BRICS
Basic Research in Computer Science

CPS Transformation of Flow Information
Part II: Administrative Reductions

Daniel Damian
Olivier Danvy

BRICS Report Series RS-01-40
ISSN 0909-0878 October 2001

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/01/40/`

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)

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. Since we know how to construct least solutions, preservation of least solutions solves a problem that was left open in Palsberg and Wand's paper "CPS Transformation of Flow Information."

Therefore, together, Palsberg and Wand's article "CPS Transformation of Flow Information" 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.

*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 [10], 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 [6, 13], Meyer and Wand noticed that the CPS transformation preserves types and constructed a CPS transformation of types.

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

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

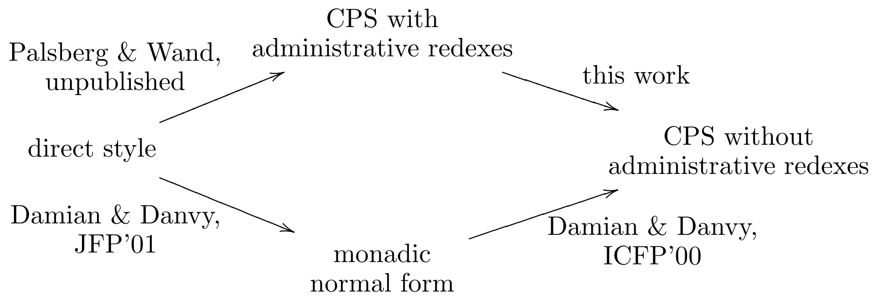
$$\begin{tikzcd}
\substack{\text{flow}\\\text{information}} \arrow[rr, dashed, "\text{CPS transformation}", "\text{of flow}"'] & & \substack{\text{flow}\\\text{information}} \\
\text{direct-style program} \arrow[u, "\substack{\text{flow}\\\text{analysis}}"] \arrow[rr, "\text{CPS transformation}", "\text{of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

Independently, and with a different motivation, we have also designed a CPS transformation of flow information for control flow and binding times [1, 2, 3]. The two CPS transformations of flow information correspond to two different takes on the CPS transformation of λ-terms:

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

The CPS transformation is Plotkin's [9]. It is a first-order, compositional rewriting system generating numerous administrative redexes that need to be post-reduced in practice [12]. Alternatively [5, 11], 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 [8, 9]. 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

We consider that 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 consider that there exists a bijection between λ-abstractions and their labels.

We consider that the language has a standard call-by-value semantics, which is left 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.*

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

Figure 1: The language of labeled λ-terms

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

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

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

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

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 [7].

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.

3 Linear reductions

We observed 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* $[\cdot]$.

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

Figure 3: Control-flow analysis relation for a program p: definition

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.

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 λ-abstraction $\lambda^\pi x.e^\ell$ is* linear *if and only if is properly labeled and e^ℓ and contains a unique free occurrence of x, i.e., if there exists a linear context E such that $x \notin FV(E)$ and $e = E[x^{\ell_1}]$ for some label ℓ_1.*

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 [9].

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 exists 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 $\mathcal{F}_p : (Cache_p \times Env_p) \rightarrow (Cache_{p'} \times Env_{p'})$ 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 $\mathcal{G}_p : (Cache_{p'} \times Env_{p'}) \rightarrow (Cache_p \times Env_p)$ as follows. If $\mathcal{G}_p(\widehat{C}', \widehat{\rho}') = (\widehat{C}, \widehat{\rho})$ then:

- 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.

□

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 two 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.

□

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. Using the monotonicity of the two functions, we obtain that $\mathcal{F}_p(\widehat{C}, \widehat{\rho}) = (\widehat{C}', \widehat{\rho}')$ and $\mathcal{G}_p(\widehat{C}', \widehat{\rho}') = (\widehat{C}, \widehat{\rho})$. □

4.2 The CPS transformation of flow information and administrative reductions

Lemma 4.3 says that the least analysis after a linear 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. □

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

5 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.

[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, pages 209–220, Montréal, Canada, September 2000. ACM Press.

[3] Daniel Damian and Olivier Danvy. Syntactic accidents in program analysis: On the impact of the CPS transformation. *Journal of Functional Programming*, 2001. To appear.

[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] 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, June 1985. Springer-Verlag.

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

[8] Jens Palsberg and Mitchell Wand. CPS transformation of flow information. Unpublished manuscript, available at `http://www.cs.purdue.edu/~palsberg/publications.html`, June 2001.

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

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

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

[12] 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.

[13] 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.

Recent BRICS Report Series Publications

RS-01-40 Daniel Damian and Olivier Danvy. ***CPS Transformation of Flow Information, Part II: Administrative Reductions*****. October 2001. 9 pp. To appear in the Journal of Functional Programming.**

RS-01-39 Olivier Danvy and Mayer Goldberg. ***There and Back Again*****. October 2001. 14 pp. To appear in the proceedings of the 2002 ACM SIGPLAN International Conference on Functional Programming.**

RS-01-38 Zoltán Ésik. ***Free De Morgan Bisemigroups and Bisemilattices*****. October 2001. 13 pp. To appear in the journal** ***Algebra Colloquium*****.**

RS-01-37 Ronald Cramer and Victor Shoup. ***Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption*****. October 2001. 34 pp.**

RS-01-36 Gerth Stølting Brodal, Rolf Fagerberg, and Riko Jacob. ***Cache Oblivious Search Trees via Binary Trees of Small Height*****. October 2001. 20 pp. Appears in Eppstein, editor,** ***The Thirteens Annual ACM-SIAM Symposium on Discrete Algorithms,*** **SODA '02 Proceedings, 2002, pages 39–48.**

RS-01-35 Mayer Goldberg. ***A General Schema for Constructing One-Point Bases in the Lambda Calculus*****. September 2001. 8 pp.**

RS-01-34 Flemming Friche Rodler and Rasmus Pagh. ***Fast Random Access to Wavelet Compressed Volumetric Data Using Hashing*****. August 2001. 31 pp. To appear in** ***ACM Transactions on Graphics*****.**

RS-01-33 Rasmus Pagh and Flemming Friche Rodler. ***Lossy Dictionaries*****. August 2001. 14 pp. Short version appears in Meyer auf der Heide, editor,** ***9th Annual European Symposium on Algorithms,*** **ESA '01 Proceedings, LNCS 2161, 2001, pages 300–311.**

RS-01-32 Rasmus Pagh and Flemming Friche Rodler. ***Cuckoo Hashing*****. August 2001. 21 pp. Short version appears in Meyer auf der Heide, editor,** ***9th Annual European Symposium on Algorithms,*** **ESA '01 Proceedings, LNCS 2161, 2001, pages 121–133.**