Symmetric Electoral Systems for Ambient Calculi [1]

Iain Phillips, Maria Grazia Vigliotti [*]

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

Abstract

This paper compares the expressiveness of different fragments of ambient calculi via leader election problems. We consider Mobile Ambients (MA), Safe Ambients (SA) and the Push and Pull Ambient Calculus (PAC). Cardelli and Gordon encoded the asynchronous pi-calculus into MA. Zimmer has shown that the synchronous pi-calculus without choice can be encoded in pure (no communication) SA. We show that pure MA without restriction has symmetric electoral systems, that is, it is possible to solve the problem of electing a leader in a symmetric network. By the work of Palamidessi, this implies that this fragment of MA is not encodable (under certain conditions) in the pi-calculus with separate choice. Moreover, we use the same technique to show that fragments of SA and PAC are not encodable (under certain conditions) in the pi-calculus with separate choice. We also show that particular fragments of ambient calculi do not admit a solution to leader election problems, in the same way as the pi-calculus with separate choice. This yields a fine-grained hierarchy within ambient calculi.

Key words: Ambient calculi, π-calculus, leader election, electoral system, expressiveness

[*] 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 version of this article appeared in FoSSaCS 2004 [27]. That version contains results about matching for Mobile Ambients and the non-encodability of Mobile Ambients in the π-calculus with mixed choice; those results do not relate to electoral systems, and are not reported here.

1 Introduction

The π-calculus [22,21] plays a fundamental role in modelling concurrent systems. In particular, the name passing paradigm, on which the π-calculus is based, has proved to be a powerful and simple framework for describing different scenarios appearing in concurrency.

In recent years many approaches [9,10,13] have been proposed in order to represent locations, code mobility, abstract domains and security, which seem to be the main features of computation over the World Wide Web. Mobile Ambients (MA) [8] has been advocated [7] as a foundational calculus for representing distributed computation, mobility in terms of software and hardware moving around, authorisation control etc., i.e. phenomena present over the Internet. The main advantage of MA is the simple underpinning unifying concept of *ambient*. Ambients are meant to represent bounded places for computation such as: concrete locations, concrete domains, abstract domains and laptop computers. Ambients move into and out of other ambients, bringing along moving code, static processes and possibly other ambients. Due to its simplicity and power, MA (together with its variants) has been widely studied.

When it comes to the comparison between these two fundamental process calculi, a basic issue is the extent to which the π-calculus (or any of its dialects) can be encoded into MA (or any of its dialects). The asynchronous π-calculus [16,3] (a fragment without the choice operator and with no continuation for the output), has been encoded into MA with the use of the communication primitives. There has been an encoding of the asynchronous π-calculus in the Push and Pull Ambient Calculus (PAC) [26], which preserves the contextual barbed congruence equivalence relation [31]. The synchronous π-calculus without choice has been encoded by Zimmer [32] into Safe Ambients (SA) [17] without communication; the encoding satisfies an operational correspondence. These encodings show that the behaviour of the asynchronous π-calculus can be simulated in the ambient world. This seems to imply that MA and some of its dialects are at least as expressive as the π-calculus (without choice).

This poses the question whether the ambient calculus (or any of its dialects) is more (or equally) expressive with respect to any of the dialects of the π-calculus. This paper directly addresses this open question. We approach the problem via the *leader election problem in symmetric networks*. This requires that the processes in a network, all programmed in the same way (i.e. symmetric), elect one of their member as their leader. The difficulty consists in breaking the initial symmetry to achieve a situation which is inherently asymmetric (one is the leader and the others are not) without the help of a centralised server.

A seminal result on expressiveness for the π-calculus is due to Palamidessi [24], who established that the π-calculus with mixed choice (i.e. where the summands in a choice can be a mixture of inputs and outputs) is strictly more expressive than the π-calculus with separate choice (i.e. where the summands must be all inputs or all outputs). This is proved by showing that the π-calculus with mixed choice can solve the symmetric leader election problem, while the π-calculus with separate choice cannot. This implies that there does not exist an encoding from the π-calculus with mixed choice to the π-calculus with separate choice that respects certain conditions.

These conditions are chosen in order to preserve important features of leader election. Typically leader election algorithms are run after a reconfiguration or crash of a distributed system (such as a sensor network, LAN etc.), to establish which process can start the initialisation, and at a later stage will act as server. It is crucial that the leader is elected without any help from processes not present in the initial network. Roughly speaking, Palamidessi's results establish that no encoding that does not introduce a centralised server exists from the π-calculus with mixed choice to the π-calculus with separate choice.

In this paper, we use the power to solve (or not solve) the leader election problem in symmetric networks as a measure for distinguishing different calculi. Taking inspiration from Palamidessi's work, we separate the π-calculus with separate choice from MA, and separate different fragments of ambient calculi. Our main contributions are as follows:

- We show that the fragment of MA *without restriction, communication primitives and the* open *capability* can solve the leader election problem for networks of any finite size (Theorem 4.6). This fragment (and so MA as a whole) is therefore not encodable in the π-calculus with separate choice (Corollary 6.4). In similar fashion we show that fragments of SA and PAC *without restriction, communication primitives and the* open *capability* can perform leader election (Theorems 4.8 and 4.7), and are therefore not encodable in the π-calculus with separate choice (Corollary 6.6). We can deduce that the converse to Zimmer's encoding result mentioned above does not hold.
- We show that a fragment of SA *without restriction, communication primitives and the* out *capability* can perform leader election (Theorem 4.10), and is therefore not encodable in the π-calculus with separate choice (Corollary 6.6 again).
- We show that if the in capability is removed from MA or SA, or if the pull capability is removed from PAC, then the leader election problem cannot be solved (Theorems 5.8 and 5.9).
- We show that (a dialect of) MA with *objective moves*, which contains a special form of the in capability, does not admit a solution to the leader

3

election problem (Theorem 7.1).
- We show that SA without *grave interferences* [17]—which does not allow certain forms of interference among redexes—does not admit a solution to the leader election problem (Theorem 7.5).

One important point here is that, by Palamidessi's work, in the π-calculus mixed choice seems crucial for writing a program that solves the problem of electing a leader in a symmetric network. Choice however is not present as a primitive construct in the ambient world. Our results shed light on the preemptive power of the in capability used in different ways in the ambient calculus. All our results give a fine-grained hierarchy in ambient calculi graphically shown in Figure 2 in the Concluding Remarks.

Our results are expressed in a reduction semantics framework instead of via labelled transition systems as in Palamidessi's work. There are a number of reasons for this choice:

(1) In MA, reduction semantics is simpler and more perspicuous than any labelled transition system.
(2) The original and intended semantics for MA was reduction semantics. Since then, a few different labelled transition systems have been devised; however they are all faithful, as far as silent actions are concerned, to reduction semantics.
(3) By globally restricting all names which are not used to report the winner, one can easily convert an electoral system in reduction semantics to one in the labelled semantics, and vice-versa.
(4) By working in reduction semantics we are able to give solutions to the leader election problem which do not use restriction. We therefore do not need to assume that encodings preserve restriction, when asserting that there is no encoding from a language which has electoral systems to one which does not.

As explained also earlier, our results crucially depend on the definition of *encoding*. In this paper we employ encodings which are "distribution preserving", "permutation preserving" and "observation respecting", very much following the same criteria as in [24]. These criteria are not proposed as a universal measure to evaluate the robustness or faithfulness of encodings in general. They are chosen to preserve solutions to the leader election problem, without introducing a central server. "Distribution preserving" means preserving parallel composition in the encoding, to avoid the translation making use of third parties—i.e. introducing the equivalent of a centralised server. "Permutation preserving" means that the encodings are well-behaved with respect to bijective renaming. "Observation respecting" means that processes are distinguished if they differ on the observable properties of their maximal computations. This last condition reflects the fact that failure or success of election

4

of a leader is tested for maximal computations only. These criteria will be formalised in our semantics in Section 6.

This paper is a substantial extension of our previous paper [27]. In this new version we have extended our results on MA to PAC and SA, and have introduced a new solution for the leader election problems in MA for a network of size k and extended the solution to PAC and SA. The material on grave interference in SA is new. We also consider a side issue of failure to elect a leader in the case in which the winner cannot be reported, due to some deficiency in the language (we might compare this situation to a machine that cannot output anything). This is of lesser importance than the breaking of the initial symmetry, which is at the core of leader election. We show that the symmetric leader election problem cannot be solved in MA without the out capability (Proposition 8.1). We also give corresponding results for SA and PAC (Propositions 8.2 and 8.3).

The rest of the paper is organised as follows: in Section 2 we present the preliminaries for the π-calculus, MA, SA and PAC; in Section 3 we define a general framework for electoral systems in reduction semantics; in Section 4 we present calculi that admit a symmetric electoral system and in Section 5 we present calculi that fail to elect a leader in symmetric networks. In Section 6 we use the results of Sections 4 and 5 to obtain separation theorems. In Section 7.1 we consider MA with objective moves and SA without grave interferences. In Section 8 we show that the out capability of MA is necessary in order to declare the result of a leader election. Concluding remarks follow.

2 Calculi

In this section we review the π-calculus and the ambient calculi considered in this paper.

2.1 The π-calculus

The π-calculus was originally introduced [22] with the aim of representing systems whose topology changes during computation. Communication involves two processes, and a common link of communication: a channel on which messages can be passed. The novelty of the π-calculus is that channel names can be transferred from process to process as messages, and then used as channels in later computation. In this setting, both channels and messages are drawn from a set of atomic entities, called *names*.

The π-calculus marked an advance on previous process calculi, such as CSP [15], CCS [20] and ACP [2], in pioneering the notions of both *mobility* and *scope extrusion*. *Mobility* is represented by the changing pattern of links between processes. *Scope extrusion* means that the scope of private names can be extended by communication.

We consider two forms of the π-calculus, namely with mixed choice and separate choice. For further details we refer to [29].

2.1.1 The π-calculus with mixed choice

We let π_{m} denote the π-calculus with mixed choice. We shall assume the existence of a set of names $\mathcal{N}$. The metavariables $m, n, x, y, z, \ldots$ range over this set.

Definition 2.1 *The set of process terms of π_{m} is given by the following syntax:*

$$P, Q ::= \mathbf{0} \ \mid \ \sum_{i \in I} \alpha_i.P_i \ \mid \ P \mid Q \ \mid \ (\nu m)P \ \mid \ !P$$

where I is a finite set. The prefixes *of processes, ranged over by α, are defined by the following syntax:*

$$\alpha ::= m(n) \ \mid \ \overline{m}\langle n \rangle$$

Here $m(n)$ represents input on channel m, with n bound, and $\overline{m}\langle n \rangle$ is output of n on channel m. *Summation* $\sum_{i \in I} \alpha_i.P_i$ represents a finite choice among the different processes $\alpha_i.P_i$. This operator is also called *mixed choice*, since both input and output prefixes can be present at the same time. The symbol $\mathbf{0}$, called *nil*, is the inactive process. Commonly in the π-calculus, $\mathbf{0}$ is an abbreviation for the empty choice. Although redundant, we introduce it here as a primitive for uniformity with the syntax of other calculi. *Replication* $!P$ can spin off an unbounded number of copies of P. The *parallel composition* of two processes $P \mid Q$ represents P and Q computing in parallel with each other. *Restriction* $(\nu n)P$ creates a new name n in P, which is bound.

In the rest of this paper, we shall feel free to omit trailing $\mathbf{0}$s. Thus we write α instead of $\alpha.\mathbf{0}$. We shall write $(\nu n_1 \ldots n_k)P$ instead of $(\nu n_1) \ldots (\nu n_k)P$, and sometimes we write $\tilde{n}$ for $n_1 \ldots n_k$, when k is irrelevant or clear from the context.

The notion of *free names*, $fn(P)$, of a term P is standard, taking into account that the only binding operators are input prefix and restriction. We write $P\{n/m\}$ to mean the process where each free occurrence of m is substituted by n in P.

The notion of α-convertibility, which aims to capture the equivalence between terms that differ in their bound names only, will be used throughout this work. In contrast to other treatments of the π-calculus such as [21,25], α-conversion will be performed silently. In general, we conventionally assume that free and bound names are different.

Reduction semantics is usually defined in two steps: firstly *structural congruence*, and secondly the *reduction relation* that captures computation on terms. *Structural congruence*, written as $\equiv$, identifies processes that we do not want to differentiate for any semantic reason; it allows syntactical rearrangement of contiguous terms not in the syntactical form for being reduced.

Definition 2.2 Structural congruence $\equiv$ *is the smallest congruence on* π_m *processes that satisfies the following equations:*

$$
\begin{aligned}
P \mid \mathbf{0} &\equiv P \\
P \mid Q &\equiv Q \mid P \\
(P \mid Q) \mid R &\equiv P \mid (Q \mid R) \\
(\nu m)\mathbf{0} &\equiv \mathbf{0} \\
(\nu m)(\nu n)P &\equiv (\nu n)(\nu m)P \\
(\nu m)(P \mid Q) &\equiv P \mid (\nu m)Q \quad \text{if } m \notin \mathit{fn}(P) \\
!P &\equiv P \mid !P \\
\textstyle\sum_{i \in I} \alpha_i.P_i &\equiv \textstyle\sum_{i \in I} \alpha_{\eta(i)}.P_{\eta(i)}
\end{aligned}
$$

where η *is a bijection on* I.

The computational step is captured by a rewriting rule from terms to terms, as defined below. We let S, T range over summations.

Definition 2.3 *The* reduction relation $\longrightarrow$ *on* π_m *is the smallest relation*

7

satisfying the following rules:

$$(m(x).P + S) \mid (\overline{m}\langle y\rangle.Q + T) \longrightarrow P\{y/x\} \mid Q \quad \text{RED COMM}$$

$$\frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \quad \text{RED PAR}$$

$$\frac{P \longrightarrow P'}{(\nu m)P \longrightarrow (\nu m)P'} \quad \text{RED RESTR}$$

$$\frac{P \equiv Q \quad Q \longrightarrow Q' \quad Q' \equiv P'}{P \longrightarrow P'} \quad \text{RED CONG}$$

We shall write $P \nrightarrow$ to mean that for no P' does $P \longrightarrow P'$.

The following barbs represent the most basic observations we can make of processes.

Definition 2.4 *A process P exhibits an output barb $\overline{n}$, written $P \downarrow \overline{n}$, if and only if, for some $P', P'', S,\ P \equiv (\nu m_1 \ldots m_k)((\overline{n}\langle q\rangle.P' + S) \mid P'')$ with $n \notin \{m_1, \ldots, m_k\}$.*

2.1.2 The π-calculus with separate choice

The choice operator as described in Section 2.1 is called *mixed choice* because both input and output are allowed within the same 'choice'.

The π-calculus with separate choice π_s is the sub-calculus of π_m where summations cannot mix input and output guards. The set of processes is given by the following grammar:

$$P, Q ::= \mathbf{0} \mid \sum_{i \in I} \alpha_i^I.P_i \mid \sum_{i \in I} \alpha_i^O.P_i \mid\ !P \mid P \mid Q \mid (\nu n)P$$

$$\alpha^I ::= m(n) \qquad \alpha^O ::= \overline{m}\langle n\rangle$$

The semantics of this calculus is the same as that of π_m taking into account the syntactic restrictions. One could regard π_s as having the same expressive strength as the asynchronous π-calculus [16,3], in view of the results on encoding of separate choice [23].

Cardelli and Gordon introduced Mobile Ambients (MA) [8] in order to model new computational phenomena over wide-area networks or the Internet. Ambients represent bounded places for computation, such as concrete locations, concrete domains, abstract domains, or laptop computers. Ambients move into and out of other ambients bringing along moving code, static processes and possibly other ambients.

In this section we describe MA together with two variant calculi, Safe Ambients and the Push and Pull Ambient Calculus.

2.2.1 Mobile Ambients

The language of MA inherits a number of operators from the π-calculus. The new primitives are the *ambient* and a special form of guard that goes under the name of *capability*. We use the same set of names $\mathcal{N}$ as for π_{m} and π_{s}.

Definition 2.5 *The set of process terms of MA is given by the following syntax:*

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

where M stands for the capabilities *defined by the following grammar:*

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

An *ambient* $n[\,P\,]$ is composed of two parts: n is the name of the ambient and P is the active process inside. The square brackets around P indicate the perimeter of the ambient. If the ambient moves, everything inside moves with it. *Parallel composition*, *restriction*, *nil* and *replication* have the same meaning as in the π-calculus.

Unlike the π-calculus, communication happens without channels (anonymously). *Anonymous input*, written $(n).P$, represents a process waiting for a name to be sent. This operator binds n in P. *Anonymous output*, written $\langle n \rangle$, represents an asynchronous sending primitive. The output is not a prefix, unlike the input. We have chosen that only names can be sent. This formulation is simpler than in [8], where also capabilities can be communicated, and it is adequate for the purpose of this work.

In the process $M.P$, P is enabled only if the capability M has been consumed. Capabilities can be thought of as terms that enable the ambients to perform

some actions. An ambient gains the ability to go inside another ambient whose name is n with the in n capability. An ambient gains the ability to leave a parent ambient whose name is n with the out n capability. An ambient named n can be dissolved by the means of the open n capability.

There is a notion of *free names* ($fn(P)$), taking into account that the only binding operators are restriction and anonymous input. We write $P\{n/m\}$ to mean that each free occurrence of m is substituted by n in P. Where no confusion is possible, we will use the shorthand M instead of $M.\mathbf{0}$, and $n[\,]$ instead of $n[\,\mathbf{0}\,]$.

Computation in MA consists of entering an ambient, exiting an ambient, dissolving an ambient, and communication. Formally, steps of computation are represented by a *reduction relation* which is defined below.

Definition 2.6 *The* structural congruence *relation* $\equiv$ *is the smallest congruence over MA processes that satisfies the following equations:*

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

Definition 2.7 *The* reduction relation $\longrightarrow$ *on MA processes is the smallest relation satisfying the following set of rules:*

$$m[\,\text{in } n.P \mid Q\,] \mid n[\,R\,] \longrightarrow n[\,m[\,P \mid Q\,] \mid R\,] \quad \text{RED IN}$$
$$n[\,m[\,\text{out } n.P \mid Q\,] \mid R\,] \longrightarrow m[\,P \mid Q\,] \mid n[\,R\,] \quad \text{RED OUT}$$
$$\text{open } n.Q \mid n[\,R\,] \longrightarrow Q \mid R \qquad \text{RED OPEN}$$
$$\langle m \rangle \mid (n).P \longrightarrow P\{m/n\} \qquad \text{RED A-COMM}$$

$$\frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \text{ RED PAR} \qquad \frac{P \longrightarrow P'}{(\nu n)P \longrightarrow (\nu n)P'} \text{ RED RESTR}$$

$$\frac{P \longrightarrow P'}{n[\,P\,] \longrightarrow n[\,P'\,]} \text{ RED AMB} \quad \frac{P \equiv P' \quad P' \longrightarrow Q' \quad Q' \equiv Q}{P \longrightarrow Q} \text{ RED CONG}$$

Note that MA inherits the rules RED PAR, RED RESTR and RED CONG from the π-calculus. We use the same notation for structural congruence and reduction in all calculi we shall consider; even though they are different relations, no confusion will arise.

For MA the canonical observable is the name of an ambient at the top level. Thus, the predicate $P \downarrow n$ intuitively says that in P there is a top level process which is an ambient, whose unrestricted name is n.

Definition 2.8 *A process P exhibits a barb n, written as $P \downarrow n$, if and only if $P \equiv (\nu m_1 \ldots m_k)(n[\,P'\,] \mid P'')$, for some P', P'' and $n \notin \{m_1 \ldots m_k\}$.*

We shall be interested in various fragments of MA. We refer to MA without communication as *pure* MA, and MA without the restriction operator as *public* MA. This applies also to the other ambient calculi we shall consider.

2.2.2 Safe Ambients

Levi and Sangiorgi proposed Safe Ambients (SA) [17] as a substantial modification of MA, which retains the same computational model while improving the underpinning algebraic theory. They argued that the basic operational semantics for MA led to the phenomenon of *grave interference*, where two or more redexes of different kinds destroy each other. In order to overcome this problem, Levi and Sangiorgi added *co-capabilities* to the ambient primitives and a sophisticated type system. The type system is outside the scope of the current topic, and therefore it will not be discussed in the rest of the paper. Co-capabilities are capabilities inside an ambient, that control the influence that other ambients have upon it. Since an ambient can be entered, exited or opened, there are three co-capabilities that determine if an ambient can be entered, exited or opened. This change induces a synchronisation as well, namely in order for a reduction (other than communication) to occur, there needs to be a match between a capability and the corresponding co-capability.

Definition 2.9 *The set of process terms of SA is given by the following syntax:*

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

11

M ranges over the capabilities *defined by the following grammar:*

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

All the processes have the same informal meaning as described in the previous section. There are only three new co-capabilities: $\overline{\text{out}} \, n$ expresses that an ambient n is willing to release an internal ambient; $\overline{\text{in}} \, n$ expresses that an ambient n is willing to accept another entering ambient and $\overline{\text{open}} \, n$ expresses that an ambient n can be opened. The set of free names of P is written $fn(P)$ and is defined as for MA, taking into account the syntactic differences.

Structural congruence $\equiv$ for SA is defined exactly as for MA (Definition 2.6).

Definition 2.10 *The* reduction relation $\longrightarrow$ *on SA processes is defined as for MA (Definition 2.7), except that rules* RED IN, RED OUT *and* RED OPEN *are replaced by:*

$$m[\, \text{in } n.P_1 \mid P_2 \,] \mid n[\, \overline{\text{in}} \, n.Q_1 \mid Q_2 \,] \;\longrightarrow\; n[\, m[\, P_1 \mid P_2 \,] \mid Q_1 \mid Q_2 \,] \quad \text{RED S-IN}$$

$$n[\, \overline{\text{out}} \, n.P_1 \mid P_2 \mid m[\, \text{out } n.Q_1 \mid Q_2 \,]\,] \;\longrightarrow\; n[\, P_1 \mid P_2 \,] \mid m[\, Q_1 \mid Q_2 \,] \quad \text{RED S-OUT}$$

$$\text{open } n.P \mid n[\, \overline{\text{open}} \, n.Q_1 \mid Q_2 \,] \;\longrightarrow\; P \mid Q_1 \mid Q_2 \qquad\qquad \text{RED S-OPEN}$$

Definition 2.11 *A process P exhibits a barb n, written as $P \downarrow n$, if and only if $P \equiv (\nu m_1 \dots m_k)(n[\, \text{cap } n.P' \mid P'' \,] \mid P''')$ for some P', P'', P'' and $n \notin \{m_1 \dots m_k\}$, where $\text{cap} = \overline{\text{in}}$ or $\text{cap} = \overline{\text{open}}$.*

SA can be viewed at least as expressive as MA since there exists an obvious encoding of MA into SA. We report below the clause for the ambient. The definition for the other operators is homomorphic.

$$[\![\, n[\, P \,] \,]\!] \stackrel{def}{=} n[\, !\,\overline{\text{in}} \, n \mid !\,\overline{\text{out}} \, n \mid \overline{\text{open}} \, n \mid [\![P]\!] \,]$$

2.2.3   The Push and Pull Ambient Calculus

The third ambient calculus we shall discuss is the Push and Pull Ambient Calculus (PAC) [26]. In comparison with MA, two new capabilities are introduced: $\text{push } n$ and $\text{pull } n$ instead of $\text{in } n$ and $\text{out } n$; the rest of the syntax remains unchanged. We believe that this calculus can be useful for modelling client-server architecture as argued in [26].

Definition 2.12 *The set of process terms of PAC is given by the following syntax:*

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

M ranges over the capabilities defined by the following grammar:

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

We omit the explanation for the operators that are common to MA. The meaning of the capabilities is intuitively the following: $\mathsf{pull}\, n$ causes an ambient with name n to be pulled inside the current one, $\mathsf{push}\, n$ pushes an ambient with name n out of the current ambient, and $\mathsf{open}\, n$ behaves as in MA. The capability open is necessary for PAC, just as in MA, to allow the exchange of messages between different ambients.

Similarly to MA, PAC is also equipped with an operational semantics defined in terms of reduction semantics. We present below only the reduction rules, since structural congruence is identical to the definition for MA.

Definition 2.13 *The* reduction relation $\longrightarrow$ *on PAC processes is defined as for MA (Definition 2.7), except that rules* RED IN *and* RED OUT *are replaced by:*

$$m[\,\mathsf{pull}\, n.P \mid Q\,] \mid n[\,R\,] \;\longrightarrow\; m[\,P \mid Q \mid n[\,R\,]\,] \quad \text{RED PULL}$$

$$n[\,\mathsf{push}\, m.P \mid m[\,Q\,] \mid R\,] \;\longrightarrow\; n[\,P \mid R\,] \mid m[\,Q\,] \quad \text{RED PUSH}$$

For PAC the canonical observable is the name of an ambient at top level, exactly as for MA (Definition 2.8).

3 Leader election problems

In this section we discuss how to formalise leader election in process calculi, and in particular how to do it using reduction semantics.

In the field of distributed algorithms [18,30], the leader election problem consists of finding an algorithm such that, starting from a configuration of processes in the *same state*, any possible computation reaches a configuration where *exactly one* process is in a *leader* state and all the other processes are in *non-leader* states (i.e. they have lost the election). We may write of *problems* in the plural, since as we shall see there are parameters that can be varied.

The crucial criteria for leader election problems are the following:

Symmetry *Each process in the configuration has to have the same duties.* Leader election problems are run in order to configure a distributed system. Processes in the network are programmed identically. In symmetric configurations if one process can declare itself the winner, every other process in

the configuration can do the same. Thus, in symmetric networks, for the winner to be elected, the initial symmetry has to somehow be broken.

Distribution The computation has to be *decentralised*, in the sense that the computation has to start from any subset of processes in the network or configuration. In general, leader election algorithms are run after a reconfiguration or crash of a system, in order to select a process to start the initialisation. In this context, the configuration of processes has to be able to elect a leader without any help from outside.

Uniqueness of the leader The processes in a network reach a *terminal configuration* from *any* computation. In the terminal configuration there is *one process only* that is elected the *winner* and the other processes in the configuration have lost.

Leader election problems vary according to the following parameters:

TOPOLOGY OF THE NETWORK The network could be a *fully connected graph* or a *ring* or *tree* or any other graph or hyper-graph [1,30,18].

SIZE OF THE NETWORK The number of processes can be known or unknown before starting the election [30].

DECLARATION OF THE LEADER The leader could be announced by one process only, either the leader itself or any other process. Alternatively every process in the configuration has to be aware of the winner. The latter requirement is considered standard, although the weaker one (the former one) is also acceptable, since the winner could inform the other processes of the outcome of the election.

We do not take complexity issues into account here. As Bougé [4] points out, lower bound results can depend on whether the identifiers of the processes are integers and so on. These kinds of issues are not relevant in our setting.

3.1 Leader election problems and process calculi

The first person to exploit leader election within a process calculus with a formal semantics was Bougé [4], working in CSP [14,15]. He defined the notion of a *symmetric electoral system*, which is a symmetric network where a unique winner is elected by every computation. The most remarkable achievements are the separation results between CSP with input and output guards and CSP with input guards only, and between the latter and CSP without guards, based on the notion of *symmetric reasonable implementation*.

A further formalisation of the notion of leader election problem was made by Palamidessi [24] for the π-calculus. This work has been the major source of inspiration for the present work. While Bougé defined an encoding to be "reasonable" if it maps electoral systems to electoral systems, Palamidessi

gives specific conditions which an encoding should satisfy in order to preserve electoral systems. She proves that any symmetric network in π-calculus with separate choice admits a computation that never breaks the initial symmetry. This result is used to show that there is no encoding of the π-calculus with mixed choice into the π-calculus with separate choice. In her paper Palamidessi uses a graph framework, in the tradition of distributed algorithms [18,30,1,4], and she proves that CCS [20] does not admit a symmetric electoral system in a ring, as opposed to the π-calculus with mixed choice.

Using an approach similar to Palamidessi's, Ene and Muntean [11] show that the π-calculus with broadcasting primitives cannot be encoded in the standard π-calculus.

3.2 A reduction semantics framework for leader election problems

Our predecessors [24,11] used labelled transition systems when defining the computations performed by electoral systems. We prefer to use reduction semantics, for the reasons stated in the Introduction. In this section we shall define networks and electoral systems for calculi equipped with reduction semantics. The definitions below apply equally well to ambient calculi and to the π-calculus, and to any other calculus that uses the reduction semantics framework. We shall compare MA, SA, and PAC against the π-calculus, but also our framework could be used for comparing other calculi, such as the Seal calculus [9] and $D\pi$ [13].

We assume that the set of names $\mathcal{N}$ includes a set of *observables*:

$$\mathsf{Obs} = \{\omega_i : i \in \mathbb{N}\}$$

such that for all i, j, $\omega_i \neq \omega_j$ if $i \neq j$. The observables will be used by networks to communicate with the outside world, and *can never be restricted* (i.e. they never occur under the scope of restriction).

It is convenient to let $\mathsf{Obs}_k = \{\omega_0, \omega_1, \ldots, \omega_{k-1}\}$. This will be the set of names indicating possible winners in a network of size k.

We shall use *natural numbers* as indices of processes in a network, for instance P_i where P is a process and i is a natural number.

Networks are just collections of processes running in parallel, possibly equipped with some globally restricted names.

Definition 3.1 (Network) *A* network N *of size k is a process in the form*

$$(\nu m_0, \ldots, m_{l-1})(P_0 \mid P_1 \mid P_2 \mid \cdots \mid P_{k-1})$$

We will use the notation $[P_0 \mid P_1 \mid \cdots \mid P_{k-1}]$ for representing the network above when the globally bound names $m_0, \ldots, m_{l-1}$ are not relevant.

When dealing with indices for a network of size k we shall always use arithmetic modulo k when writing expressions such as P_{i+1}.

Notice that the size of a network is really a matter of how it is presented (i.e. divided up) rather than the process itself.[2] For instance the process $n[\ ]$ is clearly a network of size one. But it is structurally congruent to $n[\ ] \mid \mathbf{0}$, which can be seen as a network of size two. We are interested in symmetric networks, however, and size cannot then be varied arbitrarily as in the example just given.

Definition 3.2 (Permutation on names) *(1) A permutation on names is a bijection $\sigma : \mathcal{N} \to \mathcal{N}$ which respects observables, i.e. $m \in \mathsf{Obs}$ if and only if $\sigma(m) \in \mathsf{Obs}$.*
(2) A permutation σ induces a bijection on the natural numbers $\hat{\sigma} : \mathbb{N} \to \mathbb{N}$ as follows: if $\sigma(\omega_i) = \omega_j$ then $\hat{\sigma}(i) = j$.

Note that for all $i \in \mathbb{N}$, $\sigma(\omega_i) = \omega_{\hat{\sigma}(i)}$.

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. In other words, the definition of $\sigma(P)$, where σ is a permutation, is by recursion on the syntax of the language, avoiding that names are captured by binding operators.

Definition 3.3 (Automorphism in a network) *Let N be a network of size k:*
$$\mathsf{N} \stackrel{def}{=} (\nu m_0, \ldots, m_{l-1})(P_0 \mid P_1 \mid \cdots \mid P_{k-1}).$$
A network automorphism σ is a permutation such that: $\hat{\sigma}$ restricted to the finite subset of natural numbers $\{0, 1, \ldots, k-1\}$ is a bijection and σ preserves the distinction between free and bound names, i.e. $m \in \{m_0, \ldots, m_{l-1}\}$ iff $\sigma(m) \in \{m_0, \ldots, m_{l-1}\}$.

Definition 3.4 (Orbit) *Let N be a network of size k and σ an automorphism on it. For any $i \in \{0, \ldots, k-1\}$ the orbit $\mathcal{O}_{\hat{\sigma}}(i)$ generated by σ is defined as follows:*
$$\mathcal{O}_{\hat{\sigma}}(i) \stackrel{def}{=} \{i, \hat{\sigma}(i), \hat{\sigma}^2(i), \ldots, \hat{\sigma}^{h-1}(i)\}$$
where $\hat{\sigma}^j$ represents the composition of $\hat{\sigma}$ with itself j times. and h is least number such that $\sigma^h(i) = i$.

---

[2] We make the distinction between network presentations and their interpretation as processes more formal in [28, Section 3.1].

Intuitively a network N is symmetric with respect to an automorphism σ if and only if, for each i, the renaming of the process associated to i is the same up to alpha-conversion as the process associated to the permuted index $\hat{\sigma}(i)$. This is a way of specifying in this setting that each process has the same duties.

In the first part of the following definition we closely follow Palamidessi. In the second part however, we restrict the notion of symmetry to networks where the automorphism has one orbit only.

Definition 3.5 (Symmetric Network)

(1) *Let* $\mathsf{N} = [P_0 \mid \cdots \mid P_{k-1}]$ *be a network and* σ *an automorphism on it. We say that* N *is* symmetric with respect to σ *iff for each* $i \in \{0, \ldots, k-1\}$, $P_{\hat{\sigma}(i)} = \sigma(P_i)$ *holds.*

(2) *A network* $\mathsf{N} = [P_0 \mid \cdots \mid P_{k-1}]$ *is called* symmetric *if it is symmetric with respect to some automorphism with a single orbit.*

Our definitions of network automorphism and symmetric network differ from those of Palamidessi, since she takes the network topology into account, and associates a hypergraph with a network. Automorphisms are defined with respect to this hypergraph, and a network is symmetric if it is symmetric with respect to *every* automorphism.

Definition 3.6 *Consider a network* $\mathsf{N} = [P_0 \mid P_1 \mid \cdots P_{k-1}]$. *A computation* $\mathcal{C}$ *is a (finite or infinite) sequence:*

$$\mathsf{N} = \mathsf{N}^0 \longrightarrow \mathsf{N}^1 \longrightarrow \mathsf{N}^2 \longrightarrow \cdots \longrightarrow \mathsf{N}^j \longrightarrow \cdots .$$

- *A computation* $\mathcal{C}$ *is* maximal *if it is infinite, or else it is of the form* $\mathsf{N} \longrightarrow \mathsf{N}^1 \longrightarrow \mathsf{N}^2 \cdots \longrightarrow \mathsf{N}^h$ *where* $\mathsf{N}^h \nrightarrow$.
- *The* composition $\mathcal{C} \cdot \mathcal{C}'$ *of computations* $\mathcal{C}$ *and* $\mathcal{C}'$ *is defined in an obvious manner if* $\mathcal{C}$ *is finite and the last state of the computation* $\mathcal{C}$ *coincides with the initial state of* $\mathcal{C}'$.
- *We say that* $\mathcal{C}'$ *extends* $\mathcal{C}$, *written* $\mathcal{C} \prec \mathcal{C}'$, *if there exists a computation* $\mathcal{C}''$ *such that* $\mathcal{C}' = \mathcal{C} \cdot \mathcal{C}''$.

Our notion of computation is defined using the reduction relation only. This is a substantial difference from previous authors [24,11], where computation is defined as a sequence of transitions derived from the labelled transition system.

In our definition of network computation, we assume that a network reduces to a network. This might be seen as restrictive; however since any process can be seen as a network, the above definition is general enough. We do not require that a network preserves its own size. In fact in general this might not

even be true, as will become clear when presenting electoral systems in the ambient calculus.

Definition 3.7 *Let $\mathcal{C}$ be a computation* $\mathsf{N} \longrightarrow \cdots \longrightarrow \mathsf{N}^h \longrightarrow \cdots$. *We define the* observables of $\mathcal{C}$ *as*

$$\mathsf{Obs}(\mathcal{C}) = \{\omega \in \mathsf{Obs} : \exists h \ \mathsf{N}^h \downarrow \omega\}.$$

Computations have some rather nice properties on the observables: observation on computation is compositional and monotonic.

Lemma 3.8 *(1) Let $\mathcal{C} \stackrel{def}{=} \mathcal{C}' \cdot \mathcal{C}''$ be a computation. Then $\mathsf{Obs}(\mathcal{C}) = \mathsf{Obs}(\mathcal{C}') \cup \mathsf{Obs}(\mathcal{C}'')$.*

(2) Let $\mathcal{C}$ and $\mathcal{C}'$ be two computations such that $\mathcal{C} \prec \mathcal{C}'$. Then $\mathsf{Obs}(\mathcal{C}) \subseteq \mathsf{Obs}(\mathcal{C}')$.

PROOF. Trivial. □

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

Definition 3.9 (Electoral system) *A network $\mathsf{N} = [P_0 \mid \cdots \mid P_{k-1}]$ is an electoral system if for every maximal computation $\mathcal{C}$ of N there exists an $i < k$ such that $\mathsf{Obs}(\mathcal{C}) = \{\omega_i\}$. An electoral system is said to be* symmetric *if the network is symmetric.*

Thus each maximal computation gives exactly one winner. It does not matter which process in the original network displays the observable barb; indeed, in ambient calculi this is not even necessarily meaningful, as processes can intermingle using movement capabilities.

For Palamidessi the requirement for an electoral system is that *every* process in the electoral system can execute a special action $\overline{out}\langle i \rangle$. In other words everyone is aware of the leader. As she states, her results would hold under the alternative requirement that *exactly one* process announces the winner. Our notion is weaker, in that we merely require that *at least one* process announces the winner, and it is left open how many processes make the announcement. It has been proved in [31] that this notion of electoral system gives the same separation results for the π-calculus as those obtained by Palamidessi.

4 Calculi with symmetric electoral systems

In this section we present solutions to the leader election problem in symmetric networks of any finite size in fragments of the π-calculus, MA, PAC and SA defined as follows.

Definition 4.1 *(1) Let $\pi_{\mathsf{m}}^{-\nu}$ be π_{m} but without restriction.*
(2) Let MA^{io} be pure public MA without the open *capability.*
(3) Let PAC^{pp} be pure public PAC without the open *capability.*
(4) Let SA^{io} be pure public SA without the open *capability.*
(5) Let SA^{iop} be pure public SA without the out *capability.*

4.1 The π-calculus with mixed choice

The π-calculus with mixed choice can elect a leader in a symmetric network according to Palamidessi's criteria [24]. It is not difficult to see that π_{m} admits a symmetric electoral system also according to our new and weaker criteria. In the reduction semantics frameworks, we are able to improve slightly Palamidessi's result; in fact we can show that there exists an electoral system in π_{m} but without restriction. In the following proposition we show that there exists a symmetric electoral system of size 2 in $\pi_{\mathsf{m}}^{-\nu}$, which is sufficient for restating Palamidessi's separation result between π_{m} and π_{s}.

Proposition 4.2 *In $\pi_{\mathsf{m}}^{-\nu}$ there exists a symmetric electoral system of size 2.*

PROOF. We define a network N as follows:

$$P_0 \stackrel{def}{=} x_0(y) + \overline{x_1}\langle z\rangle.\overline{\omega_0}\langle z\rangle \qquad P_1 \stackrel{def}{=} x_1(y) + \overline{x_0}\langle z\rangle.\overline{\omega_1}\langle z\rangle$$
$$\mathsf{N} \stackrel{def}{=} P_0 \mid P_1$$

The network is symmetric with respect to a single-orbit automorphism σ defined as follows:

$$\sigma(x_0) = x_1 \quad \sigma(x_1) = x_0 \quad \sigma(\omega_0) = \omega_1 \quad \sigma(\omega_1) = \omega_0$$

with σ the identity on all other names. There are only two possible computations. We present one in detail; the other one is identical up to the renaming of σ.

$$\mathcal{C} : \mathsf{N} \longrightarrow \overline{\omega_1}\langle z\rangle$$

Clearly $\mathsf{Obs}(\mathcal{C}) = \{\omega_1\}$. $\square$

In the previous proposition there are two important features to notice. The first one is that the link-passing capability of the π-calculus plays no rôle; it is the mixed choice which is important. In other words, the previous electoral system could have been written in CCS without link passing.

The second feature is that the process above would not be an electoral system in Palamidessi's framework. In fact, not every computation leads to the election of a leader, if computation is defined in terms of a labelled transition system. Thus restriction is necessary for Palamidessi.

Remark 4.3 *Proposition 4.2 can be generalised to networks of any finite size. The construction can be derived from Palamidessi's algorithm for a fully-connected network of size four in the proof of her Proposition 5.1.*

Remark 4.4 *We have used output barbs in our definition of electoral system. If we instead used input barbs to report the winner, then it is straightforward to modify the electoral system of Proposition 4.2 to fit the revised definition.*

4.2   Ambient calculi

We now turn to showing that the existence of symmetric electoral systems in MA^{io}. First we show that there exists an electoral system of size 2. Then we present a more general solution, for a network of any size. The solution presented here is different from the one in [27]. This new solution is slightly more complex, but has the advantage of being easily adapted to PAC^{pp}.

Proposition 4.5 *In MA^{io} there exists a symmetric electoral system of size* 2.

PROOF. Let

$$P_0 \stackrel{def}{=} n_0[\, \text{in } n_1.\omega_0[\, \text{out } n_0.\text{out } n_1 \,]\,]$$
$$P_1 \stackrel{def}{=} n_1[\, \text{in } n_0.\omega_1[\, \text{out } n_1.\text{out } n_0 \,]\,]$$
$$\mathsf{N} \stackrel{def}{=} P_0 \mid P_1 \ .$$

The network is symmetric with respect to a single-orbit automorphism σ defined as follows:

$$\sigma(n_0) = n_1 \quad \sigma(n_1) = n_0 \quad \sigma(\omega_0) = \omega_1 \quad \sigma(\omega_1) = \omega_0$$

There are only two possible computations. We shall present the first one in detail:

$$\mathcal{C} : n_0[\,\text{in } n_1.\omega_0[\,\text{out } n_0.\text{out } n_1\,]\,] \mid n_1[\,\text{in } n_0.\omega_1[\,\text{out } n_1.\text{out } n_0\,]\,] \longrightarrow$$

$$n_1[\,n_0[\,\omega_0[\,\text{out } n_0.\text{out } n_1\,]\,] \mid \text{in } n_0.\omega_1[\,\text{out } n_1.\text{out } n_0\,]\,] \longrightarrow$$

$$n_1[\,\omega_0[\,\text{out } n_1\,] \mid n_0[\,]\mid \text{in } n_0.\omega_1[\,\text{out } n_1.\text{out } n_0\,]\,] \longrightarrow$$

$$\omega_0[\,]\mid n_1[\,n_0[\,]\mid \text{in } n_0.\omega_1[\,\text{out } n_1.\text{out } n_0\,]\,]$$

Thus we conclude $\mathsf{Obs}(\mathcal{C}) = \{\omega_0\}$. The other computation is identical up to renaming via σ. $\square$

Theorem 4.6 *In MA^{io}, for any $k \geq 1$ there exists a symmetric electoral system of size k.*

PROOF. (Sketch) Let $k \geq 1$. The electoral system is defined by $\mathsf{N} \overset{def}{=} \prod_{i<k} P_i$ where

$$P_i \overset{def}{=} n_i[\,\textstyle\prod_{j\neq i} \text{in } n_j.lose_i[\,Outn\,]\,] \mid c_i[\,C_{i,i+1}\,]$$

$$Outn \overset{def}{=} \textstyle\prod_{j<k} !\,\text{out } n_j$$

$$C_{i,i} \overset{def}{=} \omega_i[\,\text{out } c_i\,]$$

$$C_{i,j} \overset{def}{=} \text{in } lose_j.C'_{i,j} \qquad (j \neq i)$$

$$C'_{i,j} \overset{def}{=} \text{out } lose_j.C_{i,j+1} \qquad (j \neq i)$$

(We use arithmetic modulo k for the indices.) The idea is that process j loses to process i if ambient n_j enters ambient n_i. When this happens, the ambient $lose_j$ is unleashed, and makes its way up to the top level. When only one n_i ambient is left at the top level then process i has won. It detects that it has won using the ambient c_i, which announces the winner after checking for the presence of $lose_j$ ambients at the top level (for all $j \neq i$). Note that there may well be multiple $lose_j$ ambients unleashed as ambient n_j may continue to enter other $n_{j'}$ ambients even after it has lost.

21

The full and detailed proof can be found in Appendix A; we present here for readability the case of the network of size 3.

$$\mathsf{N} \stackrel{def}{=} P_0 \mid P_1 \mid P_2$$

$$P_0 \stackrel{def}{=} n_0[\,\mathsf{in}\,n_1.lose_0[\,!\mathsf{out}\,n_0\,|!\mathsf{out}\,n_1\,|!\mathsf{out}\,n_2\,]\,|$$
$$\mathsf{in}\,n_2.lose_0[\,!\mathsf{out}\,n_0\,|!\mathsf{out}\,n_1\,|!\mathsf{out}\,n_2\,]\,]\,|$$
$$c_0[\,\mathsf{in}\,lose_1.\mathsf{out}\,lose_1.\mathsf{in}\,lose_2.\mathsf{out}\,lose_2.\omega_0[\,\mathsf{out}\,n_0\,]\,]$$

$$P_1 \stackrel{def}{=} n_1[\,\mathsf{in}\,n_2.lose_1[\,!\mathsf{out}\,n_1\,|!\mathsf{out}\,n_2\,|!\mathsf{out}\,n_0\,]\,|$$
$$\mathsf{in}\,n_0.lose_1[\,!\mathsf{out}\,n_1\,|!\mathsf{out}\,n_2\,|!\mathsf{out}\,n_0\,]\,]\,|$$
$$c_1[\,\mathsf{in}\,lose_2.\mathsf{out}\,lose_2.\mathsf{in}\,lose_0.\mathsf{out}\,lose_0.\omega_1[\,\mathsf{out}\,n_1\,]\,]$$

$$P_2 \stackrel{def}{=} n_2[\,\mathsf{in}\,n_0.lose_2[\,!\mathsf{out}\,n_2\,|!\mathsf{out}\,n_0\,|!\mathsf{out}\,n_1\,]\,|$$
$$\mathsf{in}\,n_1.lose_2[\,!\mathsf{out}\,n_2\,|!\mathsf{out}\,n_0\,|!\mathsf{out}\,n_1\,]\,]\,|$$
$$c_2[\,\mathsf{in}\,lose_0.\mathsf{out}\,lose_0.\mathsf{in}\,lose_1.\mathsf{out}\,lose_1.\omega_2[\,\mathsf{out}\,n_2\,]\,]$$

Note that we used a different construction in the proof of Theorem 4.6 in an earlier version of this paper [27]. The construction we use here has the advantage that it can be easily adapted to show the corresponding result for PAC$^{\mathsf{pp}}$:

Theorem 4.7 *In PAC$^{\mathsf{pp}}$, for any $k \geq 1$ there exists a symmetric electoral system of size k.*

PROOF. (Sketch) We dualise the construction given in the proof of Theorem 4.6, essentially replacing in by pull and out by push. The electoral system is defined for $k \geq 2$ by

$$\mathsf{N} \stackrel{def}{=} \prod_{i<k} P_i$$

where

$$P_i \stackrel{def}{=} n_i[\,\textstyle\prod_{j\neq i}(\mathsf{pull}\,n_j.lose_j[\,]\mid\mathsf{push}\,lose_j)\,]\mid c_i[\,C_{i,i+1}\,]$$
$$C_{i,j} \stackrel{def}{=} \mathsf{pull}\,lose_j.\mathsf{push}\,lose_j \qquad (j \neq i)$$
$$C_{i,i} \stackrel{def}{=} \omega_i[\,]\mid\mathsf{push}\,\omega_i$$

We omit the proof that this is indeed an electoral system, which is similar to the proof of Theorem 4.6. □

We now consider electoral systems in fragments of SA. We can take the symmetric electoral system in the proof of Theorem 4.6 and adapt it for SA^{io} using the standard encoding (Section 2.2.2), with the one change that we omit the $\overline{\text{open}}\, n$ from the encoding of $n[P]$. Hence:

Theorem 4.8 *In SA^{io}, for any $k \geq 1$ there exists a symmetric electoral system of size k.* $\square$

In constructing electoral systems in MA^{io}, we use the in capability to break symmetry and the out to report the winner at the top level. An interesting feature of SA is that we can also construct electoral systems using just the in and the open capabilities, with the open enabling the reporting of the winner.

Proposition 4.9 *In SA^{iop} there exists a symmetric electoral system of size 2.*

PROOF. The electoral system is the following:

$$P_0 \stackrel{def}{=} \text{open}\, n_0 \mid n_0[\, \text{in}\, n_1 \mid \overline{\text{in}}\, n_0.\overline{\text{open}}\, n_0.\omega_0[\, \overline{\text{open}}\, \omega_0\,]\,]$$

$$P_1 \stackrel{def}{=} \text{open}\, n_1 \mid n_1[\, \text{in}\, n_0 \mid \overline{\text{in}}\, n_1.\overline{\text{open}}\, n_1.\omega_1[\, \overline{\text{open}}\, \omega_1\,]\,]$$

$$\mathsf{N} \stackrel{def}{=} P_0 \mid P_1$$

The first process to perform an in reduction loses. $\square$

We can generalise to systems of arbitrary size:

Theorem 4.10 *In SA^{iop}, for any $k \geq 1$ there exists a symmetric electoral system of size k.*

PROOF. See Appendix B. $\square$

Remark 4.11 *Recall that a barb in SA is a top-level unrestricted ambient containing either an $\overline{\text{open}}$ or an $\overline{\text{in}}$ capability. Both Proposition 4.9 and Theorem 4.10 still hold if we vary the definition of barb by just using $\overline{\text{open}}$ (in which case the proofs are unchanged), or by just using $\overline{\text{in}}$ (in which case we simply replace $\omega_i[\,\overline{\text{open}}\, \omega_i\,]$ by $\omega_i[\,\overline{\text{in}}\, \omega_i\,]$ in our electoral systems).*

5 Calculi without symmetric electoral systems

In this section we are going to show that there are calculi that do not admit a symmetric electoral system. First of all, we shall reestablish Palamidessi's

result on the π-calculus with separate choice, which states that π_s does not admit a symmetric electoral system [24]. We then prove that MA and SA without in and PAC without pull do not admit a symmetric electoral system either.

5.1 The π-calculus with separate choice

The following theorem claims that there does not exist a symmetric electoral system of any finite size in π_s. The statement is identical to Palamidessi's, although the proof is different in style. The proof is based on the idea of showing that there exists a maximal computation which fails to elect the leader. This means that, either more than one leader is elected or else the computation has no leader.

Theorem 5.1 *[24] Let* $N = [P_0 \mid \cdots \mid P_{k-1}]$ *with* $k \geq 2$ *be a symmetric network in* π_s. *Then* N *cannot be an electoral system.*

The full proof is very similar to the proof for MA without the in capability that follows. The details of this proof can be found in [31]. It must be remarked that our proof is very constructive in the logical sense. The computation that fails to elect a leader is constructed step by step in the proof, so that it is quite clear how the initial symmetry is preserved during computation.

5.2 Mobile Ambients without the in capability

We have shown earlier that MA^{io} can solve the leader election problem in a symmetric network. This kind of problem can be solved in π_m, but not in π_s. It is clear that the mixed choice operator is the key for the expressiveness result. It is an interesting problem as to which operator makes a difference in expressiveness for ambient calculi. This question is also interesting for the following reason. It might be argued that leader election problems are not interesting in the ambient setting, because of the inherent tree structure of processes. The next theorem (Theorem 5.8) will show that this is not completely true. The tree structure of ambients is not indeed the key to expressiveness. In fact, we could keep the tree structure and remove one capability, in n, in which case the leader election problem cannot be solved. This means, in simple words, that for breaking symmetry the in capability is crucial.

Definition 5.2 *Let* MA^{-in} *denote MA without the* in *capability.*

Before proving Theorem 5.8 we need to show some lemmas.

Lemma 5.3 *Let P be a process in $MA^{-\text{in}}$ and σ a permutation. Then $P \downarrow n$ if and only if $\sigma(P) \downarrow \sigma(n)$.*

PROOF. By definition of barbs, $P \downarrow n$ if and only if $P \equiv (\nu m_1 \ldots m_l)(n[\,P'\,] \mid P'')$ with $n \notin \{m_1 \ldots m_l\}$ if and only if $\sigma(P) \equiv (\nu\sigma(m_1) \ldots \sigma(m_l))(\sigma(n)[\,\sigma(P')\,] \mid \sigma(P''))$ by definition of permutation, if and only if $\sigma(P) \downarrow \sigma(n)$ by definition of barbs. $\square$

Lemma 5.4 *Let P be a process in $MA^{-\text{in}}$ and σ a substitution. If $P \longrightarrow P'$, then $\sigma(P) \longrightarrow \sigma(P')$.*

PROOF. By induction on $\longrightarrow$. $\square$

The following lemma says that initially a symmetric network cannot elect a leader. If one process declares a winner, everyone else is declared a winner as well.

Lemma 5.5 *Let $\mathsf{N} = [P_0 \mid \cdots \mid P_{k-1}]$ be a symmetric network in $MA^{-\text{in}}$. If for some i such that $0 \le i \le k-1$ we have $\mathsf{N} \downarrow \omega_i$, then for all l with $0 \le l \le k-1$ we have $\mathsf{N} \downarrow \omega_l$.*

PROOF. Assume that N is symmetric with respect to σ with one orbit only. First of all it is important to observe that for such a σ for all $i \in \{0, 1, \ldots, k-1\}$

$$\mathcal{O}_{\hat{\sigma}}(i) = \{i, \hat{\sigma}^1(i) \ldots \hat{\sigma}^{k-1}(i)\} = \{0, 1, \ldots, k-1\}.$$

If $\mathsf{N} \downarrow \omega_i$ then there exists an r ($0 \le r \le k-1$) such that $P_r \downarrow \omega_i$. By symmetry, for all $h < k$, $P_{\hat{\sigma}^h(r)} = \sigma^h(P_r)$ holds. Hence we have $P_{\hat{\sigma}^h(r)} \downarrow \omega_{\hat{\sigma}^h(i)}$ by Lemma 5.3. By assumption, σ has one orbit. Hence for all $j \in \{0, 1, \ldots, k-1\}$ it holds that

$$[P_r \mid P_{\hat{\sigma}(r)} \mid \cdots \mid P_{\hat{\sigma}^{k-1}(r)}] \downarrow \omega_j.$$

$\square$

The next lemma is crucial; it shows that for a symmetric network there exists a computation that never breaks the initial symmetry (note that in the presence of the in capability it is certainly possible to break symmetry, as witnessed by the network used in the proof of Proposition 4.5).

Lemma 5.6 *Let $\mathsf{N} = [P_0 \mid \cdots \mid P_{k-1}]$ be a symmetric network in $MA^{-\text{in}}$. Assume that $\mathsf{N} \longrightarrow \mathsf{N}^1$. Then there exists a computation $\mathcal{C} : \mathsf{N}^1 \longrightarrow \cdots \longrightarrow \mathsf{N}^k$ such that:*

(1) N^k *is symmetric;*

(2) *if for some i we have $\mathsf{N}^1 \downarrow \omega_i$ and $\mathsf{N} \not\Downarrow \omega_i$, then for all h such that $0 \le h \le k-1$ we have $\mathsf{N}^k \downarrow \omega_h$;*

(3) *if for all j such that $0 \le j \le k-1$ we have $\mathsf{N}^1 \not\Downarrow \omega_j$, then for all t, i such that $1 \le t \le k$ and $0 \le i \le k-1$ we have $\mathsf{N}^t \not\Downarrow \omega_i$.*

PROOF. Assume that N is symmetric with respect to an automorphism σ with one orbit only and that $\mathsf{N} \longrightarrow \mathsf{N}^1$. There are two cases to consider:

Reduction derived from one process only. $\mathsf{N} \longrightarrow \mathsf{N}^1$ is the result of the computation of one process in the network. Assume P_r in N has reduced as follows: $P_r \longrightarrow P_r^\dagger$. Before proceeding with the proof, we remind the reader that by symmetry the following statements hold.

$$[P_0 \mid P_1 \mid \cdots \mid P_{k-1}] \equiv [P_r \mid P_{\hat{\sigma}(r)} \cdots \mid P_{\hat{\sigma}^{k-1}(r)}]$$

$$P_{\hat{\sigma}(r)} = \sigma(P_r)$$
$$P_{\hat{\sigma}^2(r)} = \sigma^2(P_r) = \sigma(P_{\hat{\sigma}(r)})$$
$$P_{\hat{\sigma}^3(r)} = \sigma^3(P_r) = \sigma(P_{\hat{\sigma}^2(r)})$$
$$\vdots$$
$$P_{\hat{\sigma}^k(r)} = \sigma^k(P_r) = \sigma(P_{\hat{\sigma}^{k-1}(r)})$$

If $P_r \longrightarrow P_r^\dagger$ and by symmetry $\sigma(P_r) = P_{\hat{\sigma}(r)}$ then by Lemma 5.4 $P_{\hat{\sigma}(r)} \longrightarrow P_{\hat{\sigma}(r)}^\dagger$ where we let $P_{\hat{\sigma}(r)}^\dagger = \sigma(P_r^\dagger)$. By repeating the previous reasoning for each process in the network, we conclude there are the following $k-1$ reductions.

$$P_{\hat{\sigma}(r)} \longrightarrow P_{\hat{\sigma}(r)}^\dagger = \sigma(P_r^\dagger)$$
$$P_{\hat{\sigma}^2(r)} \longrightarrow P_{\hat{\sigma}^2(r)}^\dagger = \sigma^2(P_r^\dagger)$$
$$\vdots$$
$$P_{\hat{\sigma}^{k-1}(r)} \longrightarrow P_{\hat{\sigma}^{k-1}(r)}^\dagger = \sigma^{k-1}(P_r^\dagger)$$

Therefore starting from the network $\mathsf{N}^1 = [P_r^\dagger \mid P_{\hat{\sigma}(r)} \mid \cdots \mid P_{\hat{\sigma}^{k-1}(r)}]$ there exists the following computation.

$$\mathsf{N}^1 : [P_r^\dagger \mid P_{\hat{\sigma}(r)} \mid P_{\hat{\sigma}^2(r)} \mid \cdots \mid P_{\hat{\sigma}^{k-1}(r)}] \longrightarrow$$
$$\mathsf{N}^2 : [P_r^\dagger \mid P_{\hat{\sigma}(r)}^\dagger \mid P_{\hat{\sigma}^2(r)} \mid \cdots \mid P_{\hat{\sigma}^{k-1}(r)}] \longrightarrow$$
$$\mathsf{N}^3 : [P_r^\dagger \mid P_{\hat{\sigma}(r)}^\dagger \mid P_{\hat{\sigma}^2(r)}^\dagger \mid \cdots \mid P_{\hat{\sigma}^{k-1}(r)}] \longrightarrow$$
$$\vdots$$
$$\mathsf{N}^k : [P_r^\dagger \mid P_{\hat{\sigma}(r)}^\dagger \mid P_{\hat{\sigma}^2(r)}^\dagger \mid \cdots \mid P_{\hat{\sigma}^{k-1}(r)}^\dagger]$$

This defines $\mathcal{C} : \mathsf{N}^1 \longrightarrow \cdots \longrightarrow \mathsf{N}^k$.

(1) By symmetry we have:

$$[P_r^\dagger \mid P_{\hat{\sigma}(r)}^\dagger \mid P_{\hat{\sigma}^2(r)}^\dagger \mid \cdots \mid P_{\hat{\sigma}^{k-1}(r)}^\dagger] \equiv [P_0^\dagger \mid P_1^\dagger \mid \cdots \mid P_{k-1}^\dagger]$$

from which we conclude that N^k is symmetric with respect to σ. Since σ has not changed, it has one orbit only.

(2) If for some h such that $0 \le h \le k-1$ we have $\mathsf{N} \not\Downarrow \omega_h$ and $\mathsf{N}^1 \downarrow \omega_h$, it is the case that ω_h has appeared during the first step of computation as follows.

$$P_r \longrightarrow P_r^\dagger \downarrow \omega_h$$

By Lemma 5.3 the remaining processes will exhibit the other barbs as follows:

$$P_{\hat{\sigma}(r)} \quad \longrightarrow \quad \sigma(P_r^\dagger) \downarrow \omega_{\hat{\sigma}(h)}$$
$$P_{\hat{\sigma}^2(r)} \quad \longrightarrow \quad \sigma^2(P_r^\dagger) \downarrow \omega_{\hat{\sigma}^2(h)}$$
$$\vdots$$
$$P_{\hat{\sigma}^{k-1}(r)} \quad \longrightarrow \quad \sigma^{k-1}(P_r^\dagger) \downarrow \omega_{\hat{\sigma}^{k-1}(h)}.$$

Hence, since σ has one orbit only, for all j such that $0 \le j \le k-1$ we have $\mathsf{N}^k \downarrow \omega_j$.

(3) If for all j such that $0 \le j \le k-1$ we have $\mathsf{N}^1 \not\Downarrow \omega_j$ then $P_r^\dagger \not\Downarrow \omega_j$ for all j. Now assume for a contradiction that for some h and i $(0 \le i, h \le k-1)$ $\mathsf{N}^{i+1} \downarrow \omega_{\hat{\sigma}^i(h)}$ and $\mathsf{N}^i \not\Downarrow \omega_{\hat{\sigma}^i(h)}$. Then $P_{\hat{\sigma}^i(r)}^\dagger \downarrow \omega_{\hat{\sigma}^i(h)}$ and by Lemma 5.3 $P_r^\dagger \downarrow \omega_h$, which contradicts our previous assumption.

Reduction derived from the interaction of two processes. We assume, without loss of generality and to the mere end of simplifying the notation, that P_0 and P_d are the two processes involved in the reduction. Moreover, in order to simplify the notation, we assume that $\hat{\sigma}(i) = i+1$ for each $i < k$.

By the operational semantics of this limited calculus, there are two possible ways in which the reduction can occur, namely by the rules RED OPEN and RED A-COMM (note that the rule RED OUT just involves a single process, and has therefore already been covered in the previous case).

(1) P_0 and P_d reduce by the use of the rule RED OPEN. We assume, without loss of generality that

$$P_0 = (\nu p_1^0 \ldots p_s^0)(\mathsf{open}\, n.S_0 \mid P_0')$$
$$P_d = (\nu p_1^d \ldots p_s^d)(n[\, Q_d\,] \mid P_d')$$

with $n \notin \{p_1^0 \ldots p_s^0\} \cup \{p_1^d \ldots p_s^d\}$ (otherwise the two processes would not be able to reduce).

Moreover we will silently use structural congruence to move processes to adjacent positions in order to perform a reduction, and to move them back again to their original positions.

By symmetry, we conclude the following statements starting from P_0.

$$P_{\hat{\sigma}(0)} = (\nu p_1^1 \dots p_s^1)(\text{open } \sigma(n).\sigma(S_0) \mid \sigma(P_0'))$$
$$P_{\hat{\sigma}^2(0)} = (\nu p_1^2 \dots p_s^2)(\text{open } \sigma^2(n).\sigma^2(S_0) \mid \sigma^2(P_0'))$$
$$\vdots$$
$$P_{\hat{\sigma}^{k-1}(0)} = (\nu p_1^{k-1} \dots p_s^{k-1})(\text{open } \sigma^{k-1}(n).\sigma^{k-1}(S_0) \mid \sigma^{k-1}(P_0'))$$

Similar conclusions can be drawn starting from P_d.

$$P_{\hat{\sigma}(d)} = (\nu p_1^{d+1} \dots p_s^{d+1})(\sigma(n)[\,\sigma(Q_d)\,] \mid \sigma(P_d'))$$
$$P_{\hat{\sigma}^2(d)} = (\nu p_1^{d+2} \dots p_s^{d+2})(\sigma^2(n)[\,\sigma^2(Q_d)\,] \mid \sigma^2(P_d'))$$
$$\vdots$$
$$P_{\hat{\sigma}^{k-1}(d)} = (\nu p_1^{d-1} \dots p_s^{d-1})(\sigma^{k-1}(n)[\,\sigma^{k-1}(Q_d)\,] \mid \sigma^{k-1}(P_d'))$$

We see that

$$P_0 = (\nu p_1^0 \dots p_s^0)(\text{open } n.S_0 \mid \sigma^{-d}(n)[\,Q_0\,] \mid R_0)$$
$$P_d = (\nu p_1^d \dots p_s^d)(\text{open } \sigma^d(n).S_d \mid n[\,Q_d\,] \mid R_d)$$

and in general, for all h such that $0 \le h \le k-1$ we have

$$P_h = (\nu p_1^h \dots p_s^h)(\text{open } \sigma^h(n).S_h \mid \sigma^{h-d}(n)[\,Q_h\,] \mid R_h)$$

where for all $i < k$ we have $\sigma(S_i) = S_{i+1}$, $\sigma(Q_i) = Q_{i+1}$ and $\sigma(R_i) = R_{i+1}$.

Thus each P_h contains both an ambient, which we may regard as a "positive" charge, and an open capability, which we may regard as a "negative" charge, giving an overall neutral process. Now if $P_0 \mid P_d$ performs an open reduction, we may denote the residue by $P_0^+ \mid P_d^-$, where the "charged" residual processes have the syntactical form:

$$P_0^+ = (\nu p_1^0 \dots p_s^0)(S_0 \mid \sigma^{-d}(n)[\,Q_0\,] \mid R_0)$$
$$P_d^- = (\nu p_1^d \dots p_s^d)(\text{open } \sigma^d(n).S_d \mid Q_d \mid R_d)$$

In general the residues of the processes are the following:

$$P_h^+ = (\nu p_1^h \dots p_s^h)(S_h \mid \sigma^{h-d}(n)[\,Q_h\,] \mid R_h)$$
$$P_h^+ = (\nu p_1^h \dots p_s^h)(\text{open } \sigma^h(n).S_h \mid Q_h \mid R_h)$$
$$P_h^{+-} = P_d^{-+} = (\nu p_1^h \dots p_s^h)(S_h \mid Q_h \mid R_h).$$

Now in order to show concretely how the computation goes around let $k - d = r$. Now we need to consider the relationship between r and d. There are two cases to consider: $r \leq d$ or $d < r$. We consider $d < r$ and we assume that $k = 2d + m$ (the other cases are similar). Hence the computation proceeds as shown below:

$$P_1 \mid P_{d+1} \longrightarrow P_1^+ \mid P_{d+1}^-$$
$$P_2 \mid P_{d+2} \longrightarrow P_2^+ \mid P_{d+2}^-$$
$$\vdots$$
$$P_{d-1} \mid P_{2d-1} \longrightarrow P_{d-1}^+ \mid P_{2d-1}^-$$
$$P_d^- \mid P_{2d} \longrightarrow P_d^{-+} \mid P_{2d}^-$$
$$P_{d+1}^- \mid P_{2d+1}^+ \longrightarrow P_{d+1}^{-+} \mid P_{2d+1}^-$$
$$\vdots$$
$$P_{d+(m-1)}^- \mid P_{k-1} \longrightarrow P_{d+(m-1)}^{-+} \mid P_{k-1}^-$$
$$P_{2d}^{-+} \mid P_0^+ \longrightarrow P_{d+m}^{-+} \mid P_0^{+-}$$
$$\vdots$$
$$P_{k-1}^- \mid P_{d-1}^+ \longrightarrow P_{k-1}^{-+} \mid P_{d-1}^{+-}$$

Then the network is partitioned in this way:

$$\mathsf{N}^h = [P_0 \mid \cdots \mid P_d \mid \cdots \mid P_{2d} \mid \cdots \mid P_{k-1}].$$

The computation of the network after the initial step is the following:

$$\mathsf{N}^1 : \quad [P_0^+ \mid P_1 \mid \cdots \mid P_d^- \mid P_{d+1} \cdots \mid P_{k-1}] \qquad \longrightarrow$$
$$\mathsf{N}^2 : \quad [P_0^+ \mid P_1^+ \mid \cdots \mid P_d^- \mid P_{d+1}^- \mid \cdots \mid P_{k-1}] \qquad \longrightarrow$$
$$\vdots$$
$$\mathsf{N}^d : \quad [P_0^+ \mid P_1^+ \mid \cdots \mid P_d^{-+} \mid \cdots \mid P_{2d}^- \mid \cdots \mid P_{k-1}] \qquad \longrightarrow$$
$$\vdots$$
$$\mathsf{N}^k : [P_0^{+-} \mid P_1^{+-} \mid \cdots \mid P_d^{-+} \mid \cdots \mid P_{2d}^{-+} \mid \cdots \mid P_{k-1}^{-+}]$$

Hence we have $\mathcal{C} : \mathsf{N}^1 \longrightarrow \cdots \longrightarrow \mathsf{N}^k$.

(a) Now we have to show that the network N^k is symmetric. Now, N^k is symmetric with respect to σ, since for all s ($0 \leq s \leq k - 1$) the

following holds:

$$P^{+-}_{\hat{\sigma}^h(0)} = (\nu p^h_1 \dots p^h_s)(S_h \mid Q_h \mid R_0))$$
$$= (\nu p^h_1 \dots p^h_s)(\sigma^h(S_0) \mid \sigma^h(Q_0) \mid \sigma^h(R_0))$$
$$= \sigma^h((\nu p^0_1 \dots p^0_s)(S_0 \mid Q_0 \mid R_0))$$
$$= \sigma^h(P^{+-}_0)$$

The automorphism σ has one orbit only since it has not changed.

(b) If for some h such that $0 \le h \le k-1$ we have $\mathsf{N} \;\not\downarrow\; \omega_h$ and $\mathsf{N}^1 \downarrow \omega_h$, then either $P^+_0 \downarrow \omega_h$ or $P^-_d \downarrow \omega_h$. Let us consider $P^+_0 \downarrow \omega_h$ (the other case is similar). We have $P^{+-}_0 \downarrow \omega_h$ from which we conclude by Lemma 5.3 that for all s we have $P^{+-}_{\hat{\sigma}^s(0)} \downarrow \omega_{\hat{\sigma}^s(h)}$. Hence, since σ has one orbit only, for all h such that $0 \le h \le k$ we have $\mathsf{N}^k \downarrow \omega_h$.

(c) If for all j such that $0 \le j \le k-1$ we have $\mathsf{N}^1 \;\not\downarrow\; \omega_j$, then $P_0 \mid P_d \longrightarrow P^+_0 \mid P^-_d \;\not\downarrow\; \omega_j$ if and only if we have $P^+_0 \;\not\downarrow\; \omega_j$ and $P^-_d \;\not\downarrow\; \omega_j$. Now assume for a contradiction that for some h and t such that $0 \le t, h \le k-1$ we have $\mathsf{N}^t \downarrow \omega_{\hat{\sigma}^{t-1}(h)}$ and $\mathsf{N}^{t-1} \;\not\downarrow\; \omega_{\hat{\sigma}^{t-1}(h)}$. Then there are different cases to consider: $t < d$ or $d \le t \le 2d$ or $2d+1 \le t \le k-1$.

  (i) If $t < d$ then we have $P_{\hat{\sigma}^{t-1}(0)} \mid P_{\hat{\sigma}^{t-1}(d)} \longrightarrow P^+_{\hat{\sigma}^{t-1}(0)} \mid P^-_{\sigma^{t-1}(d)} \downarrow \omega_{\hat{\sigma}^{t-1}(h)}$, which means by definition of barbs that either $P^+_{\hat{\sigma}^{t-1}(0)} \downarrow \omega_{\hat{\sigma}^{t-1}(h)}$ or $P^-_{\hat{\sigma}^{t-1}(d)} \downarrow \omega_{\hat{\sigma}^{t-1}(h)}$. We consider $P^+_{\hat{\sigma}^{t-1}(0)} \downarrow \omega_{\hat{\sigma}^{t-1}(h)}$ (the other case is similar). By Lemma 5.3 it holds that $P^+_0 \downarrow \omega_h$ which implies that $\mathsf{N}^1 \downarrow \omega_h$. This is a clear contradiction of the assumption.

  (ii) If $d \le t \le 2d$ then we have $P^-_{\hat{\sigma}^{t-1}(0)} \mid P_{\hat{\sigma}^{t-1}(d)} \longrightarrow P^{-+}_{\hat{\sigma}^{t-1}(0)} \mid P_{\hat{\sigma}^{t-1}(d)} \downarrow \omega_{\hat{\sigma}^{t-1}(h)}$, which implies that either $P^{-+}_{\hat{\sigma}^{t-1}(0)} \downarrow \omega_{\hat{\sigma}^{t-1}(h)}$ or $P_{\hat{\sigma}^{t-1}(d)} \downarrow \omega_{\hat{\sigma}^{t-1}(h)}$. Now we consider $P^{-+}_{\hat{\sigma}^{t-1}(0)} \downarrow \omega_{\hat{\sigma}^{t-1}(h)}$ (the other case is similar to the previous one). Since we have $P^{-+}_{\hat{\sigma}^{t-1}(0)} = (\nu p^{t-1}_1 \dots p^{t-1}_s)(P_{\hat{\sigma}^{t-1}(0)} \mid Q_{\hat{\sigma}^{t-1}(0)} \mid R_{\hat{\sigma}^{t-1}(0)})$ we conclude that either $P_{\hat{\sigma}^{t-1}(0)} \downarrow \omega_{\hat{\sigma}^{t-1}(h)}$ or $Q_{\hat{\sigma}^{s-1}(0)} \downarrow \omega_{\hat{\sigma}^{t-1}(h)}$. Assume that $P_{\hat{\sigma}^{t-1}(0)} \downarrow \omega_{\hat{\sigma}^{t-1}(h)}$. Then by Lemma 5.3 we have $P^+_0 \downarrow \omega_h$, which implies $\mathsf{N}^1 \downarrow \omega_h$. This contradicts our previous assumption. On the other hand, if we have $Q_{\hat{\sigma}^{t-1}(0)} \downarrow \omega_{\hat{\sigma}^{t-1}(h)}$ then for some s we have $\sigma^s(d) = \sigma^{t-1}(0)$; hence $Q_{\hat{\sigma}^s(d)} \downarrow \omega_{\hat{\sigma}^{t-1}(h)}$. Therefore by Lemma 5.3 $Q_d \downarrow \omega_{\hat{\sigma}^{(t-1)-s}(h)}$ and by definition of barbs $P^-_d \downarrow \omega_{\hat{\sigma}^{(t-1)-s}(h)}$ and $\mathsf{N}^1 \downarrow \omega_{\hat{\sigma}^{(t-1)-s}(h)}$, which contradicts our assumption.

  (iii) The case $2d+1 \le t \le k-1$ is similar to the previous one.

(2) The reduction is triggered by the rule RED A-COMM. This means the

redex is formed with communication primitives. Then we have:

$$P_0 = (\nu p_1^0 \dots p_s^0)(\langle n \rangle \mid S_0)$$
$$P_d = (\nu p_1^d \dots p_s^d)((y).Q_d \mid P_d').$$

Thus, with reasoning similar to the previous case, we can conclude that for each j such that $0 \le j \le k - 1$:

$$P_j = (\nu p_1^j \dots p_s^j)(\langle \sigma^j(n) \rangle \mid (\sigma^{j-d}(y)).Q_j \mid R_j)$$

where for each $i < k$ we have $\sigma(Q_i) = Q_{i+1}$ and $\sigma(R_i) = R_{i+1}$. Thus, each process can reduce independently as in the case of 'Computation derived from one process only'. The proof then is identical to that case.

$\square$

So far we have shown that starting from a symmetric state, a network of size k reaches in k steps of computation another symmetric state, where either everyone is a winner or nobody has won the election. In both cases the election for the leader has failed. It remains to show that for any maximal computation, the property described above is preserved. The next lemma expresses exactly this. Recall that $\mathsf{Obs}_k = \{\omega_0, \omega_1, \dots, \omega_{k-1}\}$.

Lemma 5.7 *Let* $\mathsf{N} = [P_0 \mid \cdots \mid P_{k-1}]$ *with* $k \ge 2$ *be a symmetric network in* $MA^{-\mathsf{in}}$. *Then there exists a maximal computation* $\mathcal{C}$ *such that either* $\mathsf{Obs}(\mathcal{C}) \supseteq \mathsf{Obs}_k$ *or* $\mathsf{Obs}(\mathcal{C}) \cap \mathsf{Obs}_k = \emptyset$.

PROOF. First of all, we define a *symmetry-preserving computation of index* n, written $\mathcal{C}_n$, as:

$$\mathcal{C}_0 = \quad \mathsf{N}^{0k} = \mathsf{N}$$

$$\mathcal{C}_n = \begin{cases} \mathcal{C}_{n-1} \text{ if } \mathcal{C}_{n-1} \not\longrightarrow \\[2em] \mathcal{C}_{n-1} \cdot \mathcal{C} \text{ otherwise,} \\ \qquad \text{where } \mathsf{N}^{(n-1)k} \text{ is the final state of } \mathcal{C}_{n-1} \\ \qquad \text{and } \mathcal{C}' : \mathsf{N}^{(n-1)k} \longrightarrow \mathsf{N}^{(n-1)k+1} \\ \qquad \text{and there exists a computation } \mathcal{C}'' \text{ as in Lemma 5.6} \\ \qquad \text{such that } \mathcal{C} = \mathcal{C}' \cdot \mathcal{C}'' \end{cases}$$

For a network of size k, if the computation $\mathcal{C}_n$ of index n has not terminated, then the length of the computation is nk. For each n, the final state N^{nk} of $\mathcal{C}_n$ is symmetric by Lemma 5.6. The definition of observable in a computation naturally extends to the symmetry-preserving computation. By induction on n, the index of the symmetry-preserving computation $\mathcal{C}_n$, we show that $\mathsf{Obs}(\mathcal{C}_n) \supseteq \mathsf{Obs}_k$ or $\mathsf{Obs}(\mathcal{C}_n) \cap \mathsf{Obs}_k = \emptyset$.

$n = 0$) The network $\mathsf{N}^{0k} = \mathsf{N}$ either does not display any winner i.e. for all j such that $0 \leq j \leq k - 1$ we have $\mathsf{N} \not\downarrow \omega_j$, or if for some i we have $\mathsf{N} \downarrow \omega_i$, then by Lemma 5.6 for all h such that $0 \leq h \leq k - 1$ we have $\mathsf{N} \downarrow \omega_h$. Thus, we conclude that either $\mathsf{Obs}(\mathcal{C}_0) \supseteq \mathsf{Obs}_k$ or $\mathsf{Obs}(\mathcal{C}_0) \cap \mathsf{Obs}_k = \emptyset$.

$n > 0$) Assume by induction hypothesis that there exists a symmetry-preserving computation with index $n-1$, written $\mathcal{C}_{n-1}$, which either displays more than one winner, $\mathsf{Obs}(\mathcal{C}_{n-1}) \supseteq \mathsf{Obs}_k$, or displays no winner, $\mathsf{Obs}(\mathcal{C}_{n-1}) \cap \mathsf{Obs}_k = \emptyset$.

According to the definition above of symmetry-preserving computation, there are two cases to consider:

(1) If $\mathcal{C}_{n-1} \not\longrightarrow$, then $\mathcal{C}_n = \mathcal{C}_{n-1}$ and $\mathcal{C}_n$ is maximal. Thus the lemma is proved.

(2) Otherwise, we have $\mathcal{C}_n = \mathcal{C}_{n-1} \cdot \mathcal{C}$ where $\mathcal{C}' : \mathsf{N}^{(n-1)k} \longrightarrow \mathsf{N}^{(n-1)k+1}$, $\mathcal{C}'' : \mathsf{N}^{(n-1)k+1} \longrightarrow \ldots \longrightarrow \mathsf{N}^{nk}$ is as in Lemma 5.6, and $\mathcal{C} = \mathcal{C}' \cdot \mathcal{C}''$.

If $\mathsf{Obs}(\mathcal{C}_{n-1}) \supseteq \mathsf{Obs}_k$ then by Lemma 3.8 $\mathsf{Obs}(\mathcal{C}_n) \supseteq \mathsf{Obs}_k$ and the proof is concluded. Otherwise $\mathsf{Obs}(\mathcal{C}_{n-1}) \cap \mathsf{Obs}_k = \emptyset$. Then there are two cases to consider:

(a) If for some $i < k$ we have $\mathsf{N}^{(n-1)k+1} \downarrow \omega_i$, then, since $\mathsf{N}^{(n-1)k}$ is symmetric, by Lemma 5.6 for all j such that $0 \leq j \leq k$ we have $\mathsf{N}^{(n-1)k+k} \downarrow \omega_j$. Thus, we have that $\mathsf{Obs}(\mathcal{C}'') \supseteq \mathsf{Obs}_k$. Now by Lemma 3.8 it is the case that $\mathsf{Obs}(\mathcal{C}) = \mathsf{Obs}(\mathcal{C}') \cup \mathsf{Obs}(\mathcal{C}'') \supseteq \mathsf{Obs}_k$. Also, $\mathcal{C}_n = \mathcal{C}_{n-1} \cdot \mathcal{C}$. Hence, we conclude that the symmetry-preserving computation with index n, admits all the observables i.e. $\mathsf{Obs}(\mathcal{C}_n) = \mathsf{Obs}(\mathcal{C}_{n-1}) \cup \mathsf{Obs}(\mathcal{C}) \supseteq \mathsf{Obs}_k$. Thus the proof is concluded.

(b) If for all $i < k$ we have $\mathsf{N}^{(n-1)k+1} \not\downarrow \omega_i$, then, since $\mathsf{N}^{(n-1)k}$ is symmetric, by Lemma 5.6, $\mathcal{C}'' : \mathsf{N}^{(n-1)k+1} \longrightarrow \ldots \longrightarrow \mathsf{N}^{nk}$ is such that $\mathsf{N}^{(n-1)k+t} \not\downarrow \omega_i$ for all t, i such that $1 \leq t \leq k$ and $0 \leq i \leq k - 1$. Thus, $\mathsf{Obs}(\mathcal{C}') \cap \mathsf{Obs}_k = \emptyset$ and $\mathsf{Obs}(\mathcal{C}'') \cap \mathsf{Obs}_k = \emptyset$. Now it is the case that $\mathsf{Obs}(\mathcal{C}) = \mathsf{Obs}(\mathcal{C}') \cup \mathsf{Obs}(\mathcal{C}'')$. Hence $\mathsf{Obs}(\mathcal{C}) \cap \mathsf{Obs}_k = \emptyset$. Also $\mathcal{C}_n = \mathcal{C}_{n-1} \cdot \mathcal{C}$. and $\mathsf{Obs}(\mathcal{C}_n) = \mathsf{Obs}(\mathcal{C}_{n-1}) \cup \mathsf{Obs}(\mathcal{C})$. Hence, we conclude that the symmetry-preserving computation with index n for any n, admits no observable i.e. $\mathsf{Obs}(\mathcal{C}_n) \cap \mathsf{Obs}_k = \emptyset$.

The required maximal computation $\mathcal{C}$, either finite or infinite, is obtained by joining (for all indexes n) the symmetry-preserving computations $\mathcal{C}_n$. Clearly, it is the case that either $\mathsf{Obs}(\mathcal{C}) \supseteq \mathsf{Obs}_k$ or $\mathsf{Obs}(\mathcal{C}) \cap \mathsf{Obs}_k = \emptyset$, which concludes the proof of the lemma. $\quad \square$

Theorem 5.8 *Let* $\mathsf{N} = [P_0 \mid \cdots \mid P_{k-1}]$ *with* $k \geq 2$ *be a symmetric network in* $MA^{-\mathsf{in}}$. *Then* N *cannot be an electoral system.*

PROOF. Assume for a contradiction that N is an electoral system. Then for every maximal computation $\mathcal{C}$ a winner has to be elected and $\mathsf{Obs}(\mathcal{C}) = \{\omega_i\}$ with $0 \leq i \leq k - 1$. By Lemma 5.7 there exists a maximal computation $\mathcal{C}'$ such that either $\mathsf{Obs}(\mathcal{C}') \supseteq \mathsf{Obs}_k$ or $\mathsf{Obs}(\mathcal{C}') \cap \mathsf{Obs}_k = \emptyset$. In either case we have a contradiction, and so N cannot be an electoral system. $\square$

We conclude this section by pointing out that negative results hold also for PAC and SA. Let us define $SA^{-\mathsf{in}}$ to be SA without the in capability, and $PAC^{-\mathsf{pull}}$ to be PAC without the pull capability. Then $PAC^{-\mathsf{pull}}$ and $SA^{-\mathsf{in}}$ do not admit symmetric electoral systems either. This can be proved quite easily by minor modification of the proof of Lemma 5.6.

Theorem 5.9 *(1) Let* $\mathsf{N} = [P_0 \mid \cdots \mid P_{k-1}]$ *with* $k \geq 2$ *be a symmetric network in* $SA^{-\mathsf{in}}$. *Then* N *cannot be an electoral system.*

  (2) Let $\mathsf{N} = [P_0 \mid \cdots \mid P_{k-1}]$ *with* $k \geq 2$ *be a symmetric network in* $PAC^{-\mathsf{pull}}$. *Then* N *cannot be an electoral system.*

6 Separation results

In this section, we present the separation results that can be derived from the work done so far.

6.1 When do encodings not exist?

Expressiveness results depend on the existence or not of encodings among calculi. As much as it is necessary to argue that encodings obey certain semantic conditions, one has to argue that encodings that are not respectful of specific semantic conditions do not exist. We argue that in dealing with leader election problems, an encoding must: *preserve the fundamental criteria of the problem* and *not introduce a solution*. We present below the *conditions* for an encoding to preserve symmetric electoral systems, and we show the general result in Lemma 6.2. Afterwards we shall discuss this notion of encoding.

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 $\sigma(\omega_i) = \theta(\omega_i)$, so that $\hat{\sigma}(i) = \hat{\theta}(i)$;*

(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.

The first two items in Definition 6.1 (i.e. uniformity) are as in Palamidessi [24]. The condition of preserving distribution is important in ruling out encodings which make use of a *central server*. The second condition prevents a trivial solution from being introduced by collapsing all the set of natural numbers $\{0, 1, \ldots, k-1\}$ to a $j \in \mathbb{N}$. Notice that the first two items aim to map symmetric networks to symmetric networks of the same size and with the same orbit. The third item aims to preserve the uniqueness of the winner. Because the winner in this framework is represented with a barb, the condition is on barbs.

The condition of respecting observations is our interpretation of Palamidessi's requirement of "preserving a reasonable semantics". She states that a reasonable semantics should distinguish processes which differ on the observables of their maximal computations. In fact, we only require part (3b) to ensure that electoral systems are mapped to electoral systems; part (3a) is added to make the condition more natural. In their version of Palamidessi's work, Sangiorgi and Walker [29] use a condition that if the observables of every maximal computation of a process P are singletons, then the same is true for the encoding of P. This obviously relates very directly to the need to preserve electoral systems. Finally, Ene and Muntean [11] use yet another formulation. As it only refers to finite computations, it would not be enough for our purposes.

There are other criteria on encodings which have been discussed in the literature, such as full abstraction and operational correspondence. Our particular choice is simply motivated by the need to find conditions which are strong enough to preserve electoral systems, and which are no stronger than necessary, so that our separation results are as strong as possible.

Symmetric electoral systems are mapped to symmetric electoral systems by encodings satisfying Definition 6.1:

Lemma 6.2 *Suppose $[\![-]\!] : L \to L'$ is a uniform observation-respecting encoding. Suppose that N is a symmetric electoral system of size k with no globally*

bound names. Let $\mathsf{N}' \stackrel{def}{=} [\![\mathsf{N}]\!]$. *Then* N' *is also a symmetric electoral system of size* k.

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

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

By symmetry, for all i such that $0 \leq i \leq k-1$ we have $\sigma(P_i) = P_{\hat{\sigma}(i)}$ and since $[\![-]\!]$ is permutation-preserving (Definition 6.1(2)), then there exists a θ such that for all $i \in \mathsf{IN}$ we have $\hat{\theta}(i) = \hat{\sigma}(i)$.

$$
\begin{aligned}
\theta([\![P_i]\!]) &= [\![\sigma(P_i)]\!] \text{ by Definition 6.1(2)} \\
&= [\![P_{\hat{\sigma}(i)}]\!] \quad \text{by symmetry} \\
&= [\![P_{\hat{\theta}(i)}]\!] \quad \text{since } \hat{\theta}(i) = \hat{\sigma}(i).
\end{aligned}
$$

Hence $[\![\mathsf{N}]\!]$ is symmetric with respect to θ (with one orbit only). It remains to show that $[\![\mathsf{N}]\!]$ is an electoral system. Consider a maximal computation $\mathcal{C}'$ of $[\![\mathsf{N}]\!]$. By condition (3b) of Definition 6.1 there must exist a computation $\mathcal{C}$ of N such that $\mathsf{Obs}(\mathcal{C}) = \mathsf{Obs}(\mathcal{C}')$. Now since N 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 $[\![\mathsf{N}]\!]$, the lemma is proved. $\quad\square$

6.2   Negative results for the π-calculus

First of all, we show that the π-calculus with mixed choice and without restriction cannot be encoded into the π-calculus with separate choice. This is a stronger version of Palamidessi's result [24].

Corollary 6.3 *There does not exist a uniform observation-respecting encoding from* $\pi_\mathsf{m}^{-\nu}$ *into* π_s.

PROOF. By Proposition 4.2, Theorem 5.1 and Lemma 6.2. $\quad\square$

6.3 Comparing the π-calculus with ambients

In this section the negative results aim to compare the relative strength of the the ambient calculi and the π-calculus dialects.

First we show that pure (i.e. without communication) MA without open cannot be encoded into the π-calculus with separate choice.

Corollary 6.4 *There does not exist a uniform observation-respecting encoding from MA^{io} into π_{s}.*

PROOF. By Proposition 4.5, Theorem 5.1 and Lemma 6.2. □

On the other hand, the π-calculus with mixed choice admits a symmetric electoral system, which implies that there is no encoding from the π-calculus with mixed choice to MA without the in capability.

Corollary 6.5 *There does not exist a uniform observation-respecting encoding from $\pi_{\mathsf{m}}^{-\nu}$ into $MA^{-\mathsf{in}}$.*

PROOF. By Proposition 4.2, Theorem 5.7 and Lemma 6.2. □

Zimmer [32] encoded the synchronous π-calculus without choice into pure SA. He created unique ambients for each channel name, to give inputs and outputs a place to synchronise. The key issue was simulating substitution, which he handled by special "forwarder" ambients. He showed that his encoding satisfies an operational correspondence. Below we show that an encoding in the reverse direction is not possible under our conditions.

Corollary 6.6 *(1) There does not exist a uniform observation-respecting encoding from SA^{iop} into π_{s}.*
(2) There does not exist a uniform observation-respecting encoding from SA^{io} into π_{s}.

PROOF.

(1) By Proposition 4.9 (or Theorem 4.10), Theorem 5.1 and Lemma 6.2.
(2) By Theorem 4.8, Theorem 5.1 and Lemma 6.2.

   □

$\pi_m^{-\nu}$ [24]	MA^{io}	SA^{io}	SA^{iop}	PAC^{pp}
Remark 4.3	Theorem 4.6	Theorem 4.8	Theorem 4.10	Theorem 4.7

π_s [24]	MA^{-in}	SA^{-in}	PAC^{-pull}
Theorem 5.1	Theorem 5.8	Theorem 5.9	Theorem 5.9

Fig. 1. Summary of results so far

6.4   Comparing ambient calculi

This section aims to show that, similarly to the π-calculus world, also within the ambient world there is a hierarchy that is induced by the solution of the leader election problem in some dialects of the ambient calculus.

Corollary 6.7 *There does not exist a uniform observation-respecting encoding from SA^{iop} or SA^{io} into MA^{-in}.*

PROOF. By Proposition 4.9 and Theorem 4.8 together with Theorem 5.7 and Lemma 6.2.   □

Corollary 6.8 *There does not exist a uniform observation-respecting encoding from MA^{io} into MA^{-in}.*

PROOF. By Proposition 4.5, Theorem 5.7 and Lemma 6.2.   □

Corollary 6.9 *There does not exist a uniform observation-respecting encoding from PAC^{pp} into MA^{-in}.*

PROOF. By Theorem 4.7, Theorem 5.7 and Lemma 6.2.   □

In Figure 1 we provide a graphical view of the separation results between the calculi and their dialects dealt with in this section. All calculi above the line have symmetric electoral systems of every size. Those calculi below the line do not have symmetric electoral systems of size greater than one. By Lemma 6.2 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 others.

7 Other calculi

One possible way of interpreting the result that MA with the in and out capabilities only (MA$^{\text{io}}$) has symmetric electoral systems (Theorem 4.6), is that in the presence of trees (as computation on MA can be interpreted), the solution to symmetrical electoral systems is trivial. It has been shown in Theorem 5.8 that this is not strictly true, since in presence of trees but without the ability to enter them, then it does not exist a symmetrical electoral system. One could regard the calculus without the in capability as an 'uninteresting' fragment of MA and still believe that symmetric electoral systems, in calculi with an inherent tree structure, are not interesting.

We conjecture that the existence of electoral systems is related to the syntactic form of *redexes* in calculi; we will show in this section that variants of ambient calculus, namely MA with *objective moves* and SA without *grave interferences*, which retain a form of the in capability and tree structure, can preserve symmetry throughout computation, making a solution to leader election problems impossible.

To see the relationship between the form of redexes in a calculus and solution to leader election problems it suffices to observe that to break the initial symmetry some possible computations have to be pre-empted. In reduction semantics, a term computes if it contains an unguarded redex as subterm. Thus, certain redexes have a syntactic form that destroys redexes of contiguous processes and inhibits certain paths of computation. Consider a concrete example in MA and π_{m}:

$$P = n[\,\text{in}\,m\,] \mid m[\,\text{in}\,n\,] \qquad (1)$$

$$S = n(x) + \overline{m}\langle x \rangle \mid m(x) + \overline{n}\langle x \rangle \qquad (2)$$

The processes P and S contain two redexes, and happen to be a symmetric networks of size 2, with an automorphism $\sigma = \{n \mapsto m, m \mapsto n\}$. There are two common features to observe: neither subprocess of the networks contains composition, and if one of the two redexes reduces, then the other is destroyed.

Consider now the equivalent of the network above in π_{s}:

$$R = R_1 \mid R_2$$

$$R_1 = n(x) \mid \overline{m}\langle x \rangle$$

$$R_2 = m(x) \mid \overline{n}\langle x \rangle \qquad (3)$$

The process R contains two redexes and it is a symmetric network of size 2,

with an automorphism $\sigma = \{n \mapsto m, m \mapsto n\}$. Each subprocess R_1 and R_2 contains composition, and each redex does not pre-empt the other, therefore symmetry is reinstated after two reductions. We leave for future research to find a suitable format on redexes to explain in a general way this phenomenon. For now, we note that the syntactic form of redexes influences preservation of symmetry.

In the remainder of this section, we present two variants of ambient calculi: MA with *objective moves* and SA without *grave interferences*. Both calculi present redexes that contain composition (as in the example of π_s above) and since they preserve symmetry throughout computation, a solution to leader election problems is impossible.

7.1 Objective moves

We consider in this subsection a variant of the ambient calculus with *objective moves*, which we call $\mathrm{MA}^{\mathsf{ob}}$. This variant of the calculus was discussed by Cardelli and Gordon [8]. The objective calculus has two different capabilities with respect to standard MA. Instead of $\mathsf{in}\, n$ and $\mathsf{out}\, n$ there are $\mathsf{mvin}\, n$ and $\mathsf{mvout}\, n$. We replace RED IN and RED OUT by the following reduction rules:

$$\mathsf{mvin}\, n.P \mid n[\, Q\,] \longrightarrow n[\, P \mid Q\,] \quad \text{RED OBJ-IN}$$

$$n[\, \mathsf{mvout}\, n.P \mid Q\,] \longrightarrow P \mid n[\, Q\,] \quad \text{RED OBJ-OUT}$$

Cardelli and Gordon call this type of movement "objective" to distinguish it from the "subjective" movement of standard MA, where ambients move by using their own internal capabilities. Movement in PAC is also objective, since ambients are moved from outside themselves, though PAC movement is defined quite differently from that of $\mathrm{MA}^{\mathsf{ob}}$.

In this variant of the ambient calculus the in capability preserve symmetry, to give a concrete intuition consider equivalent of network (1) above:

$$Q_1 = \mathsf{mvin}\, n.\mathbf{0} \mid m[\,\,]$$

$$Q_2 = \mathsf{mvin}\, m.\mathbf{0} \mid n[\,\,]$$

$$Q = Q_1 \mid Q_2$$

The network Q contains two redexes as well, and similarly to P is symmetric with respect to the automorphism σ above; moreover there are two possible computations only. In comparison to the MA network P in equation (1), P and Q enjoy the same properties so far. However, in Q each subprocess Q_1 and Q_2

contains parallel composition and each redex can reduce without destroying the other (similarly to the π-calculus with separate choice).

We now show that MA$^{\mathsf{ob}}$ does not have symmetric electoral systems, essentially because symmetry cannot be broken.

Theorem 7.1 *Let* $\mathsf{N} = [P_0 \mid \cdots \mid P_{k-1}]$ *be a symmetric network in* MA^{ob} *with* $k \geq 2$*. Then* N *cannot be an electoral system.*

PROOF. See Appendix C. □

Corollary 7.2 *There does not exist a uniform observation-respecting encoding from* MA^{io} *into* MA^{ob}*, or from* π_{m} *into* MA^{ob}*.*

PROOF. By Theorems 4.6 and 7.1, and Lemma 6.2. □

Remark 7.3 *Cardelli and Gordon [8] also discuss a variant form of objective moves with a reduction rule of the form*

$$\mathsf{mv}\, m\, \mathsf{in}\, n.P \mid m[\,Q\,] \mid n[\,R\,] \longrightarrow P \mid n[\,m[\,Q\,] \mid R\,]$$

This form of objective move can *break symmetry (like standard* in*).*

7.2 Safe Ambients without grave interferences

We consider in this subsection a variant of SA without grave interferences, written SA$^{-\mathsf{gi}}$, which was presented by Levi and Sangiorgi [17]. They define syntactically what are grave interferences, and they characterise with a type system a fragment of SA which is free from grave interferences. The type system is beyond the scope of this paper; thus we shall characterise the interference-free fragment syntactically.

In this section we shall prove that SA without grave interferences (SA$^{-\mathsf{gi}}$) does not admit a solution to leader election problems in symmetric networks. We assume a countable set of colours C and let χ range over it. We define coloured Safe Ambients (CSA) with coloured capabilities. Definition 2.9 is updated as follows:

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

Moreover we define coloured processes to be well-formed if:

- capabilities are not allowed after replication;
- the function from colours to capabilities is injective.

We shall only consider well-formed CSA processes. Structural congruence rules are easily extended to coloured processes, and the coloured reduction relation $\longrightarrow_C$, where $C \subset \mathsf{C}$ is a set of colours, is defined as follows:

$$m[\,\mathsf{in}\,n_{\chi_1}.P_1 \mid P_2\,] \mid n[\,\overline{\mathsf{in}}\,n_{\chi_2}.Q_1 \mid Q_2\,]$$
$$\longrightarrow_{\{\chi_1,\chi_2\}} n[\,m[\,P_1 \mid P_2\,] \mid Q_1 \mid Q_2\,] \quad \text{RED S-IN}^*$$

$$n[\,\overline{\mathsf{out}}\,n_{\chi_1}.P_1 \mid P_2 \mid m[\,\mathsf{out}\,n_{\chi_2}.Q_1 \mid Q_2\,]\,]$$
$$\longrightarrow_{\{\chi_1,\chi_2\}} n[\,P_1 \mid P_2\,] \mid m[\,Q_1 \mid Q_2\,] \quad \text{RED S-OUT}^*$$

$$\mathsf{open}\,n_{\chi_1}.P \mid n[\,\overline{\mathsf{open}}\,n_{\chi_2}.Q_1 \mid Q_2\,] \longrightarrow_{\{\chi_1,\chi_2\}} P \mid Q_1 \mid Q_2 \;\; \text{RED S-OPEN}^*$$

$$\langle m \rangle \mid (n).P \longrightarrow_\emptyset P\{m/n\} \qquad\qquad\qquad \text{RED A-COMM}^*$$

Reduction rules for contexts are identical to Definition 2.7. There is an obvious encoding, which strips out all colours, from CSA processes to SA processes. Such an encoding would also preserve closely steps of computation [17].

Grave interferences are defined as follows.

Definition 7.4 *Let $P \in CSA$, and let C_1 and C_2 be two non-empty sets of colours.*

- *Two reductions $P \longrightarrow_{C_1} P_1$ and $P \longrightarrow_{C_2} P_2$* interfere on C_1 and C_2 *if there is no P_3 such that $P_1 \longrightarrow_{C_2} P_3$ and $P_2 \longrightarrow_{C_1} P_3$.*
- *P has* an interference *on C_1 and C_2 if there are two transitions, $P \longrightarrow_{C_1} P_1$ and $P \longrightarrow_{C_2} P_2$, that interfere with C_1 and C_2.*
- *P has a* grave interference *if there exist two disjoint sets of colours C_1 and C_2 such that P has an interference on C_1 and C_2.*

We define Safe Ambients without grave interferences, $\text{SA}^{-\mathsf{gi}}$, to be CSA without grave interferences as in Definition 7.4, with the colours removed.

We can finally introduce our result on processes without grave interferences, namely that if a symmetric network does not have grave interferences, i.e. belongs to $\text{SA}^{-\mathsf{gi}}$, then the leader election problem does not admit a solution. We observe that the network for SA of size 2 presented in Proposition 4.9 contains grave interferences. In fact consider the (partially) coloured version of the process in Proposition 4.9.

$$P_0 \stackrel{def}{=} \text{open } n_0 \mid n_0[\, \text{in } n_{1_{\chi_1}} \mid \overline{\text{in}}\, n_{0_{\chi_2}}.\overline{\text{open}}\, n_0.\omega_0[\, \overline{\text{open}}\, \omega_0\,]\,]$$

$$P_1 \stackrel{def}{=} \text{open } n_1 \mid n_1[\, \text{in } n_{0_{\chi_3}} \mid \overline{\text{in}}\, n_{1_{\chi_4}}.\overline{\text{open}}\, n_1.\omega_1[\, \overline{\text{open}}\, \omega_1\,]\,]$$

$$\mathsf{N} \stackrel{def}{=} P_0 \mid P_1$$

Clearly $P_0 \mid P_1 \longrightarrow {}_{\{\chi_1,\chi_4\}}Q$ and $P_0 \mid P_1 \longrightarrow {}_{\{\chi_3,\chi_2\}}R$ for some Q and R, and $\{\chi_1, \chi_4\} \cap \{\chi_3, \chi_2\} = \emptyset$ and there exists no process S such that Q and R reduce to S.

To prove our main result we need to show some intermediate steps. First of all we define a calculus $SA^{-\mathsf{g^{in}}}$ as the calculus where only the rule RED S-IN is replaced by RED S-IN* and has no grave interferences - as in Definition 7.4. In other words, this is a calculus that has only colours on the in and $\overline{\text{in}}$ and where grave interferences can be defined on in-redexes only. Observing that in Theorem 5.9(1) we have shown that $SA^{-\mathsf{in}}$ does not admit a symmetric electoral system - even in the presence of grave interferences for open and out - then considering the rather restricted form of grave interferences as in $SA^{-\mathsf{g^{in}}}$ makes sense in order to see how constraining the in -redex could lead to a weaker calculus.

Theorem 7.5 *Let* $\mathsf{N} = [P_0 \mid \cdots \mid P_{k-1}]$ *be a symmetric network in* $SA^{-\mathsf{g^{in}}}$ *with* $k \geq 2$. *Then* N *cannot be an electoral system.*

PROOF. (Sketch) By Theorem 5.9(1) we only need to consider the case of two communicating processes using the RED S-IN* rule. Without loss of generality we assume that P_0 and P_d make a transition and for some P_0', P_0'', P_0''' and Q_d', Q_d'', Q_d''' we have:

$$P_0 = n_0[\, \text{in } n_d.P_0' \mid P_0''\,] \mid P_0'''$$

$$P_d = n_d[\, \overline{\text{in}}\, n_d.Q_d' \mid Q_d''\,] \mid Q_d'''.$$

Observing that $P_d = P_{\hat{\sigma}^d(0)}$ then for all i such that $0 \leq i \leq k-1$ we have:

$$P_i = n_i[\, \text{in } n_{i+d}.P_i' \mid P_i''\,] \mid n_i[\, \overline{\text{in}}\, n_i.Q_i' \mid Q_i''\,] \mid Q_i''' \qquad (1)$$

with $P_i''' \equiv n_i[\, \overline{\text{in}}\, n_i.Q_i' \mid Q_i''\,] \mid Q_i'''$ or

$$P_i = n[\, \text{in } n_{i+d}.P_i' \mid \overline{\text{in}}\, n_i.Q_i' \mid S_i\,] \mid R_i \qquad (2)$$

with $P_i'' \equiv \overline{\text{in}}\, n_i.Q_i' \mid S_i$ and $P_i''' \equiv R_i \not\equiv (\nu m_1 \ldots m_h)(n_i[\, \overline{\text{in}}\, n_i.T_1 \mid T_2\,] \mid T_3)$ for some processes T_1, T_2, T_3. If the P_i have the form in equation (1) then, with a reasoning similar to Lemma C.3 of the Appendix, we see that symmetry is

re-established in k reductions. Otherwise, we consider the network coloured in the following way: $\text{in}\,n_j$ takes colour χ_j and $\overline{\text{in}}\,n_j$ takes colour $\overline{\chi_j}$. Thus:

$$[P_0 \mid P_1 \mid \cdots \mid P_d \mid \cdots \mid P_{k-1}] \longrightarrow {}_{\{\chi_0,\overline{\chi_d}\}}$$
$$[R_0 \mid P_1 \ldots \mid (n_d[\,n_0[\,P_0' \mid P_0''\,] \mid P_d''\,] \mid R_d) \mid \cdots \mid P_{k-1}].$$

We consider also the pair P_{-d} and P_0. Then:

$$[P_0 \mid P_1 \mid \cdots \mid P_{-d} \mid \cdots \mid P_{k-1}] \longrightarrow {}_{\{\chi_{-d},\overline{\chi_0}\}}$$
$$[(n_0[\,n_{-d}[\,P_{-d}' \mid P_{-d}''\,] \mid P_0''\,] \mid R_0) \mid P_1 \mid \cdots \mid R_{-d} \mid \cdots \mid P_{k-1}].$$

Thus $\mathsf{N} \longrightarrow {}_{\{\chi_0,\overline{\chi_d}\}}\mathsf{N}'$ and $\mathsf{N} \longrightarrow {}_{\{\chi_{-d},\overline{\chi_0}\}}\mathsf{N}''$ and $\{\chi_0,\overline{\chi_d}\}\cap\{\chi_{-d},\overline{\chi_0}\} = \emptyset$. There does not exist a process M such that $\mathsf{N}' \longrightarrow {}_{\{\chi_{-d},\overline{\chi_0}\}}M$ and $\mathsf{N}'' \longrightarrow {}_{\{\chi_0,\overline{\chi_d}\}}M$ since R_0 would not be able to communicate with P_{-d} in N'. Contradiction with the assumption that N did not have grave interferences. Therefore for each i such that $0 \leq i \leq k - 1$ the processes P_i in the network have the syntactic from in equation (1). $\square$

The result above implies that $\text{SA}^{-\mathsf{gi}}$ does not admit a symmetric electoral system.

Corollary 7.6 *Let* $\mathsf{N} = [P_0 \mid \cdots \mid P_{k-1}]$ *be a symmetric network in* $\text{SA}^{-\mathsf{gi}}$ *with* $k \geq 2$. *Then* N *cannot be an electoral system.*

As consequence of Corollary 7.6 and [17, Corollary 5.14], Levi and Sangiorgi's single-threadness typed SA does not admit a symmetric electoral system, and thus it is less expressive than full SA. The result presented in this section shed light on the strength of a very restrictive form of in-redex. In [17, Definition A.2] Levi and Sangiorgi present the complete list of grave interference patterns. We claim that SA with grave interference in the form of their pattern 5 suffices to have a calculus that can solve leader election problems in networks of size at least three. To solve leader election problems in symmetric networks of size two, the degenerate $n[\,\text{in}\,h.P \mid \overline{\text{in}}\,n.Q\,] \mid h[\,\text{in}\,n.P' \mid \overline{\text{in}}\,h.Q'\,]$ form of pattern 5 is needed. Moreover the exclusion of grave interference pattern 5 is sufficient to make symmetric leader elections impossible.

8 Failure to declare winner

In this section we introduce a completely different kind of failure to elect the leader in symmetric network. Just as $\text{MA}^{-\mathsf{in}}$ does not have symmetric electoral

systems, we can show that MA with only the **out** capability removed, which we call MA$^{-\mathsf{out}}$, does not have symmetric electoral systems. The idea is that even if symmetry is broken we need the **out** capability to declare the winner at the top level. This is not obvious, since we saw in Theorem 4.10 that the **in** and **open** capabilities of SA can be used together to form an electoral system. The two capabilities are complementary, in that **in** can increase depth, while **open** can reduce it. But in MA we do not have enough control to know when to apply **open** reductions, so that a symmetric electoral system with **in** and **open** (and even communication, but not **out**) is impossible.

Proposition 8.1 *For every $k \geq 2$, MA$^{-\mathsf{out}}$ does not have a symmetric electoral system of size k.*

PROOF. See Appendix D. □

Recall that we can construct symmetric electoral systems in SA using the **in** and **out** capabilities (Theorem 4.8), or using **in** and **open** (Theorem 4.10). The next result is the analogue of Proposition 8.1. Let SA$^{-\mathsf{out},\mathsf{open}}$ denote SA without the **out** and **open** capabilities.

Proposition 8.2 *For every $k \geq 2$, SA$^{-\mathsf{out},\mathsf{open}}$ does not have a symmetric electoral system of size k.*

PROOF. See Appendix D. □

In a similar fashion, let PAC$^{-\mathsf{push}}$ denote PAC without the **push** capability.

Proposition 8.3 *For every $k \geq 2$, PAC$^{-\mathsf{push}}$ does not have a symmetric electoral system of size k.*

PROOF. Similar to that of Proposition 8.1, and omitted. □

9 Concluding remarks

We summarise our results in a diagram (Figure 2). All calculi above the double line have symmetric electoral systems of every size. Those calculi below the double line do not have symmetric electoral systems of size greater than one. Of these latter calculi, those above the single line fail because they are not guaranteed to break symmetry, while those below the single line fail because

$$SA^{\mathsf{iop}}$$
$$\text{Theorem 4.10}$$

$\pi_{\mathsf{m}}^{-\nu}$			
[24]	MA^{io}	SA^{io}	PAC^{pp}
Remark 4.3	Theorem 4.6	Theorem 4.8	Theorem 4.7

$$SA^{-\mathsf{gi}}$$
$$\text{Corollary 7.6}$$

π_{s}				
[24]	$MA^{-\mathsf{in}}$	$SA^{-\mathsf{in}}$	$PAC^{-\mathsf{pull}}$	MA^{ob}
Theorem 5.1	Theorem 5.8	Theorem 5.9	Theorem 5.9	Theorem 7.1

$MA^{-\mathsf{out}}$	$SA^{-\mathsf{out,open}}$	$PAC^{-\mathsf{push}}$
Proposition 8.1	Proposition 8.2	Proposition 8.3

Fig. 2. Summary of results

they cannot report the result of the election. By Lemma 6.2 there is no arrow going from any calculus above the double line to any calculus below the double line, which yields the separation results in this paper, together with a number of other results.

In this paper we have dealt with expressiveness results via the leader election problem, in order to compare the π-calculus and ambient calculi. We have seen that a fragment of MA is not encodable in the π-calculus with separate choice. We have shown that for MA, the crucial capability for the solution of the leader election problem in symmetric networks is the in capability as *subjective move*. In fact, without this capability, the election problem cannot be solved in MA. We have extended our results to other ambient calculi. We showed that a small fragment of PAC and a small fragment of SA can solve leader election problems in symmetric networks. Also, in PAC without the push capability and SA without the in capability the election problem cannot be solved either. We have also considered MA with objective moves, which is a dialect of MA with a prefixing form of the in capability. We have shown that in this case the initial symmetry cannot be broken, and a leader cannot be elected. These results give insight into why symmetry cannot be broken. In fact we conjecture that the syntactic form of redexes can discriminate between calculi that can break symmetry and ones that cannot. Our results give a fine-grained hierarchy in ambient calculi.

We also considered a side issue of the failure to elect a leader in the case in which the winner cannot be reported, for instance in cases in which the out capability is missing. We have reported some results, bearing in mind that this is a different problem from not being able to break the initial symmetry, as happens in π_{s}.

Expressiveness of different fragments of the ambient calculi has been considered also under the aspect of Turing completeness [6,19]. This is an orthogonal issue with respect to leader election problems. In fact by looking at the algorithms implemented in each calculus, we can see that leader election problems can be solved in finite fragments of calculi - i.e. not Turing-complete - while solutions to leader election problems may not exist in Turing-complete calculi like π_s. It is, however worth observing that both MA^{io} and PAC^{pp} are in a sense minimal calculi that solve leader election problems and are Turing complete (by the results of this paper and of [19], respectively). By "minimal", we mean that if we remove either of the two capabilities in each fragment, then we can no longer perform leader election, and Turing completeness fails.

We briefly discuss some issues related to implementation of MA. In [12] there is given an encoding of MA into the Distributed Join Calculus, which has led to the implementation of MA into JoCaml. For that encoding, the migration of ambients, both entering and exiting, has been separated into three atomic steps. Moreover, each migration of ambients happens via the centralised control of the parent. For example, if $a[\,in\,b\,]$ wants to migrate to sibling ambient b, then firstly a has to forward a request to the parent ambient c; secondly c informs b of the request, and finally b becomes the parent of a. The ambient c deals with all the requests regarding its child ambients; thus we cannot regard this encoding as fully distributed in the sense specified by Definition 6.1, since the encoding of the electoral system in Proposition 4.5 would make use of a parent ambient to deal with the request of entering. This can be regarded as introducing a centralised server, which makes electing a leader trivial.

We regard the work done in this paper as the starting point in exploring further the relationship between the π-calculus and the new generation of process calculi. The work has a wide applicability to similar process calculi, such as Boxed Ambients, the Seal Calculus, etc. From Theorem 4.6 it follows trivially that Boxed Ambients [5] can solve the leader election problem in symmetric networks. We conjecture that the Seal calculus [9] should also admit a solution to the leader election problem, since the *move in* seems to be a *symmetry-breaking* operator. On the other hand we speculate that the pure version of $D\pi$ [13] cannot solve such a problem.

Acknowledgements

A Proof of Theorem 4.6

We give here the detailed proof of Theorem 4.6. We have to introduce some notation, which will be helpful in the proofs of Theorems 4.6 and 4.10.

Definition A.1 *Let* $k \geq 1$. *Let* $i, j \in \{0, \ldots, k-1\}$. *By* $(i \mathinner{..} j) \mod k$ *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:*

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

In particular, $(i \mathinner{..} i{+}1) \mod k = \emptyset$ *and* $(i \mathinner{..} i) \mod k = \{0, \ldots, k-1\} - \{i\}$. *We shall tend to suppress the "mod* k*" and write* $(i \mathinner{..} j)$.

Theorem 4.6 *In* $\mathsf{MA^{io}}$, *for any* $k \geq 1$ *there exists a symmetric electoral system of size* k.

PROOF. Let $k \geq 1$. The electoral system is defined by $\mathsf{N} \overset{def}{=} \prod_{i<k} P_i$ where

$$
P_i \overset{def}{=} n_i[\, \textstyle\prod_{j \neq i} \mathsf{in}\, n_j.lose_i[\, Outn\,]\,] \mid c_i[\, C_{i,i+1}\,]
$$
$$
Outn \overset{def}{=} \textstyle\prod_{j<k} !\,\mathsf{out}\, n_j
$$
$$
C_{i,i} \overset{def}{=} \omega_i[\, \mathsf{out}\, c_i\,]
$$
$$
C_{i,j} \overset{def}{=} \mathsf{in}\, lose_j.C'_{i,j} \qquad (j \neq i)
$$
$$
C'_{i,j} \overset{def}{=} \mathsf{out}\, lose_j.C_{i,j+1} \qquad (j \neq i)
$$

(We use arithmetic modulo k for the indices.) The proof that we have indeed defined an electoral system follows an outline similar to the proof of Lemma 4.4 in [28], although the details are quite different. First we formulate an invariant which describes the state of the network at each stage of the computation up to the state immediately before a winner is declared.

Invariant: the network is of the form

$$
\prod_{i \in N} n_i[\, R_i\,] \mid \prod_{i \in A} c_i[\, C_{i,s(i)}\,] \mid \prod_{j<k} \prod_{l<d_j} lose_j[\, Outn \mid \prod_{i \in B_{jl}} c_i[\, C'_{i,s(i)}\,]\,]
$$

where each R_i is of the form

$$
\prod_{j \in E_i} \mathsf{in}\, n_j.lose_i[\, Outn\,] \mid \prod_{j<k} (lose_j[\, Outn\,])^{d_{ij}} \mid \prod_{l \in L_i} n_l[\, R_l\,]\, .
$$

47

(Here P^d means d copies of P in parallel.) We also require the following conditions:

(1) $N \neq \emptyset$ (N is the set of processes which have not (yet) lost.)
(2) $N \cup \bigcup_{j<k} L_j$ is a partition of $\{0, \ldots, k-1\}$. (This states that every ambient n_i is either still at the top level, or has already lost, and is contained in some other n_j ambient.)
(3) For all $i \in N$ we have $E_i = \{0, \ldots, k-1\} - \{i\}$. (This implies that every top-level n_i ambient can enter any other top-level n_j ambient.)
(4) For all $j < k$, $j \notin N$ iff $d_j > 0$ or for some $i < k$ we have $d_{ij} > 0$. (This states that process j has lost iff there is an unguarded $lose_j$ ambient.)
(5) $A \cup \bigcup_{j<k} \bigcup_{l<d_j} B_{jl}$ is a partition of $\{0, \ldots, k-1\}$. (This states that every c_i ambient occurs exactly once, either at the top level, or inside some top-level $lose_j$ ambient.)
(6) For all $i < k$ we have $0 \leq s(i) < k$. For all $i, j < k$, if $j \in (i \mathrel{..} s(i))$ then $d_j > 0$. Also if $i \in B_{jl}$ (some $j < k$, $l < d_j$) then $s(i) = j \neq i$ and $d_j > 0$.

The invariant is established initially with $N = A = \{0, \ldots, k-1\}$, $E_i = \{0, \ldots, k-1\} - \{i\}$, $L_i = \emptyset$, $d_j = d_{ij} = 0$, $s(i) = i+1$ (all $i, j < k$).

Immediately before i is announced as the winner we shall show that the network will have the form

$$(*) \qquad n_i[\, R_i\,] \mid \prod_{j \in A} c_j[\, C_{j,s(j)}\,] \mid \prod_{j<k} \prod_{l<d_j} lose_j[\, Outn \mid \prod_{j' \in B_{jl}} c_{j'}[\, C'_{j',s(j')}\,]\,]$$

with $i \in A$, $s(i) = i$, $d_i = 0$ and $d_j \geq 1$ (all $j \neq i$). At this point $\omega_i[\, \mathsf{out}\, c_i\,]$ can emerge from ambient c_i, yielding an ω_i barb.

We need to show five properties:

(1) The invariant is maintained by any reduction, apart from an $\mathsf{out}\, c_i$ reduction.
(2) Every computation is finite.
(3) A computation can always make progress if it has not yet reached form $(*)$. Since all computations are finite (previous item), this shows that every computation does reach $(*)$, from which the winner can be announced in a single reduction. Hence every computation announces a winner.
(4) A computation can only announce a winner by first reaching form $(*)$. In particular, if an $\mathsf{out}\, c_i$ reduction occurs then the network is of form $(*)$ immediately before the reduction.
(5) Once a winner is announced no further reductions can produce a second winner. So every computation has a unique winner.

We first show property (1). There are four cases, depending on the type of reduction:

(in n_j) An in n_j reduction must come from an ambient n_i entering ambient n_j. This can be either at the top level or at a lower level. In the first case we have $i, j \in N$ with $i \neq j$. The effect is that i is removed from N and added to L_j. Also $lose_i[\,Outn\,]$ is unleashed inside n_i, which means that d_{ii} increases from 0 to 1. In the second case we have $i, j \in L_l$ for some l. Then i is removed from L_l and added to L_j, and d_{ii} increases by 1.

(out n_j) An out n_j reduction arises when a $lose_i$ ambient exits n_j. It will arrive either at the top level (if $j \in N$) or in an ambient n_l (if $j \in L_l$). In the first case d_{ji} decreases by 1 and d_i increases by 1. In the second case, d_{ji} also decreases by 1 while d_{li} increases by 1.

(in $lose_j$) An in $lose_j$ reduction arises when a $c_i[\,C_{i,s(i)}\,]$ ambient enters a top-level $lose_j$ ambient. Then $s(i) = j$ and $d_j > 0$. The effect of the reduction is that $s(i)$ remains unchanged, while i is removed from A and added to B_{jl} (some $l < d_j$).

(out $lose_j$) An out $lose_j$ reduction arises when a $c_i[\,C'_{i,s(i)}\,]$ ambient exits a top-level $lose_j$ ambient. Then $s(i) = j$, $d_j > 0$ and $i \in B_{jl}$ for some $l < d_j$. The effect of the reduction is that $s(i)$ increases by 1, while i is removed from B_{jl} and added to A.

Note that in each of the four cases nothing is added to N and d_i is not reduced. We omit the straightforward checks that the conditions of the invariant are preserved by the four types of reduction.

We now show that every computation is finite (property (2)). The only issue is the replicated out n_j capabilities in the $lose_i$ ambients. Consider a single $lose_i$ ambient. When first unleashed it is at an ambient nesting depth of between 1 and k. Every time it performs an out n_j the depth decreases by 1. The only way its depth can increase is if a containing n_j ambient enters another n_l ambient. But this can only happen a finite number of times. Hence there is a finite bound on the number of times $lose_i$ can perform out n_j reductions.

We now show that a reduction is always possible if the network has not reached form (*) (property (3)). First suppose that $|N| \geq 2$. Take $i, j \in N$ with $i \neq j$. By condition (3) it is possible for ambient n_i to enter ambient n_j. So by condition (1) we can assume that $N = \{i\}$ for some $i < k$.

Now suppose that $d_j = 0$ for some $j \neq i$. Since $j \notin N$, by condition (4) we have $d_{lj} > 0$ for some l. This means that there is a $lose_i$ ambient inside ambient n_l, and it can exit n_l. Hence we can assume that $d_j > 0$ for all $j \neq i$.

Next suppose that $i \notin A$. Then by condition (5) we see that c_i is inside some top-level $lose_j$ ambient, and can exit by condition (6). Hence we can assume $i \in A$.

Finally suppose $s(i) \neq i$. Then $d_{s(i)} > 0$, and c_i can enter some top-level $lose_{s(i)}$ ambient. This establishes property (3).

We now show that in order for a winner to be announced form $(*)$ must first be reached (property (4)). Suppose that barb ω_i appears for the first time. Then the invariant holds for all previous stages of the computation. The last reduction must have been an $\mathsf{out}\, c_i$, and so at the immediately preceding stage we must have $c_i[\, C_{i,i}\,]$ at the top level, so that $s(i) = i$. For all $j \neq i$ we have $d_j > 0$ (using condition (6)) and $j \notin N$ (using condition (4)). Therefore we have $N = \{i\}$ and we are in form $(*)$ immediately before i is announced as the winner.

Finally we show that once a winner is announced no further computation can produce another winner (property (5)). By property (4) we know that if process i wins then form $(*)$ has been reached, and $N = \{i\}$. Now $N = \{i\}$ will continue to hold for the rest of the computation, since $|N| \geq 1$ (condition (1)) and no element can be added to N, as noted when verifying property (1). So $d_i = 0$ by condition (4). Hence if $j \neq i$ then $i \notin (j \,..\, s(j))$ (using condition (6)) and so $s(j) \neq j$, so that j cannot win.

If the replication operator in the process $Outn$ is omitted we still have an electoral system. This can be shown by a refinement of the proof given. We used replication merely in order to simplify the statement of the invariant. $\quad\square$

B Proof of Theorem 4.10

Theorem 4.10 In SA$^{\mathsf{iop}}$, for any $k \geq 1$ there exists a symmetric electoral system of size k.

PROOF. Let $k \geq 1$. For $i, j < k$ we define

$$P_i \stackrel{def}{=} \mathsf{open}\, n_i \mid C_{i,i+1} \mid n_i[\, \overline{\mathsf{in}}\, n_i.\overline{\mathsf{open}}\, n_i.lose_i[\, \overline{\mathsf{open}}\, lose_i\,] \mid \textstyle\prod_{j \neq i} \mathsf{in}\, n_j\,]$$
$$C_{i,i} \stackrel{def}{=} \omega_i[\, \overline{\mathsf{open}}\, \omega_i\,]$$
$$C_{i,j} \stackrel{def}{=} \mathsf{open}\, lose_j.(lose_j[\, \overline{\mathsf{open}}\, lose_j\,] \mid C_{i,j+1}) \quad (j \neq i)$$

Then $\prod_{i<k} P_i$ is an electoral system. The idea is that process j loses to process i when ambient n_i enters ambient n_j. Once a process j has lost, n_j cannot be entered further, but n_j can be opened if it is at the top level, in which case an ambient $lose_j$ appears at the top level to signify that i has lost. Eventually all but one processes have lost. Suppose that only n_i is left. The checking process $C_{i,i+1}$ checks for the presence of $lose_j$ (all $j \neq i$), and then announces i as the winner.

To prove that $\prod_{i<k} P_i$ is indeed an electoral system, we formulate an invariant,

which will hold for every possible state of the network.

Invariant. The network is of the form

$$\prod_{i\in O} \text{open}\, n_i \mid \prod_{i<k} C_{i,s(i)} \mid \prod_{i\in T} n_i[\,R_i\,] \mid \prod_{j\in L} lose_j[\,\overline{\text{open}}\, lose_j\,] \mid \prod_{i<k}(\text{in}\, n_i)^{r(i)}$$

with $O, T, L \subseteq \{0, \ldots, k-1\}$, and with R_i in one of the following two forms:

$$R_i = \overline{\text{in}}\, n_i.\overline{\text{open}}\, n_i.lose_i[\,\overline{\text{open}}\, lose_i\,] \mid \prod_{j\in E_i} \text{in}\, n_j \qquad (NE)$$

$$R_i = \overline{\text{open}}\, n_i.lose_i[\,\overline{\text{open}}\, lose_i\,] \mid n_{w(i)}[\,R_{w(i)}\,] \mid \prod_{j\in E_i} \text{in}\, n_j \ (E)$$

Here the form (NE) is for n_i ambients which have not (yet) been entered, and (E) is for n_i ambients which have been entered by ambient $n_{w(i)}$, and have therefore lost to process $w(i)$. We impose the following conditions:

(1) For each $i < k$, either there is exactly one n_i ambient present in the network, in which case $i \in O$, or else there is no n_i ambient present in the network, in which case $i \notin O$.
(2) $w(i)$ is a partial one-one function taking values in $\{0, \ldots, k-1\}$. $w(i)$ is defined iff ambient n_i is present in the network (i.e. not yet opened) and R_i is of form (E). (It follows from this, condition (1) and the structure of the invariant itself that T and $\{w(i) : i < k$ and $w(i)$ is defined$\}$ together form a partition of O. This states that if n_i has not yet been opened, then it is present either at the top level, or at a lower level inside another n_j.)
(3) $|T| \geq 1$.
(4) For all $i, j < k$, if $j \neq i$, $j \notin L$ and $w(j)$ is undefined then $j \in E_i$. This states that n_i has the capability to enter n_j as long as n_j has not been opened and has not been entered.
(5) O and L together form a partition of $\{0, \ldots, k-1\}$. This states that each n_i is either not yet opened or else has been opened, in which case $lose_i$ is present.
(6) For all $i < k$, $(i \mathrel{..} s(i)) \subseteq L$. This states that if j has been checked by i as having lost then ambient $lose_j$ is present at the top level.

The invariant is established initially with $O = T = \{0, \ldots, k-1\}$, $s(i) = i+1$, $L = \emptyset$, $r(i) = 0$, $E_i = \{0, \ldots, k-1\} - \{i\}$, $w(i)$ undefined.

There are three types of reduction: $\text{in}\, n_j$, $\text{open}\, n_j$ and $\text{open}\, lose_j$. We consider each in turn.

($\text{in}\, n_j$) Suppose that n_i enters n_j. This can only happen at the top level, with $i, j \in T$ and $i \neq j$. Clearly R_j must have been of form (NE), and changes as a result of the reduction to be of form (E). Also $w(j)$ was undefined, and is now set to i. At the same time i is removed from T. Also j is removed from E_i.

(**open** n_j) Suppose that n_j is opened. Again this can only happen at the top level. R_j must be of type (E). We have $j \in T$ and $w(j)$ defined. As a result of the reduction, ambient $lose_j$ is unleashed. We remove j from T and add it to L. Also the index $w(j)$ is added to T and then $w(j)$ becomes undefined. The term $\prod_{i \in E_j}$ **in** n_i is now garbage; for each $i \in E_j$ we increase $r(i)$ by one.

(**open** $lose_j$) Suppose that $C_{i,s(i)}$ opens $lose_j$. Then $j \in L$ and $s(i) = j \neq i$. The effect of the reduction is to leave L unchanged and increase $s(i)$ by one.

We omit the straightforward checks that in each of the three cases the invariant is maintained.

Now we argue that progress can always be made unless $s(i) = i$ for some i. This will imply that there is at least one winner in every maximal computation. Suppose that no **in** n_j reduction is enabled. Then either $|T| = 1$ or else $|T| \geq 1$ and for every $j \in T$, function $w(j)$ is defined (using conditions (2), (3), (4)). Suppose also that no **open** n_j reduction is enabled. Then for every $j \in T$, function $w(j)$ is undefined (using conditions (1), (2)). Combining, we must have $|T| = 1$. Let $T = \{i\}$, with $w(i)$ undefined. It is clear from the invariant that no n_j ambient can be present for $j \neq i$. Hence $O = \{i\}$ by (1) and $L = \{0, \ldots, k-1\} - \{i\}$ by (5).

Suppose also that no **open** $lose_j$ reduction is enabled. If $s(i) \neq i$ then $s(i) \in L$, and $C_{i,s(i)}$ can perform **open** $lose_{s(i)}$, which is a contradiction. Therefore $s(i) = i$, and so ω_i is unguarded at the top level of the system, announcing i as the winner.

Finally we argue that there can be at most one winner. Suppose for a contradiction that in some computation, i is announced as the winner, and that at that stage or later some $j \neq i$ is also announced as the winner. Then by the invariant we have $s(i) = i$ and $s(j) = j$. By condition (6) we must have $(i \mathbin{..} i) \subseteq L$ and $(j \mathbin{..} j) \subseteq L$, which implies $L = \{0, \ldots, k-1\}$. Hence $O = \emptyset$ by (5), and so $T = \emptyset$ by (1). But this contradicts (3). □

C Proof of Theorem 7.1

First we need some lemmas.

Lemma C.1 *Let P be a process in MA^{ob} and σ a permutation. Then $P \downarrow n$ if and only if $\sigma(P) \downarrow \sigma(n)$.*

PROOF. The proof is identical to the one in Lemma 5.3. □

Lemma C.2 *Let P be a process in MA^{ob} and σ a substitution. If $P \longrightarrow P'$, then $\sigma(P) \longrightarrow \sigma(P')$.*

PROOF. By induction on $\longrightarrow$. $\square$

Lemma C.3 *Let $\mathsf{N} = [P_0 \mid \cdots \mid P_{k-1}]$ be a symmetric network in MA^{ob}. Assume that $\mathsf{N} \longrightarrow \mathsf{N}^1$. Then there exists a computation $\mathcal{C} : \mathsf{N}^1 \longrightarrow \cdots \longrightarrow \mathsf{N}^k$ such that:*

(1) N^k is symmetric;
(2) if for some i such that $0 \le i \le k-1$ we have $\mathsf{N}^1 \downarrow \omega_i$ and $\mathsf{N} \not\downarrow \omega_i$ then for all h such that $0 \le h \le k-1$ we have $\mathsf{N}^k \downarrow \omega_h$;
(3) if for all j such that $0 \le j \le k-1$ we have $\mathsf{N}^1 \not\downarrow \omega_j$ then for all t, j such that $1 \le t \le k$ and $0 \le j \le k-1$ we have $\mathsf{N}^t \not\downarrow \omega_j$.

PROOF. Assume that N is symmetric with respect to an automorphism σ with one orbit only and that $\mathsf{N} \longrightarrow \mathsf{N}^1$. There are two cases to consider:

Reduction derived from one process only. In this case $\mathsf{N} \longrightarrow \mathsf{N}^1$ is the result of the computation of one process in the network. Assume P_r in N has reduced as follows $P_r \longrightarrow P_r^\dagger$. The proof for this case is identical to Lemma 5.6.

Reduction derived from the interaction of two processes. We assume, without loss of generality and to the mere end of simplifying the notation, that P_0 and P_d are the two processes involved in the computation. We will silently use structural congruence to move processes to adjacent positions in order to perform a reduction, and to move them back to their original positions. Moreover, in order to simplify the notation, we assume that $\hat{\sigma}(i) = i + 1$ for each $i < k$.

By the operational semantics of this calculus, there are three cases to consider.

(1) The reduction has occurred by using the rule RED OBJ-IN. Then without loss of generality we can assume that P_0 performs the entry into another ambient. Then the syntactic form of the two processes is:

$$P_0 = (\nu p_1^0 \ldots p_s^0)(\mathsf{mvin}\, n.S_0 \mid P_0')$$
$$P_d = (\nu p_1^d \ldots p_s^d)(n[\, Q_d \,] \mid P_d')$$

with $n \notin \{p_1^0 \ldots p_s^0\} \cup \{p_1^d \ldots p_s^d\}$ (otherwise the two processes would not be able to compute). By symmetry, we conclude the following state-

ments starting from P_0.

$$P_{\hat\sigma(0)} = (\nu p_1^1 \ldots p_s^1)(\mathsf{mvin}\ \sigma(n).\sigma(S_0) \mid \sigma(P_0'))$$

$$P_{\hat\sigma^2(0)} = (\nu p_1^2 \ldots p_s^2)(\mathsf{mvin}\ \sigma^2(n).\sigma^2(S_0) \mid \sigma^2(P_0'))$$

$$\vdots$$

$$P_{\hat\sigma^{k-1}(0)} = (\nu p_1^{k-1} \ldots p_s^{k-1})(\mathsf{mvin}\ \sigma^{k-1}(n).\sigma^{k-1}(S_0) \mid \sigma^{k-1}(P_0')).$$

Similar conclusions can be drawn starting from P_d.

$$P_{\hat\sigma(d)} = (\nu p_1^{d+1} \ldots p_s^{d+1})(\sigma(n)[\,\sigma(Q_d)\,] \mid \sigma(P_d'))$$

$$P_{\hat\sigma^2(d)} = (\nu p_1^{d+2} \ldots p_s^{d+2})(\sigma^2(n)[\,\sigma^2(Q_d)\,] \mid \sigma^2(P_d'))$$

$$\vdots$$

$$P_{\hat\sigma^{k-1}(d)} = (\nu p_d^{d-1} \ldots p_s^{d-1})(\sigma^{k-1}(n)[\,\sigma^{k-1}(Q_d)\,] \mid \sigma^{k-1}(P_d')).$$

We see that

$$P_0 = (\nu p_1^0 \ldots p_s^0)(\mathsf{mvin}\ n.S_0 \mid \sigma^{-d}(n)[\,Q_0\,] \mid R_0)$$

$$P_d = (\nu p_1^d \ldots p_s^d)(\mathsf{mvin}\ \sigma^d(n).S_d \mid n[\,Q_d\,] \mid R_d)$$

(we use arithmetic modulo k, so that $\sigma^{-d}(n)$ stands for $\sigma^{k-d}(n)$) and in general, for all h such that $0 \le h \le k-1$ we have

$$P_h = (\nu p_1^h \ldots p_s^h)(\mathsf{mvin}\ \sigma^h(n).S_h \mid \sigma^{h-d}(n)[\,Q_h\,] \mid R_h)$$

where for all $i < k$ we have $\sigma(S_i) = S_{i+1}$, $\sigma(Q_i) = Q_{i+1}$ and $\sigma(R_i) = R_{i+1}$.

Now if $P_0 \mid P_d \longrightarrow P_0^+ \mid P_d^-$ then the residual processes must have the syntactical form:

$$P_0^+ = (\nu p_1^0 \ldots p_s^0)(\sigma^{-d}(n)[\,Q_0\,] \mid R_0)$$

$$P_d^- = (\nu p_1^d \ldots p_s^d)(\mathsf{mvin}\ \sigma^d(n).S_d \mid n[\,Q_d \mid S_0\,] \mid R_d)$$

In general the residues of the processes are the following:

$$P_h^+ = (\nu p_1^h \ldots p_s^h)(\sigma^{h-d}(n)[\,Q_h\,] \mid R_h)$$

$$P_h^- = (\nu p_1^h \ldots p_s^h)(\mathsf{mvin}\ \sigma^d(n).S_h \mid \sigma^{h-d}(n)[\,S_{\hat\sigma^{h-d}(0)} \mid Q_h\,] \mid R_h).$$

$$P_h^{+-} = P_h^{-+} = (\nu p_1^h \ldots p_s^h)(\sigma^{h-d}(n)[\,S_{\hat\sigma^{h-d}(0)} \mid Q_h\,] \mid R_h).$$

Now in order to show concretely how the computation goes around let $k - d = r$. Now we need to consider the relationship between r and

d. There are two cases to consider: $r \leq d$ or $d < r$. We analyse the case $r \leq d$ (the other case has been analysed in the similar proof of Lemma 5.6).

If $r \leq d$ then for some m we have $d - r = m$. Hence the steps of reduction among the different processes in the network proceed as follows:

$$P_1 \mid P_{d+1} \longrightarrow P_1^+ \mid P_{d+1}^-$$

$$P_2 \mid P_{d+2} \longrightarrow P_2^+ \mid P_{d+2}^-$$

$$\vdots$$

$$P_{r-1} \mid P_{d+(r-1)} \longrightarrow P_{r-1}^+ \mid P_{d+(r-1)}^-$$

$$P_r \mid P_0^+ \longrightarrow P_r^+ \mid P_0^{+-}$$

$$P_{r+1} \mid P_1^+ \longrightarrow P_{r+1}^+ \mid P_1^{+-}$$

$$\vdots$$

$$P_{r+(m-1)} \mid P_{m-1}^+ \longrightarrow P_{r+(m-1)}^+ \mid P_{m-1}^{+-}$$

$$P_d^- \mid P_m^+ \longrightarrow P_d^{-+} \mid P_m^{+-}$$

$$\vdots$$

$$P_{d+(r-1)}^- \mid P_{m+(r-1)}^+ \longrightarrow P_{d+(r-1)}^{-+} \mid P_{m+(r-1)}^{+-}$$

Then, we can think that the network is partitioned in this way:

$$\mathsf{N} = [P_0 \mid \cdots \mid P_r \mid \cdots \mid P_d \mid \cdots \mid P_{d+r}]$$

The computation of the network is the following:

$$\mathsf{N}^1: \qquad [P_0^+ \mid \cdots \mid P_r \mid \cdots \mid P_d^- \mid \cdots \mid P_{k-1}] \qquad \longrightarrow$$

$$\mathsf{N}^2: \quad [P_0^+ \mid P_1^+ \mid \cdots \mid P_r \mid \cdots \mid P_d^- \mid P_{d+1}^- \mid \cdots \mid P_{k-1}] \qquad \longrightarrow$$

$$\vdots$$

$$\mathsf{N}^r: \quad [P_0^{+-} \mid P_1^+ \mid \cdots \mid P_r^+ \mid \cdots \mid P_d^- \mid P_{d+1}^- \mid \cdots \mid P_{k-1}^-] \qquad \longrightarrow$$

$$\vdots$$

$$\mathsf{N}^k: [P_0^{+-} \mid P_1^{+-} \mid \cdots \mid P_r^{+-} \mid \cdots \mid P_d^{+-} \mid P_{d+1}^{-+} \mid \cdots \mid P_{k-1}^{-+}].$$

Hence we have $\mathcal{C}: \mathsf{N}^1 \longrightarrow \cdots \longrightarrow \mathsf{N}^k$.

(a) Now we have to show that the network N^k is symmetric. Now, N^k is symmetric with respect to σ, since for all s such that $0 \leq s \leq k-1$

the following holds:

$$P^{+-}_{\sigma^h(0)} = (\nu p^h_1 \dots p^h_s)(\sigma^{h-d}(n)[\, S_{h-d} \mid Q_h \,] \mid R_h)$$
$$= (\nu p^h_1 \dots p^h_s)(\sigma^{h-d}(n)[\, S_{\hat{\sigma}^{h-d}(0)} \mid Q_{\hat{\sigma}^h(0)} \,] \mid R_{\hat{\sigma}^h(0)})$$
$$= (\nu p^h_1 \dots p^h_s)(\sigma^{h-d}(n)[\, \sigma^h(S_{\hat{\sigma}^{-d}(0)}) \mid \sigma^h(Q_0) \,] \mid \sigma^h(R_0))$$
$$= \sigma^h((\nu p^0_1 \dots p^0_s)(\sigma^{-d}(n)[\, S_{\hat{\sigma}^{-d}(0)} \mid Q_0 \,] \mid R_0))$$
$$= \sigma^h(P^{+-}_0)$$

The automorphism σ has one orbit only since it has not changed.

(b) The proof for this part of the lemma is similar to that of Lemma 5.6.

(c) Observe that for this particular reduction rule the following holds: for all t, h such that $0 \le t, h \le k-1$, if $\mathsf{N}^{t+1} \downarrow \omega_h$ then $\mathsf{N}^1 \downarrow \omega_h$. By assumption, for all i such that $0 \le i \le k-1$ we have $\mathsf{N}^1 \not\downarrow \omega_i$. We conclude that for all t such that $1 \le t \le k$ the following holds $\mathsf{N}^t \not\downarrow \omega_i$.

(2) In the case in which the reduction is derived by means of the rule RED OPEN, then the proof is identical to Lemma 5.6.

(3) If the reduction is triggered by the rule RED A-COMM, this means the redex is formed with communication primitives, and then the case is identical to the proof supplied in Lemma 5.6 for the rule RED A-COMM.

□

Recall that $\mathsf{Obs}_k = \{\omega_0, \omega_1, \dots, \omega_{k-1}\}$.

Lemma C.4 *Let* $\mathsf{N} = [P_0 \mid \cdots \mid P_{k-1}]$ *with* $k \ge 2$ *be a symmetric network in* MA^{ob}. *Then there exists a computation* $\mathcal{C}$ *such that either* $\mathsf{Obs}(\mathcal{C}) \supseteq \mathsf{Obs}_k$ *or* $\mathsf{Obs}(\mathcal{C}) \cap \mathsf{Obs}_k = \emptyset$.

PROOF. The proof is identical to that of Lemma 5.7. □

Theorem 7.1 *Let* $\mathsf{N} = [P_0 \mid \cdots \mid P_{k-1}]$ *be a symmetric network in* MA^{ob} *with* $k \ge 2$. *Then* N *cannot be an electoral system.*

PROOF. The proof is identical to that of Theorem 5.1 by using Lemma C.4. □

D   Proofs of Propositions 8.1 and 8.2

In this section we prove Propositions 8.1 and 8.2.

Let $\longrightarrow_{\mathsf{tco}}$ denote the reduction relation on $\mathrm{MA}^{-\mathsf{out}}$ obtained from the usual rules of MA, but without the rules for the in capability and reduction inside an ambient (RED IN and RED AMB). This *ad hoc* notion allows top-level communication and open reductions, explaining the abbreviation "tco".

Definition D.1 *The reduction relation* $\longrightarrow_{\mathsf{tco}}$ *on* $\mathrm{MA}^{-\mathsf{out}}$ *is the least relation generated by the rules* RED OPEN, RED A-COMM, RED PAR, RED RESTR *and* RED CONG.

The key lemma we need to show Proposition 8.1 is the following:

Lemma D.2 *Let* P, Q *be* $\mathrm{MA}^{-\mathsf{out}}$ *processes. Suppose that* $P \not\longrightarrow_{\mathsf{tco}}$ *and* $P \longrightarrow Q$. *Then* $Q \not\longrightarrow_{\mathsf{tco}}$ *and for any name* n, *if* $Q \downarrow n$ *then* $P \downarrow n$.

To see informally why this holds, let P, Q be as in the statement of the lemma. The reduction $P \longrightarrow Q$ either (1) uses rule RED IN but not RED AMB (so that it is top-level) or (2) uses the rule RED AMB. In case (1) an ambient is removed from the top level, with no ambient added. Hence there are no new top-level redexes (and some may have been removed). So $Q \longrightarrow_{\mathsf{tco}}$ implies $P \longrightarrow_{\mathsf{tco}}$. Also, no new barbs are created (a barb may have been removed). In case (2) the reduction happens inside an ambient, and therefore the top level is unaffected. So $Q \longrightarrow_{\mathsf{tco}}$ iff $P \longrightarrow_{\mathsf{tco}}$.

In order to give a rigorous proof of Lemma D.2, first we must characterise when a process P can perform a $\longrightarrow_{\mathsf{tco}}$ transition in terms of the structure of P. We define a number of predicates which are similar to barbs. The first two predicates are relevant to whether a top-level communication can be performed. They express whether a process has a top-level unguarded input and output. Here "unguarded" means not guarded by a capability or an input. The third predicate expresses whether a process has a top-level unguarded open capability, while the final predicate concerns whether a process has both a top-level unguarded open and a matching top-level unguarded ambient (note that the name concerned may or may not be restricted).

Definition D.3 *Let* P *be an* $\mathrm{MA}^{-\mathsf{out}}$ *process.*

- $P \downarrow_{()}$ *iff* $P \equiv \nu\vec{m}\,((n).P' \mid P'')$.
- $P \downarrow_{\langle\rangle}$ *iff* $P \equiv \nu\vec{m}\,(\langle n\rangle \mid P')$.
- $P \downarrow_{\mathsf{open}\,n}$ *iff* $P \equiv \nu\vec{m}\,(\mathsf{open}\,n.P' \mid P'')$ *with* $n \notin \vec{m}$.
- $P \downarrow_{\mathsf{op}[\,]}$ *iff* $P \equiv \nu\vec{m}\,(\mathsf{open}\,n.P' \mid n[\,P''\,] \mid P''')$.

Using these predicates we can characterise $\longrightarrow_{\mathsf{tco}}$ -reduction:

Lemma D.4 *Let P be an $\mathrm{MA}^{-\mathsf{out}}$ process. Then $P \longrightarrow_{\mathsf{tco}}$ iff (1) $P \downarrow_{\mathsf{op[\,]}}$ or (2) both $P \downarrow_{()}$ and $P \downarrow_{\langle\rangle}$.*

PROOF. $(\Rightarrow)$ By induction on the derivation of $P \longrightarrow_{\mathsf{tco}}$.

$(\Leftarrow)$ Trivial. $\square$

Lemma D.5 *Let P, Q be $\mathrm{MA}^{-\mathsf{out}}$ processes and let n be a name. Then the following properties hold:*

(1) $(P \mid Q) \downarrow_{()}$ iff $P \downarrow_{()}$ or $Q \downarrow_{()}$.
(2) $(P \mid Q) \downarrow_{\langle\rangle}$ iff $P \downarrow_{\langle\rangle}$ or $Q \downarrow_{\langle\rangle}$.
(3) $(P \mid Q) \downarrow_{\mathsf{open}\,n}$ iff $P \downarrow_{\mathsf{open}\,n}$ or $Q \downarrow_{\mathsf{open}\,n}$.
(4) $(P \mid Q) \downarrow n$ iff $P \downarrow n$ or $Q \downarrow n$.
(5) $(P \mid Q) \downarrow_{\mathsf{op[\,]}}$ iff $P \downarrow_{\mathsf{op[\,]}}$ or $Q \downarrow_{\mathsf{op[\,]}}$ or there exists n such that either (1) $P \downarrow n$ and $Q \downarrow_{\mathsf{open}\,n}$ or (2) $P \downarrow_{\mathsf{open}\,n}$ and $Q \downarrow n$.

PROOF. These properties can be proved by showing the various predicates to be equivalent to versions defined by structural induction on processes. We omit the details. $\square$

Lemma D.6 *Let P be an $\mathrm{MA}^{-\mathsf{out}}$ process and let m, n be names. Then the following properties hold:*

(1) $(\nu m\, P) \downarrow_{()}$ iff $P \downarrow_{()}$.
(2) $(\nu m\, P) \downarrow_{\langle\rangle}$ iff $P \downarrow_{\langle\rangle}$.
(3) $(\nu m\, P) \downarrow_{\mathsf{open}\,n}$ iff $P \downarrow_{\mathsf{open}\,n}$ and $m \neq n$.
(4) $(\nu m\, P) \downarrow n$ iff $P \downarrow n$ and $m \neq n$.
(5) $(\nu m\, P) \downarrow_{\mathsf{op[\,]}}$ iff $P \downarrow_{\mathsf{op[\,]}}$.

PROOF. Similar to that of Lemma D.5. $\square$

Lemma D.7 *Let P, Q be $\mathrm{MA}^{-\mathsf{out}}$ processes. Suppose that $P \not\longrightarrow_{\mathsf{tco}}$ and $P \longrightarrow Q$. Then the following properties hold:*

(1) If $Q \downarrow_{()}$ then $P \downarrow_{()}$.
(2) If $Q \downarrow_{\langle\rangle}$ then $P \downarrow_{\langle\rangle}$.
(3) For any n, if $Q \downarrow_{\mathsf{open}\,n}$ then $P \downarrow_{\mathsf{open}\,n}$.
(4) For any n, if $Q \downarrow n$ then $P \downarrow n$.
(5) $Q \downarrow_{\mathsf{op[\,]}}$ is false (note that $P \downarrow_{\mathsf{op[\,]}}$ is false by Lemma D.4).

PROOF. By induction on the derivation of $P \longrightarrow Q$. There are seven rules which can be applied to derive $P \longrightarrow Q$ in MA$^{-\text{out}}$. Of these, RED OPEN and RED A-COMM can be ruled out, as we have $P \not\longrightarrow_{\text{tco}}$.

RED IN Suppose $P = n[\,\text{in}\,m.P_1 \mid P_2\,] \mid m[\,P_3\,] \longrightarrow m[\,n[\,P_1 \mid P_2\,] \mid P_3\,] = Q$. The five properties can be easily checked. Notice that Q only has an m barb, whereas P has both m and n barbs.

RED PAR Suppose $P = P_1 \mid P_2 \longrightarrow P_1' \mid P_2 = Q$ is deduced from $P_1 \longrightarrow P_1'$. The first four properties are straightforward using Lemma D.5. For the last property, suppose that $Q \downarrow_{\text{op}[\,]}$ holds. Looking at the possible cases in the characterisation of $(P_1' \mid P_2) \downarrow_{\text{op}[\,]}$ in Lemma D.5, we cannot have $P_1' \downarrow_{\text{op}[\,]}$, by the induction hypothesis. If $P_2 \downarrow_{\text{op}[\,]}$ then $P \downarrow_{\text{op}[\,]}$, which is false by Lemma D.4. If $P_1' \downarrow_{\text{open}\,n}$ and $P_2 \downarrow n$ then $P_1 \downarrow_{\text{open}\,n}$ by the induction hypothesis, and so again $P \downarrow_{\text{op}[\,]}$, which is false. If $P_1' \downarrow n$ and $P_2 \downarrow_{\text{open}\,n}$ then $P_1 \downarrow n$ by the induction hypothesis, and so again $P \downarrow_{\text{op}[\,]}$, which is false.

RED RESTR Straightforward using Lemma D.6.

RED AMB and RED CONG Straightforward.

$\square$

Now Lemma D.2 follows immediately from Lemmas D.4 and D.7.

Before proving Proposition 8.1 we need another lemma.

Lemma D.8 *Let* $k \geq 1$. *Let* $P = \nu\vec{m}\,(P_0 \mid \cdots \mid P_{k-1})$ *be an* MA$^{-\text{out}}$ *process. If* $P \longrightarrow_{\text{tco}}$ *then (1)* $P_i \longrightarrow_{\text{tco}}$ *for some* $i < k$, *or (2)* $P_i \downarrow_{\text{open}\,n}$ *and* $P_j \downarrow n$ *for some* n *and* $i, j < k$ *with* $i \neq j$, *or (3)* $P_i \downarrow_{()}$ *and* $P_j \downarrow_{\langle\rangle}$ *for some* $i, j < k$ *with* $i \neq j$.

PROOF. By Lemmas D.4, D.5 and D.6. $\square$

Proposition 8.1 For every $k \geq 2$, MA$^{-\text{out}}$ does not have a symmetric electoral system of size k.

PROOF. Let $k \geq 2$. Suppose that N is a symmetric electoral system in MA$^{-\text{out}}$ with size k. We shall construct a maximal computation $\mathcal{C}'$ with no observables (i.e. $\mathsf{Obs}(\mathcal{C}') \cap \mathsf{Obs}_k = \emptyset$), which will show that N cannot be an electoral system. Let $\mathcal{C}$ be the computation so far, and suppose that $\mathsf{Obs}(\mathcal{C}) \cap \mathsf{Obs}_k = \emptyset$. Let $P = \nu\vec{n}\,(P_0 \mid \cdots \mid P_{k-1})$ be the current state of the network, and suppose that P is symmetric with respect to a single-orbit automorphism σ. Without loss of generality suppose that $\hat{\sigma}(i) = i + 1 \mod k$ for all $i < k$.

We first exhaust all $\longrightarrow_{\mathsf{tco}}$ reductions. Suppose that $P \longrightarrow_{\mathsf{tco}}$. By Lemma D.8 we have (1) $P_i \longrightarrow_{\mathsf{tco}}$ for some $i < k$, or (2) $P_i \downarrow_{\mathsf{open}\, n_i}$ and $P_j \downarrow n_i$ for some n_i and $i, j < k$ with $i \neq j$, or (3) $P_i \downarrow_{()}$ and $P_j \downarrow_{\langle\rangle}$ for some $i, j < k$ with $i \neq j$. In each of the three cases P can perform k $\longrightarrow_{\mathsf{tco}}$ reductions to reach a new symmetric state P'. If any barb in Obs_k appears, then by symmetry all members of Obs_k appear, which contradicts N being an electoral system. Hence if $\mathcal{C}''$ is the computation up to P' then $\mathsf{Obs}(\mathcal{C}'') \cap \mathsf{Obs}_k = \emptyset$. The argument is very much as in the proof of Theorem 5.8, except that we have $\longrightarrow_{\mathsf{tco}}$ reductions instead of $\longrightarrow$ reductions.

If we never run out of $\longrightarrow_{\mathsf{tco}}$ reductions then we have a maximal computation with $\mathsf{Obs}(\mathcal{C}') \cap \mathsf{Obs}_k = \emptyset$, which is a contradiction. So suppose instead that by a computation $\mathcal{C}$ such that $\mathsf{Obs}(\mathcal{C}) \cap \mathsf{Obs}_k = \emptyset$ we reach a symmetric network P where $P \not\longrightarrow_{\mathsf{tco}}$. If $P \longrightarrow$ then we can continue the computation. But by Lemma D.2 no further $\longrightarrow_{\mathsf{tco}}$ reduction will be possible, and no new observables will appear. Hence any maximal extension of $\mathcal{C}$ to $\mathcal{C}'$ will have $\mathsf{Obs}(\mathcal{C}') \cap \mathsf{Obs}_k = \emptyset$, which is a contradiction. $\quad\square$

Proposition 8.2 For every $k \geq 2$, $\mathsf{SA}^{-\mathsf{out,open}}$ does not have a symmetric electoral system of size k.

PROOF. (Sketch) Suppose that we have a symmetric electoral system of size k. We follow the proof of Proposition 8.1 and construct a maximal computation with no observables, which is a contradiction. We start by confining ourselves to top-level communication reductions. We can do this while maintaining symmetry, as in Case (3) of the proof of Proposition 8.1. Therefore no observables can appear. If we reach a point where no top-level communication is possible, then any further reductions will not enable any further top-level communication, and also no new top-level ambients can appear, by the analogue of Lemma D.2. Hence any extension to a maximal computation will have no observables. $\quad\square$

References

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

[2] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge University Press, 1990.

[3] G. Boudol. Asynchrony and the π-calculus. Technical Report 1702, INRIA Sophia-Antipolis, 1992.

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

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

[6] N. Busi and G. Zavattaro. On the Expressive Power of Movement and Restriction in Pure Mobile Ambients. *Theoretical Computer Science*, 322(3):477–515, 2004.

[7] L. Cardelli. Mobility and security. In F.L. Bauer and R. Steinbrggen, editors, *Proceedings of the NATO Advanced Study Institute on Foundations of Secure Computation, Marktoberdorf, Germany, 27 July - 8 August 1999*, NATO Science Series, pages 3–37. IOS Press, 2000.

[8] L. Cardelli and A.D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, 2000.

[9] G. Castagna, J. Vitek, and F. Zappa Nardelli. The Seal calculus. *Information and Computation*, 201(1):1–54, 2005.

[10] T. Chothia and I. Stark. Encoding distributed areas and local communication into the pi-calculus. In *Proceedings of 8th International Workshop on Expressiveness in Concurrency (EXPRESS'01)*, volume 52 of *Electronic Notes in Theoretical Computer Science*, pages 101–119. Elsevier Science Publishers, 2002.

[11] C. Ene and T. Muntean. Expressiveness of point-to-point versus broadcast communications. In *Proceedings of 12th International Symposium on Fundamentals of Computation Theory (FCT'99)*, volume 1684 of *Lecture Notes in Computer Science*, pages 258–268. Springer-Verlag, 1999.

[12] C. Fournet, J.J. Lévy, and A. Schmitt. An asynchronous, distributed implementation of Mobile Ambients. In *Proceedings of IFIP International Conference on Theoretical Computer Science (TCS 2000)*, volume 1872 of *Lecture Notes in Computer Science*, pages 348–364. Springer-Verlag, 2000.

[13] M.C.B. Hennessy and J. Riely. Resource access control in systems of mobile agents. *Information and Computation*, 173(1):82–120, 2002.

[14] C.A.R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[15] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[16] K. Honda and M. Tokoro. An object calculus for asynchronous communication. In *Proceedings of European Conference on Object-Oriented Programming (ECOOP'91)*, volume 512 of *Lecture Notes in Computer Science*, pages 133–147. Springer-Verlag, 1991.

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

[18] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

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

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

[21] R. Milner. *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, 1999.

[22] R. Milner, J. Parrow, and D. Walker. A calculus for mobile processes, parts I and II. *Information and Computation*, 100(1):1–77, 1992.

[23] U. Nestmann. What is a 'good' encoding of guarded choice? *Information and Computation*, 156:287–319, 2000.

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

[25] J. Parrow. An introduction to the π-calculus. In *Handbook of Process Algebra*, chapter 8, pages 479–543. Elsevier, 2001.

[26] I.C.C. Phillips and M.G. Vigliotti. On reduction semantics for the Push and Pull Ambient Calculus. In *Proceedings of IFIP International Conference on Theoretical Computer Science (TCS 2002)*, pages 550–562. Kluwer, 2002.

[27] I.C.C. Phillips and M.G. Vigliotti. Electoral systems in ambient calculi. In *Proceedings of 7th International Conference on Foundations of Software Science and Computation Structures, FoSSaCS 2004*, volume 2987 of *Lecture Notes in Computer Science*, pages 408–422. Springer-Verlag, 2004.

[28] I.C.C. Phillips and M.G. Vigliotti. Leader election in rings of ambient processes. *Theoretical Computer Science*, 356(3):468–494, 2006.

[29] D. Sangiorgi and D. Walker. *The π-Calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[30] G. Tel. *Distributed Algorithms*. Cambridge University Press, 2000.

[31] M.G. Vigliotti. *Reduction Semantics for Ambient Calculi*. PhD thesis, Imperial College London, 2004.

[32] P. Zimmer. On the expressiveness of pure Safe Ambients. *Mathematical Structures in Computer Science*, 13(5):721–770, 2003.