ELSEVIER Theoretical Computer Science 183 (1997) 215–228

Theoretical Computer Science

On the decidability of process equivalences for the π-calculus[1]

Mads Dam*

SICS, Box 1263, S-164 28 Kista, Sweden

Abstract

We present general results for showing process equivalences applied to the finite control fragment of the π-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 π-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 λ-calculus, and reactive systems such as CCS or CSP, on the other, has recently had considerable attention. The π-calculus [8] 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 [6], lambda calculus [7], object-oriented programming languages [15], and higher-order processes [13] 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 [11, 10]. It is therefore important to investigate to what extent methods and tools developed for, say, CCS lift to the more expressive setting of the π-calculus.

* E-mail: mfd@sics.se.

[1] Work partially supported by ESPRIT BRA project 8130 "LOMAPS", and the Human Capital and Mobility Project EXPRESS.

One such set of tools of fundamental importance are process equivalence checking algorithms, as exemplified by the Paige–Tarjan algorithm [12, 5]. 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 π-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 π-calculus. This is far from a trivial result, since even very simple π-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 [4]. 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[2] **if** $x = y$ **then** P **else** Q to the π-calculus, as we do, we can for instance encode the memory cell

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

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

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

where

$$Gen(x) = (\nu y)\bar{x}y.Gen(y)$$

$$Listen(x) = x(y).Listen(y)$$

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 (cf. [9]), and open (or uniform [1]) bisimulation equivalence [14].[3] The tools consist of two key Lemmas, proofs of which are given in the paper:

[2] In the paper we actually use the notation $[x = y]PQ$ instead of **if** $x = y$ **then** P **else** Q for the conditional.
[3] For open bisimulation equivalence decidability is already known [14].

(i) 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.

(ii) 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 σ 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 [6]. For this equivalence we show how the partition refinement algorithm of Paige and Tarjan [12] can be applied resulting, as for the case of e.g. CCS, in a single exponential worst-case complexity.

Our results considerably strengthens previous results in the area of value-passing process calculi. Besides the decidability result of Jonsson and Parrow [4] for data-independent programs, Hennessy and Lin [3] 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 μ-calculus of [2].

2. The Polyadic π-calculus, Syntax

We use a slight extension of Milner's polyadic π-calculus, introduced in [2]. Letters $x, y, z, \ldots$ range over *names* of which there is a countably infinite supply, A, B range over *agents*, and D over *agent identifiers*. *Actions*, α, β, are either names, co-names of the form $\bar{x}$, or the distinguished constant τ. If α is a name x then $\mathrm{n}(\alpha)$ (the name of α) is x. and $\mathrm{p}(\alpha)$ (the polarity of α) is $-$. Otherwise if $\alpha = \bar{x}$ then $\mathrm{n}(\alpha) = x$ and $\mathrm{p}(x) = +$. The syntax of agents is the following:

$$\begin{aligned} A ::= {} & 0 \mid A + A \mid \alpha.A \mid A \mid A \mid [x = y]AA \mid \\ & (\lambda x)A \mid Ax \mid (\nu x)A \mid D \mid \mathrm{fix} D.A \mid \ [x]A \end{aligned}$$

Name binders are the operators λ and ν. We use the notation $\mathrm{fn}(A)$ 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. An agent is *closed* if it does not contain any free occurrences of agent identifiers. The intended meaning of connectives is familiar from CCS and the π-calculus. The present version is based on the polyadic π-calculus of [6]. 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 $\mathrm{fix} D.A$, only allowing free occurrences of D in A within the scope of a prefix operator), and for which uses of the parallel combinator $|$ within recursive definitions are disallowed. We refer to this fragment as the *finite control* fragment. In addition, we require for technical reasons that recursions fix$D.A$ are *fully parametrised* in the sense that recursive agents 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 $(\mathrm{fix}D.A)(x_1,\dots,x_n)$ with

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

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 π-calculus (cf. [14]). 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 [6]. 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:

$$\frac{}{0:0}\qquad\frac{A:0\quad B:0}{A+B:0}\qquad\frac{A:n\quad n\leqslant 0}{x.A:0}\qquad\frac{A:n\quad n\geqslant 0}{\overline{x}.A:0}\qquad\frac{A:0}{\tau.A:0}$$

$$\frac{A:0\quad B:0}{A\,|\,B:0}\qquad\frac{A:n\quad B:n}{[x=y]AB:n}\qquad\frac{A:n\quad n\leqslant 0}{(\lambda x)A:n-1}\qquad\frac{A:n-1\quad n\leqslant 0}{Ax:n}$$

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

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

3. Operational semantics

In [6] the semantics of the π-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 ε on the set of names. A partitioning ε 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 ε. 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

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

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

Proposition 1. *Let A be a closed, well-formed agent, and ε a name partitioning. Then* $\mathrm{nf}(A,\varepsilon)$ *is a well-formed agent in normal form.*

Proof. Structural induction. □

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 [6]. Using the definitions of [6] it is quite easy to show that for all well-formed, conditional-free agents A, $\mathrm{nf}(A,\varepsilon)$ is independent of ε, and for all ε, $A \equiv \mathrm{nf}(A,\varepsilon)$.

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

- $A \| B = A \mid B$ when $A : 0$ and $B : 0$. If $A : 0$ and $B : n \neq 0$ then $A \| [x]B' = [x](A \| B')$, $A \| (\nu x)[x]B' = (\nu y)[y](A \| B'\{y/x\})$, and $A \| (\lambda x)B' = (\lambda y)(A \| B'\{y/x\})$, where in the two last cases it is assumed that $y \notin \mathrm{fn}(A)$. 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 $z \notin (\mathrm{fn}(A') - \{x\}) \cup (\mathrm{fn}(B') - \{y\})$. 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:

$$\text{ACT: } \frac{}{\alpha.A \succ_\varepsilon \alpha.A} \qquad \text{SUM: } \frac{A_1 \succ_\varepsilon B}{A_1 + A_2 \succ_\varepsilon B}$$

$$\text{COMM: } \frac{A_1 \succ_\varepsilon x.B_1 \quad A_2 \succ_\varepsilon \overline{y}.B_2 \quad \varepsilon \models x = y}{A_1 \mid A_2 \succ_\varepsilon \tau.(\mathrm{nf}(B_1,\varepsilon) \cdot \mathrm{nf}(B_2,\varepsilon) \quad)}$$

$$\text{PAR: } \frac{A_1 \succ_\varepsilon \alpha.B}{A_1 \mid A_2 \succ_\varepsilon \alpha.(\mathrm{nf}(B,\varepsilon) \parallel A_2)}$$

$$\text{RES-1: } \frac{A \succ_{(\nu x)\varepsilon} \tau.B}{(\nu x)A \succ_\varepsilon \tau.(\nu x)B}$$

$$\text{RES-2: } \frac{A \succ_{(\nu x)\varepsilon} \alpha.B}{(\nu x)A \succ_\varepsilon \alpha.(\nu x)B} \quad (x \neq \mathrm{n}(\alpha))$$

+ symmetrical versions of rules SUM, COMM and PAR

Relating to [6] let the *full* name partitioning ε_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 [6] if and only if for some B', $\mathrm{nf}(A,\varepsilon_f) \succ_{\varepsilon_f} B'$, and $\mathrm{nf}(B,\varepsilon_f) = \mathrm{nf}(B',\varepsilon_f)$.

Definition 2 (*Simulations, bisimulations*). A (strong, late) *partition-relativised simulation* (or pr-simulation) is an ε-indexed family of binary relations R_ε on well-formed agents satisfying the following conditions:

(i) If $AR_\varepsilon B$ then $\mathrm{nf}(A,\varepsilon)R_\varepsilon\mathrm{nf}(B,\varepsilon)$.

(ii) If $AR_\varepsilon B$ and $A : n$ then $B : n$.

(iii) If $[x]A'R_\varepsilon[y]B'$ then $A'R_\varepsilon B'$ and $\varepsilon \models x = y$.

(iv) If $(\nu x)[x]A'R_\varepsilon(\nu y)[y]B'$ then $A'\{z/x\}R_{(\nu z)\varepsilon}B'\{z/y\}$ for all z such that $z \notin (\mathrm{fn}(A') - \{x\}) \cup (\mathrm{fn}(B') - \{y\})$.

(v) If $(\lambda x)A'R_\varepsilon(\lambda y)B'$ then for all z, $A'\{z/x\}R_\varepsilon B'\{z/y\}$.

(vi) 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 ε both R_ε and R_ε^{-1} are pr-simulations; A and B are ε-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 ε.

Serving as justification for Definition 2 it can be shown quite easily that for the fragment of well-formed, conditional-free agents, $\sim_{\varepsilon_f}$ is the (strong) bisimulation equivalence of [6], and $\sim$ is strong congruence.

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 λ or ν.

We restrict these choices by assuming an enumeration $x_0, x_1, \ldots$ 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 3 (*Reachability relation*). Relative to a choice of n_0 the reachability relation $\rightsquigarrow$ on well-formed agents is defined as follows:

(i) For all ε, $A \rightsquigarrow \mathrm{nf}(A,\varepsilon)$,

(ii) $[x]A \rightsquigarrow A$,

(iii) $(\nu x)[x]A \rightsquigarrow A\{y/x\}$ whenever y is small enough and $y \notin \mathrm{fn}(A) - \{x\}$,

(iv) $(\lambda x)A \rightsquigarrow A\{y/x\}$ whenever y is small enough,

(v) If P is a process in normal form and for some ε, $P \succ_\varepsilon \alpha.A$ while choosing only names that are small enough, then $P \rightsquigarrow A$.

The Relation $\rightarrow$: Our aim is to show that for all well-formed, finite control A and n_0, $\{B \mid A \rightsquigarrow^* B\}$ is finite. To show this 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

(i) $A + B \rightarrow A$, $A + B \rightarrow B$

(ii) $\alpha.A \rightarrow A$

(iii) $[x = y]AB \rightarrow A$, $[x = y]AB \rightarrow B$

(iv) $(\lambda x)A \rightarrow A\{y/x\}$ whenever y is small enough

(v) $Ax \rightarrow A$

(vi) $\mathrm{fix}D.A \rightarrow A\{\mathrm{fix}D.A/D\}$

(vii) $[x]A \rightarrow A$

(viii) If $A \rightarrow B$ and $x \in \mathrm{fn}(B)$ then $(\nu x)A \rightarrow (\nu x)B$

(ix) $(\nu x)A \rightarrow A$

(x) $(\nu x)(\lambda y)A \to (\lambda y)(\nu x)A$
(xi) If $x \neq y$ then $(\nu x)[y]A \to [y](\nu x)A$
(xii) $(\nu x)(\nu y)[y]A \to (\nu y)[y](\nu x)A$
(xiii) If $A \to A'$ then $A \mid B \to A' \mid B$ and $B \mid A \to B \mid A'$
(xiv) $((\lambda x)A) \mid B \to (\lambda x)(A \mid B)$, $A \mid ((\lambda x)B) \to (\lambda x)(A \mid B)$
(xv) $([x]A) \mid B \to [x](A \mid B)$, $A \mid ([x]B) \to [x](A \mid B)$
(xvi) $((\nu x)A) \mid B \to (\nu x)(A \mid B)$, $A \mid ((\nu x)B) \to (\nu x)(A \mid B)$

Proposition 4. $\rightsquigarrow^* \subseteq \to^*$.

Proof. We need to show
(i) For all $\varepsilon, A \to^* nf(A,\varepsilon)$.
(ii) $[x]A \to^* A$.
(iii) $(\nu x)[x]A \to^* A\{y/x\}$ whenever y is small enough and $y \notin \mathrm{fn}(A) - \{x\}$.
(iv) $(\lambda x)A \to^* A\{y/x\}$ whenever y is small enough.
(v) $P \to^* A$ whenever P is a process in normal form and for some ε and α, $P \succ_\varepsilon \alpha.A$.
Of these (i) and (v) use structural induction, and (ii)–(iv) follow directly from the conditions given. □

Let ρ be an infinite derivation of the form

$$\rho = A_0 \to \cdots \to A_n \to \cdots \tag{1}$$

from A (i.e. $A_0 = A$). Since $\to$ is finitely-branching, to show that $\{B \mid A \to^* B\}$ is finite for all A, it suffices to show that for all such derivations ρ, the set $R(\rho) = \{A_i \mid i \in \omega\}$ is finite. To prove this we analyse the structure of terms A_i.

Contexts: Let an m'ary *context* be a non-recursive term C with m occurrences of the empty context $[\cdot]$, that is, a term generated by the abstract syntax

$$C ::= [\cdot] \mid 0 \mid C + C \mid \alpha.C \mid C \mid C \mid [x = y]CC \mid (\lambda x)C \mid Cx \mid (\nu x)C \mid [x]C$$

Using the finite control assumption, we can find an m such that each A_i can be written in the form

$$A_i = C_i(B_{i,1}, \ldots, B_{i,m}) \tag{2}$$

where C_i is an m'ary context, and where each $B_{i,j}$ is a term that does not contain occurrences of the parallel composition operator. By convention contexts are filled from left to right such that $C(A'_1, \ldots, A'_m)$ is C with the ith leftmost occurrence of $[\cdot]$ substituted for A'_i, etc. Contexts are equipped with a transition structure corresponding to the relation $\to$ on terms. The conditions (1)–(16) above are readily applicable to contexts, except that the notion of free name is not quite appropriate. Instead, 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. Now we can define the relation $\rightarrow$ on contexts by the conditions (1)–(16) plus the following, where Ω ranges over operators among (νx), (λx), and $[x]$ with x small enough:

(I) $[\cdot] \rightarrow (\nu x)[\cdot]$, $[\cdot] \rightarrow (\lambda x)[\cdot]$, and $[\cdot] \rightarrow [x][\cdot]$ where x is small enough,

(II) if $C_1 \rightarrow C_1'$ and x is visible through C_1' then $(\nu x)C_1 \rightarrow (\nu x)C_1'$

Proposition 5. *Each A_i in* (1) *can be written in the form* (2) *such that, for all $i \in \omega$ and $j : 1 \leqslant j \leqslant m$,*

(i) *$B_{i,j}$ does not contain occurrences of the parallel composition operator,*

(ii) *C does not contain occurrences of the fixed point operator,*

(iii) *either $B_{i,j} = B_{i+1,j}$ or else $B_{i,j} \rightarrow B_{i+1,j}$, and*

(iv) *either $C_i = C_{i+1}$ or else $C_i \rightarrow^* C_{i+1}$.*

Proof. By induction on i and the structure of proof that $A_i \rightarrow A_{i+1}$. Observe that in fact, in (4), $C_i \rightarrow^* C_{i+1}$ in either 1 or 2 steps corresponding to applications of one of the rules (14)–(16). In the first step an operator (λx), (νx) or $[x]$ is transferred from an $B_{i,j}$ to the context using (I), and in second step the original rule application ((14)–(16)) is mimicked on contexts. □

Note that 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. 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 C_0 is built using only operators of the form $[x]$, (λx), (νx), or $|$. Call such a context a *restricted context.*

We have thus reduced the problem of showing that $R(\rho)$ is finite to the problem of showing

(i) only a finite number of distinct contexts C_i are reachable,

(ii) any derivation ρ that does not involve parallel composition visits a finite number of distinct agents only, under the assumption that C_0 is a restricted context.

Finite reachability for contexts: The property (i) is proved using the notion of *legitimate prefix.*

Definition 6 (*Context prefix, legitimate prefix*).

(i) A (context) *prefix* is a string $p = \Omega_1 \cdots \Omega_n$ where each Ω_i is either (νx), (λx), or $[x]$, and where x is small enough. Write $p :: C$ for the context obtained by prefixing C with p.

(ii) A prefix $\Omega_1 \cdots \Omega_n$ is *legitimate* if

(a) at most one Ω_i has the form either (λx) or $[x]$, and

(b) the total number of occurrences of operators of the form (νx) or (λx) for some small enough x is at most n_0.

Lemma 7. *For all C, $\{C' \mid C \rightarrow^* C'\}$ is finite.*

Proof. By induction in the structure of C:

$C=[\cdot]$: Any context reachable from $[\cdot]$ has the form $p :: [\cdot]$ where p is a prefix. It suffices to show that p is legitimate. To show this assume that p is legitimate and that $p :: [\cdot] \rightarrow C'$. Then C' has the form $p' :: [\cdot]$. Clearly condition (a) above is satisfied. To see that also (b) 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 Ω binds x. But this is impossible since the justification of $p :: [\cdot] \rightarrow p' :: [\cdot]$ must have appealed to (II) for justifying $(\nu x)p'' :: [\cdot] \rightarrow (\nu x)p_2\Omega p_3 :: [\cdot]$ for some p''. But x is not visible through $p_2\Omega p_3$ – a contradiction.

$C=(\nu x)C'$: Using the inductive hypothesis it suffices to show that any context reachable from C (in 1 step or more) has the form $p :: C_1$ where p is a legitimate prefix and C_1 is reachable from C' (in 1 step or more). So assume that $p :: C_1 \rightarrow C_2$. 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 $p\Omega(\nu x) :: C_1'$. 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 $p :: (C_1' \mid C_2')$ 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 one of the rules (14)–(16), but these follow as in the case for ν.

The remaining cases are quite straightforward. □

Finite reachability for $\mid$-free agents: We then proceed to show finite reachability for the relation $\rightarrow$ on agents that do not contain occurrences of $\mid$. Define the *size*, $|A|$, of such an agent A in the following manner:

$$|0| = |D| = 2$$

$$|A + B| = |[x=y]AB| = |A| + |B| + 1$$

$$|\alpha.A| = |(\lambda x)A| = |Ax| = |[x]A| = |\mathrm{fix}\, D.A| = |A| + 1$$

$$|(\nu x)A| = 2 \cdot |A| + 1$$

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

Thus, we can assume that the unfolding axiom (6) is used infinitely often along ρ.

Lemma 9. *Suppose that A has no occurrences of $\mid$. For all derivations $\rho = A_0 \rightarrow \cdots \rightarrow A_n \rightarrow \cdots$ with $A_0 = A$, $R(\rho)$ is finite.*

Proof. The proof proceeds by induction in the size of A. Most cases (0, $A + B$, $\alpha.A$, $[x=y]AB$, $(\lambda x)A$, Ax, $[x]A$) are direct consequences of the induction hypothesis. For

$(\nu x)A$ the proof follows the corresponding case in the preceding proof. So assume that $A = \mathrm{fix}\,D.A'$. We show that any agent reachable from A has the form $p :: (A''\{A/D\})$ where p is a legitimate prefix and A'' is reachable from A'. This completes the proof by the induction hypothesis. To each transition $A_i \to A_{i+1}$ is associated a proof using the rules (1)–(16). Say that *step i refers to A*, if the justification of the transition $A_i \to A_{i+1}$ involves an appeal to (6) with D instantiated to itself, and A to A'. Suppose that A_i has the form $p :: (A''\{A/D\})$ where A'' is reachable from A'. The situation where step i is an instance of one of the axioms (10)–(12) can be handled as in the proof of Lemma 7. Thus, we only need to consider the case where step i refers to A. But then A'' must have the form $p' :: D$ for p' a prefix, and in this situation the prefix pp' must, as we have seen, be legitimate, and thus A_{i+1} has been brought into the desired form. □

We have thus established the following:

Theorem 10 (Finite reachability). *For all well-formed A and n_0, $\{B \mid A \rightsquigarrow^* B\}$ is finite.*

5. Choosing names

The problem with Theorem 10 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 $|A|_1$ be the maximal number of free names in any subterm of A, and $|A|_2$ be the number of occurrences of the parallel combinator $|$ in A.

Lemma 11. *For all A_n, if $A_0 \to^* A_n$ then $|\mathrm{fn}(A_n)| \leqslant |A_0|_1 \cdot (|A_0|_2 + 1)$.*

Proof. Let ρ be an infinite derivation $\rho = A_0 \to \cdots \to A_n \to \cdots$. We proceed by induction in the structure of A_0.

$A_0 = (\lambda x)A_0'$. If $n > 0$ it must be the case that (forgetting about a possible initial sequence of alpha-conversions) $A_1 = A_0'\{y/x\}$ where y is small enough, such that $A_0' \to^* A_n$. By the induction hypothesis, $|\mathrm{fn}(A_n)| \leqslant |A_0'|_1 \cdot (|A_0'|_2 + 1)$ which in turn is equal to $|A_0|_1 \cdot (|A_0|_2 + 1)$.

$A_0 = A_{0,1} \mid A_{0,2}$. A_n must have the form $p :: (A_{n,1} \mid A_{n,2})$ where p is a legitimate prefix. For each $i \in \{1,2\}$ we find legitimate prefixes p_i such that $A_{0,i} \to^* p_i :: A_{n,i}$, 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, $|\mathrm{fn}(p_i :: A_{n,i})| \leqslant |A_{0,i}|_1 \cdot (|A_{0,i}|_2 + 1)$ for $i = 1$ and $i = 2$. Now $|\mathrm{fn}(A_n)| \leqslant |\mathrm{fn}(p_1 :: A_{n,1})| + |\mathrm{fn}(p_2 :: A_{n,2})|$. Let now B be whichever of $A_{0,1}/A_{0,2}$ such that $|B|_1$ is maximal. Then $|\mathrm{fn}(A_n)| \leqslant |B|_1 \cdot (|A_{0,1}|_2 + |A_{0,2}|_2 + 2)$, and then, since $|A_{0,1}|_2 + |A_{0,2}|_2 = |A_0|_2$, we obtain $|\mathrm{fn}(A_n)| \leqslant |A_0|_1 \cdot (|A_0|_2 + 1)$ as required.

$A_0 = \mathrm{fix}\,D.A_0'$. By the assumption of finite control $|A_0|_2 = 0$. Thus it suffices to show $|\mathrm{fn}(A_n)| \leqslant |A_0|_1$. We can assume that there are legitimate prefixes p and p' such that

$\mathrm{fix}\,D.A'_0 \to^* p :: \mathrm{fix}\,D.A \to^* A_n$, where A_n has the form $p' :: A'_n\{\mathrm{fix}\,D.A/D\}$, and $A'_0 \to^* A'_n$. Without loss of generality, we can assume that p has no free occurrences of names, since any free occurrences of a name in p would be eliminated before a subsequent unfolding of $\mathrm{fix}\,D.A'_0$. We know that $|\mathrm{fn}(A'_n)| \leqslant |A'_0|_1 = |A_0|_1$ by the induction hypothesis. The only case in which $|\mathrm{fn}(A_n)|$ could be greater than $|\mathrm{fn}(A'_n)|$ is when p' contains an occurrence of an output prefix $[y]$ such that that particular occurrence of y is free in p'. This, however, can only happen if the derivation $p :: \mathrm{fix}\,D.A'_0 \to^* A_n$ can be factorised in the manner $p :: \mathrm{fix}\,D.A'_0 \to^* p'' :: A''_n\{\mathrm{fix}\,D.A'_0/D\} \to^* A_n$ such that p'' has no free occurrence of y, and such that $A'_0 \to^* A''_n$. Moreover, $\mathrm{fn}(A''_n) = \mathrm{fn}(A'_n) \cup \{y\}$. But, by the induction hypothesis, $|\mathrm{fn}(A''_n)| \leqslant |A_0|_1$, which is what we need to show.

The remaining cases follow directly by the induction hypothesis. □

6. Decidability

Consider now a version of pr-bisimulation of Definition 2 modified such that z in 2.2 and 2.3 is required to be small enough, and such that commitment in 2.4 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 11 any infinite path must visit the same agent expression infinitely often, the following decidability result obtains:

Theorem 12. *Name-bounded pr-bisimulation equivalence is decidable.*

Using this result we can then easily establish our first main theorem.

Theorem 13. *Strong late bisimulation equivalence is decidable.*

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

$$|A|_1 \cdot (|A|_2 + 1) + |B|_1 \cdot (|B|_2 + 1).$$

Let a *name representation* be any pair of maps $(f_{\mathrm{free}}, f_{\mathrm{bound}})$ 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

(i) if x and y are distinct, both occurs freely in some subterm of A or B, and both are occurrences of bound names, then $f_{\mathrm{bound}}(x) \neq f_{\mathrm{bound}}(y)$, and

(ii) 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 $f_{\mathrm{bound}}(x) \neq f_{\mathrm{free}}(y)$.

For a name partition ε let $f_{\text{free}}(\varepsilon) = \{\{f(x) \mid x \in U\} \mid U \in \varepsilon\}$. 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 $(f_{\text{free}}, f_{\text{bound}})$ such that if A' and B' are the agents resulting from renaming free and bound names according to $(f_{\text{free}}, f_{\text{bound}})$, then A' and B' are $f_{\text{free}}(\varepsilon)$-nbpr-bisimulations. It is then easy to verify that, due to the choice of n_0, S_ε is an ε-pr-bisimulation. This completes the proof. □

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 Definition 2 (cf. [9] for a definition of early equivalence). The proofs of Theorems 12 and 13 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 14. (i) *Strong early bisimulation equivalence is decidable.*
(ii) *Weak late and early bisimulation equivalence are decidable.*

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 [12, 5] 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, ε). 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 $\mathcal{O}(m^2n^22^{n^2})$. Thus the running time of the Paige–Tarjan algorithm is bounded by $\mathcal{O}(n^42^{n^2}m^2 \log m)$. 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 $\mathcal{O}(n2^{n^2 \log(n^2)})$. Thus m is bounded by $\mathcal{O}((n2^{n^2 \log(n^2)})^n) = \mathcal{O}(2^{n^3 \log(n^2) + n \log n})$, 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 π-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 [4] 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 π-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 [14], or the symbolic bisimulations of Hennessy and Lin [3]. For the weak equivalences in particular we expect such efficient characterisations to be indispensable.

References

[1] R. Amadio, A uniform presentation of CHOCS and π-calculus, Rapport de Recherche 1726, INRIA-Lorraine, Nancy, 1992.
[2] M. Dam, Model checking mobile processes, *Inform. Comput.* **129** (1996) 35–51.
[3] M. Hennessy and H. Lin, Symbolic bisimulations, Report 1/92, Dept. of Computer Science, University of Sussex, 1992.
[4] B. Jonsson and J. Parrow, Deciding bisimulation equivalences for a class of non-finite-state programs, *Inform. Comput.* **107** (1993) 272–302.
[5] P.C. Kannellakis and S.A. Smolka, CCS expressions, finite state processes, and three problems of equivalence, *Inform. Comput.* **86** (1990) 43–68.
[6] R. Milner, The polyadic π-calculus: a tutorial, Tech. Report ECS-LFCS-91-180, Laboratory for the Foundations of Computer Science, Department of Computer Science, University of Edinburgh, 1991.
[7] R. Milner, Functions as processes, *Math. Struct. Comput. Sci.* **2** (1992) 119–141.
[8] R. Milner, J. Parrow and D. Walker, A calculus of mobile processes, I and II, *Inform. Comput.* **100** (1992) 1–40 and 41–77.
[9] R. Milner, J. Parrow and D. Walker, Modal logics for mobile processes, *Theoret. Comput. Sci.* **114** (1993) 149–171.
[10] F. Orava, On the formal analysis of telecommunication protocols, Ph.D. Thesis, Dept. of Computer Systems, Uppsala University and Swedish Institute of Computer Science, 1994.
[11] F. Orava and J. Parrow, An algebraic verification of a mobile network, *Formal Aspects Comput.* **4** (1992) 497–543.
[12] R. Paige and R.E. Tarjan, Three partition refinement algorithms, *SIAM J. Comput.* **16** (1987) 973–989.
[13] D. Sangiorgi, From π-calculus to higher-order π-calculus – and back, in: *Proc. TAPSOFT'93*, Lecture Notes in Computer Science, Vol. 668 (Springer, Berlin, 1993).
[14] D. Sangiorgi, A theory of bisimulation for the π-calculus, in: *Proc. CONCUR'93*, Lecture Notes in Computer Science, Vol. 715 (Springer, Berlin, 1993) 127–142.
[15] D. Walker, Objects in the π-calculus, *Inform. Comput.* **116** (1995) 253–271.