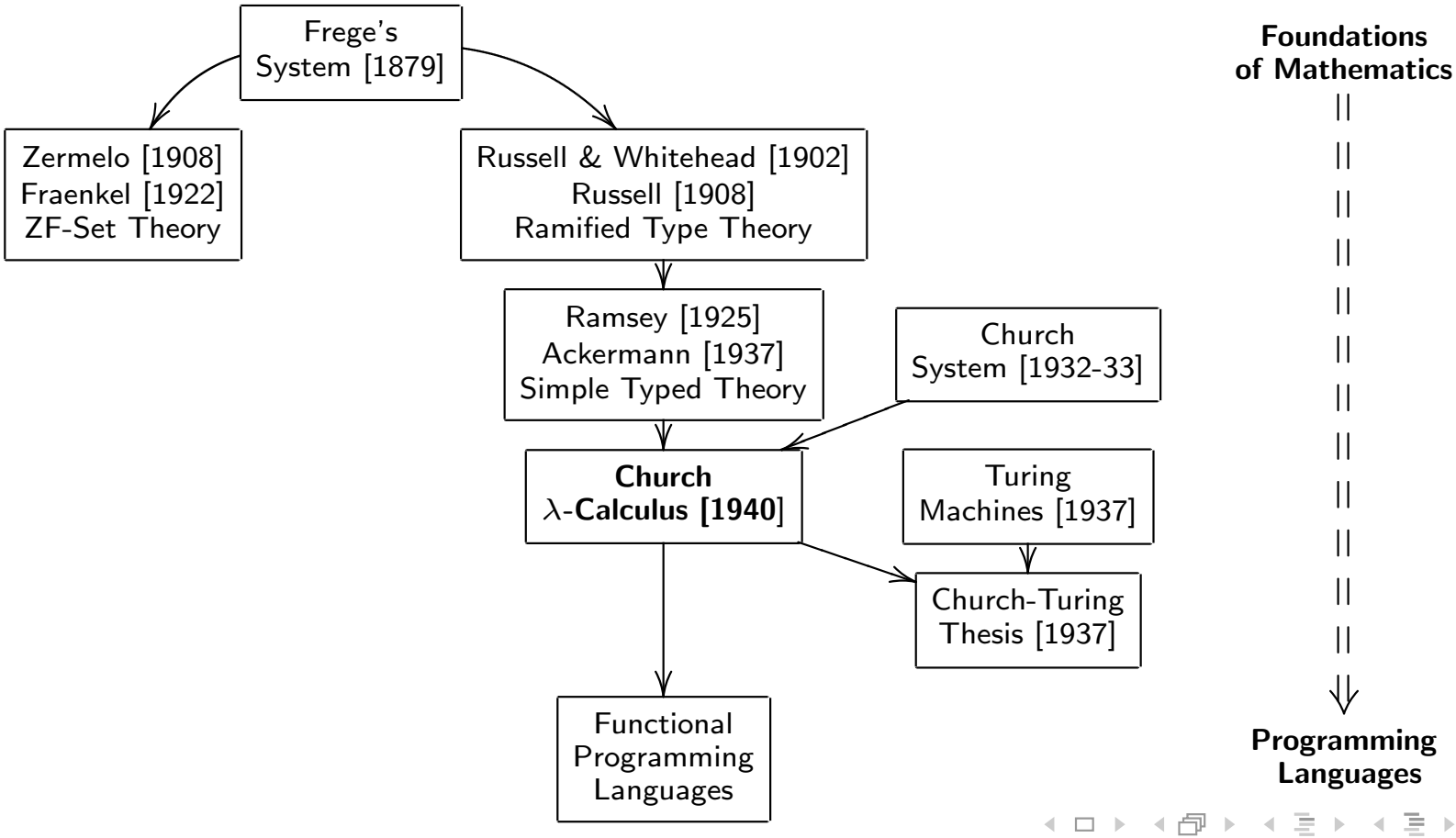
El paradigma de los Isomorfísmos de Tipos

Carlos Martínez Méndez

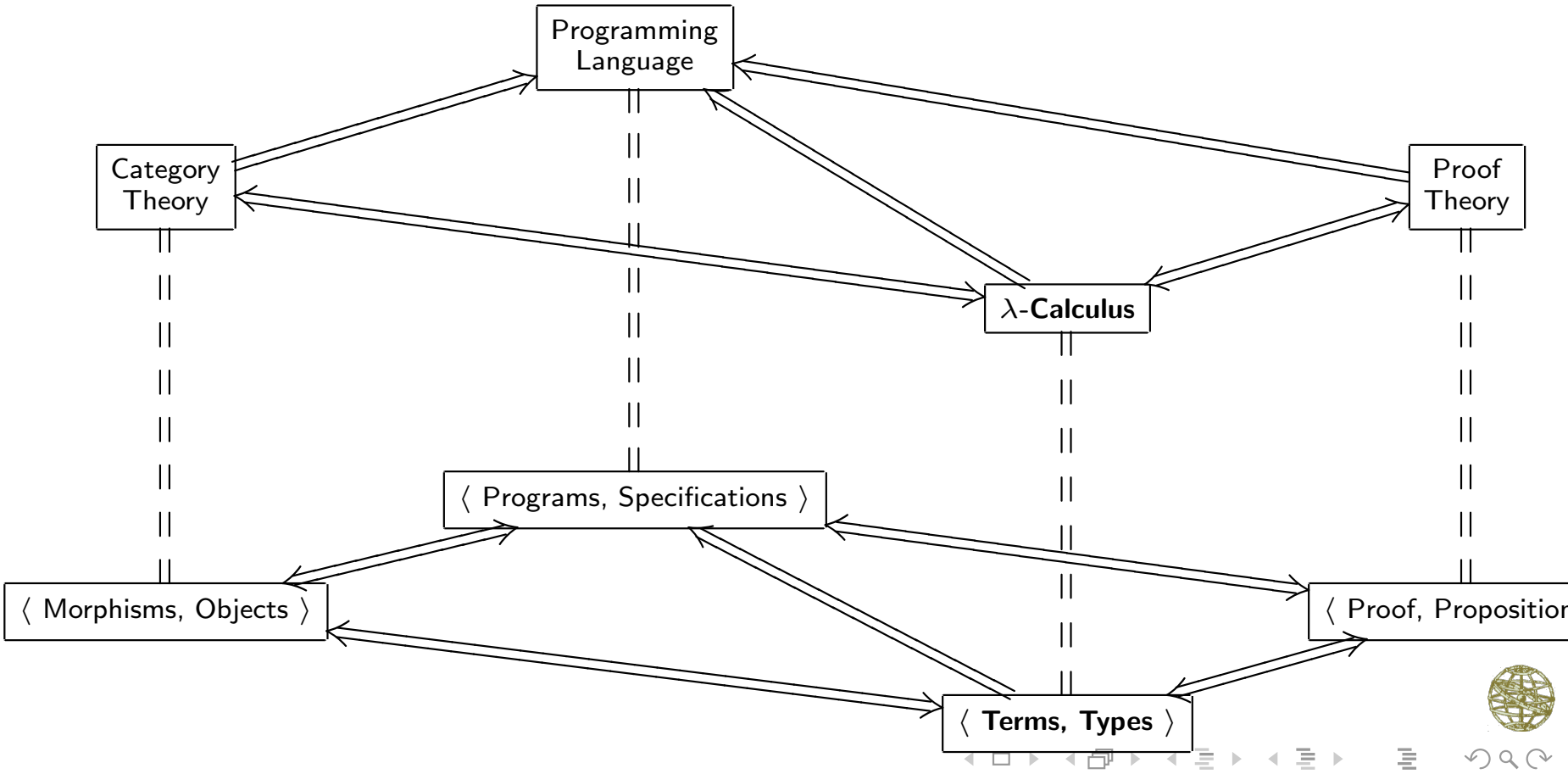
Escuela de Ingeniería en Computación
Facultad de Ciencias Físicas y Matemáticas
Universidad de Central de Chile

Abril 23, 2008

Fundamento Lógico en el Diseño de Lenguajes de Programación



El paradigma formal de un Lenguajes de Programación Funcional



La paradoja de Russell:

1879-93 Frege propone una base formal fundacional para la Matemática.

La paradoja de Russell:

1879-93 Frege propone una base formal fundacional para la Matemática.

- **[Problema]** La paradoja de Russell desadía la consistencia del sistema de Frege.

$$R = \{x | x \notin x\}$$

Q: ¿ Esta $R \in R$?

La paradoja de Russell:

1879-93 Frege propone una base formal fundacional para la Matemática.

- **[Problema]** La paradoja de Russell desadía la consistencia del sistema de Frege.

$$R = \{x | x \notin x\}$$

  Q: ¿ Esta $R \in R$?
- **[Solución]** Russell propone una solución a la paradoja acuñando el concepto de *tipos*: una forma de etiqueta sobre términos que restrige la expresividad de estos. Este nuevo sistema formal de Russell paso a llamarse *Ramified Type Theory*.

Teorema de Fubini para Rectángulos

[Fubini's Theorem for rectangles] Sea $f(x, y)$ una función continua sobre el rectángulo $A \times B$, donde $A = \{x \in \mathbb{R} : a \leq x \leq b\}$; $B = \{x \in \mathbb{R} : c \leq y \leq d\}$, entonces

$$\int\int_{A\times B} f(x,y)dR = \int_a^b (\int_c^d f(x,y)dy)dx = \int_c^d (\int_a^b f(x,y)dx)dy$$

Denotando:

Teorema de Fubini para Rectángulos

[Fubini's Theorem for rectangles] Sea $f(x, y)$ una función continua sobre el rectángulo $A \times B$, donde $A = \{x \in \mathbb{R} : a \leq x \leq b\}$; $B = \{x \in \mathbb{R} : c \leq y \leq d\}$, entonces

$$\int\int_{A\times B} f(x,y)dR = \int_a^b (\int_c^d f(x,y)dy)dx = \int_c^d (\int_a^b f(x,y)dx)dy$$

Denotando:

$$\begin{cases} Int_1^f(A \times B) = & \int\int_{A\times B} f(x,y)dA \\ Int_2^f(A,B) = & \int_A (\int_B f(x,y)dy)dx \\ Int_3^f(B,A) = & \int_B (\int_A f(x,y)dx)dy \end{cases}$$

Cambio de Variable en Fubini

$$Int_1^f(A \times B) \underset{g:(y,x)\to(x,y)}{\overset{g:(x,y)\to(y,x)}{\rightleftarrows}} Int_1^f(B \times A)$$

$$\Uparrow \text{Fubini} \qquad \Uparrow \text{Fubini}$$

$$\Downarrow \qquad \Downarrow$$

$$Int_2^f(A, B) = = = = = = = Int_3^f(B, A)$$

Detalles en Fubini

$$
\begin{aligned}
Int_1^f(A \times B) &= \int_{A\times B} f(x,y)dR \\
&= \int_{g(B\times A)} f(x,y)dR \\
&= \int_{B\times A} f(g(x,y))|J_g(g(x,y))|dR' \\
&= \int_{B\times A} f(y,x)dR' \\
&= Int_1^f(B \times A)
\end{aligned}
$$

Detalles en Fubini

$$
\begin{aligned}
Int_1^f(A \times B) &= \int_{A\times B} f(x,y)dR \\
&= \int_{g(B\times A)} f(x,y)dR \\
&= \int_{B\times A} f(g(x,y))|J_g(g(x,y))|dR' \quad \text{Donde} \\
&= \int_{B\times A} f(y,x)dR' \\
&= Int_1^f(B \times A)
\end{aligned}
$$

$$
|J_g(g(x,y))| = |J_g(g_1(x,y), g_2(x,y))| = \left| \begin{vmatrix} \frac{\partial g_1(x,y)}{\partial x} & \frac{\partial g_1(x,y)}{\partial y} \\ \frac{\partial g_2(x,y)}{\partial x} & \frac{\partial g_2(x,y)}{\partial y} \end{vmatrix} \right|
$$

Detalles en Fubini

$$
\begin{aligned}
Int_1^f(A \times B) &= \int_{A\times B} f(x,y)dR \\
&= \int_{g(B\times A)} f(x,y)dR \\
&= \int_{B\times A} f(g(x,y))|J_g(g(x,y))|dR' \quad \text{Donde} \\
&= \int_{B\times A} f(y,x)dR' \\
&= Int_1^f(B \times A)
\end{aligned}
$$

$$
|J_g(g(x,y))| = |J_g(g_1(x,y), g_2(x,y))| = \left| \begin{vmatrix} \frac{\partial g_1(x,y)}{\partial x} & \frac{\partial g_1(x,y)}{\partial y} \\ \frac{\partial g_2(x,y)}{\partial x} & \frac{\partial g_2(x,y)}{\partial y} \end{vmatrix} \right|
$$

$$
\text{Especifícamente, } |J_g(y,x)| = \left| \begin{vmatrix} \frac{\partial y}{\partial x} & \frac{\partial y}{\partial y} \\ \frac{\partial x}{\partial x} & \frac{\partial x}{\partial y} \end{vmatrix} \right| = \left| \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix} \right| = |-1| = 1
$$

Cálculo λ

Tipos: τ

Cálculo λ

Tipos: τ

- Sea ι un *tipo básico*

Cálculo λ

Tipos: τ

- Sea ι un *tipo básico*

$$\tau ::= \iota \mid (\tau \to \tau) \mid \tau \times \tau$$

Cálculo λ

Tipos: τ

- Sea ι un *tipo básico*

$$\tau ::= \iota \mid (\tau \to \tau) \mid \tau \times \tau$$

Términos: t

Cálculo λ

Tipos: τ

- Sea ι un *tipo básico*

$$\tau ::= \iota \mid (\tau \to \tau) \mid \tau \times \tau$$

Términos: t

- Sea x una *variable*

Cálculo λ

Tipos: τ

- Sea ι un *tipo básico*

$$\tau ::= \iota \mid (\tau \to \tau) \mid \tau \times \tau$$

Términos: t

- Sea x una *variable*

$$t ::= x \mid (\lambda x.t) \mid (tt) \mid \langle t, t \rangle$$

Cálculo λ

Tipos: τ

- Sea ι un *tipo básico*

$$\tau ::= \iota \mid (\tau \to \tau) \mid \tau \times \tau$$

Términos: t

- Sea x una *variable*

$$t ::= x \mid (\lambda x.t) \mid (tt) \mid \langle t, t \rangle$$

Computaciones: $\triangleright$

Cálculo λ

Tipos: τ

- Sea ι un *tipo básico*

$$\tau ::= \iota \mid (\tau \to \tau) \mid \tau \times \tau$$

Términos: t

- Sea x una *variable*

$$t ::= x \mid (\lambda x.t) \mid (tt) \mid \langle t, t \rangle$$

Computaciones: $\triangleright$

- $(\lambda x.t)u \triangleright_\beta t[x := u]$ *(beta-reducción)*.

Cálculo λ

Tipos: τ

- Sea ι un *tipo básico*

$$\tau ::= \iota \mid (\tau \to \tau) \mid \tau \times \tau$$

Términos: t

- Sea x una *variable*

$$t ::= x \mid (\lambda x.t) \mid (tt) \mid \langle t, t \rangle$$

Computaciones: $\triangleright$

- $(\lambda x.t)u \triangleright_\beta t[x := u]$ *(beta-reducción)*.
- $(\lambda x.(tx)) \triangleright_\eta t$, if x no es libre en t *(eta-reducción)*.

Computaciones.

Considere los siguientes términos:

$$f_1 = \lambda\langle x, y\rangle,2 * x + 3 * y;; \quad | \quad f_2 = \lambda\langle y, x\rangle,2 * x + 3 * y;;$$

Computaciones.

Considere los siguientes términos:

$$f_1 = \lambda\langle x, y\rangle, 2 * x + 3 * y;; \quad f_1 : int \times int \longrightarrow int$$

$$f_2 = \lambda\langle y, x\rangle, 2 * x + 3 * y;; \quad f_2 : int \times int \longrightarrow int$$

Computaciones.

Considere los siguientes términos:

$f_1 = \lambda\langle x, y\rangle, 2 * x + 3 * y;;$ $f_1 : int \times int \longrightarrow int$	$f_2 = \lambda\langle y, x\rangle, 2 * x + 3 * y;;$ $f_2 : int \times int \longrightarrow int$
$f_3 = \lambda x \lambda y, 2 * x + 3 * y;;$	$f_4 = \lambda y.\lambda x, 2 * x + 3 * y;;$

Computaciones.

Considere los siguientes términos:

$$f_1 = \lambda\langle x, y\rangle, 2 * x + 3 * y;; \quad f_1 : int \times int \longrightarrow int$$

$$f_2 = \lambda\langle y, x\rangle, 2 * x + 3 * y;; \quad f_2 : int \times int \longrightarrow int$$

$$f_3 = \lambda x \lambda y, 2 * x + 3 * y;; \quad f_3 : int \longrightarrow (int \longrightarrow int)$$

$$f_4 = \lambda y.\lambda x, 2 * x + 3 * y;; \quad f_4 : int \longrightarrow (int \longrightarrow int)$$

Computaciones.

Considere los siguientes términos:

$$f_1 = \lambda\langle x, y\rangle, 2 * x + 3 * y;; \quad f_1 : int \times int \longrightarrow int$$

$$f_2 = \lambda\langle y, x\rangle, 2 * x + 3 * y;; \quad f_2 : int \times int \longrightarrow int$$

$$f_3 = \lambda x \lambda y, 2 * x + 3 * y;; \quad f_3 : int \longrightarrow (int \longrightarrow int)$$

$$f_4 = \lambda y.\lambda x, 2 * x + 3 * y;; \quad f_4 : int \longrightarrow (int \longrightarrow int)$$

Observe:

Computaciones.

Considere los siguientes términos:

$$f_1 = \lambda\langle x, y\rangle, 2 * x + 3 * y;; \quad f_1 : int \times int \longrightarrow int$$

$$f_2 = \lambda\langle y, x\rangle, 2 * x + 3 * y;; \quad f_2 : int \times int \longrightarrow int$$

$$f_3 = \lambda x \lambda y, 2 * x + 3 * y;; \quad f_3 : int \longrightarrow (int \longrightarrow int)$$

$$f_4 = \lambda y.\lambda x, 2 * x + 3 * y;; \quad f_4 : int \longrightarrow (int \longrightarrow int)$$

Observe:

- $f_1(4, 5) = (\lambda\langle x, y\rangle, 2 * x + 3 * y)(4, 5) \rhd_\beta 2 * 4 + 3 * 5 = \mathbf{23}$

Computaciones.

Considere los siguientes términos:

$$f_1 = \lambda\langle x, y\rangle, 2 * x + 3 * y;; \quad f_1 : int \times int \longrightarrow int$$

$$f_2 = \lambda\langle y, x\rangle, 2 * x + 3 * y;; \quad f_2 : int \times int \longrightarrow int$$

$$f_3 = \lambda x \lambda y, 2 * x + 3 * y;; \quad f_3 : int \longrightarrow (int \longrightarrow int)$$

$$f_4 = \lambda y.\lambda x, 2 * x + 3 * y;; \quad f_4 : int \longrightarrow (int \longrightarrow int)$$

Observe:

- $f_1(4,5) = (\lambda\langle x, y\rangle, 2 * x + 3 * y)(4,5) \rhd_\beta 2 * 4 + 3 * 5 = \mathbf{23}$
- $f_2(5,4) = (\lambda\langle y, x\rangle, 2 * x + 3 * y)(5,4) \rhd_\beta 2 * 4 + 3 * 5 = \mathbf{23}$

Computaciones.

Considere los siguientes términos:

$$f_1 = \lambda\langle x, y\rangle, 2 * x + 3 * y;; \quad f_1 : int \times int \longrightarrow int$$

$$f_2 = \lambda\langle y, x\rangle, 2 * x + 3 * y;; \quad f_2 : int \times int \longrightarrow int$$

$$f_3 = \lambda x \lambda y, 2 * x + 3 * y;; \quad f_3 : int \longrightarrow (int \longrightarrow int)$$

$$f_4 = \lambda y.\lambda x, 2 * x + 3 * y;; \quad f_4 : int \longrightarrow (int \longrightarrow int)$$

Observe:

- $f_1(4,5) = (\lambda\langle x, y\rangle, 2 * x + 3 * y)(4,5) \rhd_\beta 2 * 4 + 3 * 5 = \mathbf{23}$
- $f_2(5,4) = (\lambda\langle y, x\rangle, 2 * x + 3 * y)(5,4) \rhd_\beta 2 * 4 + 3 * 5 = \mathbf{23}$
- $(f_3 4)5 = ((\lambda x \lambda y, 2 * x + 3 * y)4)5 \rhd_\beta (\lambda y, 8 + 3 * y)5 \rhd_\beta 8 + 3 * 5 = \mathbf{23}$

Computaciones.

Considere los siguientes términos:

$$f_1 = \lambda\langle x, y\rangle,2 * x + 3 * y;; \quad f_1 : int \times int \longrightarrow int$$

$$f_2 = \lambda\langle y, x\rangle,2 * x + 3 * y;; \quad f_2 : int \times int \longrightarrow int$$

$$f_3 = \lambda x\lambda y,2 * x + 3 * y;; \quad f_3 : int \longrightarrow (int \longrightarrow int)$$

$$f_4 = \lambda y.\lambda x,2 * x + 3 * y;; \quad f_4 : int \longrightarrow (int \longrightarrow int)$$

Observe:

- $f_1(4,5) = (\lambda\langle x, y\rangle,2 * x + 3 * y)(4,5) \triangleright_\beta 2 * 4 + 3 * 5 = \mathbf{23}$
- $f_2(5,4) = (\lambda\langle y, x\rangle,2 * x + 3 * y)(5,4) \triangleright_\beta 2 * 4 + 3 * 5 = \mathbf{23}$
- $(f_3 4)5 = ((\lambda x\lambda y,2 * x + 3 * y)4)5 \triangleright_\beta (\lambda y,8 + 3 * y)5 \triangleright_\beta 8 + 3 * 5 = \mathbf{23}$
- $(f_4 5)4 = ((\lambda y\lambda x,2 * x + 3 * y)5)4 \triangleright_\beta (\lambda x,2 * x + 15)4 \triangleright_\beta 2 * 4 + 15 = \mathbf{23}$

Otros Cálculos

Consideeremos los siguientes términos (programas):

$perm = \lambda z \lambda \langle x, y \rangle . z \langle y, x \rangle$;;
$perm : ((A \times B) \to C)) \to ((B \times A) \to C))$

$curry = \lambda x \lambda x \lambda y . z \langle x, y \rangle$;;
$curry : ((A \times B) \to C)) \to (A \to (B \to C))$

Otros Cálculos

Consideeremos los siguientes términos (programas):

$perm = \lambda z \lambda \langle x, y \rangle . z \langle y, x \rangle$;;
$perm : ((A \times B) \to C)) \to ((B \times A) \to C))$

$curry = \lambda x \lambda x \lambda y . z \langle x, y \rangle$;;
$curry : ((A \times B) \to C)) \to (A \to (B \to C))$

Determinemos $permf_1 =?$ y $curryf_1 =?$ *(ejercicio propuesto)*.

Otros Cálculos

Consideeremos los siguientes términos (programas):

$perm = \lambda z \lambda \langle x, y \rangle . z \langle y, x \rangle$;;
$perm : ((A \times B) \to C)) \to ((B \times A) \to C))$

$curry = \lambda x \lambda x \lambda y . z \langle x, y \rangle$;;
$curry : ((A \times B) \to C)) \to (A \to (B \to C))$

Determinemos $permf_1 =?$ y $curryf_1 =?$ *(ejercicio propuesto)*.

Otros Cálculos

Consideeremos los siguientes términos (programas):

$perm = \lambda z \lambda \langle x, y \rangle . z \langle y, x \rangle$;;
$perm : ((A \times B) \to C)) \to ((B \times A) \to C))$

$curry = \lambda x \lambda x \lambda y . z \langle x, y \rangle$;;
$curry : ((A \times B) \to C)) \to (A \to (B \to C))$

Determinemos $permf_1 =?$ y $curryf_1 =?$ *(ejercicio propuesto)*.

$permf_1 \quad = \quad (\lambda z \lambda \langle x, y \rangle . z \langle y, x \rangle) \lambda \langle x, y \rangle, 2 * x + 3 * y$

Otros Cálculos

Consideeremos los siguientes términos (programas):

$perm = \lambda z \lambda \langle x, y \rangle . z \langle y, x \rangle$;;
$perm : ((A \times B) \to C)) \to ((B \times A) \to C))$

$curry = \lambda x \lambda x \lambda y . z \langle x, y \rangle$;;
$curry : ((A \times B) \to C)) \to (A \to (B \to C))$

Determinemos $permf_1 =?$ y $curryf_1 =?$ *(ejercicio propuesto)*.

$$\begin{array}{lcl} permf_1 & = & (\lambda z \lambda \langle x, y \rangle . z \langle y, x \rangle) \lambda \langle x, y \rangle, 2 * x + 3 * y \\ & \triangleright_\beta & \lambda \langle x, y \rangle . (\lambda \langle x, y \rangle, 2 * x + 3 * y) \langle y, x \rangle \end{array}$$

Otros Cálculos

Consideeremos los siguientes términos (programas):

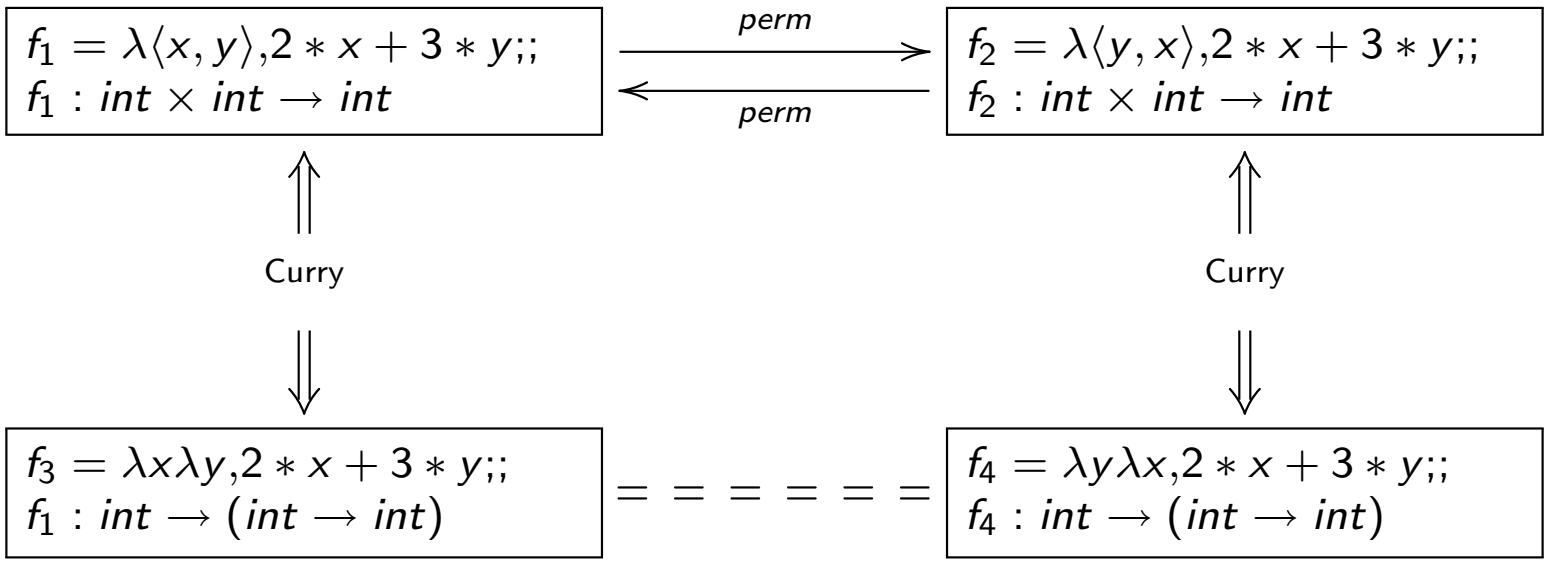
$perm = \lambda z \lambda \langle x, y \rangle . z \langle y, x \rangle$;;
$perm : ((A \times B) \to C)) \to ((B \times A) \to C))$

$curry = \lambda x \lambda x \lambda y . z \langle x, y \rangle$;;
$curry : ((A \times B) \to C)) \to (A \to (B \to C))$

Determinemos $permf_1 =?$ y $curryf_1 =?$ *(ejercicio propuesto)*.

$$\begin{aligned} permf_1 &= (\lambda z \lambda \langle x, y \rangle . z \langle y, x \rangle) \lambda \langle x, y \rangle, 2 * x + 3 * y \\ &\triangleright_\beta \lambda \langle x, y \rangle . (\lambda \langle x, y \rangle, 2 * x + 3 * y) \langle y, x \rangle \\ &\triangleright_\beta \lambda \langle x, y \rangle, 2 * y + 3 * x = f_2 \end{aligned}$$

perm y curry



¿ Qué nos llama la atención?

¿ Fubini?

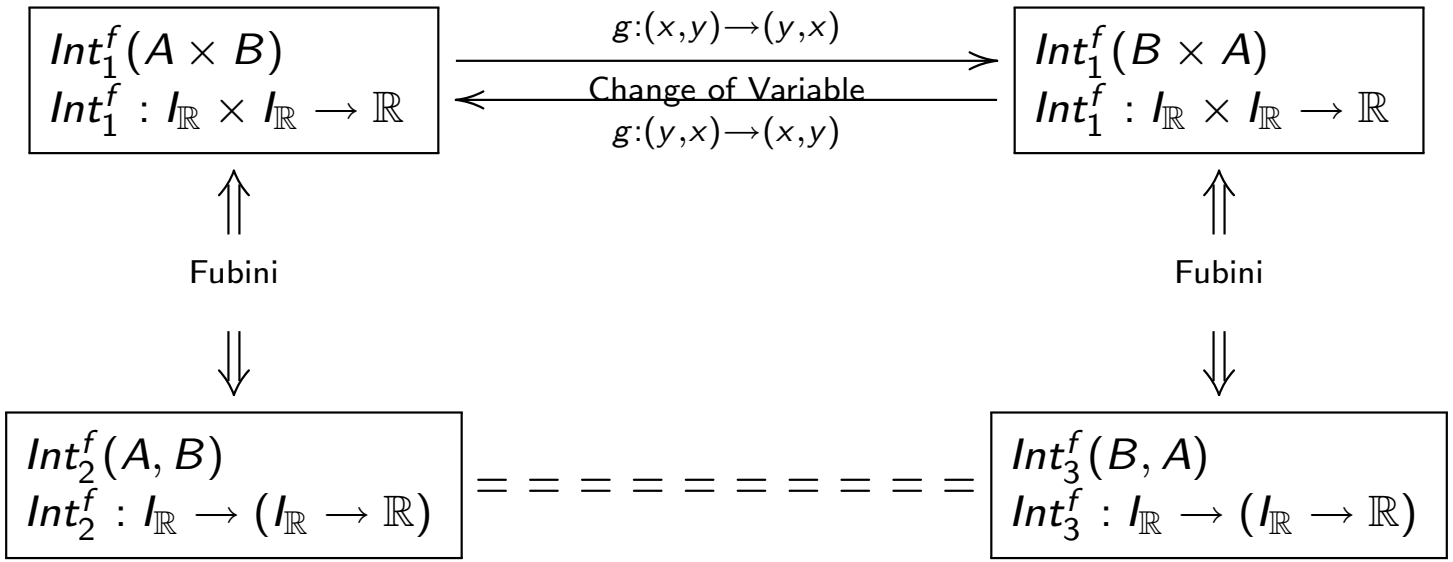
Algo muy parecido a lo que sucedio con el concepto de *Cambio de Variables*, mejor dicho la misma idea!

¿ Fubini?

Algo muy parecido a lo que sucedio con el concepto de *Cambio de Variables*, mejor dicho la misma idea!
Recordemos:

¿ Fubini?

Algo muy parecido a lo que sucedio con el concepto de *Cambio de Variables*, mejor dicho la misma idea!
Recordemos:



Invetibilidad de Programas

Definition (Programas Invertibles) *Un programa* $M : A \longrightarrow B$ *es invertible si y sólo si existe un programa* $N : B \longrightarrow A$ *tal que* $N \circ M = Id_A$ *and* $M \circ N = Id_B$.

Invetibilidad de Programas

Definition (Programas Invertibles) *Un programa* $M : A \longrightarrow B$ *es invertible si y sólo si existe un programa* $N : B \longrightarrow A$ *tal que* $N \circ M = Id_A$ *and* $M \circ N = Id_B$.

¿Qué sentido tienen $\circ$ y Id?

Invetibilidad de Programas

Definition (Programas Invertibles) *Un programa* $M : A \longrightarrow B$ *es invertible si y sólo si existe un programa* $N : B \longrightarrow A$ *tal que* $N \circ M = Id_A$ *and* $M \circ N = Id_B$.

¿Qué sentido tienen $\circ$ y Id?

Comente..

Algunos ejemplos:

#**Let** curry f $xy = \mathrm{f}(x, y)$;;
curry: $((A \times B) \to C) \to (A \to (B \to C)) = \langle \textbf{fun} \rangle$
#**Let** uncurry f $(x, y) = \mathrm{f}xy$;;
uncurry: $(A \to (B \to C)) \to ((A \times B) \to C) = \langle \textbf{fun} \rangle$

Algunos ejemplos:

#**Let** curry f xy = f(x, y);;
curry: $((A \times B) \to C) \to (A \to (B \to C)) = \langle$**fun**$\rangle$
#**Let** uncurry f (x, y) = fxy;;
uncurry: $(A \to (B \to C)) \to ((A \times B) \to C) = \langle$**fun**$\rangle$

#**Let** perm f xy = fyx;;
perm : $(A \to (B \to C)) \to (B \to (A \to C)) = \langle$**fun**$\rangle$

Algunos ejemplos:

#**Let** curry f xy = f(x, y);;
curry: $((A \times B) \to C) \to (A \to (B \to C)) = \langle$**fun**$\rangle$
#**Let** uncurry f (x, y) = fxy;;
uncurry: $(A \to (B \to C)) \to ((A \times B) \to C) = \langle$**fun**$\rangle$

#**Let** perm f xy = fyx;;
perm : $(A \to (B \to C)) \to (B \to (A \to C)) = \langle$**fun**$\rangle$

#**Let** perm' f (x, y) = f(y, x);;
perm' : $((A \times B) \to C)) \to ((B \times A) \to C))= \langle$**fun**$\rangle$

Usemos estos programas invertibles:

Consideremos los siguientes cálculos.

perm' f_1 = #**Let** perm' $f(x,y) = f(y,x)$;;
perm': $((int \times int) \longrightarrow int) \longrightarrow (int \times int) \longrightarrow int)) = \langle \textbf{fun} \rangle$ $[f_1]$

Usemos estos programas invertibles:

Consideremos los siguientes cálculos.

$$\begin{aligned} \text{perm' } f_1 &= \#\textbf{Let perm' } f(x,y) = f(y,x);; \\ &\quad \text{perm': } ((int \times int) \to int) \to (int \times int) \to int)) = \langle \textbf{fun} \rangle \\ &= \#\textbf{Let perm' } f_1(x,y) = f_1(y,x);; \\ &\quad \text{perm' } f_1\text{: } int \times int \to int = \langle \textbf{fun} \rangle \end{aligned} \qquad [f_1]$$

Usemos estos programas invertibles:

Consideremos los siguientes cálculos.

perm' f_1 $=$ #**Let** perm' $f(x,y) = f(y,x)$;;
perm': $((int \times int) \to int) \to (int \times int) \to int)) = \langle \textbf{fun} \rangle$ $[f_1]$

$=$ #**Let** perm' $f_1(x,y) = f_1(y,x)$;;
perm' f_1: $int \times int \to int = \langle \textbf{fun} \rangle$

$=$ #**Let** perm' $f_1(x,y) = 2 * y + 3 * x$;;
perm' f_1: $int \times int \to int = \langle \textbf{fun} \rangle$

Rename $g :=$ *perm' f*$_1$,
$x := y$ *and* $y := x$

Usemos estos programas invertibles:

Consideremos los siguientes cálculos.

$$\begin{aligned}
\text{perm' } f_1 &= \#\textbf{Let perm' } f(x,y) = f(y,x);; \\
&\quad \text{perm': } ((int \times int) \to int) \to (int \times int) \to int)) = \langle \textbf{fun} \rangle && [f_1] \\
&= \#\textbf{Let perm' } f_1(x,y) = f_1(y,x);; \\
&\quad \text{perm' } f_1\text{: } int \times int \to int = \langle \textbf{fun} \rangle \\
&= \#\textbf{Let perm' } f_1(x,y) = 2 * y + 3 * x;; && \textit{Rename } g := \textit{perm' } f_1, \\
&\quad \text{perm' } f_1\text{: } int \times int \to int = \langle \textbf{fun} \rangle && x := y \textit{ and } y := x \\
&= \boxed{\begin{array}{l} \#\textbf{Let } g(y,x) = 2 * x + 3 * y;; \\ g\text{: } int \times int \to int = \langle \textbf{fun} \rangle \end{array}} = f_2
\end{aligned}$$

Otro ejemplo

$$\text{curry } f_1 \quad = \quad \begin{array}{l} \#\textbf{Let curry } fxy = f(x,y);; \\ \text{curry: } ((int \times int) \longrightarrow int) \longrightarrow (int \longrightarrow (int \longrightarrow int)) = \langle \textbf{fun} \rangle \end{array} \qquad [f_1]$$

Otro ejemplo

$$\begin{aligned} \text{curry } f_1 &= \#\textbf{Let curry } fxy = f(x,y);; \\ &\quad \text{curry: } ((int \times int) \to int) \to (int \to (int \to int)) = \langle \textbf{fun} \rangle \\ &= \#\textbf{Let curry } f_1 xy = f_1(x,y);; \\ &\quad \text{curry } f_1\text{: } int \to (int \to int) = \langle \textbf{fun} \rangle \end{aligned} \qquad [f_1]$$

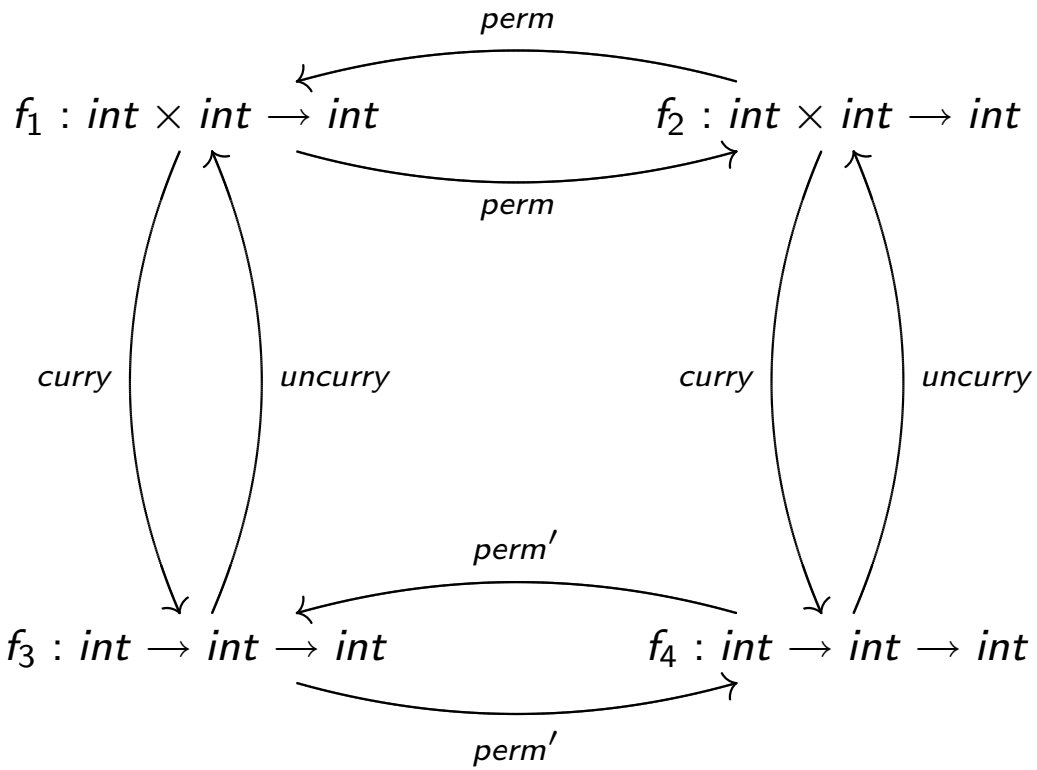
Otro ejemplo

$$
\begin{aligned}
\text{curry } f_1 \quad &= \quad \#\textbf{Let curry } fxy = f(x,y);; \\
&\quad\ \text{curry: } ((int \times int) \to int) \to (int \to (int \to int)) = \langle \textbf{fun} \rangle \qquad [f_1] \\
&= \quad \#\textbf{Let curry } f_1 xy = f_1(x,y);; \\
&\quad\ \text{curry } f_1\text{: } int \to (int \to int) = \langle \textbf{fun} \rangle \\
&= \quad \#\textbf{Let curry } f_1 xy = 2 * x + 3 * y;; \\
&\quad\ \text{curry } f_1\text{: } int \to (int \to int) = \langle \textbf{fun} \rangle \qquad \textit{Rename } g := \textit{curry } f_1
\end{aligned}
$$

Otro ejemplo

$$
\begin{aligned}
\text{curry } f_1 \quad &= \quad \#\textbf{Let curry } fxy = f(x,y);; \\
&\quad\ \text{curry: } ((int \times int) \to int) \to (int \to (int \to int)) = \langle \textbf{fun} \rangle \qquad [f_1] \\
&= \quad \#\textbf{Let curry } f_1 xy = f_1(x,y);; \\
&\quad\ \text{curry } f_1\text{: } int \to (int \to int) = \langle \textbf{fun} \rangle \\
&= \quad \#\textbf{Let curry } f_1 xy = 2 * x + 3 * y;; \qquad \textit{Rename } g := \textit{curry } f_1 \\
&\quad\ \text{curry } f_1\text{: } int \to (int \to int) = \langle \textbf{fun} \rangle \\
&= \quad \boxed{\begin{array}{l} \#\textbf{Let } gxy = 2 * x + 3 * y;; \\ g\text{: } int \to (int \to int) = \langle \textbf{fun} \rangle \end{array}} = f_3
\end{aligned}
$$

En resumen:


perm
$f_1 : int \times int \longrightarrow int$
$f_2 : int \times int \longrightarrow int$
perm
curry
uncurry
curry
uncurry
perm′
$f_3 : int \longrightarrow int \longrightarrow int$
$f_4 : int \longrightarrow int \longrightarrow int$
perm′

Definiciones Esenciales: Invertibilidad e Isomorfismos

Definición (Programas Invertibles) *Un programa* $M : A \longrightarrow B$ *es invertible si y sólo si existe un programa* $N : B \longrightarrow A$ *tal que* $N \circ M = Id_A$ *and* $M \circ N = Id_B$.

Definiciones Esenciales: Invertibilidad e Isomorfismos

Definición (Programas Invertibles) *Un programa $M : A \to B$ es invertible si y sólo si existe un programa $N : B \to A$ tal que $N \circ M = Id_A$ and $M \circ N = Id_B$.*

Definición (Isomorfismos de Tipos) *Dos tipos A y B son isomorficos, lo que denotamos por $A \simeq_{Tp} B$, si y sólo si existe un programas $M : A \to B$, o bien $N : B \to A$ invertibles.*

Definiciones Esenciales: Invertibilidad e Isomorfismos

Definición (Programas Invertibles) *Un programa* $M : A \to B$ *es invertible si y sólo si existe un programa* $N : B \to A$ *tal que* $N \circ M = Id_A$ *and* $M \circ N = Id_B$.

Definición (Isomorfismos de Tipos) *Dos tipos* A *y* B *son isomorficos, lo que denotamos por* $A \simeq_{Tp} B$, *si y sólo si existe un programas* $M : A \to B$, *o bien* $N : B \to A$ *invertibles.*

Definición (Isomorfismos de Programas) *Dos programas* $P : A$ *y* $Q : B$ *son isomorficos, lo que denotamos por* $P \simeq_{Pr} Q$, *si y sólo si* $A \simeq_{Tp} B$ *y existe un programa invertible* $F : A \to B$ *tal que* $FP = Q$.

Decisiones

Definición (Problemas Decidibles) *Sea* **Q** *una clase de preguntas que tienen por respuesta* **Si** *, o bien* **No**. *Decimos que la clase* **Q** *es decidable si y sólo si existe* **Alg** *un algoritmo, tal que para toda pregunta* $q \in \mathbf{Q}$, $\mathbf{Alg}(q) \in \{\mathbf{Yes}, \mathbf{No}\}$.

Nuestros problemas de decisión:

Q1: (Validez) *Sean M y N dos programas con tipos isomorficos. Decidir efectivemente, si se cumple o no que $M \simeq_{Pr} N$.*

Nuestros problemas de decisión:

Q1: (Validez) *Sean M y N dos programas con tipos isomorficos. Decidir efectivemente, si se cumple o no que $M \simeq_{Pr} N$.*

Q2: (Unificación) *Sean M y N dos fragmentos de programas, [Esto es, programas que poseen variables libres] Decidir efectivamente, si se cumple o no que existe una sustitución σ sobre las variables libres tal que $\sigma(M) \simeq_{Pr} \sigma(N)$.*

Nuestros problemas de decisión:

Q1: (Validez) *Sean M y N dos programas con tipos isomorficos. Decidir efectivemente, si se cumple o no que $M \simeq_{Pr} N$.*

Q2: (Unificación) *Sean M y N dos fragmentos de programas, [Esto es, programas que poseen variables libres] Decidir efectivamente, si se cumple o no que existe una sustitución σ sobre las variables libres tal que $\sigma(M) \simeq_{Pr} \sigma(N)$.*

Q3: (Correspondencia) *Sea M un programa y N un fragmento de programa. Decidir efectivamente, si se cumple o no que existe una sustitución σ tal que $M \simeq_{Pr} \sigma(N)$.*

Isomorpfismos de Tipos: Marco Teórico

Bruce K., Di Cosmo R., Longo G.
Provable Isomorphism of Types, *Mathematical Structures in Computer Science*, **2**, 231 - 247, 1991.

Dezani-Ciancaglini M.
Characterization of Normal Forms Possessing Inverse in the $\lambda_{\beta\eta}$-Calculus, *Theoretical Computer Science*, **2**, 323 - 337, 1976.

Di Cosmo R.
Isomorphism of Types: from λ-calculus to information retrieval and language design, *Birkhäuser*, 1995.

Isomorfismos de Tipos: Algoritmos y Aplicaciones

Rittri M.
Retrieving Library Functions by Unifying Types Modulo Linear Type Isomorphism, *Theoretical Informatics and Applications*, **27**, 71 -89, 1993.

Soloviev S.
The Category of Finite Sets and Cartesian Closed Categories, *Journal of Soviet Mathematics*, **22(3)**, 1387 - 1400, 1983.

Zibin Y., Gil J., Considine J.
Efficient Algorithm for Isomorphism of Simple Types, *Proceedings POPL'03 in ACM SIGPLAN*, **38**, 160 - 171, 2003.

Transformación de Programas

Huet G. and Lang B.,
Proving and Applying Program Transformations Expressed with Second Order Patterns, *Acta Informatica*, 11:31–55, 1978.

Algoritmos de decisión eficientes

Estudiaremos algunos algoritmos que decidan, en primera instancia el problema de isomorphismos de tipos.

Algoritmos de decisión eficientes

Estudiaremos algunos algoritmos que decidan, en primera instancia el problema de isomorphismos de tipos.

Q2: (Unificación) *Sean M y N dos fragmentos de programas, [Esto es, programas que poseen variables libres] Decidir wefectivamente, si se cumple o no que existe una sustitución σ sobre las variables libres tal que $\sigma(M) \simeq_{Pr} \sigma(N)$.*

Algoritmos de decisión eficientes

Estudiaremos algunos algoritmos que decidan, en primera instancia el problema de isomorphismos de tipos.

Q2: (Unificación) *Sean M y N dos fragmentos de programas, [Esto es, programas que poseen variables libres] Decidir wefectivamente, si se cumple o no que existe una sustitución σ sobre las variables libres tal que $\sigma(M) \simeq_{Pr} \sigma(N)$.*

Paso siguiente explorar soluciones algorítmicas eficientes.