*JFP* **13** (5): 925–933, September 2003. © 2003 Cambridge University Press
DOI: 10.1017/S0956796803004702 Printed in the United Kingdom


# *CPS transformation of flow information, Part II: administrative reductions**

DANIEL DAMIAN

*LION Bioscience Ltd., Compass House, 80–82 Newmarket Road, Cambridge CB5 8DZ, UK*
(*e-mail:* Daniel.Damian@uk.lionbioscience.com)

OLIVIER DANVY

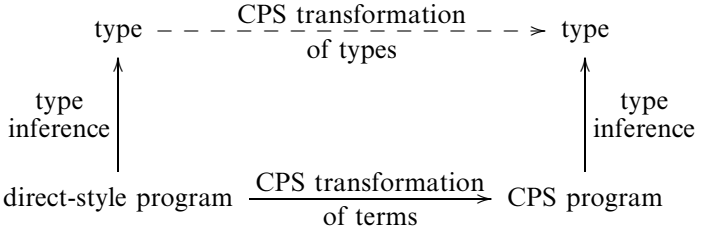
*BRICS, Department of Computer Science, University of Aarhus, Ny Munkegade, Building 540, DK-8000 Aarhus C, Denmark*
(*e-mail:* danvy@brics.dk)

**Abstract**

We characterize the impact of a linear $\beta$-reduction on the result of a control-flow analysis. (By 'a linear $\beta$-reduction' we mean the $\beta$-reduction of a linear $\lambda$-abstraction, i.e., of a $\lambda$-abstraction whose parameter occurs exactly once in its body.) As a corollary, we consider the administrative reductions of a Plotkin-style transformation into Continuation-Passing Style (CPS), and how they affect the result of a constraint-based control-flow analysis and, in particular, the least element in the space of solutions. We show that administrative reductions preserve the least solution. Preservation of least solutions solves a problem that was left open in Palsberg and Wand's article 'CPS Transformation of Flow Information.' Together, Palsberg and Wand's article and the present article show how to map in linear time the least solution of the flow constraints of a program into the least solution of the flow constraints of the CPS counterpart of this program, after administrative reductions. Furthermore, we show how to CPS transform control-flow information in one pass.

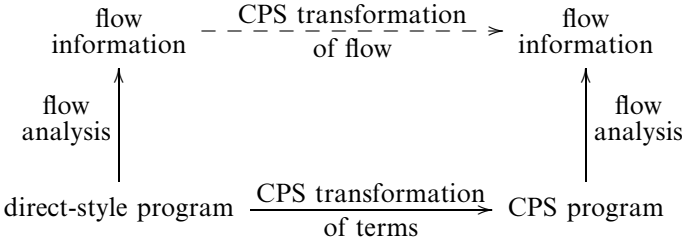
## 1 Background and introduction

Since their inception, over 30 years ago (Reynolds, 1993), continuations and the transformation into Continuation-Passing Style (CPS) have been the topic of much study, ranging from semantics and logic to implementations of sequential, concurrent, and distributed programming languages and systems. Fifteen years ago, Meyer and Wand (Meyer & Wand, 1985; Wand, 1985) noticed that the CPS transformation preserves types and they constructed a CPS transformation of types.

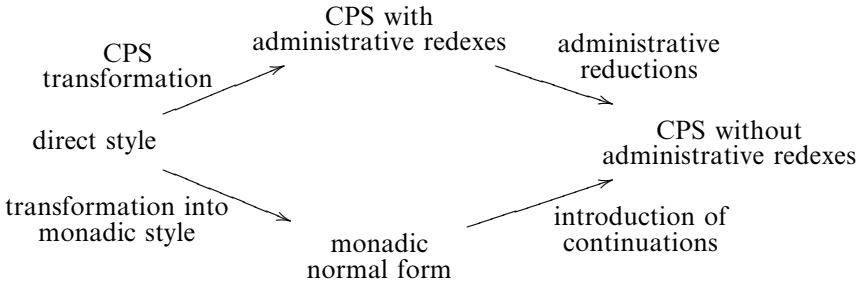



* This work was carried out while the first author was at BRICS. Basic Research in Computer Science (www.brics.dk), funded by the Danish National Research Foundation.

Over the last couple of years, Palsberg and Wand have extended this observation to flow types and the flow information gathered by a control-flow analysis (Palsberg & Wand, 2002), designing a CPS transformation of flow information.

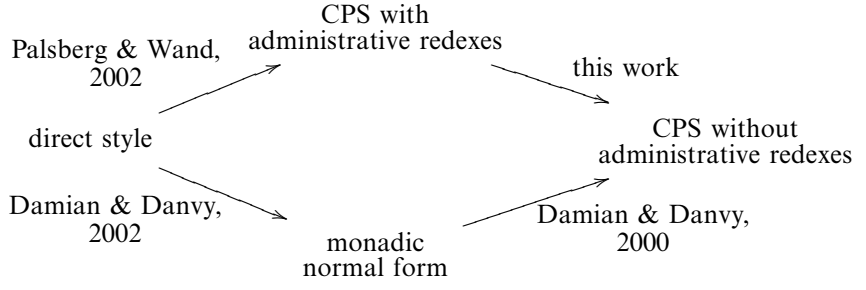



Independently, and with a different motivation, we have also designed a CPS transformation of flow information for control flow and binding times (Damian, 2001; Damian & Danvy, 2000; Damian & Danvy, 2002). The two CPS transformations of flow information correspond to two different takes on the CPS transformation of $\lambda$-terms:




The CPS transformation is Plotkin's (1975). It is a first-order, compositional rewriting system generating numerous administrative redexes that need to be post-reduced in practice (Steele, 1978). Alternatively (Hatcliff & Danvy, 1994; Sabry & Wadler, 1997), the CPS transformation can be staged into a transformation into monadic normal form followed by an introduction of continuations.

The two CPS transformations of flow information can be depicted as follows:




Palsberg and Wand show how to construct in linear time the flow information corresponding to a CPS program obtained through a Plotkin-style CPS transformation (Palsberg & Wand, 2002; Plotkin, 1975). The resulting programs contain all administrative redexes induced by Plotkin's transformation. Therefore,

$$
\begin{array}{lcll}
e & \in & Exp & ::= \quad x \mid n \mid e_1^{\ell^1}\ e_2^{\ell^2} \mid \textbf{if0}\ e^{\ell}\ e_0^{\ell^0}\ e_1^{\ell^1} \mid \lambda^{\pi} x.e^{\ell} \\
\pi & \in & Lam & (\lambda\text{-abstraction labels}) \\
\ell & \in & Lab & (\text{term labels}) \\
n & \in & Lit & (\text{integer literals})
\end{array}
$$

Fig. 1. The language of labelled $\lambda$-terms.

the corresponding CPS information of flow also contains spurious information which accounts for the extraneous $\lambda$-abstractions and their flow. The problem of eliminating this spurious information is open.

Damian and Danvy show how to construct in linear time the flow information corresponding to the introduction of continuations, starting from monadic normal forms (Damian & Danvy, 2000; Hatcliff & Danvy, 1994). They also show how to construct in linear time the flow information corresponding to the transformation into monadic normal forms (Damian, 2001; Damian & Danvy, 2002).

In this work, we complete the picture above by showing how to perform in linear time administrative reductions on CPS-transformed programs (Section 4). Our result hinges on linear reductions (Section 3). But first, we present the source language and a constraint-based control-flow analysis (Section 2).

## 2 Preliminaries

### *2.1 The source language*

Input terms are given by the grammar in Figure 1. Terms are annotated with distinct labels taken from a countable set *Lab*. Each $\lambda$-abstraction is annotated with a distinct label $\pi$ from a set *Lam*, and we assume that there exists a bijection between $\lambda$-abstractions and their labels.

The language has a standard call-by-value semantics, which we leave unspecified. A program $p$ is a closed labelled expression $e^{\ell}$.

*Definition 1*
A *properly labelled* expression is a labelled expression in which all labels are distinct and all variables are distinct.

### *2.2 Control-flow analysis*

We consider a constraint-based control-flow analysis. We use the same notations and definitions as in Nielson, Nielson and Hankin's recent textbook on program analysis (Nielson *et al*., 1999).

Given a program $p$, the control-flow analysis is defined as a relation $\models_p$ whose functionality is displayed in Figure 2.

A solution of the analysis of $p$ is a pair $(\widehat{C}, \widehat{\rho})$ such that $(\widehat{C}, \widehat{\rho}) \models p$. The set of solutions of the analysis is ordered by the natural pointwise ordering of functions, and has a least element. This property ensures the existence of a least solution of the analysis of $p$. The analysis relation is defined inductively on the syntax as defined in Figure 3.

$$
\begin{array}{rl}
Lam^p & \text{The set of } \lambda\text{-abstraction labels in } p \\
Var^p & \text{The set of identifiers in } p \\
Lab^p & \text{The set of term labels in } p
\end{array}
$$

$$
\begin{array}{rcll}
 & Val^p & = \mathscr{P}(Lam^p) & \text{Abstract values} \\
\widehat{C} \in & Cache_p & = Lab^p \to Val^p & \text{Abstract cache} \\
\widehat{\rho} \in & Env_p & = Var^p \to Val^p & \text{Abstract environment}
\end{array}
$$

$$\vDash_p \subseteq (Cache_p \times Env_p) \times Lab^p$$

Fig. 2. Control-flow analysis relation for a program $p$: functionality.

$$
\begin{array}{lcl}
(\widehat{C},\widehat{\rho}) \vDash_p n^{\ell} & \Longleftrightarrow & true \\
(\widehat{C},\widehat{\rho}) \vDash_p x^{\ell} & \Longleftrightarrow & \widehat{\rho}(x) \sqsubseteq \widehat{C}(\ell) \\
(\widehat{C},\widehat{\rho}) \vDash_p (\lambda^{\pi} x.e^{\ell})^{\ell_1} & \Longleftrightarrow & \pi \in \widehat{C}(\ell_1) \wedge (\widehat{C},\widehat{\rho}) \vDash_p e^{\ell} \\
(\widehat{C},\widehat{\rho}) \vDash_p (e_1^{\ell_1}\ e_2^{\ell_2})^{\ell} & \Longleftrightarrow & (\widehat{C},\widehat{\rho}) \vDash_p e_1^{\ell_1} \wedge (\widehat{C},\widehat{\rho}) \vDash_p e_2^{\ell_2} \wedge \\
 & & \forall \lambda^{\pi} x.e_0^{\ell_0} \in \widehat{C}(\ell_1).\widehat{C}(\ell_2) \subseteq \widehat{\rho}(x) \wedge \\
 & & \widehat{C}(\ell_0) \subseteq \widehat{C}(\ell) \\
(\widehat{C},\widehat{\rho}) \vDash_p (\textbf{if0}\ e^{\ell}\ e_0^{\ell_0}\ e_1^{\ell_1})^{\ell_2} & \Longleftrightarrow & (\widehat{C},\widehat{\rho}) \vDash_p e^{\ell} \wedge (\widehat{C},\widehat{\rho}) \vDash_p e_0^{\ell_0} \wedge (\widehat{C},\widehat{\rho}) \vDash_p e_1^{\ell_1} \wedge \\
 & & \widehat{C}(\ell_0) \subseteq \widehat{C}(\ell_2) \wedge \widehat{C}(\ell_1) \subseteq \widehat{C}(\ell_2)
\end{array}
$$

Fig. 3. Control-flow analysis relation for a program $p$: definition.

## 3 Linear reductions

We observe that linear reductions preserve flow information. A linear reduction is a $\beta$-reduction in which the $\lambda$-abstraction in the function position is linear, i.e., such that it uses its argument once and only once. Let us formalize the notion of linear reduction using linear contexts.

*Definition 2*

A *linear context* is a labelled expression with a unique hole $[\cdot]$. Linear contexts are defined by the grammar:

$$
\begin{array}{rcl}
E & ::= & [\cdot] \mid x^{\ell} \mid n^{\ell} \mid (E\ e_2^{\ell_2})^{\ell} \mid (e_1^{\ell_1}\ E)^{\ell} \mid \\
 & & (\textbf{if0}\ E\ e_0^{\ell_0}\ e_1^{\ell_1})^{\ell} \mid (\textbf{if0}\ e^{\ell}\ E\ e_1^{\ell_1})^{\ell_0} \mid (\textbf{if0}\ e^{\ell}\ e_0^{\ell_0}\ E)^{\ell_1} \mid \\
 & & (\lambda^{\pi} x.E)^{\ell}
\end{array}
$$

Given a linear context $E$ and a labelled expression $e^{\ell}$, we use $E[e^{\ell}]$ to denote the context $E$ with the hole $[\cdot]$ replaced with $e^{\ell}$. It is trivial to see that $E[e^{\ell}]$ is a well-formed expression. Note that pluging as defined here does not avoid variable capturing. We use plugging however only in the context of properly-labelled programs, where there is no danger of variable capture.

We also use $FV(e)$ to denote the set of free variables of the expression $e$. This notation naturally extends to contexts: given the context $E$, by definition $FV(E) = FV(E[n])$, where $n$ is an arbitrary literal. We use $L$ as the function extracting the label of an expression. By definition, for any labelled expression $e^{\ell}$, $L(e^{\ell}) = \ell$.

*Definition 3*
A labelled $\lambda$-abstraction $(\lambda^{\pi} x.e^{\ell_1})^{\ell_2}$ is *linear* if and only if is properly labelled and $e^{\ell_1}$ contains a unique occurrence of $x$, i.e., if there exists a linear context $E$ such that $x \notin FV(E)$ and $e = E[x^{\ell}]$ for some label $\ell$.

*Definition 4*
A *linear redex* is a $\beta$-redex $(\lambda x.e_1)\ e_2$ such that $\lambda x.e_1$ is a linear $\lambda$-abstraction.

*Definition 5*
A *linear reduction* is the $\beta$-reduction of a linear redex.

*Example*

$$((\lambda^{\pi} x.E[x^{\ell}])^{\ell_1}\ e^{\ell_2})^{\ell_3} \rightarrow E[e^{\ell_2}]$$

where $E$ is a linear context where $x$ does not occur free. Note that such a reduction might not necessarily be sound wrt. a call-by-value semantics. Nevertheless, we show that it does not affect the result of control-flow analysis. In any case, we treat linear reductions in CPS, which is evaluation-order independent (Plotkin, 1975).

## 4 Control-flow analysis and linear reduction

We show that performing a linear reduction does not alter the results of the analysis of a properly labelled program. More precisely, we show that, given a properly labelled program which contains a linear $\beta$-redex, control-flow analysis yields strictly equivalent results before and after performing a linear $\beta$-reduction.

We are given a program that contains a linear redex and the least solution of its analysis. The goal of this section is to construct the least solution of the analysis of this program after a linear $\beta$-reduction.

Let $p$ be a properly labelled program containing a linear $\beta$-redex. Therefore there exist two linear contexts $E$ and $E_1$, an expression $e$, a fresh variable $x$, and labels $\pi, \ell_0, \ell_1, \ell_2$ and $\ell_3$ such that

$$p = E[((\lambda^{\pi} x.E_1[x^{\ell_0}])^{\ell_1}\ e^{\ell_2})^{\ell_3}]$$

and $x \notin FV(E)$. Let then

$$p' = E[E_1[e^{\ell_2}]]$$

be the program $p$ with the linear redex above reduced. It is immediate to see that $p'$ is also a properly labelled program.

In the rest of this section, we define a monotone function $\mathscr{F}_p$ which, given a solution of the analysis of $p$, constructs a solution of the analysis of $p'$. We then define a reverse function $\mathscr{G}_p$, monotone as well, which, given a solution of the analysis of $p'$, constructs a solution of the analysis of $p$. Using the two functions and their monotonicity, we show that the best solution for $p$ is transformed into the best solution for $p'$. We then show how to construct in linear time the least solution of the analysis of $p'$ from the least solution of the analysis of $p$.

### 4.1 Flow constructions

For the programs $p$ and $p'$ defined as above, by construction,

- $Lab^p = Lab^{p'} \cup \{\ell_0, \ell_1, \ell_3\}$,
- $Lam^p = Lam^{p'} \cup \{\pi\}$, and
- $Var^p = Var^{p'} \cup \{x\}$.

We define a function $\mathscr{F}_p : (Cache_p \times Env_p) \to (Cache_{p'} \times Env_{p'})$ as $\mathscr{F}_p(\widehat{C}, \widehat{\rho}) = (\widehat{C}|_{Lam^{p'}}, \widehat{\rho}|_{Lam^{p'}})$. Obviously, $\mathscr{F}_p$ is a projection function and it is monotone with respect to the ordering of solutions.

We define a reverse function $\mathscr{G}_p : (Cache_{p'} \times Env_{p'}) \to (Cache_p \times Env_p)$ as follows. $\mathscr{G}_p(\widehat{C}', \widehat{\rho}') = (\widehat{C}, \widehat{\rho})$ such that:

- for all $\ell \in Lab^{p'}$, $\widehat{C}(\ell) = \widehat{C}'(\ell)$; $\widehat{C}(\ell_3) = \widehat{C}'(L(E_1[e^{\ell_2}]))$; $\widehat{C}(\ell_0) = \widehat{\rho}(x) = \widehat{C}'(\ell_2)$; $\widehat{C}(\ell_1) = \{\pi\}$; and
- for all $y \in Var^{p'}$, $\widehat{\rho}(y) = \widehat{\rho}'(y)$.

Obviously, $\mathscr{G}_p$ is an embedding function and it is monotone as well.

*Lemma 4.1*
Let $(\widehat{C}, \widehat{\rho}) \in (Cache_p \times Env_p)$ such that $(\widehat{C}, \widehat{\rho}) \vDash_p p$. Then $\mathscr{F}_p(\widehat{C}, \widehat{\rho}) \vDash_{p'} p'$.

*Proof*
Let $(\widehat{C}', \widehat{\rho}') = \mathscr{F}_p(\widehat{C}, \widehat{\rho})$. We show that $(\widehat{C}', \widehat{\rho}') \vDash_{p'} p'$. The proof has two steps:

(i) A proof of the fact that $(\widehat{C}', \widehat{\rho}') \vDash_{p'} E_1[e^{\ell_2}]$. The proof is by structural induction on the context $E_1$, using the assumption that $(\widehat{C}', \widehat{\rho}') \vDash_{p'} E_1[x^{\ell_0}]$.
(ii) A proof of the fact that $(\widehat{C}', \widehat{\rho}') \vDash_{p'} E[E_1[e^{\ell_2}]]$. The proof is by structural induction on the context $E$. □

*Lemma 4.2*
Let $(\widehat{C}', \widehat{\rho}') \in (Cache_{p'} \times Env_{p'})$ such that $(\widehat{C}', \widehat{\rho}') \vDash_{p'} p'$. Then $\mathscr{G}_p(\widehat{C}', \widehat{\rho}') \vDash_p p$.

*Proof*
Let $(\widehat{C}, \widehat{\rho}) = \mathscr{G}_p(\widehat{C}', \widehat{\rho}')$. We show that $(\widehat{C}, \widehat{\rho}) \vDash_p p$. The proof has three steps:

(i) A proof of the fact that $(\widehat{C}, \widehat{\rho}) \vDash_p E_1[x^{\ell_0}]$. The proof is by structural induction on the context $E_1$, using the assumption that $(\widehat{C}', \widehat{\rho}') \vDash_{p'} E_1[e^{\ell_2}]$.
(ii) A proof of the fact that $(\widehat{C}, \widehat{\rho}) \vDash_p ((\lambda^{\pi} x.E_1[x^{\ell_0}])^{\ell_1}\ e^{\ell_2})^{\ell_3}$. Using (i), the proof amounts to showing that a small set of constraints are satisfied.
(iii) A proof of the fact that $(\widehat{C}, \widehat{\rho}) \vDash_p E[((\lambda^{\pi} x.E_1[x^{\ell_0}])^{\ell_1}\ e^{\ell_2})^{\ell_3}]$. The proof is by structural induction on the context $E$. □

*Lemma 4.3*
Let $(\widehat{C}, \widehat{\rho})$ be the least solution of the analysis of $p$. Let $(\widehat{C}', \widehat{\rho}')$ be the least solution of the analysis of $p'$. Then $\mathscr{F}_p(\widehat{C}, \widehat{\rho}) = (\widehat{C}', \widehat{\rho}')$ and $\mathscr{G}_p(\widehat{C}', \widehat{\rho}') = (\widehat{C}, \widehat{\rho})$.

*Proof*
We can immediately see that $\mathscr{F}_p(\mathscr{G}_p(\widehat{C}', \widehat{\rho}')) = (\widehat{C}', \widehat{\rho}')$, and that $\mathscr{G}_p(\mathscr{F}_p(\widehat{C}, \widehat{\rho})) \sqsubseteq (\widehat{C}, \widehat{\rho})$. Therefore, $\mathscr{G}_p$ and $\mathscr{F}_p$ form an embedding/projection pair.

Since $(\widehat{C}, \widehat{\rho})$ is the least solution, then $(\widehat{C}, \widehat{\rho}) \sqsubseteq \mathscr{G}_p(\widehat{C}', \widehat{\rho}')$. Using the monotonicity of $\mathscr{G}_p$, we obtain that $\mathscr{F}_p(\widehat{C}, \widehat{\rho}) \sqsubseteq \mathscr{F}_p(\mathscr{G}_p(\widehat{C}', \widehat{\rho}')) = (\widehat{C}', \widehat{\rho}')$. Since $\mathscr{F}_p(\widehat{C}, \widehat{\rho})$ is a solution and $(\widehat{C}', \widehat{\rho}')$ is the least solution, we obtain that $\mathscr{F}_p(\widehat{C}, \widehat{\rho}) = (\widehat{C}', \widehat{\rho}')$ and then that $\mathscr{G}_p(\widehat{C}', \widehat{\rho}') = (\widehat{C}, \widehat{\rho})$. □

### *4.2 The CPS transformation of flow information and administrative reductions*

Lemma 4.3 says that the least analysis after a linear $\beta$-reduction is a restriction of the least analysis of the initial term. From this, we can infer that any linear $\beta$-reduction does not alter the results of the CFA. We use this result to show that administrative reductions after Plotkin's CPS transformation do not change the result of the flow analysis.

*Theorem 4.4*
Let $p$ be a program, $p_1$ be its CPS counterpart without administrative reductions, and $p_2$ be its CPS counterpart after administrative reduction. Let $(\widehat{C}_1, \widehat{\rho}_1)$ be the least solution of the analysis of $p_1$. The least solution $(\widehat{C}_2, \widehat{\rho}_2)$ of the analysis of $p_2$ can be obtained in linear time from $(\widehat{C}_1, \widehat{\rho}_1)$, by restricting $(\widehat{C}_1, \widehat{\rho}_1)$ to the program points preserved by the administrative reductions.

*Proof*
All administrative reductions are linear, and furthermore, administrative reduction is known to terminate (Danvy & Filinski, 1992). We apply Lemma 4.3. □

*Corollary 4.5*
Let $p$ be a program, $p_1$ be its CPS counterpart without administrative reductions, and $p_2$ be its CPS counterpart after administrative reduction. Let $(\widehat{C}, \widehat{\rho})$ be the least solution of the analysis of $p$. The least solution $(\widehat{C}_2, \widehat{\rho}_2)$ of the analysis of $p_2$ can be obtained in linear time from $(\widehat{C}, \widehat{\rho})$.

*Proof*
We compose the construction given by Theorem 4.4 with Palsberg and Wand's CPS transformation of flow information (Palsberg & Wand, 2002), which also works in linear time. □

## 5 Related work

Our key observation is that linear $\beta$-reduction preserves leastness. This property is a corollary of Henglein's subject invariance property of linear $\beta$-reductions for monomorphic program analyses (Henglein, 1994; Henglein & Mairson, 1991; Henglein & Mairson, 1994). Non-linear $\beta$-reductions, while they do preserve correctness of an analysis (Wand & Williamson, 2001), are known not to preserve leastness.

## 6 Conclusion and issues

We have shown how to complement Palsberg and Wand's CPS transformation of flow information with administrative reductions, while preserving its linear-time complexity. Our extension hinges on the linearity of administrative redexes.

Let us now show how to integrate administrative reductions in Palsberg and Wand's CPS transformation, therefore making it operate in one pass, still in linear time. As shown in 'Representing Control' (Danvy & Filinski, 1992), at CPS-transformation time, one can segregate the administrative lambdas and applications and the residual ones. (The residual lambdas and applications are the ones preserved by the administrative reductions.) Therefore, in Palsberg and Wand's CPS transformation of flow information, we can segregate the labels of the administrative lambdas and applications and the labels of the residual ones as well. In practice, the solution after administrative reduction is thus obtained simply by restricting Palsberg and Wand's solution to the residual labels. In the overall process of (1) CPS transformation, and (2) administrative reduction, the administrative labels are used transitorily, just as in the one-pass CPS transformation, which is conceptually fitting.

## References

Damian, D. (2001) *On static and dynamic control-flow information in program analysis and transformation*. PhD thesis, BRICS PhD School, University of Aarhus, Aarhus, Denmark. BRICS DS-01-5.

Damian, D. and Danvy, O. (2000) Syntactic accidents in program analysis: On the impact of the CPS transformation. In: Wadler, P. (ed.), *Proceedings 2000 ACM SIGPLAN International Conference on Functional Programming*. (*SIGPLAN Notices*, **35**(9).) ACM Press.

Damian, D. and Danvy, O. (2002) Syntactic accidents in program analysis: On the impact of the CPS transformation. *J. Functional Program.* To appear.

Danvy, O. and Filinski, A. (1992) Representing control, a study of the CPS transformation. *Math. Struct. Comput. Sci.* **2**(4), 361–391.

Hatcliff, J. and Danvy, O. (1994) A generic account of continuation-passing styles. In: Boehm, H.-J. (ed.), *Proceedings 21st Annual ACM Symposium on Principles of Programming Languages*, pp. 458–471. ACM Press.

Henglein, F. (1994) Dynamic typing: Syntax and proof theory. *Sci. Comput. Program.* **22**(3), 197–230.

Henglein, F. and Mairson, H. (1991) The complexity of type inference for higher-order typed lambda calculi. In: Cartwright, R. (Corky) (ed.), *Proceedings 18th Annual ACM Symposium on Principles of Programming Languages*, pp. 119–130. ACM Press.

Henglein, F. and Mairson, H. (1994) The complexity of type inference for higher-order typed lambda calculi. *J. Functional Program.* **4**(4), 435–477.

Meyer, A. R. and Wand, M. (1985) Continuation semantics in typed lambda-calculi (summary). In: Parikh, R. (ed.), *Logics of Programs – Proceedings: Lecture Notes in Computer Science 193*, pp. 219–224. Springer-Verlag.

Nielson, F., Nielson, H. R. and Hankin, C. (1999) *Principles of Program Analysis*. Springer Verlag.

Palsberg, J. and Wand, M. (2002) CPS transformation of flow information. *J. Functional Program.* To appear.

Plotkin, G. D. (1975) Call-by-name, call-by-value and the $\lambda$-calculus. *Theo. Comput. Sci.* **1**, 125–159.

Reynolds, J. C. (1993) The discoveries of continuations. *Lisp and Symbolic Computation*, **6**(3/4), 233–247.

Sabry, A. and Wadler, P. (1997) A reflection on call-by-value. *ACM Trans. Program. Lang. & Syst*. **19**(6), 916–941.

Steele Jr., G. L. (1978) *Rabbit: A compiler for Scheme*. Technical report AI-TR-474, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Wand, M. (1985) Embedding type structure in semantics. In: Van Deusen, M. S. and Galil, Z. (eds.), *Proceedings 12th Annual ACM Symposium on Principles of Programming Languages*, pp. 1–6. ACM Press.

Wand, M. and Williamson, G. B. (2001) A modular, extensible proof method for small-step flow analyses. In: Le Métayer, D. (ed.), *Programming Languages and Systems, 11th European Symposium on Programming, ESOP 2002: Lecture Notes in Computer Science 2305*, pp. 213–227. Springer-Verlag.