aligned}
$$

Ejemplos

$$\begin{aligned}
& (\lambda x.\, ((x \mathrel{\dot{=}} \mathbf{0}); \mathbf{none}) \oplus \nu y.\, \mathbf{just}\, ((x \mathrel{\dot{=}} \mathbf{s}\, y); y))\, (\mathbf{s}\, (\mathbf{s}\, \mathbf{0})) \\
\xrightarrow{\texttt{alloc}}\ & (\lambda^{\ell} x.\, ((x \mathrel{\dot{=}} \mathbf{0}); \mathbf{none}) \oplus \nu y.\, \mathbf{just}\, ((x \mathrel{\dot{=}} \mathbf{s}\, y); y))\, (\mathbf{s}\, (\mathbf{s}\, \mathbf{0})) \\
\xrightarrow{\texttt{beta}}\ & ((\mathbf{s}\, (\mathbf{s}\, \mathbf{0}) \mathrel{\dot{=}} \mathbf{0}); \mathbf{none}) \oplus \nu y.\, \mathbf{just}\, ((\mathbf{s}\, (\mathbf{s}\, \mathbf{0}) \mathrel{\dot{=}} \mathbf{s}\, y); y)
\end{aligned}$$

Ejemplos

$$
\begin{array}{rl}
& (\lambda x.\, ((x \overset{\bullet}{=} \mathbf{0}); \mathbf{none}) \oplus \nu y.\, \mathbf{just}\, ((x \overset{\bullet}{=} \mathbf{s}\, y); y))\, (\mathbf{s}\, (\mathbf{s}\, \mathbf{0})) \\
\xrightarrow{\texttt{alloc}} & (\lambda^{\ell} x.\, ((x \overset{\bullet}{=} \mathbf{0}); \mathbf{none}) \oplus \nu y.\, \mathbf{just}\, ((x \overset{\bullet}{=} \mathbf{s}\, y); y))\, (\mathbf{s}\, (\mathbf{s}\, \mathbf{0})) \\
\xrightarrow{\texttt{beta}} & ((\mathbf{s}\, (\mathbf{s}\, \mathbf{0}) \overset{\bullet}{=} \mathbf{0}); \mathbf{none}) \oplus \nu y.\, \mathbf{just}\, ((\mathbf{s}\, (\mathbf{s}\, \mathbf{0}) \overset{\bullet}{=} \mathbf{s}\, y); y) \\
\xrightarrow{\texttt{fail}} & \nu y.\, \mathbf{just}\, ((\mathbf{s}\, (\mathbf{s}\, \mathbf{0}) \overset{\bullet}{=} \mathbf{s}\, y); y)
\end{array}
$$

Ejemplos

$$
\begin{array}{rl}
 & (\lambda x.\, ((x \stackrel{\bullet}{=} \mathbf{0}); \mathbf{none}) \oplus \nu y.\, \mathbf{just}\, ((x \stackrel{\bullet}{=} \mathbf{s}\, y); y))\, (\mathbf{s}\, (\mathbf{s}\, \mathbf{0})) \\
\xrightarrow{\texttt{alloc}} & (\lambda^{\ell} x.\, ((x \stackrel{\bullet}{=} \mathbf{0}); \mathbf{none}) \oplus \nu y.\, \mathbf{just}\, ((x \stackrel{\bullet}{=} \mathbf{s}\, y); y))\, (\mathbf{s}\, (\mathbf{s}\, \mathbf{0})) \\
\xrightarrow{\texttt{beta}} & ((\mathbf{s}\, (\mathbf{s}\, \mathbf{0}) \stackrel{\bullet}{=} \mathbf{0}); \mathbf{none}) \oplus \nu y.\, \mathbf{just}\, ((\mathbf{s}\, (\mathbf{s}\, \mathbf{0}) \stackrel{\bullet}{=} \mathbf{s}\, y); y) \\
\xrightarrow{\texttt{fail}} & \nu y.\, \mathbf{just}\, ((\mathbf{s}\, (\mathbf{s}\, \mathbf{0}) \stackrel{\bullet}{=} \mathbf{s}\, y); y) \\
\xrightarrow{\texttt{fresh}} & \mathbf{just}\, ((\mathbf{s}\, (\mathbf{s}\, \mathbf{0}) \stackrel{\bullet}{=} \mathbf{s}\, z); z)
\end{array}
$$

Ejemplos

$$
\begin{array}{rl}
& (\lambda x.\,((x \stackrel{\bullet}{=} \mathbf{0}); \mathbf{none}) \oplus \nu y.\,\mathbf{just}\,((x \stackrel{\bullet}{=} \mathbf{s}\,y); y))\,(\mathbf{s}\,(\mathbf{s}\,\mathbf{0})) \\
\xrightarrow{\texttt{alloc}} & (\lambda^{\ell} x.\,((x \stackrel{\bullet}{=} \mathbf{0}); \mathbf{none}) \oplus \nu y.\,\mathbf{just}\,((x \stackrel{\bullet}{=} \mathbf{s}\,y); y))\,(\mathbf{s}\,(\mathbf{s}\,\mathbf{0})) \\
\xrightarrow{\texttt{beta}} & ((\mathbf{s}\,(\mathbf{s}\,\mathbf{0}) \stackrel{\bullet}{=} \mathbf{0}); \mathbf{none}) \oplus \nu y.\,\mathbf{just}\,((\mathbf{s}\,(\mathbf{s}\,\mathbf{0}) \stackrel{\bullet}{=} \mathbf{s}\,y); y) \\
\xrightarrow{\texttt{fail}} & \nu y.\,\mathbf{just}\,((\mathbf{s}\,(\mathbf{s}\,\mathbf{0}) \stackrel{\bullet}{=} \mathbf{s}\,y); y) \\
\xrightarrow{\texttt{fresh}} & \mathbf{just}\,((\mathbf{s}\,(\mathbf{s}\,\mathbf{0}) \stackrel{\bullet}{=} \mathbf{s}\,z); z) \\
\xrightarrow{\texttt{unif}} & \mathbf{just}\,(\mathbf{ok}; \mathbf{s}\,\mathbf{0})
\end{array}
$$

Ejemplos

$$
\begin{array}{rl}
& (\lambda x.\,((x \stackrel{\bullet}{=} \mathbf{0}); \mathbf{none}) \oplus \nu y.\,\mathbf{just}\,((x \stackrel{\bullet}{=} \mathsf{s}\,y); y))\,(\mathsf{s}\,(\mathsf{s}\,\mathbf{0})) \\
\xrightarrow{\texttt{alloc}} & (\lambda^{\ell} x.\,((x \stackrel{\bullet}{=} \mathbf{0}); \mathbf{none}) \oplus \nu y.\,\mathbf{just}\,((x \stackrel{\bullet}{=} \mathsf{s}\,y); y))\,(\mathsf{s}\,(\mathsf{s}\,\mathbf{0})) \\
\xrightarrow{\texttt{beta}} & ((\mathsf{s}\,(\mathsf{s}\,\mathbf{0}) \stackrel{\bullet}{=} \mathbf{0}); \mathbf{none}) \oplus \nu y.\,\mathbf{just}\,((\mathsf{s}\,(\mathsf{s}\,\mathbf{0}) \stackrel{\bullet}{=} \mathsf{s}\,y); y) \\
\xrightarrow{\texttt{fail}} & \nu y.\,\mathbf{just}\,((\mathsf{s}\,(\mathsf{s}\,\mathbf{0}) \stackrel{\bullet}{=} \mathsf{s}\,y); y) \\
\xrightarrow{\texttt{fresh}} & \mathbf{just}\,((\mathsf{s}\,(\mathsf{s}\,\mathbf{0}) \stackrel{\bullet}{=} \mathsf{s}\,z); z) \\
\xrightarrow{\texttt{unif}} & \mathbf{just}\,(\mathbf{ok}; \mathsf{s}\,\mathbf{0}) \\
\xrightarrow{\texttt{seq}} & \mathbf{just}\,(\mathsf{s}\,\mathbf{0})
\end{array}
$$

Tipado del cálculo-$\lambda^{\mathtt{U}}$

$$A ::= \alpha \mid A \to A$$

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}\ \mathtt{t\text{-}var} \qquad \frac{}{\Gamma \vdash \mathbf{c} : \mathcal{T}_{\mathbf{c}}}\ \mathtt{t\text{-}cons}$$

$$\frac{\Gamma, x : A \vdash P : B}{\Gamma \vdash \lambda x.\, P : A \to B}\ \mathtt{t\text{-}lam} \qquad \frac{\Gamma, x : A \vdash P : B}{\Gamma \vdash \lambda^{\ell} x.\, P : A \to B}\ \mathtt{t\text{-}laml}$$

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash s : A}{\Gamma \vdash t\, s : B}\ \mathtt{t\text{-}app} \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash s : A}{\Gamma \vdash t \overset{\bullet}{=} s : \mathcal{T}_{\mathbf{ok}}}\ \mathtt{t\text{-}unif}$$

$$\frac{\Gamma \vdash t : \mathcal{T}_{\mathbf{ok}} \qquad \Gamma \vdash s : A}{\Gamma \vdash t; s : A}\ \mathtt{t\text{-}seq} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \nu x.\, t : B}\ \mathtt{t\text{-}fresh}$$

$$\frac{\Gamma \vdash t_1 : A \qquad \dots \qquad \Gamma \vdash t_n : A}{\Gamma \vdash t_1 \oplus \dots \oplus t_n : A}\ \mathtt{t\text{-}alt}$$

Propiedades básicas

Sea X un término o programa.

Debilitamiento

Si $\Gamma \vdash X : A$, entonces $\Gamma, y : B \vdash X : A$.

Fortalecimiento

Sea $\Gamma, y : B \vdash X : A$, y supongamos que $y \notin \mathrm{fv}(X)$. Entonces $\Gamma \vdash X : A$.

Preservación de tipos

Sea $\Gamma \vdash P : A$ y supongamos que $P \to Q$. Entonces $\Gamma' \vdash Q : A$ para cierto contexto $\Gamma' \supseteq \Gamma$, que extiende a Γ con las variables simbólicas introducidas.

Propiedades básicas

Sea X un término o programa.

Debilitamiento

Si $\Gamma \vdash X : A$, entonces $\Gamma, y : B \vdash X : A$.

Fortalecimiento

Sea $\Gamma, y : B \vdash X : A$, y supongamos que $y \notin \mathrm{fv}(X)$. Entonces $\Gamma \vdash X : A$.

Preservación de tipos

Sea $\Gamma \vdash P : A$ y supongamos que $P \to Q$. Entonces $\Gamma' \vdash Q : A$ para cierto contexto $\Gamma' \supseteq \Gamma$, que extiende a Γ con las variables simbólicas introducidas.

El caso interesante es el de la unificación, porque requiere probar preservación de tipos para el algoritmo de unificación.

Semántica denotacional para el cálculo-$\lambda^{\mathtt{U}}$

Interpretación de los tipos

A cada tipo se le asocia un conjunto:

$$
\begin{aligned}
[\![\alpha]\!] &\stackrel{\text{def}}{=} \mathsf{S}_\alpha \\
[\![A \to B]\!] &\stackrel{\text{def}}{=} \mathcal{P}([\![B]\!])^{[\![A]\!]}
\end{aligned}
$$

Ejemplo

Si $\mathsf{S}_\alpha = \mathbb{R}$ entonces $[\![\alpha \to \alpha]\!] = \mathcal{P}(\mathbb{R})^{\mathbb{R}}$.

Dominios no vacíos

Exigimos que $\mathsf{S}_\alpha \neq \emptyset$.
Si $\mathsf{S}_\alpha = \emptyset$, la semántica no podría ser correcta.
Por ejemplo, si $x : \alpha$, tendríamos $[\![\mathbf{ok}]\!] = [\![x \stackrel{\bullet}{=} x]\!] = \emptyset = [\![\mathtt{fail}]\!]$.

Como consecuencia $[\![A]\!] \neq \emptyset$.

Semántica denotacional para el cálculo-$\lambda^{\mathtt{U}}$

Decisiones técnicas

1. **Términos como conjuntos.**
   Cada término o programa de tipo A se interpreta como un subconjunto de $[\![A]\!]$.

Decisiones técnicas

1. **Términos como conjuntos.**
   Cada término o programa de tipo A se interpreta como un subconjunto de $[\![A]\!]$.
2. **Valores como conjuntos unitarios.**

Decisiones técnicas

1. **Términos como conjuntos.**
   Cada término o programa de tipo A se interpreta como un subconjunto de $[\![A]\!]$.
2. **Valores como conjuntos unitarios.**
   Si $\#([\![\mathtt{v}]\!]) \neq 1$, la semántica no resultaría correcta.

   Por ejemplo, si $[\![\mathtt{v}]\!] = \emptyset$,
   tendríamos que $[\![(\lambda^{\ell} x.\, \mathbf{c})\, \mathtt{v}]\!] = \emptyset$,
   pero querríamos que $[\![(\lambda^{\ell} x.\, \mathbf{c})\, \mathtt{v}]\!] = [\![\mathbf{c}]\!] \neq \emptyset$.

Semántica denotacional para el cálculo-$\lambda^{\mathtt{U}}$

Decisiones técnicas

1. **Términos como conjuntos.**
   Cada término o programa de tipo A se interpreta como un subconjunto de $[\![A]\!]$.
2. **Valores como conjuntos unitarios.**
   Si $\#([\![\mathrm{v}]\!]) \neq 1$, la semántica no resultaría correcta.

   Por ejemplo, si $[\![\mathrm{v}]\!] = \emptyset$,
   tendríamos que $[\![(\lambda^{\ell} x.\, \mathbf{c})\, \mathrm{v}]\!] = \emptyset$,
   pero querríamos que $[\![(\lambda^{\ell} x.\, \mathbf{c})\, \mathrm{v}]\!] = [\![\mathbf{c}]\!] \neq \emptyset$.

   Más aún, si $[\![\mathrm{v}]\!]$ es una función, $[\![\mathrm{v}]\!](a)$ debe ser a su vez unitario para todo a.

Interpretación de los términos

Una *asignación de variables* es una función ρ que a cada variable $x : A$ le asocia un elemento $\rho(x) \in [\![A]\!]$.

Cada término o programa de tipo A se interpreta como un subconjunto de $[\![A]\!]$.

$$
\begin{array}{rcll}
[\![x]\!]_\rho & \stackrel{\text{def}}{=} & \{\rho(x)\} & \\
[\![\mathbf{c}]\!]_\rho & \stackrel{\text{def}}{=} & \{\mathsf{R}_{\mathbf{c}}\} & \text{que cumple el principio de } \textit{no confusión} \\
[\![\lambda x.\, P]\!]_\rho & \stackrel{\text{def}}{=} & \{f\} \text{ con } f : [\![A]\!] \to \mathcal{P}([\![B]\!]) \text{ dado por } f(a) = [\![P]\!]_{\rho[x \mapsto a]} & \\
[\![\lambda^\ell x.\, P]\!]_\rho & \stackrel{\text{def}}{=} & \{f\} \text{ con } f : [\![A]\!] \to \mathcal{P}([\![B]\!]) \text{ dado por } f(a) = [\![P]\!]_{\rho[x \mapsto a]} & \\
[\![t\, s]\!]_\rho & \stackrel{\text{def}}{=} & \{b \mid f \in [\![t]\!]_\rho, a \in [\![s]\!]_\rho, b \in f(a)\} & \\
[\![t \stackrel{\bullet}{=} s]\!]_\rho & \stackrel{\text{def}}{=} & \{\mathsf{R}_{\mathbf{ok}} \mid a \in [\![t]\!]_\rho, b \in [\![s]\!]_\rho, a = b\} & \\
[\![t; s]\!]_\rho & \stackrel{\text{def}}{=} & \{a \mid b \in [\![t]\!]_\rho, a \in [\![s]\!]_\rho\} & \\
[\![\nu x.\, t]\!]_\rho & \stackrel{\text{def}}{=} & \{b \mid a \in [\![A]\!], b \in [\![t]\!]_{\rho[x \mapsto a]}\} & \\
[\![t_1 \oplus \ldots \oplus t_n]\!]_\rho & \stackrel{\text{def}}{=} & [\![t_1]\!]_\rho \cup \ldots \cup [\![t_n]\!]_\rho &
\end{array}
$$

Además, si X es un término o programa, $[\![X]\!] = \bigcup_\rho [\![X]\!]_\rho$.

Ejemplo

Sea $[\![\mathsf{Bool}]\!] = \{0, 1\}$, donde $[\![\mathbf{T}]\!] = \{1\}$ y $[\![\mathbf{F}]\!] = \{0\}$. Y sea

$$\mathsf{or} \quad \stackrel{\text{def}}{=} \quad \lambda^{\ell} x. \lambda^{\ell'} y. \ (x \stackrel{\bullet}{=} \mathbf{T}; \mathbf{T}) \oplus (y \stackrel{\bullet}{=} \mathbf{T}; \mathbf{T}) \oplus (x \stackrel{\bullet}{=} \mathbf{F}; ((y \stackrel{\bullet}{=} \mathbf{F}); \mathbf{F}))$$

Ejemplo

Sea $[\![\mathsf{Bool}]\!] = \{0, 1\}$, donde $[\![\mathbf{T}]\!] = \{1\}$ y $[\![\mathbf{F}]\!] = \{0\}$. Y sea

$$\mathsf{or} \stackrel{\text{def}}{=} \lambda^{\ell} x.\, \lambda^{\ell'} y.\ (x \stackrel{\bullet}{=} \mathbf{T}; \mathbf{T}) \oplus (y \stackrel{\bullet}{=} \mathbf{T}; \mathbf{T}) \oplus (x \stackrel{\bullet}{=} \mathbf{F}; ((y \stackrel{\bullet}{=} \mathbf{F}); \mathbf{F}))$$

Entonces:

$$\begin{aligned} & [\![\mathsf{or}\ \mathbf{T}\ \mathbf{F}]\!] \\ = \ & [\![(x \stackrel{\bullet}{=} \mathbf{T}; \mathbf{T}) \oplus (y \stackrel{\bullet}{=} \mathbf{T}; \mathbf{T}) \oplus (x \stackrel{\bullet}{=} \mathbf{F}; ((y \stackrel{\bullet}{=} \mathbf{F}); \mathbf{F}))]\!]_{[x \mapsto 1][y \mapsto 0]} \\ = \ & \{1\} \end{aligned}$$

Ejemplo

Sea $[\![\mathrm{Bool}]\!] = \{0,1\}$, donde $[\![\mathbf{T}]\!] = \{1\}$ y $[\![\mathbf{F}]\!] = \{0\}$. Y sea

$$\mathrm{or} \stackrel{\mathrm{def}}{=} \lambda^{\ell} x.\, \lambda^{\ell'} y.\ (x \stackrel{\bullet}{=} \mathbf{T}; \mathbf{T}) \oplus (y \stackrel{\bullet}{=} \mathbf{T}; \mathbf{T}) \oplus (x \stackrel{\bullet}{=} \mathbf{F}; ((y \stackrel{\bullet}{=} \mathbf{F}); \mathbf{F}))$$

Entonces:

$$\begin{aligned} & [\![\mathrm{or}\ \mathbf{T}\ \mathbf{F}]\!] \\ = \ & [\![(x \stackrel{\bullet}{=} \mathbf{T}; \mathbf{T}) \oplus (y \stackrel{\bullet}{=} \mathbf{T}; \mathbf{T}) \oplus (x \stackrel{\bullet}{=} \mathbf{F}; ((y \stackrel{\bullet}{=} \mathbf{F}); \mathbf{F}))]\!]_{[x \mapsto 1][y \mapsto 0]} \\ = \ & \{1\} \end{aligned}$$

donde, más en general, si $\rho = [x \mapsto a][y \mapsto b]$, se tiene:

$$\begin{aligned} & [\![(x \stackrel{\bullet}{=} \mathbf{T}; \mathbf{T}) \oplus (y \stackrel{\bullet}{=} \mathbf{T}; \mathbf{T}) \oplus (x \stackrel{\bullet}{=} \mathbf{F}; ((y \stackrel{\bullet}{=} \mathbf{F}); \mathbf{F}))]\!]_{\rho} \\ = \ & [\![x \stackrel{\bullet}{=} \mathbf{T}; \mathbf{T}]\!]_{\rho} \cup [\![y \stackrel{\bullet}{=} \mathbf{T}; \mathbf{T}]\!]_{\rho} \cup [\![x \stackrel{\bullet}{=} \mathbf{F}; ((y \stackrel{\bullet}{=} \mathbf{F}); \mathbf{F})]\!]_{\rho} \\ = \ & \{1 \mid a = 1\} \cup \{1 \mid b = 1\} \cup \{0 \mid a = 0, b = 0\} \end{aligned}$$

Propiedades de la semántica denotacional

Irrelevancia

Sea X un término o programa tipable, es decir, $\Gamma \vdash X : A$. Si ρ, ρ' son asignaciones de variables que coinciden en $\mathsf{fv}(X)$, entonces $[\![X]\!]_\rho = [\![X]\!]_{\rho'}$.

Composicionalidad

1. $[\![P \oplus Q]\!]_\rho = [\![P]\!]_\rho \cup [\![Q]\!]_\rho$.
2. Si W es un contexto cuyo agujero tiene tipo A, entonces $[\![\mathsf{W}\langle t\rangle]\!]_\rho = \{b \mid a \in [\![t]\!]_\rho,\ b \in [\![\mathsf{W}]\!]_{\rho[\Box \mapsto a]}\}$.

Correctitud

$P \to Q$ implica $[\![P]\!] = [\![Q]\!]$

Teorema (Correctitud débil)

Sea $\Gamma \vdash P : A$ y $P \to Q$. Entonces:

$$[\![P]\!] \supseteq [\![Q]\!]$$

Además, la inclusión es una igualdad para cualquier regla de reducción a excepción de la regla `fail`.

Ejemplo

Consideremos la reducción

$$(\lambda p.\, \nu x.\, \nu y.\, (p \overset{\bullet}{=} \boldsymbol{t}\, x\, y); y)\, (\boldsymbol{t}\, \mathbf{0}\, \mathbf{5}) \to^{*} \mathbf{5}$$

Ejemplo

Consideremos la reducción

$$(\lambda p.\, \nu x.\, \nu y.\, (p \stackrel{\bullet}{=} \boldsymbol{t}\, x\, y); y)\, (\boldsymbol{t}\, \mathbf{0}\, \mathbf{5}) \to^* \mathbf{5}$$

Si $[\![\mathtt{Nat}]\!] = \mathbb{N}$, $[\![\mathtt{Tuple}]\!] = [\![\mathtt{Nat}]\!] \times [\![\mathtt{Nat}]\!] = \mathbb{N} \times \mathbb{N}$, los constructores $\mathbf{0} : \mathtt{Nat}$, $\mathbf{5} : \mathtt{Nat}$ se interpretan de la manera obvia y $\boldsymbol{t} : \mathtt{Nat} \to \mathtt{Nat} \to \mathtt{Tuple}$ es la función constructora de pares, Consideremos la reducción

$$(\lambda p.\, \nu x.\, \nu y.\, (p \stackrel{\bullet}{=} \boldsymbol{t}\, x\, y); y)\, (\boldsymbol{t}\, \mathbf{0}\, \mathbf{5}) \to^* \mathbf{5}$$

Si $[\![\mathtt{Nat}]\!] = \mathbb{N}$, $[\![\mathtt{Tuple}]\!] = [\![\mathtt{Nat}]\!] \times [\![\mathtt{Nat}]\!] = \mathbb{N} \times \mathbb{N}$, los constructores $\mathbf{0} : \mathtt{Nat}$, $\mathbf{5} : \mathtt{Nat}$ se interpretan de la manera obvia y $\boldsymbol{t} : \mathtt{Nat} \to \mathtt{Nat} \to \mathtt{Tuple}$ es la función constructora de pares, entonces para cualquier asignación de variables ρ:

$$\begin{array}{rl}
 & [\![(\lambda p.\, \nu x.\, \nu y.\, (p \stackrel{\bullet}{=} \boldsymbol{t}\, x\, y); y)\, (\boldsymbol{t}\, \mathbf{0}\, \mathbf{5})]\!]_\rho \\
= & \{b \mid f \in [\![\lambda p.\, \nu x.\, \nu y.\, (p \stackrel{\bullet}{=} \boldsymbol{t}\, x\, y); y]\!]_\rho, a \in [\![\boldsymbol{t}\, \mathbf{0}\, \mathbf{5}]\!]_\rho, b \in f(a)\} \\
= & \{b \mid a \in [\![\boldsymbol{t}\, \mathbf{0}\, \mathbf{5}]\!]_\rho, b \in [\![\nu x.\, \nu y.\, (p \stackrel{\bullet}{=} \boldsymbol{t}\, x\, y); y]\!]_{\rho[p \mapsto a]}\} \\
= & \{b \mid a \in [\![\boldsymbol{t}\, \mathbf{0}\, \mathbf{5}]\!]_\rho, n, m \in \mathbb{N}, b \in [\![(p \stackrel{\bullet}{=} \boldsymbol{t}\, x\, y); y]\!]_{\rho[p \mapsto a][x \mapsto n][y \mapsto m]}\} \\
= & \{m \mid n, m \in \mathbb{N}, a \in [\![\boldsymbol{t}\, \mathbf{0}\, \mathbf{5}]\!]_\rho, c \in [\![p \stackrel{\bullet}{=} \boldsymbol{t}\, x\, y]\!]_{\rho[p \mapsto a][x \mapsto n][y \mapsto m]}\} \\
= & \{m \mid n, m \in \mathbb{N}, a \in \{(0,5)\}, a = (n,m)\} \\
= & \{m \mid n \in \{0\}, m \in \{5\}, a \in \{(0,5)\}\} \\
= & \{5\} \\
= & [\![\mathbf{5}]\!]_\rho
\end{array}$$

La correctitud es débil

Si $t \xrightarrow{\texttt{fail}} s$, la inclusión $[\![t]\!] \supseteq [\![s]\!]$ puede ser estricta.

Ejemplo

$$
\begin{array}{rl}
 & (\lambda^{\ell_1} x.\, x \stackrel{\bullet}{=} \lambda^{\ell_2} x.\, x) \xrightarrow{\texttt{fail}} \texttt{fail} \\
\text{pero} & [\![\lambda^{\ell_1} x.\, x \stackrel{\bullet}{=} \lambda^{\ell_2} x.\, x]\!] = \{\mathsf{R}_{\mathbf{ok}}\} \supsetneq \emptyset = [\![\texttt{fail}]\!]
\end{array}
$$

Completitud

Completitud

$[\![P]\!] = [\![Q]\!]$ implica $P \leftrightarrow^* Q$.

Completitud

Completitud

$[\![P]\!] = [\![Q]\!]$ implica $P \leftrightarrow^* Q$.

Ejemplos que evidencian que no se verifica completitud

- **SO distingue locaciones, SD no.**

$$
\begin{array}{ll}
 & (\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_1} x.\, P) \not\leftrightarrow^* (\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_2} x.\, P) \\
\text{pues} & (\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_1} x.\, P) \xrightarrow{\texttt{unif}} \mathbf{ok} \\
\text{y} & (\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_2} x.\, P) \xrightarrow{\texttt{fail}} \texttt{fail}
\end{array}
$$

Completitud

Completitud

$[\![P]\!] = [\![Q]\!]$ implica $P \leftrightarrow^* Q$.

Ejemplos que evidencian que no se verifica completitud

- **SO distingue locaciones, SD no.**

$$
\begin{array}{ll}
 & (\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_1} x.\, P) \not\leftrightarrow^* (\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_2} x.\, P) \\
\text{pues} & (\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_1} x.\, P) \xrightarrow{\texttt{unif}} \textbf{ok} \\
\text{y} & (\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_2} x.\, P) \xrightarrow{\texttt{fail}} \texttt{fail} \\
\text{pero} & [\![\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_1} x.\, P]\!] = [\![\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_2} x.\, P]\!]
\end{array}
$$

Completitud

Completitud

$[\![P]\!] = [\![Q]\!]$ implica $P \leftrightarrow^* Q$.

Ejemplos que evidencian que no se verifica completitud

- **SO distingue locaciones, SD no.**

$$
\begin{array}{ll}
 & (\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_1} x.\, P) \not\leftrightarrow^* (\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_2} x.\, P) \\
\text{pues} & (\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_1} x.\, P) \xrightarrow{\texttt{unif}} \mathbf{ok} \\
\text{y} & (\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_2} x.\, P) \xrightarrow{\texttt{fail}} \texttt{fail} \\
\text{pero} & [\![\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_1} x.\, P]\!] = [\![\lambda^{\ell_1} x.\, P \stackrel{\bullet}{=} \lambda^{\ell_2} x.\, P]\!]
\end{array}
$$

- **SO sólo unifica valores, SD no tiene esa restricción.**

$$(x\, y \stackrel{\bullet}{=} \mathbf{c}) \not\leftrightarrow^* \mathbf{ok}$$

Completitud

Completitud

$[\![P]\!] = [\![Q]\!]$ implica $P \leftrightarrow^* Q$.

Ejemplos que evidencian que no se verifica completitud

- **SO distingue locaciones, SD no.**

$$
\begin{array}{ll}
 & (\lambda^{\ell_1}x.\,P \stackrel{\bullet}{=} \lambda^{\ell_1}x.\,P) \not\leftrightarrow^* (\lambda^{\ell_1}x.\,P \stackrel{\bullet}{=} \lambda^{\ell_2}x.\,P) \\
\text{pues} & (\lambda^{\ell_1}x.\,P \stackrel{\bullet}{=} \lambda^{\ell_1}x.\,P) \xrightarrow{\texttt{unif}} \textbf{ok} \\
\text{y} & (\lambda^{\ell_1}x.\,P \stackrel{\bullet}{=} \lambda^{\ell_2}x.\,P) \xrightarrow{\texttt{fail}} \texttt{fail} \\
\text{pero} & [\![\lambda^{\ell_1}x.\,P \stackrel{\bullet}{=} \lambda^{\ell_1}x.\,P]\!] = [\![\lambda^{\ell_1}x.\,P \stackrel{\bullet}{=} \lambda^{\ell_2}x.\,P]\!]
\end{array}
$$

- **SO sólo unifica valores, SD no tiene esa restricción.**

$$
\begin{array}{ll}
 & (x\,y \stackrel{\bullet}{=} \mathbf{c}) \not\leftrightarrow^* \textbf{ok} \\
\text{pero} & [\![x\,y \stackrel{\bullet}{=} \mathbf{c}]\!] = \{\mathsf{R}_{\textbf{ok}}\} = [\![\textbf{ok}]\!] \\
\text{pues} & x\,y \stackrel{\bullet}{=} \mathbf{c} \text{ tiene unificador}
\end{array}
$$

Ejemplos que evidencian que no se verifica completitud

- **SO es sensible a términos trabados, SD no.**

$$((x\,y);\mathbf{ok}) \not\leftrightarrow^{*} \mathbf{ok}$$

Completitud

Ejemplos que evidencian que no se verifica completitud

- **SO es sensible a términos trabados, SD no.**

$$((x\,y); \mathbf{ok}) \not\leftrightarrow^* \mathbf{ok}$$

$$\text{pero} \quad [\![(x\,y); \mathbf{ok}]\!] = \{\mathsf{R}_{\mathbf{ok}}\} = [\![\mathbf{ok}]\!]$$

Ejemplos que evidencian que no se verifica completitud

- **SO es sensible a términos trabados, SD no.**

$$((x\,y);\mathbf{ok}) \not\leftrightarrow^* \mathbf{ok}$$
$$\text{pero} \quad [\![(x\,y);\mathbf{ok}]\!] = \{\mathsf{R}_{\mathbf{ok}}\} = [\![\mathbf{ok}]\!]$$

- **SO admite multiplicidad de resultados, SD no.**

$$\mathbf{c} \oplus \mathbf{c} \not\leftrightarrow^* \mathbf{c}$$

Completitud

Ejemplos que evidencian que no se verifica completitud

- **SO es sensible a términos trabados, SD no.**

$$
\begin{array}{rl}
 & ((x\,y); \mathbf{ok}) \not\leftrightarrow^{*} \mathbf{ok} \\
\text{pero} & [\![(x\,y); \mathbf{ok}]\!] = \{\mathrm{R}_{\mathbf{ok}}\} = [\![\mathbf{ok}]\!]
\end{array}
$$

- **SO admite multiplicidad de resultados, SD no.**

$$
\begin{array}{rl}
 & \mathbf{c} \oplus \mathbf{c} \not\leftrightarrow^{*} \mathbf{c} \\
\text{pero} & [\![\mathbf{c} \oplus \mathbf{c}]\!] = \{\mathrm{R}_{\mathbf{c}}\} = [\![\mathbf{c}]\!]
\end{array}
$$

Ejemplos que evidencian que no se verifica completitud

- **SO es sensible a términos trabados, SD no.**

$$((x\,y);\mathbf{ok}) \not\leftrightarrow^* \mathbf{ok}$$
$$\text{pero} \quad [\![(x\,y);\mathbf{ok}]\!] = \{\mathsf{R}_{\mathbf{ok}}\} = [\![\mathbf{ok}]\!]$$

- **SO admite multiplicidad de resultados, SD no.**

$$\mathbf{c} \oplus \mathbf{c} \not\leftrightarrow^* \mathbf{c}$$
$$\text{pero} \quad [\![\mathbf{c} \oplus \mathbf{c}]\!] = \{\mathsf{R}_{\mathbf{c}}\} = [\![\mathbf{c}]\!]$$

- **SO no es extensional, SD sí.**

$$\mathbf{c} \not\leftrightarrow^* \lambda x.\,\mathbf{c}\,x$$

Completitud

Ejemplos que evidencian que no se verifica completitud

- **SO es sensible a términos trabados, SD no.**

$$((x\,y);\mathbf{ok}) \not\leftrightarrow^* \mathbf{ok}$$
$$\text{pero} \quad [\![(x\,y);\mathbf{ok}]\!] = \{\mathrm{R}_{\mathbf{ok}}\} = [\![\mathbf{ok}]\!]$$

- **SO admite multiplicidad de resultados, SD no.**

$$\mathbf{c} \oplus \mathbf{c} \not\leftrightarrow^* \mathbf{c}$$
$$\text{pero} \quad [\![\mathbf{c} \oplus \mathbf{c}]\!] = \{\mathrm{R}_{\mathbf{c}}\} = [\![\mathbf{c}]\!]$$

- **SO no es extensional, SD sí.**

$$\mathbf{c} \not\leftrightarrow^* \lambda x.\,\mathbf{c}\,x$$
$$\text{pero} \quad [\![\mathbf{c}]\!] = [\![\lambda x.\,\mathbf{c}\,x]\!]$$

Alternativas

Para tratar de dar con una semántica correcta y completa exploramos dos alternativas.

Alternativa 1 — Semántica denotacional con memoria

- Incorpora una noción de estado (memoria).
- Una abstracción $\lambda x.\, P$ es una operación que reserva espacio en la memoria.
- Una abstracción alojada $\lambda^{\ell} x.\, P$ es un valor que representa un puntero a una posición de memoria ℓ.

Alternativas

Para tratar de dar con una semántica correcta y completa exploramos dos alternativas.

Alternativa 1 — Semántica denotacional con memoria

- Incorpora una noción de estado (memoria).
- Una abstracción $\lambda x.\, P$ es una operación que reserva espacio en la memoria.
- Una abstracción alojada $\lambda^{\ell} x.\, P$ es un valor que representa un puntero a una posición de memoria ℓ.
- Problema: no está claro cómo formular una semántica composicional, porque aparecen dependencias cíclicas en la memoria:

$$((x \stackrel{\bullet}{=} \lambda z.\, z); y\, \mathbf{c})\, ((y \stackrel{\bullet}{=} \lambda z.\, z); x\, \mathbf{c})$$

Alternativas

Alternativa 2 — Cálculo-$\lambda^{\mathtt{U}}$ con clausuras

- Hay *clausuras* $(\lambda x.\, P)[x_1 \backslash \mathtt{v}_1] \ldots [x_n \backslash \mathtt{v}_n]$ en lugar de abstracciones.
- Dos clausuras unifican si y sólo si son iguales, salvo permutación de los $[x_i \backslash \mathtt{v}_i]$.
- Probamos que esta variante del cálculo es confluente.

Alternativa 2 — Cálculo-$\lambda^{\mathtt{U}}$ con clausuras

- Hay *clausuras* $(\lambda x.\, P)[x_1\backslash\mathtt{v}_1]\ldots[x_n\backslash\mathtt{v}_n]$ en lugar de abstracciones.
- Dos clausuras unifican si y sólo si son iguales, salvo permutación de los $[x_i\backslash\mathtt{v}_i]$.
- Probamos que esta variante del cálculo es confluente.
- Problema: no está claro cómo definir el dominio de interpretación asociado a un tipo de la forma $A \to B$, porque una clausura de tipo $A \to B$ contiene datos de tipos posiblemente más grandes.

Conclusiones y trabajo futuro

En esta tesis:

- Presentamos un sistema de tipos para el cálculo-$\lambda^{\mathtt{U}}$.

Conclusiones y trabajo futuro

En esta tesis:

- Presentamos un sistema de tipos para el cálculo-$\lambda^{\mathtt{U}}$.
    - Demostramos que vale la preservación de tipos.

Conclusiones y trabajo futuro

En esta tesis:

- Presentamos un sistema de tipos para el cálculo-$\lambda^{\mathtt{U}}$.
  - Demostramos que vale la preservación de tipos.
- Propusimos una semántica denotacional para el cálculo-$\lambda^{\mathtt{U}}$.

Conclusiones y trabajo futuro

En esta tesis:

- Presentamos un sistema de tipos para el cálculo-$\lambda^{\mathtt{U}}$.
  - Demostramos que vale la preservación de tipos.
- Propusimos una semántica denotacional para el cálculo-$\lambda^{\mathtt{U}}$.
  - Demostramos que vale una versión débil de correctitud de la semántica operacional con respecto a la denotacional.

Conclusiones y trabajo futuro

En esta tesis:

- Presentamos un sistema de tipos para el cálculo-$\lambda^{\mathtt{U}}$.
  - Demostramos que vale la preservación de tipos.
- Propusimos una semántica denotacional para el cálculo-$\lambda^{\mathtt{U}}$.
  - Demostramos que vale una versión débil de correctitud de la semántica operacional con respecto a la denotacional.
  - La semántica operacional está lejos de ser completa con respecto a la denotacional.

Conclusiones y trabajo futuro

En esta tesis:

- Presentamos un sistema de tipos para el cálculo-$\lambda^{\mathtt{U}}$.
  - Demostramos que vale la preservación de tipos.
- Propusimos una semántica denotacional para el cálculo-$\lambda^{\mathtt{U}}$.
  - Demostramos que vale una versión débil de correctitud de la semántica operacional con respecto a la denotacional.
  - La semántica operacional está lejos de ser completa con respecto a la denotacional.
- Estudiamos dos alternativas, tratando de dar con una semántica correcta y completa.

Conclusiones y trabajo futuro

En esta tesis:

- Presentamos un sistema de tipos para el cálculo-$\lambda^{\mathtt{U}}$.
  - Demostramos que vale la preservación de tipos.
- Propusimos una semántica denotacional para el cálculo-$\lambda^{\mathtt{U}}$.
  - Demostramos que vale una versión débil de correctitud de la semántica operacional con respecto a la denotacional.
  - La semántica operacional está lejos de ser completa con respecto a la denotacional.
- Estudiamos dos alternativas, tratando de dar con una semántica correcta y completa.

Como trabajo futuro queda:

- Proponer una semántica denotacional para la cual la operacional resulte correcta y completa.

Conclusiones y trabajo futuro

En esta tesis:

- Presentamos un sistema de tipos para el cálculo-$\lambda^{\mathtt{U}}$.
  - Demostramos que vale la preservación de tipos.
- Propusimos una semántica denotacional para el cálculo-$\lambda^{\mathtt{U}}$.
  - Demostramos que vale una versión débil de correctitud de la semántica operacional con respecto a la denotacional.
  - La semántica operacional está lejos de ser completa con respecto a la denotacional.
- Estudiamos dos alternativas, tratando de dar con una semántica correcta y completa.

Como trabajo futuro queda:

- Proponer una semántica denotacional para la cual la operacional resulte correcta y completa.
- Relacionar el cálculo-$\lambda^{\mathtt{U}}$ con otros cálculos (ej. PPC).

¡Gracias!

¿Preguntas?