

ISRN SICS-R--94/20--SE

# **On the Decidability of Process Equivalences for the $\pi$ -calculus**

by

**Mads Dam**

# On the Decidability of Process Equivalences for the $\pi$ -calculus

Mads Dam<sup>1</sup>  
Swedish Institute of Computer Science

November 14, 1994

SICS Research Report RR:94-20

<sup>1</sup>Work supported by ESPRIT BRA project 8130 "LOMAPS"



## Abstract

We present general results for showing process equivalences applied to the finite control fragment of the  $\pi$ -calculus decidable. Firstly a Finite Reachability Theorem states that up to finite name spaces and up to a static normalisation procedure, the set of reachable agent expressions is finite. Secondly a Boundedness Lemma shows that no potential computations are missed when name spaces are chosen large enough, but finite. We show how these results lead to decidability for a number of  $\pi$ -calculus equivalences such as strong or weak, late or early bismulation equivalence. Furthermore, for strong late equivalence we show how our techniques can be used to adapt the well-known Paige-Tarjan algorithm. Strikingly this results in a single exponential running time not much worse than the running time for the case of for instance CCS. Our results considerably strengthens previous results on decidable equivalences for parameter-passing process calculi.



# 1 Introduction

The problem of obtaining a unified view of on the one hand sequential computation as embodied by the  $\lambda$ -calculus, and reactive systems such as CCS or CSP on the other has recently had considerable attention. The  $\pi$ -calculus [11] was proposed as a calculus for mobile processes, i.e. processes whose interconnection topology may be dynamically changed. It extends CCS by features for the transmission and generation of channel names. Considerable expressive power is gained by this. For instance, data types [9], lambda calculus [10], object-oriented programming languages [18], and higher-order processes [16] can all be captured, underlining the foundational importance of the calculus. Moreover the practical usefulness of the calculus have been demonstrated in application studies on mobile telecommunication networks and high speed networks [14, 13]. It is therefore important to investigate to what extent methods and tools developed for, say, CCS lift to the more expressive setting of the  $\pi$ -calculus.

One such set of tools of fundamental importance are process equivalence checking algorithms, as exemplified by the Paige-Tarjan algorithm [15, 8]. Algorithms like these apply in general only to finite-state processes, characterised, in the case of CCS, by disallowing occurrences of the parallel combinator as well as unguarded occurrences of process identifiers in recursive definitions. The corresponding fragment of the  $\pi$ -calculus is termed the *finite control* fragment. In this paper we show that for a range of equivalences the finite control conditions are in fact sufficient to lift algorithms to the  $\pi$ -calculus. This is far from a trivial result, since even very simple  $\pi$ -calculus agents exhibit infinite-state behaviour while satisfying these conditions. One example, using a CCS-like notation, is the memory cell

$$Mem(x) = in(y).Mem(y) + \overline{out}x.Mem(x)$$

that can either input a channel name  $y$  along channel  $in$  and then proceed as  $Mem(y)$  or else output  $x$  along  $out$  and then proceed as  $Mem(x)$ . This is an example of a *data-independent* agent such as those considered previously by Jonsson and Parrow [7]. However, the finite-control fragment goes beyond this, since it allows synchronisation, or testing, on channel names passed as parameters. By adding a positive and negative conditional <sup>1</sup> **if  $x = y$  then  $P$  else  $Q$**  to the  $\pi$ -calculus, as we do, we can for instance encode the memory cell

$$KillableMem(x) = \\ in(y).(\mathbf{if } y = KILL \mathbf{ then } NIL \mathbf{ else } Mem(y)) + \overline{out}x.Mem(x)$$

that is killed in case the channel  $KILL$  is passed to it. Other examples concern the facility of the  $\pi$ -calculus to declare new private names and to pass them on to other parallel components. Consider for instance the agent

$$(\nu x)(Gen(x)|Listen(x))$$

---

<sup>1</sup>In the paper we actually use the notation  $[x = y]PQ$  instead of **if  $x = y$  then  $P$  else  $Q$**  for the conditional.

where

$$\begin{aligned} Gen(x) &= (\nu y)\bar{x}y.Gen(y) \\ Listen(x) &= x(y).Listen(y) \end{aligned}$$

In this system  $Gen$  repeatedly declares a new channel name  $y$ , transmits  $y$  to  $Listen$  along  $x$  and then proceeds as  $Gen(y)$ . Since the  $y$ 's are known to be fresh and thus different from any other name previously encountered during a computation, the state space generated by  $(\nu x)(Gen(x)|Listen(x))$  is infinite.

In this paper we provide the basic tools to show decidability for the finite control fragment for a number of equivalences, including late or early, strong or weak bisimulation equivalence (c.f. [12]), and open (or uniform [1]) bisimulation equivalence [17]<sup>2</sup>. Though the technical details differ, the basic approach we use to show decidability is quite familiar from other recent decidability results in process algebra such as [2, 3, 6, 7], namely by showing that up to the equivalence concerned the state space can be represented in a finitary manner. The tools consist of two key Lemmas, proofs of which are given in the paper:

1. A Finite Reachability Theorem showing that up to a finite name space and up to a deterministic static normalisation procedure only a finite number of distinct agents are reachable.
2. A proof that the number of distinct free names needed at any point during a computation can be bounded.

Put together these results imply that a bound can be put on the size of the name space, and decidability for a given process equivalence  $\equiv$  then consists of showing for all agents  $A$  and  $B$ , that  $A \equiv B$  iff  $A\sigma \equiv_N B\sigma$  where  $\equiv_N$  represents equivalence with respect to a large enough but finite name space  $N$ , and  $\sigma$  is a map representing names as names in  $N$ . We establish this result for all the equivalences mentioned above with particular focus on strong late bisimulation equivalence [9]. For this equivalence we show how the partition refinement algorithm of Paige and Tarjan [15] can be applied resulting, as for the case of e.g. CCS, in a single exponential worst case complexity.

Our results very considerably strengthens previous results in the area of value-passing process calculi. Besides the decidability result of Jonsson and Parrow [7] for data-independent programs, Hennessy and Lin [5] showed decidability of bisimulation equivalence for a certain class of symbolic transition graphs. Both these results are subsumed by the work presented here. The notion of open bisimulation equivalence was specifically formulated with an eye on efficiency concerns. In the area of model checking a close relative to the present work is the decidability result with respect to an extended version of the modal  $\mu$ -calculus of [4].

---

<sup>2</sup>For open bisimulation equivalence decidability is already known [17]

## 2 The Polyadic $\pi$ -calculus, syntax

We use a slight extension of Milner's polyadic  $\pi$ -calculus, introduced in [4]. Letters  $x, y, z, \dots$  range over *names* of which there is a countably infinite supply,  $A, B$  range over *agents*, and  $D$  over *agent identifiers*. *Actions*,  $\alpha, \beta$ , are either names, co-names of the form  $\bar{x}$ , or the distinguished constant  $\tau$ . If  $\alpha$  is a name  $x$  then  $n(\alpha)$  (the name of  $\alpha$ ) is  $x$ , and  $p(\alpha)$  (the polarity of  $\alpha$ ) is  $-$ . Otherwise if  $\alpha = \bar{x}$  then  $n(\alpha) = x$  and  $p(\alpha) = +$ . The syntax of agents is the following:

$$A ::= \mathbf{0} \mid A + A \mid \alpha.A \mid A \mid A \mid [x = y]AA \mid (\lambda x)A \mid Ax \mid (\nu x)A \mid D \mid \text{fix}D.A \mid [x]A$$

Name binders are the operators  $\lambda$  and  $\nu$ . We use the notation  $\text{fn}(A)$  for the set of names occurring freely in  $A$  and  $A\{x/y\}$  ( $A\{A'/D\}$ ) for substitution of  $x$  ( $A'$ ) for  $y$  ( $D$ ) in  $A$ . The intended meaning of connectives is familiar from CCS and the  $\pi$ -calculus. The present version is based on the polyadic  $\pi$ -calculus of [9]. There are three main differences:

**Recursion.** We use recursive definitions rather than replication. Just as for CCS we require restriction to those expressions that are well-guarded (in the sense of, for an expression  $\text{fix}D.A$ , only allowing free occurrences of  $D$  in  $A$  within the scope of a prefix operator, and for which uses of the parallel combinator  $\mid$  within recursive definitions are disallowed. We refer to this fragment as the *finite control* fragment. In addition we require for technical reasons that recursions  $\text{fix}D.A$  are *fully parametrised* in the sense that recursive agents  $\text{fix}D.A$  have no free occurrences of names. The expressive power of the language is unaffected by this latter restriction since all equivalences considered here will respect the identification of  $(\text{fix}D.A)(x_1, \dots, x_n)$  with

$$(\text{fix}D.(\lambda x_1) \cdots (\lambda x_n)A\{Dx_1 \cdots x_n/D\})x_1 \cdots x_n.$$

**Conditionals.** We admit the conditional  $[x = y]AB$ , identified with  $A$  when  $x = y$ , and  $B$  when  $x \neq y$ . The admission of negative as well as positive matching has been an issue of some controversy in the  $\pi$ -calculus (c.f. [17]). It is accommodated (though not required) in our framework by a relativisation of the operational semantics to complete descriptions of name identities and inequalities.

**Well-formedness.** A well-formedness condition is imposed, reflecting the stratified syntax of [9]. Agents  $A$  that are to be considered *well-formed* are assigned an integer *arity*  $n$ , written  $A : n$ . *Processes* are agents of arity 0, *abstractions* are agents of negative arity, and *concretions* are agents of positive arity. Given an assignment  $D : n$  of arities to agent identifiers, agent arities are computed as follows:

$$\frac{}{\mathbf{0} : 0} \quad \frac{A : 0 \quad B : 0}{A + B : 0} \quad \frac{A : n \quad n \leq 0}{x.A : 0} \quad \frac{A : n \quad n \geq 0}{\bar{x}.A : 0} \quad \frac{A : 0}{\tau.A : 0}$$

$$\frac{A : 0 \quad B : 0}{A \mid B : 0} \quad \frac{A : n \quad B : n}{[x = y]AB : n} \quad \frac{A : n \quad n \leq 0}{(\lambda x)A : n - 1} \quad \frac{A : n - 1 \quad n \leq 0}{Ax : n}$$

$$\frac{A : n}{(\nu x)A : n} \quad \frac{D : n \quad A : n}{\text{fix}D.A : n} \quad \frac{A : n \quad n \geq 0}{[x]A : n + 1}$$

For the remainder of the paper we restrict attention to well-formed agents.

### 3 Operational Semantics

In [9] the semantics of the  $\pi$ -calculus is given in terms of a structural congruence relation together with a relation of commitment. Here we choose a different, more operational approach, replacing the structural congruence relation with a normalisation procedure.

**Name partitionings.** Since the decision procedure handles names in a symbolic fashion, normalisation needs to know what identities and inequalities are assumed of names. This information is supplied by partitionings  $\varepsilon$  on the set of names. A partitioning  $\varepsilon$  identifies the names  $x$  and  $y$  (written  $\varepsilon \models x = y$ ) if and only if  $x$  and  $y$  are members of the same element of  $\varepsilon$ . The operation  $(\nu x)\varepsilon$  of *name generation* is defined by

$$(\nu x)\varepsilon = \{S - \{x\} \mid S \in \varepsilon\} \cup \{\{x\}\}.$$

**Normal forms and normalisation.** Processes in normal form, ranged over by  $P$ , are generated by the grammar

$$P ::= \mathbf{0} \mid P + P \mid \alpha.A \mid P \mid P \mid (\nu x)P$$

Abstractions in normal form have the form  $(\lambda x)A$ , and concretions in normal form have one of the forms  $[x]A$  or  $(\nu x)[x]A$ . The normalisation procedure is given by the pseudo-ML function **nf**:

```

fun nf(0,  $\varepsilon$ ) = 0 |
  nf( $A + B$ ,  $\varepsilon$ ) = nf( $A$ ,  $\varepsilon$ ) + nf( $B$ ,  $\varepsilon$ ) |
  nf( $\alpha.A$ ,  $\varepsilon$ ) =  $\alpha.A$  |
  nf( $A \mid B$ ,  $\varepsilon$ ) = (nf( $A$ ,  $\varepsilon$ ) | nf( $B$ ,  $\varepsilon$ )) |
  nf( $[x = y]AB$ ,  $\varepsilon$ ) = if  $\varepsilon \models x = y$  then nf( $A$ ,  $\varepsilon$ ) else nf( $B$ ,  $\varepsilon$ ) |
  nf( $(\lambda x)A$ ,  $\varepsilon$ ) =  $(\lambda x)A$  |
  nf( $Ax$ ,  $\varepsilon$ ) = (case nf( $A$ ,  $\varepsilon$ ) of  $(\lambda y)A_1 \Rightarrow$  nf( $A_1\{x/y\}$ ,  $\varepsilon$ )) |
  nf( $(\nu x)A$ ,  $\varepsilon$ ) =
    let  $A_1 =$  nf( $A$ ,  $(\nu x)\varepsilon$ ) in
    if  $x \in \text{fn}(A_1)$ 
    then if  $A_1 : 0$  then  $(\nu x)A_1$  else
      (case  $A_1$  of
         $(\lambda y)A_2 \Rightarrow$  if  $x = y$  then  $(\lambda y)A_2$  else  $(\lambda y)(\nu x)A_2$  |

```

$$\begin{aligned}
& [y]A_2 \Rightarrow \text{if } x = y \text{ then } (\nu x)[y]A_2 \text{ else } [y](\nu x)A_2 \mid \\
& (\nu y)[y]A_2 \Rightarrow \text{if } x = y \text{ then } (\nu y)[y]A_2 \text{ else } (\nu y)[y](\nu x)A_2 \\
& \text{else } A_1 \\
& \text{end } \mid \\
\text{nf}(\text{fix}D.A, \varepsilon) &= \text{nf}(A\{\text{fix}D.A/D\}, \varepsilon) \mid \\
\text{nf}([x]A, \varepsilon) &= [x]A
\end{aligned}$$

The following points are easily verified:

1.  $\text{nf}$  preserves well-formedness.
2.  $\text{nf}$  is total (for agents without free occurrences of agent identifiers).
3. For all well-formed  $A$  and  $\varepsilon$ ,  $\text{nf}(A, \varepsilon)$  is in normal form.

Restricting to well-formed agents in the conditional-free fragment it is possible to compare the normalisation procedure with the structural congruence relation  $\equiv$  of [9]. It is quite easy to show, appealing to [9] for the definition of  $\equiv$ , that for all well-formed, conditional-free agents  $A$ ,  $\text{nf}(A, \varepsilon)$  is independent of  $\varepsilon$ , and for all  $\varepsilon$ ,  $A \equiv \text{nf}(A, \varepsilon)$ .

**Commitment.** The definition of the commitment relation needs the ancillary operations  $\parallel$  and  $\cdot$  on normal forms:

- $A \parallel B = A \mid B$  when  $A : 0$  and  $B : 0$ . If  $A : 0$  and  $B : n \neq 0$  then  $A \parallel [x]B' = [x](A \parallel B')$ ,  $A \parallel (\nu x)[x]B' = (\nu y)[y](A \parallel B'\{y/x\})$ , and  $A \parallel (\lambda x)B' = (\lambda y)(A \parallel B'\{y/x\})$ , where in the two last cases it is assumed that  $y \notin \text{fn}(A)$ . The case for  $A : n \neq 0$  and  $B : 0$  is defined symmetrically.
- $A \cdot B$  is defined only when  $A : -n$  and  $B : n$  for some (positive or negative)  $n$ . For  $n = 0$ ,  $A \cdot B = A \mid B$ . If  $n > 0$ ,  $(\lambda x)A' \cdot [y]B' = A\{y/x\} \cdot B'$  and  $(\lambda x)A' \cdot (\nu y)[y]B' = A\{z/x\} \cdot B'\{z/y\}$  where  $z \notin (\text{fn}(A') - \{x\}) \cup (\text{fn}(B') - \{y\})$ . The case for  $n < 0$  is defined symmetrically.

As the normalisation procedure the commitment relation is relativised to name partitions too. It is defined as follows:

$$\begin{aligned}
\text{ACT: } & \frac{}{\alpha.A \succ_\varepsilon \alpha.A} & \text{SUM: } & \frac{A_1 \succ_\varepsilon B}{A_1 + A_2 \succ_\varepsilon B} \\
\text{COMM: } & \frac{A_1 \succ_\varepsilon x.B_1 \quad A_2 \succ_\varepsilon \bar{y}.B_2 \quad \varepsilon \models x = y}{A_1 \mid A_2 \succ_\varepsilon \tau.(\text{nf}(B_1, \varepsilon) \cdot \text{nf}(B_2, \varepsilon))} \\
\text{PAR: } & \frac{A_1 \succ_\varepsilon \alpha.B}{A_1 \mid A_2 \succ_\varepsilon \alpha.(\text{nf}(B, \varepsilon) \parallel A_2)} \\
\text{RES-1: } & \frac{A \succ_{(\nu x)\varepsilon} \tau.B}{(\nu x)A \succ_\varepsilon \tau.(\nu x)B} & \text{RES-2: } & \frac{A \succ_{(\nu x)\varepsilon} \alpha.B}{(\nu x)A \succ_\varepsilon \alpha.(\nu x)B} \quad (x \neq \text{n}(\alpha))
\end{aligned}$$

+ symmetrical versions of rules SUM, COMM and PAR

Relating to [9] let the *full* name partitioning  $\varepsilon_f$  be the one containing only singleton sets. The full partitioning identifies names only if they are literally the same. It can then be shown for the well-formed fragment without conditionals that  $A \succ B$  according to [9] if and only if for some  $B'$ ,  $\text{nf}(A, \varepsilon_f) \succ_{\varepsilon_f} B'$ , and  $\text{nf}(B, \varepsilon_f) = \text{nf}(B', \varepsilon_f)$ .

**Definition 3.1** (Simulations, Bisimulations) A (strong, late) *partition-relativised simulation* (or pr-simulation) is an  $\varepsilon$ -indexed family of binary relations  $R_\varepsilon$  on well-formed agents satisfying the following conditions:

1. If  $AR_\varepsilon B$  then  $\text{nf}(A, \varepsilon)R_\varepsilon\text{nf}(B, \varepsilon)$ .
2. If  $AR_\varepsilon B$  and  $A : n$  then  $B : n$ .
3. If  $[x]A'R_\varepsilon[y]B'$  then  $A'R_\varepsilon B'$  and  $\varepsilon \models x = y$ .
4. If  $(\nu x)[x]A'R_\varepsilon(\nu y)[y]B'$  then  $A'\{z/x\}R_{(\nu z)\varepsilon}B'\{z/y\}$  whenever  $z \notin (\text{fn}(A') - \{x\}) \cup (\text{fn}(B') - \{y\})$ .
5. If  $(\lambda x)A'R_\varepsilon(\lambda y)B'$  then for all  $z$ ,  $A'\{z/x\}R_\varepsilon B'\{z/y\}$ .
6. If  $AR_\varepsilon B$  and  $A \succ_\varepsilon \alpha.A'$  then  $B \succ_\varepsilon \beta.B'$  for some  $B'$  such that  $\varepsilon \models \alpha = \beta$  and  $A'R_\varepsilon B'$ .

Then  $R$  is a *partition-relativised bisimulation* (pr-bisimulation) if for each  $\varepsilon$  both  $R_\varepsilon$  and  $R_\varepsilon^{-1}$  are pr-simulations;  $A$  and  $B$  are  $\varepsilon$ -bisimilar ( $A \sim_\varepsilon B$ ) if there is a pr-bisimulation  $R$  such that  $AR_\varepsilon B$ ; and  $A$  and  $B$  are pr-bisimilar ( $A \sim B$ ) if there is a pr-bisimulation  $R$  such that  $AR_\varepsilon B$  for all  $\varepsilon$ .

The following is stated without proof:

**Proposition 3.2** *For the fragment of well-formed, conditional-free agents,  $\sim_{\varepsilon_f}$  is the (strong) bisimulation equivalence of [9], and  $\sim$  is strong congruence.*  $\square$

## 4 A Finite Reachability Theorem

The main ingredient in the decidability proof is a finite reachability Theorem, showing that for finite control agents, if names are always chosen from a fixed finite number of candidates then only a finite number of distinct agent expressions are reachable.

**Small enough names.** Names are chosen at the following points:

- When computing  $A \parallel B$  or  $A \cdot B$ .
- When instantiating names bound by  $\lambda$  or  $\nu$ .

We restrict these choices by assuming an enumeration  $x_0, x_1, \dots$  of names and imposing a maximal index  $n_0$  such that whenever a name is to be chosen then it is chosen *small enough*, i.e. with an index not exceeding  $n_0$ . If no such name exists (because otherwise confusion of names would ensue) then the result is left undefined. We show later that by picking  $n_0$  large enough all choices can in fact be made.

**Definition 4.1** (Reachability relation) Relative to a choice of  $n_0$  the reachability relation  $\rightsquigarrow$  on well-formed agents is defined as follows:

1. For all  $\varepsilon$ ,  $A \rightsquigarrow \mathbf{nf}(A, \varepsilon)$ ,
2.  $[x]A \rightsquigarrow A$ ,
3.  $(\nu x)[x]A \rightsquigarrow A\{y/x\}$  whenever  $y$  is small enough and  $y \notin \text{fn}(A) - \{x\}$ ,
4.  $(\lambda x)A \rightsquigarrow A\{y/x\}$  whenever  $y$  is small enough,
5. If  $P$  is a process in normal form and for some  $\varepsilon$ ,  $P \succ_\varepsilon \alpha.A$  while choosing only names that are small enough, then  $P \rightsquigarrow A$ .

**Theorem 4.2** (Finite reachability) *For all well-formed  $A$  and  $n_0$ ,  $\{B \mid A \rightsquigarrow^* B\}$  is finite.*

**PROOF** We define a reduction relation  $\rightarrow$  such that  $\rightarrow^*$  includes  $\rightsquigarrow^*$ , and such that we can prove  $\{B \mid A \rightarrow^* B\}$  finite. The relation  $\rightarrow$  is determined by the following closure properties:

0.  $A \rightarrow B$  whenever  $A$  and  $B$  are alpha-congruent, and  $B$  results from  $A$  by replacing small enough bound names with small enough bound names
1.  $A + B \rightarrow A$ ,  $A + B \rightarrow B$
2.  $\alpha.A \rightarrow A$
3.  $[x = y]AB \rightarrow A$ ,  $[x = y]AB \rightarrow B$
4.  $(\lambda x)A \rightarrow A\{y/x\}$  whenever  $y$  is small enough
5.  $Ax \rightarrow A$
6.  $\text{fix}D.A \rightarrow A\{\text{fix}D.A/D\}$
7.  $[x]A \rightarrow A$

8. If  $A \rightarrow B$  and  $x \in \text{fn}(B)$  then  $(\nu x)A \rightarrow (\nu x)B$
9.  $(\nu x)A \rightarrow A$
10.  $(\nu x)(\lambda y)A \rightarrow (\lambda y)(\nu x)A$
11. If  $x \neq y$  then  $(\nu x)[y]A \rightarrow [y](\nu x)A$
12.  $(\nu x)(\nu y)[y]A \rightarrow (\nu y)[y](\nu x)A$
13. If  $A \rightarrow A'$  then  $A \mid B \rightarrow A' \mid B$  and  $B \mid A \rightarrow B \mid A'$
14.  $((\lambda x)A) \mid B \rightarrow (\lambda x)(A \mid B)$ ,  $A \mid ((\lambda x)B) \rightarrow (\lambda x)(A \mid B)$
15.  $([x]A) \mid B \rightarrow [x](A \mid B)$ ,  $A \mid ([x]B) \rightarrow [x](A \mid B)$
16.  $((\nu x)A) \mid B \rightarrow (\nu x)(A \mid B)$ ,  $A \mid ((\nu x)B) \rightarrow (\nu x)(A \mid B)$

**Proposition 4.3**  $\rightsquigarrow^* \subseteq \rightarrow^*$ .

PROOF We need to show

- (i) For all  $\varepsilon, A \rightarrow^* \text{nf}(A, \varepsilon)$ .
- (ii)  $[x]A \rightarrow^* A$ .
- (iii)  $(\nu x)[x]A \rightarrow^* A\{y/x\}$  whenever  $y$  is small enough and  $y \notin \text{fn}(A) - \{x\}$ .
- (iv)  $(\lambda x)A \rightarrow^* A\{y/x\}$  whenever  $y$  is small enough.
- (v)  $P \rightarrow^* A$  whenever  $P$  is a process in normal form and for some  $\varepsilon$  and  $\alpha$ ,  $P \succ_\varepsilon \alpha.A$ .

Of these (i) and (v) use structural induction, and (ii)–(iv) follow directly from the conditions given.  $\square$  (Proposition 4.3)

We then proceed to prove finiteness:

**Lemma 4.4** (Finiteness) *For all  $A$ ,  $\{B \mid A \rightarrow^* B\}$  is finite.*

PROOF By König's Lemma, since well-guardedness ensures that  $\{B \mid A \rightarrow B\}$  is always finite, it suffices to show that any infinite derivation

$$d = A_0 \rightarrow \dots \rightarrow A_n \rightarrow \dots$$

with  $A_0 = A$  visits a finite number of distinct agents only, i.e.  $\mathcal{R}(d) = \{A_i \mid i \in \omega\}$  is finite. To show this we define the *size*,  $|A|$ , of  $A$  in the following manner:

$$\begin{aligned} |\mathbf{0}| &= |D| = 2 \\ |A + B| &= |[x = y]AB| = |A| + |B| + 1 \\ |\alpha.A| &= |(\lambda x)A| = |Ax| = |[x]A| = |\text{fix}D.A| = |A| + 1 \\ |(\nu x)A| &= 2 \cdot |A| + 1 \\ |A \mid B| &= |A| \cdot |B| \end{aligned}$$

**Lemma 4.5** *Axiom (0) does not increase size. All axioms among (1)–(16) except (6) decrease size. Rules (8), (11), (13) preserve size decrease*  $\square$ (Lemma 4.5)

We can assume that the unfolding axiom (6) is used infinitely often along  $d$ . By the finite control assumption each  $A_i$  will have the form  $A_i = C_i(B_{i,1}, \dots, B_{i,m})$  such that  $C_i$  is an  $m$ -ary context that does not contain occurrences of the fixed point operator, and for which each  $B_j$  has no occurrences of parallel composition operator. Moreover  $m$  will be independent of  $i$ . The number of occurrences in  $C_i$  of operators among  $+$ , prefixing, the conditional, or application will decrease with increasing  $i$  since the only reduction that can cause such occurrences to duplicate is axiom (6) which does not apply due to the finite control assumption. Moreover, for each occurrence of one of these operators, either it is never reduced, and then the subterm in question can be viewed as a constant, or else the number of occurrences of that particular operator in the  $C_i$  is reduced by 1. Thus there is no loss of generality in assuming that we can find some  $i_0$  for which  $C_{i_0}$  is an  $m$ 'ary context built using only operators of the form  $[x]$ ,  $(\lambda x)$ ,  $(\nu x)$ , or  $|$ .

Now, for all  $i \geq i_0$ ,  $A_i$  will have a similar form  $C_i(B_{i,1}, \dots, B_{i,m})$ , and for each  $j : 1 \leq j \leq m$ , either  $B_{i,j} = B_{i+1,j}$ , or else  $B_{i,j} \rightarrow B_{i+1,j}$ . In addition we can assume that for infinitely many  $i$ , does  $B_{i,j} \rightarrow B_{i+1,j}$ , since otherwise it suffices to pick a larger  $i_0$ . Thus the proof has been reduced to showing

- (i) only a finite number of distinct  $C_i$  are reachable
- (ii) any derivation  $d$  that does not involve parallel composition visits a finite number of distinct agents only.

**Contexts.** To prove (i) we introduce a transition system on contexts, and prove it finite. *Contexts* are terms  $C$  generated by the abstract syntax

$$C ::= [\cdot] \mid (\nu x)C \mid (\lambda x)C \mid [x]C \mid C \mid C$$

Here  $[\cdot]$  is the empty context. Say of a context  $C$  that  $x$  is *visible through*  $C$  if either there is some occurrence of  $[\cdot]$  in  $C$  not within the scope of a binding occurrence of  $x$ , or else  $x$  occurs unbound in  $C$ . Rule (6) below shows where this notion is needed. The transition relation  $\rightarrow$  is now determined in the following way where  $\Omega$  ranges over operators among  $(\nu x)$ ,  $(\lambda x)$ , and  $[x]$  with  $x$  small enough:

1. If  $C_1$  and  $C_2$  are alpha congruent then  $C_1 \rightarrow C_2$
2.  $[\cdot] \rightarrow \Omega[\cdot]$
3.  $(\Omega C_1) \mid C_2 \rightarrow \Omega(C_1 \mid C_2)$ ,  $C_1 \mid (\Omega C_2) \rightarrow \Omega(C_1 \mid C_2)$
4.  $[x]C \rightarrow C$ ,  $(\nu x)C \rightarrow C$ ,  $(\lambda x)C \rightarrow C\{y/x\}$  whenever  $y$  is small enough
5.  $(\nu x)\Omega C \rightarrow \Omega(\nu x)C$

6. if  $C_1 \rightarrow C'_1$  and  $x$  is visible through  $C'_1$  then  $(\nu x)C_1 \rightarrow (\nu x)C'_1$
7. if  $C_1 \rightarrow C'_1$  then  $C_1 \mid C_2 \rightarrow C'_1 \mid C_2$  and  $C_2 \mid C_1 \rightarrow C_2 \mid C'_1$

It is easy to verify that for  $i \geq i_0$ , if  $A_i$  has the form  $C_i(B_{i,1}, \dots, B_{i,m})$  and  $A_{i+1}$  similarly the form  $C_{i+1}(B_{i+1,1}, \dots, B_{i+1,m})$  and for each  $j : 1 \leq j \leq m$ , either  $B_{i,j} = B_{i+1,j}$  or  $B_{i,j} \rightarrow B_{i+1,j}$ , then either  $C_i = C_{i+1}$  or  $C_i \rightarrow C_{i+1}$ . To prove (i) it therefore suffices to show that only a finite number of contexts are reachable from any context  $C$ . We use the notion of *legitimate prefix* to establish the fact.

**Definition 4.6** (Context prefix, Legitimate prefix)

1. A (context) *prefix* is a string  $p = \Omega_1 \cdots \Omega_n$  where each  $\Omega_i$  is either  $(\nu x)$ ,  $(\lambda x)$ , or  $[x]$ , where  $x$  is small enough. Write  $p \triangleright C$  for the context obtained by prefixing  $C$  with  $p$ .
2. A prefix  $\Omega_1 \cdots \Omega_n$  is *legitimate* if
  - (a) at most one  $\Omega_i$  has the form either  $(\lambda x)$  or  $[x]$ , and
  - (b) the total number of occurrences of operators of the form  $(\nu x)$  or  $(\lambda x)$  for some small enough  $x$  is at most  $n_0$ .

**Lemma 4.7** For all  $C$ ,  $\{C' \mid C \rightarrow^* C'\}$  is finite.

**PROOF** By induction in the size of  $C$ :

$C = [\cdot]$ : It suffices to show that any context reachable from  $[\cdot]$  has the form  $p \triangleright [\cdot]$  where  $p$  is a legitimate prefix. To show this assume that  $p$  is legitimate and that  $p \triangleright [\cdot] \rightarrow C'$ . Then  $C'$  has the form  $p' \triangleright [\cdot]$ . Clearly condition (i) above is satisfied. To see that also (ii) is satisfied suppose for a contradiction that it is not, so that  $p'$  has  $n_0 + 1$  occurrences of a binding operator. Then  $p'$  must have the form  $p_1(\nu x)p_2\Omega p_3$  for some  $x$  where  $\Omega$  binds  $x$ . But this cannot happen since the justification of  $p \triangleright [\cdot] \rightarrow p' \triangleright [\cdot]$  must have appealed to rule (6) for justifying  $(\nu x)p'' \triangleright [\cdot] \rightarrow (\nu x)p_2\Omega p_3 \triangleright [\cdot]$  for some  $p''$ . But  $x$  is not visible through  $p_2\Omega p_3$ —a contradiction.

$C = (\nu x)C'$ : We show that any context reachable from  $C$  (in 1 step or more) has the form  $p \triangleright C_1$  where  $p$  is a legitimate prefix and  $C_1$  is reachable from  $C'$  (in 1 step or more). So assume that  $p \triangleright C_1 \rightarrow C_2$ . The only case that needs considering is when  $p$  has the form  $p'(\nu x)$ ,  $C_1$  the form  $\Omega C'_1$ , and  $C_2$  the form  $p\Omega(\nu x) \triangleright C'_1$ . We then need to show that  $p\Omega(\nu x)$  is legitimate, but this follows exactly as in the previous case.

$C = C_1 \mid C_2$ : We show that any context reachable from  $C$  (in 1 step or more) has the form  $p \triangleright (C'_1 \mid C'_2)$  where  $p$  is legitimate,  $C'_1$  is reachable from  $C_1$ , and  $C'_2$  reachable from  $C_2$ . The only cases that need considering are applications of rule (3), but these follow as in the case for restriction above.

The remaining cases are quite straightforward.

□ (Lemma 4.7)

**Non-parallel agents.** We then proceed to the proof of (ii).

**Lemma 4.8** *Suppose that  $A$  has no occurrences of  $|$ . For all derivations  $d = A_0 \rightarrow \dots \rightarrow A_n \rightarrow \dots$  with  $A_0 = A$ ,  $\mathcal{R}(d)$  is finite.*

PROOF The proof proceeds by induction in the size of  $A$ . The cases for  $\mathbf{0}$ ,  $+$ , prefixing, conditional, abstraction, application, and concretion follow directly from the induction hypothesis. This leaves two cases to be considered. For restriction the proof is a correlate of the corresponding case in the proof of Lemma 4.7. So assume that  $A = \text{fix}D.A'$ . We to show that any agent reachable from  $A$  has the form  $p \triangleright (A''\{A/D\})$  where  $p$  is a legitimate prefix and  $A''$  is reachable from  $A'$ , thus completing the proof by the induction hypothesis. To each transition  $A_i \rightarrow A_{i+1}$  is associated a unique justification, a proof using the axioms and rules among (1)–(16) together with alpha-conversion. Say that *step  $i$  refers to  $A$* , if the justification of the transition  $A_i \rightarrow A_{i+1}$  involves an appeal to (6) with  $D$  instantiated to itself, and  $A$  to  $A'$ . Suppose now that  $A_i$  has the form  $p \triangleright (A''\{A/D\})$  such that  $A''$  is reachable from  $A'$ . Handling the case where step  $i$  is an instance of one of the axioms (10)–(12) as in the proof of Lemma 4.7 only one potentially problematic case remains, namely where step  $i$  refers to  $A$ . This, however, can only be the case when  $A''$  has the form  $p' \triangleright D$  for  $p'$  a prefix, and in this situation it must, as we have seen, be the case that the prefix  $pp'$  is legitimate. Thus  $A_{i+1}$  has been brought into the desired form.  $\square$  (Lemmas 4.8 and 4.4, Theorem 4.2)

## 5 Choosing names

The problem with Theorem 4.2 is that potential derivations might be lost because at some point it becomes impossible to choose a small enough name. In this section we show that we can avoid this problem by choosing  $n_0$  sufficiently large. Let  $\#_{fns}(A)$  be the maximal number of free names in any subterm of  $A$ , and  $\#_{par}(A)$  be the number of occurrences of the parallel combinator  $|$  in  $A$ .

**Lemma 5.1** *For all  $A_n$ , if  $A_0 \rightarrow^* A_n$  then  $|\text{fn}(A_n)| \leq \#_{fns}(A_0) \cdot (\#_{par}(A_0) + 1)$*

PROOF Assume that

$$d = A_0 \rightarrow \dots \rightarrow A_n \rightarrow \dots$$

We show  $|\text{fn}(A_n)| \leq \#_{fns}(A_0) \cdot (\#_{par}(A_0) + 1)$  by structural induction, using the notion of legitimate prefix introduced in the proof of 4.7. For all cases except  $\lambda$ , recursion, and parallel composition, the result follows directly from the induction hypothesis, so only these three are considered:

$A_0 = (\lambda x)A'_0$ . If  $n > 0$  it must be the case that (up to an initial sequence of alpha-conversions)  $A_0 \rightarrow A_1$  is an instance of (4), i.e. that  $A_1 = A'_0$ , so that

$A'_0 \rightarrow^* A_n$ . Then by the induction hypothesis,

$$\begin{aligned} |\text{fn}(A_n)| &\leq \#_{fns}(A'_0) \cdot (\#_{par}(A'_0) + 1) \\ &= \#_{fns}(A_0) \cdot (\#_{par}(A_0) + 1) \end{aligned}$$

$A_0 = A_{0,1} \mid A_{0,2}$ . In this case  $A_n$  has the form  $p \triangleright (A_{n,1} \mid A_{n,2})$  for some legitimate prefix  $p$ . Then for each  $i \in \{1, 2\}$  we find legitimate prefixes  $p_i$  such that  $A_{0,i} \rightarrow^* p_i \triangleright A_{n,i}$ , and  $p$  is the merge of  $p_1$  and  $p_2$  in a manner such that if  $[x]$  occurs in  $p$  with  $x$  in a bound position then so it does in whichever  $p_i$  that contains  $[x]$ . By the induction hypothesis,

$$|\text{fn}(p_i \triangleright A_{n,i})| \leq \#_{fns}(A_{0,i}) \cdot (\#_{par}(A_{0,i}) + 1)$$

for  $i = 1$  and  $i = 2$ . Now

$$\begin{aligned} |\text{fn}(A_n)| &\leq |\text{fn}(p_1 \triangleright A_{n,1})| + |\text{fn}(p_2 \triangleright A_{n,2})| \\ &\leq \#_{fns}(A_{0,1}) \cdot (\#_{par}(A_{0,1}) + 1) \\ &\quad + \#_{fns}(A_{0,2}) \cdot (\#_{par}(A_{0,2}) + 1) \end{aligned}$$

Let  $B$  be whichever of  $A_{0,1}/A_{0,2}$  such that  $\#_{fns}(B)$  is maximal. Then

$$\begin{aligned} &\#_{fns}(A_{0,1}) \cdot (\#_{par}(A_{0,1}) + 1) \\ &+ \#_{fns}(A_{0,2}) \cdot (\#_{par}(A_{0,2}) + 1) \\ &\leq \#_{fns}(B) \cdot (\#_{par}(A_{0,1}) + \#_{par}(A_{0,2}) + 2) \\ &= \#_{fns}(B) \cdot (\#_{par}(A_0) + 1) \\ &\leq \#_{fns}(A_0) \cdot (\#_{par}(A_0) + 1) \end{aligned}$$

completing the case.

$A_0 = \text{fix}D.A'_0$ . Since  $\#_{par}(A_0) = 0$  by the assumption of finite control it suffices to show that  $|\text{fn}(A_n)| \leq \#_{fns}(A_0)$ . We then find legitimate prefixes  $p$  and  $p'$  such that

$$\text{fix}D.A'_0 \rightarrow^* p \triangleright \text{fix}D.A \rightarrow^* A_n,$$

$A_n$  has the form  $p' \triangleright A'_n\{\text{fix}D.A/D\}$ , and  $A'_0 \rightarrow^* A'_n$ . Note that we can assume that  $p$  has no free occurrences of names since if it had, before  $\text{fix}D.A'_0$  would be subsequently unfolded, the free name (occurring in an output prefix) would be eliminated by an application of (7). By the induction hypothesis we know that

$$\begin{aligned} |\text{fn}(A'_n)| &\leq \#_{fns}(A'_0) \\ &= \#_{fns}(A_0) \end{aligned}$$

The only case in which  $|\text{fn}(A_n)|$  could be greater than  $|\text{fn}(A'_n)|$  is when  $p'$  contains an occurrence of an output prefix  $[y]$  such that the occurrence of  $y$  in  $[y]$  is free in  $p'$ .

This, however, can only happen if we can factorise the derivation  $p \triangleright \text{fix}D.A'_0 \rightarrow^* A_n$  as follows

$$p \triangleright \text{fix}D.A'_0 \rightarrow^* p'' \triangleright A''_n\{\text{fix}D.A'_0/D\} \rightarrow^* A_n$$

such that  $p''$  has no free occurrence of  $y$  (i.e.  $A_n$  results from  $A''_n\{\text{fix}D.A'_0/D\}$  by applications of 8, 9, and 11), and such that  $A'_0 \rightarrow^* A''_n$ . Moreover  $\text{fn}(A''_n) = \text{fn}(A'_0) \cup \{y\}$ . But then, since we know by the induction hypothesis that  $|\text{fn}(A''_n)| \leq \#_{fns}(A_0)$  the proof is complete.  $\square$  (Lemma 5.1)

Thus by choosing  $n_0$  greater than  $\#_{fns}(A_0) \cdot (\#_{par}(A_0) + 1)$  it will always be possible to choose a small enough name.

## 6 Decidability

Consider now a version of pr-bisimulation of Definition 3.1 modified such that  $z$  in 3.1.2 and 3.1.3 is required to be small enough, and such that commitment in 3.1.4 is conditional on only small enough names being chosen. Call the ensuing variant of pr-bisimulation for *name-bounded pr-bisimulation*, or nbpr-bisimulation, for short. By König's Lemma, since the number of transitions that use only small enough names and that emanate from a given agent is finite, and by Lemma 5.1 any infinite path must visit the same agent expression infinitely often, the following decidability result obtains:

**Theorem 6.1** *Name-bounded pr-bisimulation equivalence is decidable.*  $\square$

Using this result we can then easily establish our first main Theorem:

**Theorem 6.2** *Strong late bisimulation equivalence is decidable.*

PROOF Both decidability results follow from decidability of  $\varepsilon$ -pr-bisimulation which we go on to demonstrate. Assume first if  $R_\varepsilon$  and  $R_\varepsilon^{-1}$  are both pr-simulations. Then they are also nbpr-simulations for any  $n_0$ . Suppose on the other hand that  $R_\varepsilon$  and  $R_\varepsilon^{-1}$  are both nbpr-bisimulations for some  $n_0$  greater than

$$\#_{fns}(A) \cdot (\#_{par}(A) + 1) + \#_{fns}(B) \cdot (\#_{par}(B) + 1).$$

Let a *name representation* be any pair of maps  $(f_{free}, f_{bound})$  such that  $f_{free}$  is an injection, and for each binding occurrence of a name in  $A$  or  $B$   $f_{bound}$  maps that name into a small enough name, such that

1. if  $x$  and  $y$  are distinct, both occurs freely in some subterm of  $A$  or  $B$ , and both are occurrences of bound names, then  $f_{bound}(x) \neq f_{bound}(y)$ , and
2. if  $x$  and  $y$  are distinct, both occurs in a subterm of  $A$  or  $B$  and, say,  $x$  is an occurrence of a bound name and  $y$  an occurrence of a free name, then  $f_{bound}(x) \neq f_{free}(y)$ .

For a name partition  $\varepsilon$  let  $f_{free}(\varepsilon) = \{\{f(x) \mid x \in U\} \mid U \in \varepsilon\}$ . Because of the choice of  $n_0$ , a name representation exists. Let then  $AS_\varepsilon B$  if and only if there is a name representation  $(f_{free}, f_{bound})$  such that if  $A'$  and  $B'$  are the agents resulting from renaming free and bound name according to  $(f_{free}, f_{bound})$ , then  $A'$  and  $B'$  are  $f_{free}(\varepsilon)$ -nbpr-bisimulations. It is then easy to verify that, due to the choice of  $n_0$ ,  $S_\varepsilon$  is an  $\varepsilon$ -pr-bisimulation. This completes the proof.  $\square$  (Theorem 6.2)

**Other equivalences.** In a similar manner we can prove decidability for other versions of bisimulation equivalence, notably early strong bisimulation equivalence, late and early weak bisimulation equivalence. Decidability of open bisimulation equivalence can also be shown in this manner. However, open bisimulation equivalence is already known to be decidable (indeed it was formulated with this as a central concern).

Early equivalence is characterised by permuting the quantifications over transitions and inputs which is implicit in clauses (3) and (4) of Def. 3.1 (c.f. [12] for a definition of early equivalence). The proofs of Theorems 6.1 and 6.2 are only minimally affected by this modification. For the weak late and early equivalences again only small modifications are needed (though alternative characterisations of these equivalences are likely to be mandatory for reasons of efficiency). We thus obtain:

**Theorem 6.3** 1. *Strong early bisimulation equivalence is decidable.*

2. *Weak late and early bisimulation equivalence are decidable.*  $\square$

## 7 Complexity and Discussion

The obvious backtracking-based algorithm for deciding name-bounded bisimulation equivalence is quite inefficient. As for standard bisimulation equivalence a better solution is obtained using the Paige-Tarjan algorithm [15, 8] with a worst-case running time of  $\mathcal{O}(n_t \log n_s + n_s)$  where  $n_t$  is the number of transitions and  $n_s$  the number of states. With minor modifications to cater for bound output this algorithm is applicable once the full state spaces have been constructed, as pairs  $(A, \varepsilon)$ . If the total number of reachable agents  $A$  is  $m$  and the sum of the length of the input agents is  $n$  then, since in the worst case  $n_0$  is quadratic in  $n$ ,  $n_s$  is  $\mathcal{O}(m2^{n^2})$  and  $n_t$  is  $\mathcal{O}(m^2 n^2 2^{n^2})$ . Thus the running time of the Paige-Tarjan algorithm is bounded by  $\mathcal{O}(n^4 2^{n^2} m^2 \log m)$ . To estimate  $m$  note first that up to the choice of names the number of agents reachable from one parallel component is  $\mathcal{O}(n)$ . Since each name can be instantiated in  $n_0$  different ways the entire number of agents reachable from one parallel component is  $\mathcal{O}(n 2^{n^2 \log(n^2)})$ . Thus  $m$  is bounded by  $\mathcal{O}((n 2^{n^2 \log(n^2)})^n) = \mathcal{O}(2^{n^3 \log(n^2) + n \log n})$ , which is  $2^{\mathcal{O}(n^3 \log(n^2))}$ . This is strikingly close to the similarly approximated upper bound for CCS of  $2^{\mathcal{O}(n \log n)}$ . Both the parallel combinator already present in CCS and the  $\pi$ -calculus features

of name generation and passing causes an exponential blow-up in the size of the state space. One might fear that these two causes of state space blow-up could interfere in a serious manner, resulting in double exponential running times or worse. However, even though some interference does take place because of scope extrusion, our results show that this fear is unfounded.

**Lower bounds.** Concerning lower bounds Jonsson and Parrow [7] shows that the bisimulation problem for data-independent programs (not including the parallel combinator  $|$ ) is NP-hard. Since data-independent programs are subsumed by those considered here that lower bound applies here as well.

**Efficiency.** As for efficiency, based on the asymptotically quite similar worst-case bounds for CCS and for the  $\pi$ -calculus, since the Paige-Tarjan algorithm has been applied to quite realistically sized examples in CCS one might hope that this applies here too. Whether this in fact turns out to be the case remains to be seen. It may well be that alternative characterisations of the equivalences can be exploited to improve the efficiency of our algorithms, along the lines of for instance the efficient characterisation of strong open bisimulation equivalence [17], or the symbolic bisimulations of Hennessy and Lin [5]. For the weak equivalences in particular we expect such efficient characterisations to be indisposable.

## References

- [1] R. Amadio. A uniform presentation of CHOCS and  $\pi$ -calculus. Rapport de Recherche 1726, INRIA-Lorraine, Nancy, 1992.
- [2] D. Caucal. Graphes canoniques des graphes algébriques. *Informatique Théorique et Applications (RAIRO)*, 24(4):339–352, 1990.
- [3] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. In Proc. CONCUR'92, W. R. Cleaveland (ed.), *Lecture Notes in Computer Science*, 630:138–147, 1992.
- [4] M. Dam. Model checking mobile processes. In *Proc. CONCUR'93*, Lecture Notes in Computer Science, 715:22–36, 1993. Full version in SICS report RR94:1, 1994.
- [5] M. Hennessy and H. Lin. Symbolic bisimulations. Dept. of Computer Science, University of Sussex, Report 1/92, 1992.
- [6] Y. Hirschfeld and F. Moller. A fast algorithm for deciding bisimilarity of normed context-free processes. In Proc. CONCUR'94, B. Jonsson, J. Parrow (eds.), *Lecture Notes in Computer Science*, 836:48–63, 1994.

- [7] B. Jonsson and J. Parrow. Deciding bisimulation equivalences for a class of non-finite-state programs. *Information and Computation*, 1992.
- [8] P. C. Kannelakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86:43–68, 1990.
- [9] R. Milner. The polyadic  $\pi$ -calculus: A tutorial. Technical Report ECS-LFCS-91-180, Laboratory for the Foundations of Computer Science, Department of Computer Science, University of Edinburgh, 1991.
- [10] R. Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2:119–141, 1992.
- [11] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–40 and 41–77, 1992.
- [12] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. *Theoretical Computer Science*, 114:149–171, 1993.
- [13] F. Orava. *On the Formal Analysis of Telecommunication Protocols*. PhD thesis, Dept. of Computer Systems, Uppsala University and Swedish Institute of Computer Science, 1994. Forthcoming.
- [14] F. Orava and J. Parrow. An algebraic verification of a mobile network. *Formal Aspects of Computing*, pages 497–543, 1992.
- [15] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal of Computing*, 16(6):973–989, 1987.
- [16] D. Sangiorgi. From  $\pi$ -calculus to higher-order  $\pi$ -calculus—and back. To appear in Proc. TAPSOFT’93, 1993.
- [17] D. Sangiorgi. A theory of bisimulation for the  $\pi$ -calculus. in *Proc. CONCUR’93* Lecture Notes in Computer Science, 715:127–142, 1993.
- [18] D. Walker. Objects in the  $\pi$ -calculus. *Information and Computation*, 1994. (To appear).