Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Processing Letters 94 (2005) 217–224
Information Processing Letters
www.elsevier.com/locate/ipl

CPS transformation of beta-redexes

Olivier Danvy *, Lasse R. Nielsen

BRICS [1], *Department of Computer Science, University of Aarhus, IT-Parken, Aabogade 34, DK-8200 Aarhus N, Denmark*

Received 25 May 2004; received in revised form 2 February 2005

Available online 14 March 2005

Communicated by J. Chomicki

Abstract

The extra compaction of the most compacting CPS transformation in existence, which is due to Sabry and Felleisen, is generally attributed to (1) making continuations occur first in CPS terms and (2) classifying more redexes as administrative. We show that this extra compaction is actually independent of the relative positions of values and continuations and furthermore that it is solely due to a context-sensitive transformation of beta-redexes. We stage the more compact CPS transformation into a first-order uncurrying phase and a context-insensitive CPS transformation. We also define a context-insensitive CPS transformation that provides the extra compaction. This CPS transformation operates in one pass and is dependently typed.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Functional programming; Program derivation; Continuation-passing style (CPS); Plotkin; Fischer; One-pass CPS transformation; Two-level λ-calculus; Generalized reduction

1. Introduction

1.1. Continuation-passing style (CPS)

The meaning of a λ-term, in general, depends on its evaluation order. Evaluation-order independence was one of the motivations for continuations [19,25], and continuation-passing style was developed as an evaluation-order independent λ-encoding of λ-terms [7,18]. In this λ-encoding, each evaluation context is represented by a λ-abstraction, called a *continuation*, and each λ-abstraction is passed a continuation in addition to its usual argument. All intermediate results are sent to a continuation and thus all calls are tail-calls. This λ-encoding gives rise to a variety of continuation-passing styles, whose structure is a subject of study in itself [11,20,24].

* Corresponding author.
E-mail addresses: danvy@brics.dk (O. Danvy), lrn@brics.dk (L.R. Nielsen).
[1] Basic Research in Computer Science (http://www.brics.dk), funded by the Danish National Research Foundation.

0020-0190/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.ipl.2005.02.002

1.2. The CPS transformation

The format of CPS λ-terms was soon noticed to be of interest for the compiler writer [23], which in turn fostered interest in automating the transformation of λ-terms into CPS. Over the last twenty years, a wide body of CPS transformations has thus been developed for various purposes, e.g., to compile and to analyze programs, and to generate compilers [1,9,13,22,23,26].

The naïve λ-encoding into CPS, however, generates a quite impressive inflation of lambdas, most of which form *administrative redexes* that can be safely reduced. Administrative reductions yield CPS terms corresponding to what one could write by hand. It has therefore become a challenge to eliminate as many administrative redexes as possible, at CPS-transformation time. (Contracting other β-redexes would correspond to simplifying the source term, which falls out of the scope of the CPS transformation.)

1.3. Sabry and Felleisen's optimization

In their article "Reasoning about programs in continuation-passing style" [21], Sabry and Felleisen present a CPS transformation that yields more compact terms than existing CPS transformations. For example [21, footnote 6], CPS-transforming

$$\big((\lambda x.\lambda y.x)a\big)b,$$

where a and b are (free) variables, yields the term

$$\lambda k.\big(\big(\lambda x.((\lambda y.kx)b)\big)\,a\big)$$

whereas earlier transformations, such as Steele's [23] or Danvy and Filinski's [5], yield the more voluminous term

$$\lambda k.\big(\big(\lambda x.\lambda k_1.(k_1(\lambda y.\lambda k_2.k_2 x))\big)a(\lambda m.mbk)\big).$$

Sabry and Felleisen's optimization relies on using Fischer's CPS (where continuations occur first, as in $\lambda k.\lambda x.e$), whereas earlier transformations use Plotkin's CPS (where values occur first, as in $\lambda x.\lambda k.e$).[2]

1.4. This article

Section 2 reviews administrative reductions in the CPS transformation and characterizes Sabry and Felleisen's optimization, independently of the relative positions of values and continuations in CPS terms (i.e., both for Fischer's and Plotkin's CPS). Section 3 constructs a similarly compact CPS transformation by composing an uncurrying phase and an ordinary CPS transformation. Section 4 integrates the optimization in a context-insensitive, one-pass CPS transformation. Section 5 concludes.

2. Administrative reductions in the CPS transformation

2.1. Context-insensitive administrative reductions

Appel, Danvy and Filinski, and Wand each independently developed a "one-pass" CPS transformation for call by value [1,5,26]. This CPS transformation relies on a context-free characterization of administrative reductions, i.e., a characterization that is independent of any source term. This one-pass transformation, shown below for Plotkin's

[2] As traditional, the reference to Fischer is a little bit stretched since Fischer's domain of discourse was uncurried Lisp functions [7]. But we also follow the tradition here.

Download English Version:

https://daneshyari.com/en/article/10331364

Download Persian Version:

https://daneshyari.com/article/10331364

Daneshyari.com