Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 136 (2005) 153–172
www.elsevier.com/locate/entcs

Intersection and Union Types in the $\overline{\lambda}\mu\tilde{\mu}$-calculus

Daniel J. Dougherty [1]

Worcester Polytechnic Institute
Worcester MA 01609 USA

Silvia Ghilezan[2]

Faculty of Engineering, University of Novi Sad
Novi Sad, Serbia

Pierre Lescanne[3]

LIP, École Normale Supérieure de Lyon
Lyon, France

Abstract

The original $\overline{\lambda}\mu\tilde{\mu}$ of Curien and Herbelin has a system of simple types, based on sequent calculus, embodying a Curry-Howard correspondence with classical logic. We introduce and discuss three type assignment systems that are extensions of $\overline{\lambda}\mu\tilde{\mu}$ with intersection and union types. The intrinsic symmetry in the $\overline{\lambda}\mu\tilde{\mu}$ calculus leads to an essential use of both intersection and union types.

Keywords: Intersection types, union types, $\overline{\lambda}\mu\tilde{\mu}$-calculus, classical logic, Curry-Howard correspondence.

[1] Email: dd@cs.wpi.edu
[2] Email: gsilvia@uns.ns.ac.yu
[3] Email: Pierre.Lescanne@ens-lyon.fr

1571-0661/$ – see front matter
doi:10.1016/j.entcs.2005.06.010

1 Introduction

Intersection types were introduced into the lambda calculus in the late 1970s by Coppo and Dezani [5,6], Pottinger [19] and Sallé [23]. Intersection type assignment systems were devised in order to type more lambda terms than the basic functional, or simply typed, system; indeed, these intersection types systems can characterize exactly all strongly normalizing lambda terms. In addition, these systems are suitable for analyzing λ-models and various normalization properties of λ-terms. A summary of the early research was given by van Bakel [24].

Barbanera et al. [1] added union types to intersection type systems. This work was motivated by the observation that union types arise naturally in denotational semantics and that they can generate more informative types for some terms. However one cannot type with union types more terms than with intersection types only. That is, the system with intersection and union types exactly characterize all strongly normalizing terms as well.

In the 1980's and early 1990's Reynolds explored the role that intersection types can play in a practical programming language (see for example the report [20] on the language Forsythe). Pierce [18] explored the use of union types in programming, for example as a generalization of variant records. More recently Buneman and Pierce [3], have shown how union types can play a key role in the design of query languages for semistructured data union types.

Under the Curry-Howard correspondence formulae provable in intuitionistic logic coincide with types inhabited in simply typed lambda calculus. Griffin extended the Curry-Howard correspondence to classical logic in his seminal 1990 POPL paper [13], by observing that classical tautologies suggest typings for certain control operators. This initiated an active line of research; in particular the $\lambda\mu$ calculus of Parigot [17] embodies a Curry-Howard correspondence for classical logic based on natural deduction.

Meanwhile Curien and Herbelin [7], building on earlier work in [14], defined the system $\overline{\lambda}\mu\widetilde{\mu}$. In contrast to Parigot's $\lambda\mu$-calculus, which bases its type system on a natural deduction system for classical logic, terms in $\overline{\lambda}\mu\widetilde{\mu}$ represent derivations in a *sequent calculus* proof system and reduction reflects the process of cut-elimination.

In this paper we recount our experience in extending the language $\overline{\lambda}\mu\widetilde{\mu}$ in an untyped language which we propose to call Gemini (see [21] for similar attempt) an then deriving a type system for this extended language which characterizes the strongly normalizing terms. We are naturally led to enrich the Curien-Herbelin system of simple types by introducing intersection types. But it turns out [9] that union types are also necessary in order to completely