Journal of Computer and System Sciences 80 (2014) 1163–1173

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

On polymorphic types of untyped terms

Rick Statman

Carnegie Mellon University, Department of Mathematical Sciences, Pittsburgh, PA 15213, United States

A R T I C L E   I N F O

Article history:
Received 16 December 2012
Received in revised form 21 October 2013
Accepted 3 February 2014
Available online 4 April 2014

Keywords:
Lambda calculus
Lambda calculus with surjective pairing
Polymorphic typings
Böhm's theorem

A B S T R A C T

Let **S** be a finite set of β normal closed terms and M and N a pair of β normal, η distinct, closed terms. Then there exist polymorphic types a, b such that every member of **S** can be typed as a, and M and N have η expansions which can be typed as b; where, in the resulting typings, the members of **S** can be simultaneously consistently identified, and the η expansions of M and N are $\beta\eta$ inconsistent (no model with more than one element of any type). A similar result holds in the presence of surjective pairing.

1. Introduction

Alonzo Church believed that only β normal untyped lambda terms have meaning and Corrado Böhm [1] showed that no two such η inconvertible terms can be consistently identified in the untyped calculus. More precisely, such an identification implies that all terms are equal.

Theorem 1.1. *(See Böhm [1].) Let M and N be distinct $\beta\eta$ closed normal forms and P, Q arbitrary. Then there exists R such that $RM\beta$ converts to P and $RN\beta$ converts to Q.*

In addition, any β normal term can be assigned a polymorphic type in Girard's system F [3] (I learned this from John Reynolds some 25 year ago, but I am not sure of the origin), so the question arises as to the consistency of identification after various polymorphic typings. Indeed, Corrado had already asked me, during a visit to Roma, whether his theorem is true in various typed contexts. Below we shall prove a very strong result.

We work in the untyped lambda calculus, but we will type certain terms with polymorphic types (from Girard's system F_0 [3]). We shall also consider the untyped lambda calculus with surjective pairing. In this case we will type untyped terms with polymorphic types which we will assume are closed under pairing. This is somewhat stronger than having products but somewhat weaker than assuming surjective pairing on the typed side.

Polymorphic types $a, b, c, \ldots$ are built up from type variables $p, q, r, \ldots$ by $\rightarrow$ and $\forall$ (under the Curry–Howard isomorphism, the universal quantifier). When typing an untyped term we require that each term variable have a fixed type and that all subterms are simultaneously compatibly typed (sometimes referred to as "Church typing"). In addition, when typing an untyped term, the type operations of abstraction ($\Lambda p, \Lambda q, \Lambda r, \ldots$) corresponding to $\forall$, and application can be inserted before and after symbols in the term. We shall prove the following theorem which we state first for the case without pairing.

E-mail address: statman@cs.cmu.edu.

http://dx.doi.org/10.1016/j.jcss.2014.04.008
0022-0000/© 2014 Elsevier Inc. All rights reserved.

Let **S** be a finite set of β normal closed terms and M and N a pair of β normal, η distinct, closed terms. Then there exist polymorphic types a, b such that every member of **S** can be typed as a, and M and N have η expansions which can be typed as b; where, in the resulting typings, the members of **S** can be simultaneously consistently identified, and the η expansions of M and N are $\beta\eta$ inconsistent (no model with more than one element of any type).

Here, all the members of **S** can be identified in a non-trivial $\beta\eta$ model of the polymorphic lambda calculus with no empty types.

2. Untyped terms

We begin with some preliminaries for the untyped calculus with pairing. The atoms of the language consist of variables $x, y, z, \ldots$ and constants P, L, R. Terms of the language are defined recursively: atoms are terms and if X, Y are terms then so are (XY) and λxX.

We shall adopt the customary conventions:

- parens are deleted and restored by left association and the use of Church's infixed "dot" notation;
- parens are added around abstractions, and additional unary operations for readability.

The axiom and rules of untyped lambda calculus are the following.

The first 5 axioms correspond to the classical theory of untyped lambda calculus with surjective pairing *SP* [2].

$$\begin{array}{lll}
(\beta) & (\lambda xX)\,Y & = [Y/x]X \\
(\eta) & X & = \lambda x.\,Xx \quad x \text{ not free in } X \\
(L/Pa) & L(PXY) & = X \\
(R/Pa) & R(PXY) & = Y \\
(P/Dp) & P(LX)(RX) & = X
\end{array}$$

The next 6 axioms correspond to the extended theory of Stovering (FP) [6] and Statman (PSP, in the combinator case) [5], which enjoys the Church Rosser property when formulated by reductions.

$$\begin{array}{lll}
(Ap/P) & PXYZ & = P(XZ)(YZ) \\
(L/Ap) & LXY & = L(XY) \\
(R/Ap) & RXY & = R(XY) \\
(L/Ab) & L(\lambda xX) & = \lambda x(LX) \\
(R/Ab) & R(\lambda xX) & = \lambda x(RX) \\
(P/Ab) & P(\lambda xX)(\lambda xY) & = \lambda x(PXY)
\end{array}$$

There are certain useful derived rules.

(1) (P/Dp) and $(Ap/P) \Rightarrow (L/Ap)$ and (R/Ap)

$$L(XY) = L\big(P(LX)(RX)Y\big) = L\big(P(LXY)(RXY)\big) = LXY$$

similarly for R

(2) (L/Ap) and (R/Ap) and $(P/Dp) \Rightarrow (Ap/P)$

$$L(PXYZ) = L(PXY)Z = XZ \quad \text{and} \quad R(PXYZ) = R(PXY)Z = YZ$$

therefore

$$PXYZ = P\big(L(PXYZ)\big)\big(R(PXYZ)\big) = P(XZ)(YZ).$$

(3) (η) and $(P/Ap) \Rightarrow (P/Ab)$

$$P(\lambda xX)(\lambda xY) = \lambda y.\,P(\lambda xX)(\lambda xY)y = \lambda y.\,P\big((\lambda xX)y\big)\big((\lambda XY)y\big) = \lambda x\,PXY$$

(4) (η) and $(L/Ap) \Rightarrow (L/Ab)$

$$L(\lambda xX) = \lambda y.\,L(\lambda xX)y = \lambda y.\,L\big((\lambda xX)y\big) = \lambda x(LX)$$

similarly for R

(5) $(\eta) \Rightarrow LP = K$ and $RP = K^*$

$$L(PX) = \lambda x.\,L(PX)x = \lambda x.\,L(PXx) = \lambda x.\,X$$

thus