Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 249 (2009) 169–192

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

A Context-based Approach to Proving Termination of Evaluation

Małgorzata Biernacka [1] Dariusz Biernacki [2]

Institute of Computer Science
University of Wrocław
Wrocław, Poland

Abstract

We present a context-based approach to proving termination of evaluation in reduction semantics (i.e., a form of operational semantics with explicit representation of reduction contexts), using Tait-style reducibility predicates defined on both terms and contexts. We consider the simply typed lambda calculus as well as its extension with abortive control operators for first-class continuations under the call-by-value and the call-by-name evaluation strategies. For each of the proofs we present its computational content that takes the form of an evaluator in continuation-passing style and is an instance of normalization by evaluation.

Keywords: reduction semantics, evaluation context, weak head normalization, control operators, normalization by evaluation

1 Introduction

In the term-rewriting setting, a typical presentation of the lambda calculus as a prototypical programming language relies on the grammar of terms and a reduction relation defined on these terms. Felleisen et al. have introduced the notion of reduction/evaluation contexts [15–17] that proved useful in expressing various reduction strategies concisely, building on the notion of context as a term with a hole [2]. Felleisen's contexts represent "the surrounding term" of the current subterm, or "the rest of the computation", and they directly correspond to continuations: the latter can be seen as functional representations of contexts. More precisely, Danvy observed that reduction contexts arise as defunctionalized continuations of a one-step reduction function whereas evaluation contexts arise as defunctionalized continuations of an evaluation function (i.e., big-step) [11, 12]. Since these defunctionalized representations of continuations are in both cases the same, the terms "evaluation

1 Email: mabi@ii.uni.wroc.pl

2 Email: dabi@ii.uni.wroc.pl

doi:10.1016/j.entcs.2009.07.090

context" and "reduction context" are usually used interchangeably, and we will adhere to this practice in the remainder of this article.

Because of their close relation to continuations, the benefits of using contexts can be seen perhaps most prominently in languages with control operators, i.e., syntactic constructs that manipulate the current continuation/context [15]. Moreover, as shown by Wright and Felleisen [27], context-based reduction semantics of a programming language provide a convenient formalism for expressing and proving type soundness properties.

In this article we present yet another application of contexts: we give novel proofs of termination of evaluation in the simply typed lambda calculus under the call-by-value and call-by-name reduction strategies where reduction contexts play a major role. Subsequently we extend the simply typed lambda calculus with common abortive control operators: *callcc*, *abort* and Felleisen's $\mathcal{C}$ and we use the same approach as for the pure lambda calculus to prove termination for the extended language, using its standard context-based reduction semantics.

The method of proof we apply in this work—using context-based variant of Tait-style reducibility predicates [25]—is a modification of the method considered in a previous work of Biernacka et al. that used "direct-style" reducibility predicates [9]. In effect, we obtain direct, simple proofs of termination that take advantage of the context-based formulation of the reduction semantics. In contrast, many of the existing proofs of normalization properties for typed lambda calculi with control operators are indirect and they use a translation to another language already known to be normalizable [1, 18, 24]. This line of work on proof-theoretic properties of typed control operators was originated by Griffin who gave a type assignment to Felleisen's $\mathcal{C}$ operator, *abort* and *callcc*, and who proved termination of evaluation for his language using a translation to the simply typed lambda calculus akin to Plotkin's colon translation [18].

On the other hand, the method of proving normalization using Tait-style reducibility predicates has been applied to the pure lambda calculus, both for weak and strong normalization [5,25,26] as well as for weak head normalization under call by name (essentially due to Martin-Löf) and call by value (due to Hoffmann) [9]. An extension to control operators has been considered by Parigot who modified Girard's reducibility candidates to prove strong normalization for his second-order $\lambda\mu$-calculus corresponding to classical natural deduction [21]. Berger and Schwichtenberg identified the computational content of their constructive proof of strong normalization that uses the reducibility method to be an instance of normalization by evaluation, and subsequently this observation has been applied to proofs of weak head normalization by Coquand and Dybjer for combinatory logic [10] and by Biernacka et al. for the lambda calculus [9]. Some of the proofs have been formalized in proof assistants and normalizers have been extracted from them in the form of functional programs [4,6]. Not surprisingly, the computational content of our proofs are instances of normalization by evaluation; the extracted programs are evaluators in continuation-passing style, whose continuations arise by extraction from a context reducibility predicate. Thus the present article provides a logical confirmation of