Fundamentos: Desde Lógica Matemática hasta Diseño de Lenguages de Programación Funcional

Carlos Martínez Méndez

Department of Mathematics and Computer Science,

Wesleyan University

cmartinez@wesleyan.edu

August, 2006

Overview:

- Foundations of Mathematics and Historical Remarks.
- Category Theory, λ-Calculus, Proof Theory and Programming Languages.
- Type Isomorphisms and Program Isomorphisms.

Russell's Paradox:

(1879-93) Frege's global formalization of Mathematics.

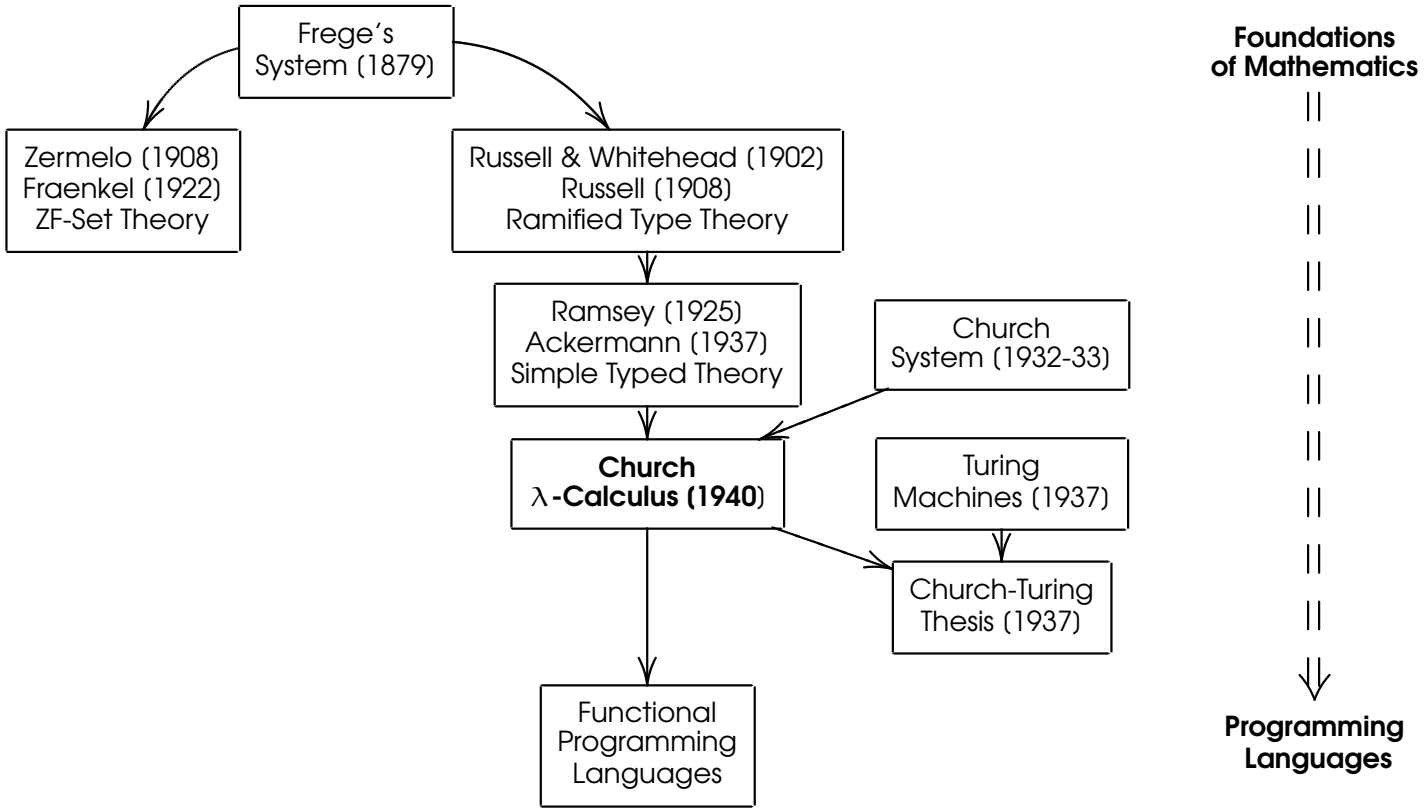
(Problem) Russell's Paradox challenges its consistency.

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

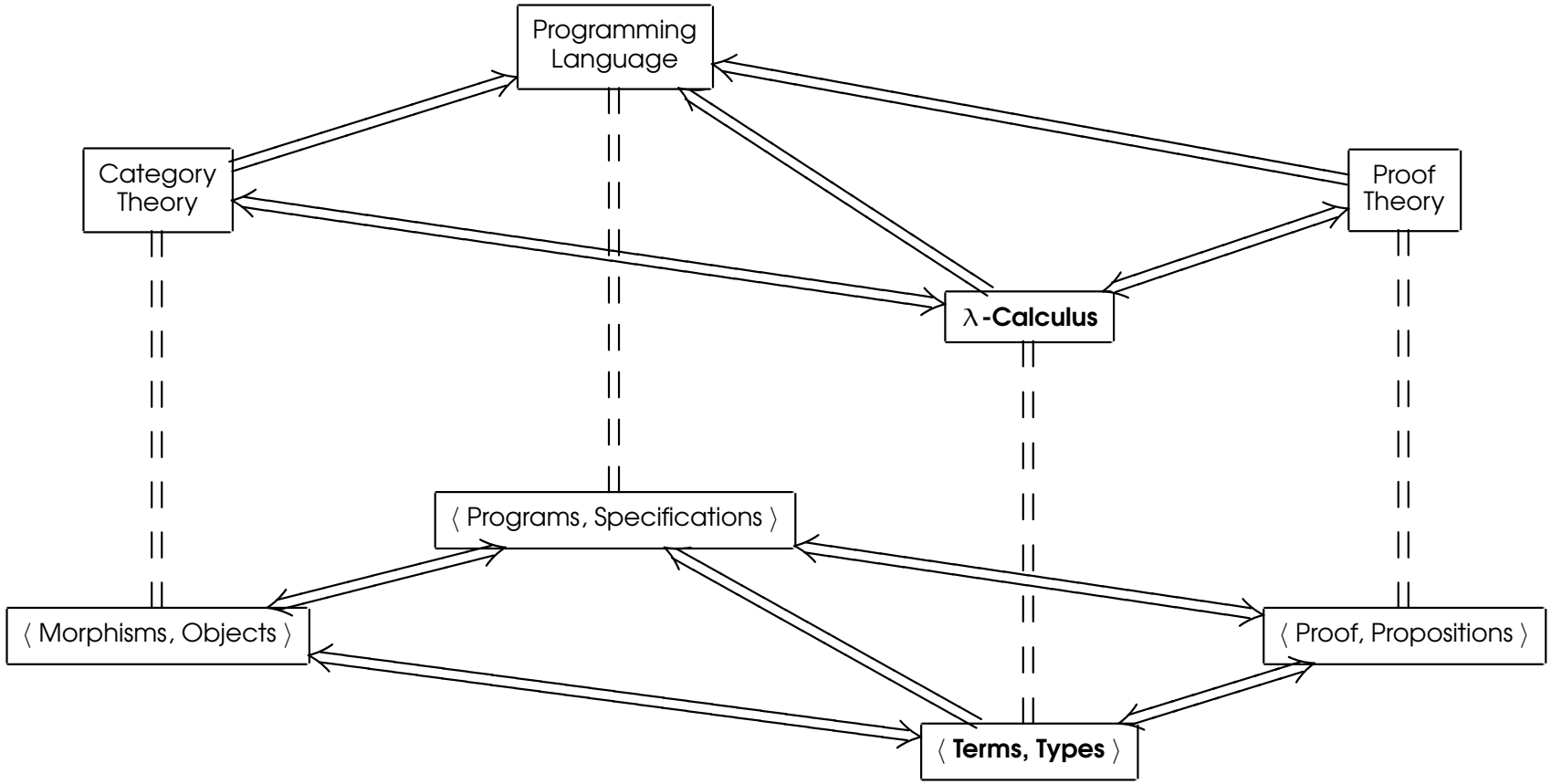
Q: Is $R \in R$?

(Solution) Russell classified sets by using *types*: a kind of label that constrains expressiveness. This became known as Russell's *Ramified Type Theory*.

From Foundations of Mathematics to Programming Languages:



Programming Language Design: Big Picture



Metaphor:

Theorem (Fubini's Theorem for rectangles) Let $f(x,y)$ be continuous on the rectangle $A \times B$, where $A = \{x \in \mathbb{R} : a \leq x \leq b\}$; $B = \{x \in \mathbb{R} : c \leq y \leq d\}$. Then

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

Denote:

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

Change of Variable and Fubini's Theorem:

$$\begin{array}{ccc}
\mathrm{Int}_1^f(A\times B) & \xrightarrow[\text{Change of Variable}]{g:(x,y)\to(y,x)} & \mathrm{Int}_1^f(B\times A) \\
\Uparrow & \xleftarrow{g:(y,x)\to(x,y)} & \Uparrow \\
\text{Fubini} & & \text{Fubini} \\
\Downarrow & & \Downarrow \\
\mathrm{Int}_2^f(A,B) & = = = = = = = & \mathrm{Int}_3^f(B,A)
\end{array}$$

Change of Variable:

$$
\begin{aligned}
\mathrm{Int}_1^f(A \times B) &= \int_{A\times B} f(x,y)\,dR \\
&= \int_{g(B\times A)} f(x,y)\,dR \\
&= \int_{B\times A} f(g(x,y))|J_g(g(x,y))|\,dR' \\
&= \int_{B\times A} f(y,x)\,dR' \\
&= \mathrm{Int}_1^f(B \times A)
\end{aligned}
$$

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

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

λ-Calculus

Types: τ

- Let ι be a *basic type*

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

Terms: t

- Let x be a *variable*

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

Computations: $\triangleright$

- $(\lambda x.t)u \triangleright_\beta t[x := u]$ *(beta-reduction)*.
- $(\lambda x.(tx)) \triangleright_\eta t$, if x is not free in t *(eta-reduction)*.

Computations:

Consider the following terms:

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

Observe:

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

More Computations:

Consider the following terms:

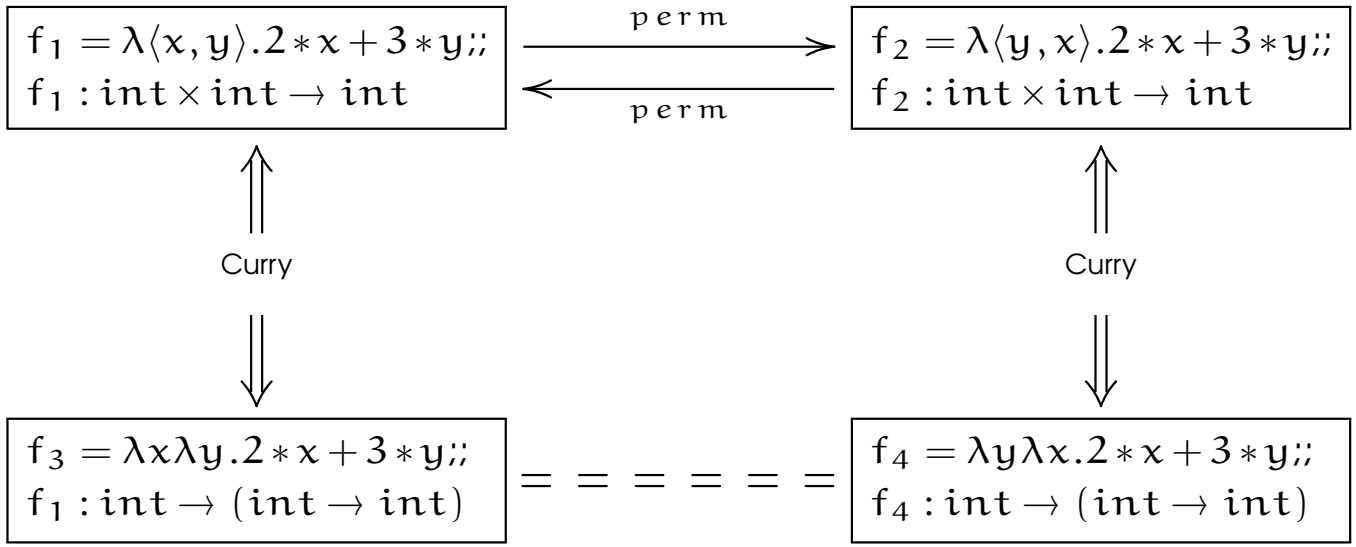
$\mathtt{perm} = \lambda z \lambda \langle x, y \rangle . z \langle y, x \rangle$;;
$\mathtt{perm} : ((A \times B) \to C)) \to ((B \times A) \to C))$

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

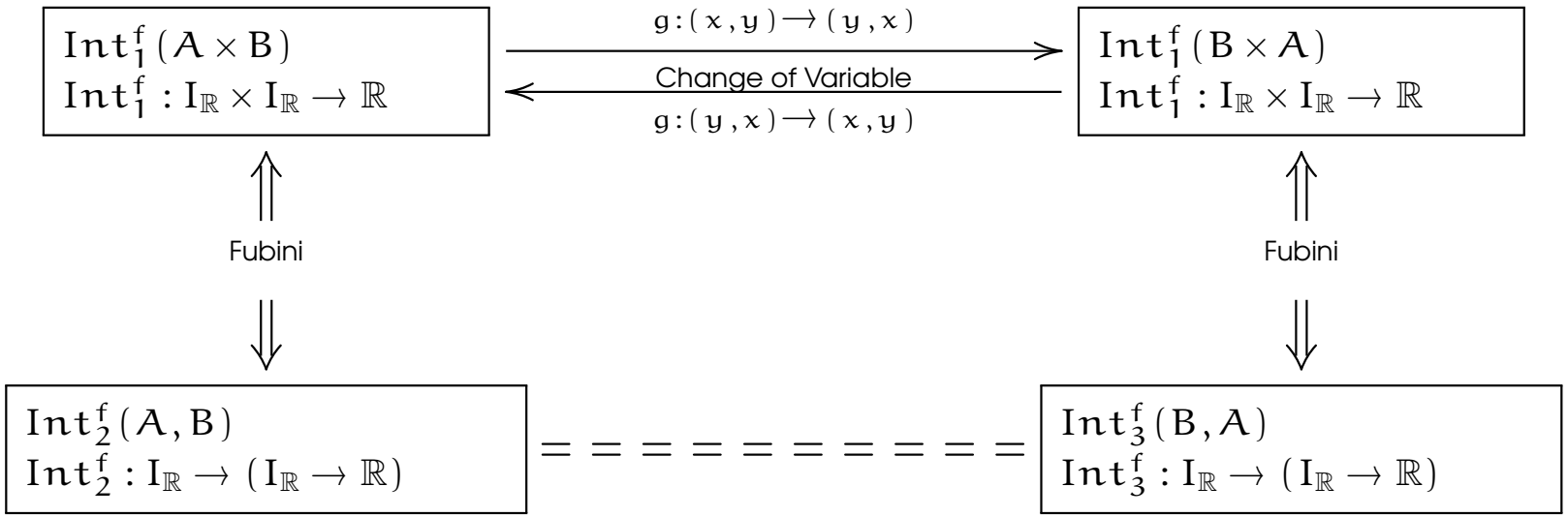
Let us compute $\mathtt{permf}_1 = ?$ and $\mathtt{curryf}_1 = ?$ *(exercise)*.

$$
\begin{array}{lcl}
\mathtt{permf}_1 & = & (\lambda z \lambda \langle x, y \rangle . z \langle y, x \rangle) \lambda \langle x, y \rangle . 2 * x + 3 * y \\
& \rhd_\beta & \lambda \langle x, y \rangle . (\lambda \langle x, y \rangle . 2 * x + 3 * y) \langle y, x \rangle \\
& \rhd_\beta & \lambda \langle x, y \rangle . 2 * y + 3 * x = f_2
\end{array}
$$

perm and curry:



Change of Variable and Fubini's Theorem: Revisited



Invertible Programs:

Definition *(Invertible Programs)* *A program* $\mathrm{M} : \mathrm{A} \to \mathrm{B}$ *is invertible if there exists a program* $\mathrm{N} : \mathrm{B} \to \mathrm{A}$ *such that* $\mathrm{N} \circ \mathrm{M} = \mathrm{Id}_{\mathrm{A}}$ *and* $\mathrm{M} \circ \mathrm{N} = \mathrm{Id}_{\mathrm{B}}$.

Examples:

```
#Let curry f x y = f(x,y);;
curry: ((A × B) → C) → (A → (B → C)) = ⟨fun⟩
#Let uncurry f (x,y) = f x y;;
uncurry: (A → (B → C)) → ((A × B) → C) = ⟨fun⟩
```

```
#Let perm f x y = f y x;;
perm : (A → (B → C)) → (B → (A → C)) = ⟨fun⟩
```

```
#Let perm' f (x,y) = f(y,x);;
perm' : ((A × B) → C)) → ((B × A) → C))= ⟨fun⟩
```

Mapping:

Consider the following computation:

perm′ f_1 = **#Let** perm′ $f(x,y) = f(y,x)$;;
perm′: $((int \times int) \to int) \to ((int \times int) \to int)$ = ⟨**fun**⟩ $[f_1]$

= **#Let** perm′ $f_1(x,y) = f_1(y,x)$;;
perm′ f_1: $int \times int \to int$ = ⟨**fun**⟩

= **#Let** perm′ $f_1(x,y) = 2*y + 3*x$;;
perm′ f_1: $int \times int \to int$ = ⟨**fun**⟩
Rename $g := perm'\ f_1$, $x := y$ *and* $y := x$

= ***#Let*** $g(y,x) = 2*x + 3*y$;;
g: $int \times int \to int$ = ⟨***fun***⟩ = f_2

Mapping:

Consider the following computation:

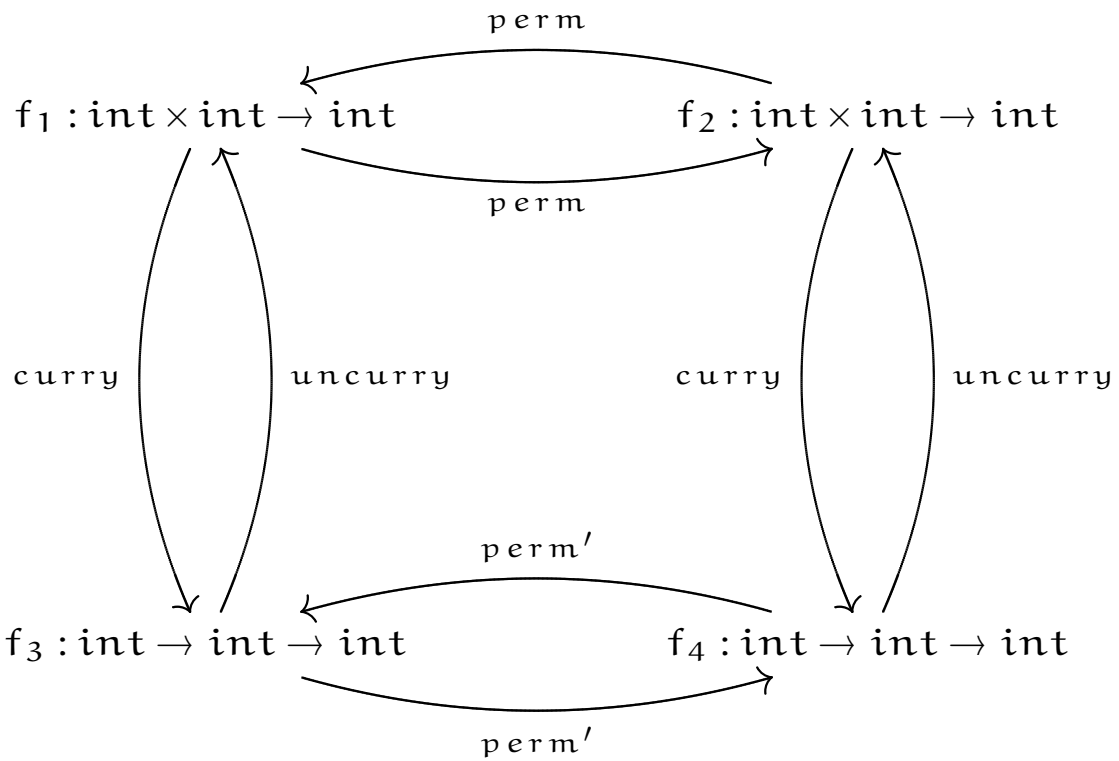
curry f_1 = **#Let** curry $fxy = f(x,y)$;;
curry: $((int \times int) \to int) \to (int \to (int \to int))$ = ⟨**fun**⟩ $[f_1]$

= **#Let** curry $f_1\,xy = f_1(x,y)$;;
curry f_1: $int \to (int \to int)$ = ⟨**fun**⟩

= **#Let** curry $f_1\,xy = 2*x+3*y$;;
curry f_1: $int \to (int \to int)$ = ⟨**fun**⟩ *Rename* $g :=$ *curry* f_1

= ***#Let*** $g\,xy = 2*x+3*y$;;
g: $int \to (int \to int)$ = ⟨***fun***⟩ = f_3

Synthesis:


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

Invertibility and Isomorphisms:

Definition *(Invertible Programs)* *A program* $\mathrm{M} : \mathrm{A} \to \mathrm{B}$ *is invertible if there exists a program* $\mathrm{N} : \mathrm{B} \to \mathrm{A}$ *such that* $\mathrm{N} \circ \mathrm{M} = \mathrm{Id}_{\mathrm{A}}$ *and* $\mathrm{M} \circ \mathrm{N} = \mathrm{Id}_{\mathrm{B}}$.

Definition *(Type Isomorphism)* *Two types* A *and* B *are type isomorphic, denoted* $\mathrm{A} \simeq_{\mathrm{Tp}} \mathrm{B}$*, if and only if there exist programs* $\mathrm{M} : \mathrm{A} \to \mathrm{B}$ *and* $\mathrm{N} : \mathrm{B} \to \mathrm{A}$ *such that* $\mathrm{N} \circ \mathrm{M} = \mathrm{Id}_{\mathrm{A}}$ *and* $\mathrm{M} \circ \mathrm{N} = \mathrm{Id}_{\mathrm{B}}$.

Definition *(Program Isomorphism)* *Two programs* $\mathrm{P} : \mathrm{A}$ *and* $\mathrm{Q} : \mathrm{B}$ *are program isomorphic, denoted* $\mathrm{P} \simeq_{\mathrm{Pr}} \mathrm{Q}$*, if and only if* $\mathrm{A} \simeq_{\mathrm{Tp}} \mathrm{B}$ *and there is an invertible program* $\mathrm{F} : \mathrm{A} \to \mathrm{B}$ *such that* $\mathrm{FP} = \mathrm{Q}$.

Decision Problems:

Definition ***(Decidable Problem)*** *Let* ***Q*** *be a class of questions whose expect* ***Yes*** *or* ***No*** *as their answers. We say that* ***Q*** *is decidable is and only is there exists* ***Alg*** *an algorithm, such that for all* $\mathrm{q} \in$ ***Q****,* ***Alg***$(\mathrm{q}) \in \{$***Yes***$,$ ***No***$\}$.

Decision Problems:

Question 1: ***(Validity)*** *Let* M *and* N *be two programs having isomorphic types. Decide whether or not* $\mathrm{M} \simeq_{\mathrm{Pr}} \mathrm{N}$.

Question 2: ***(Unification)*** *Let* M *and* N *be two program fragments, that is, programs having free variables. Decide whether or not there exists an instantiation* σ *of these variables such that* $\sigma(\mathrm{M}) \simeq_{\mathrm{Pr}} \sigma(\mathrm{N})$.

Question 3: ***(Matching)*** *Let* M *be a program and* N *be a program fragment. Decide whether or not there exists an instantiation* σ *such that* $\mathrm{M} \simeq_{\mathrm{Pr}} \sigma(\mathrm{N})$.

Decision Procedures:

$$\boxed{=_{\beta\eta} \quad \subseteq \quad \simeq_{\mathrm{Pr}}}$$

$=_{\beta\eta}$-validity	$\surd$		$\simeq_{\mathrm{Pr}}$-validity	$\surd$
$=_{\beta\eta}$-unification	X	$\Rightarrow$	$\simeq_{\mathrm{Pr}}$-unification	?
$=_{\beta\eta}$-matching	$\surd^{?}$		$\simeq_{\mathrm{Pr}}$-matching	$\surd^{?}$

References:

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.

References:

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.

References:

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

Russell's Ramified Type Theory:

Definition (Ramified Types) The ramified types $\mathcal{T}$

1. $\iota^{\mathbf{0}}$ is a ramified type (**0** is called the order of this type),
2. if $t_1, \ldots t_n$ are ramified types of order $a_1, \ldots, a_n$ respectively, and $a = MAX\{a_1, \ldots, a_n\}$ then $(t_1, \ldots t_n)^a$ is a ramified type of order a. (if $n = 0$ then take $a \geq 1$),
3. All ramified types can be constructed using the rules 1 and 2.