Leader Election in Rings of Ambient Processes [1]

Iain Phillips [*], Maria Grazia Vigliotti

Department of Computing
Imperial College London
London SW7 2AZ, England

Abstract

Palamidessi has shown that the π-calculus with mixed choice is powerful enough to solve the leader election problem on a symmetric ring of processes. We show that this is also possible in the calculus of Mobile Ambients (MA), without using communication or restriction. Following Palamidessi's methods, we deduce that there is no encoding satisfying certain conditions from MA into CCS. We also show that the calculus of Boxed Ambients is more expressive than its communication-free fragment.

Key words: Ambient calculi, leader election, electoral system, ring, expressiveness

1 Introduction

The π-calculus [14] is a simple, yet extremely powerful, formalism that models concurrency and the passing around of resources that can later be used by other processes. It is based on a very simple and uniform concept of *names*. Names are both *channels*, on which communication takes place, and *values*, i.e. the resources passed around. Names sent as values can be used later as channels for communication. This particular feature seems unique to the π-calculus, and allows processes to establish connection during computation.

[*] Corresponding author. Tel:+44 (0)20 7594 8265 Fax:+44 (0)20 7581 8024
Email addresses: `iccp@doc.ic.ac.uk` (Iain Phillips), `mgv98@doc.ic.ac.uk` (Maria Grazia Vigliotti).
[1] A shorter version of this paper appeared in Express 2004 [18].
[2] Email: {iccp,mgv98}@doc.imperial.ac.uk

This seems to add extra power that was not previously available in CCS [12] or other similar calculi such as CSP [8] or ACP [2].

In fact, while CCS (with value-passing) may be regarded as a subcalculus of the π-calculus, Palamidessi [15] has shown that (under certain conditions) there exists no encoding from the π-calculus to CCS by exploiting the possibility of creating new connections in the π-calculus. Palamidessi establishes her result within the framework of the leader election problem. Leader election problems arise in the field of distributed systems, where they are widely studied for practical reasons, and are also used to differentiate models of computation. The problem is stated as follows: given a symmetric network, a composition of processes that differ in their free variables only, one process has to be elected a leader without the help of a centralised server.

CCS and the π-calculus with mixed choice (where input and output can occur in the choice operator together) can both elect a leader in a fully connected symmetric network. This differentiates these two calculi from the π-calculus with separate choice (meaning that inputs and outputs cannot be mixed in the same choice), where such election is not possible. Moreover, the π-calculus can also solve the problem of electing a leader in a symmetric *ring* of processes, in other words, in a network where each process is connected only to its two neighbours in the ring. Palamidessi's algorithm, which works on rings of any size, has two phases. In phase one the processes pass names around the ring so that every process becomes directly connected to every other process. Here there is an essential use of the π-calculus, though without any use of choice. CCS would not do, since it cannot increase connectivity. In the second phase the processes elect a leader. Here there is an essential use of mixed choice, but CCS would suffice, rather than the π-calculus. Building on the work of Angluin [1] and Bougé [4], Palamidessi proves that CCS cannot perform leader election on symmetric rings of composite (i.e. non-prime) size, by showing that there is a maximal computation where symmetry is never broken, so that no single leader emerges. She deduces that there is no encoding (of a certain kind) from the π-calculus with mixed choice into CCS.

In the present work we explore how Palamidessi's techniques apply to rings in ambient calculi, studying how new connections between processes can be established. In previous work [17], we have shown that in Mobile Ambients (MA) [6] without the communication primitives, the open capability and restriction, the leader election problem can be solved in a fully-connected symmetric network. We might call this fragment of MA the *minimal fragment*. This fragment, which is also a sub-calculus of Boxed Ambients (BA) [5], is choice-free; the solution to electing a leader in symmetric network is achieved through the pre-emptive power of migration inside ambients [17,20]. The communication primitives of MA have the same operational semantics as the π-calculus, except that they are *anonymous*, in the sense that there are no channels on which communication happens (in the π-calculus one would write $a(x).P$ for an input on the channel a, while in MA one would write $(x).P$ for an anonymous input). Thus, since the communication primitives in ambients are very similar to those of the π-calculus, it would be not surprising if Palamidessi's algorithm for rings could be formulated in MA. However, in this paper we solve the leader election problem for symmetric rings of any size in *pure public* MA, i.e. MA without communication primitives and restriction (Theorem 4.3). The link-passing in this case has to be simulated, since there is no explicit way of passing names in the absence of communication. This yields immediately the result that pure public MA cannot be encoded into CCS under certain conditions (Theorem 6.4).

The second major result that we present here is that, even if we add communication and restriction to the minimal fragment of MA to form *boxed* MA, i.e. MA without the open capability, the leader election problem cannot be solved in symmetric rings of composite (i.e. non-prime) size (Theorem 5.3). This clearly shows that in MA, the open capability (but not communication) is crucial in order to pass resources around. In connection with our results, we recall that Zimmer [21] proved that the synchronous choice-free π-calculus can be encoded into pure Safe Ambients (SA) [9], showing that link-passing can be simulated in pure SA. The encoding uses the open capability. Thus the open capability seems quite powerful. This is in contrast with other expressiveness results based on Turing completeness [10,3], where it was shown that the open capability is not crucial, since the minimal fragment is still Turing-complete.

We easily adapt our results on MA to the setting of SA: leader election can be performed on rings of any size in pure public SA (Corollary 4.6), but it is impossible for rings of composite size in boxed SA (Theorem 5.21).

The situation is different for BA, where the open capability is missing as a design choice. Communications between parent and child ambients are allowed, and the synchronous choice-free π-calculus can be encoded, and with that, clearly, the power of creating new links. Thus we can show (Theorem 4.7) that in BA the leader election problem in a ring of any size can be solved by converting the ring into a fully-connected network and then using the algorithm of [17]. However, *pure* BA (i.e. BA without communication) is a subcalculus of boxed MA; it is therefore less expressive than full BA, in view of our result that only with the presence of the open capability can MA elect a leader in rings of composite size.

In distributed systems, leader election problems are categorised according to the connectivity of the network, the knowledge of the size of the network and the methods of election. In this paper we present a solution in MA for a ring of any size, providing that the processes are given information about the size of the ring. Palamidessi's algorithm also uses this information. However, for the

Any size	Not composite size
pure public MA (Thm. 4.3)	boxed MA (Thm. 5.3)
pure public SA (Cor. 4.6)	boxed SA (Thm. 5.21)
public BA (Thm. 4.7)	pure BA (Thm. 5.3)
pure public PAC (uniform solution) (Thm. 4.2)	boxed PAC (Thm. 5.21)

Fig. 1. Summary of leader election results on rings

Push and Pull Ambient Calculus [16], we present a solution for rings of any size where the processes do not know the size of the ring (Theorem 4.2). Thus a single uniform solution will work for any size of ring. As far as we know, this is the first time that such an algorithm has been devised in the setting of process calculi. It remains for future work to find suitable conditions that differentiate those calculi that admit a solution to the leader election problem without having to know the size of the ring from those that do need to know the size.

We summarise our results on leader election on rings in Figure 1. The left-hand column lists those calculi where the leader election problem can be solved for symmetric rings of any size; the right-hand column lists those calculi where the leader election problem cannot be solved for symmetric rings of composite size. For our results on encodings, see Figure 2 at the end of Section 6.

The rest of the paper is structured as follows. In Section 2 we describe the calculi we are considering, and in Section 3 we discuss electoral systems. In Section 4 we consider calculi which admit symmetric electoral systems of rings of processes, while in Section 5 we consider calculi that do not admit symmetric electoral systems for certain rings. In Section 6 we examine the consequences of our results for expressiveness of ambient calculi. Finally we draw some conclusions in Section 7.

2 Calculi

In this section we recall Cardelli and Gordon's Mobile Ambients (MA) and related calculi, as well as Milner's CCS with value passing. As in previous work [17,20], we shall use reduction semantics, as opposed to the more traditional labelled transition semantics. We prefer this setting, since it is a uniform way of describing calculi with different primitives. It is also the standard semantics for MA, where it is unclear what labelled transition system should be preferred.

We follow [6], except for communication, as noted below. Let $P, Q, \ldots$ range over processes and $M, \ldots$ over capabilities. We assume a set of names $\mathcal{N}$, ranged over by $m, n, \ldots$. Processes are defined as follows:

$$P, Q ::= \mathbf{0} \mid P \mid Q \mid \nu n\, P \mid !\, P \mid n[\, P\,] \mid M.P \mid (n).P \mid \langle n \rangle$$

Here $\mathbf{0}$ is the nil process which is inactive; $P \mid Q$ is the parallel composition of processes P and Q; $\nu n\, P$ is P with name n restricted; $!\, P$ (replication) is a process which can spin off as many copies of P as are required; $n[\, P\,]$ is an ambient named n containing process P; $M.P$ performs capability M before continuing as P; and $(n).P$ receives input on an anonymous channel, with the input name replacing free occurrences of name n in P; and finally $\langle n \rangle$ is a process which outputs name n. Notice that output is *asynchronous*, that is, it has no continuation. Restriction and input are name-binding. We let $\mathsf{fn}(P)$ denote the set of free names of P. We omit trailing $\mathbf{0}$s and write $n[\,]$ instead of $n[\,\mathbf{0}\,]$.

Capabilities are defined as follows:

$$M ::= \mathsf{in}\, n \mid \mathsf{out}\, n \mid \mathsf{open}\, n$$

Capabilities allow movement of ambients ($\mathsf{in}\, n$ and $\mathsf{out}\, n$) and dissolution of ambients ($\mathsf{open}\, n$).

We confine ourselves in this paper to communication of names, rather than full communication including capabilities (as in [6]). This serves to streamline the presentation; the results would also hold for full communication.

Structural congruence $\equiv$ allows rearrangement of processes; it is the least congruence generated by the following laws:

$$\begin{array}{ll}
P \mid Q \equiv Q \mid P & \nu n\, \nu m\, P \equiv \nu m\, \nu n\, P \\
(P \mid Q) \mid R \equiv P \mid (Q \mid R) & \nu n\, (P \mid Q) \equiv P \mid \nu n\, Q \quad \text{if } n \notin \mathsf{fn}(P) \\
P \mid \mathbf{0} \equiv P & \nu n\, m[\, P\,] \equiv m[\, \nu n\, P\,] \quad \text{if } n \neq m \\
!\, P \equiv P \mid !\, P & \nu n\, \mathbf{0} \equiv \mathbf{0} \\
!\, \mathbf{0} \equiv \mathbf{0} &
\end{array}$$

together with α-conversion of bound names.

The *reduction* relation $\to$ is generated by the following rules:

$$\begin{array}{ll}
\text{(In)} & n[\,\mathsf{in}\, m.P \mid Q\,] \mid m[\,R\,] \to m[\,n[\,P \mid Q\,] \mid R\,] \\
\text{(Out)} & m[\,n[\,\mathsf{out}\, m.P \mid Q\,] \mid R\,] \to n[\,P \mid Q\,] \mid m[\,R\,] \\
\text{(Open)} & \mathsf{open}\, n.P \mid n[\,Q\,] \to P \mid Q \\
\text{(Comm)} & \langle m' \rangle \mid (m).P \to P\{m'/m\}
\end{array}$$

$$\text{(Amb)}\ \frac{P \to P'}{n[\,P\,] \to n[\,P'\,]} \qquad \text{(Par)}\ \frac{P \to P'}{P \mid Q \to P' \mid Q}$$

$$\text{(Res)}\ \frac{P \to P'}{\nu n\, P \to \nu n\, P'} \qquad \text{(Str)}\ \frac{P \equiv P' \quad P' \to Q' \quad Q' \equiv Q}{P \to Q}$$

Here $P\{m'/m\}$ denotes P with m' substituted for every free occurrence of m. Notice that movement in MA is *subjective*: ambients move themselves using the in and out capabilities. We write $\twoheadrightarrow$ for the reflexive and transitive closure of $\to$. The notation $P \not\to$ means that there does not exist a process to which P can reduce.

The most basic observation we can make of an MA process is the presence of an unrestricted top-level ambient. A process P *exhibits barb* n, written as $P \downarrow n$, iff $P \equiv \nu\vec{m}(n[\,Q\,] \mid R)$ with $n \notin \vec{m}$. Here $\vec{m}$ represents a tuple of names. A process P *eventually exhibits barb* n, written $P \Downarrow n$, iff $P \twoheadrightarrow Q$ and $Q \downarrow n$ for some Q.

We shall be interested in various subcalculi: *pure* MA is MA without communication; *public* MA is MA without restriction; and *boxed* MA is MA without the open capability.

2.2 Safe Ambients

The calculus of Safe Ambients (SA) [9] is a variant of MA where new *co-capabilities* are added to complement the existing in, out and open capabilities. The syntax of processes is the same as for MA. Capabilities are defined as follows:

$$M ::= \mathsf{in}\, n \mid \mathsf{out}\, n \mid \mathsf{open}\, n \mid \overline{\mathsf{in}}\, n \mid \overline{\mathsf{out}}\, n \mid \overline{\mathsf{open}}\, n$$

Structural congruence and the *reduction* relation $\to$ are defined as for MA, except that rules (In), (Out) and (Open) are replaced by the following:

$$\begin{array}{ll}
\text{(CoIn)} & n[\,\mathsf{in}\, m.P \mid Q\,] \mid m[\,\overline{\mathsf{in}}\, m.R \mid S\,] \to m[\,n[\,P \mid Q\,] \mid R \mid S\,] \\
\text{(CoOut)} & m[\,n[\,\mathsf{out}\, m.P \mid Q\,] \mid \overline{\mathsf{out}}\, m.R \mid S\,] \to n[\,P \mid Q\,] \mid m[\,R \mid S\,] \\
\text{(CoOpen)} & \mathsf{open}\, n.P \mid n[\,\overline{\mathsf{open}}\, n.Q \mid R\,] \to P \mid Q \mid R
\end{array}$$

Barbs are defined slightly differently from MA. A process P *exhibits barb* n, written as $P \downarrow n$, iff $P \equiv \nu\vec{m}\,(n[\,M.Q \mid R\,] \mid S)$ with $n \notin \vec{m}$ and M either $\overline{\mathsf{in}}\,n$ or $\overline{\mathsf{open}}\,n$.

2.3 The Push and Pull Ambient Calculus

The Push and Pull Ambient Calculus (PAC) [16,20] is a variant of MA where the subjective moves enabled by the in and out capabilities are replaced by objective moves whereby ambients can be pulled in or pushed out by other ambients. The syntax of processes is the same as for MA. Capabilities are defined as follows:

$$M ::= \mathsf{pull}\,n \;\mid\; \mathsf{push}\,n \;\mid\; \mathsf{open}\,n$$

The reduction rules are the same as for MA, except that (In) and (Out) are replaced by the following:

$$\begin{array}{ll}(\text{Pull}) & n[\,\mathsf{pull}\,m.P \mid Q\,] \mid m[\,R\,] \rightarrow n[\,P \mid Q \mid m[\,R\,]\,] \\ (\text{Push}) & n[\,m[\,P\,] \mid \mathsf{push}\,m.Q \mid R\,] \rightarrow n[\,Q \mid R\,] \mid m[\,P\,]\end{array}$$

Barbs are defined as for MA.

2.4 Boxed Ambients

The calculus of Boxed Ambients (BA) [5] is derived from MA by removing the open capability and allowing parent-child communication as well as same-level communication. Processes are defined as follows:

$$P, Q ::= \mathbf{0} \;\mid\; P \mid Q \;\mid\; \nu n\, P \;\mid\; !\,P \;\mid\; n[\,P\,] \;\mid\; M.P \;\mid\; (\vec{n})^{\eta}.P \;\mid\; \langle\vec{n}\rangle^{\eta}.P$$

Here $\vec{n}$ denotes a tuple of names, and η ranges over *locations*, defined as follows:

$$\eta ::= n \;\mid \uparrow \mid\; \star$$

The "local" location $\star$ is elided. Notice that output $\langle\vec{n}\rangle^{\eta}.P$ is synchronous, unlike in MA. Capabilities M are defined as for MA but without open. The reduction rules are the same as for boxed MA, except for communication, where the rule (Comm) is replaced by the following five rules:

$$\begin{array}{ll}(\text{Local}) & (\vec{m}).P \mid \langle\vec{m'}\rangle.Q \rightarrow P\{\vec{m'}/\vec{m}\} \mid Q \\ (\text{Input } n) & (\vec{m})^{n}.P \mid n[\,\langle\vec{m'}\rangle.Q \mid R\,] \rightarrow P\{\vec{m'}/\vec{m}\} \mid n[\,Q \mid R\,] \\ (\text{Input } \uparrow) & n[\,(\vec{m})^{\uparrow}.P \mid Q\,] \mid \langle\vec{m'}\rangle.R \rightarrow n[\,P\{\vec{m'}/\vec{m}\} \mid Q\,] \mid R \\ (\text{Output } n) & n[\,(\vec{m}).P \mid Q\,] \mid \langle\vec{m'}\rangle^{n}.R \rightarrow n[\,P\{\vec{m'}/\vec{m}\} \mid Q\,] \mid R \\ (\text{Output } \uparrow) & (\vec{m}).P \mid n[\,\langle\vec{m'}\rangle^{\uparrow}.Q \mid R\,] \rightarrow P\{\vec{m'}/\vec{m}\} \mid n[\,Q \mid R\,]\end{array}$$

Clearly, rule (Local) extends rule (Comm), so that communication in BA is at least as powerful as communication in MA. Note that *pure* BA (i.e. BA without communication) is the same as pure boxed MA.

Barbs are defined as for MA.

2.5 CCS

In this paper we shall use the version of CCS presented in [13], with the addition of value-passing. As well as names $n \in \mathcal{N}$, we use co-names $\bar{n} \in \overline{\mathcal{N}}$, a set $\mathcal{V}$ of values, ranged over by $v, \ldots$, and a set $\mathcal{W}$ of variables, ranged over by $x, \ldots$. The sets $\mathcal{N}$, $\overline{\mathcal{N}}$, $\mathcal{V}$ and $\mathcal{W}$ are mutually disjoint. *Prefixes* are defined by

$$\pi ::= n(x) \;\mid\; \bar{n}\langle v\rangle \;\mid\; \tau$$

(where τ is the silent action) and *summations* are defined by

$$G, H ::= \sum_{i \in I} \pi_i.P_i$$

(where I is a finite set). The empty summation is denoted by $\mathbf{0}$. Processes are defined as follows:

$$P, Q ::= G \;\mid\; P \mid Q \;\mid\; \nu n\, P \;\mid\; A\langle a_1, \ldots, a_k\rangle$$

Here recursion is handled by process identifiers with parameters; each identifier A is equipped with a defining equation $A(\vec{a}) \overset{\text{df}}{=} P_A$.

CCS is usually presented with a labelled transition system, but in this paper we use unlabelled transitions (reduction semantics) as a uniform framework. So we follow the reduction rules given in [13]. Structural congruence has the same laws as for MA, except that we omit the rule for ambient, we allow reordering of summations, and we add the law

$$A\langle\vec{b}\rangle \equiv P_A\{\vec{b}/\vec{a}\} \text{ if } A(\vec{a}) \overset{\text{df}}{=} P_A \ .$$

The reduction relation has the rules

$$\tau.P + G \to P \qquad (n(x).P + G) \mid (\bar{n}\langle v\rangle.Q + H) \to P\{v/x\} \mid Q$$

together with rules (Par), (Res) and (Str) as given for MA.

Barbs are much as for MA. A process P *exhibits barb* n, written as $P \downarrow n$, iff $P \equiv \nu\vec{m}\,(\bar{n}\langle x\rangle.Q \mid R)$ with $n \notin \vec{m}$. We only use barbs on outputs; input barbs are not needed, and we thereby obtain greater uniformity across the calculi we are considering.

3 Electoral Systems and Rings

In this section we present a general definition of network, ring and electoral system. As in previous work [17,20], we base everything on reduction semantics, so that our framework could be applied to any calculus equipped with reduction semantics. In this respect we differ from Palamidessi [15] and Bougé [4], who worked in a labelled transition system framework. All the notions of this section apply equally to MA, SA, PAC, BA and CCS.

Networks are informally compositions of processes; the size of the networks is the number of processes that can be "regarded as separate units". This means that a composition of processes can be seen as one process only in counting the size of the network. A *symmetric* network is a network where components differ only on their names. Components of a network are connected if they share names, using which they can engage in communication. *Rings* are networks where each process is connected just to its left-hand and right-hand neighbours. A network elects a leader by exhibiting a special name, and an *electoral system* is a network where every possible maximal computation elects a leader. Notice that some of these definitions were already used in our previous work [17,18], yet they are still necessary here. The notion of ring in process calculi is however novel here, and was not present in Palamidessi's work or in Bougé's.

3.1 Networks and Electoral Systems

We briefly recall electoral systems as formulated in [17], building on [15]. We assume that $\mathcal{N}$ includes a set of *observables* $\mathsf{Obs} = \{\omega_i : i \in \mathbb{N}\}$ such that for all i, j we have $\omega_i \neq \omega_j$ if $i \neq j$. The observables will be used by networks to communicate with the outside world.

Definition 3.1 *Let P be a process. A* computation *$\mathcal{C}$ of P is a (finite or infinite) sequence $P = P_0 \to P_1 \to \cdots$. It is* maximal *if it cannot be extended, i.e. either $\mathcal{C}$ is infinite, or else it is of the form $P_0 \to \cdots \to P_h$ where $P_h \not\to$.*

Definition 3.2 *Let $\mathcal{C}$ be a computation $P_0 \to \cdots \to P_h \to \cdots$. We define the* observables *of $\mathcal{C}$ as $\mathsf{Obs}(\mathcal{C}) = \{\omega \in \mathsf{Obs} : \ \exists h \ \ P_h \downarrow \omega\}$.*

Networks are collections of processes running in parallel:

Definition 3.3 *(cf. [15]) A* network Net *of size k is a pair $(A, \langle P_0, \ldots, P_{k-1} \rangle)$, where A is a finite set of names and $P_0, \ldots, P_{k-1}$ are processes. The* process interpretation $\mathsf{Net}^\natural$ *of* Net *is the process $\nu A\, (P_0 \mid \ldots \mid P_{k-1})$. We shall always work up to structural congruence, so that the order in which the restrictions* *in A are applied is immaterial.*

Networks are to be seen as presentations of processes, showing how the global process is distributed to the k nodes of the network. Trivially, any process P can be presented as a network of size k for any k: the network $(\emptyset, \langle P, \mathbf{0}, \dots, \mathbf{0}\rangle)$ has the process interpretation $P \mid \mathbf{0} \mid \cdots \mid \mathbf{0}$, which is structurally congruent to P. However, networks become more interesting once we impose conditions of symmetry.

We shall tend to write networks in their process interpretation (i.e. as restricted parallel compositions), while still making it clear which process belongs to each node of the network.

Networks inherit a notion of computation from processes through the process interpretation: $\mathsf{Net} \to \mathsf{Net}'$ if $\mathsf{Net}^{\natural} \to \mathsf{Net}'^{\natural}$. Overloading notation, we shall let $\mathcal{C}$ range over network computations. Also, we define the observables of a network computation $\mathcal{C}$ to be the observables of the corresponding process computation: $\mathsf{Obs}(\mathcal{C}) = \mathsf{Obs}(\mathcal{C}^{\natural})$.

During the course of a network computation we do not require the resulting networks to have the same size as the original. This is in contrast to the approach taken in [15]. Such variation in size is a natural feature of ambient languages: the network $n[\,\mathsf{in}\, m\,] \mid m[\;]$ of size two reduces to $m[\,n[\;]\,]$, which is a network of size one. Of course if we wish we can pad the derivative out with $\mathbf{0}$ to retain a network of size two.

Definition 3.4 *A* permutation *is a bijection $\sigma : \mathcal{N} \to \mathcal{N}$ such that σ preserves the distinction between observable and non-observable names, i.e. $n \in \mathsf{Obs}$ iff $\sigma(n) \in \mathsf{Obs}$. Any permutation σ gives rise in a standard way to a mapping on processes, where $\sigma(P)$ is the same as P, except that any free name n of P is changed to $\sigma(n)$ in $\sigma(P)$, with bound names being adjusted as necessary to avoid clashes.*

A permutation σ induces a bijection $\hat{\sigma} : \mathbb{N} \to \mathbb{N}$ defined as follows: $\hat{\sigma}(i) = j$ where $\sigma(\omega_i) = \omega_j$. Thus for all $i \in \mathbb{N}$, $\sigma(\omega_i) = \omega_{\hat{\sigma}(i)}$. We use $\hat{\sigma}$ to permute the indices of processes in a network.

Definition 3.5 *Let $\mathsf{Net} = \nu\vec{n}(P_0 \mid \dots \mid P_{k-1})$ be a network of size k. An* automorphism *on Net is a permutation σ such that (1) $\hat{\sigma}$ restricted to $\{0, \dots, k-1\}$ is a bijection, and (2) σ preserves the distinction between free and bound names, i.e. $n \in \vec{n}$ iff $\sigma(n) \in \vec{n}$. If $\hat{\sigma}$ restricted to $\{0, \dots, k-1\}$ is not the identity we say σ is* non-trivial.

Definition 3.6 *Let σ be an automorphism on a network of size k. For any*

$i \in \{0, \ldots, k-1\}$ *the* orbit $\mathcal{O}_{\hat{\sigma}}(i)$ *generated by* $\hat{\sigma}$ *is defined as follows:*

$$\mathcal{O}_{\hat{\sigma}}(i) = \{i, \hat{\sigma}(i), \hat{\sigma}^2(i), \ldots, \hat{\sigma}^{h-1}(i)\}$$

where $\hat{\sigma}^j$ *represents the composition of* $\hat{\sigma}$ *with itself* j *times, and* h *is least such that* $\hat{\sigma}^h(i) = i$*. If every orbit has the same size then* σ *is* well-balanced.

Definition 3.7 *Let* $\mathsf{Net} = \nu\vec{n}\,(P_0 \mid \ldots \mid P_{k-1})$ *be a network of size* k *and let* σ *be an automorphism on it. We say that* Net *is* symmetric with respect to σ *iff for each* $i = 0, \ldots, k-1$ *we have* $P_{\hat{\sigma}(i)} = \sigma(P_i)$*. We say that* Net *is* symmetric *if it is symmetric with respect to some automorphism with a single orbit (which must have size* k*).*

Intuitively an electoral system is a network which reports a unique winner, no matter how the computation proceeds.

Definition 3.8 *A network* Net *of size* k *is an* electoral system *if for every maximal computation* $\mathcal{C}$ *of* Net *there exists an* $i < k$ *such that* $\mathsf{Obs}(\mathcal{C}) = \{\omega_i\}$*.*

3.2 Rings and Independence

In this paper we are interested in the connectivity between processes, and in rings of processes in particular. Given a network $\mathsf{Net} = \nu\vec{n}\,(P_0 \mid \cdots \mid P_{k-1})$, we can associate a graph with Net by letting the set of nodes be $\{0, \ldots, k-1\}$ and letting $i, j < k$ be adjacent iff $\mathsf{fn}(P_i) \cap \mathsf{fn}(P_j) \neq \emptyset$. A network forms a ring if the processes can be arranged in a cycle, and each node i is adjacent to at most its two neighbours in the cycle.

Definition 3.9 *A* ring *is a network* $\mathsf{Net} = \nu\vec{n}\,(P_0 \mid \cdots \mid P_{k-1})$ *which has a single-orbit automorphism* σ *such that for all* $i, j < k$*, if* $\mathsf{fn}(P_i) \cap \mathsf{fn}(P_j) \neq \emptyset$ *then one of* $i = j$*,* $\hat{\sigma}(i) = j$ *or* $\hat{\sigma}(j) = i$ *must hold. A ring is* symmetric *if it is symmetric with respect to such an automorphism* σ*.*

Notice that the definition bans links between non-adjacent nodes in the ring, but does not require the existence of links between adjacent nodes. Thus a completely disconnected network is a ring.

Recall that an *independent set* in a graph is a set of nodes such that no two nodes of the set are adjacent.

Definition 3.10 *Two processes* P *and* Q *are* independent *if they do not share any free names:* $\mathsf{fn}(P) \cap \mathsf{fn}(Q) = \emptyset$*.*

Definition 3.11 *Let* σ *be an automorphism on a network* $\mathsf{Net} = \nu\vec{n}\,(P_0 \mid \cdots \mid P_{k-1})$*. Then* Net *is* independent *with respect to* σ *if every orbit forms an* *independent set, in the sense that if $i, j < k$ are in the same orbit of $\hat{\sigma}$ with $i \neq j$, then P_i and P_j are independent.*

Bougé [4] and Palamidessi [15] showed the non-existence of electoral systems in networks equipped with a well-balanced non-trivial automorphism with independent orbits. We shall require a somewhat stronger condition for our non-existence results in Section 5.

We first note that rings do not necessarily satisfy the Bougé-Palamidessi condition; neither are networks satisfying the Bougé-Palamidessi condition necessarily rings. A ring of k processes (where k is prime and where at least one link between adjacent nodes exists) does not have a well-balanced non-trivial automorphism with independent orbits. This is because any non-trivial automorphism must have an orbit of size > 1. If it is well-balanced then, since k is prime, there must be a single orbit of size k. But this orbit is not independent. Conversely, the network of 6 processes corresponding to the graph $K_{3,3}$ (the complete bipartite graph on two sets of 3 nodes) satisfies the Bougé-Palamidessi condition, but is not a ring.

For our non-existence results we shall restrict to rings of composite (non-prime) size, where the Bougé-Palamidessi condition does hold, as the next lemma shows.

Lemma 3.12 *Let* Net *be a symmetric ring of size k, where k is composite. Then there is a well-balanced non-trivial automorphism σ such that* Net *is symmetric and independent with respect to σ.*

PROOF. Let $k = rs$ where $r, s \geq 2$. Since Net is a symmetric ring, let τ be an automorphism as in Definition 3.9. Let $\sigma = \tau^r$. Then σ is non-trivial and Net is symmetric with respect to σ. Also, each orbit of σ has size s, so that σ is well-balanced. Finally, each orbit of σ is independent. □

4 Calculi with Electoral Systems for Rings

In this section we show that we can solve leader election on symmetric rings in ambient calculi. We present solutions for MA, BA and PAC. The solution for MA can be carried over to SA by a standard encoding. There is a fundamental difference between the solution for PAC and those for MA and BA: the PAC solution works for a ring of any size with a single uniform definition of each component, so that the processes do not need to know the size of the ring.

We will start with PAC, since the symmetric ring for PAC is the simplest to define. First we introduce some notation which will be helpful in the proofs of

Theorems 4.2 and 4.3.

Definition 4.1 *Let $k \geq 1$. Let $i, j \in \{0, \ldots, k-1\}$.*

- *By* $(i \mathrel{..} j) \bmod k$ *we mean the set of all numbers between i and j, going round the numbers in $\{0, \ldots, k-1\}$ cyclically modulo k in ascending order, and* excluding *i and j. More formally:*

$$(i \mathrel{..} j) \bmod k \overset{\text{df}}{=} \begin{cases} \{h : i < h < j\} & \text{if } i < j \\ \{h : i < h \leq k-1 \text{ or } 0 \leq h < j\} & \text{if } i \geq j \end{cases}$$

In particular, $(i \mathrel{..} i{+}1) \bmod k = \emptyset$ *and* $(i \mathrel{..} i) \bmod k = \{0, \ldots, k{-}1\} - \{i\}$.

- *Let* $[i \mathrel{..} j] \bmod k$ *be the numbers going from i to j cyclically mod k* including *i and j. Formally*

$$[i \mathrel{..} j] \bmod k \overset{\text{df}}{=} \begin{cases} \{h : i \leq h \leq j\} & \text{if } i \leq j \\ \{h : i \leq h \leq k-1 \text{ or } 0 \leq h \leq j\} & \text{if } i > j \end{cases}$$

Note that $[i \mathrel{..} i] \bmod k = \{i\}$.

We shall tend to suppress the mod k *and write* $(i \mathrel{..} j)$, $[i \mathrel{..} j]$.

4.1 Pure Public PAC

We show that using push and pull we can build a symmetric ring of processes which can elect a leader. Moreover, the construction is such that individual processes do not know the size of the ring.

Theorem 4.2 *For any $k \geq 1$, there is a symmetric ring of size k which is an electoral system in pure public PAC.*

PROOF. Let $k \geq 1$. For $i = 0, \ldots, k-1$, let P_i be defined as follows:

$$P_i \overset{\text{df}}{=} n_i[\, Q_i \mid \mathsf{pull}\, n_{i+1} \mid \mathsf{open}\, n_{i+1}\,]$$

where

$$Q_i \overset{\text{df}}{=} n_i[\, \omega_i[\,]\,] \mid \mathsf{push}\, \omega_i$$

We have $\mathsf{fn}(P_i) = \{n_i, n_{i+1}, \omega_i\}$. Let $\mathsf{Net} \overset{\text{df}}{=} P_0 \mid \cdots \mid P_{k-1}$. Note that Net belongs to pure public PAC—there is no use of communication or restriction. Moreover, the construction of P_i does not depend on k. It is now easy to see that Net forms a symmetric ring: define an automorphism σ by $\sigma(n_i) = n_{i+1}$, $\sigma(\omega_i) = \omega_{i+1}$ (with addition modulo k). Then σ has a single orbit and Net is symmetric with respect to σ. Also, if $\mathsf{fn}(P_i) \cap \mathsf{fn}(P_j) \neq \emptyset$ then $i = j$ or $j = i+1$ or $i = j+1$.

We claim that Net forms an electoral system. The idea is that a process P_i can pull in its right-hand neighbour P_j and open it. The neighbour P_j thereby loses and drops out of the ring, which now has one fewer process. Process P_i is now joined to P_j's right-hand neighbour. Eventually only one process is left. It will have the form:

$$(\star) \quad n_i[\, \prod_{j<k} Q_j \mid \mathsf{pull}\, n_i \mid \mathsf{open}\, n_i \,]$$

for some $i \in \{0, \dots, k-1\}$. This has the capability to open $n_i[\,\omega_i[\,]\,]$ and push $\omega_i[\,]$ to the top level, announcing i as the winner.

We shall now prove that every computation produces a unique winner. In particular we shall show that every computation must arrive at the form $(\star)$, and that only once it has arrived at $(\star)$ can it announce a winner, which must be unique.

We formulate an invariant: at any stage in the computation, the network is of the form

$$\prod_{i \in I} n_i[\, R_i \,]$$

where $I \subseteq \{0, \dots, k-1\}$ and R_i is in one of the following two forms:

(1) $R_i = Q_i \mid \prod_{j \in I_i} Q_j \mid \mathsf{pull}\, n_{r_i} \mid \mathsf{open}\, n_{r_i}$
with r_i defined, $r_i \in I$, I_i defined, $I_i = (i \mathinner{..} r_i)$.

(2) $R_i = Q_i \mid \prod_{j \in I_i} Q_j \mid \mathsf{open}\, n_{s_i} \mid n_{s_i}[\, R_{s_i} \,]$
with s_i defined, $s_i \in \{0, \dots, k-1\} - I$, I_i defined, $I_i = (i \mathinner{..} s_i)$, and R_{s_i} in form (1) or (2).

(Recall the notation $(i \mathinner{..} j)$ from Definition 4.1.) It is understood that r_i, s_i, I_i are undefined except where explicitly defined. In particular, for any i at most one of r_i, s_i is defined.

Furthermore, the sets I, I_i (for i where I_i is defined), $\{s_i\}$ (for i where s_i is defined) together form a partition of $\{0, \dots, k-1\}$. The intuition is that the set I consists of the indices of processes which have not (yet) lost; the s_i are indices of processes which have lost (by being pulled in), but have not yet been opened; and the sets I_i consist of those processes which have lost and been opened.

This completes the formulation of the invariant. Notice that we can deduce that whenever s_i is defined then $i \neq s_i$. For if $i = s_i$ then i would be the single process remaining, and so $i \in I$. But we know that $s_i \notin I$ from the invariant.

We see that form ($\star$) satisfies the invariant with $I = \{i\}$, $r_i = i$. We set up the invariant initially by defining $I = \{0, \ldots, k-1\}$, and, for all $i \in \{0, \ldots, k-1\}$: $r_i = i + 1$, s_i undefined, $R_i = Q_i \mid \mathsf{pull}\, n_{r_i} \mid \mathsf{open}\, n_{r_i}$. Thus $I_i = \emptyset$ and R_i is in form (1) (all i).

At an arbitrary point in the computation, there are two types of reduction possible:

(1) A reduction between two top-level processes. For $i \in I$, r_i defined, $i \neq r_i$, we have process i pulling in process r_i:

$$\begin{aligned}&n_i[\, Q_i \mid \textstyle\prod_{j \in I_i} Q_j \mid \mathsf{pull}\, n_{r_i} \mid \mathsf{open}\, n_{r_i} \,] \mid n_{r_i}[\, R_{r_i} \,]\\ &\to n_i[\, Q_i \mid \textstyle\prod_{j \in I_i} Q_j \mid \mathsf{open}\, n_{r_i} \mid n_{r_i}[\, R_{r_i} \,]\,]\end{aligned}$$

We reestablish the invariant by removing r_i from I, letting $s_i := r_i$, and making r_i undefined. Thus process i changes from form (1) to form (2). It is straightforward to see that the invariant is reestablished.

(2) A reduction within a single R_i. For i where s_i is defined we have process i opening process s_i:

$$\begin{aligned}&n_i[\, Q_i \mid \textstyle\prod_{j \in I_i} Q_j \mid \mathsf{open}\, n_{s_i} \mid n_{s_i}[\, R_{s_i} \,]\,]\\ &\to n_i[\, Q_i \mid \textstyle\prod_{j \in I_i} Q_j \mid R_{s_i} \,]\end{aligned}$$

We reestablish the invariant as follows: Firstly, I_i is augmented with $\{s_i\}$ and I_{s_i}, and then I_{s_i} is made undefined. Secondly, if R_{s_i} is of form (1), we let $r_i := r_{s_i}$, and make s_i, r_{s_i} undefined; if R_{s_i} is of form (2), we let $s_i := s_{s_i}$, and make s_{s_i} undefined. We omit the check that the invariant is reestablished.

Suppose that no reduction of either type is enabled. Then there must be a single top-level process i which is in form (1):

$$n_i[\, Q_i \mid \prod_{j \in I_i} Q_j \mid \mathsf{pull}\, n_{r_i} \mid \mathsf{open}\, n_{r_i} \,]$$

But then by the partition property of the invariant we have $\{i\} \cup I_i = \{0, \ldots, k-1\}$, implying that $r_i = i$, and so we have reached ($\star$) and the winner i can be announced. Hence every computation is guaranteed to end by announcing a winner.

Conversely, suppose that a computation announces a winner. This can only be precipitated by some top-level process i having $r_i = i$ or $s_i = i$ for some i, allowing $\mathsf{open}\, i$ to occur and thereby allowing process i to release its ω_i ambient. As noted earlier, we can never have $s_i = i$. So $r_i = i$. Then by the invariant, i must be the only process left, and we must have reached ($\star$). This means that the winner can only be announced once all reductions of the two types described above have been completed, and we have reached $(\star)$, from where there is clearly a unique winner i. $\square$

We do not see how to express this algorithm using the different movement capabilities available in MA, or in SA, or in ROAM [7].

4.2 Pure Public MA

We now solve the leader election problem for rings in pure public MA.

Theorem 4.3 *For any $k \geq 1$ there is a symmetric ring of size k which is an electoral system in pure public MA.*

In the shorter version of this paper [18] we established Theorem 4.3 for the special case $k = 4$, which is the smallest interesting case, and is sufficient for establishing separation results between calculi (Section 6).

Our strategy for proving Theorem 4.3 is to follow Palamidessi's general scheme and define an algorithm which works in two phases: distribution followed by election. In the distribution phase we distribute names around the ring so as to convert it into a complete graph of processes which can all interact with each other. Then in the election phase we elect the winner. In both phases the methods used will be quite different from those of Palamidessi in the π-calculus setting.

It was shown in [17] that pure public *boxed* MA can solve leader election on fully-connected symmetric networks (so that in fact the open capability is not needed). However it turns out that the electoral system presented there is not suitable for our present purposes. The general form is $n_0[\,Q_0\,] \mid \cdots \mid n_{k-1}[\,Q_{k-1}\,]$, and within each Q_i we have sequences of capabilities which mention all the n_j $(j < k)$. If we are to start from a ring, it would seem that these capabilities must start off distributed round the ring, from where they are then formed into sequences during the distribution phase. But it does not seem possible to do this within pure public MA. This is because in order to build up a sequence of capabilities involving names acquired from round the ring we would have to be able to either (1) modify a process P into a process $M.P$ (where M is an in or out capability), or (2) take a process $M.P$ and modify P to produce $M.P'$. Neither of these options is possible with the reduction rules (In), (Out) and (Open). (Option (2) is possible with communication, but we are here considering *pure* public MA.) We therefore use a different electoral system for the election phase, which is "flat", in the sense that we do not have sequences of capabilities which seem to have to come from different processes in the original ring.

We start by defining and proving correct the new electoral system for the election phase (Lemma 4.4). We then prove Theorem 4.3 by defining a symmetric ring which evolves to this electoral system.

We write $n^{(r)}[\,P\,]$ as a shorthand for $n[\,n[\,\ldots n[\,P\,]\ldots]\,]$ (r embedded ambients named n, with P as the contents of the innermost ambient).

Lemma 4.4 *Let $k \geq 1$, and let*

$$R_i \overset{\text{df}}{=} n_i[\,\omega_i^{(k)}[\,\mathsf{out}\,n_i\,] \mid \prod_{j \neq i} R_{ij}\,]$$

where

$$R_{ij} \overset{\text{df}}{=} \mathsf{in}\,n_j \mid \mathsf{open}\,n_j.\mathsf{dm}_j[\,] \mid \mathsf{open}\,\mathsf{dm}_j.(\mathsf{dm}_j[\,] \mid \mathsf{open}\,a_j^i) \mid a_j^i[\mathsf{open}\,\omega_i\,]\ .$$

Then $R_0 \mid \cdots \mid R_{k-1}$ is a symmetric electoral system.

Note that the slightly simpler

$$R_{ij} \overset{\text{df}}{=} \mathsf{in}\,n_j \mid \mathsf{open}\,n_j.\mathsf{dm}_j[\,] \mid \mathsf{open}\,\mathsf{dm}_j.(\mathsf{dm}_j[\,] \mid \mathsf{open}\,\omega_i)$$

would also be an electoral system. However it is not suitable, as $\mathsf{opendm}_j.(\mathsf{dm}_j[\,] \mid \mathsf{open}\,\omega_i)$ is not flat (see discussion above).

PROOF of Lemma 4.4. The idea of the electoral system is that process j loses to process i when ambient n_j enters ambient n_i. The last ambient left at the top level is the winner. But of course it needs some mechanism to detect that it has won. So once an ambient n_j has lost it can be opened by its parent ambient i and replaced by a "dummy" ambient dm_j. By opening dm_j, process i can then open its a_j^i ambient and thereby open one of the nested ω_i ambients. Once process i has opened dm_j for all $j \neq i$, the nesting of ω_i ambients can be reduced to one, enabling an ω_i ambient to leave n_i and appear at the top level to announce i as the winner. We need the dummy ambients, since if process i opens n_j, process i may not be the eventual winner. Thus a third process h may also need to discover that j has already lost. It does this by opening dm_j.

Recall that $(i \mathinner{..} i) = \{0,\ldots,k-1\} - \{i\}$ (Definition 4.1). We start by defining an invariant which describes the form of the network at any point in any computation, up until the state immediately before a winner is declared.

Invariant: the network is of the form

$$\prod_{i \in TN} n_i[\,U_i\,]$$

where for each $i \in TN \cup \bigcup_{j<k} LN_j$:

$$U_i = V_i \mid \prod_{j \in LN_i} n_j[\,U_j\,] \mid \prod_{j \in LD_i} (\mathsf{dm}_j[\,] \mid V_j)$$

and for each $i < k$:

$$\begin{aligned} V_i = {} & \omega_i^{(w_i)}[\,\mathsf{out}\; n_i\,] \mid \textstyle\prod_{j \in IN_i}(\mathsf{in}\; n_j) \mid \prod_{j \in ON_i} \mathsf{open}\; n_j.\mathsf{dm}_j[\,] \\ & \mid \textstyle\prod_{j \in OD_i} \mathsf{open}\; \mathsf{dm}_j.(\mathsf{dm}_j[\,] \mid \mathsf{open}\; a_j^i) \\ & \mid \textstyle\prod_{j \in UA_i - OD_i}(\mathsf{open}\; a_j^i) \mid \prod_{j \in UA_i}(a_j^i[\,\mathsf{open}\; \omega_i\,]) \mid (\mathsf{open}\; \omega_j)^{w_i - 1 - |UA_i|} \end{aligned}$$

Here TN is the set of top-level n_i ambients, LN_i is the set of lower-level n_j ambients with parent n_i, and LD_i is the set of (lower-level) dm_j ambients with parent n_i. We adopt the convention that the sets LN_i, LD_i are defined for all i, being set to $\emptyset$ if n_i has already been opened.

We use IN_i for the set of all j such that process i can still perform $\mathsf{in}\, n_j$; ON_i for the set of all j such that process i can still perform $\mathsf{open}\, n_j$; OD_i for the set of all j such that process i can still perform $\mathsf{open}\, \mathsf{dm}_j$; and UA_i for the set of all j such that process i still has an unopened a_j^i ambient.

We require these conditions as part of the invariant:

(1) Basic well-formedness:

$$IN_i \subseteq (i \mathrel{..} i) \quad ON_i \subseteq (i \mathrel{..} i) \quad UA_i \subseteq (i \mathrel{..} i) \quad OD_i \subseteq UA_i \quad w_i \geq 1 + |UA_i|$$

(2) The sets TN and LN_i, LD_i (all $i < k$) partition $\{0, \ldots, k-1\}$. This expresses the fact that for every $j < k$ the ambient n_j (or its dummy replacement dm_j) occurs exactly once in the network, either at the top level (not yet having lost), or at a lower level inside another n_j.

(3) There must always be at least one process left at the top level: $TN \neq \emptyset$.

(4) For any $i, j < k$ such that $i \neq j$, if $i, j \in TN$ then $j \in IN_i$ (and $i \in IN_j$). This says that any top-level n_i can enter any other top-level n_j.

(5) For any $i < k$:

$$TN \cup \bigcup_{j<k} LN_j \subseteq ON_i \cup \{i\}$$

This says that any process i always has the capability $\mathsf{open}\; n_j$ for any unopened n_j ($j \neq i$).

(6) For any $i < k$:

$$(i \mathrel{..} i) - OD_i \subseteq \bigcup_{j<k} LD_j$$

This says that if process i has performed $\mathsf{open}\ \mathsf{dm}_j$ then n_j must have already been opened and replaced by dm_j.

This completes the description of the invariant.

Initially $TN = \{0, \ldots, k-1\}$, and for all $i < k$: $w_i = k$, $LN_i = LD_i = \emptyset$ and $IN_i = ON_i = OD_i = UA_i = (i \mathrel{..} i)$. It is straightforward to check that this establishes the invariant.

Immediately before i is announced as the winner, we shall see that the network will be of the following form:

$$(\star) \quad \begin{array}{c} n_i[\, V_i \mid \prod_{j \neq i}(\mathsf{dm}_j[\,] \mid V_j)\,] \\ \text{where } V_i = \omega_i[\,\mathsf{out}\, n_i\,] \mid \prod_{j \in IN_i}(\mathsf{in}\, n_j) \mid \prod_{j \in ON_i} \mathsf{open}\, n_j.\mathsf{dm}_j[\,] \end{array}$$

with V_j satisfying the conditions of the invariant for $j \neq i$. Thus $TN = \{i\}$, $LN_i = OD_i = UA_i = \emptyset$, $LD_i = (i \mathrel{..} i)$, $w_i = 1$, and $LN_j = LD_j = \emptyset$ for all $j \neq i$.

To complete the proof we show four things:

- The invariant is maintained when performing any reduction, apart from an $\mathsf{out}\, n_i$ reduction.
- A computation can always make progress if it has not reached form $(\star)$. Since clearly all computations are finite, this will show that every computation reaches $(\star)$, from where a winner can be announced in one step. Hence every computation announces a winner.
- A computation can only produce a winner by reaching form $(\star)$. In particular, if an $\mathsf{out} n_i$ reduction occurs, then the network is of form $(\star)$ immediately before the reduction occurs.
- Once a winner is announced, no further computation can produce a second winner. Thus no computation can announce more than one winner.

We start by checking that the invariant is maintained by all reductions apart from $\mathsf{out}\, n_i$. There are five possible types of reduction:

($\mathsf{in}\, n_j$) Suppose that ambient n_i enters ambient n_j. We must have $j \in IN_i$ or $j \in IN_h$ for some $h \in LD_i$. In the first case the reduction entails $IN_i := IN_i - \{j\}$, while in the second case $IN_h := IN_h - \{j\}$. Furthermore, either $i, j \in TN$ or $i, j \in LN_{h'}$ for some h'. In the first case we have $TN := TN - \{i\}$ and in the second case $LN_{h'} := LN_{h'} - \{i\}$. In all cases we set $LN_j := LN_j \cup \{i\}$.

($\mathsf{open}\, n_j$) Suppose that n_j is opened inside n_i. Then $j \in LN_i$. The reduction causes the following changes:

$$LN_i := (LN_i - \{j\}) \cup LN_j \qquad LN_j := \emptyset$$
$$LD_i := LD_i \cup \{j\} \cup LD_j \qquad LD_j := \emptyset$$

($\mathsf{open}\ \mathsf{dm}_j$) Suppose that dm_j is opened inside n_i. Then $j \in LD_i$. Either $j \in OD_i$ or $j \in OD_h$ for some $h \in LD_i$ (both could be true). After the reduction LD_i is unchanged and one of either $OD_i := OD_i - \{j\}$ or $OD_h := OD_h - \{j\}$.

($\mathsf{open}\ a^i_j$) Suppose that a^i_j is opened. Then $j \in UA_i - OD_i$. After the reduction we set $UA_i := UA_i - \{j\}$.

($\mathsf{open}\ \omega_i$) Suppose that ω_i is opened. Then $w_i > 1 + |UA_i|$. After the reduction we set $w_i := w_i - 1$.

We omit the straightforward checks that for all five types of reduction the invariant is maintained.

Next we show that a computation can always make progress towards form ($\star$).

- First suppose that $|TN| \geq 2$. Then one top-level ambient can enter another by condition (4). Hence we can assume that $|TN| \leq 1$, and so by condition (3) we have $TN = \{i\}$ for some $i < k$.
- Next suppose that $LN_i \neq \emptyset$. Take $j \in LN_i$. Then $j \neq i$ by condition (2). So $j \in ON_i$ by condition (5). Hence an $\mathsf{open}\ n_j$ reduction can take place. We can therefore assume that $LN_i = \emptyset$. Hence $LD_i = (i \mathrel{..} i)$.
- Next suppose that $OD_i \neq \emptyset$. Take $j \in OD_i$. Then $j \neq i$ by condition (1). So $j \in LD_i$ and we can perform an $\mathsf{open}\ \mathsf{dm}_j$ reduction. Hence we may assume that $OD_i = \emptyset$.
- If $UA_i \neq \emptyset$ then we can perform an $\mathsf{open}\ a^i_j$ reduction. So we can assume that $UA_i = \emptyset$.
- Finally, if $w_i > 1$ then we can perform an $\mathsf{open}\ \omega_i$ reduction. So we can assume that $w_i = 1$.

Putting all this together shows that we have reached form ($\star$) as required.

Next we show that for a winner to be announced, form ($\star$) must have been reached. For i to win, an ω_i ambient must leave n_i and arrive at the top level. This implies that n_i still exists and is at the top level, so that $i \in TN$. Also $w_i = 1$, and so $UA_i = \emptyset$ by condition (1). Hence $OD_i = \emptyset$, also by condition (1). But then $\bigcup_{j<k} LD_j = (i \mathrel{..} i)$ by condition (6). This implies that $\bigcup_{j<k} LN_j = \emptyset$ and $TN = \{i\}$ by condition (2). So all n_j, $j \neq i$, have been opened, and we must have $LD_i = (i \mathrel{..} i)$. Hence immediately before the ω_i ambient leaves n_i the network must be of form ($\star$).

Finally we show that once i has won, for $j \neq i$, process j cannot subsequently be announced as a winner. Since we know that stage ($\star$) has been reached, we know that ambient n_j has already been opened. So an $\mathsf{out}\, n_j$ reduction cannot take place, and so ambient ω_j cannot emerge at the top level, and there is no way in which j can win. $\Box$

We now give the proof of Theorem 4.3, which we restate for convenience.

Theorem 4.3. *For any $k \geq 1$ there is a symmetric ring of size k which is an electoral system in pure public MA.*

PROOF. For $k = 1, 2, 3$, any network of size k is trivially a ring. So we can use the electoral system of Lemma 4.4 directly. Therefore we can assume that $k \geq 4$.

Our aim is to define a symmetric ring of processes $P_0 \mid \cdots \mid P_{k-1}$ which is guaranteed to reduce to the electoral system $R_0 \mid \cdots \mid R_{k-1}$ of Lemma 4.4. Clearly $P_0 \mid \cdots \mid P_{k-1}$ will then satisfy the conditions of the theorem. The reduction from $P_0 \mid \cdots \mid P_{k-1}$ to $R_0 \mid \cdots \mid R_{k-1}$ is what we earlier called the distribution phase.

The idea is that for each $j < k$ we have $P_j = \prod_{i<k} P_j^i$. The subprocess P_j^i is designed to convey a process Q_j^i round the ring from process j to process i. Once all the Q_j^i $(j \neq i)$ arrive at process i to join Q_i^i, interaction between the Q_j^i produces R_i as in Lemma 4.4 (roughly speaking).

We shall adopt the convention that names which are proper to the original process P_i are subscripted with i. We also allow P_i to use names subscripted with $i+1$, allowing P_i to interact with P_{i+1} and P_{i-1}. This helps to clarify matters, and is a convenient way to ensure that $P_0 \mid \cdots \mid P_{k-1}$ forms a ring. As far as possible we adopt a similar convention with processes, such as P_j^i (which is part of P_j and not P_i).

Recall the notation $(i \mathinner{..} j)$, $[i \mathinner{..} j]$ from Definition 4.1. We start by defining processes Q_h^{ij} for $j \neq i$, $h \in [j \mathinner{..} i]$, such that R_{ij} (as in the statement of Lemma 4.4) is got from $\prod_{h \in [j \,..\, i]} Q_h^{ij}$. For $j \neq i$:

$$\begin{aligned}
Q_j^{ij} &\overset{\text{df}}{=} \mathsf{in}\, n_j \mid \mathsf{open}\, n_j.\mathsf{dm}_j[\,] \mid \mathsf{open}\, \mathsf{dm}_j.(\mathsf{dm}_j[\,] \mid \mathsf{open}\, a_j^i) \\
&\qquad \mid a_j^i[\,\mathsf{open}\, b_{j+1}^{ij}.\mathsf{ack}_{j+1}^{ij}[\,\mathsf{out}\, a_j^i\,]\,] \mid b_{j+1}^{ij}[\,\mathsf{open}\, a_{j+1}^{ij}.\mathsf{in}\, a_j^i\,] \\
Q_h^{ij} &\overset{\text{df}}{=} a_h^{ij}[\,\mathsf{open}\, b_{h+1}^{ij}.\mathsf{in}\, b_h^{ij}\,] \mid b_{h+1}^{ij}[\,\mathsf{open}\, a_{h+1}^{ij}.\mathsf{in}\, a_h^{ij}\,] \mid \mathsf{open}\, \mathsf{ack}_h^{ij}.\mathsf{ack}_{h+1}^{ij}[\,] \\
&\qquad\qquad \text{for } h \in (j \mathinner{..} i) \\
Q_i^{ij} &\overset{\text{df}}{=} a_i^{ij}[\,\mathsf{in}\, b_i^{ij} \mid \mathsf{open}\, \omega_i\,]
\end{aligned}$$

We claim that for $j \neq i$:

$$\prod_{h \in [j\ ..\ i]} Q_h^{ij} \twoheadrightarrow R_{ij} \mid \mathsf{ack}_i^{ij}[\,]$$

The sequence of reductions is deterministic. Note that Q_i^{ij} must be present before any reduction can start. The ack_i^{ij} acknowledgement ambient is used to signal that the reductions have been completed. The idea is that the $\mathsf{open}\ \omega_i$ capability (which must originate in P_i) is progressively moved (using subscripts in descending order) from the a_i^{ij} ambient to the a_j^i ambient where it is required to reside for R_{ij}. Once this is achieved the acknowledgement ack_{j+1}^{ij} is progressively converted into ack_i^{ij} (using subscripts in ascending order), in which form it is usable by process i.

For $h \neq i$ let

$$Q_h^i \stackrel{\text{df}}{=} \prod_{j \in [i+1\ ..\ h]} Q_h^{ij} \ .$$

Also let

$$Q_i^i \stackrel{\text{df}}{=} \prod_{j \neq i} Q_i^{ij} \ .$$

As stated above, the idea is that Q_h^i is originally part of P_h, and for $h \neq i$ we pass Q_h^i around the ring to process i.

Now

$$\prod_{h \leq k} Q_h^i = (\prod_{h \neq i} \prod_{j \in [i+1\ ..\ h]} Q_h^{ij}) \mid \prod_{j \neq i} Q_i^{ij} = \prod_{j \neq i} \prod_{h \in [j\ ..\ i]} Q_h^{ij} \ .$$

Hence

$$\prod_{h \leq k} Q_h^i \twoheadrightarrow \prod_{j \neq i} (R_{ij} \mid \mathsf{ack}_i^{ij}[\,]) \ .$$

There are $k-1$ (one for each $j \neq i$) separate deterministic sequences of reductions which are interleaved.

We now define the ring of processes P_i:

$$P_i \stackrel{\text{df}}{=} \prod_{j<k} P_i^j$$

where P^i_h is defined for each $h < k$ as follows:

$$
\begin{aligned}
P^i_{i+1} &\stackrel{\text{df}}{=} d^i_{i+2}[\,\mathsf{in}\ c^i_{i+2} \mid e^i_{i+2}[\,Q^i_{i+1}\,]\,]\\
P^i_h &\stackrel{\text{df}}{=} c^i_h[\,\mathsf{open}\ d^i_h.\mathsf{in}\ d^i_{h+1}.\mathsf{in}\ e^i_{h+1}\,]\\
&\quad \mid d^i_{h+1}[\,\mathsf{open}\ f^i_h.\mathsf{in}\ c^i_{h+1} \mid e^i_{h+1}[\,Q^i_h \mid \mathsf{open}\ c^i_h.\mathsf{open}\ e^i_h.f^i_h[\,\mathsf{out}\ e^i_{h+1}\,]\,]\,]\\
&\quad \text{for } h \in (i+1 \mathrel{..} i-1)\\
P^i_{i-1} &\stackrel{\text{df}}{=} c^i_{i-1}[\,\mathsf{open}\ d^i_{i-1}.\mathsf{in}\ d^i_i.\mathsf{in}\ e^i_i\,]\\
&\quad \mid d^i_i[\,\mathsf{open}\ f^i_{i-1}.\mathsf{in}\ r_i.\mathsf{in}\ n_i \mid e^i_i[\,Q^i_{i-1} \mid \mathsf{open}\ c^i_{i-1}.\mathsf{open}\ e^i_h.f^i_{i-1}[\,\mathsf{out}\ e^i_i\,]\,]\,]\\
P^i_i &\stackrel{\text{df}}{=} r_i[\,n_i[\,\omega^{(k)}_i[\,\mathsf{out}\ n_i\,] \mid Q^i_i\\
&\quad \mid \mathsf{open}\ d^i_i.\mathsf{open}\ e^i_i.\mathsf{open}\ \mathsf{ack}^{i,i+1}_i.\ \ldots\ .\mathsf{open}\ \mathsf{ack}^{i,i-1}_i.s_i[\,\mathsf{out}\ n_i.\mathsf{out}\ r_i\,]\,]\,]\\
&\quad \mid \mathsf{open}\ s_i.\mathsf{open}\ r_i
\end{aligned}
$$

(Recall that we are assuming $k \geq 4$, so that there is no clash between the definitions of P^i_{i+1} and P^i_{i-1}.) The idea is that Q^i_{i+1} is conveyed round the ring from $i+1$ to i, accumulating with it the processes $Q^i_{i+2}, \ldots, Q^i_{i-1}$ on its way.

Each P_i only uses names with subscripts i or $i+1$. Hence $P_0 \mid \cdots \mid P_{k-1}$ forms a symmetric ring of size k.

Let $S_i \stackrel{\text{df}}{=} \prod_{h<k} P^i_h$. Then

$$\prod_{i<k} S_i = \prod_{i<k} P_i$$

so that the S_i just represent a rearrangement of the P_i. We claim that S_i evolves into R_i, which can then carry out the election. First

$$
\begin{aligned}
S_i &\twoheadrightarrow d^i_i[\,\mathsf{in}\ r_i.\mathsf{in}\ n_i \mid e^i_i[\,\textstyle\prod_{h\neq i} Q^i_h\,]\,] \mid P^i_i\\
&\twoheadrightarrow r_i[\,n_i[\,\omega^{(k)}_i[\,\mathsf{out}\ n_i\,] \mid \textstyle\prod_{h\leq k} Q^i_h\\
&\qquad \mid \mathsf{open}\ \mathsf{ack}^{i,i+1}_i.\ \ldots\ .\mathsf{open}\ \mathsf{ack}^{i,i-1}_i.s_i[\,\mathsf{out}\ n_i.\mathsf{out}\ r_i\,]\,]\,] \qquad (*)\\
&\qquad \mid \mathsf{open}\ s_i.\mathsf{open}\ r_i\\
&\twoheadrightarrow r_i[\,n_i[\,\omega^{(k)}_i[\,\mathsf{out}\ n_i\,] \mid \textstyle\prod_{j\neq i}(R_{ij} \mid \mathsf{ack}^{ij}_i[\,])\\
&\qquad \mid \mathsf{open}\ \mathsf{ack}^{i,i+1}_i.\ \ldots\ .\mathsf{open}\ \mathsf{ack}^{i,i-1}_i.s_i[\,\mathsf{out}\ n_i.\mathsf{out}\ r_i\,]\,]\,] \qquad (\dagger)\\
&\qquad \mid \mathsf{open}\ s_i.\mathsf{open}\ r_i\\
&\twoheadrightarrow r_i[\,n_i[\,\omega^{(k)}_i[\,\mathsf{out}\ n_i\,] \mid \textstyle\prod_{j\neq i}(R_{ij}) \mid s_i[\,\mathsf{out}\ n_i.\mathsf{out}\ r_i\,]\,]\,] \mid \mathsf{open}\ s_i.\mathsf{open}\ r_i \qquad (\ddagger)\\
&\twoheadrightarrow n_i[\,\omega^{(k)}_i[\,\mathsf{out}\ n_i\,] \mid \textstyle\prod_{j\neq i} R_{ij}\,] = R_i
\end{aligned}
$$

The sequence of reductions is deterministic up to stage $(*)$. From $(*)$ until $(\ddagger)$ there are various possible interleavings as each $R_{ij} \mid \mathsf{ack}^{ij}_i[\,]$ is formed separately for $j \neq i$. Thus (†) just illustrates one possible intermediate stage. We have that S_i is guaranteed to evolve into R_i.

We must establish that each reduction sequence $S_i \twoheadrightarrow R_i$ is independent of the others. Let S'_i be any state reached from S_i up to immediately before R_i is reached. Looking at the definition of S_i, we see that all names are indexed with i, apart from n_j and dm_j (used in the subsequent election phase) occurring in Q_j^{ij}, which is part of Q_h^i for various h. As long as these names are kept below the top level, and in the case of $\mathsf{in}\ n_j$ at least two levels down, then there can be no interaction between any Q'_i and Q'_j. Now we made the definition of the P_h^i such that at each stage of the transmission of Q_j^{ij} round the ring from j to i it is enclosed in at least *two* ambients. This continues to be the case once Q_j^{ij} enters process i, until the very final reduction to reach R_i, when the shell ambient r_i is stripped off to expose n_i. So within S'_i, the names n_j and dm_j (for $j \neq i$) are always at least two levels down, giving us the required independence.

It is not strictly true that $\prod_{i<k} P_i = \prod_{i<k} S_i$ is guaranteed to evolve into $\prod_{i<k} R_i$. This is because the various S_i will evolve into R_i at different rates for different $i < k$. Thus the processes enter the election phase at different times. This does not cause a problem, as S_i is guaranteed to reach R_i and the intermediate S'_i states cannot take part in the election phase. The latter follows from the fact that within the election phase the top-level ambients are all n_j for various $j < k$. These n_j ambients cannot interact with any S'_i, since S'_i keeps its $\mathsf{in}\ n_j$ capabilities at least two levels down, and its n_i ambient one level down, until R_i is reached. $\square$

Remark 4.5 *There is a standard encoding of MA into SA, as follows:*

$$[\![n[\,P\,]]\!] \overset{\mathrm{df}}{=} n[\ !\,\overline{\mathsf{in}}\,n \mid !\,\overline{\mathsf{out}}\,n \mid \overline{\mathsf{open}}\,n \mid [\![P]\!]\,]$$

(with $[\![-]\!]$ homomorphic on the remaining operators) [9].

Corollary 4.6 *For any $k \geq 1$ there is a symmetric ring of size k which is an electoral system in pure public SA.*

PROOF. With P_i $(i < k)$ as in the proof of Theorem 4.3, we have that $[\![P_0]\!] \mid \cdots \mid [\![P_{k-1}]\!]$ (using the encoding of Remark 4.5) also forms a symmetric ring which is an electoral system in pure public SA.

This could be derived from the general theory of Section 6, which defines conditions on encodings under which rings which are electoral systems in the source language are mapped into rings which are electoral systems in the target language. $\square$

4.3 Boxed Ambients

We now consider Boxed Ambients. The solutions for MA and PAC depend on open to pass capabilities around. In the case of BA where the open capability is missing by design choice, one might wonder whether leader election is possible in rings. However, perhaps surprisingly, it turns out that the parent-child communication of BA enables the construction of symmetric rings forming electoral systems.

Theorem 4.7 *For any $k \geq 1$, there is a symmetric ring of size k which is an electoral system in public BA.*

PROOF. We define a symmetric ring $P_0 \mid \cdots \mid P_{k-1}$ which is an electoral system. As in the proof of Theorem 4.3, we follow Palamidessi's method of first distributing names round the ring to create a complete graph and then running the election on it. Palamidessi shows how to distribute the names in the (choice-free) asynchronous π-calculus [15, Proposition 5.1]. Her argument is only stated for $k = 4$, but it generalises to arbitrary k. We shall give a variant of her construction, using the public choice-free synchronous π-calculus. Suppose that process P_i has a channel n_i initially known only to itself, and can send messages to P_{i-1} along channel x_i. Then the names n_i are passed around the ring so that all processes share them and can use them in the election phase. As Palamidessi points out, we have to be careful that for each P_i the outputs occur in the same order as the inputs, so that names do not get confused. We therefore allocate to each P_i a "synchroniser" name y_i which ensures that each successive output is completed before the next one is enabled. We elide the dummy names passed along y_i.

For $0 \leq i \leq k$, we let $P_i \stackrel{\text{df}}{=} P_i^0\langle x_i, x_{i+1}, y_i, n_i\rangle$, where for $0 \leq j \leq k-2$ we let

$$\begin{aligned} &P_i^j(x_i, x_{i+1}, y_i, n_i, \ldots, n_{i+j}) \\ &\qquad \stackrel{\text{df}}{=} \bar{x}_i\langle n_{i+j}\rangle.\bar{y}_i \mid x_{i+1}(n_{i+j+1}).y_i.P_i^{j+1}\langle x_i, x_{i+1}, y_i, n_i, \ldots, n_{i+j+1}\rangle \end{aligned}$$

and $P_i^{k-1}(x_i, x_{i+1}, y_i, n_i, \ldots, n_{i-1}) \stackrel{\text{df}}{=} Q_i\langle n_i, \ldots, n_{i-1}\rangle$. Here Q_i is a process which has acquired all the n_i and is ready to carry out the election phase. Once Q_i is reached, the names x_i, x_{i+1} and y_i are no longer required.

Since BA can encode choice-free synchronous π-calculus [5], we can carry out the distribution phase in BA. We use the following translation of the π-calculus input and synchronous output:

$$[\![x(y).P]\!] \overset{\text{df}}{=} (y,z)^x.(z[\,]\mid[\![P]\!])$$
$$[\![\bar{x}\langle y\rangle.P]\!] \overset{\text{df}}{=} x[\,\langle y,z\rangle\,]\mid()^z.[\![P]\!]$$

where z is fresh. This translation is adapted from [5], which used restriction. Note that we do not need restriction, since in our particular setting there is no harm in introducing fresh public names. Note also that only the (Input n) rule of BA is needed to simulate π-calculus communication, and not the remaining four communication rules of BA.

The algorithm for the election phase (i.e. the definition of the Q_i for $i < k$) is the same as the one presented in [17,20] for pure public boxed MA, which is of course pure public BA. We recall the definition: for $i < k$, let $S_i^k = \{n_j : j \in (i \mathrel{..} i)\}$, and let T_i^k be the set of all strings of length $k-1$ using the members of S_i^k exactly once each. Given an element s of T_i^k we denote by s^- the string which is s in reverse order. By $\mathsf{in}\,(s)$ we mean the sequence of $\mathsf{in}\,n_j$ capabilities for each successive $n_j \in s$ (similarly for out). We set:

$$Q_i \overset{\text{df}}{=} n_i[\prod_{j\in(i\ ..\ i)} \mathsf{in}\,n_j \mid \prod_{s\in T_i^k} \omega_i[\,\mathsf{in}\,(s).\mathsf{out}\,(s^-).\mathsf{out}\,n_i\,]\,]$$

□

5 Calculi without Electoral Systems for Rings

In this section, we show that the open capability is crucial for electing a leader in symmetric rings. If fact, if the open capability is dropped, then election in symmetric rings is not possible. We present below the proof for MA (Theorem 5.3). The same techniques also establish a similar result for PAC and SA (Theorem 5.21).

We start this section by restating Palamidessi's result on the non-existence of electoral systems for CCS.

Theorem 5.1 *If* Net *is a CCS network which is symmetric and independent with respect to a non-trivial well-balanced automorphism* σ*, then* Net *is not an electoral system.*

PROOF. This is essentially Theorem 6.1 of [15], recast in the present setting of unlabelled rather than labelled transitions. □

Remark 5.2 *Theorem 6.1 of [15] is stated for π-calculus with internal mobility (π_I) [19], as well as CCS. This part of the result also carries over to the present reduction semantics setting.*

Recall that by *boxed* MA we mean MA without the open capability.

Theorem 5.3 *For any composite $k > 1$, boxed MA does not have a symmetric ring of size k which is an electoral system.*

Since the proof is quite elaborate, we start by giving an overview of the method.

Suppose that we have a symmetric ring of composite size k. Then by Lemma 3.12 there is a non-trivial, well-balanced automorphism σ with independent orbits of size $s \geq 2$. The overall idea is that whatever reduction Net makes, we can retain symmetry and independence with respect to σ by propagating that move round the orbit(s) concerned to complete a round of s reductions. If a process ever declares itself a winner, then by symmetry all processes in the same orbit can declare themselves winners on the same round. With orbits of size $s \geq 2$ this means that there is a computation of Net which does not declare a unique winner, so that Net is not an electoral system. The significance of Net being independent with respect to σ is that if a reduction involves two processes interacting then they must come from different orbits. This means that when the reduction is propagated round the two orbits concerned we have restored symmetry with respect to σ.

So far, the method we have outlined essentially follows Palamidessi's proof for CCS. However there are two key differences in the case of boxed MA, both of which threaten the independence of the ring with respect to σ.

(1) As a result of ambients entering other ambients, processes can acquire new names, and processes which were independent may no longer be independent. For example, let

$$P_1 \overset{\text{df}}{=} n_1[\,\mathsf{in}\,n_2\,] \quad P_2 \overset{\text{df}}{=} n_2[\,\mathsf{in}\,n_3\,] \quad P_3 \overset{\text{df}}{=} n_3[\,\mathsf{in}\,n_4\,]$$

Then P_1 and P_3 are independent. However, after ambient n_2 enters n_3 we have the following:

$$P_1 = n_1[\,\mathsf{in}\,n_2\,] \quad P_2' = \mathbf{0} \quad P_3' = n_3[\,\mathsf{in}\,n_4 \mid n_2[\,]\,]$$

So P_3' has acquired the name n_2, and P_1 and P_3' are no longer independent. Notice however that although $n_2 \in \mathsf{fn}(P_3')$, P_3' does not really have access to the ambient n_2 or its possible contents. P_3' has not really acquired any new capabilities.

The situation would be different if we allowed the open capability. As a simple example, if $P \overset{\text{df}}{=} m[\,\mathsf{in}\,n.\mathsf{open}\,n'\,]$ and $Q \overset{\text{df}}{=} n[\,\mathsf{open}\,m\,]$, then when ambient m enters ambient n and is opened, we get $Q' = n[\,\mathsf{open}\ n'\,]$. Thus Q has acquired a new capability. We exploited this in the proof of Theorem 4.3.

Our solution to this problem is to maintain a weaker form of independence by labelling ambients and their contents. Initially all of process i will be labelled with i (each i). By preserving these labels during the computation we keep track of which ambients truly belong to a process, and which ambients have entered from another process. "Foreign" ambients can move around, but they can never transfer their capabilities to the host process, since the open capability is not available. We will then be able to show that processes from the same orbit cannot interact (though they may share names).

(2) Since communication is anonymous, processes can interact even if they do not share any names. As a result of the interaction they can come to share names. For instance, let $P \overset{\mathrm{df}}{=} \langle m\rangle \mid m[\,]$ and $Q \overset{\mathrm{df}}{=} (n).n[\,]$. Then $\mathsf{fn}(P) \cap \mathsf{fn}(Q) = \emptyset$. However $P \mid Q \to P' \mid Q'$, where $P' = Q' = m[\,]$, so that P' and Q' now share name m.

To get round this problem we exploit the fact that we have a ring, which has a single-orbit automorphism τ. If a communication is possible, then by symmetry all processes can participate both in input and output, and so we can choose that they communicate with themselves, which maintains symmetry and independence.

Thus item (2) is the reason that we use a stronger condition than Palamidessi. We do not know whether Theorem 5.3 still holds if we follow Palamidessi in assuming the existence of a non-trivial automorphism with independent orbits. However, consider the following example:

$$\begin{aligned} P_0 &\overset{\mathrm{df}}{=} (x_0).x_0[\,\mathsf{in}\ y_1\,] \qquad P_1 \overset{\mathrm{df}}{=} \langle y_1\rangle \\ P_2 &\overset{\mathrm{df}}{=} (x_2).x_2[\,\mathsf{in}\ y_3\,] \qquad P_3 \overset{\mathrm{df}}{=} \langle y_3\rangle \\ \mathsf{Net} &\overset{\mathrm{df}}{=} P_0 \mid P_1 \mid P_2 \mid P_3 \end{aligned}$$

Then Net forms a ring which is symmetric with respect to the automorphism which maps P_i to P_{i+2} modulo k (all i). Also the orbits are independent. However Net is clearly not a symmetric ring.

Suppose that P_0 communicates with P_3, producing $P_0' = y_3[\,\mathsf{in}\ y_1\,]$. The next reduction must be a communication between P_1 and P_2, producing $P_2' = y_1[\,\mathsf{in}\ y_3\,]$. The next move will be between P_0' and P_2' (which are no longer independent), and will break symmetry. By slightly elaborating the example we could then declare a unique winner. So if we want to preserve symmetry we cannot allow P_0 to communicate with P_3 initially. However, in the basic strategy of Palamidessi outlined above, *any* reduction involving processes in different orbits can be chosen initially.

If, on the other hand, we start by allowing P_0 and P_1 to communicate, then we get a computation which does not break symmetry. So Net is not an electoral system, and therefore not a counterexample to Theorem 5.3 holding when we assume a non-trivial automorphism with independent orbits.

Before proceeding to the proof of Theorem 5.3, we develop the theory of labelled processes alluded to in item (1).

Definition 5.4 *A labelled boxed MA process R is a boxed MA process P, with each occurrence of a name in P (whether free or bound) given a label from $\mathbb{N}$, subject to the condition that if n is bound in P then it gets the same label wherever it occurs in R.*

In the remainder of this section, we let $R, S, \ldots$ range over labelled processes, while $P, Q, \ldots$ range over standard processes. We write the labels as superscripts. Thus a possible labelling of $P \stackrel{\text{df}}{=} n[\,] \mid \nu m\,(m[\,] \mid \mathsf{in}\, m.n[\,])$ is $R \stackrel{\text{df}}{=} n^i[\,] \mid \nu m^j\,(m^j[\,] \mid \mathsf{in}\, m^j.n^k[\,])$; the two occurrences of n can have different labels, but the three occurrences of m must have the same label.

Alpha-conversion of bound names is allowed as usual, provided that there is no capture of free names and provided that the new name has the same label as the old name. Labelled processes inherit the usual notions of structural congruence and reduction from standard processes. We give the key rules for reduction:

$$\begin{array}{ll}
\text{(In)} & n^i[\mathsf{in}\, m^j.R \mid R'\,] \mid m^k[\,S\,] \to m^k[\,n^i[\,R \mid R'\,] \mid S\,] \\
\text{(Out)} & m^k[\,n^i[\mathsf{out}\, m^j.R \mid R'\,] \mid S\,] \to n^i[\,R \mid R'\,] \mid m^k[\,S\,] \\
\text{(Comm)} & \langle n^i \rangle \mid (m^j).R \to R\{n^i/m\}
\end{array}$$

In rule (In), notice that the capability $\mathsf{in}\, m^j$ can have a different label from the ambient m^k it is used to enter. This is so that labelling processes does not inhibit any reduction available to the the underlying processes. A similar remark applies to (Out). In (Comm), n^i is substituted for all occurrences of m; these must all have the label j by Definition 5.4. It can be checked that the property in Definition 5.4 that all bound names have the same label is preserved under structural congruence and reduction.

Given a labelled process R we can obtain a standard process by just omitting all the labels. Call this process $R^\dagger$. The next lemma shows that labelled and unlabelled processes have essentially the same reduction sequences.

Lemma 5.5 • *Let R, S be labelled processes. If $R \to S$ then $R^\dagger \to S^\dagger$.*
• *Let R be a labelled process and Q a process. If $R^\dagger \to Q$ then there is a labelled process S such that $R \to S$ and $S^\dagger \equiv Q$. If $R^\dagger \twoheadrightarrow Q$ then there is a labelled process S such that $R \twoheadrightarrow S$ and $S^\dagger \equiv Q$.*

PROOF. Straightforward and omitted. The point of interest is that in the unlabelled world we may be able to fold a process into a replication to get Q, while in the labelled world we may not be able to, because of different labels. As an example, let $R \stackrel{\mathrm{df}}{=} !\, m^i[\,] \mid \langle m^j \rangle \mid (n^l).n^l[\,]$. Then $R^\dagger \to Q = !\, m[\,]$. Also $R \to S = m^l[\,] \mid !\, m^i[\,]$, and $S^\dagger \equiv Q$. $\Box$

Just as no new free names can be created during a computation, no new free labelled names can be created:

Lemma 5.6 *Let R, S be labelled processes. Suppose $R \to S$ and n^i occurs free in S. Then there is a free occurrence of n^i in R.*

PROOF. Straightforward and omitted. $\Box$

We are interested in what we shall call *coherent* labelled processes. This means roughly that:

(1) all occurrences of names (other than ambient names) at the top level within an ambient have the same label as that ambient
(2) each top-level thread has a single label (a *thread* is a process which is an output, or which starts with a capability or an input)

We give some examples. Suppose that $i \neq j$. Then the process $\mathsf{in}\, m^i.\mathsf{in}\, n^j$ is incoherent. So is $n^i[\,\langle m^j \rangle \mid (a^i).R\,]$, since within an ambient labelled with i all names must be labelled with i unless they are contained in a subambient of the main ambient. On the other hand, $\langle m^j \rangle \mid (n^i).n^i[\,]$ is coherent, since we allow the parallel composition of threads with different labels at the top level.

We shall show that coherence is preserved by all reductions, apart from top-level communications where the labels are different on input and output. (Here "top-level" means not inside an ambient.)

Before defining coherence we need an auxiliary predicate. We say that an occurrence of a name in a process is *ambient-unguarded* if it does not occur inside any ambient, or as the name of any ambient. The predicate $\mathsf{ug}(R, i)$ says that all ambient-unguarded names in R are labelled with i.

Definition 5.7 *The predicate $\mathsf{ug}(R, i)$ is defined by structural induction on* *labelled processes:*

$$\begin{array}{rlrl}
\mathsf{ug}(\mathbf{0}, i) & \textit{always} & \mathsf{ug}(R \mid S, i) & \Leftrightarrow \mathsf{ug}(R, i) \wedge \mathsf{ug}(S, i) \\
\mathsf{ug}(\nu n^j\, R, i) & \Leftrightarrow \mathsf{ug}(R, i) & \mathsf{ug}(\mathsf{in}\; n^j.R, i) & \Leftrightarrow i = j \wedge \mathsf{ug}(R, i) \\
\mathsf{ug}(\,!\,R, i) & \Leftrightarrow \mathsf{ug}(R, i) & \mathsf{ug}(\mathsf{out}\; n^j.R, i) & \Leftrightarrow i = j \wedge \mathsf{ug}(R, i) \\
\mathsf{ug}(n^j[\,R\,], i) & \textit{always} & \mathsf{ug}((n^j).R, i) & \Leftrightarrow i = j \wedge \mathsf{ug}(R, i) \\
\mathsf{ug}(\langle n^j \rangle, i) & \Leftrightarrow i = j & &
\end{array}$$

The interesting cases are ambient and restriction. An ambient serves as a base case for the predicate, meaning that we do not look inside it. In the case of restriction, since the scope of a restriction can be altered using structural congruence, we do not insist that the binder νn^j is labelled with i. However, all other ambient-unguarded occurrences of a bound name will be labelled with i.

Definition 5.8 *We say that R is* coherent *if the predicate* $\mathsf{ch}(R)$ *holds. This predicate is defined by structural induction on labelled processes:*

$$\begin{array}{rlrl}
\mathsf{ch}(\mathbf{0}) & \textit{always} & \mathsf{ch}(R \mid S) & \Leftrightarrow \mathsf{ch}(R) \wedge \mathsf{ch}(S) \\
\mathsf{ch}(\nu n^i\, R) & \Leftrightarrow \mathsf{ch}(R) & \mathsf{ch}(\mathsf{in}\; n^i.R) & \Leftrightarrow \mathsf{ch}(R) \wedge \mathsf{ug}(R, i) \\
\mathsf{ch}(\,!\,R) & \Leftrightarrow \mathsf{ch}(R) & \mathsf{ch}(\mathsf{out}\; n^i.R) & \Leftrightarrow \mathsf{ch}(R) \wedge \mathsf{ug}(R, i) \\
\mathsf{ch}(n^i[\,R\,]) & \Leftrightarrow \mathsf{ch}(R) \wedge \mathsf{ug}(R, i) & \mathsf{ch}((n^i).R) & \Leftrightarrow \mathsf{ch}(R) \wedge \mathsf{ug}(R, i) \\
\mathsf{ch}(\langle n^i \rangle) & \textit{always} & &
\end{array}$$

Coherence is preserved by structural congruence:

Lemma 5.9 *Let R, S be labelled processes with $R \equiv S$.*

- *For any label i, if* $\mathsf{ug}(R, i)$ *then* $\mathsf{ug}(S, i)$.
- *If* $\mathsf{ch}(R)$ *then* $\mathsf{ch}(S)$.

PROOF. Straightforward and omitted. $\square$

Definition 5.10 *Let the reduction relation $\rightarrow_{\text{iosc}}$ on labelled processes be defined by the usual rules for boxed MA (Section 2), but where the rule (Comm) is replaced by the following rule:*

$$\textit{(SameComm)}\ \langle n^i \rangle \mid (m^i).R \rightarrow R\{n^i/m^i\}$$

This means that communication is only allowed when the input and output labels match. ("iosc" abbreviates "in, out, same-label communication".)

By a *same-label* substitution we mean a substitution on labelled names where a labelled name is replaced by a labelled name with the same label. For example $\sigma = \{m^i/p^k\}$ is not a same-label substitution if $i \neq k$, whereas $\sigma = \{m^k/p^k\}$ *is* a same label substitution. Same-label substitutions preserve coherence:

Lemma 5.11 *For all labelled processes S and same-label substitutions σ:*

(1) for all i, if $\mathsf{ug}(S, i)$ then $\mathsf{ug}(\sigma(S), i)$.
(2) If $\mathsf{ch}(S)$ then $\mathsf{ch}(\sigma(S))$.

PROOF. Both cases are proved by induction on S. $\square$

Coherence is preserved by $\rightarrow_{\text{iosc}}$ reductions:

Lemma 5.12 *Suppose $R \rightarrow_{\text{iosc}} S$.*

(1) For any label i, if $\mathsf{ug}(R, i)$ then $\mathsf{ug}(S, i)$.
(2) If $\mathsf{ch}(R)$ then $\mathsf{ch}(S)$.

PROOF.

(1) By induction on $\rightarrow_{\text{iosc}}$.
(In) Suppose that

$$m^j[\,\mathsf{in}\, n^k.R \mid R'\,] \mid n^h[\,R''\,] \rightarrow_{\text{iosc}} S$$

where $S = n^h[\,m^j[\,R \mid R'\,] \mid R''\,]$. Then $\mathsf{ug}(S, i)$ follows straight from Definition 5.7 (there is no need to assume that $\mathsf{ug}(m^j[\,\mathsf{in}\, n^k.R \mid R'\,] \mid n^h[\,R''\,], i)$ holds).
(Out) Assume that

$$n^h[\,m^j[\,\mathsf{out}\, n^k.R \mid R'\,] \mid R''\,] \rightarrow_{\text{iosc}} S$$

and $\mathsf{ug}(n^h[\,m^j[\,\mathsf{out}\, n^k.R \mid R'\,] \mid R''\,], i)$. Then

$$S = m^j[\,R \mid R'\,] \mid n^h[\,R''\,]$$

and, observing that by Definition 5.7 $\mathsf{ug}(m^j[\,R \mid R'\,], i)$ and $\mathsf{ug}(n^h[\,R''\,], i)$, we conclude $\mathsf{ug}(S, i)$.
(SameComm) Assume that $\mathsf{ug}(\langle n^j\rangle \mid (m^j).R, i)$. Then $\mathsf{ug}(R, i)$ by Definition 5.7 (also $i = j$, but we do not need this information). Then $\mathsf{ug}(R\{n^j/m^j\}, i)$ follows from Lemma 5.11.

(Par) Assume that $\mathsf{ug}(R \mid S, i)$ and $R \mid S \rightarrow_{\text{iosc}} R' \mid S$. By Definition 5.7, $\mathsf{ug}(R, i)$ and $\mathsf{ug}(S, i)$. By induction $\mathsf{ug}(R', i)$, and we conclude $\mathsf{ug}(R' \mid S, i)$.

(Res) The proof is similar to the previous case.

(Amb) Trivial, by Definition 5.7.

(Str) Assume that $R \rightarrow_{\text{iosc}} S$ where $R \equiv R' \rightarrow_{\text{iosc}} S' \equiv S$. Suppose $\mathsf{ug}(R, i)$. By Lemma 5.9 we have $\mathsf{ug}(R', i)$ and by induction, $\mathsf{ug}(S', i)$. Again by Lemma 5.9 we have $\mathsf{ug}(S, i)$, as required.

(2) By induction on $\rightarrow_{\text{iosc}}$.

(In) Assume that

$$m^i[\,\mathsf{in}\, n^i.R \mid R'\,] \mid n^h[\,R''\,] \rightarrow_{\text{iosc}} S.$$

and that $\mathsf{ch}(m^i[\,\mathsf{in}\, n^i.R \mid R'\,] \mid n^h[\,R''\,])$ holds. Thus:

$$\begin{aligned}
\mathsf{ch}(m^i[\,\mathsf{in}\, n^i.R \mid R'\,]) \mid n^h[\,R''\,] &\text{ iff } \mathsf{ch}(m^i[\,\mathsf{in}\, n^i.R \mid R'\,]) \wedge \mathsf{ch}(n^h[\,R''\,]) \\
&\text{ iff } \mathsf{ch}(\mathsf{in}\, n^i.R) \wedge \mathsf{ch}(R') \\
&\qquad \wedge \mathsf{ug}(\mathsf{in}\, n^i.R, i) \wedge \mathsf{ug}(R', i) \\
&\qquad \wedge \mathsf{ch}(R'') \wedge \mathsf{ug}(R'', h)
\end{aligned}$$

By the reduction, $S = n^h[\,m^i[\,R \mid R'\,] \mid R''\,]$. Now we are going to show that S is coherent. Since $\mathsf{ch}(\mathsf{in}\, n^i.R)$, we have $\mathsf{ch}(R)$ by Definition 5.8. Since $\mathsf{ug}(\mathsf{in}\, n^i.R, i)$, we have $\mathsf{ug}(R, i)$ by Definition 5.7. We deduce $\mathsf{ch}(m^i[\,R \mid R'\,])$ and $\mathsf{ug}(m^i[\,R \mid R'\,], h)$. Since $\mathsf{ch}(R'')$ and $\mathsf{ug}(R'', h)$ hold, we conclude $\mathsf{ch}(m^i[\,R \mid R'\,] \mid R'')$ and $\mathsf{ug}(m^i[\,R \mid R'\,] \mid R'', h)$, from which we conclude by Definition 5.8 that $\mathsf{ch}(S)$.

(Out) Assume that $n^h[\,m^i[\,\mathsf{out}\, n^i.R \mid R'\,] \mid R''\,]$ is coherent and

$$n^h[\,m^i[\,\mathsf{out}\, n^i.R \mid R'\,] \mid R''\,] \rightarrow_{\text{iosc}} S\ .$$

Thus:

$$\begin{aligned}
\mathsf{ch}(n^h[\,m^i[\,\mathsf{out}\, n^i.R \mid R'\,] \mid R''\,]) &\text{ iff } \mathsf{ch}(m^i[\,\mathsf{out}\, n^i.R \mid R'\,] \mid R'') \\
&\qquad \wedge \mathsf{ug}(m^i[\,\mathsf{out}\, n^i.R \mid R'\,] \mid R'', h) \\
&\text{ iff } \mathsf{ch}(\mathsf{out}\, n^i.R) \wedge \mathsf{ch}(R') \\
&\qquad \wedge \mathsf{ug}(\mathsf{out}\, n^i.R, i) \wedge \mathsf{ug}(R', i) \\
&\qquad \wedge \mathsf{ch}(R'') \wedge \mathsf{ug}(R'', h)
\end{aligned}$$

Now $S = m^i[\,R \mid R'\,] \mid n^h[\,R''\,]$. Since $\mathsf{ch}(\mathsf{out}\, n^i.R)$, we have $\mathsf{ch}(R)$ by Definition 5.8. Similarly, since $\mathsf{ug}(\mathsf{out}\, n^i.R,)$, we have $\mathsf{ug}(R, i)$ by Definition 5.7. Therefore $\mathsf{ch}(R \mid R')$ and $\mathsf{ug}(R \mid R', i)$, from which we deduce that $\mathsf{ch}(m^i[\,R \mid R'\,])$. Moreover, it is easy to see from above that $\mathsf{ch}(n^h[\,R''\,])$; thus from Definition 5.8 we conclude $\mathsf{ch}(S)$.

(SameComm) Assume that $\langle n^i \rangle \mid (y^i).R$ is coherent and $\langle n^i \rangle \mid (y^i).R \rightarrow_{\text{iosc}} S$. We show that S is coherent.

$$\begin{aligned}\mathsf{ch}(\langle n^i \rangle \mid (y^i).R) &\text{ iff } \mathsf{ch}(\langle n^i \rangle) \wedge \mathsf{ch}((y^i).R)\\ &\text{ iff } \mathsf{ch}(R) \wedge \mathsf{ug}(R, i)\end{aligned}$$

But since $S = R\{n^i/y^i\}$ and $\{n^i/y^i\}$ is a same-label substitution, by Lemma 5.11(2) we conclude $\mathsf{ch}(S)$.

(Par) Assume that $\mathsf{ch}(S \mid T)$ and $S \mid T \rightarrow_{\text{iosc}} S' \mid T$ where $S \rightarrow_{\text{iosc}} S'$. Then by Definition 5.8 $\mathsf{ch}(S)$ and $\mathsf{ch}(T)$ and by induction $\mathsf{ch}(S')$. By Definition 5.8 we deduce $\mathsf{ch}(S' \mid T)$.

(Res) Assume that $\mathsf{ch}(\nu n^i\, S)$ holds and $\nu n^i\, S \rightarrow_{\text{iosc}} \nu n^i\, S'$ where $S \rightarrow_{\text{iosc}} S'$. Then by Definition 5.8 we have $\mathsf{ch}(S)$. By induction $\mathsf{ch}(S')$ holds, and by Definition 5.7 we have $\mathsf{ch}(\nu n^i\, S')$.

(Amb) Assume that $\mathsf{ch}(n^i[\,R\,])$ and $n^i[\,R\,] \rightarrow_{\text{iosc}} n^i[\,R'\,]$ with $R \rightarrow_{\text{iosc}} R'$. By Definition 5.8 we deduce $\mathsf{ch}(R)$ and $\mathsf{ug}(R, i)$. By induction $\mathsf{ch}(R')$ and from part (1) of this lemma we deduce $\mathsf{ug}(R', i)$, implying that $\mathsf{ch}(n^i[\,R'\,])$.

(Str) This follows from induction and Lemma 5.9.

□

We define three further reduction relations:

Definition 5.13 • *For both standard processes and labelled processes, the reduction relation $\rightarrow_{\text{tc}}$ is defined by the rules (Comm), (Par), (Res) and (Str) (Section 2). We refer to these reductions as* top-level communications. *Notice that we omit rule (Amb). ("tc" abbreviates "top-level communication".)*

- *For labelled processes only, the reduction relation $\rightarrow_{\text{stc}}$ is defined by the rules (SameComm), (Par), (Res) and (Str). ("stc" abbreviates "same-label top-level communication".)*
- *For both standard processes and labelled processes, the reduction relation $\rightarrow_{\text{iolc}}$ is defined by the usual rules for boxed MA, but where any uses of (Comm) must be* lower-level, *i.e. must be combined with a use of (Amb). ("iolc" abbreviates "in, out, lower-level communication".)*

Clearly, if $P \rightarrow Q$ then at least one of $P \rightarrow_{\text{tc}} Q$ or $P \rightarrow_{\text{iolc}} Q$, and similarly for labelled processes.

Coherence is preserved by $\rightarrow_{\text{stc}}$ and $\rightarrow_{\text{iolc}}$-reductions:

Lemma 5.14 *Let R, S be labelled processes and let $\mathsf{ch}(R)$. Suppose $R \rightarrow_{\text{stc}} S$ or $R \rightarrow_{\text{iolc}} S$. Then $R \rightarrow_{\text{iosc}} S$, so that by Lemma 5.12 we have $\mathsf{ch}(S)$.*

PROOF. Any $\rightarrow_{\mathrm{stc}}$ reduction is an $\rightarrow_{\mathrm{iosc}}$-reduction by definition. Any $\rightarrow_{\mathrm{iolc}}$ reduction on a coherent process is an $\rightarrow_{\mathrm{iosc}}$-reduction, since coherence ensures that any lower-level (Comm) redex must have the same labels in both input and output. $\square$

Lemma 5.15 *Suppose that $P \not\rightarrow_{\mathrm{tc}}$ and $P \rightarrow Q$. Then $Q \not\rightarrow_{\mathrm{tc}}$. Similarly for labelled processes, if $R \not\rightarrow_{\mathrm{tc}}$ and $R \rightarrow S$ then $S \not\rightarrow_{\mathrm{tc}}$.*

The idea of the lemma is that (In) or (Out) reductions and lower-level communications can never create (Comm) redexes at the top level. Reductions inside an ambient cannot produce anything new at the top level. A top-level (In) reduction removes an ambient from the top-level, while a top-level (Out) reduction out can produce only a new ambient at the top level, which cannot be part of a top-level (Comm) redex. Notice that the lemma would not hold if (Open) reductions were allowed.

When attempting to prove Lemma 5.15 by induction on the derivation of $Q \rightarrow_{\mathrm{tc}}$, we come up against a problem with the rule (Par), since it is not in general the case that $P_1 \mid P_2 \rightarrow_{\mathrm{tc}}$ implies $P_1 \rightarrow_{\mathrm{tc}}$ or $P_2 \rightarrow_{\mathrm{tc}}$. Being able to perform a $\rightarrow_{\mathrm{tc}}$-reduction entails having both input and output present. Our strategy for proving Lemma 5.15 is to prove versions for input and output separately (Lemma 5.17 below).

We define predicates which express whether a process has a top-level unguarded input or output. By "unguarded" we here mean unguarded by a capability or an input.

Definition 5.16 *(Top-level unguarded input and output) Let predicates $P \downarrow_{()}$, $P \downarrow_{\langle\rangle}$ be defined by induction on boxed MA processes P to be the least relations satisfying:*

- *$(n).P \downarrow_{()}$; if $P \downarrow_{()}$ then $\nu n\, P \downarrow_{()}$ and $!\,P \downarrow_{()}$; and if $P \downarrow_{()}$ or $Q \downarrow_{()}$ then $(P \mid Q) \downarrow_{()}$.*
- *$\langle n\rangle \downarrow_{\langle\rangle}$; if $P \downarrow_{\langle\rangle}$ then $\nu n\, P \downarrow_{\langle\rangle}$ and $!\,P \downarrow_{\langle\rangle}$; and if $P \downarrow_{\langle\rangle}$ or $Q \downarrow_{\langle\rangle}$ then $(P \mid Q) \downarrow_{\langle\rangle}$.*

Similarly for labelled processes.

Lemma 5.17 *(1) If $P \equiv Q$ then $P \downarrow_{()}$ iff $Q \downarrow_{()}$.*
(2) If $P \equiv Q$ then $P \downarrow_{\langle\rangle}$ iff $Q \downarrow_{\langle\rangle}$.
(3) If $P \rightarrow Q$ and $Q \downarrow_{()}$ then $P \downarrow_{()}$.
(4) If $P \rightarrow Q$ and $Q \downarrow_{\langle\rangle}$ then $P \downarrow_{\langle\rangle}$.

Similarly for labelled processes.

PROOF. Straightforward and omitted. $\Box$

Lemma 5.18 *(1) $P \downarrow_{()}$ iff $P \equiv \nu\vec{m}\,((n).P' \mid P'')$ for some $\vec{m}, n, P', P''$*
(2) $P \downarrow_{\langle\rangle}$ iff $P \equiv \nu\vec{m}\,(\langle n\rangle \mid P')$ for some $\vec{m}, n, P'$

Similarly for labelled processes.

PROOF. Straightforward and omitted. $\Box$

Lemma 5.19 *Let P be a process. Then $P \rightarrow_{\rm tc}$ iff both $P \downarrow_{()}$ and $P \downarrow_{\langle\rangle}$.*

Similarly for labelled processes.

PROOF. ($\Rightarrow$) By induction on the derivation of $P \rightarrow_{\rm tc}$.

($\Leftarrow$) By structural induction on P. The only non-trivial case is parallel composition. Suppose that $(P \mid Q) \downarrow_{()}$ and $(P \mid Q) \downarrow_{\langle\rangle}$. Then $P \downarrow_{()}$ or $Q \downarrow_{()}$. Also $P \downarrow_{\langle\rangle}$ or $Q \downarrow_{\langle\rangle}$. If $P \downarrow_{()}$ and $P \downarrow_{\langle\rangle}$ then we use the induction hypothesis. Similarly if $Q \downarrow_{()}$ and $Q \downarrow_{\langle\rangle}$. So suppose that $P \downarrow_{()}$ and $Q \downarrow_{\langle\rangle}$. Then by Lemma 5.18 we have $P \equiv \nu\vec{m}\,((m).P' \mid P'')$ and $Q \equiv \nu\vec{n}\,(\langle n\rangle \mid Q')$. It follows that $P \mid Q \rightarrow_{\rm tc}$. Similarly if $P \downarrow_{\langle\rangle}$ and $Q \downarrow_{()}$.

Similarly for labelled processes. $\Box$

We now have enough to prove Lemma 5.15:

PROOF of Lemma 5.15. We give the proof for standard processes. The proof for labelled processes is similar.

Suppose that $P \not\rightarrow_{\rm tc}$ and $P \rightarrow Q$. Suppose for a contradiction that $Q \rightarrow_{\rm tc}$. Then $Q \downarrow_{()}$ and $Q \downarrow_{\langle\rangle}$ by Lemma 5.19. So $P \downarrow_{()}$ and $P \downarrow_{\langle\rangle}$ by Lemma 5.17. Hence $P \rightarrow_{\rm tc}$, again by Lemma 5.19. Contradiction. $\Box$

Now we can give the proof of Theorem 5.3.

PROOF of Theorem 5.3. Let $k = rs$, with $r, s \geq 2$. Suppose that $\mathsf{Net}' \overset{\rm df}{=} \nu A\,(P_0 \mid \cdots \mid P_{k-1})$ is a symmetric ring in boxed MA. We must show that Net' is not an electoral system. First we observe that we can eliminate the globally-bound names A. Let $\mathsf{Net} \overset{\rm df}{=} P_0 \mid \cdots \mid P_{k-1}$. Then Net is also a symmetric ring of size k. If Net' is an electoral system then so is Net. Hence it is enough to show that Net is not an electoral system.

By Definition 3.9, there is a single-orbit automorphism τ such that Net is symmetric with respect to τ, and such that for all $i, j < k$, if $\mathsf{fn}(P_i) \cap \mathsf{fn}(P_j) \neq \emptyset$ then one of $i = j$, $\hat{\tau}(i) = j$ or $\hat{\tau}(j) = i$ must hold. Let $\sigma \stackrel{\mathrm{df}}{=} \tau^r$. Then, as in the proof of Lemma 3.12, σ is non-trivial, well-balanced and has independent orbits (of size s). Also Net is symmetric with respect to σ.

We start by labelling every name (whether free or bound) occurring in P_i with i, to create the coherent labelled process R_i. Our aim is to build a maximal computation of $R \stackrel{\mathrm{df}}{=} R_0 \mid \cdots \mid R_{k-1}$ which preserves symmetry with respect to σ. When the labels are removed (Lemma 5.5) this yields a maximal computation of Net which preserves symmetry with respect to σ. This computation cannot declare a unique winner, and so Net is not an electoral system.

An arbitrary state of the labelled network during the computation will be of the form $R' = \nu \vec{n^l}\,(R'_0 \mid \cdots R'_{k-1})$, where restrictions are brought to the outside as much as possible. We ensure that R' is coherent by only ever using $\rightarrow_{\mathrm{iosc}}$-reductions (Lemma 5.12). The labelling tells us which portions of R' belong to which R'_i: any top-level threads or ambients whose names are labelled with i are assigned to R_i. This is well-defined by coherence of R'.

The first phase is to exhaust all possible $\rightarrow_{\mathrm{tc}}$-reductions. During this phase R' remains symmetric with respect to the single-orbit automorphism τ. Suppose that $R' \rightarrow_{\mathrm{tc}}$. Then some R'_i must have a top-level input, and some R'_j must have a top-level output. By symmetry all R'_is must have top-level inputs and outputs. But then R'_i must be of the form $\langle n^i \rangle \mid (m^i).S_i \mid \cdots$, and $R'_i \rightarrow_{\mathrm{stc}} S_i\{n^i/m\} \mid \cdots$. By symmetry, we cause each R'_i to perform a same-label communication within itself. These k reductions preserve coherence (Lemma 5.14) and symmetry with respect to τ (and also σ).

We continue this process until R' cannot perform any more $\rightarrow_{\mathrm{tc}}$-reductions. (Of course, the process may carry on for ever, but this gives us the desired symmetric maximal computation.) By Lemma 5.15, any further reductions must be $\rightarrow_{\mathrm{iolc}}$-reductions, which preserve coherence by Lemma 5.14.

Note that using the more refined τ-symmetry in the first phase allowed us to keep all top-level communications internal to individual R'_is. We now need to use σ-symmetry to deal with (In) reductions, which do not preserve τ-symmetry. Any further communications will be lower-level, and therefore within a single process.

In the second phase, we preserve symmetry with respect to σ, by propagating each reduction around the orbit(s) of the processes concerned.

We consider each type of $\rightarrow_{\mathrm{iolc}}$-reduction: If it comes from a lower-level communication, then it only involves one R'_i: $R'_i \rightarrow_{\mathrm{iolc}} R''_i$. We restore symmetry by performing $\sigma(R'_i) \rightarrow_{\mathrm{iolc}} \sigma(R''_i), \ldots$ round the σ-orbit of i.

If the reduction comes from the (In) rule, then either a single R'_i is involved, in which case we proceed as in the first case, or else R'_i interacts with R'_j ($i \neq j$). In this case R'_i is of the form $n^i[\,\mathsf{in}\, m^i.S \mid S'\,] \mid S''$, and R'_j is of the form $m^j[\,S'''\,] \mid S''''$. After the reduction we have $R''_i = S''$ and $R''_j = m^j[\,n^i[\,S \mid S'\,] \mid S'''\,] \mid S''''$. We claim that i and j are in different orbits. This is because m^i and m^j must both be free in R' (if they were bound then they would have the same label), and so must both occur free in the original R (Lemma 5.6). But this means that $m \in \mathsf{fn}(P_i) \cap \mathsf{fn}(P_j)$, and so i and j are indeed in different orbits by the independence of Net with respect to σ. We now restore σ-symmetry by performing $\sigma(R'_i \mid R'_j) \rightarrow_{\mathrm{iolc}} \sigma(R''_i \mid R''_j), \ldots$ round the σ-orbits of i and j.

If the reduction comes from a lower-level application of the (Out) rule, then a single R'_i is involved and we proceed as in the first case. If the reduction comes from a top-level application of the (Out) rule, then a new top-level ambient emerges from a single R'_i:

$$R'_i = m^i[\,n^j[\,\mathsf{out}\, m^j.S \mid S'\,] \mid S''\,] \mid S''' \rightarrow_{\mathrm{iolc}} m^i[\,S''\,] \mid S''' \mid n^j[\,S \mid S'\,]$$

If $i = j$ then we set $R''_i = m^i[\,S''\,] \mid S''' \mid n^j[\,S \mid S'\,]$. As in the first case we restore symmetry by performing $\sigma(R'_i) \rightarrow_{\mathrm{iolc}} \sigma(R''_i), \ldots$ round the σ-orbit of i. If $i \neq j$ then, as in the (In) case, i and j are in different orbits. The new top-level ambient will become part of the new jth process: $R''_i = m^i[\,S''\,] \mid S'''$ and $R''_j = R'_j \mid n^j[\,S \mid S'\,]$. We restore symmetry by performing $\sigma(R'_i \mid R'_j) \rightarrow_{\mathrm{iolc}} \sigma(R''_i \mid R''_j), \ldots$ round the σ-orbits of i and j.

$\square$

Remark 5.20 *All three capabilities (in, out and open) are essential to solving the leader election problem in symmetric rings of MA processes. The necessity for the open capability comes from Theorem 5.3 above. The need for the in capability has been shown elsewhere [17,20]: leader election cannot be solved on arbitrary symmetric networks of size ≥ 2 in MA without the in capability. A similar result in fact holds for the out capability.*

As stated at the start of this section, Theorem 5.3 also holds for PAC and SA. Let *boxed* PAC (resp. SA) denote PAC (resp. SA) without the open capability.

Theorem 5.21 *For any composite $k > 1$, boxed PAC does not have a symmetric ring of size k which is an electoral system. Similarly for boxed SA.*

PROOF. The proof is similar to that of Theorem 5.3. We omit the details.

Note that the result for boxed SA is in fact a strengthening of Theorem 5.3, since there is an encoding from boxed MA into boxed SA—see Remark 6.6 below. $\square$

Remark 5.22 *An alternative form of the (CoOut) rule of SA has been considered in [11]:*

$$m[\, n[\, \mathsf{out}\, m.P \mid P' \,] \mid Q \,] \mid \overline{\mathsf{out}}\, m.Q' \rightarrow n[\, P \mid P' \,] \mid m[\, Q \,] \mid Q'$$

If we adopted this formulation then the analogue of Lemma 5.15 would no longer hold, since the continuation Q' might give a new top-level (Comm) redex.

6 Separation Results

We use the results of Sections 4 and 5 to show that certain languages cannot be encoded in others.

We recall the following from [17] (building on [15]):

Definition 6.1 *Let L, L' be process languages. An encoding $[\![-]\!] : L \rightarrow L'$ is*

(1) distribution-preserving *if for all processes P, Q of L, $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$;*
(2) permutation-preserving *if for any permutation of names σ in L there exists a permutation θ in L' such that $[\![\sigma(P)]\!] = \theta([\![P]\!])$ and the permutations are compatible on observables, in that for all $i \in \mathbb{N}$ we have $\sigma(\omega_i) = \theta(\omega_i)$;*
(3) observation-respecting *if for any P in L,*
   (a) for every maximal computation $\mathcal{C}$ of P there exists a maximal computation $\mathcal{C}'$ of $[\![P]\!]$ such that $\mathsf{Obs}(\mathcal{C}) = \mathsf{Obs}(\mathcal{C}')$
   (b) for every maximal computation $\mathcal{C}$ of $[\![P]\!]$ there exists a maximal computation $\mathcal{C}'$ of P such that $\mathsf{Obs}(\mathcal{C}) = \mathsf{Obs}(\mathcal{C}')$

An encoding which preserves distribution and permutation is uniform.

Unlike in [17], we are considering encodings which map rings to rings. We therefore need a further property:

Definition 6.2 *An encoding is* independence-preserving *if for any processes P, Q, if P and Q are independent then $[\![P]\!]$ and $[\![Q]\!]$ are also independent.*

Palamidessi says that such an encoding "does not increase the level of connectivity of the network". Not all encodings preserve independence. For instance, Zimmer's [21] encoding of the synchronous π-calculus without choice into pure

Safe Ambients [9] introduces a new global ambient whose name is shared by all processes.

Lemma 6.3 *Suppose $[\![-]\!] : L \to L'$ is a uniform, observation-respecting and independence-preserving encoding. Suppose that Net is a symmetric ring of size $k \geq 1$ with no globally-bound names which is an electoral system. Then $[\![\mathsf{Net}]\!]$ is also a symmetric ring of size k with no globally-bound names which is an electoral system.*

PROOF. Assume that the network $\mathsf{Net} = P_0 \mid P_1 \mid \ldots \mid P_{k-1}$ of size k is a symmetric ring and an electoral system in L and that $[\![-]\!] : L \to L'$ is a uniform observation-respecting and independence-preserving encoding. We are going to show that $[\![P_0 \mid P_1 \mid \ldots \mid P_{k-1}]\!]$ is a symmetric ring and electoral system, i.e. every maximal computation yields one winner only. Since $[\![-]\!]$ is distribution-preserving (Definition 6.1(1)), it preserves the size of the network:

$$[\![P_0 \mid P_1 \mid \ldots \mid P_{k-1}]\!] = [\![P_0]\!] \mid [\![P_1]\!] \mid \ldots \mid [\![P_{k-1}]\!]$$

Let Net be symmetric with respect to σ with one orbit only, satisfying the ring property (Definition 3.9) that for all $i, j < k$, if $\mathsf{fn}(P_i) \cap \mathsf{fn}(P_j) \neq \emptyset$ then one of $i = j$, $\hat{\sigma}(i) = j$ or $\hat{\sigma}(j) = i$ must hold. By symmetry for all i $(0 \leq i \leq k-1)$ we have $\sigma(P_i) = P_{\hat{\sigma}(i)}$ and since $[\![-]\!]$ is permutation-preserving (Definition 6.1(2)), there exists a θ such that $\theta([\![P_i]\!]) = [\![\sigma(P_i)]\!]$ and $\hat{\theta}(i) = \hat{\sigma}(i)$ for all $i \in \mathbb{N}$.

$$\begin{aligned} \theta([\![P_i]\!]) &= [\![\sigma(P_i)]\!] \\ &= [\![P_{\hat{\sigma}(i)}]\!] \quad \text{by symmetry} \\ &= [\![P_{\hat{\theta}(i)}]\!] \quad \text{since } \hat{\theta}(i) = \hat{\sigma}(i) \end{aligned}$$

Hence $[\![\mathsf{Net}]\!]$ is symmetric with respect to θ (with one orbit only).

Now we shall see that $[\![\mathsf{Net}]\!]$ is a ring. Assume that $i, j < k$ are such that $\mathsf{fn}([\![P_i]\!]) \cap \mathsf{fn}([\![P_j]\!]) \neq \emptyset$. Then since the encoding is independence-preserving (Definition 6.2) we have $\mathsf{fn}(P_i) \cap \mathsf{fn}(P_j) \neq \emptyset$. Then we have one of $i = j$, $\hat{\sigma}(i) = j$ or $\hat{\sigma}(j) = i$. But then one of $i = j$, $\hat{\theta}(i) = j$ or $\hat{\theta}(j) = i$ must hold, showing that $[\![\mathsf{Net}]\!]$ is a ring.

It remains to show that $[\![\mathsf{Net}]\!]$ is an electoral system. Consider a maximal computation $\mathcal{C}'$ of $[\![\mathsf{Net}]\!]$. By condition 3(3b) of Definition 6.1 there must exist a computation $\mathcal{C}$ of Net such that $\mathsf{Obs}(\mathcal{C}) = \mathsf{Obs}(\mathcal{C}')$. Now since Net is an electoral system, every maximal computation exhibits one winner only. Hence $\mathsf{Obs}(\mathcal{C}) = \{\omega_j\}$ for some j such that $0 \leq j \leq k-1$, which implies that $\mathsf{Obs}(\mathcal{C}') = \{\omega_j\}$. Since this is true for every maximal computation $\mathcal{C}'$ of $[\![\mathsf{Net}]\!]$, the lemma is proven. $\square$

Theorem 6.4 *There is no uniform, observation-respecting and independence-preserving encoding from pure public MA into CCS (with value passing).*

PROOF. Suppose that such an encoding exists. Then by Theorem 4.3 and Lemma 6.3, for any $k \geq 1$ we would have a symmetric ring of size k in CCS which is an electoral system. But by Lemma 3.12 and Theorem 5.1, for any composite k, CCS does not have a symmetric ring of k processes which is an electoral system. □

Theorem 6.5 *(1) There is no uniform, observation-respecting and independence-preserving encoding from pure public MA into boxed MA.*
(2) There is no uniform, observation-respecting and independence-preserving encoding from pure public PAC into boxed PAC.

PROOF. The existence of symmetric rings of all sizes which are electoral systems in pure public MA and pure public PAC was shown in Theorems 4.3 and 4.2. The non-existence of symmetric rings of composite size which are electoral systems in boxed MA and boxed PAC was shown in Theorems 5.3 and 5.21. Hence the results follow from Lemma 6.3. □

It follows from Theorem 6.5 that the open capability of MA does indeed add expressive power not present in the other operators of MA.

Remark 6.6 *In fact part (1) of Theorem 6.5 can be strengthened to state that there is no uniform, observation-respecting and independence-preserving encoding from pure public MA into boxed SA. This is using Theorem 5.21 for boxed SA. We say "strengthened", because there is an encoding from boxed MA into boxed SA, namely*

$$[\![n[\,P\,]]\!] \stackrel{\mathrm{df}}{=} n[\ !\,\overline{\mathsf{in}}\,n \mid !\,\overline{\mathsf{out}}\,n \mid [\![P]\!]\,]$$

(with $[\![-]\!]$ homomorphic on the remaining operators) (cf. Remark 4.5). This encoding is uniform, observation-respecting and independence-preserving. Also the uniform, observation-respecting and independence-preserving conditions are preserved by composition of encodings.

Theorem 6.7 *There is no uniform, observation-respecting and independence-preserving encoding from BA into boxed MA (and therefore into pure BA).*

PROOF. From Theorem 4.7, Theorem 5.3 and Lemma 6.3. □

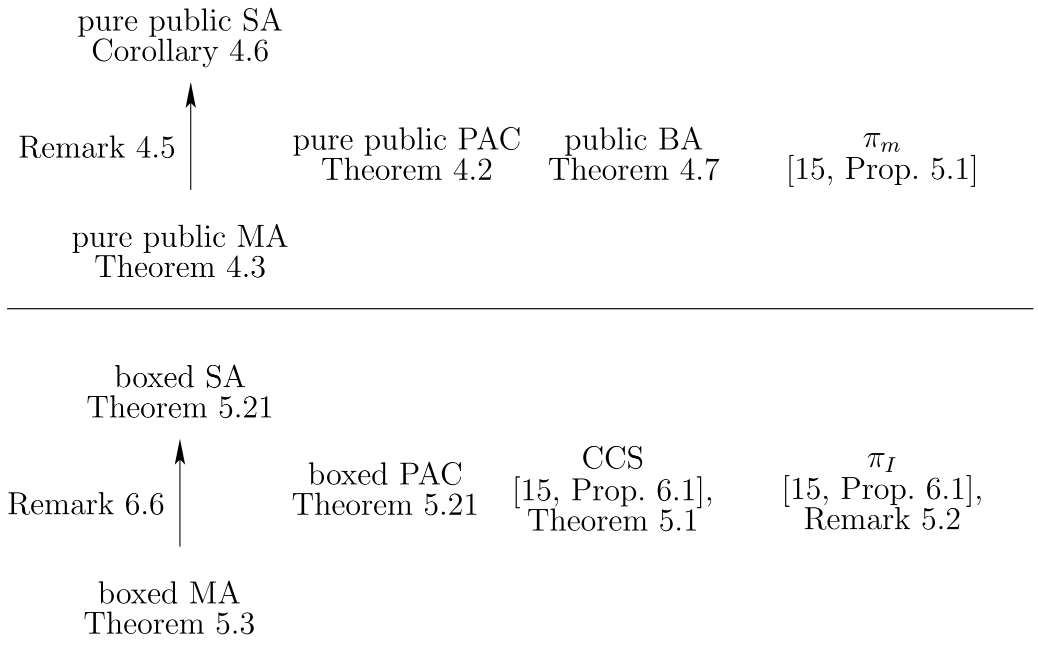

Fig. 2. Summary of results on leader election and encodings

It follows from Theorem 6.7 that the parent-child communication in BA does indeed add expressive power (without it, BA would be essentially boxed MA). Just like for part (1) of Theorem 6.5, we can strengthen Theorem 6.7 by replacing boxed MA by boxed SA.

In Proposition 5.1 of [15] Palamidessi showed how π-calculus with mixed choice, denoted π_m, can carry out an election on a ring of size four (which clearly generalises to larger rings). It is straightforward to adapt this to our reduction semantics setting. As we saw in the proof of Theorem 4.7 for BA, there are two phases. The first phase (passing names round the ring) can be carried out without choice. We adapted this for BA. The subsequent election phase makes essential use of mixed choice; in fact it can be carried out in CCS. Together with the negative result for π-calculus with internal mobility π_I (Remark 5.2) this yielded the result that π_I is somehow weaker than π_m. This also carries over to the present setting. Furthermore:

Theorem 6.8 *There is no uniform, observation-respecting and independence-preserving encoding from π_m into boxed MA.*

We summarise our results in a diagram (Figure 2). All calculi above the line have symmetric rings of every size which are electoral systems. Those calculi below the line do not have symmetric rings of composite size which are electoral systems. The arrows represent encodings which are uniform, observation-respecting and independence-preserving. By Lemma 6.3 there is no arrow going from any calculus above the line to any calculus below the line, which yields the separation results in this section, together with a number of other results.

7 Conclusions

In this paper we have shown how to elect a leader in a symmetric ring of processes in MA and its variants. We have seen that it can be done in pure public MA for a ring of any size, so that communication is unnecessary. On the other hand, the open capability is essential, since the election cannot be carried out in boxed MA (in fact the in and out capabilities are also essential—cf. [17]). Thus, simulating link-passing requires the open capability, but does not require the (anonymous) communication of MA. This shows that pure MA cannot be encoded either into CCS or into pure BA. In the case of BA, however, (parent-child) communication *is* necessary in order to elect a leader in rings, since the open capability is not present.

While our results shed light on the expressive power provided by the open capability, in the presence of the latter, leader election problems in both rings and fully connected graphs do not give any separation results between MA with communication primitives and pure MA. In this framework one could regard them as equal, since, when it comes to passing names around, pure MA can do just as well as the full calculus.

It is worth spending a few words on Theorem 5.3, which says that MA without the open capability (boxed MA) cannot solve the election problem on rings with a composite number of processes. If the number of processes is prime, then any well-balanced automorphism different from the identity has one orbit only, and our proof methods would not apply. This would be true for Palamidessi's work as well. Thus it is an open question as to whether election is impossible in rings of any prime size greater than three.

We also saw that boxed SA cannot solve the election problem on rings with a composite number of processes (Theorem 5.21). In connection with this, and recalling that Zimmer has encoded the synchronous choice-free π-calculus into pure SA, we conjecture that for boxed SA such an encoding would not be possible, even in the presence of communication. For if it were possible, then it would seem that boxed SA *could* perform election on rings, much as shown for BA (Theorem 4.7).

A challenge for the future is to find suitable conditions that differentiate those calculi that admit a solution to the leader election problem without having to know the size of the ring (such as PAC) from those that do need to know the size.

Acknowledgements

We wish to thank Catuscia Palamidessi for helpful discussions and the anonymous journal and Express 2004 referees for their helpful suggestions for improvement. M.G. Vigliotti was supported by the EPSRC grant PROFORMA GR/545140.

References

[1] D. Angluin, Local and global properties in networks of processors, in: Proceedings of the 12th Annual ACM Symposium on Theory of Computing, 1980, pp. 82–93.

[2] J. Bergstra, J. Klop, Process algebra for synchronous communication, Information and Control 60 (1-3) (1984) 109–137.

[3] I. Boneva, J.-M. Talbot, When ambients cannot be opened, Theoretical Computer Science 333 (1-2) (2005) 127–169.

[4] L. Bougé, On the existence of symmetric algorithms to find leaders in networks of communicating sequential processes, Acta Informatica 25 (1988) 179–201.

[5] M. Bugliesi, G. Castagna, S. Crafa, Access control for mobile agents: the calculus of Boxed Ambients, ACM Transactions on Programming Languages and Systems 26 (1) (2004) 57–124.

[6] L. Cardelli, A. Gordon, Mobile ambients, Theoretical Computer Science 240 (1) (2000) 177–213.

[7] X. Guan, Y. Yang, J. You, Making ambients more robust, in: Proceedings of International Conference on Software: Theory and Practice, Beijing, China, August 2000, 2000, pp. 377–384.

[8] C. Hoare, Communicating Sequential Processes, Prentice-Hall, 1985.

[9] F. Levi, D. Sangiorgi, Mobile safe ambients, ACM Transactions on Programming Languages and Systems 25 (1) (2003) 1–69.

[10] S. Maffeis, I. Phillips, On the computational strength of pure ambient calculi, Theoretical Computer Science 330 (3) (2005) 501–551.

[11] M. Merro, M. Hennessy, A bisimulation-based semantic theory of Safe Ambients, ACM Transactions on Programming Languages and Systems, in press 2005.

[12] R. Milner, Communication and Concurrency, Prentice-Hall, 1989.

[13] R. Milner, Communicating and Mobile Systems: the π-calculus, Cambridge University Press, 1999.

[14] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, Information and Computation 100 (1992) 1–77.

[15] C. Palamidessi, Comparing the expressive power of the synchronous and asynchronous π-calculus, Mathematical Structures in Computer Science 13 (5) (2003) 685–719.

[16] I. Phillips, M. Vigliotti, On reduction semantics for the push and pull ambient calculus, in: Proceedings of IFIP International Conference on Theoretical Computer Science (TCS 2002), IFIP 17th World Computer Congress, August 2002, Montreal, Kluwer, 2002, pp. 550–562.

[17] I. Phillips, M. Vigliotti, Electoral systems in ambient calculi, in: Proceedings of 7th International Conference on Foundations of Software Science and Computation Structures, FoSSaCS 2004, Vol. 2987 of Lecture Notes in Computer Science, Springer-Verlag, 2004, pp. 408–422.

[18] I. Phillips, M. Vigliotti, Leader election in rings of ambient processes, in: Proceedings of Express: Workshop on Expressiveness in Concurrency, London, August 2004, Vol. 128.2 of Electronic Notes in Theoretical Computer Science, Elsevier, 2005, pp. 185–199.

[19] D. Sangiorgi, Pi-calculus, internal mobility and agent-passing calculi, Theoretical Computer Science 167 (1-2) (1996) 235–274.

[20] M. Vigliotti, Reduction semantics for ambient calculi, Ph.D. thesis, Imperial College, University of London (2004).

[21] P. Zimmer, On the expressiveness of pure safe ambients, Mathematical Structures in Computer Science 13 (5) (2003) 721–770.