Available online at www.sciencedirect.com
ScienceDirect
Information and Computation 205 (2007) 419–473
Information and Computation
www.elsevier.com/locate/ic

Resource operators for λ-calculus

Delia Kesner [a,*], Stéphane Lengrand [a,b]

[a] *PPS, CNRS and Université Paris 7, France*
[b] *School of Computer Science, University of St. Andrews, UK*

Received 12 September 2005; revised 22 June 2006
Available online 31 January 2007

Abstract

We present a simple term calculus with an explicit control of erasure and duplication of substitutions, enjoying a sound and complete correspondence with the intuitionistic fragment of Linear Logic's proof-nets. We show the operational behaviour of the calculus and some of its fundamental properties such as confluence, preservation of strong normalisation, strong normalisation of simply typed terms, step by step simulation of β-reduction and full composition.

1. Introduction

The Curry-Howard paradigm, according to which the terms/types/reduction of a term calculus respectively correspond to the proofs/propositions/normalisation of a logical system, has already shown its numerous merits in the computer science community. Such a correspondence gives a double reading of proofs as programs and programs as proofs, so that insight into one aspect helps the understanding of the other.

A typical example of the Curry-Howard correspondence is obtained by taking the simply typed λ-calculus [11] as term calculus and Natural Deduction for Intuitionistic Logic as logical system.

* Corresponding author. Fax: +33 1 44 27 86 54.
E-mail address: kesner@pps.jussieu.fr (D. Kesner).

0890-5401/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2006.08.008

Both formalisms can be decomposed in the following sense: on the one hand the evaluation rule of λ-calculus, known as β-reduction, can be decomposed into more elementary operations by implementing (higher-order) substitution as the interaction between (and the propagation of) erasure, duplication and linear substitution operators. On the other hand Linear Logic [24] decomposes the intuitionistic logical connectives into more elementary connectives, such as the linear implication and the exponentials, thus providing a more refined and controlled use of resources (formulae) than that of Intuitionistic Logic.

We show that there is a deep connection between these two elementary decompositions. In order to relate them, we bridge the conceptual gap between the term syntax formalism and that of proof-nets [24] that we use to denote proofs in Linear Logic. Visually convenient to manipulate, proof-nets retain from the structure of a proof the part that is logically relevant, thus giving geometric insight into proof transformations. However, they are quite cumbersome in proof formalisations, owing to a certain lack of proof techniques and corresponding proof assistants. On the other hand, term notation is more convenient to formalise and carry out detailed proofs of properties, and also when one wants to implement them via some proof-assistant [13,32].

Several works [15,16,1,58,21] have already explored the relation between these two approaches, but none of them has pushed the formalism far enough to obtain a computational counterpart to proof-nets that is sound and complete with respect to the underlying logical model.

In this paper, we present a calculus called λlxr with erasure, duplication and linear substitution operators, which can be seen as a λ-calculus with *explicit substitutions*. Its simply typed version can be considered, via the Curry-Howard paradigm, as a functional computational counterpart to the intuitionistic fragment of proof-nets. The major features of this calculus are

- Simple syntax and intuitive operational semantics via reduction rules and equations;
- Sound and complete correspondence with the proof-nets model, where the equations and reductions of terms have a natural correspondence with those of proof-nets;
- Full composition of explicit substitutions;
- Nice properties such as confluence, preservation of strong normalisation, strong normalisation for Curry-style simply typed terms, and step by step simulation of β-reduction.

1.1. Explicit control of resources and proof-nets

Much work on explicit substitutions has been done in the last 15 years, for example [3,7,10,37]. In particular, an unexpected result was given by Melliès [45] who has shown that there are β-strongly normalisable terms in λ-calculus that are not strongly normalisable when evaluated by the reduction rules of an explicit version of the λ-calculus, such as for example $\lambda\sigma$ [3] or $\lambda\sigma_{\Uparrow}$ [30]. In other words, $\lambda\sigma$ and $\lambda\sigma_{\Uparrow}$ do not enjoy the property known as preservation of strong normalisation (PSN) [7].

This phenomenon shows a flaw in the design of these calculi with explicit substitutions in that they are supposed to implement their underlying calculus without losing its good properties such as strong normalisation of simply typed terms. However, there are many ways to avoid Melliès' counter-example in order to recover the PSN property. One of them is to simply forbid the substitution operators to cross lambda-abstractions [44,22]; another consists of avoiding composition of substitutions [7]; another one imposes a simple strategy on the calculus with explicit substitutions to mimic exactly the calculus without explicit substitutions [25]. The first solution leads to *weak*