Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

Applications of infinitary lambda calculus

Henk Barendregt [a,*], Jan Willem Klop [a,b]

[a] Faculty of Science, Radboud University, P.O. Box 9010, 6500GL Nijmegen, The Netherlands
[b] Department of Computer Science, VU University Amsterdam, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

A R T I C L E   I N F O

Article history:
Received 21 October 2007
Accepted 4 September 2008
Available online 24 January 2009

A B S T R A C T

We present an introduction to infinitary lambda calculus, highlighting its main properties. Subsequently we give three applications of infinitary lambda calculus. The first addresses the non-definability of Surjective Pairing, which was shown by the first author not to be definable in lambda calculus. We show how this result follows easily as an application of Berry's Sequentiality Theorem, which itself can be proved in the setting of infinitary lambda calculus. The second pertains to the notion of relative recursiveness of number-theoretic functions. The third application concerns an explanation of counterexamples to confluence of lambda calculus extended with non-left-linear reduction rules: Adding non-left-linear reduction rules such as $\delta xx \to x$ or the reduction rules for Surjective Pairing to the lambda calculus yields non-confluence, as proved by the second author. We discuss how an extension to the infinitary lambda calculus, where Böhm trees can be directly manipulated as infinite terms, yields a more simple and intuitive explanation of the correctness of these Church-Rosser counterexamples.

1. Introduction

The aim of this paper is to present some well-known results in λ-calculus from the point of view of infinitary λ-calculus, where terms may be infinitely deep and reduction sequences may be of transfinite length α, for a countable ordinal α. Infinitary λ-terms are already familiar in λ-calculus in the form of Böhm trees (BTs), but in the extended setting of infinitary λ-calculus (or λ^∞ for short) BTs are just a particular kind of infinite normal forms, and in this extended setting we can even apply a BT to another BT. In Section 2 we will give a somewhat more detailed exposition of λ^∞ with β-reduction, $\lambda^\infty\boldsymbol{\beta}$ for short. (We will not consider η-reduction in this paper.) First we will describe why in our view infinitary λ-calculus is of interest.

The first reason pertains to *semantics* of λ-calculus. By now it is classic that infinite λ-terms constitute a syntactic approach to the semantics of finite λ-terms with (e.g.) β-reduction, in various forms, in particular the semantics given by the three families of infinite λ-trees known as Böhm trees, Lévy-Longo trees, and Berarducci trees. Whereas the first family seems to be the most important, the second family is instrumental for a closer connection to the practice of functional programming using notions as lazy reduction and weak head normal form, see Abramsky and Ong [1], while the third family is a sophisticated tool for consistency studies as demonstrated in Berarducci and Intrigila [13].

The second reason concerns the *pragmatics* of computing with λ-terms. Some computations are most naturally presented as transfinite sequences, rather than as compressed sequences of length at most ordinal ω, even though this always can be done by dove-tailing. Below we give some illustrating examples.

* Corresponding author.
E-mail address: henk@cs.ru.nl (H. Barendregt).

0890-5401/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2008.09.003

The third reason is found in the feature of *expressivity*. Infinite λ-terms can be nonrecursive. This can be used to give a direct representation of notions that otherwise need some circumlocution for their definition: a recursion-theoretic oracle, used in the definition of relative computability, can be defined in various ways, but the representation as an infinite λ-term has an appealing directness, since the oracle can now directly be processed by a finite λ-term, standing for a finite program. Below, in Section 4, we will substantiate this.

The last reason, illustrated by Section 3 on Berry's Sequentiality Theorem (BST) and Section 5 on the failure of confluence in extensions of λ-calculus with non-left linear reduction rules, is theoretical *coherence and transparency*, including a better understanding of phenomena in finite (!) λ-calculus. The section on BST provides such a better understanding for the inherent sequentiality of finitary λ-calculus, with as corollaries some non-definability results treated there, among them the fundamental fact that (just like parallel-or), it is not possible to define Surjective Pairing in λ-calculus. We present a succinct and new proof of this non-definability fact. Finally, Section 5 contributes to a better understanding of the extension of λ-calculus with rules like $\delta xx \to x$, encoding a discriminator δ for syntactic equality (of its two arguments); such an extension $\lambda + \delta$ looses the confluence property, but the deeper reason is best understood via an excursion to the realm of infinite λ-terms.

Concluding this Introduction, let us point out once more that our paper has in part the character of a survey and introduction, albeit of modest scope. This entails that our primary concern is not to communicate new results on this subject. Yet there are some new elements. Next to some new proofs, such as for the undefinability of Surjective Pairing in (finitary and now also in infinitary) λ-calculus, and for the non-confluence of this same system viewed as a rewrite system, there are a few new results, notably the short solution of an open problem of Scott [37], and a theorem building on work of Kleene [30], capturing the notion of relative recursiveness directly in (infinitary) λ-calculus.

2. Preliminaries

In this section we will lay out various notions and notations, and some basic properties, of finitary as well as infinitary λ-calculus.

2.1. Lambda calculus and two extensions

We assume familiarity with ordinary untyped λ-calculus, see e.g. Barendregt [8]. In particular the following notations will be used. The notation follows common practise. Closed λ-terms are usually denoted by Roman capitals, but sometimes by Greek letters (upper or lower case). As often in mathematics and programming languages, there are sometimes innocent examples of overloading: for example ω is a λ-term, but also the first infinite ordinal, in which sense it is used in the notation M^ω, an infinite λ-term.

Notation 2.1. $M \equiv N$ stands for syntactic equality between the (possibly infinitary) terms M, N and $M = N$ for their convertibility (w.r.t. a notion of reduction clear from the context, usually β or an extension). We use the combinators (closed λ-terms) $\mathsf{I} \equiv \lambda x.x$, $\mathsf{K} \equiv \lambda xy.x$, $\mathsf{S} \equiv \lambda xyz.xz(yz)$, $\mathsf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$, $\mathsf{B} \equiv \lambda xyz.x(yz)$, $\Theta \equiv (\lambda xy.y(xxy))(\lambda xy.y(xxy))$. We also often use the combinators $\omega \equiv (\lambda x.xx)$, in some papers denoted by Δ, and $\Omega \equiv (\omega\omega)$.

The set of λ-terms is denoted by Λ, that of normal forms (under β-reduction) by Λ_{NF}. The set of closed λ-terms is denoted by $\Lambda^{\emptyset}$. For $M, N \in \Lambda$ the following notations are used. For pairing $[M, N] \equiv \lambda z.zMN$, with z a fresh variable; for applicative iteration $M^n N$ is defined recursively: $M^0 N \equiv N$; $M^{k+1}N \equiv M(M^k N)$. Using this notation, the Church numerals are $\mathbf{c}_n \equiv \lambda fx.f^n x$. For iterated arguments $MN^{\sim n}$ is also defined recursively: $MN^{\sim 0} \equiv M$; $MN^{\sim(k+1)} \equiv MN^{\sim k}N$.

Definition 2.2. (i) Extend the set of λ-terms Λ with a constant f, intended to represent an $f : \mathbb{N} \to \mathbb{N}$. The resulting set of terms will be denoted by $\Lambda(\mathsf{f})$.

(ii) On $\Lambda(\mathsf{f})$ one can extend $\boldsymbol{\beta}$-reduction with the notion of reduction f axiomatized by the contraction rule: $\mathsf{f}\mathbf{c}_n \to_{\mathsf{f}} \mathbf{c}_{f(n)}$.

Lemma 2.3. *The notions of reduction* f *and* $\boldsymbol{\beta}\mathsf{f}$ *are Church-Rosser.*

Proof. Similar to the proof of Mitschke's Theorem 15.3.3 in Barendregt [8]. Alternatively, observe that f and $\boldsymbol{\beta}\mathsf{f}$ constitute orthogonal higher-order rewriting systems (in the form of CRSs or HRSs) and use Theorem 11.6.19 in Terese [39]. □

Remember that every λ-term M is of one of the following forms:

$$M \equiv \lambda x_1 \ldots x_n.yM_1 \ldots M_m \text{ or } \lambda x_1 \ldots x_n.(\lambda y.P)QM_1 \ldots M_k.$$

In the first case M is said to be a *head normal form (hnf)*; in the second case M has the head-redex $(\lambda y.P)Q$.