# Comparing the Expressive Power of the Synchronous and the Asynchronous $\pi$-calculus*

Catuscia Palamidessi

DISI, Università di Genova, via Dodecaneso, 35, 16146 Genova, Italy

catuscia@disi.unige.it

**Abstract**

The Asynchronous $\pi$-calculus, as recently proposed by Boudol and, independently, by Honda and Tokoro, is a subset of the $\pi$-calculus which contains no explicit operators for choice and output-prefixing. The communication mechanism of this calculus, however, is powerful enough to simulate output-prefixing, as shown by Boudol, and input-guarded choice, as shown recently by Nestmann and Pierce. A natural question arises, then, whether or not it is possible to embed in it the full $\pi$-calculus. We show that this is not possible, i.e. there does not exist any uniform, parallel-preserving, translation from the $\pi$-calculus into the asynchronous $\pi$-calculus, up to any "reasonable" notion of equivalence. This result is based on the incapablity of the asynchronous $\pi$-calculus of breaking certain symmetries possibly present in the initial communication graph. By similar arguments, we prove a separation result between the $\pi$-calculus and CCS.

## 1 Introduction

Communication is one of the fundamental "ingredients" of concurrent and distributed computation. This mechanism can be of two kinds: synchronous and asynchronous. The first one is usually understood as *simultaneous* exchange of information between the two partners; an example of it, in "real life", is the telephone. The latter arises when the action of sending a message, and the action of receiving it, do not have to occur at the same time. An example of it is e.mail. Advantages and disadvantages of the one and the other method are easy to imagine: more efficient but more expensive the first, allowing for more independence the second, etc.

In the field of models for concurrency, it arises naturally the question whether these two mechanisms are equivalent; i.e., whether they can be implemented the one in the other. Actually, one direction is clear: asynchronous communication can be simulated by inserting between each pair of communicating agents a "queue" process (see for instance [JJH90]). The other direction, on the contrary, is not clear and researchers in the field seem to have radically different opinions about it.

The motivation for this work arises from the attempt of solving, or at least clarifying, this question. The initial guess of the author was that asynchronous communication is less powerful. This intuition is supported by the example of two people who try to take a common decision by using e.mail instead of telephone: If they act always in the same way, i.e. send at the same time identical mails and react in the same way to what they read, then an agreement might never be reached.

Since we were trying to show a separation result, it seemed convenient to study this problem in the framework of the $\pi$-calculus ([MPW92]). This is a synchronous paradigm, and a fragment of it has been presented recently as "asynchronous" ([Bou92, HT91]). We could thus work in a uniform context. But, more important, the $\pi$-calculus (and also its asynchronous subset) is one of the richest paradigm for concurrency introduced so far, hence a separation result in this context would be more significant.

The asynchronous $\pi$-calculus differs from the $\pi$-calculus for the lack of the choice and the output prefix operators. The underlying model of interaction among processes, however, is the same as in the $\pi$-calculus (handshaking). The reason why it is considered asynchronous is that, due to the lack of output prefix, an output action can only be written "in parallel" with other activities, thus it is not possible to control when it will actually be executed. From the point of view of the process in which such an action occurs, it amounts to the impossibility

*This work has been supported by the HCM project "EXPRESS".

Proceedings of the 24th Annual SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Paris, France, January 15–17, 1997.

of controlling when the message will actually be read by the receiver.

In recent years the interest in this asynchronous fragment has grown, in particular concerning the question of its expressiveness. Boudol has shown in [Bou92] that the lazy lambda calculus can still be encoded into it (as it is the case for the $\pi$-calculus). Honda and Tokoro, and independently Boudol, have shown that output prefix can be simulated ([HT91, Bou92]). Concerning choice, the local (or internal) kind can be easily encoded ([HT92]). More interestingly, it has been proved recently by Nestmann and Pierce that also input-guarded choice can be encoded ([NP96]). Note that this justifies the more recent presentations of asynchronous $\pi$-calculus, which include input-guarded choice as an explicit operator ([BS96, ACS96]).

The only question that remains open is whether the asynchronous $\pi$-calculus can simulate the output-guarded choice (or to be more precise, the mixed choice, i.e. the presence of both kinds of guards). In this work, we show that it is not possible. For proving this result, we use techniques from the field of Distributed Computing. In particular, we show that in symmetric networks, it is not possible, with the asynchronous $\pi$-calculus, to solve the leader election problem, i.e. to guarantee that all processes will reach a common agreement (elect the leader) in a finite amount of time. It is possible, on the contrary, to solve this problem with the full $\pi$-calculus.

The use of this technique has been inspired by the work of Bougé ([Bou88]), who has shown a similar separation result concerning the $CSP$ ([Hoa78]) and the fragment of $CSP$ with no output guards, $CSP_{in}$. The main difference is that the asynchronous $\pi$-calculus is a much richer language than $CSP_{in}$, hence our result is not a consequence of the result of Bougé. Some evidence of this is provided by the fact that a second result of Bougé, concerning the non-encodability of $CSP_{in}$ into its choice-free fragment, does not extend to the context of the $\pi$-calculus, as shown by the above mentioned result of Nestmann and Pierce. For a more extended and technical discussion about the relation with [Bou88] see the last section of this paper.

Another problem we consider is the question to what extent the $\pi$-calculus is more powerful than its "ancestor" CCS ([Mil89]). Also CCS can be seen as a subset of the $\pi$-calculus; the main difference is the presence, in the latter, of a mechanism of name passing, which allows to change dynamically the structure of the communication graph. By similar arguments as above (existence/non-existence of symmetric electoral systems) we show that this capability makes the $\pi$-calculus strictly more expressive than CCS.

The rest of the paper is organized as follows: next section recalls basic definitions. Section 3 reformulates in the setting of the $\pi$-calculus the notions of symmetric and electoral system. Section 4 shows the main result of the paper, i.e. the non-existence of symmetric electoral systems in the asynchronous $\pi$-calculus. Section 5 discusses existence of symmetric electoral systems for the synchronous case, i.e. the $\pi$-calculus and CCS. Section 6 interprets previous results as non-encodability results. Section 7 discusses related work and concludes.

## 2 Preliminaries

In this section we recall the definition of the $\pi$-calculus, the asynchronous $\pi$-calculus, and the notion of hypergraph, which will be used to represent the communication structure of a network of processes.

### 2.1 The $\pi$-calculus

Many variants of the $\pi$-calculus have been proposed. Here we basically follow the presentation given in [BS96, San95]. The main difference with the original version ([MPW92]) is the absence of the matching operator, and a construct for guarded choice instead of free choice.

Let $\mathcal{N}$ be a countable set of *names*, $x, y, \ldots$. The set of prefixes, $\alpha, \beta, \ldots$, and the set of $\pi$-calculus processes, $P, Q, \ldots$, are defined by the following syntax:

$$\begin{array}{llll} \textit{Prefixes} & \alpha & ::= & x(y) \mid \bar{x}y \mid \tau \\ \textit{Processes} & P & ::= & \sum_i \alpha_i.P_i \mid \nu x P \mid P|P \mid\ !P \end{array}$$

Prefixes represent the basic actions of processes: $x(y)$ is the *input* of the (formal) name $y$ from channel $x$; $\bar{x}y$ is the *output* of the name $y$ on channel $x$; $\tau$ stands for any silent (non-communication) action.

The process $\sum_i \alpha_i.P_i$ represents guarded (global) choice and it is usually assumed to be finite. We will use the abbreviations $\mathbf{0}$ (*inaction*) to represent the empty sum, $\alpha.P$ (*prefix*) to represent sum on one element only, and $P + Q$ for the binary sum. The symbols $\nu x$, $|$, and $!$ are the *restriction*, the *parallel*, and the *replication* operator, respectively.

The operators $\nu x$ and $y(x)$ are $x$-*binders*, i.e. in the processes $\nu x P$ and $y(x).P$ the occurrences of $x$ in $P$ are considered *bounded*, with the usual rules of scoping. The *free names* of $P$, i.e. those names which do not occur in the scope of any binder, are denoted by $fn(P)$. The *alpha-conversion* of bounded names is defined as usual, and the renaming (or substitution) $P\{y/x\}$ is defined as the result of replacing all occurrences of $x$ in $P$ by $y$, possibly applying alpha-conversion to avoid capture.

The operational semantics is specified via a transition system labeled by *actions* $\mu, \mu' \ldots$. These are given by the following grammar:

$$\textit{Actions} \quad \mu \ ::= \ x(y) \ | \ \bar{x}y \ | \ \bar{x}(y) \ | \ \tau$$

Essentially, we have all the actions corresponding to prefixes, plus the *bounded output* $\bar{x}(y)$. This is introduced to model *scope extrusion*, i.e. the result of sending to another process a private ($\nu$-bounded) name. The bounded names of an action $\mu$, $bn(\mu)$, are defined as follows: $bn(x(y)) = bn(\bar{x}(y)) = \{y\}$; $bn(\bar{x}y) = bn(\tau) = \emptyset$. Furthermore, we will indicate by $n(\mu)$ all the *names* which occur in $\mu$.

In literature there have been considered two definitions for the transition system of the $\pi$-calculus, which induce two different semantics: the *early* and the *late* bisimulation semantics. Here we choose to present the first one because the early bisimulation is coarser than the other, but it should be noted that the results of this paper are independent from the bisimulation semantics adopted at this point. (No notion of bisimulation can identify an electoral system and a non-electoral one.)

The rules for the early semantics are given in Table 1. The symbol $\equiv$ used in Rule Cong stands for *structural congruence*, a form of equivalence which identifies "statically" two processes. Again, there are several definition of this relation in literature. For our purposes we do not need a very rich notion, we will just use it to simplify the presentation. Hence we only assume this congruence to satisfy the following:

(i) $P \equiv Q$ if $Q$ can be obtained from $P$ by alpha-renaming, notation $P \equiv_\alpha Q$,

(ii) $P|Q \equiv Q|P$,

(iii) $(P|Q)|R \equiv P|(Q|R)$,

(iv) $(\nu x P)|Q \equiv \nu x(P|Q)$ if $x \notin fv(Q)$.

| | | |
|---|---|---|
| I-Sum | $\sum_i \alpha_i.P_i \xrightarrow{x(z)} P_j\{z/y\}$ | $\alpha_j = x(y)$ |
| O/$\tau$-Sum | $\sum_i \alpha_i.P_i \xrightarrow{\alpha_j} P_j$ | $\alpha_j = \bar{x}y$ or $\alpha_j = \tau$ |
| Open | $\dfrac{P \xrightarrow{\bar{x}y} P'}{\nu y P \xrightarrow{\bar{x}(y)} P'}$ | $x \neq y$ |
| Res | $\dfrac{P \xrightarrow{\mu} P'}{\nu y P \xrightarrow{\mu} \nu y P'}$ | $y \notin n(\mu)$ |
| Par | $\dfrac{P \xrightarrow{\mu} P'}{P\|Q \xrightarrow{\mu} P'\|Q}$ | $bn(\mu) \cap fn(Q) = \emptyset$ |
| Com | $\dfrac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}y} Q'}{P\|Q \xrightarrow{\tau} P'\|Q'}$ | |
| Close | $\dfrac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}(y)} Q'}{P\|Q \xrightarrow{\tau} \nu y(P'\|Q')}$ | |
| Rep | $\dfrac{P\|!P \xrightarrow{\mu} P'}{!P \xrightarrow{\mu} P'}$ | |
| Cong | $\dfrac{P \equiv P' \quad P' \xrightarrow{\mu} Q' \quad Q' \equiv Q}{P \xrightarrow{\mu} Q}$ | |

Table 1: The early-instantiation transition system of the $\pi$-calculus.

## 2.2 The asynchronous $\pi$-calculus

In accordance with [HT91, Bou92], we consider the following definition of the asynchronous $\pi$-calculus ($\pi_a$-calculus for short).

$$\textit{Processes} \quad P \ ::= \ \bar{x}y \ | \ x(y).P \ | \ \nu x P \ | \ P|P \ | \ !P$$

The difference wrt the $\pi$-calculus is that $\sum_i \alpha_i.P_i$ is replaced by the output-action process $\bar{x}y$ and by the input-prefix process $x(y).P$. The rule for the output-action process is described in Table 2, where $\mathbf{0}$ stands again for inaction (see [Bou92] for the encoding of inaction into the $\pi_a$-calculus.) All the rules for the other operators are like in Table 1.

Note that the $\pi_a$-calculus is a proper subset of the $\pi$-calculus. The output-action process $\bar{x}y$, in fact, could be equivalently replaced by the special case of output prefix $\bar{x}y.\mathbf{0}$.

| | |
|---|---|
| Out | $\bar{x}y \xrightarrow{\bar{x}y} \mathbf{0}$ |

Table 2: The output rule for the $\pi_a$-calculus.

### 2.3 Hypergraphs and automorphisms

In this section we recall the definition of *hypergraph*, which generalize the concept of graph essentially by allowing an arc to connect more than two nodes.

A hypergraph is a pair $H = \langle N, X, t\rangle$ where $N, X$ are finite sets whose elements are called *nodes* and (*hyper*)*arcs* respectively, and $t$ (*type*) is a function which assigns to each $x \in X$ a set of nodes, representing the nodes *connected* by $x$. We will also use the notation $x : n_1, \ldots, n_k$ to indicate $t(x) = \{n_1, \ldots, n_k\}$.

The concept of graph automorphism extends naturally to hypergraphs: Given a hypergraph $H = \langle N, X, t\rangle$, an *automorphism* on $H$ is a pair $\sigma = \langle \sigma_N, \sigma_X\rangle$ such that $\sigma_N : N \to N$ and $\sigma_X : X \to X$ are permutations which preserve the type of arcs, namely for each $x \in X$, if $x : n_1, \ldots, n_k$, then $\sigma_X(x) : \sigma_N(n_1), \ldots, \sigma_N(n_k)$.

It is easy to see that the *composition* of automorphisms, defined componentwise as $\sigma \circ \sigma' = \langle \sigma_N \circ \sigma'_N, \sigma_X \circ \sigma'_X\rangle$, is still an automorphism. Its identity is the pair of identity functions on $N$ and $X$, i.e. $id = \langle id_N, id_X\rangle$. It is easy to show that the set of automorphisms on $H$ with the composition forms a group.

Given $H$ and $\sigma$ as above, the *orbit* of $n \in N$ generated by $\sigma$ is defined as the set of nodes in which the various iterations of $\sigma$ map $n$, namely:

$$O_\sigma(n) = \{n, \sigma(n), \sigma^2(n), \ldots, \sigma^{h-1}(n)\}$$

where $\sigma^i$ represents the composition of $\sigma$ with itself $i$ times, and $\sigma^h = id$. It is possible to show that the orbits generated by $\sigma$ constitute a partition of $N$.

## 3 Electoral and Symmetric systems

In this section we adapt to the $\pi$-calculus (a simplified version of) the notions of electoral system and symmetric network as given by Bougé in [Bou88].

### 3.1 Election of a leader in a network

We first need to define the concepts of *network computation* and its *projection* over a component of the network. A network is a system of parallel process $P = P_1|P_2|\ldots|P_k$. A computation $C$ for this system is a (possibly $\omega$-infinite) sequence of transitions[1]

$$\begin{array}{rcl} P_1|P_2|\ldots|P_k & \xrightarrow{\mu^0} & P_1^1|P_2^1|\ldots|P_k^1 \\ & \xrightarrow{\mu^1} & P_1^2|P_2^2|\ldots|P_k^2 \\ & \vdots & \\ & \xrightarrow{\mu^{n-1}} & P_1^n|P_2^n|\ldots|P_k^n \\ & (\xrightarrow{\mu^n} & \ldots) \end{array}$$

with $n \geq 0$. We will represent it also by $C : P \stackrel{\tilde{\mu}}{\Longrightarrow} P^n$ (by $C : P \stackrel{\tilde{\mu}}{\Longrightarrow}$ if it is infinite), $\tilde{\mu}$ being the sequence $\mu^0\mu^1\ldots\mu^{n-1}(\mu^n\ldots)$, and $P^n$ being the process $P_1^n|P_2^n|\ldots|P_k^n$. The relation $C \preceq C'$ ($C'$ *extends* $C$) is defined as usual. Namely, let $C : P \stackrel{\tilde{\mu}}{\Longrightarrow} P^n$. Then $C \preceq C'$ iff there exists $C'' : P^n \stackrel{\tilde{\mu}'}{\Longrightarrow} P^{n+n'}$ or $C'' : P^n \stackrel{\tilde{\mu}'}{\Longrightarrow}$, and $C' = CC''$ (identifying the two occurrences of $P^n$). We will denote by $C' \setminus C$ the continuation $C''$. The notation $C \prec C'$ will indicate that $C'$ is a *strict extension* of $C$. Note that if $C$ is infinite then it cannot be strictly extended, because we admit only $\omega$-infinite (i.e. not transfinite) computations.

Given $P$ and $C$ as above, the projection of $C$ over $P_i$, $Proj(C, P_i)$[2] is defined as the "contribution" of $P_i$ to the computation. More formally, $Proj(C, P_i)$ is the computation

$$P_i \stackrel{\tilde{\mu}^0}{\Longrightarrow} P_i^1 \stackrel{\tilde{\mu}^1}{\Longrightarrow} P_i^2 \stackrel{\tilde{\mu}^2}{\Longrightarrow} \ldots \stackrel{\tilde{\mu}^{n-1}}{\Longrightarrow} P_i^n \;(\stackrel{\tilde{\mu}^n}{\Longrightarrow} \ldots)$$

where, depending on the application of the rule (Par, Com, or Close) which generate the $m+1$-th transition of $C$, $P_i^m \stackrel{\tilde{\mu}^m}{\Longrightarrow} P_i^{m+1}$ is:

- $P_i^m \xrightarrow{\mu^m} P_i^{m+1}$, if the rule is Par with this transition as premise,
- $P_i^m \xrightarrow{\alpha} P_i^{m+1}$, if the rule is Com or Close and this transition is one of the two premises.
- empty (and therefore $P_i^m = P_i^{m+1}$ and $\tilde{\mu}^m$ is empty) if, in the $m+1$-th transition of $C$, $P_i^m$ is idle, i.e. it does not appear in the premises of the rule.

To give the definition of electoral system, we assume the existence of a special output channel name, $o$, shared

[1] For the sake of keeping the notation simple, we assume that each binder $\nu x$ generated by a possible application of the Close rule, is pushed "to the top level" by repeated applications of the properties $(i)$ and $(iv)$ of $\equiv$. Furthermore, we do not represent explicitly the binders at the top level; we just assume that the network will never perform a visible action on one of the names restricted by those binders.

[2] For the sake of brevity here we have introduced an abuse of notation: the projection is not a function of $C$, but of the sequence of proof-trees which generate $C$.

by all processes. Furthermore we assume that $\mathcal{N}$ contains the natural numbers, which will represent the identifier of processes in a network.

Intuitively, an electoral system has the property that at each possible run the processes will agree sooner or later on "which of them has to be the leader", and will communicate this decision to the "external world" by using the channel $o$.

**Definition 3.1 (Electoral system)** A process $P = P_1|P_2|\ldots|P_k$ is an electoral system if for every computation $C$ for $P$ there exists an extension $C'$ of $C$ and there exists $n \in \{1,\ldots,k\}$ (the "leader") such that for each $i \in \{1,\ldots,k\}$ the projection $\mathit{Proj}(C',i)$ contains one output action of the form $\bar{o}n$, and no extension of $C'$ contain any other action of the form $\bar{o}m$, with $m \neq n$.

Note that for such a system an infinite computation $C$ must contain already all the output actions of each process because $C$ cannot be strictly extended.

### 3.2 Symmetric networks

In order to define the notion of *symmetric network*, we have to consider its initial communication structure, which we will represent as an hypergraph. Intuitively, the nodes represent the processes, and the arcs the free communication channels, connecting the nodes which share them. It will be convenient, although not necessary, not to consider as an arc the"channel to the external world" $o$.

**Definition 3.2 (Hypergraph associated to a network)** Given a network $P = P_1|P_2|\ldots|P_k$, the hypergraph associated to $P$ is $H(P) = \langle N, X, t\rangle$ with $N = \{1,\ldots,k\}$, $X = \mathit{fn}(P)\setminus\{o\}$, and for each $x \in X$, $t(x) = \{n | x \in \mathit{fn}(P_n)\}$.

Intuitively, a system $P$ is symmetric with respect to an automorphism $\sigma$ on $H(P)$ iff for each $i$

> *the process associated to the node $\sigma(i)$ is identical (modulo alpha-conversion) to the process obtained by $\sigma$-renaming the process associated to the node $i$.*

The notion of $\sigma$-renaming is the obvious extension of the standard notion of renaming (see the preliminaries). More formally, given a process $Q$, first apply alpha-conversion so to rename all bounded names into fresh ones, extend $\sigma$ to be the identity on these new names, and define $\sigma(Q)$ by structural induction as indicated below. For the sake of simplicity, here we use $\sigma(\cdot)$ to represent both $\sigma_N(\cdot)$ and $\sigma_X(\cdot)$. Furthermore we extend $\sigma$ on prefixes in the obvious way, i.e. $\sigma(x(y)) = \sigma(x)(\sigma(y))$, $\sigma(\bar{x}y) = \overline{\sigma(x)}\sigma(y)$, and $\sigma(\tau) = \tau$.

$$\begin{aligned}
\sigma(\textstyle\sum_i \alpha_i.P_i) &= \textstyle\sum_i \sigma(\alpha_i).\sigma(P_i)\\
\sigma(\nu x P) &= \nu x\; \sigma(P)\\
\sigma(P|Q) &= \sigma(P)|\sigma(Q)\\
\sigma(!P) &= !\sigma(P)
\end{aligned}$$

We are now ready to give the formal definition of symmetric system:

**Definition 3.3 (Symmetric system)** Consider a network $P = P_1|P_2|\ldots|P_k$, and let $\sigma$ be an isomorphism on its associated hypergraph $H(P) = \langle N, X, t\rangle$. We say that $P$ is *symmetric wrt* $\sigma$ iff for each node $i \in N$, $P_{\sigma(i)} \equiv_\alpha \sigma(P_i)$ holds; $P$ is *symmetric* if it is symmetric wrt all the automorphisms on $H(P)$.

Note that if $P$ is symmetric wrt $\sigma$ then $P$ is symmetric wrt all the powers of $\sigma$.

## 4 Symmetric electoral systems: the asynchronous case

This section contains the main result of the paper, which is that, for certain communication graphs, it is not possible to write in $\pi_a$-calculus a symmetric network solving the election problem.

We first need to show that the $\pi_a$-calculus enjoyes a certain kind of *confluence* property:

**Lemma 4.1** *Let $P$ be a process of the $\pi_a$-calculus. Assume that $P$ can make two transitions $P \xrightarrow{\mu} Q$ and $P \xrightarrow{\mu'} Q'$, where $\mu$ is an output action while $\mu'$ is an input action. Then there exists $R$ such that $Q \xrightarrow{\mu'} R$ and $Q' \xrightarrow{\mu} R$.*

**Proof** Assume that $\mu$ is of the form $\bar{x}y$ or $\bar{x}(y)$, and that $\mu'$ is of the form $z(w)$. Observe that $x, y, z$ must be free names in $P$. The rule which has produced the $\mu$-transition can be only Out, Open, Res, Par, Rep, or Cong. In the last (five) cases the assumption is again a $\mu$-transition. By repeating this reasoning (descending the tree), we must arrive to a leaf of the form $\bar{x}y \xrightarrow{\bar{x}y} \mathbf{0}$. Analogously, by descending the tree for the $\mu'$-transition we must arrive to a leaf of the form $z(w).S \xrightarrow{z(w')} S\{w'/w\}$. Now, $\bar{x}y$ and $z(w).S$ must be two parallel processes in $P$, i.e. there must be a subprocess in $P$ of the form $T[\bar{x}y]|T'[z(w).S]$ (modulo $\equiv$), i.e. $P \equiv U[T[\bar{x}y]|T'[z(w).S]]$ (here $T[\,]$, $T'[\,]$ and $U[\,]$ represent contexts, with the usual definition). Furthermore, the $\mu$ and $\mu'$ transitions must have been obtained by the application of the rule PAR to this subprocess, i.e. $Q \equiv U[T[\mathbf{0}]|T'[z(w).S]]$ and $Q' \equiv U[T[\bar{x}y]|T'[S\{w'/w\}]]$. By applying again the rule PAR (plus all the other rules in the trees for the $\mu'$ and the $\mu$ transition respectively) we obtain the transitions $Q \xrightarrow{\mu'} U[T[\mathbf{0}]|T'[S\{w'/w\}]]$ and $Q' \xrightarrow{\mu} U[T[\mathbf{0}]|T'[S\{w'/w\}]]$. □

We are now ready to prove the announced non-existence result. The intuition is the following: In the attempt to reach an agreement about the leader, the processes of a symmetric network have to "break the initial symmetry", and therefore have to communicate. The first such communication, however, can be repeated, by the above lemma, and by symmetry, by all the pair of processes of the network. The result of all these transitions will still lead to a symmetric situation. Thus there is a (infinite) computation in which the processes never succeed to break the symmetry, which means no leader is elected.

**Theorem 4.2** *Consider a network $P = P_1|P_2|\ldots|P_k$ in the $\pi_a$-calculus, and assume that the associated hypergraph $H(P)$ admits an automorphism $\sigma \neq id$ with only one orbit, and that $P$ is symmetric wrt $\sigma$. Then $P$ cannot be an electoral system.*

**Proof** Assume by contradiction that $P$ is an electoral system. We will show that we can then construct an infinite increasing sequence of computations for $P$, $C_0 \prec C_1 \prec \ldots \prec C_h \ldots$, such that for each $j$, $C_j : P \overset{\tilde{\mu}^j}{\Longrightarrow} P^j$ does not contain any output action on $o$, and $P^j$is still symmetric wrt $\sigma_j$, where $\sigma^j$ is the original authomorphism enriched with associations on the new names possibly introduced by the communication actions (for simplicity of notation, in the following $\sigma_j$ will still be indicated as $\sigma$). This gives a contradiction, because the limit of this sequence is an infinite computation for $P$ which does not contain any output action on $o$.

We prove the above by induction wrt $h$. In order to understand the proof, it is important to notice that the hypothesis of $\sigma$ generating only one orbit implies that for each $i \in \{1, 2, \ldots, k\}$, $O_\sigma(i) = \{i, \sigma(i), \ldots, \sigma^{k-1}(i)\} = \{1, 2, \ldots, k\}$.

$h = 0$) Define $C_0$ to be the empty computation.

$h+1$) Given $C_h : P \overset{\tilde{\mu}^h}{\Longrightarrow} P^h$, we construct $C_{h+1} : P \overset{\tilde{\mu}^{h+1}}{\Longrightarrow} P^{h+1}$ as follows.

Since $P$ is an electoral system, it must be possible to extend $C_h$ to a computation $C$ which contains $(k)$ actions $\bar{o}n$, for a particular $n \in \{1, \ldots, k\}$. Observe that the first action $\mu$ of $C \setminus C_h$ cannot be $\bar{o}n$. Otherwise, let $P_i^h$ be the component which performs this action. Then $P_i^h$ must contain the subprocess $\bar{o}n$ and must have no restriction on $n$. By symmetry, $P_\sigma^h(i) \equiv \sigma(P_i^h)$ must contain the subprocess $\bar{o}\sigma(n)$ and have no restriction on $\sigma(n)$. Hence there must be an extension of $C$ where the action $\bar{o}\sigma(n)$ occurs. This implies (for the hypothesis that $P$ is an electoral system), that $\sigma(n) = n$, and, since $\sigma$ generates only one orbit, that $\sigma = id$ (and $k = 1$). Contradiction.

Hence, $\mu$ must be either $\tau$ or an action on a channel different from $o$. Let us distinguishes the two cases.

$\mu \neq \tau$) Let $P_i^h$ be the component which performs this action. Let $P_i^{h+1}$ be such that

$$P_i^h \xrightarrow{\mu} P_i^{h+1}$$

By symmetry we also have

$$\begin{array}{ccc} P^h_{\sigma(i)} & \xrightarrow{\sigma(\mu)} & P^{h+1}_{\sigma(i)} \\ P^h_{\sigma^2(i)} & \xrightarrow{\sigma^2(\mu)} & P^{h+1}_{\sigma^2(i)} \\ & \vdots & \\ P^h_{\sigma^{k-1}(i)} & \xrightarrow{\sigma^{k-1}(\mu)} & P^{h+1}_{\sigma^{k-1}(i)} \end{array}$$

Since $\sigma$ generates only one orbit, $P^h \equiv P_i^h \mid P^h_{\sigma(i)} \mid P^h_{\sigma^2(i)} \mid \ldots \mid P^h_{\sigma^{k-1}(i)}$. Hence we can compose the displayed transitions into a computation

$$P^h \overset{\tilde{\mu}}{\Longrightarrow} P^{h+1},$$

where $\tilde{\mu} = \mu\sigma(\mu)\sigma^2(\mu)\ldots\sigma^{k-1}(\mu)$ and $P^{h+1} \equiv P_i^{h+1} \mid P^{h+1}_{\sigma(i)} \mid P^{h+1}_{\sigma^2(i)} \mid \ldots \mid P^{h+1}_{\sigma^{k-1}(i)}$. Finally, observe that $P^{h+1}$ is still symmetric.

$\mu = \tau$) In this case, the transition is the result of a communication between two agents. The interesting case is when the two agents are in *different nodes* of the communication graph. (If the agents are inside the same node, say $P_i^h$, then we have a transition $P_i^h \xrightarrow{\tau} P_i^{h+1}$ and we proceed like in previous case.) Let $P_i^h$ and $P_j^h$ be the two processes, with $i \neq j$. We have two transitions $P_i^h \xrightarrow{\mu_i} Q_i$ and $P_j^h \xrightarrow{\mu_j} R_j$, where $\mu_i$ and $\mu_j$ are complementary. Assume without loss of generality that $\mu_i$ is the input action, and $\mu_j$ is the output action. Since $\sigma$ generates only one orbit, there exists $r \in \{1, \ldots, k-1\}$ such that $j = \sigma^r(i)$. Assume for simplicity that $r$ and $k$ are relatively prime[3], and let $\theta = \sigma^r$. Then $P_j^h = P^h_{\theta(i)}$ and $R_j = R_{\theta(i)}$. Let us first consider the case in which the first step of $C \setminus C_h$ has been produced by an application of the COM rule. Then we have a transition

$$P_i^h \mid P^h_{\theta(i)} \xrightarrow{\tau} Q_i \mid R_{\theta(i)}$$

[3] If they are not, then in the rest of the proof $k$ has to be replaced by the least $p$ such that $pk = rq$, for some $q$.

By symmetry, we have that $P^h_{\theta(i)} \xrightarrow{\theta(\mu_i)} \theta(Q_i)$. By Lemma 4.1 we then have the transitions $R_{\theta(i)} \xrightarrow{\theta(\mu_i)} R'$ and $\theta(Q_i) \xrightarrow{\mu_j} R'$ for some $R'$. Let us define $P^{h+1}_{\theta(i)} = R'$. By symmetry, we also have $P^h_{\theta^2(i)} \equiv P^h_{\theta(j)} \xrightarrow{\theta(\mu_j)} \theta(R_j)$, and $\theta(\mu_i)$, $\theta(\mu_j)$ are complementary, hence we can combine them into a transition

$$R_{\theta(i)} \mid P^h_{\theta^2(i)} \;\xrightarrow{\tau}\; P^{h+1}_{\theta(i)} \mid R_{\theta^2(i)}$$

with $R_{\theta^2(i)} = \theta(R_j)$. By repeatedly applying this reasoning, we obtain

$$\begin{array}{rcl} R_{\theta^2(i)} \mid P^h_{\theta^3(i)} & \xrightarrow{\tau} & P^{h+1}_{\theta^2(i)} \mid R_{\theta^3(i)} \\ & \vdots & \\ R_{\theta^{k-2}(i)} \mid P^h_{\theta^{k-1}(i)} & \xrightarrow{\tau} & P^{h+1}_{\theta^{k-2}(i)} \mid R_{\theta^{k-1}(i)} \end{array}$$

and $R_{\theta^{k-1}(i)} \xrightarrow{\theta^{k-1}(\mu_i)} P^{h+1}_{\theta^{k-1}(i)}$. Finally, observe that from the transition $\theta(Q_i) \xrightarrow{\mu_j} R'$ above we can derive $\theta^k(Q_i) \xrightarrow{\theta^{k-1}(\mu_j)} \theta^{k-1}(R')$. But $\theta^k = \sigma^{kr} = id$, hence we have $Q_i \xrightarrow{\theta^k(\mu_j)} P^{h+1}_i$, where we have defined $P^{h+1}_i$ to be $\theta^{k-1}(R')$. Therefore we can compose also these transitions, thus "closing the circle", as we obtain

$$R_{\theta^{k-1}(i)} \mid Q_i \;\xrightarrow{\tau}\; P^{h+1}_{\theta^{k-1}(i)} \mid P^{h+1}_i$$

The composition of the displayed transitions gives us the intended continuation[4]:

$$\begin{array}{l} P^h \equiv P^h_i | P^h_{\theta(i)} | \ldots | P^h_{\theta^{k-1}(i)} \\ \qquad\qquad \stackrel{\tilde{\tau}}{\Longrightarrow} \\ \qquad\qquad\qquad P^{h+1}_i | P^{h+1}_{\theta(i)} | \ldots | P^{h+1}_{\theta^{k-1}(i)} \end{array}$$

Finally define $P^{h+1} = P^{h+1}_i | P^{h+1}_{\theta(i)} | \ldots | P^{h+1}_{\theta^{k-1}(i)}$ and observe that it is still symmetric with respect to $\sigma$.

Consider now the case in which the first step of $C \setminus C_h$ is obtained by an application of the Close rule. Then the transition would be of the form

$$P^h_i \mid P^h_{\theta(i)} \;\xrightarrow{\tau}\; \nu y(Q_i \mid R_{\theta(i)})$$

where $y$ is the name transmitted in the communication. In order to reason as before we have to eliminate the $\nu y$ interposed between $Q_i \mid R_{\theta(i)}$ and the rest of the network. This can be done by applying $\alpha$-conversion and scope extrusion (Rules $(i)$ and $(iv)$ of $\equiv$), so to push the restriction operator at the top-level of the network. However, by doing this, we add new (free) names and enrich the communication structure of the network. To preserve the simmetry, we must then dynamically enrich $\sigma$ with suitable associations among these new names, in the obvious way. For instance, if a communication action occurs between the node $i$ and the node $j$, in which a private name $x$ of $i$ is transmitted, then an analogous communication will happen between the nodes $\sigma(i)$ and $\sigma(j)$, with transmission of another private name (of $\sigma(i)$), say $y$. Correspondingly, we must add the association $\sigma(x) = y$. □

[4] Under the assumption that $r$ and $k$ are relatively prime, also $\theta$ has only one orbit. If we drop this assumption, and hence we replace $k$ by the smallest $p$ such that $pk = rq$ for some $q$, then the computation we have constructed involves only the processes of the nodes in $O_\theta(i) = \{i, \theta(i), \ldots, \theta^{p-1}(i)\}$. To complete computation we have to repeat the reasoning for the other orbits of $\theta$: $O_\theta(\sigma(i))$, $O_\theta(\sigma^2(i)) \ldots O_\theta(\sigma^{q-1}(i))$.

Note that, for the above result, we could have considered a simpler (more permissive) notion of electoral system, obtained by requiring, in Definition 3.1, that $C'$ contains one (or more) actions of the form $\bar{o}n$, instead of requiring it for all the projections of $C'$. We have defined the electoral system in that way only to remain closer to the notion in literature.

In [Bou88] a more permissive notion of symmetry is considered for proving negative results. Namely, the automorphism $\sigma$ can have more orbits, provided that they all have the same cardinality. An automorphism with this property is called *well-balanced*. In the framework of [Bou88] this is a significant generalization, because the language considered there, $CSP_{in}$, can have the parallel operator only at the top level. Hence the condition of a single orbit, there, would impose that *all* the parallel processes present in the network have the same code (modulo renaming).

In our framework, on the contrary, we do not have this restriction, and the above mentioned generalization is not essential. In fact, we can easily extend Theorem 4.2 to well-balanced automorphisms:

**Corollary 4.3** *Consider a network $P = P_1|P_2|\ldots|P_k$ in the $\pi_a$-calculus, and assume that the associated hypergraph $H(P)$ admits a well-balanced automorphism $\sigma \neq id$, and that $P$ is symmetric wrt $\sigma$. Then $P$ cannot be an electoral system.*

**Proof** Assume that $\sigma$ generates $p$ orbits of cardinality $q$, and let $i_1, i_2, \ldots, i_p$ be arbitrary nodes from these orbits. Consider the processes

$$\begin{array}{rcl} Q_1 & = & P_{i_1}|P_{i_2}|\ldots|P_{i_p} \\ Q_2 & = & P_{\sigma(i_1)}|P_{\sigma(i_2)}|\ldots|P_{\sigma(i_p)} \\ & \vdots & \\ Q_q & = & P_{\sigma^{q-1}(i_1)}|P_{\sigma^{q-1}(i_2)}|\ldots|P_{\sigma^{q-1}(i_p)} \end{array}$$

Consider now the network $Q = Q_1|Q_2|\ldots|Q_q$. Clearly $Q \equiv P$, but the associated hypergraph, $H(Q)$, is different. More precisely, $H(Q)$ is "an abstraction" of $H(P)$ in the sense that certain nodes of $H(P)$ are "grouped together" in the same node of $H(Q)$. (The way this grouping is done depends on the choice of $i_1, i_2, \ldots, i_p$ and it is inessential for this proof.) The arcs $X$ of $H(P)$ are the same as the ones of $H(Q)$; the type function is the obvious one.

Now, consider the pair $\theta = \langle \sigma_N, \sigma_X \rangle$ with $\theta_N(1) = 2$, $\theta_N(2) = 3, \ldots, \theta_N(q) = 1$, and $\theta_X = \sigma_X$. It is easy to see that $\theta$ is a well balanced automorphism on $H(Q)$, and that $Q$ is symmetric wrt $\theta$. Then apply Theorem 4.2, and consider that a leader in $P$ determines immediately a leader in $Q$. □

## 5 Symmetric electoral systems: the synchronous case

In the (synchronous) $\pi$-calculus, the guarded choice construct makes it possible to establish a simultaneous agreement among two processes, thus breaking the symmetry. The point is that the presence of choice invalidates the confluence property of Lemma 4.1.

Consider for example the election problem in a symmetric network consisting of two nodes $P_0$ and $P_1$ only, and two arcs, $x_0$ and $x_1$, connecting them. A $\pi$-calculus specification which solves the problem is:

$$\begin{array}{rcl} P_i & :: & \overline{x_i}(y).\bar{o}i \\ & & + \\ & & x_{i\oplus 1}(y).\bar{o}(i \oplus 1) \end{array}$$

with $i \in \{1, 2\}$ and $\oplus$ being the binary sum.

The following results shows that with the $\pi$-calculus the existence of symmetric electoral systems is guaranteed in a large number of cases:

**Theorem 5.1** *Let $H$ be a connected hypergraph (i.e. each pair of nodes are connected by a sequence of arcs). Then there exists a symmetric electoral system $P$, in the $\pi$ calculus, such that $H(P) = H$.*

**Proof** (Hint) One possible algorithm is the following. Let $k$ be the number of nodes. The generic process $P_i$:

1. Broadcasts a private name $x_i$ to all the other processes (which is possible thanks to the connectivity hypothesis) and, meanwhile, receives the private name $x_j$ of each other process $P_j$.

2. Repeats (at most $k$ times) a choice where one guard is an output action on $x_i$, while the others are input actions on the $x_j$'s. If at a certain point an input is selected, then goes to 4.

3. If this point has been reached, then $P_i$ is the leader. It broadcasts this information to all the other processes, outputs $\bar{o}i$ and terminates.

4. Waits to receive the name of the leader. Then sends it on $o$ and terminates. □

Note that in the above proof we assume that each process know what's the total number of processes in the network.

The mechanisms of name-passing and scope extrusion, which makes it possible in the $\pi$-calculus to extend dynamically the communication structure of the network, are essential for the above result. In fact, such result would not hold for the "static subset" of the $\pi$-calculus i.e. CCS [Mil89], as shown by the following:

**Theorem 5.2** *Let $P = P_1|P_2|\ldots|P_k$ be a CCS network and let the associated hypergraph $H(P) = \langle N, X, t\rangle$ admit a well-balanced automorphism $\sigma$ such that $P$ is symmetric wrt $\sigma$ and, for each $n \in N$, there exist no $h$ such that $\{n, \sigma^h(n)\} \subseteq t(x)$ for some $x \in X$. Then $P$ cannot be an electoral system.*

**Proof** (Hint) Let $Q = Q_1|Q_2|\ldots|Q_q$ and $\theta$ be defined as in Corollary 4.3. An analysis of the kind of interactions possible between $Q_i$ and $Q^r_\theta(i)$ shows that, limited to the those transitions, these processes enjoy the confluence property (Lemma 4.1). In fact a (parallel) component $P_j$ of $Q_i$ can only interact with a (parallel) component $P_{\sigma^r(h)}$ of $Q^r_\theta(i)$ different from the component $P_{\sigma^r(j)}$. □

## 6 Uniform encoding

In this section we use the above results to show the non-encodability of the $\pi$-calculus into its asynchronous subsets and into CCS, under certain requirements on the notion of encoding $[\![\cdot]\!]$.

There is no agreement on what should be a good notion of encoding, ad perhaps indeed there should not be a unique notion, but several, depending on the purpose. However, it seems reasonable to require at least the two following properties:

1. compositionality,

2. preservation of some intended semantics.

For a distributed system, however, it seems reasonable to strengthen the notion of compositionality on the parallel operator by requiring that it is mapped exactly in the parallel operator, i.e. that

$$[\![P|Q]\!] \;=\; [\![P]\!] \,|\, [\![Q]\!] \qquad (1)$$

Likewise, it seems reasonable to require that the encoding "behaves well" wrt renamings, i.e.

$$[\![\sigma(P)]\!] \;=\; \sigma([\![P]\!]) \tag{2}$$

We will call *uniform* an encoding which satisfies (1) and (2).

Concerning the notion of semantics, we call "reasonable" a semantics which distinguishes two processes $P$ and $Q$ whenever in some computation of $P$ the actions on certain intended channels are different from those of any computation of $Q$. In the following, our intended channel is $o$.

**Remark 6.1** *There exist no uniform encoding of the $\pi$-calculus into the $\pi_a$-calculus preserving a reasonable semantics.*

**Proof** Uniformity preserves symmetry, and a reasonable semantics distinguishes an electoral system from a non-electoral one. Hence apply Theorems 5.1 and 4.2. □

**Remark 6.2** *There exist no uniform encoding of the $\pi$-calculus into CCS preserving a reasonable semantics.*

**Proof** Analogous, by Theorems 5.1 and 5.2. □

Note that if we relax condition (1), imposing just generic compositionality instead, i.e.

$$[\![P|Q]\!] \;=\; C[\,[\![P]\!],[\![Q]\!]\,] \tag{3}$$

with $C[\cdot,\cdot]$ generic context, then these non-encodability results do not hold anymore. In fact, we could give an encoding of the form

$$[\![P|Q]\!] \;=\; \nu y_1 \nu y_2 \ldots \nu y_n([\![P]\!]|M|[\![Q]\!])$$

where $M$ is a "monitor" process which coordinates the activities of $P$ and $Q$, interacting with them via the fresh channels $y_1, y_2, \ldots, y_n$. The translation of a network $P_1|P_2|\ldots|P_n$ would then be a tree with the $P_i$'s as leaves, and the monitors as the other nodes. The disadvantage of this solution is that it is not a distributed implementation; on the contrary, it is a very centralized one.

## 7 Conclusion and related work

One way to interpret the results presented in this paper is that they show that, even in a rich language like $\pi$-calculus, the full choice cannot be implemented into its sublanguage without choice. Actually, we can easily see that Lemma 4.1, and therefore Theorem 4.2, hold even if we consider a language with both input-guarded choice and output-guarded choice, but fail when we consider mixed choice (input and output guards in the same choice construct). Hence it is this latter mechanisms which induces a separation in expressive power. This seems to reinforce the impression that the mixed choice is a really difficult mechanism to implement. So far, the only really distributed, but approximated solutions we are aware of are the probabilistic methods based on randomization (see for instance [FR80]).

Another way to interpret them is by saying that the "real", i.e. simultaneous, synchronous communication cannot be implemented in the asynchronous one. In this sense, the translation of [Bou92] would not be acceptable since the randez-vous discipline introduces a delay. In this view of things, it is not the choice that is the hard operator: mixed choice would be easy to realize if real synchronous communication would be available. It is difficult, however, to argue in favor of this interpretation by using the results of this paper, because the underlying model of the $\pi_a$-calculus formalizes communication via simultaneous interaction (i.e. "handshaking", via the Com rule). In ongoing work, we are studying the impossibility results in the context of a "real" model for asynchronous communication, like the one of Asynchronous ACP ([BKT85]).

The non-existence results of this work hold even if we restrict to *fair* computations. The proof of Theorem 4.2 in fact can be slightly modified so that for the construction of $C_{h+1}$ from $C_h$ we consider each time a different process in the network. In this way, the limit of the sequence is a fair computation.

Our Theorems 4.2 and 5.2 correspond to Theorems 3.2.1 and 4.2.1 in [Bou88], for $CSP_{in}$ and $CSP$ respectively. The main difference with those results is that here we are dealing with much richer languages. In particular, both the $\pi_a$-calculus and CCS admit the parallel operator inside every process, and not just at the top-level as it is the case for $CSP_{in}$ and $CSP$ (at least, for the versions considered in [Bou88]: all processes in a network are strictly sequential). This leads to an essential difference. Namely, the proof of Bougé shows that the network can get stucked in the attempt to elect a leader: since an output action in $CSP_{in}$ can be only sequential, the prefix of a computation which leads to the first output action, repeated by all processes, brings to a global deadlock. Our proof, on the contrary, shows that the system can run forever without reaching an agreement: whenever a first output action occurs, all the other processes can execute their corresponding output action as well, and so on, thus generating an infinite computation which never breaks the symmetry. Another difference is that in the $\pi$-calculus the network can evolve dynamically. This is the reason why Theorem 4.2.1 in [Bou88] does not hold for the $\pi$-calculus (as shown by our Theorem 5.1). This feature complicates the proof of Theorems 4.2 since we have to take into account a corresponding evolution of the automorphism.

The use of the parallel operator as a free constructor usually enhances significatively the expressive power of a language. It is for instance essential for implementing choice (at least in a restricted form). In fact, Bougé has shown in [Bou88] that it is not possible to encode $CSP_{in}$ into $CSP_{no}$ (the sublanguage of $CSP$ with neither input nor output guards in the choice), while Nestmann and Pierce have shown in [NP96] that the $\pi_a$-calculus can be embedded into its subset with no choice. The crucial point is that the parallel operator allows to represent the main characteristic of the choice, namely the simultaneous availability of its guards.

## Acknowledgements

I would like to thank Ilaria Castellani, Pat Lincoln, Dale Miller, Uwe Nestmann, Prakash Panangaden, Benjamin Pierce, Rosario Pugliese, Scott Smolka and Eugene Stark for stimulating and insightful discussions.

## References

[ACS96] R. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous $\pi$-calculus. In U. Montanari, editor, *CONCUR 96*, volume 1119 of *Lecture Notes in Computer Science*, pages 147–162. Springer-Verlag, 1996.

[BKT85] J.A. Bergstra, J.W. Klop, and J.V. Tucker. Process algebra with asynchronous communication mechanisms. In S.D. Brookes, A.W. Roscoe, and G. Winskel, editors, *Proc. Seminar on Concurrency*, volume 197 of *Lecture Notes in Computer Science*, pages 76–95. Springer-Verlag, 1985.

[Bou88] L. Bougé. On the existence of symmetric algorithms to find leaders in networks of communicating sequential processes. *Acta Informatica*, 25:179–201, 1988.

[Bou92] G. Boudol. Asynchrony and the $\pi$-calculus. Technical Report 1702, INRIA, Sophia-Antipolis, 1992.

[BS96] M. Boreale and D. Sangiorgi. Some congruence properties for $\pi$-calculus bisimilarities. Technical report, INRIA, Sophia-Antipolis, 1996.

[FR80] N. Francez and M. Rodeh. A distributed abstract data type implemented by a probabilistic communication scheme. In *Proc. of the 21st Annual Symp. on Formal Aspects of Computing*, pages 373–379, 1980.

[Hoa78] C.A.R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[HT91] K. Honda and M. Tokoro. An object calculus for asynchronous communication. In P. America, editor, *ECOOP 91*, volume 512 of *Lecture Notes in Computer Science*, pages 133–147. Springer-Verlag, 1991.

[HT92] K. Honda and M. Tokoro. On asynchronous communication semantics. In M. Tokoro, O. Nierstrasz, and P. Wegner, editors, *Object-Based Concurrent Computing*, volume 612 of *Lecture Notes in Computer Science*, pages 21–51. Springer-Verlag, 1992.

[JJH90] He Jifeng, M.B. Josephs, and C.A.R. Hoare. A theory of synchrony and asynchrony. In *Proc. of IFIP Working Conference on Programming Concepts and Methods*, pages 459–478, 1990.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall International, 1989.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, Part I/II. *Information and Computation*, 100(1):1–77, 1992.

[NP96] U. Nestmann and B.C. Pierce. Decoding choice encodings. In U. Montanari, editor, *CONCUR 96*, volume 1119 of *Lecture Notes in Computer Science*, pages 179–194. Springer-Verlag, 1996.

[San95] D. Sangiorgi. $\pi$-calculus, internal mobility, and agent-passing calculi. *Theoretical Computer Science*, 1995. To appear.