Basic Research in Computer Science

BRICS RS-02-2   L. R. Nielsen: A Simple Correctness Proof of the Direct-Style Transformation

A Simple Correctness Proof of the Direct-Style Transformation

Lasse R. Nielsen

BRICS Report Series RS-02-2
ISSN 0909-0878 January 2002

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/02/2/`

A simple proof of the direct-style transformation

Lasse R. Nielsen
BRICS *
Department of Computer Science,
University of Aarhus †

January 2002

Abstract

We build on Danvy and Nielsen's first-order program transformation into continuation-passing style (CPS) to present a new correctness proof of the converse transformation, i.e., a one-pass transformation from CPS back to direct style. Previously published proofs were based on CPS transformations that were either higher-order, non-compositional, or operating in two passes, and were correspondingly complicated to reason about. In contrast, this work is based on a CPS transformation that is first-order, compositional, and that operates in one pass. Therefore the proof simply proceeds by structural induction on syntax.

Keywords: compositionality, CPS-transformation, direct-style transformation, correctness proof.

*Basic Research in Computer Science (`www.brics.dk`), funded by the Danish National Research Foundation.
†Ny Munkegade, Building 540, DK-8000 Aarhus C, Denmark.
E-mail: `lrn@brics.dk`

Contents

List of Figures

1 Introduction

1.1 The continuation-passing-style transformation

The CPS transformation on the λ-calculus maps direct-style expressions into semantically equivalent CPS expressions. Reynolds used it to map a functional program into an evaluation-order-independent form [9], and Plotkin later formalized it and proven it to be semantics preserving [8].

The CPS programs generated by Plotkin's CPS transformation contains so-called *administrative* redexes. Steele added a second pass to the transformation to reduce these redexes, generating equivalent but more compact CPS expressions [11]. This two-pass CPS transformation inspired several researchers to write *one-pass* CPS transformations that directly generate administratively reduced CPS expressions, making CPS transformation more practically useful. Appel, Danvy and Filinski, and Wand independently discovered a higher-order one-pass CPS transformation [1, 4, 12] while Sabry and Felleisen constructed a non-compositional one-pass CPS transformation based on syntactic theory [10]. Recently, however, Danvy and Nielsen presented a one-pass CPS transformation that is both first-order and compositional [6].

1.2 The direct-style transformation

The direct-style transformation is the inverse of the CPS transformation, and it maps CPS expressions back into direct-style expressions.

Danvy introduced the direct style transformation [2], and it was proven to preserve semantics by Lawall [7] and by Danvy, Dzafic, and Pfenning [3]. These proofs are based on the higher-order CPS transformation, and as such they require reasoning about higher-order functions, e.g., using logical relations.

This paper gives a simpler proof using only structural induction.

1.3 Overview

The rest of this paper is structured as follows: Section 2 presents the notation and definitions, Section 3 proves that the direct-style transformation preserves meaning by showing that it is a left inverse to the CPS transformation, and Section 4 concludes.

2 Definitions

A syntax for the λ-calculus is shown in Figure 1.

$$
\begin{array}{rcll}
p & ::= & e & p \in \text{DP}\textsc{rog} \\
e & ::= & t \mid s & e \in \text{DE}\textsc{xpr} \\
t & ::= & x \mid \lambda x.e & t \in \text{DT}\textsc{riv}\text{, trivial terms, i.e., values} \\
s & ::= & e\ e & s \in \text{DC}\textsc{omp}\text{, serious terms, i.e., computations} \\
 & & & x \in \text{I}\textsc{de}\text{, a set of identifiers}
\end{array}
$$

Figure 1: Syntax of the λ-calculus in direct style

A grammar of λ-expressions in continuation-passing style is shown in Figure 2.

$$
\begin{array}{rcll}
p & ::= & \lambda k.e & p \in \text{CP}\textsc{rog} \\
e & ::= & r\ c \mid c\ t & e \in \text{CE}\textsc{xpr} \\
t & ::= & v \mid x \mid \lambda x.r & t \in \text{CT}\textsc{riv} \\
r & ::= & \lambda k.e \mid t\ t & r \in \text{CR}\textsc{oot} \\
c & ::= & k \mid \lambda v.e & c \in \text{CC}\textsc{ont}
\end{array}
$$

where $x \in \text{I}\textsc{de}$, $k \in \text{CI}\textsc{de}$, and $v \in \text{VI}\textsc{de}$, disjoint sets of identifiers.

Figure 2: Syntax of the λ-calculus in continuation-passing style

The call-by-value (CBV) CPS transformation is shown in Figure 3.

Being "fresh wrt. e" and "fresh wrt. c" means that we pick, deterministically, an element of CIDE, or two different elements of VIDE, that do not occur freely in e, respectively in c.

A typing argument shows that the CPS transformation actually generates only programs in continuation-passing style.

The direct-style transformation corresponding to the CPS transformation uses a stack of expressions to keep track of the intermediate results. This stack, represented by σ, is either the empty stack ($\bullet$) or a stack with something on top ($x :: \sigma'$), and the set of stacks of direct-style expressions is represented by [DEXPR].

The definition of the Direct-Style transformation is shown in Figure 4. The direct-style transformation is not total on the set of CPS programs. In

$$
\begin{array}{rcl}
\mathcal{C} & : & \text{DP}\textsc{rog} \rightarrow \text{CP}\textsc{rog} \\
\mathcal{C}[\![e]\!] & = & \lambda k.\, \mathcal{C}^{\text{DE}\textsc{xpr}}[\![e]\!]\; k \\[1ex]
\mathcal{C}^{\text{DE}\textsc{xpr}} & : & \text{DE}\textsc{xpr} \times \text{CI}\textsc{de} \rightarrow \text{CE}\textsc{xpr} \\
\mathcal{C}^{\text{DE}\textsc{xpr}}[\![t]\!]\; k & = & k\; \mathcal{C}^{\text{DT}\textsc{riv}}[\![t]\!] \\
\mathcal{C}^{\text{DE}\textsc{xpr}}[\![s]\!]\; k & = & \mathcal{C}^{\text{DC}\textsc{omp}}[\![s]\!]\; k \\[1ex]
\mathcal{C}^{\text{DT}\textsc{riv}} & : & \text{DT}\textsc{riv} \rightarrow \text{CT}\textsc{riv} \\
\mathcal{C}^{\text{DT}\textsc{riv}}[\![x]\!] & = & x \\
\mathcal{C}^{\text{DT}\textsc{riv}}[\![\lambda x.e]\!] & = & \lambda x.\lambda k.\, \mathcal{C}^{\text{DE}\textsc{xpr}}[\![e]\!]\; k \quad \text{where } k \text{ fresh wrt. } e \\[1ex]
\mathcal{C}^{\text{DC}\textsc{omp}} & : & \text{DC}\textsc{omp} \times \text{CC}\textsc{ont} \rightarrow \text{CE}\textsc{xpr} \\
\mathcal{C}^{\text{DC}\textsc{omp}}[\![t_1\; t_2]\!]\; c & = & \mathcal{C}^{\text{DT}\textsc{riv}}[\![t_1]\!]\; \mathcal{C}^{\text{DT}\textsc{riv}}[\![t_2]\!]\; c \\
\mathcal{C}^{\text{DC}\textsc{omp}}[\![s_1\; t_2]\!]\; c & = & \mathcal{C}^{\text{DC}\textsc{omp}}[\![s_1]\!]\; (\lambda v_1.v_1\; \mathcal{C}^{\text{DT}\textsc{riv}}[\![t_2]\!]\; c) \\
\mathcal{C}^{\text{DC}\textsc{omp}}[\![t_1\; s_2]\!]\; c & = & \mathcal{C}^{\text{DC}\textsc{omp}}[\![s_2]\!]\; (\lambda v_2.\mathcal{C}^{\text{DT}\textsc{riv}}[\![t_1]\!]\; v_2\; c) \\
\mathcal{C}^{\text{DC}\textsc{omp}}[\![s_1\; s_2]\!]\; c & = & \mathcal{C}^{\text{DC}\textsc{omp}}[\![s_1]\!]\; (\lambda v_1.\mathcal{C}^{\text{DC}\textsc{omp}}[\![s_2]\!]\; (\lambda v_2.v_1\; v_2\; c)) \\
 & & \text{where } v_1 \text{ and } v_2 \text{ are distinct and fresh wrt. } c
\end{array}
$$

Figure 3: Call-by-value CPS transformation

$$
\begin{array}{rcl}
\mathcal{D} & : & \text{CP}\textsc{rog} \rightarrow \text{DP}\textsc{rog} \\
\mathcal{D}[\![\lambda k.e]\!] & = & \mathcal{D}^{\text{CE}\textsc{xpr}}[\![e]\!]\; \bullet \\[1ex]
\mathcal{D}^{\text{CE}\textsc{xpr}} & : & \text{CE}\textsc{xpr} \times [\text{DE}\textsc{xpr}] \rightarrow \text{DE}\textsc{xpr} \\
\mathcal{D}^{\text{CE}\textsc{xpr}}[\![r\; c]\!]\; \sigma & = & \mathcal{D}^{\text{CC}\textsc{ont}}[\![c]\!]\; (\mathcal{D}^{\text{CR}\textsc{oot}}[\![r]\!]\; \sigma) \\
\mathcal{D}^{\text{CE}\textsc{xpr}}[\![c\; t]\!]\; \sigma & = & \mathcal{D}^{\text{CC}\textsc{ont}}[\![c]\!]\; (\mathcal{D}^{\text{CT}\textsc{riv}}[\![t]\!]\; \sigma) \\[1ex]
\mathcal{D}^{\text{CT}\textsc{riv}} & : & \text{CT}\textsc{riv} \times [\text{DE}\textsc{xpr}] \rightarrow \text{DE}\textsc{xpr} \times [\text{DE}\textsc{xpr}] \\
\mathcal{D}^{\text{CT}\textsc{riv}}[\![v]\!]\; (e :: \sigma) & = & (e, \sigma) \\
\mathcal{D}^{\text{CT}\textsc{riv}}[\![x]\!]\; \sigma & = & (x, \sigma) \\
\mathcal{D}^{\text{CT}\textsc{riv}}[\![\lambda x.r]\!]\; \sigma & = & (\lambda x.e, \sigma) \quad \text{where } (e, \bullet) = \mathcal{D}^{\text{CR}\textsc{oot}}[\![r]\!]\; \bullet \\[1ex]
\mathcal{D}^{\text{CR}\textsc{oot}} & : & \text{CR}\textsc{oot} \times [\text{DE}\textsc{xpr}] \rightarrow \text{DE}\textsc{xpr} \times [\text{DE}\textsc{xpr}] \\
\mathcal{D}^{\text{CR}\textsc{oot}}[\![\lambda k.e]\!]\; \sigma & = & (\mathcal{D}^{\text{CE}\textsc{xpr}}[\![e]\!]\; \bullet, \sigma) \\
\mathcal{D}^{\text{CR}\textsc{oot}}[\![t_1\; t_2]\!]\; \sigma & = & (e_1\; e_2, \sigma'') \quad \text{where} \quad (e_2,\sigma') \;=\; \mathcal{D}^{\text{CT}\textsc{riv}}[\![t_2]\!]\; \sigma \\
 & & \qquad\qquad\qquad\qquad (e_1,\sigma'') \;=\; \mathcal{D}^{\text{CC}\textsc{ont}}[\![t_1]\!]\; \sigma' \\
\mathcal{D}^{\text{CC}\textsc{ont}} & : & \text{CC}\textsc{ont} \times (\text{DE}\textsc{xpr} \times [\text{DE}\textsc{xpr}]) \rightarrow \text{DE}\textsc{xpr} \\
\mathcal{D}^{\text{CC}\textsc{ont}}[\![k]\!]\; (e, \sigma) & = & e \\
\mathcal{D}^{\text{CC}\textsc{ont}}[\![\lambda v.e]\!]\; (e', \sigma) & = & \mathcal{D}^{\text{CE}\textsc{xpr}}[\![e]\!]\; (e' :: \sigma)
\end{array}
$$

Figure 4: The direct-style transformation

the next section we show that it is total on the image of the CPS transformation, and we only consider the transformation on this set.

3 Correctness

We prove that the direct-style transformation is correct and non-trivial. By correct we mean that it preserves meaning. By non-trivial we mean that the direct-style expressions generated by the transformation are not only a limited subset of λ-expressions. Since CPS expressions are a subset of direct-style expressions, the identity function could be considered a trivial direct-style transformation.

The proof shows that the direct-style transformation is a left-inverse to the CPS transformation. Since the CPS transformation preserves meaning and is defined on all terms, the direct-style transformation must also preserve meaning and be non-trivial. The CPS transformation is injective but not surjective, so when restricted to its image, it is a bijection, and the left inverse also becomes a right inverse.

Lemma 1 (Left Inverse) *The $\mathcal{D}$ function is a left inverse to the $\mathcal{C}$ function.*

$$\forall p \in \text{DP}\textsc{rog}.\mathcal{D}[\![\mathcal{C}[\![p]\!]]\!] = p$$

Proof:

The proof is by structural induction on the program. We show the following three properties by mutual structural induction.

1. If $e : \text{DE}\textsc{xpr}$ is an expression and $k : \text{CI}\textsc{de}$ a continuation identifier then for any σ
$$\mathcal{D}^{\text{CE}\textsc{xpr}}[\![\mathcal{C}^{\text{DE}\textsc{xpr}}[\![e]\!]\ k]\!]\ \sigma = e$$

2. If $t : \text{DT}\textsc{riv}$ is a value then for any σ
$$\mathcal{D}^{\text{CT}\textsc{riv}}[\![\mathcal{C}^{\text{DT}\textsc{riv}}[\![t]\!]]\!]\ \sigma = (t, \sigma)$$

3. If $s : \text{DC}\textsc{omp}$ is a computation and $c : \text{CC}\textsc{ont}$ a continuation then for any σ
$$\mathcal{D}^{\text{CE}\textsc{xpr}}[\![\mathcal{C}^{\text{DC}\textsc{omp}}[\![s]\!]\ c]\!]\ \sigma = \mathcal{D}^{\text{CC}\textsc{ont}}[\![c]\!]\ (s, \sigma)$$

Property 1: There are two cases, one for each production in the grammar.

Case $e = t$**:**

$$\begin{array}{rl} & \mathcal{D}^{\text{CExpr}}[\![\mathcal{C}^{\text{DExpr}}[\![t]\!]\ k]\!]\ \sigma \\ = & \mathcal{D}^{\text{CExpr}}[\![k\ \mathcal{C}^{\text{DTriv}}[\![t]\!]]\!]\ \sigma \\ = & \mathcal{D}^{\text{CCont}}[\![k]\!]\ (\mathcal{D}^{\text{CTriv}}[\![\mathcal{C}^{\text{DTriv}}[\![t]\!]]\!]\ \sigma) \\ = & \mathcal{D}^{\text{CCont}}[\![k]\!]\ (t, \sigma) \qquad \text{(by I.H.)} \\ = & t \end{array}$$

Case $e = s$**:**

$$\begin{array}{rcl} \mathcal{D}^{\text{CExpr}}[\![\mathcal{C}^{\text{DExpr}}[\![s]\!]\ k]\!]\ \sigma & = & \mathcal{D}^{\text{CExpr}}[\![\mathcal{C}^{\text{DComp}}[\![s]\!]\ k]\!]\ \sigma \\ & = & \mathcal{D}^{\text{CCont}}[\![k]\!]\ (s, \sigma) \\ & = & s \end{array}$$

Property 2: There are two cases, one for each production in the grammar.

Case $t = x$**:**

$$\mathcal{D}^{\text{CTriv}}[\![\mathcal{C}^{\text{DTriv}}[\![x]\!]]\!]\ \sigma = \mathcal{D}^{\text{CTriv}}[\![x]\!]\ \sigma = (x, \sigma)$$

Case $t = \lambda x.e$**:**

$$\begin{array}{rl} & \mathcal{D}^{\text{CTriv}}[\![\mathcal{C}^{\text{DTriv}}[\![\lambda x.e]\!]]\!]\ \sigma \\ = & \mathcal{D}^{\text{CTriv}}[\![\lambda x.\lambda k.\,\mathcal{C}^{\text{DExpr}}[\![e]\!]\ k]\!]\ \sigma \\ = & (\lambda x.e', \sigma') \\ & \text{where } (e', \sigma') = \mathcal{D}^{\text{CRoot}}[\![\lambda k.\,\mathcal{C}^{\text{DExpr}}[\![e]\!]\ k]\!]\ \sigma \\ = & (\lambda x.e', \sigma') \\ & \text{where } (e', \sigma') = (\mathcal{D}^{\text{CExpr}}[\![\mathcal{C}^{\text{DExpr}}[\![e]\!]\ k]\!]\ \bullet\ , \sigma) \\ = & (\lambda x.e', \sigma') \\ & \text{where } (e', \sigma') = (e, \sigma) \qquad \text{(by I.H.)} \\ = & (\lambda x.e, \sigma) \end{array}$$

Property 3: There are four cases, one for each case of the $\mathcal{C}^{\text{DComp}}$ function.

Case $s = t_1\ t_2$**:**

$$\begin{array}{rl} & \mathcal{D}^{\text{CExpr}}[\![\mathcal{C}^{\text{DComp}}[\![t_1\ t_2]\!]\ c]\!]\ \sigma \\ = & \mathcal{D}^{\text{CExpr}}[\![\mathcal{C}^{\text{DTriv}}[\![t_1]\!]\ \mathcal{C}^{\text{DTriv}}[\![t_2]\!]\ c]\!]\ \sigma \\ = & \mathcal{D}^{\text{CCont}}[\![c]\!]\ (\mathcal{D}^{\text{CRoot}}[\![\mathcal{C}^{\text{DTriv}}[\![t_1]\!]\ \mathcal{C}^{\text{DTriv}}[\![t_2]\!]]\!]\ \sigma) \\ = & \mathcal{D}^{\text{CCont}}[\![c]\!]\ (e_1\ e_2, \sigma'') \\ & \text{where } (e_2, \sigma') = \mathcal{D}^{\text{CTriv}}[\![\mathcal{C}^{\text{DTriv}}[\![t_2]\!]]\!]\ \sigma \\ & \qquad\ \ (e_1, \sigma'') = \mathcal{D}^{\text{CTriv}}[\![\mathcal{C}^{\text{DTriv}}[\![t_1]\!]]\!]\ \sigma \\ = & \mathcal{D}^{\text{CCont}}[\![c]\!]\ (t_1\ t_2, \sigma) \end{array}$$

Case $s = s_1\ t_2$**:**

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathcal{D}^{\text{CExpr}}[\![\mathcal{C}^{\text{DComp}}[\![s_1\ t_2]\!]\ c]\!]\ \sigma} \\
= & \mathcal{D}^{\text{CExpr}}[\![\mathcal{C}^{\text{DComp}}[\![s_1]\!]\ (\lambda v_1.v_1\ \mathcal{C}^{\text{DTriv}}[\![t_2]\!]\ c)]\!]\ \sigma \\
= & \mathcal{D}^{\text{CCont}}[\![\lambda v_1.v_1\ \mathcal{C}^{\text{DTriv}}[\![t_2]\!]\ c]\!]\ (s_1, \sigma) \qquad \text{(by I.H.)} \\
= & \mathcal{D}^{\text{CExpr}}[\![v_1\ \mathcal{C}^{\text{DTriv}}[\![t_2]\!]\ c]\!]\ s_1 :: \sigma \\
= & \mathcal{D}^{\text{CCont}}[\![c]\!]\ (e_1\ e_2, \sigma'') \\
 & \text{where } (e_2, \sigma') = \mathcal{D}^{\text{CTriv}}[\![\mathcal{C}^{\text{DTriv}}[\![t_2]\!]]\!]\ s_1 :: \sigma \\
 & \qquad\quad (e_1, \sigma'') = \mathcal{D}^{\text{CTriv}}[\![v_1]\!]\ \sigma' \\
= & \mathcal{D}^{\text{CCont}}[\![c]\!]\ (e_1\ e_2, \sigma'') \\
 & \text{where } (e_2, \sigma') = (t_2, s_1 :: \sigma) \\
 & \qquad\quad (e_1, \sigma'') = \mathcal{D}^{\text{CTriv}}[\![v_1]\!]\ \sigma' \\
= & \mathcal{D}^{\text{CCont}}[\![c]\!]\ (e_1\ e_2, \sigma'') \\
 & \text{where } (e_2, \sigma') = (t_2, s_1 :: \sigma) \\
 & \qquad\quad (e_1, \sigma'') = (s_1, \sigma) \\
= & \mathcal{D}^{\text{CCont}}[\![c]\!]\ (s_1\ t_2, \sigma)
\end{array}
$$

Case $s = t_1\ s_2$**:**

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathcal{D}^{\text{CExpr}}[\![\mathcal{C}^{\text{DComp}}[\![t_1\ s_2]\!]\ c]\!]\ \sigma} \\
= & \mathcal{D}^{\text{CExpr}}[\![\mathcal{C}^{\text{DComp}}[\![s_2]\!]\ (\lambda v_2.\mathcal{C}^{\text{DTriv}}[\![t_1]\!]\ v_2\ c)]\!]\ \sigma \\
= & \mathcal{D}^{\text{CCont}}[\![\lambda v_2.\mathcal{C}^{\text{DTriv}}[\![t_1]\!]\ v_2\ c]\!]\ (s_2, \sigma) \qquad \text{(by I.H.)} \\
= & \mathcal{D}^{\text{CExpr}}[\![\mathcal{C}^{\text{DTriv}}[\![t_1]\!]\ v_2\ c]\!]\ (s_2 :: \sigma) \\
= & \mathcal{D}^{\text{CCont}}[\![c]\!]\ (e_1\ e_2, \sigma'') \\
 & \text{where } (e_2, \sigma') = \mathcal{D}^{\text{CTriv}}[\![v_2]\!]\ (s_2 :: \sigma) \\
 & \qquad\quad (e_1, \sigma'') = \mathcal{D}^{\text{CTriv}}[\![\mathcal{C}^{\text{DTriv}}[\![t_1]\!]]\!]\ \sigma' \\
= & \mathcal{D}^{\text{CCont}}[\![c]\!]\ (e_1\ e_2, \sigma'') \\
 & \text{where } (e_2, \sigma') = (s_2, \sigma) \\
 & \qquad\quad (e_1, \sigma'') = (t_1, \sigma') \\
= & \mathcal{D}^{\text{CCont}}[\![c]\!]\ (t_1\ s_2, \sigma)
\end{array}
$$

Case $s = s_1\ s_2$**:**

$$\begin{array}{ll}
& \mathcal{D}^{\textsc{CExpr}}[\![\mathcal{C}^{\textsc{DComp}}[\![s_1\ s_2]\!]\ c]\!]\ \sigma \\
= & \mathcal{D}^{\textsc{CExpr}}[\![\mathcal{C}^{\textsc{DComp}}[\![s_1]\!]\ (\lambda v_1.\mathcal{C}^{\textsc{DComp}}[\![s_2]\!]\ (\lambda v_2.v_1\ v_2\ c))]\!]\ \sigma \\
= & \mathcal{D}^{\textsc{CCont}}[\![\lambda v_1.\mathcal{C}^{\textsc{DComp}}[\![s_2]\!]\ (\lambda v_2.v_1\ v_2\ c)]\!]\ (s_1, \sigma) \qquad \text{(by I.H.)} \\
= & \mathcal{D}^{\textsc{CExpr}}[\![\mathcal{C}^{\textsc{DComp}}[\![s_2]\!]\ (\lambda v_2.v_1\ v_2\ c)]\!]\ (s_1 :: \sigma) \\
= & \mathcal{D}^{\textsc{CCont}}[\![\lambda v_2.v_1\ v_2\ c]\!]\ (s_2, s_1 :: \sigma) \qquad \text{(by I.H.)} \\
= & \mathcal{D}^{\textsc{CExpr}}[\![v_1\ v_2\ c]\!]\ (s_2 :: s_1 :: \sigma) \\
= & \mathcal{D}^{\textsc{CCont}}[\![c]\!]\ \mathcal{D}^{\textsc{CRoot}}[\![v_1\ v_2]\!]\ (s_2 :: s_1 :: \sigma) \\
= & \mathcal{D}^{\textsc{CCont}}[\![c]\!]\ (e_1\ e_2, \sigma'') \\
& \text{where } (e_2, \sigma') = \mathcal{D}^{\textsc{CTriv}}[\![v_2]\!]\ (s_2 :: s_1 :: \sigma) \\
& \qquad\quad (e_1, \sigma'') = \mathcal{D}^{\textsc{CTriv}}[\![v_1]\!]\ \sigma' \\
= & \mathcal{D}^{\textsc{CCont}}[\![c]\!]\ (e_1\ e_2, \sigma'') \\
& \text{where } (e_2, \sigma') = (s_2, s_1 :: \sigma) \\
& \qquad\quad (e_1, \sigma'') = (s_1, \sigma) \\
= & \mathcal{D}^{\textsc{CCont}}[\![c]\!]\ (s_1\ s_2, \sigma)
\end{array}$$

These cases shows that the properties hold for all direct-style expressions, so in particular if $e : \textsc{DProg}$

$$\mathcal{D}[\![\mathcal{C}[\![e]\!]]\!] = \mathcal{D}[\![\lambda k.\, \mathcal{C}^{\textsc{DExpr}}[\![e]\!]\ k]\!] = \mathcal{D}^{\textsc{CExpr}}[\![\mathcal{C}^{\textsc{DExpr}}[\![e]\!]\ k]\!]\ \bullet = e$$

QED

When restricting the CPS transformation to its image, i.e., forcing it to be surjective, a left inverse is also a right inverse.

Lemma 2 (Right Inverse) *The function $\mathcal{D}$ is the right inverse of $\mathcal{C}$ on $\mathcal{C}[\![\textsc{DProg}]\!]$, the image of* $\textsc{DProg}$ *under $\mathcal{C}$.*

$$\forall p \in \mathcal{C}[\![\textsc{DProg}]\!].\mathcal{C}[\![\mathcal{D}[\![p]\!]]\!] = p$$

Proof: Let $p \in \mathcal{C}[\![\textsc{DProg}]\!]$, i.e., there exists a $p' \in \textsc{DProg}$ such that $p = \mathcal{C}[\![p']\!]$. Then $(\mathcal{C}\circ\mathcal{D})(p) = (\mathcal{C}\circ\mathcal{D})(\mathcal{C}[\![p']\!]) = \mathcal{C}[\![(\mathcal{D}\circ\mathcal{C})(p')]\!]$. From Theorem 1 we know that $\mathcal{D}\circ\mathcal{C}$ is the identity on $\textsc{DProg}$, so $\mathcal{C}[\![(\mathcal{D}\circ\mathcal{C})(p')]\!] = \mathcal{C}[\![p']\!] = p$.

QED

With these lemmas showing the following connection

$$\begin{tikzcd} \textsc{DProg} \arrow[r, bend left, "\mathcal{C}"] & \mathcal{C}[\![\textsc{DProg}]\!] \arrow[l, bend left, "\mathcal{D}"] \end{tikzcd}$$

we can directly show correctness and non-triviality

Theorem 1 *The direct-style transformation is correct and non-trivial.*

Proof: Follows from the correctness of the CPS transformation and the previous lemmas. QED

4 Conclusion

We have presented a simpler proof of the correctness of the direct-style transformation than what has previously been published. The proof, like the previous ones, is based on a CPS transformation, since the choice of CPS transformation dictates the type of proof. Earlier proofs of the higher-order CPS transformation used logical relations [5], proofs of the non-compositional CPS-transformation used well-founded induction [10], and proofs of two-pass CPS transformations need to address both passes [10]. In contrast, a first-order, compositional, and one-pass CPS transformation allows a proof using only a single structural induction [6].

One can also show correctness of the direct-style transformation on larger sets than just the image of the CPS transformation. Proofs of such properties can also be derived from correctness of a CPS transformation, and using the first-order compositional CPS transformation also gives proofs using only structural induction.

References

[1] Andrew W. Appel. *Compiling with Continuations*. Cambridge University Press, New York, 1992.

[2] Olivier Danvy. Back to direct style. *Science of Computer Programming*, 22(3):183–195, 1994.

[3] Olivier Danvy, Belmina Dzafic, and Frank Pfenning. On proving syntactic properties of CPS programs. In *Third International Workshop on Higher-Order Operational Techniques in Semantics*, volume 26 of *Electronic Notes in Theoretical Computer Science*, pages 19–31, Paris, France, September 1999. Also available as the technical report BRICS RS-99-23.

[4] Olivier Danvy and Andrzej Filinski. Representing control, a study of the CPS transformation. *Mathematical Structures in Computer Science*, 2(4):361–391, 1992.

[5] Olivier Danvy and Lasse R. Nielsen. A higher-order colon translation. In Herbert Kuchen and Kazunori Ueda, editors, *Fifth International Symposium on Functional and Logic Programming*, number 2024 in Lecture Notes in Computer Science, pages 78–91, Tokyo, Japan, March 2001. Springer-Verlag. Extended version available as the technical report BRICS RS-00-33.

[6] Olivier Danvy and Lasse R. Nielsen. A first-order one-pass CPS transformation. In Mogens Nielsen and Uffe Engberg, editors, *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002*, number 2303 in Lecture Notes in Computer Science, pages 98–113, Grenoble, France, April 2002. Springer-Verlag. Extended version available as the technical report BRICS RS-01-49.

[7] Julia L. Lawall. *Continuation Introduction and Elimination in Higher-Order Programming Languages.* PhD thesis, Computer Science Department, Indiana University, Bloomington, Indiana, July 1994.

[8] Gordon D. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1:125–159, 1975.

[9] John C. Reynolds. Definitional interpreters for higher-order programming languages. *Higher-Order and Symbolic Computation*, 11(4):363–397, 1998. Reprinted from the proceedings of the 25th ACM National Conference (1972).

[10] Amr Sabry and Matthias Felleisen. Reasoning about programs in continuation-passing style. *Lisp and Symbolic Computation*, 6(3/4):289–360, 1993.

[11] Guy L. Steele Jr. Rabbit: A compiler for Scheme. Technical Report AI-TR-474, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1978.

[12] Mitchell Wand. Correctness of procedure representations in higher-order assembly language. In Stephen Brookes, Michael Main, Austin Melton, Michael Mislove, and David Schmidt, editors, *Mathematical Foundations of Programming Semantics*, number 598 in Lecture Notes in Computer Science, pages 294–311, Pittsburgh, Pennsylvania, March 1991. Springer-Verlag. 7th International Conference.

Recent BRICS Report Series Publications

RS-02-2 Lasse R. Nielsen. *A Simple Correctness Proof of the Direct-Style Transformation*. January 2002. 11 pp.

RS-02-1 Claus Brabrand, Anders Møller, and Michael I. Schwartzbach. *The `<bigwig>` Project*. January 2002. 36 pp. This revised report supersedes the earlier BRICS report RS-00-42.

RS-01-55 Daniel Damian and Olivier Danvy. *A Simple CPS Transformation of Control-Flow Information*. December 2001. 18 pp.

RS-01-54 Daniel Damian and Olivier Danvy. *Syntactic Accidents in Program Analysis: On the Impact of the CPS Transformation*. December 2001. 41 pp. To appear in the *Journal of Functional Programming*. This report supersedes the earlier BRICS report RS-00-15.

RS-01-53 Zoltán Ésik and Masami Ito. *Temporal Logic with Cyclic Counting and the Degree of Aperiodicity of Finite Automata*. December 2001. 31 pp.

RS-01-52 Jens Groth. *Extracting Witnesses from Proofs of Knowledge in the Random Oracle Model*. December 2001. 23 pp.

RS-01-51 Ulrich Kohlenbach. *On Weak Markov's Principle*. December 2001. 10 pp. Appears in Math. Logic Quaterly.

RS-01-50 Jiří Srba. *Note on the Tableau Technique for Commutative Transition Systems*. December 2001. 19 pp. Appears in Nielsen and Engberg, editors, *Foundations of Software Science and Computation Structures*, FoSSaCS '02 Proceedings, LNCS 2303, 2002, pages 387–401.

RS-01-49 Olivier Danvy and Lasse R. Nielsen. *A First-Order One-Pass CPS Transformation*. December 2001. 21 pp. Extended version of a paper appearing in Nielsen and Engberg, editors, *Foundations of Software Science and Computation Structures*, FoSSaCS '02 Proceedings, LNCS 2303, 2002, pages 98–113.

RS-01-48 Mogens Nielsen and Frank D. Valencia. *Temporal Concurrent Constraint Programming: Applications and Behavior*. December 2001. 36 pp. Appears in Brauer, Ehrig, Karhumäki and Salomaa, editors, *Formal and Natural Computing*, LNCS 2300, 2001, pages 298–321.