Available online at www.sciencedirect.com
SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

ELSEVIER Electronic Notes in Theoretical Computer Science 136 (2005) 103–116
www.elsevier.com/locate/entcs

Lazy Strong Normalization

Luca Paolini[1,2]

Dipartimento di Informatica
Università di Torino (ITALIA)

Elaine Pimentel[1,2]

Departamento de Matemática
Universidade Federal de Minas Gerais (BRASIL)
Dipartimento di Informatica
Università di Torino (ITALIA)

Simona Ronchi Della Rocca[1,2]

Dipartimento di Informatica
Università di Torino (ITALIA)

Abstract

Among all the reduction strategies for the untyped λ-calculus, the so called *lazy β-evaluation* is of particular interest due to its large applicability to functional programming languages (e.g. Haskell [3]). This strategy reduces only redexes not inside a lambda abstraction.
The lazy strongly β- normalizing terms are the λ-terms that don't have infinite lazy β-reduction sequences.
This paper presents a logical characterization of lazy strongly β-normalizing terms using intersection types. This characterization, besides being interesting by itself, allows an interesting connection between call-by-name and call-by-value λ-calculus.
In fact, it turns out that the class of lazy strongly β-normalizing terms coincides with that of call-by-value potentially valuable terms. This last class is of particular interest since it is a key notion for characterizing solvability in the call-by-value setting.

Keywords: λ-calculus, lazy evaluation, lazy strong normalization

[1] Email: paolini@di.unito.it, pimentel@di.unito.it, ronchi@di.unito.it

[2] Paper partially supported by IST-2001-33477 DART Project, MIUR-Cofin'02 PROTOCOLLO Project. Pimentel is also supported by CNPq.

doi:10.1016/j.entcs.2005.06.013

1 Introduction

An evaluation is called *lazy* if the body of a function is evaluated only when an argument is supplied. In the λ-calculus setting, this kind of evaluation is modelled by a reduction strategy that does not reduce β-redexes occurring under the scope of a λ-abstraction. Lazy evaluation has been introduced by Plotkin [6] in order to capture into λ-calculus the conceptual difference between the notion of evaluation and that one of code optimization. [3]

The notion of strong β-normalization can be extended to the lazy case in a natural way (see [8]). Namely: a lazy β-redex is a β-redex not occurring under the scope of a λ-abstraction, and a term is in lazy β-normal form if and only if it has no occurrences of lazy β-redexes. So a term is lazy strongly β-normalizing if and only if it has lazy β-normal form and there are not infinite lazy β-reduction sequences starting from it.

In this paper we give a complete characterization of the class of lazy strongly β-normalizing terms in a logical way, using a suitable intersection type assignment system.

This characterization, besides being interesting by itself, allows an interesting connection between call-by-name and call-by-value λ-calculus. Let us remember that the classical λ-calculus is a model for the call-by-name evaluation, while the call-by-value evaluation can be modelled by a variant of λ-calculus, the $\lambda\beta_v$-calculus, introduced in [6]. The $\lambda\beta_v$-calculus is obtained from the λ-calculus by restricting the β-rule to the case where the argument is a value, i.e., it is either a variable or a λ-abstraction. The fact that all the λ-abstractions are values, independently from their bodies, implies that the natural evaluation for such a calculus is a lazy one. Some syntactical properties of the $\lambda\beta_v$-calculus have been studied in [5], where the notion of solvability has been adapted to this calculus, and the set of solvable terms has been completely characterized, in a logical way.

In particular, in order to give such a characterization, an intermediate class of terms has been introduced: the potentially valuable terms. A term M is potentially valuable if and only if there is a substitution $\mathbf{s}$, replacing free variables by closed values, such that $\mathbf{s}(M)$ reduces to a value. The importance of such a class becomes clearer when we note that, in the $\lambda\beta_v$-calculus, the restriction to the β-rule imposes that every term (or subterm), in order to be manipulated, must be first transformed into a value. The potentially valuable terms have been completely characterized in a logical way in [5], and it has been proved that the call-by-value solvable terms form a proper subclass of

[3] This must not be confused with the notion of lazy evaluation used in functional programming corresponding to a *call-by-need* evaluation strategy.