Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 325 (2016) 305–312

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

How to Think of Intersection Types as Cartesian Products

Rick Statman[1]

Mathematical Sciences
Carnegie Mellon University
Pittsburgh, PA
USA

Abstract

In their paper "Intersection types and lambda definability" [3] Bucciarelli, Piperno, and Salvo give a mapping of the strongly normalizable untyped terms into the simply typed terms via the assignment of intersection types. Here we shall both generalize their result and provide a converse. We shall do this by retracting untyped terms with surjective pairing onto untyped terms without pairing by using a special variant of Stovring's notion [8] of a symmetric term. The symmetric ones which have simple types with Cartesian products are precisely the ones which retract onto (eta expansions of) strongly normalizable untyped terms. The intersection types of the strongly normalizable terms are related to the simple types with products in that we just replace $\wedge$ by Cartesian product and vice versa.

Keywords: lambda calculus, semigroups, representation

1 Introduction

In their paper "Intersection types and lambda definability" [3] Bucciarelli, Piperno, and Salvo give a mapping of the strongly normalizable untyped terms into the simply typed terms via the assignment of intersection types. Here we shall both generalize their result and provide a converse. We shall do this by retracting untyped terms with surjective pairing onto untyped terms without pairing by using a special variant of Stovring's notion [8] of a symmetric term. The symmetric ones which have simple types with Cartesian products are precisely the ones which retract onto (eta expansions of) strongly normalizable untyped terms. The intersection types of the strongly normalizable terms are related to the simple types with products in that we just replace $\wedge$ by Cartesian product and vice versa.

[1] Email: statman@cs.cmu.edu

http://dx.doi.org/10.1016/j.entcs.2016.09.045

Definition 1.1 Atoms of the language:

- the variables $x, y, z, \ldots$ are atoms, and
- the constants P, L, R are atoms.

Definition 1.2 Terms of the language:

- atoms are terms, and
- if X, Y are terms then so are (XY) and $\lambda x X$.

We shall adopt the customary conventions:

(i) parens are deleted and restored by left association and the use of Church's infixed "dot" notation

(ii) 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.

(beta)	$(\lambda x X)Y$	$= [Y/x]X$	
(eta)	X	$= \lambda x.\, Xx$	x not free in X
(L/Pa)	$L(PXY)$	$= X$	
(R/Pa)	$R(PXY)$	$= Y$	
(P/Dp)	$P(LX)(RX)$	$= X$	

The next 6 axioms correspond to the extended theory of Stovring (FP) and Statman (PSP [7], in the combinator case), which enjoys the Church-Rosser property when formulated by reductions.

(P/Ap)	$PXYZ$	$= P(XZ)(YZ)$
(L/Ap)	LXY	$= L(XY)$
(R/Ap)	RXY	$= R(XY)$
(L/Ab)	$L(\lambda x X)$	$= \lambda x(LX)$
(R/Ab)	$R(\lambda x X)$	$= \lambda x(RX)$
(P/Ab)	$P(\lambda x X)(\lambda x Y)$	$= \lambda x\, PXY$

There are certain useful derived rules:

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

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

and similarly for R.

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

$L(PXYZ) = L(PXY)Z = XZ$ and $R(PXYZ) = R(PXY)Z = YZ$

and therefore