

Isomorphism of Types: From Foundations of Mathematics to Programming Language Design.

Carlos C. Martínez

Department of Mathematics and Computer Science,

Wesleyan University

cmartinez@wesleyan.edu

February, 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.

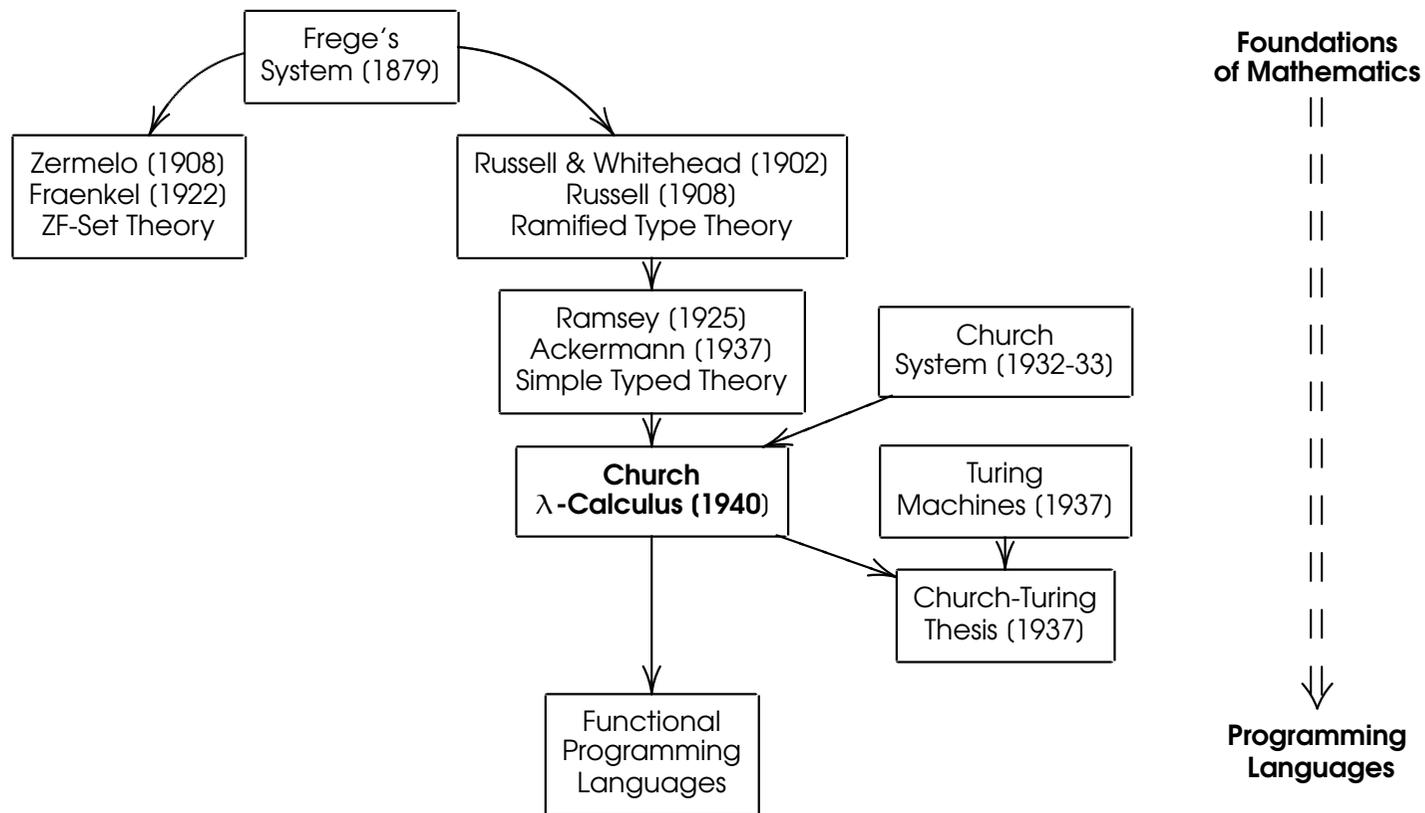
(Problem) Russell's Paradox challenges its consistency.

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

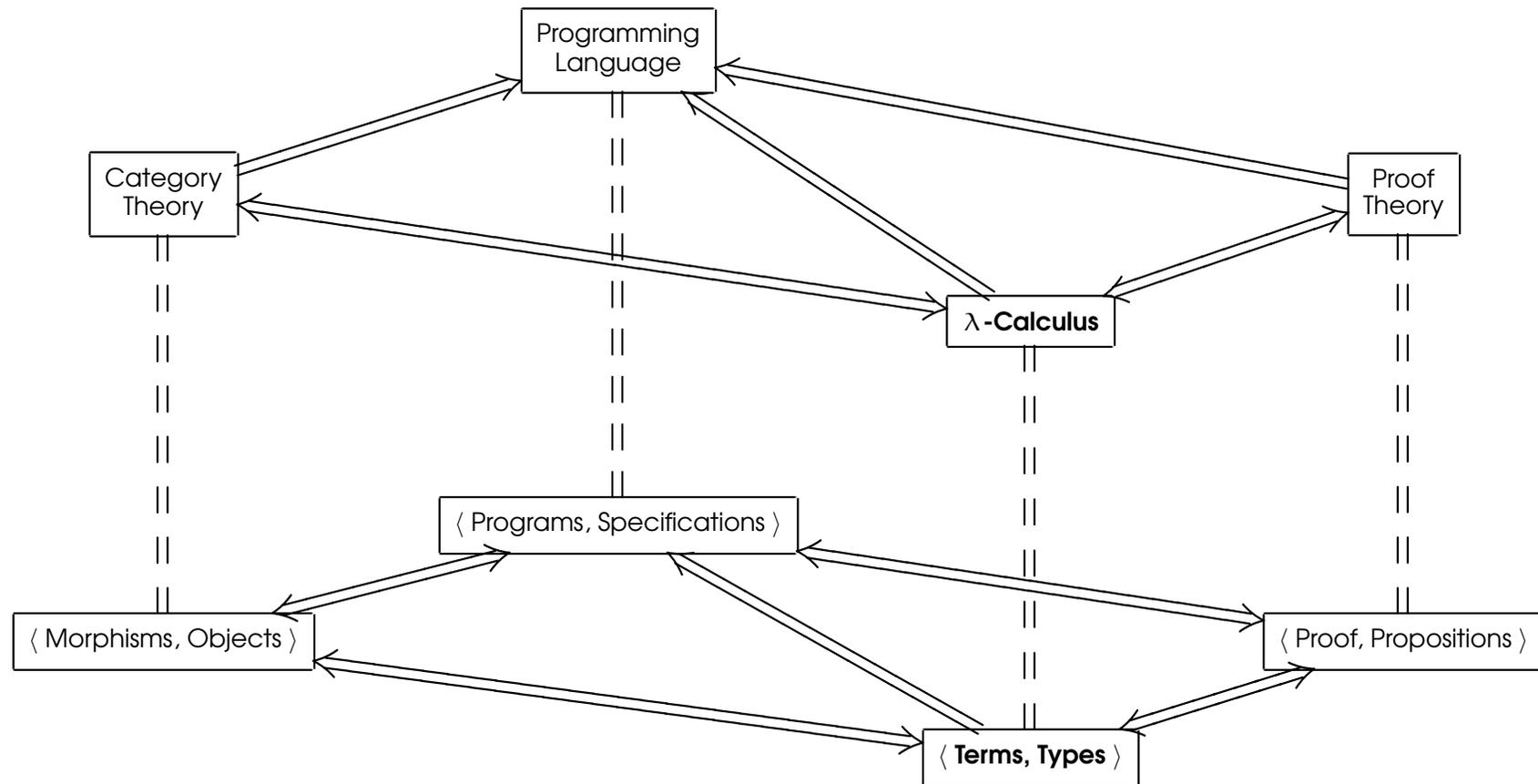
Q: Is $R \in R$?

(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 $A = \{x \in \mathbb{R} : a \leq x \leq b\}$; $B = \{y \in \mathbb{R} : c \leq y \leq d\}$. Then

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

Denote:

$$\begin{cases} \text{Int}_1^f(A \times B) = \iint_{A \times B} f(x, y) dA \\ \text{Int}_2^f(A, B) = \int_A \left(\int_B f(x, y) dy \right) dx \\ \text{Int}_3^f(B, A) = \int_B \left(\int_A f(x, y) dx \right) dy \end{cases}$$

Change of Variable and Fubini's Theorem:

$$\begin{array}{ccc}
 \text{Int}_1^f(A \times B) & \begin{array}{c} \xrightarrow{g:(x,y) \rightarrow (y,x)} \\ \text{Change of Variable} \\ \xleftarrow{g:(y,x) \rightarrow (x,y)} \end{array} & \text{Int}_1^f(B \times A) \\
 \uparrow \text{Fubini} & & \uparrow \text{Fubini} \\
 \downarrow & & \downarrow \\
 \text{Int}_2^f(A, B) = = = = = & & \text{Int}_3^f(B, A)
 \end{array}$$

Change of Variable:

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

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

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

λ -Calculus

Types: τ

- Let ι be a *basic type*

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

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:

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

Observe:

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

More Computations:

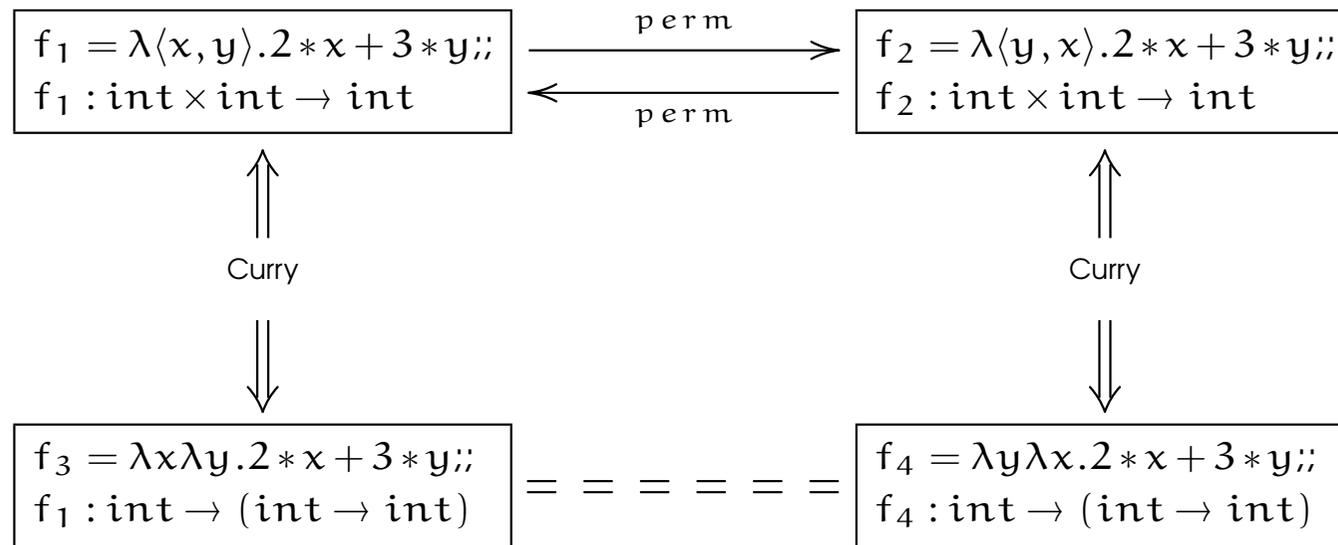
Consider the following terms:

$$\begin{array}{l|l} \text{perm} = \lambda z \lambda \langle x, y \rangle . z \langle y, x \rangle ;; & \text{curry} = \lambda x \lambda x \lambda y . z \langle x, y \rangle ;; \\ \text{perm} : ((A \times B) \rightarrow C) \rightarrow ((B \times A) \rightarrow C) & \text{curry} : ((A \times B) \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C)) \end{array}$$

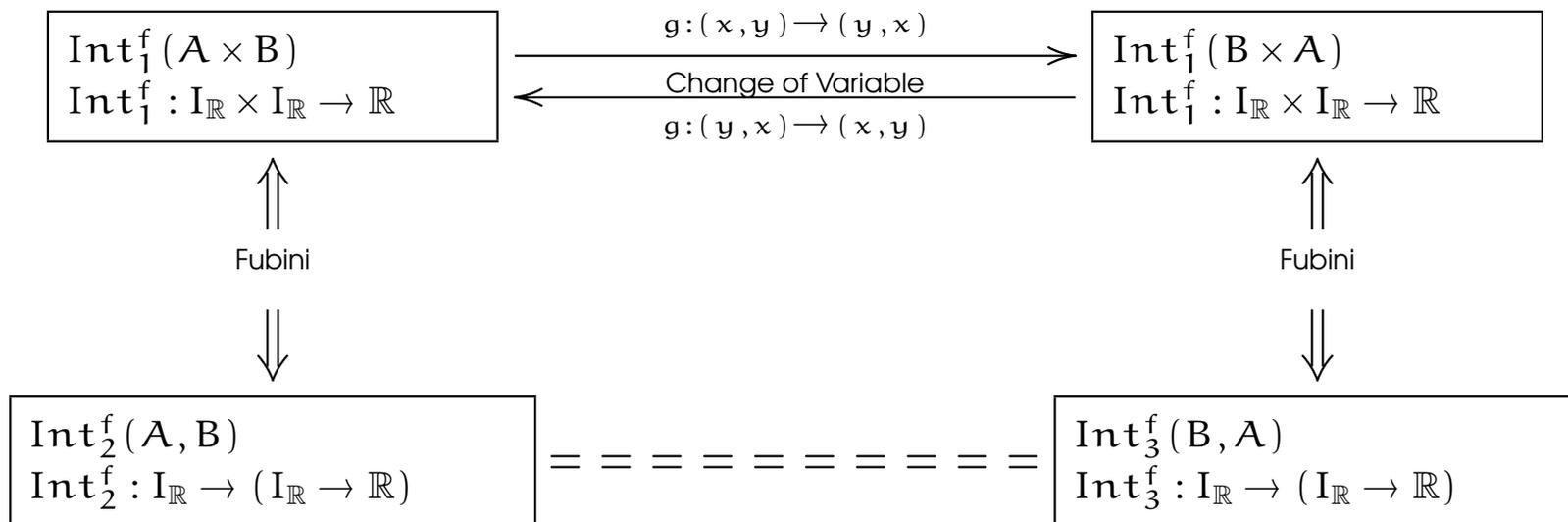
Let us compute $\text{perm} f_1 = ?$ and $\text{curry} f_1 = ?$ (*exercise*).

$$\begin{aligned} \text{perm} f_1 &= (\lambda z \lambda \langle x, y \rangle . z \langle y, x \rangle) \lambda \langle x, y \rangle . 2 * x + 3 * y \\ &\triangleright_{\beta} \lambda \langle x, y \rangle . (\lambda \langle x, y \rangle . 2 * x + 3 * y) \langle y, x \rangle \\ &\triangleright_{\beta} \lambda \langle x, y \rangle . 2 * y + 3 * x = f_2 \end{aligned}$$

perm and curry:



Change of Variable and Fubini's Theorem: Revisited



Invertibility and Isomorphisms:

Definition (Invertible Programs) A program $M : A \rightarrow B$ is invertible if there exists a program $N : B \rightarrow A$ such that $N \circ M = \text{Id}_A$ and $M \circ N = \text{Id}_B$.

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

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

Decision Problems:

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

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 $\sigma(M) \simeq_{Pr} \sigma(N)$.*

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 $M \simeq_{Pr} \sigma(N)$.*

Decision Procedures:

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

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

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. ι^0 is a ramified type (0 is called the order of this type),
2. if t_1, \dots, t_n are ramified types of order a_1, \dots, a_n respectively, and $a = \text{MAX}\{a_1, \dots, a_n\}$ then $(t_1, \dots, t_n)^a$ 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.

Decision Problems:

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

Invertible Programs:

Definition (Invertible Programs) A programs $M : A \rightarrow B$ is invertible of each other if there exists a program $N : B \rightarrow A$ such that $N \circ M = \text{Id}_A$ and $M \circ N = \text{Id}_B$.

Examples:

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

```
#Let perm f xy = fyx;;
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:

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

Synthesis:

