BRICS RS-02-40 Aceto et al.: A Note on an Expressiveness Hierarchy for Multi-exit Iteration

Basic Research in Computer Science

A Note on an Expressiveness Hierarchy for Multi-exit Iteration

Luca Aceto
Willem Jan Fokkink
Anna Ingólfsdóttir

BRICS Report Series **RS-02-40**

ISSN 0909-0878 **September 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/02/40/

A Note on an Expressiveness Hierarchy for Multi-exit Iteration

Luca Aceto* Wan Fokkink† Anna Ingólfsdóttir*‡

Abstract

Multi-exit iteration is a generalization of the standard binary Kleene star operation that allows for the specification of agents that, up to bisimulation equivalence, are solutions of systems of recursion equations of the form

$$\begin{array}{rcl} X_1 & \stackrel{\text{def}}{=} & P_1 X_2 + Q_1 \\ & \vdots & \\ X_n & \stackrel{\text{def}}{=} & P_n X_1 + Q_n \end{array}$$

where n is a positive integer, and the P_i and the Q_i are process terms. The addition of multi-exit iteration to Basic Process Algebra (BPA) yields a more expressive language than that obtained by augmenting BPA with the standard binary Kleene star. This note offers an expressiveness hierarchy, modulo bisimulation equivalence, for the family of multi-exit iteration operators proposed by Bergstra, Bethke and Ponse.

AMS Subject Classification (1991): 68Q15, 68Q70.
CR Subject Classification (1991): D.3.1, F.1.1, F.4.1.
Keywords and Phrases: Concurrency, process algebra, Basic Process Algebra (BPA), multi-exit iteration, bisimulation, expressiveness.

1 Background

For the sake of completeness and readability, we begin by recalling the relevant notions from [1] that will be needed in this note. The interested reader is referred to *op. cit.* and [5] for motivation and further information.

We assume a non-empty alphabet A of atomic actions, with typical elements a, b. The language $\mathrm{BPA}^{me*}(A)$ of terms over Basic Process Algebra (BPA) with multi-exit iteration is defined inductively as follows:

*__BRICS__ (__B__asic __R__esearch __i__n __C__omputer __S__cience), Centre of the Danish National Research Foundation, Department of Computer Science, Aalborg University, Fr. Bajersvej 7E, 9220 Aalborg Ø, Denmark. Email: {luca,annai}@cs.auc.dk.

†CWI, Department of Software Engineering, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. Email: wan@cwi.nl.

‡deCODE Genetics, Sturlugata 8, 101 Reykjavik, Iceland.

- each $a \in A$ is a term;
- $P + Q$ and $P \cdot Q$ are terms, if so are P and Q;
- $(P_1, ..., P_m)^*(Q_1, ..., Q_n)$ is a term, if so are $P_1, ..., P_m$ and $Q_1, ..., Q_n$ for some positive integers m and n.

We shall use P, Q, R (possibly subscripted and/or superscripted) to range over $\mathrm{BPA}^{me*}(A)$. In writing terms over the above syntax, we shall always assume that the operation $\cdot$ binds stronger than $+$. In the sequel the operation $\cdot$ will often be omitted, so PQ denotes $P \cdot Q$. We shall use the symbol $\equiv$ to stand for syntactic equality of terms. For every natural number n, we shall write $[n]$ in lieu of $\{1, \ldots, n\}$.

Apart from actions, the signature of the language $\mathrm{BPA}^{me*}(A)$ includes the binary operations of alternative composition $+$ and sequential composition $\cdot$ familiar from the theory of Basic Process Algebra [6, 4], and a variation on the original binary version of the Kleene star operation [9], that will be referred to as multi-exit iteration. For positive integers m and n, the process term $(P_1, ..., P_m)^*(Q_1, ..., Q_n)$ stands for an agent whose behaviour is specified by the following defining equation:

$$(P_1, ..., P_m)^*(Q_1, ..., Q_n) \;=\; P_1 \cdot (P_2, ..., P_m, P_1)^*(Q_2, ..., Q_n, Q_1) + Q_1 \;\;.$$

In order to simplify notation in the presentation of the operational semantics for $\mathrm{BPA}^{me*}(A)$, we shall use the notion of 'vectors of processes'. A vector of processes is a tuple $(P_1, ..., P_m)$, where $m \geq 0$. We shall use $\vec{Q}, \vec{S}$ to denote such vectors of processes. In multi-exit iteration, the expressions at the left- and right-hand sides of the star are non-empty vectors of processes. Enclosing parentheses will always be omitted from vectors of length one, i.e., (P) will be written P.

The operational semantics for the language $\mathrm{BPA}^{me*}(A)$ is given by the labelled transition system

$$\left(\mathrm{BPA}^{me*}(A), \left\{\stackrel{a}{\rightarrow} \mid a \in A\right\}, \left\{\stackrel{a}{\rightarrow}\surd \mid a \in A\right\}\right) \;\;,$$

where the transition relations $\stackrel{a}{\rightarrow}$ and the unary predicates $\stackrel{a}{\rightarrow}\surd$ are, respectively, the least subsets of $\mathrm{BPA}^{me*}(A) \times \mathrm{BPA}^{me*}(A)$ and $\mathrm{BPA}^{me*}(A)$ satisfying the rules in Table 1. Intuitively, a transition $P \stackrel{a}{\rightarrow} Q$ means that the system represented by the term P can perform the action a, thereby evolving into Q. The special symbol $\surd$ stands for (successful) termination; therefore the interpretation of the statement $P \stackrel{a}{\rightarrow}\surd$ is that the process term P can terminate by performing a. Note that, for every term P, there is some action a for which either $P \stackrel{a}{\rightarrow} P'$ holds for some P', or $P \stackrel{a}{\rightarrow}\surd$ does.

Definition 1.1 *The term P' is a* derivative *of P if P can evolve into P' by zero or more transitions. A derivative P' of P is* proper *if P can evolve into P' by performing at least one transition.*

$$\frac{}{a \xrightarrow{a} \surd}$$

$$\frac{P \xrightarrow{a} \surd}{P+Q \xrightarrow{a} \surd} \qquad \frac{Q \xrightarrow{a} \surd}{P+Q \xrightarrow{a} \surd} \qquad \frac{P \xrightarrow{a} P'}{P+Q \xrightarrow{a} P'} \qquad \frac{Q \xrightarrow{a} Q'}{P+Q \xrightarrow{a} Q'}$$

$$\frac{P \xrightarrow{a} \surd}{P \cdot Q \xrightarrow{a} Q} \qquad \frac{P \xrightarrow{a} P'}{P \cdot Q \xrightarrow{a} P' \cdot Q}$$

$$\frac{P \xrightarrow{a} \surd}{(P,\vec{Q})^*(R,\vec{S}) \xrightarrow{a} (\vec{Q},P)^*(\vec{S},R)} \qquad \frac{P \xrightarrow{a} P'}{(P,\vec{Q})^*(R,\vec{S}) \xrightarrow{a} P' \cdot (\vec{Q},P)^*(\vec{S},R)}$$

$$\frac{R \xrightarrow{a} \surd}{(P,\vec{Q})^*(R,\vec{S}) \xrightarrow{a} \surd} \qquad \frac{R \xrightarrow{a} R'}{(P,\vec{Q})^*(R,\vec{S}) \xrightarrow{a} R'}$$

Table 1: Transition Rules

Process terms are considered modulo bisimulation equivalence [10].

Definition 1.2 *Two process terms P and Q are* bisimilar*, denoted by $P \underline{\leftrightarrow} Q$, if there exists a symmetric binary relation $\mathcal{B}$ on process terms which relates P and Q, such that:*

- *if $R \mathcal{B} S$ and $R \xrightarrow{a} R'$, then there is a transition $S \xrightarrow{a} S'$ such that $R' \mathcal{B} S'$,*
- *if $R \mathcal{B} S$ and $R \xrightarrow{a} \surd$, then $S \xrightarrow{a} \surd$.*

Such a relation $\mathcal{B}$ will be called a bisimulation*. The relation $\underline{\leftrightarrow}$ will be referred to as* bisimulation equivalence*.*

Note that if P is bisimilar to Q, then every (proper) derivative of P is bisimilar to some (proper) derivative of Q, and vice versa.

The transition rules in Table 1 are in the 'path' format of Baeten and Verhoef [3]. Hence, bisimulation equivalence is a congruence with respect to all the operations in the signature of $\mathrm{BPA}^{me*}(A)$.

Process terms in $\mathrm{BPA}^{me*}(A)$ are *normed*, which means that they are able to terminate by embarking in a finite sequence of transitions. We call such a sequence a termination trace. The *norm* of a process term P, denoted by $|P|$, is the length of its shortest termination trace; this notion stems from [2]. Note that bisimilar process terms have the same norm. The following lemma, which is due to Caucal [8], is typical for normed processes, and will be useful in the technical developments to follow.

Lemma 1.3 *Let $P, Q, R, S \in \mathrm{BPA}^{me*}(A)$ be such that $PQ \underline{\leftrightarrow} RS$. If $|Q| = |S|$, then $P \underline{\leftrightarrow} R$ and $Q \underline{\leftrightarrow} S$.*

A technical tool we shall use below is a weight function g that associates a natural number to each process term. This is defined thus:

$$
\begin{array}{rcl}
g(a) & \triangleq & 0 \\
g(P+Q) & \triangleq & \max\{g(P), g(Q)\} + 1 \\
g(PQ) & \triangleq & \max\{g(P), g(Q)\} \\
g\big((P_1, ..., P_m)^*(Q_1, ..., Q_n)\big) & \triangleq & \max\{g(P_i), g(Q_j) + 1 \mid i \in [m], j \in [n]\} \ .
\end{array}
$$

The basic property of this weight function that we shall need is expressed in the lemma below (cf. [1, Lemma 3.5]).

Lemma 1.4 *If P' is a derivative of P, then $g(P') \leq g(P)$. Moreover, if*

- *$P \equiv P_1 + P_2$ for some terms P_1 and P_2, and P' is a proper derivative of P, or*

- *$P \equiv (P_1, ..., P_m)^*(Q_1, ..., Q_n)$, for some terms P_i ($i \in [m]$) and Q_j ($j \in [n]$), and P' is a proper derivative of some Q_j,*

then $g(P') < g(P)$.

2 An Expressiveness Hierarchy

As shown in [5], the addition of multi-exit iteration to BPA yields a language that, modulo bisimulation equivalence, is strictly more expressive than that obtained by augmenting BPA with the standard binary Kleene star. More precisely, it is proven *ibidem* that, in the presence of at least two actions, the process $(a, a)^*(a, b)$ cannot be expressed, modulo bisimulation equivalence, in ACP [4], and *a fortiori* in BPA, enriched with the binary Kleene star (cf. Lemma 3.2.3 in *op. cit.*).

Let us say that a term of the form $(P_1, \ldots, P_m)^*(Q_1, \ldots, Q_n)$ has n-exit iteration. By analogy with the aforementioned result from [5], it was proved in [1] that, in the presence of a *non-empty* set of actions, the sequence of k-exit iteration operations induces a hierarchy of super-languages of BPA with a strictly increasing expressive power modulo bisimulation equivalence. To this end, it was shown in *op. cit.* that, for every positive integer k, the process

$$a^*(a, a^2, \ldots, a^{k+1})$$

cannot be specified using h-exit iteration with $h \leq k$, modulo bisimulation equivalence. (Cf. Corollary 4.5 in [1].)

In light of the above result, increasing the maximum number of exits allowed in a multi-exit iteration increases the expressive power of the language modulo bisimulation equivalence. Our aim in this note is to show that increasing the maximum number of processes on the left-hand side of the star in a multi-exit iteration also increases the expressive power of the language modulo bisimulation equivalence.

Notation 1 *For every positive integer k, we write* BPA^{k*} *for the set of terms in the language* $\mathrm{BPA}^{me*}(A)$ *that may use multi-exit iteration operations whose first argument is a non-empty vector of processes of length at most k.*

For a positive integer i and action a, we write a^i for the term obtained by concatenating i copies of action a.

Our aim is to prove the following theorem:

Theorem 2.1 *For every positive integer k, the process $(a, a^2, \ldots, a^{k+1})^* a$ cannot be expressed in the language* BPA^{k*} *modulo bisimulation equivalence.*

The remainder of this note will be devoted to a proof of the above result. To this end, it is sufficient to establish the following special case of the statement of our main result.

Proposition 2.2 *For every positive integer k, the process $(a, a^2, \ldots, a^{k+1})^* a$ cannot be expressed, modulo bisimulation equivalence, as a term in the language* BPA^{k*} *of the form $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m)$ with $|Q_j| = 1$, for every $j \in [m]$.*

Indeed, using the above result, we can prove Theorem 2.1 thus:

Proof of Theorem 2.1: Assume, towards a contradiction, that there is a term P in the language BPA^{k*} that is bisimilar to $(a, a^2, \ldots, a^{k+1})^* a$. Assume, furthermore, that P is a process with this property with minimum weight $g(P)$. We proceed with the proof by analyzing the possible forms such a P may take.

It is easy to see that P can neither have the form a nor the form $P_1 P_2$ for some processes P_1 and P_2. Indeed, this follows because bisimilar processes have equal norm, but any process of the form $P_1 P_2$ has norm at least two and $(a, a^2, \ldots, a^{k+1})^* a$ has norm one.

We claim that P cannot have the form $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m)$ either. To see this, note, first of all, that, by Proposition 2.2, P cannot have the form $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m)$, with $|Q_j| = 1$ for every $j \in [m]$. If there is some Q_j ($j \in [m]$) whose norm is greater than one, then this Q_j affords a transition $Q_j \xrightarrow{a} Q'_j$ for some process Q'_j. It follows that, for some positive integer ℓ,

$$(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m) \xrightarrow{a^\ell} Q'_j \ .$$

Since the terms $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m)$ and $(a, a^2, \ldots, a^{k+1})^* a$ are bisimilar, there is a derivative R of the latter term such that

$$(a, a^2, \ldots, a^{k+1})^* a \xrightarrow{a^\ell} R \quad \text{and} \quad Q'_j \;\underline{\leftrightarrow}\; R \ .$$

As $(a, a^2, \ldots, a^{k+1})^* a$ is easily seen to be a derivative of R, we have that Q'_j has a derivative that is bisimilar to $(a, a^2, \ldots, a^{k+1})^* a$, and thus that Q_j has a *proper* derivative Q' that is bisimilar to $(a, a^2, \ldots, a^{k+1})^* a$. By Lemma 1.4, the value of $g(Q')$ is strictly smaller than $g(P)$. This contradicts our assumption that P was a process with minimum weight in BPA^{k*} that is bisimilar to $(a, a^2, \ldots, a^{k+1})^* a$.

From the above reasoning, it follows that P can only have the form $P_1 + P_2$ for some processes P_1 and P_2. Since

$$(a, a^2, \ldots, a^{k+1})^* a \xrightarrow{a^n} (a, a^2, \ldots, a^{k+1})^* a \qquad \left(n = \frac{(k+1)(k+2)}{2}\right)$$

and $P \equiv P_1 + P_2$ is bisimilar to $(a, a^2, \ldots, a^{k+1})^* a$, there is a process P' such that

$$P \xrightarrow{a^n} P' \qquad \text{and} \qquad P' \underline{\leftrightarrow} (a, a^2, \ldots, a^{k+1})^* a \ .$$

By Lemma 1.4, since P' is a proper derivative of $P \equiv P_1 + P_2$, the value of $g(P')$ is strictly smaller than $g(P)$. This contradicts our assumption that P was a process with minimum weight in BPA^{k*} that is bisimilar to $(a, a^2, \ldots, a^{k+1})^* a$.

It follows that no term in BPA^{k*} can be bisimilar to $(a, a^2, \ldots, a^{k+1})^* a$, which was to be shown. □

To complete the proof, we are therefore left to show Proposition 2.2. This result is an immediate consequence of the second statement in the following lemma.

Lemma 2.3 *Assume that $Q_1, \ldots, Q_m$ are processes with norm one. Then the following statements hold:*

1. *For every positive integer i, if $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m)$ is bisimilar to $(a^i, R_1, \ldots, R_n)^* a$ $(n \geq 0)$, then*
   - *$P_1 \underline{\leftrightarrow} a^i$ and*
   - *$(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1) \underline{\leftrightarrow} (R_1, \ldots, R_n, a^i)^* a$.*
2. *For every $k \geq h \geq 1$, if $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m) \underline{\leftrightarrow} (a, a^2, \ldots, a^k)^* a$, then $h = k$, and $P_i \underline{\leftrightarrow} a^i$ for every $i \in [k]$.*

Proof: We prove the two statements separately.

- Proof of Statement 1. We consider two cases, depending on whether $i = 1$ or not. In both cases of the proof, we use the fact that, as $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m) \underline{\leftrightarrow} (a^i, R_1, \ldots, R_n)^* a$ holds by assumption, P_1 can perform an a-labelled transition and no transition labelled with actions different from a.

  Assume that $i = 1$. Then P_1 has no transitions of the form $P_1 \xrightarrow{a} P_1'$. Indeed, if $P_1 \xrightarrow{a} P_1'$ holds, then so does

  $$(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m) \xrightarrow{a} P_1'(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1) \ . \quad (1)$$

  Since $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m) \underline{\leftrightarrow} (a, R_1, \ldots, R_n)^* a$ holds by assumption, there is a transition

  $$(a, R_1, \ldots, R_n)^* a \xrightarrow{a} R$$

  for some R such that

  $$P_1'(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1) \underline{\leftrightarrow} R \ .$$

The only candidate for this R is the term $(R_1, \ldots, R_n, a)^*a$. However, the term $(R_1, \ldots, R_n, a)^*a$ has norm one, whereas

$$|P_1'(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1)| \geq 2 \ .$$

It follows that $P_1'(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1)$ cannot be bisimilar to $(R_1, \ldots, R_n, a)^*a$, and thus that $P_1 \xrightarrow{a} \checkmark$ is the only transition afforded by P_1. We can now conclude that

– $P_1 \underline{\leftrightarrow} a$ and
– $(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1) \underline{\leftrightarrow} (R_1, \ldots, R_n, a)^*a$

both hold, which was to be shown.

Assume now that i is greater than 1. Reasoning as in the previous case, it is not hard to see that P_1 only affords transitions of the form $P_1 \xrightarrow{a} P_1'$. For every such transition, we have a transition of the form (1) out of $(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m)$. These transitions can only be matched by the transition

$$(a^i, R_1, \ldots, R_n)^*a \xrightarrow{a} a^{i-1}(R_1, \ldots, R_n, a^i)^*a$$

from $(a^i, R_1, \ldots, R_n)^*a$. It follows that, for every term P_1' such that $P_1 \xrightarrow{a} P_1'$, it holds that

$$P_1'(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1) \underline{\leftrightarrow} a^{i-1}(R_1, \ldots, R_n, a^i)^*a \ .$$

Since the terms $(P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1)$ and $(R_1, \ldots, R_n, a_i)^*a$ have both norm one by the proviso of the lemma, Lemma 1.3 yields that

$$\begin{array}{rcl} P_1' & \underline{\leftrightarrow} & a^{i-1} \quad \text{and} \\ (P_2, \ldots, P_h, P_1)^*(Q_2, \ldots, Q_m, Q_1) & \underline{\leftrightarrow} & (R_1, \ldots, R_n, a^i)^*a \ . \end{array}$$

To complete the proof for this case, note that since every term that can be reached from P_1 via an a-labelled transition is bisimilar to a^{i-1}, from our previous observations it follows that P_1 is bisimilar to a^i.

- Proof of Statement 2. Assume that $k \geq h \geq 1$ and

$$(P_1, \ldots, P_h)^*(Q_1, \ldots, Q_m) \underline{\leftrightarrow} (a, a^2, \ldots, a^k)^*a \ .$$

Using statement 1 of the lemma repeatedly, we have that $P_i \underline{\leftrightarrow} a^i$ for every $i \in [h]$, and

$$(P_1, \ldots, P_h)^*(Q_{\ell+1}, \ldots, Q_m, Q_1, \ldots, Q_\ell) \underline{\leftrightarrow} (a^{h+1}, \ldots, a^k, a_1, \ldots, a^h)^*a \ ,$$

where $\ell = h \mod m$.

If $h < k$, then statement 1 of the lemma would entail that

$$a \underline{\leftrightarrow} P_1 \underline{\leftrightarrow} a^{h+1} \ ,$$

which is impossible because $a \not\underline{\leftrightarrow} a^{h+1}$, as $h \geq 1$. It follows that $h = k$ holds, and we are done.

This completes the proof of the lemma. □

Acknowledgements: The research reported in this note originated from a question posed to the authors by Kim G. Larsen.

References

[1] L. Aceto and W. J. Fokkink. An Equational Axiomatization for Multi-exit Iteration. *Information and Computation*, 137(2):121–158, 15 September 1997.

[2] J. Baeten, J. Bergstra, and J. Klop, *Decidability of bisimulation equivalence for processes generating context-free languages*, J. Assoc. Comput. Mach., 40 (1993), pp. 653–682.

[3] J. Baeten and C. Verhoef, *A congruence theorem for structured operational semantics*, in Best [7], pp. 477–492.

[4] J. Baeten and W. Weijland, *Process Algebra*, Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, 1990.

[5] J. Bergstra, I. Bethke, and A. Ponse, *Process algebra with iteration*, Report CS-R9314, Programming Research Group, University of Amsterdam, 1993.

[6] J. Bergstra and J. Klop, *Fixed point semantics in process algebras*, Report IW 206, Mathematisch Centrum, Amsterdam, 1982.

[7] E. Best, ed., *Proceedings CONCUR 93,* Hildesheim, Germany, vol. 715 of Lecture Notes in Computer Science, Springer-Verlag, 1993.

[8] D. Caucal, *Graphes canoniques de graphes algébriques*, Theoretical Informatics and Applications, 24 (1990), pp. 339–352.

[9] S. Kleene, *Representation of events in nerve nets and finite automata*, in Automata Studies, C. Shannon and J. McCarthy, eds., Princeton University Press, 1956, pp. 3–41.

[10] D. Park, *Concurrency and automata on infinite sequences*, in 5^{th} GI Conference, Karlsruhe, Germany, P. Deussen, ed., vol. 104 of Lecture Notes in Computer Science, Springer-Verlag, 1981, pp. 167–183.

Recent BRICS Report Series Publications

RS-02-40 **Luca Aceto, Willem Jan Fokkink, and Anna Ingólfsdóttir.** ***A Note on an Expressiveness Hierarchy for Multi-exit Iteration*****. September 2002. 8 pp.**

RS-02-39 **Stephen L. Bloom and Zoltán Ésik.** ***Some Remarks on Regular Words*****. September 2002.**

RS-02-38 **Daniele Varacca.** ***The Powerdomain of Indexed Valuations*****. September 2002. 54 pp. Short version appears in Plotkin, editor,** ***Seventeenth Annual IEEE Symposium on Logic in Computer Science*****, LICS '02 Proceedings, 2002, pages 299–308.**

RS-02-37 **Mads Sig Ager, Olivier Danvy, and Mayer Goldberg.** ***A Symmetric Approach to Compilation and Decompilation*****. August 2002. To appear in Neil Jones's Festschrift.**

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.**