Open access • Book Chapter • DOI:10.1007/3-540-13346-1_7

Polymorphism is not set-theoretic — Source link

John C. Reynolds, John C. Reynolds

Institutions: Syracuse University, French Institute for Research in Computer Science and Automation

Published on: 27 Jun 1984 - International Symposium on Semantics of Data Types

Topics: Typed lambda calculus, System F and Cartesian closed category

INRIA

CENTRE
SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(3) 954 90 20

Rapports de Recherche

N° 296

POLYMORPHISM IS NOT SET-THEORETIC

John C. REYNOLDS

Mai 1984

(This is a preprint of a paper to be given at the International Symposium on the Semantics of Data Types, Sophia-Antipolis, France, June 1984. It will appear in the symposium proceedings, in the Springer-Verlag Lecture Notes in Computer Science.)

POLYMORPHISM IS NOT SET-THEORETIC

John C. Reynolds

INRIA
Centre de Sophia Antipolis
06560 Valbonne, France
and
Syracuse University
Syracuse, New York 13210, U.S.A.

ABSTRACT The polymorphic, or second-order, typed lambda calculus is an extension of the typed lambda calculus in which polymorphic functions can be defined. In this paper, we will prove that the standard set-theoretic model of the ordinary typed lambda calculus cannot be extended to model this language extension.

RESUME Le lambda-calcul typé polymorphe, ou du second ordre, est une extension du lambda-calcul typé dans laquelle il est possible de définir des fonctions polymorphes. Dans cet article, nous démontrons que le modèle ensembliste usuel du lambda-calcul typé ne peut pas être étendu en un modèle de cette extension du langage.

Introduction

The polymorphic typed lambda calculus [1], or second-order typed lambda calculus [2], is an extension of the typed lambda calculus in which polymorphic functions can be defined by abstraction on type variables, and such functions can be applied to type expressions. In this paper, we will prove that the standard set-theoretic model of the ordinary typed lambda calculus cannot be extended to model this language extension.

In [3] it was conjectured that such a model might be produced by restricting polymorphic functions to "parametric polymorphism" in the sense of Strachey [4]. The result of this paper shows that this conjecture is false, regardless of the particular definition of "parametric".

Work supported by National Science Foundation Grant MCS-8017577.

Syntax of the Polymorphic Typed Lambda Calculus

To define the syntax of the polymorphic typed lambda calculus, we assume that we are given

T: An infinite countable set of *type variables*,

V: An infinite countable set of *ordinary variables*.

Then Ω, the set of *type expressions*, is the least set such that

If $\tau \in T$ then $\tau \in \Omega$,

If $\omega, \omega' \in \Omega$ then $\omega \to \omega' \in \Omega$,

If $\tau \in T$ and $\omega \in \Omega$ then $\Delta\tau.\, \omega \in \Omega$.

The operator $\Delta\tau.$ binds the occurrences of τ in $\Delta\tau.\, \omega$. More precisely, the set $\mathrm{FTV}\,\omega$ of type variables occurring *free* in ω is defined by

$$\mathrm{FTV}\,\tau = \{\tau\},$$

$$\mathrm{FTV}(\omega \to \omega') = \mathrm{FTV}\,\omega \cup \mathrm{FTV}\,\omega',$$

$$\mathrm{FTV}(\Delta\tau.\, \omega) = \mathrm{FTV}\,\omega - \{\tau\}.$$

This binding structure induces an obvious notion of alpha conversion (renaming of bound type variables) for type expressions. We will regard alpha-variants of a type expression as identical, and write $(\omega'/\tau \to \omega)$ to denote the type expression obtained from ω' by substituting ω for the free occurrences of τ, with the use of alpha conversion to avoid collisions of type variables.

Next, we define a *type assignment* π to be a function from a finite set of ordinary variables to Ω. We write

$$\Omega^* = \{\pi \mid \pi \in F \to \Omega \text{ for some finite } F \subset V\}$$

for the set of type assignments.

Finally, to define the syntax of *ordinary expressions*, we define the family

$$\langle E_{\pi\omega} \mid \pi \in \Omega^*, \omega \in \Omega \rangle$$

of sets, in which $E_{\pi\omega}$ is the set of those ordinary expressions whose free ordinary variables belong to the domain of π, and which take on the type ω under the assignment of types to ordinary variables given by π. This is the least family of sets satisfying

If $v \in \mathrm{dom}\,\pi$ then $v \in E_{\pi,\pi v}$,

If $e_1 \in E_{\pi,\omega\to\omega'}$ and $e_2 \in E_{\pi\omega}$ then $e_1(e_2) \in E_{\pi\omega'}$,

If $e \in E_{[\pi|v:\omega],\omega'}$ then $\lambda v:\omega.\, e \in E_{\pi,\omega\to\omega'}$,

If $e \in E_{\pi,\Delta\tau.\omega'}$ then $e[\omega] \in E_{\pi,(\omega'/\tau\to\omega)}$,

If $e \in E_{\pi-\tau,\omega}$ then $\Lambda\tau.\, e \in E_{\pi,\Delta\tau.\omega}$.

Here $\mathrm{dom}\,\pi$ denotes the domain of π, $[\pi \mid v:\omega]$ denotes the function with domain $\mathrm{dom}\,\pi \cup \{v\}$ such that $[\pi \mid v:\omega]v' =$ **if** $v' = v$ **then** ω **else** $\pi v'$, and $\pi - \tau$ denotes the restriction of π to the set

$$\{v \mid v \in \mathrm{dom}\,\pi \text{ and } \tau \notin \mathrm{FTV}(\pi v)\}$$

of ordinary variables that are assigned by π type expressions not containing free occurrences of τ.

By structural induction, it is easily seen that if π' is an extension of π then $E_{\pi'\omega} \supseteq E_{\pi\omega}$.

The sets $\mathrm{FTV}\,e$ of type variables and $\mathrm{FOV}\,e$ of ordinary variables that occur free in the ordinary expression e are defined by

$$\begin{aligned}
&\mathrm{FTV}\,v = \{\}, && \mathrm{FOV}\,v = \{v\},\\
&\mathrm{FTV}(e_1(e_2)) = \mathrm{FTV}\,e_1 \cup \mathrm{FTV}\,e_2, && \mathrm{FOV}(e_1(e_2)) = \mathrm{FOV}\,e_1 \cup \mathrm{FOV}\,e_2,\\
&\mathrm{FTV}(\lambda v:\omega.\, e) = \mathrm{FTV}\,\omega \cup \mathrm{FTV}\,e, && \mathrm{FOV}(\lambda v:\omega.\, e) = \mathrm{FOV}\,e - \{v\},\\
&\mathrm{FTV}(e[\omega]) = \mathrm{FTV}\,e \cup \mathrm{FTV}\,\omega, && \mathrm{FOV}(e[\omega]) = \mathrm{FOV}\,e,\\
&\mathrm{FTV}(\Lambda\tau.\, e) = \mathrm{FTV}\,e - \{\tau\}, && \mathrm{FOV}(\Lambda\tau.\, e) = \mathrm{FOV}\,e.
\end{aligned}$$

Set-Theoretic Semantics

In this section, we will detail the assumptions about a "set-theoretic semantics" of the polymorphic typed lambda calculus that we will use in proving that no such semantics exists. First we assume that, under an assignment of sets to type variables, each type expression denotes a set. As in [3], we formalize this assumption by defining a *set assignment* S to be a function from T to the class of sets, and writing $S^{\#}$ for the function from Ω to the class of sets such that $S^{\#}\omega$ is the set denoted by ω under the set assignment S. Since type variables should stand for the sets they are assigned, we require that $S^{\#}\tau = S\tau$ for $\tau \in T$, i.e. that $S^{\#}$ should be an extension of S.

We then define $S^{\#*}$ to be the function from Ω^* to the class of sets that maps each $\pi \in \Omega^*$ into the Cartesian product over $v \in \mathrm{dom}\,\pi$ of the sets $S^{\#}(\pi v)$. Thus $S^{\#*}\pi$ is the set of environments appropriate to π under S. In summary:

(1) If $S \in T \to \mathbf{S}$, where $\mathbf{S}$ denotes the class of sets, then $S^{\#} \in \Omega \to \mathbf{S}$ is an extension of S, and $S^{\#*} \in \Omega^* \to \mathbf{S}$ is such that

$$S^{\#*}\pi = \prod_{\upsilon \in \mathrm{dom}\,\pi} S^{\#}(\pi\upsilon).$$

Secondly, we assume that the meaning of an ordinary expression $e \in E_{\pi\omega}$ determines, for any set assignment S, a function from the set $S^{\#*}\pi$ of environments to the set $S^{\#}\omega$ of denotations. In other words:

(2) For each $\pi \in \Omega^*$ and $\omega \in \Omega$, there is a semantic function

$$\mu_{\pi\omega} \in E_{\pi\omega} \to \prod_{S \in \mathbf{SA}} (S^{\#*}\pi \to S^{\#}\omega),$$

where **SA** denotes the class of set assignments, and $s \to s'$ denotes the set of all functions from s to s'.

Next, we assume that the denotation of a typed or ordinary expression depends only upon the denotations of the typed and ordinary variables that occur free in the expression (or its type classification):

(3) If $S\tau = S'\tau$ for all $\tau \in \mathrm{FTV}\,\omega$, then $S^{\#}\omega = S'^{\#}\omega$.

(4) Suppose $e \in E_{\pi\omega}$ and $\eta \in S^{\#*}\pi$. If $S\tau = S'\tau$ for all $\tau \in \mathrm{FTV}\,e \cup \mathrm{FTV}\,\omega \cup \bigcup_{\upsilon \in \mathrm{dom}\,\pi} \mathrm{FTV}(\pi\upsilon)$ then $\mu_{\pi\omega}[\![e]\!]S\eta = \mu_{\pi\omega}[\![e]\!]S'\eta$.

(5) If $e \in E_{\pi\omega}$, π' extends π, and $\eta' \in S^{\#*}\pi'$ extends $\eta \in S^{\#*}\pi$, then $\mu_{\pi\omega}[\![e]\!]S\eta = \mu_{\pi'\omega}[\![e]\!]S\eta'$.

Next, we assume that our semantics agrees with the classical set-theoretic semantics of the ordinary typed lambda calculus for the constructions that belong to this sublanguage, and that beta reduction is sound:

(6) $S^{\#}(\omega \to \omega') = S^{\#}\omega \to S^{\#}\omega'$.

(7) If $\upsilon \in \mathrm{dom}\,\pi$ then

$$\mu_{\pi,\pi\upsilon}[\![\upsilon]\!]S\eta = \eta\upsilon.$$

(8) If $e_1 \in E_{\pi,\omega\to\omega'}$ and $e_2 \in E_{\pi\omega}$ then

$$\mu_{\pi\omega'}[\![e_1(e_2)]\!]S\eta = \mu_{\pi,\omega\to\omega'}[\![e_1]\!]S\eta\,(\mu_{\pi\omega}[\![e_2]\!]S\eta).$$

(9) If $e \in E_{[\pi|\upsilon:\omega],\omega'}$ then

$$\mu_{\pi,\omega\to\omega'}[\![\lambda\upsilon:\omega.\, e]\!]S\eta = \lambda x \in S^{\#}\omega.\, \mu_{[\pi|\upsilon:\omega],\omega'}[\![e]\!]S[\eta \mid \upsilon : x].$$

(10) If $e_1 \in E_{[\pi|\upsilon:\omega],\omega'}$, $e_2 \in E_{\pi\omega}$, and $e_3 \in E_{\pi\omega'}$ is obtained from e_1 by substituting e_2 for the free occurrences of υ, with alpha conversion to avoid collisions of variables, then

$$\mu_{\pi\omega'}[\![(\lambda\upsilon:\omega.\, e_1)(e_2)]\!]S\eta = \mu_{\pi\omega'}[\![e_3]\!]S\eta\,.$$

In (9), note the use of a lambda expression (of the ordinary typed lambda calculus) as part of the mathematical notation for defining semantics. To alleviate confusion with the language being defined, we use $\in$ instead of : in the binders of these lambda expressions.

Finally, we assume that the meaning of a type application depends only upon the meanings of its subexpressions, and that type beta reduction is sound. In particular:

(11) If $\mu_{\pi,\Delta\tau.\omega'}[\![e_1]\!] = \mu_{\pi,\Delta\tau.\omega'}[\![e_2]\!]$ then $\mu_{\pi,(\omega'/\tau\to\omega)}[\![e_1[\omega]]\!] = \mu_{\pi,(\omega'/\tau\to\omega)}[\![e_2[\omega]]\!]$.

(12) If $e \in E_{\pi-\tau,\omega}$ then $\mu_{\pi-\tau,\omega}[\![(\Lambda\tau.\, e)[\tau]]\!] = \mu_{\pi-\tau,\omega}[\![e]\!]$.

The Contradiction

We write $[\upsilon_1{:}X_1 \mid \cdots \mid \upsilon_n{:}X_n]$ for the set assignment or environment with domain $\{\upsilon_1, \ldots, \upsilon_n\}$ that maps each υ_i into X_i. In particular, we write $[\,]$ for the set assignment or environment whose domain is empty. We write $f\,;g$ for the composition of functions in diagrammatic order, i.e. $(f\,;g)x = g(fx)$.

Our argument will involve working out the denotations of certain specific expressions. To clarify the exposition, we will use boldface lower-case letters for particular type and ordinary variables, and boldface upper-case letters for particular type and ordinary expressions. The corresponding italic letters will be used for the denotations of these variables and expressions.

Lemma 1. If the polymorphic typed lambda calculus has a set-theoretic model then there is a type expression $\mathbf{B}$, containing no free occurrences of type variables, whose denotation $B = S^{\#}\mathbf{B}$ (which is independent of S by (3) since $\mathrm{FTV}\,\mathbf{B} = \{\}$) contains more than one element.

Proof: Let $\mathbf{B} = \Delta\mathbf{s}.\, \mathbf{s}\to(\mathbf{s}\to\mathbf{s})$ and S be a set assignment mapping $\mathbf{s}$ into a set s containing more than one element. Then B contains

$$\mu_{[],\mathbf{B}}[\![\Lambda\mathbf{s}.\, \lambda\mathbf{x}{:}\,\mathbf{s}.\, \lambda\mathbf{y}{:}\,\mathbf{s}.\, \mathbf{x}]\!]S[\,] \quad\text{and}\quad \mu_{[],\mathbf{B}}[\![\Lambda\mathbf{s}.\, \lambda\mathbf{x}{:}\,\mathbf{s}.\, \lambda\mathbf{y}{:}\,\mathbf{s}.\, \mathbf{y}]\!]S[\,].$$

If these members of B were the same, we would have

$$\mu_{[],\mathbf{s}\to(\mathbf{s}\to\mathbf{s})}[\![(\Lambda\mathbf{s}.\, \lambda\mathbf{x}{:}\,\mathbf{s}.\, \lambda\mathbf{y}{:}\,\mathbf{s}.\, \mathbf{x})[\mathbf{s}]]\!]S[\,] = \mu_{[],\mathbf{s}\to(\mathbf{s}\to\mathbf{s})}[\![(\Lambda\mathbf{s}.\, \lambda\mathbf{x}{:}\,\mathbf{s}.\, \lambda\mathbf{y}{:}\,\mathbf{s}.\, \mathbf{y})[\mathbf{s}]]\!]S[\,]$$

by (11), and

$$\mu_{[],\mathbf{s}\to(\mathbf{s}\to\mathbf{s})}[\![\lambda\mathbf{x}\colon \mathbf{s}.\ \lambda\mathbf{y}\colon \mathbf{s}.\ \mathbf{x}]\!]S[\,] = \mu_{[],\mathbf{s}\to(\mathbf{s}\to\mathbf{s})}[\![\lambda\mathbf{x}\colon \mathbf{s}.\ \lambda\mathbf{y}\colon \mathbf{s}.\ \mathbf{y}]\!]S[\,]$$

by (12). Then, for all x, $y \in s$, by (9) and (7),

$$\mu_{[],\mathbf{s}\to(\mathbf{s}\to\mathbf{s})}[\![\lambda\mathbf{x}\colon \mathbf{s}.\ \lambda\mathbf{y}\colon \mathbf{s}.\ \mathbf{x}]\!]S[\,]xy = x = \mu_{[],\mathbf{s}\to(\mathbf{s}\to\mathbf{s})}[\![\lambda\mathbf{x}\colon \mathbf{s}.\ \lambda\mathbf{y}\colon \mathbf{s}.\ \mathbf{y}]\!]S[\,]xy = y\,,$$

which would contradict the assumption that s contains more than one element. *(End of proof.)*

In the sequel, **B** will stand for the closed type expression whose existence is guaranteed by Lemma 1, and B for the multi-element set that it denotes. In fact, Lemma 1 is only needed because we are considering a simple form of the polymorphic typed lambda calculus without any type constants, such as **Boolean**, that denote specific multi-element sets. The reader may find our arguments easier to follow if he regards **B** as **Boolean** rather than $\Delta\mathbf{s}.\ \mathbf{s}\to(\mathbf{s}\to\mathbf{s})$.

The next step is to introduce a functor T from the category of sets and functions to itself. This is the functor such that, for any set s,

$$Ts = (s\to B)\to B\,,$$

and for any function $\rho \in s\to s'$, $T\rho \in Ts\to Ts'$ is the function such that

$$T\rho = \lambda h \in (s\to B)\to B.\ \lambda g \in s'\to B.\ h(\rho;g)\,.$$

Both the object and morphism parts of T can be "expressed" in the polymorphic typed lambda calculus. This is the key to the following lemma:

Lemma 2. If the polymorphic typed lambda calculus has a set-theoretic model, then there is a set P and a function $H \in TP\to P$ such that, for any set s and any function $f \in Ts\to s$, there is a function $\rho_{sf} \in P\to s$ such that

$$\begin{array}{ccc} TP & \xrightarrow{T\rho_{sf}} & Ts \\ \Big\downarrow{\scriptstyle H} & & \Big\downarrow{\scriptstyle f} \\ P & \xrightarrow{\rho_{sf}} & s \end{array} \qquad \text{(A)}$$

commutes.

Proof: Let $\mathbf{W}$ and $\mathbf{P}$ be the type expressions

$$\mathbf{W} = (((\mathbf{s}\to\mathbf{B})\to\mathbf{B})\to\mathbf{s})\to\mathbf{s}, \qquad \mathbf{P} = \Delta\mathbf{s}.\,\mathbf{W},$$

and let

$$P = S^{\#}\mathbf{P},$$

which is independent of S by (3) since $\mathrm{FTV}\,\mathbf{P} = \{\}$. Let $\mathbf{M}$ and $\mathbf{H}$ be the ordinary expressions

$$\mathbf{M} = \lambda\mathbf{f}\colon((\mathbf{s}\to\mathbf{B})\to\mathbf{B})\to\mathbf{s}.\ \mathbf{f}(\lambda\mathbf{g}\colon\mathbf{s}\to\mathbf{B}.\ \mathbf{h}(\lambda\mathbf{p}\colon\mathbf{P}.\ \mathbf{g}(\mathbf{p}[\mathbf{s}]\mathbf{f}))) \in E_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{W}},$$

$$\mathbf{H} = \lambda\mathbf{h}\colon(\mathbf{P}\to\mathbf{B})\to\mathbf{B}.\ \Lambda\mathbf{s}.\ \mathbf{M} \in E_{[],((\mathbf{P}\to\mathbf{B})\to\mathbf{B})\to\mathbf{P}},$$

and let

$$H = \mu_{[],((\mathbf{P}\to\mathbf{B})\to\mathbf{B})\to\mathbf{P}}[\![\mathbf{H}]\!]S[\,] \in S^{\#}((\mathbf{P}\to\mathbf{B})\to\mathbf{B})\to\mathbf{P} = ((P\to B)\to B)\to P = TP\to P,$$

which is independent of S by (4), since $\mathrm{FTV}\,\mathbf{H}$, $\mathrm{FTV}(((\mathbf{P}\to\mathbf{B})\to\mathbf{B})\to\mathbf{P})$, and $\mathrm{dom}[\,]$ are all empty.

Finally, for any set s and function $f \in Ts\to s = ((s\to B)\to B)\to s$, let $\rho_{sf} \in P\to s$ be the function such that

$$\rho_{sf}\,p = \mu_{[\mathbf{p}:\mathbf{P}],\mathbf{W}}[\![\mathbf{p}[\mathbf{s}]]\!][S \mid \mathbf{s}\colon s][\mathbf{p}\colon p]f,$$

which is independent of S by (4), since $\mathrm{FTV}(\mathbf{p}[\mathbf{s}]) = \{\mathbf{s}\}$, $\mathrm{FTV}\,\mathbf{W} = \{\mathbf{s}\}$, and $\mathrm{FTV}\,\mathbf{P} = \{\}$.

Then, for any set s, $f \in ((s\to B)\to B)\to s$, and $h \in (P\to B)\to B$,

$$\begin{aligned}
&\rho_{sf}\,(Hh)\\
&= \rho_{sf}\,(\mu_{[],((\mathbf{P}\to\mathbf{B})\to\mathbf{B})\to\mathbf{P}}[\![\mathbf{H}]\!]S[\,]h) && \text{(def. of } H)\\
&= \rho_{sf}\,(\mu_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{P}}[\![\Lambda\mathbf{s}.\ \mathbf{M}]\!]S[\mathbf{h}\colon h]) && (9)\\
&= \mu_{[\mathbf{p}:\mathbf{P}],\mathbf{W}}[\![\mathbf{p}[\mathbf{s}]]\!][S \mid \mathbf{s}\colon s][\mathbf{p}\colon \mu_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{P}}[\![\Lambda\mathbf{s}.\ \mathbf{M}]\!]S[\mathbf{h}\colon h]]f && (\text{def. of } \rho_{sf})\\
&= \mu_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{W}}[\![(\lambda\mathbf{p}\colon\mathbf{P}.\ \mathbf{p}[\mathbf{s}])(\Lambda\mathbf{s}.\ \mathbf{M})]\!][S \mid \mathbf{s}\colon s][\mathbf{h}\colon h]f && (9,\ 5,\ 4,\ \text{and } 8)\\
&= \mu_{[\mathbf{h}:(\mathbf{P}\to\mathbf{B})\to\mathbf{B}],\mathbf{W}}[\![\mathbf{M}]\!][S \mid \mathbf{s}\colon s][\mathbf{h}\colon h]f && (10 \text{ and } 12)\\
&= f\,(\lambda g \in s\to B.\ h(\lambda p \in P.\ g\,(\mu_{[\mathbf{p}:\mathbf{P}],\mathbf{W}}[\![\mathbf{p}[\mathbf{s}]]\!][S \mid \mathbf{s}\colon s][\mathbf{p}\colon p]f))) && (9,\ 8,\ 7,\ \text{and } 5)\\
&= f\,(\lambda g \in s\to B.\ h(\lambda p \in P.\ g\,(\rho_{sf}\,p))) && (\text{def. of } \rho_{sf})\\
&= f\,(\lambda g \in s\to B.\ h(\rho_{sf}\,;g))\\
&= f\,(T\rho_{sf}\,h).
\end{aligned}$$

(End of proof.)

Lemmas 1 and 2 are the only properties of a set-theoretic semantics of the polymorphic typed lambda calculus that will be needed in our proof; the remainder of the argument depends only upon the nature of sets and functions.

Readers who are familiar with the concept of a T-algebra [5], or prefixed point of T [6], or T-dynamics [7], will recognize Lemma 2 as an assertion that there is a T-algebra $\langle P, H \rangle$ such that for any T-algebra $\langle s, f \rangle$ there is at least one morphism ρ_{sf} from $\langle P, H \rangle$ to $\langle s, f \rangle$. The purpose of the next two lemmas is to replace "at least one" by "exactly one", i.e. to show the existence of an initial T-algebra.

Lemma 3. If the polymorphic typed lambda calculus has a set-theoretic model, than there is a set P' and a function $H' \in TP' \to P'$ such that, for any set s and any function $f \in Ts \to s$, there is a function $\rho'_{sf} \in P' \to s$ such that

$$\begin{array}{ccc} TP' & \xrightarrow{T\vartheta'} & Ts \\ \downarrow{\scriptstyle H'} & & \downarrow{\scriptstyle f} \\ P' & \xrightarrow{\vartheta'} & s \end{array} \qquad \text{(B)}$$

commutes if $\vartheta' = \rho'_{sf}$, and if this diagram commutes then $\rho'_{sf} = \rho'_{P'H'}; \vartheta'$.

Proof: Let P, H, and ρ_{sf} be as in Lemma 2. For $p \in P$, we say p is *parametric* when, for all sets s_1 and s_2 and functions $\alpha \in s_1 \to s_2$, $f_1 \in Ts_1 \to s_1$, and $f_2 \in Ts_2 \to s_2$, if

$$\begin{array}{ccc} Ts_1 & \xrightarrow{T\alpha} & Ts_2 \\ \downarrow{\scriptstyle f_1} & & \downarrow{\scriptstyle f_2} \\ s_1 & \xrightarrow{\alpha} & s_2 \end{array} \qquad \text{(C)}$$

commutes then $\rho_{s_2 f_2} p = \alpha(\rho_{s_1 f_1} p)$. Let P' be the set

$$P' = \{p \mid p \in P \text{ and } p \text{ is parametric}\}$$

of parametric members of P, and J be the identity injection from P' to P.

(This definition of "parametric" is possibly weaker than any of those in [3]. In fact, it coincides with the weakening of (PAR) in [3] obtained by requiring the relation r to be the function α.)

Now suppose that diagram (C) commutes. Since all members of the image of J are parametric, we have

$$J;\rho_{s_2f_2} = J;\rho_{s_1f_1};\alpha.$$

Applying T to both sides, composing on the right with f_2, and using (C), we get

$$TJ;T\rho_{s_2f_2};f_2 = TJ;T\rho_{s_1f_1};f_1;\alpha.$$

Then since (A) commutes for s_1, f_1 and for s_2, f_2, we have

$$TJ;H;\rho_{s_2f_2} = TJ;H;\rho_{s_1f_1};\alpha.$$

Thus, for all $h' \in TP'$,

$$\rho_{s_2f_2}(H(TJh')) = \alpha(\rho_{s_1f_1}(H(TJh'))),$$

and since this result holds for all s_1, s_2, α, f_1, and f_2 that make (C) commute, $H(TJh')$ is parametric for all $h' \in TP'$.

This permits us to define H' to be the corestriction of $TJ;H$ to P', and insures that $H';J = TJ;H$. Then, if we define $\rho'_{sf} = J;\rho_{sf}$, this equation and (A) imply that, for any set s and $f \in Ts \to s$,

$$H';\rho'_{sf} = H';J;\rho_{sf} = TJ;H;\rho_{sf} = TJ;T\rho_{sf};f = T\rho'_{sf};f.$$

Thus (B) commutes when $\vartheta' = \rho'_{sf}$.

On the other hand, suppose ϑ' is any function in $P' \to s$ that makes (B) commute. Then since (B) is a special case of (C) and the members of P' are all parametric,

$$\rho_{sf}p = \vartheta'(\rho_{P'H'}p)$$

holds for all $p \in P'$, and since P' is the image of J,

$$\rho'_{sf} = J;\rho_{sf} = J;\rho_{P'H'};\vartheta' = \rho'_{P'H'};\vartheta'.$$

(End of proof.)

As an aside, consider the special case where the set-theoretic model permits only parametric polymorphic functions. In this case, all members of P would be parametric, and the proof of Lemma 3 would become a triviality, with $P' = P$, $H' = H$, and $\rho'_{sf} = \rho_{sf}$.

Lemma 4. If the polymorphic typed lambda calculus has a set-theoretic model, then there is a set P'' and a function $H'' \in TP'' \to P''$ such that, for any set s and any function $f \in Ts \to s$, there is a function $\rho''_{sf} \in P'' \to s$ such that

$$\begin{array}{ccc} TP'' & \xrightarrow{T\vartheta''} & Ts \\ \downarrow{\scriptstyle H''} & & \downarrow{\scriptstyle f} \\ P'' & \xrightarrow{\vartheta''} & s \end{array} \qquad \text{(D)}$$

commutes if and only if $\vartheta'' = \rho''_{sf}$.

Proof: Let P', H', and ρ'_{sf} be as in Lemma 3. Consider the special case where $s = P'$ and $f = H'$, and let ρ'_0 abbreviate the function $\rho'_{P'H'} \in P' \to P'$. Then taking $\vartheta' = \rho'_0$, we find that

$$\begin{array}{ccc} TP' & \xrightarrow{T\rho'_0} & TP' \\ \downarrow{\scriptstyle H'} & & \downarrow{\scriptstyle H'} \\ P' & \xrightarrow{\rho'_0} & P' \end{array} \qquad \text{(E)}$$

commutes, and that $\rho'_0 = \rho'_0; \rho'_0$.

Let P'' be the image of ρ'_0, $\Gamma \in P' \to P''$ be the corestriction of ρ'_0 to P'', and K be the identity injection from P'' to P'. Then $\Gamma; K = \rho'_0$ and, since ρ'_0 is idempotent, $K; \Gamma$ is the identity function on P''.

Let $H'' \in TP'' \to P''$ be $H'' = TK; H'; \Gamma$. Then

$$H''; K = TK; H'; \Gamma; K = TK; H'; \rho'_0 = TK; T\rho'_0; H' = TK; T\Gamma; TK; H' \qquad \text{(F)}$$
$$= TK; H',$$

and

$$H';\Gamma = H';\Gamma;K;\Gamma = H';\rho'_0;\Gamma = T\rho'_0;H';\Gamma = T\Gamma;TK;H';\Gamma = T\Gamma;H''. \qquad (G)$$

For any set s and $f \in Ts \to s$, let $\rho''_{sf} \in P'' \to s$ be $\rho''_{sf} = K;\rho'_{sf}$. From (F) and Lemma 3, we have

$$H'';\rho''_{sf} = H'';K;\rho'_{sf} = TK;H';\rho'_{sf} = TK;T\rho'_{sf};f = T\rho''_{sf};f\ ,$$

so that (D) commutes when $\vartheta'' = \rho''_{sf}$.

On the other hand, suppose ϑ'' is any function in $P'' \to s$ that makes (D) commute. From (G) and (D),

$$H';\Gamma;\vartheta'' = T\Gamma;H'';\vartheta'' = T\Gamma;T\vartheta'';f = T(\Gamma;\vartheta'');f\ ,$$

so that (B) commutes when ϑ' is taken to be $\Gamma;\vartheta''$, and Lemma 3 implies $\rho'_{sf} = \rho'_0;\Gamma;\vartheta''$. Then

$$\rho''_{sf} = K;\rho'_{sf} = K;\rho'_0;\Gamma;\vartheta'' = K;\Gamma;K;\Gamma;\vartheta'' = \vartheta''.$$

(End of proof.)

As an aside, we consider the relationship between Lemma 4 and polymorphic extensionality. To say that the "parametric polymorphic functions" in P' are extensional is to say that, for all $p_1, p_2 \in P'$, $p_1 = p_2$ whenever

$$\mu_{[\mathbf{p}:\mathbf{P}],\mathbf{w}}[\![\mathbf{p}[\mathbf{s}]]\!][S \mid \mathbf{s}:s][\mathbf{p}:p_1] = \mu_{[\mathbf{p}:\mathbf{P}],\mathbf{w}}[\![\mathbf{p}[\mathbf{s}]]\!][S \mid \mathbf{s}:s][\mathbf{p}:p_2]$$

holds for all sets s. If this property holds then, by the definitions of ρ_{sf} and ρ'_{sf} in the proofs of Lemmas 2 and 3, $p_1 = p_2$ whenever $\rho'_{sf}p_1 = \rho'_{sf}p_2$ holds for all set s and functions $f \in Ts \to s$.

By Lemma 3, however, $\rho'_{sf} = \rho'_{P'H'};\rho'_{sf}$, so that $\rho'_{sf}p = \rho'_{sf}(\rho'_{P'H'}p)$ holds for all s and f, and for all $p \in P'$. Thus extensionality would imply that $\rho'_{P'H'}$ is the identity function on P'. In this situation, Lemma 4 would become a triviality, with $P'' = P'$, $H'' = H'$, and $\rho''_{sf} = \rho'_{sf}$.

In the language of T-algebras, Lemma 4 asserts that $<P'', H''>$ is an initial T-algebra (or initial prefixed point of T). The final step in our development is based on the proof in [5, 6] that initiality implies that H'' is an isomorphism (i.e. that an initial prefixed point is a fixed point).

Theorem. There is no set-theoretic model of the polymorphic typed lambda calculus.

Proof: Assume the contrary, and let P'', H'', and ρ''_{sf} be as in Lemma 4. Let $G'' = \rho''_{TP'',TH''} \in P''\to TP''$. Taking $s = TP''$ and $f = TH'' \in T(TP'')\to TP''$ in (D), we have $H'';G'' = TG'';TH''$. Then

$$H'';(G'';H'') = TG'';TH'';H'' = T(G'';H'');H''.$$

On the other hand, obviously

$$H'';I_{P''} = I_{TP''};H'' = TI_{P''};H''$$

(where I_s denotes the identity function on s).

These equations both match against (D), taking ϑ'' to be $G'';H''$ and $I_{P''}$ respectively. Thus, since (D) commutes for a unique ϑ'', we have

$$G'';H'' = I_{P''}.$$

But then,

$$H'';G'' = TG'';TH'' = TI_{P''} = I_{TP''}.$$

Thus H'' and G'' give an isomorphism between $TP'' = (P''\to B)\to B$ and P'', where by Lemma 1 the set B contains more than one element. Since $(P''\to B)\to B$ and P'' are well-known to have different cardinalities, we have a contradiction. *(End of proof.)*

Conclusion

In summary, we have shown that a set-theoretic model of the polymorphic typed lambda calculus could only exist if there were a set P'' isomorphic to TP''. The argument seems to generalize to any functor $T \in \mathrm{SET}\to\mathrm{SET}$ that can be constructed from the functor $\to \in \mathrm{SET}^{\mathrm{op}}\times\mathrm{SET}\to\mathrm{SET}$. Thus, to find a model one must replace SET by some Cartesian closed category in which such isomorphisms possess solutions.

This obviously suggests using some subcategory of the category of complete partial orders and continuous functions. Indeed, two models using such subcategories have been devised by N. McCracken [8, 9]. However, it would be premature to conclude that complete partial orders are a prerequisite for a successful model, since the need for complete partial orders normally arises from some kind of nontermination, yet all expressions of the polymorphic typed lambda calculus have normal forms [10].

Acknowledgement

I am especially grateful to Thierry Despeyroux for his patient assistance with the mysteries of computer text-formatting.

References

[1] Reynolds, J. C., Towards a Theory of Type Structure, *Proc. Colloque sur la Programmation*, Lecture Notes in Computer Science *19*, Springer-Verlag, New York, 1974, pp. 408-425.

[2] Girard, J.-Y., Interprétation Fonctionelle et Elimination des Coupures dans l'Arithmétique d'Ordre Supérieur, Thèse de Doctorat d'Etat, Paris, 1972.

[3] Reynolds, J. C., Types, Abstraction and Parametric Polymorphism, *Information Processing 83*, R. E. A. Mason (ed.), Elsevier Science Publishers B.V. (North-Holland) 1983, pp. 513-523.

[4] Strachey, C., Fundamental Concepts in Programming Languages, Lecture Notes, International Summer School in Computer Programming, Copenhagen, August 1967.

[5] Lehmann, D., and Smyth, M. B., Algebraic Specification of Data Types: A Synthetic Approach, *Math. Systems Theory 14* (1981), pp. 97-139.

[6] Smyth, M. B., and Plotkin, G. D., The Category-Theoretic Solution of Recursive Domain Equations, *SIAM Journal on Computing 11*, 4 (November 1982), pp. 761-783.

[7] Arbib, M. A., and Manes, E. G., *Arrows, Structures, and Functors: The Categorical Imperative*, Academic Press, New York, 1975, p. 95.

[8] McCracken, N. J., An Investigation of a Programming Language with a Polymorphic Type Structure, Ph. D. dissertation, Syracuse University, June 1979.

[9] McCracken, N. J., A Finitary Retract Model for the Polymorphic Lambda-Calculus, to appear in *Information and Control*.

[10] Fortune, S., Leivant, D., and O'Donnell, M., The Expressiveness of Simple and Second-Order Type Structures, *Journal of the ACM 30*, 1 (January 1983), pp. 151-185.

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique