BRICS

Basic Research in Computer Science

BRICS RS-02-36 Damian & Danvy: CPS Transformation of Flow Information, Part II: Administrative Reductions

CPS Transformation of Flow Information Part II: Administrative Reductions

Daniel Damian
Olivier Danvy

BRICS Report Series RS-02-36

ISSN 0909-0878 August 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/02/36/`

CPS Transformation of Flow Information, Part II: Administrative Reductions *

Daniel Damian[†] and Olivier Danvy

BRICS[‡]
Department of Computer Science
University of Aarhus[§]

August 13, 2000 (updated: August 9, 2001 and August 6, 2002)

Abstract

We characterize the impact of a linear β-reduction on the result of a control-flow analysis. (By "a linear β-reduction" we mean the β-reduction of a linear λ-abstraction, i.e., of a λ-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.

*To appear in the Journal of Functional Programming.

†Current affiliation: LION Bioscience Ltd., Compass House, 80-82 Newmarket Road, Cambridge CB5 8DZ, UK.

‡Basic Research in Computer Science (`www.brics.dk`), funded by the Danish National Research Foundation.

§Ny Munkegade, Building 540, DK-8000 Aarhus C, Denmark
Email:{`damian,danvy`}`@brics.dk`

1 Background and introduction

Since their inception, over thirty years ago [13], 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 [9, 16], Meyer and Wand noticed that the CPS transformation preserves types and they constructed a CPS transformation of types.

$$\begin{tikzcd}
\text{type} \arrow[r, dashed, "\text{CPS transformation}", "\text{of types}"'] & \text{type} \\
\text{direct-style program} \arrow[u, "\text{type inference}"] \arrow[r, "\text{CPS transformation}", "\text{of terms}"'] & \text{CPS program} \arrow[u, "\text{type inference}"']
\end{tikzcd}$$

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 [11], designing a CPS transformation of flow information.

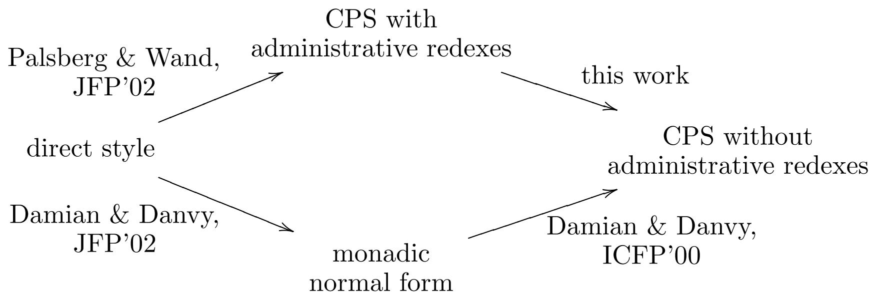
$$\begin{tikzcd}
\text{flow information} \arrow[r, dashed, "\text{CPS transformation}", "\text{of flow}"'] & \text{flow information} \\
\text{direct-style program} \arrow[u, "\text{flow analysis}"] \arrow[r, "\text{CPS transformation}", "\text{of terms}"'] & \text{CPS program} \arrow[u, "\text{flow analysis}"']
\end{tikzcd}$$

Independently, and with a different motivation, we have also designed a CPS transformation of flow information for control flow and binding times [1–3]. The two CPS transformations of flow information correspond to two different takes on the CPS transformation of λ-terms:

$$\begin{tikzcd}
 & \text{CPS with administrative redexes} \arrow[dr, "\text{administrative reductions}"] & \\
\text{direct style} \arrow[ur, "\text{CPS transformation}"] \arrow[dr, "\text{transformation into monadic style}"'] & & \text{CPS without administrative redexes} \\
 & \text{monadic normal form} \arrow[ur, "\text{introduction of continuations}"'] &
\end{tikzcd}$$

The CPS transformation is Plotkin's [12]. It is a first-order, compositional rewriting system generating numerous administrative redexes that need to be post-reduced in practice [15]. Alternatively [5, 14], 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 [11, 12]. The resulting programs contain all administrative redexes induced by Plotkin's transformation. Therefore, the corresponding CPS information of flow also contains spurious information which accounts for the extraneous λ-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 [2, 5]. They also show how to construct in linear time the flow information corresponding to the transformation into monadic normal forms [1, 3].

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 λ-abstraction is annotated with a distinct label π from a set *Lam*, and we assume that there exists a bijection between λ-abstractions and their labels.

$$
\begin{array}{cccll}
e & \in & \mathit{Exp} & ::= & x \mid n \mid e_1^{\ell_1}\ e_2^{\ell_2} \mid \mathbf{if0}\ e^{\ell}\ e_0^{\ell_0}\ e_1^{\ell_1} \mid \lambda^{\pi}x.e^{\ell} \\
\pi & \in & \mathit{Lam} & & (\lambda\text{-abstraction labels}) \\
\ell & \in & \mathit{Lab} & & (\text{term labels}) \\
n & \in & \mathit{Lit} & & (\text{integer literals})
\end{array}
$$

Figure 1: The language of labeled λ-terms

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

Definition 2.1. *A* properly labeled *expression is a labeled 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 [10].

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

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

$$\begin{array}{rcll} & \mathit{Val}^p & = \mathcal{P}(\mathit{Lam}^p) & \text{Abstract values} \\ \widehat{C} \in & \mathit{Cache}_p & = \mathit{Lab}^p \to \mathit{Val}^p & \text{Abstract cache} \\ \widehat{\rho} \in & \mathit{Env}_p & = \mathit{Var}^p \to \mathit{Val}^p & \text{Abstract environment} \end{array}$$

$$\models_p \subseteq (\mathit{Cache}_p \times \mathit{Env}_p) \times \mathit{Lab}^p$$

Figure 2: Control-flow analysis relation for a program p: functionality

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.

$$\begin{array}{lll} (\widehat{C}, \widehat{\rho}) \models_p n^\ell & \iff & \mathit{true} \\ (\widehat{C}, \widehat{\rho}) \models_p x^\ell & \iff & \widehat{\rho}(x) \sqsubseteq \widehat{C}(\ell) \\ (\widehat{C}, \widehat{\rho}) \models_p (\lambda^\pi x.e^\ell)^{\ell_1} & \iff & \pi \in \widehat{C}(\ell_1) \wedge (\widehat{C}, \widehat{\rho}) \models_p e^\ell \\ (\widehat{C}, \widehat{\rho}) \models_p (e_1^{\ell_1}\ e_2^{\ell_2})^\ell & \iff & (\widehat{C}, \widehat{\rho}) \models_p e_1^{\ell_1} \wedge (\widehat{C}, \widehat{\rho}) \models_p e_2^{\ell_2} \wedge \\ & & \forall \lambda^\pi x.e_0^{\ell_0} \in \widehat{C}(\ell_1).\widehat{C}(\ell_2) \subseteq \widehat{\rho}(x) \wedge \\ & & \qquad\qquad\qquad\quad \widehat{C}(\ell_0) \subseteq \widehat{C}(\ell) \\ (\widehat{C}, \widehat{\rho}) \models_p (\mathbf{if0}\ e^\ell\ e_0^{\ell_0}\ e_1^{\ell_1})^{\ell_2} & \iff & (\widehat{C}, \widehat{\rho}) \models_p e^\ell \wedge (\widehat{C}, \widehat{\rho}) \models_p e_0^{\ell_0} \wedge (\widehat{C}, \widehat{\rho}) \models_p e_1^{\ell_1} \wedge \\ & & \widehat{C}(\ell_0) \subseteq \widehat{C}(\ell_2) \wedge \widehat{C}(\ell_1) \subseteq \widehat{C}(\ell_2) \end{array}$$

Figure 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 β-reduction in which the λ-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 3.1. *A* linear context *is a labeled expression with a unique hole* $[\cdot]$. *Linear contexts are defined by the grammar:*

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

Given a linear context E and a labeled expression e^ℓ, we use $E[e^\ell]$ to denote the context E with the hole $[\cdot]$ replaced with e^ℓ. 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-labeled 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 labeled expression e^ℓ, $L(e^\ell) = \ell$.

Definition 3.2. *A labeled λ-abstraction $(\lambda^\pi x.e^{\ell_1})^{\ell_2}$ is* linear *if and only if is properly labeled and e^{ℓ_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 ℓ.*

Definition 3.3. *A* linear redex *is a β-redex $(\lambda x.e_1)\ e_2$ such that $\lambda x.e_1$ is a linear λ-abstraction.*

Definition 3.4. *A* linear reduction *is the β-reduction of a linear redex.*

Example:

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

where E is a linear context where x does not occur free. Note that such a reduction might not necessarily be sound wrt. 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 [12].

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 labeled program. More precisely, we show that, given a properly labeled program which contains a linear β-redex, control-flow analysis yields strictly equivalent results before and after performing a linear β-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 β-reduction.

Let p be a properly labeled program containing a linear β-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 ℓ_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 labeled program.

In the rest of this section, we define a monotone function $\mathcal{F}_p$ which, given a solution of the analysis of p, constructs a solution of the analysis of p'. We then define a reverse function $\mathcal{G}_p$, 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 $\mathcal{F}_p : (Cache_p \times Env_p) \to (Cache_{p'} \times Env_{p'})$ as $\mathcal{F}_p(\widehat{C}, \widehat{\rho}) = (\widehat{C}|_{Lam^{p'}}, \widehat{\rho}|_{Lam^{p'}})$. Obviously, $\mathcal{F}_p$ is a projection function and it is monotone with respect to the ordering of solutions.

We define a reverse function $\mathcal{G}_p : (Cache_{p'} \times Env_{p'}) \to (Cache_p \times Env_p)$ as follows. $\mathcal{G}_p(\widehat{C}', \widehat{\rho}') = (\widehat{C}, \widehat{\rho})$ 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, $\mathcal{G}_p$ 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 $(\widehat{C}, \widehat{\rho}) \models_p p$. Then $\mathcal{F}_p(\widehat{C}, \widehat{\rho}) \models_{p'} p'$.*

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

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

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

$\square$

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

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

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

ii) A proof of the fact that $(\widehat{C}, \widehat{\rho}) \models_p ((\lambda^\pi x.E_1[x^{\ell_0}])^{\ell_1}\ e^{\ell_2})^{\ell_3}$. Using i), the proof amounts to showing that a small set of constraints are satisfied.

iii) A proof of the fact that $(\widehat{C}, \widehat{\rho}) \models_p E[((\lambda^\pi x.E_1[x^{\ell_0}])^{\ell_1}\ e^{\ell_2})^{\ell_3}]$. The proof is by structural induction on the context E.

$\square$

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* $\mathcal{F}_p(\widehat{C}, \widehat{\rho}) = (\widehat{C}', \widehat{\rho}')$ *and* $\mathcal{G}_p(\widehat{C}', \widehat{\rho}') = (\widehat{C}, \widehat{\rho})$.

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

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

4.2 The CPS transformation of flow information and administrative reductions

Lemma 4.3 says that the least analysis after a linear β-reduction is a restriction of the least analysis of the initial term. From this, we can infer that any linear β-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 [4]. We apply Lemma 4.3. $\square$

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 [11], which also works in linear time. □

5 Related work

Our key observation is that linear β-reduction preserves leastness. This property is a corollary of Henglein's subject invariance property of linear β-reductions for monomorphic program analyses [6–8]. Non-linear β-reductions, while they do preserve correctness of an analysis [17], 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" [4], 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

[1] Daniel Damian. *On Static and Dynamic Control-Flow Information in Program Analysis and Transformation.* PhD thesis, BRICS PhD School, University of Aarhus, Aarhus, Denmark, July 2001. BRICS DS-01-5.

[2] Daniel Damian and Olivier Danvy. Syntactic accidents in program analysis: On the impact of the CPS transformation. In Philip Wadler, editor, *Proceedings of the 2000 ACM SIGPLAN International Conference on Functional Programming*, SIGPLAN Notices, Vol. 35, No. 9, pages 209–220, Montréal, Canada, September 2000. ACM Press. Extended version to appear in the Journal of Functional Programming.

[3] Daniel Damian and Olivier Danvy. Syntactic accidents in program analysis: On the impact of the CPS transformation. *Journal of Functional Programming*, 2002. To appear. Extended version available as the technical report BRICS-RS-01-54.

[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] John Hatcliff and Olivier Danvy. A generic account of continuation-passing styles. In Hans-J. Boehm, editor, *Proceedings of the Twenty-First Annual ACM Symposium on Principles of Programming Languages*, pages 458–471, Portland, Oregon, January 1994. ACM Press.

[6] Fritz Henglein. Dynamic typing: Syntax and proof theory. *Science of Computer Programming*, 22(3):197–230, 1994.

[7] Fritz Henglein and Harry Mairson. The complexity of type inference for higher-order typed lambda calculi. In Robert (Corky) Cartwright, editor, *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Programming Languages*, pages 119–130, Orlando, Florida, January 1991. ACM Press.

[8] Fritz Henglein and Harry Mairson. The complexity of type inference for higher-order typed lambda calculi. *Journal of Functional Programming*, 4(4):435–477, 1994.

[9] Albert R. Meyer and Mitchell Wand. Continuation semantics in typed lambda-calculi (summary). In Rohit Parikh, editor, *Logics of Programs – Proceedings*, number 193 in Lecture Notes in Computer Science, pages 219–224, Brooklyn, New York, June 1985. Springer-Verlag.

[10] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer Verlag, 1999.

[11] Jens Palsberg and Mitchell Wand. CPS transformation of flow information. *Journal of Functional Programming*, 2002. To appear.

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

[13] John C. Reynolds. The discoveries of continuations. *Lisp and Symbolic Computation*, 6(3/4):233–247, 1993.

[14] Amr Sabry and Philip Wadler. A reflection on call-by-value. *ACM Transactions on Programming Languages and Systems*, 19(6):916–941, 1997.

[15] 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.

[16] Mitchell Wand. Embedding type structure in semantics. In Mary S. Van Deusen and Zvi Galil, editors, *Proceedings of the Twelfth Annual ACM Symposium on Principles of Programming Languages*, pages 1–6, New Orleans, Louisiana, January 1985. ACM Press.

[17] Mitchell Wand and Galen B. Williamson. A modular, extensible proof method for small-step flow analyses. In Daniel Le Métayer, editor, *Programming Languages and Systems, 11th European Symposium on Programming, ESOP 2002*, number 2305 in Lecture Notes in Computer Science, pages 213–227, Grenoble, France, April 2001. Springer-Verlag.

Recent BRICS Report Series Publications

RS-02-36 **Daniel Damian and Olivier Danvy.** ***CPS Transformation of Flow Information, Part II: Administrative Reductions*****. August 2002. 9 pp. To appear in the *Journal of Functional Programming*. This report supersedes the earlier BRICS report RS-01-40.**

RS-02-35 **Patricia Bouyer.** ***Timed Automata May Cause Some Troubles*****. August 2002. 44 pp.**

RS-02-34 **Morten Rhiger.** ***A Foundation for Embedded Languages*****. August 2002. 29 pp.**

RS-02-33 **Vincent Balat and Olivier Danvy.** ***Memoization in Type-Directed Partial Evaluation*****. July 2002. 18 pp. To appear in Batory and Consel, editors, *ACM SIGPLAN/SIGSOFT Conference on Generative Programming and Component Engineering*, GPCE '02 Proceedings, LNCS, 2002.**

RS-02-32 **Mads Sig Ager, Olivier Danvy, and Henning Korsholm Rohde.** ***On Obtaining Knuth, Morris, and Pratt's String Matcher by Partial Evaluation*****. July 2002. 43 pp. To appear in Chin, editor, *ACM SIGPLAN ASIAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, ASIA-PEPM '02 Proceedings, 2002.**

RS-02-31 **Ulrich Kohlenbach and Paulo B. Oliva.** ***Proof Mining: A Systematic Way of Analysing Proofs in Mathematics*****. June 2002. 47 pp.**

RS-02-30 **Olivier Danvy and Ulrik P. Schultz.** ***Lambda-Lifting in Quadratic Time*****. June 2002.**

RS-02-29 **Christian N. S. Pedersen and Tejs Scharling.** ***Comparative Methods for Gene Structure Prediction in Homologous Sequences*****. June 2002. 20 pp.**

RS-02-28 **Ulrich Kohlenbach and Laurenţiu Leuştean.** ***Mann Iterates of Directionally Nonexpansive Mappings in Hyperbolic Spaces*****. June 2002. 33 pp.**

RS-02-27 **Anna Östlin and Rasmus Pagh.** ***Simulating Uniform Hashing in Constant Time and Optimal Space*****. 2002. 11 pp.**

RS-02-26 **Margarita Korovina.** ***Fixed Points on Abstract Structures without the Equality Test*****. June 2002.**