On the Expressive Power of Global and Local Priority in Process Calculi

Cristian Versari Nadia Busi Roberto Gorrieri

Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, 40127 Bologna, Italy
{versari,busi,gorrieri}@cs.unibo.it

Abstract

Priority is a frequently used feature of many computational systems. In this paper we study the expressiveness of two process algebras enriched with different priority mechanisms. In particular, we consider a finite (i.e. recursion-free) fragment of asynchronous CCS with global priority (FAP, for short) and Phillips' CPG (CCS with local priority), and we contrast their expressive power with that of two non-prioritised calculi, namely the π-calculus and its broadcast-based version, called bπ. We prove, by means of leader-election-based separation results, that there exists no encoding of FAP into π-Calculus or CPG, under certain conditions. Moreover, we single out another problem in distributed computing, we call the *last man standing* problem (LMS for short), that better reveals the gap between the two prioritised calculi above and the two non prioritised ones, by proving that there exists no parallel-preserving encoding of the prioritised calculi into the non-prioritised calculi retaining any *sincere* (complete but partially correct, i.e., admitting divergence or premature termination) semantics.

1 Introduction

Priority is a frequently used feature of many computational systems. High-priority processes dispose of more central processing unit time in workstations, or preempt the execution of low priority processes through hardware/software-driven interrupt mechanisms. In order to model such systems, many basic process algebras have been enriched with some priority mechanisms (see, e.g., [1, 2, 3, 4, 5]). Priority is also implicitly used in many stochastic process calculi, where immediate actions take precedence over timed actions (see, e.g., [6, 7, 8], or where actions are equipped with an explicit priority level (e.g., [9]).

In this paper we investigate the expressiveness of priority in (untimed) concurrent systems, in order to delineate the expressive power gained by the addition of priority and to compare the relative expressive power of different priority mechanisms, by studying a couple of problems in distributed systems.

According to the classification in [4], the basic priority mechanisms reported in the literature can be divided into two main groups: those based on *global* priority (see, e.g., [10, 3]) and those based on *local* priority (see, e.g., [2, 5]). The difference is motivated by the *scope* of the priority effects on the system: in the case of global priority, a high-priority action is able to preempt any other low-priority action in the system, so that only higher priority processes are allowed to evolve. In the case of local priority, this effect is limited to the *location* of the process, where a location can be thought as a site on a distributed system and may be represented by the scope of some name or the guarded choice between actions with different priorities. An example may be helpful in showing the difference between the two. Consider the system S composed of five processes

$$S \equiv \overline{a}.P \mid a.Q_1 + \underline{b}.Q_2 \mid \overline{\underline{b}}.R \mid c.T_1 + d.T_2 \mid \overline{c}.V$$

where the sum operator represents the usual choice between different actions, output actions are overlined and high-priority actions are underlined. According to the semantics of CCS^{sg} (CCS with a global notion of priority) and CCS^{sl} (CCS with local priority) reported in [4], the processes $a.Q_1 + \underline{b}.Q_2$ and $\overline{\underline{b}}.R$ are ready to perform a high-priority action on channel $\underline{b}$. In CCS^{sg} semantics this action is forced to happen before any other low priority transition in S, while in CCS^{sl} semantics, the action $\underline{b}$ only preempts the

execution of the action a, so that the synchronisation on c of the last two processes may even happen first. In other words, with a global priority notion the only possible internal transition of the system S is

$$S \to \overline{a}.P \mid Q_2 \mid R \mid c.T_1 + d.T_2 \mid \overline{c}.V$$

while in presence of local priority also the evolution

$$S \to \overline{a}.P \mid a.Q_1 + \underline{b}.Q_2 \mid \overline{\underline{b}}.R \mid T_1 \mid V$$

can happen. In both cases only the reaction on channel a is preempted.

As a basic representative for a calculus with global priority, we consider in this paper a very minimal fragment, we call FAP, of CCS [11] (without restriction and recursion, with asynchronous communication), enriched with static priority and global preemption (like in CCS^{sg} reported in [4]) where only the prefix operator on inputs is present and the asynchronous output is characterised by the possibility of assigning different priority to the outgoing messages.

As a representative for a calculus with local priority, we consider in this paper Phillips' CCS with priority guards (CPG for short) [5].

Moreover, we consider two well-known unprioritised calculi, namely the π-calculus [12, 13] and its broadcast-based version bπ-Calculus [14], that will be compared with the two prioritised calculi above.

The two problems in distributed systems we will use to distinguish the expressive power of these four calculi are the *leader-election* problem [15], already used to study expressiveness gap between, e.g., synchronous and asynchronous π-calculus [16], and an apparently new problem we have called *the last man standing* problem (LMS for short), consisting in the capability for processes of recognising the absence of other processes ready to perform synchronisations or input/output operations. In other words, the LMS problem is solvable if a process is able to check that it is the only one active in a network.

1.1 Contribution of this paper

The first application of the leader election problem to the π-Calculus [16] allowed to observe the superior expressiveness of mixed-choice with respect to separated-choice in the π-Calculus. Following the same approach many other separation results have been obtained, each one based on the capability or impossibility to solve the leader election under appropriate conditions. An adapted version was used in [14] to show that the broadcast-based bπ-Calculus, is more expressive than the standard π-Calculus provided with point-to-point communication primitives. The result is based on the idea that broadcast permits to solve the leader election problem even without any knowledge of the number of processes participating to the election.

The same approach was used in [5] to separate CPG from CCS and also from the π-Calculus by exploiting the broadcast-like power of preemption introduced by priority, while π-Calculus capability of creating new communication links between processes is exploited to prove the converse separation from CPG. This result is the only on the expressiveness of priorities in process algebras we are aware of.

In this paper we first analyse the expressiveness of global priority, in order to check if it can be proved to be more expressive than local priority as it would be natural expecting. In order to represent a global priority model we choose FAP, a fragment of CCS added with stratified, global priority (a slight variant of the CCS with static priority and global preemption, CCS^{sg} studied in [4]) where the only operator is the prefix on inputs, while the asynchronous output models the dispatch of messages with two different levels of priority: the delivery of high-priority messages is ensured to happen before that of low-priority ones.

We prove that this very simple language (deprived of synchronous communication, choice, recursion or replication and hence finite) consents to write programs capable of solving the leader election problem in any connected graph of processes without knowledge of the number of processes involved in the election.

By applying the idea used in [14, 5] we have as corollary that FAP cannot be distributively encoded into the π-Calculus, but we prove this to remain true also for partially correct encodings which introduce divergence or failure in their computations as consequences of livelocked or deadlocked conditions. This result can also be extended to the translations of bπ-Calculus and CPG into the π-Calculus, thus relaxing the encoding conditions already stated in [14, 5].

Another consequence of the above leader election result in FAP is the impossibility of its encoding into CPG under uniformity and independence-preserving conditions, which constitutes the expected result on

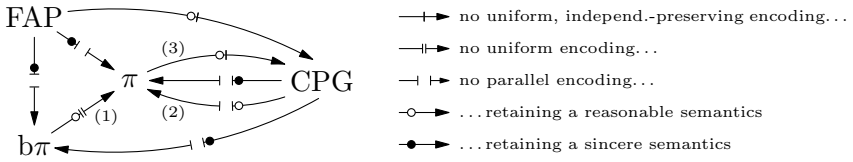

Figure 1: Impossibility results: (1) C.Ene, T.Muntean [14]; (2, 3) I. Phillips [5, 17]; the remaining ones are presented in this paper.

the expressiveness gap between global and local priority in the chosen process algebra framework. It is worth considering that the separation between these two prioritised languages and the π-Calculus is stronger than that between FAP and CPG themselves, which is a first hint on the expressive power of priority on both global and local approaches.

In order to strengthen the separation between prioritised and non-prioritised languages, we then introduce a new setting denoted as the *last man standing problem*. In this setting a bunch of n processes must realise if there is only one process participating to the LMS (and in that case, the only process would be the "last man standing"), i.e. understand if $n = 1$ or $n > 1$ in a distributed way. We prove that it is possible to solve the LMS both in FAP and CPG (but we claim that it is also possible within other priority approaches like [2, 4]), while it is not possible in non-prioritised languages like the π-Calculus but also the bπ-Calculus. This result implies that there exist no distributed encodings of FAP and CPG into the bπ-Calculus, hence showing that the greatest expressiveness of priority does not derive from the broadcast-like power of preemption, but from the capability of processes to know if another process is ready to perform a synchronisation on some channel *or not*. In non-prioritised calculi it is possible to know if some process *is* ready to perform some synchronisation, but it is not decidable if, on the contrary, the condition does not hold. We show that in a distributed setting this simple capability is proper of priority (global or local, stratified or not) and cannot be obtained anyhow even if providing broadcast-like primitives and admitting divergent or deadlocked computations.

The above results are summarised in figure 1.

1.2 Structure of the paper

The rest of the paper is structured as follows. In section 2 the four process algebras involved in the separation results are introduced and a brief explanation of their main features is given. Section 3 contains a short discussion and the formal definitions of the properties of an encoding. In Sect. 3.1 and 3.2 the leader election and LMS problems are formalised. In Sect. 4.1 and 4.2 the respective separation results are shown, then some conclusive remarks are reported.

2 Calculi

We introduce now the calculi of interest in this paper by giving their syntax and a short explanation of their functioning while for the respective semantics we refer to [18, 19, 14, 5, 17] for lack of space, except for FAP whose definition is comprehensive.

2.1 The π-Calculus

The π-Calculus [12, 13] is a derivation of CCS [11] where processes interact through synchronisation over named channels, with the capability of receiving new channels and subsequently using them for the interaction with other processes in order to model mobility.

Definition 1 *Let $\mathcal{N}$ be a set of names on a finite alphabet, $x, y, \ldots \in \mathcal{N}$. The syntax of the π-Calculus is defined as*

$$P \quad ::= \quad \mathbf{0} \;\Big|\; \sum_{i \in I} \pi_i.P_i \;\Big|\; P \mid Q \;\Big|\; !P \;\Big|\; (\nu x)P$$

$$\pi \quad ::= \quad \tau \;\Big|\; x(y) \;\Big|\; \overline{x}\langle y \rangle$$

where: $\mathbf{0}$ represents the null process, $x(y)$ expresses the capability of performing an input on the channel x and receiving a datum which is then bound to the name y; $\overline{x}\langle y\rangle$ expresses the capability of sending the name y on the channel x; τ is the invisible, uncontrollable action; $P \mid Q$ represents the parallel composition of processes; $!P$ stands for the unlimited replication of process P; $\sum_{i\in I}\pi_i.P_i$ represents the nondeterministic choice between several input / output communication capabilities, denoted also as $\pi_1.P_1 + \pi_2.P_2 + \ldots$; $(\nu x)P$ represents the scope restriction of the name x to process P.

In order to define the observables of a π-Calculus process P, we introduce the notion of barb.

Definition 2 *Let P be a π-Calculus process. P exhibits barb α, written $P \downarrow \alpha$, iff*

- $P \equiv (\nu\tilde{y})(x(z).Q + R \mid S)$, *with* $\alpha = x$, $x \notin \tilde{y}$ *or*
- $P \equiv (\nu\tilde{y})(\overline{x}\langle z\rangle.Q + R \mid S)$, *with* $\alpha = \overline{x}$, $x \notin \tilde{y}$.

Each barb α represents one action that P is immediately ready to perform.

For a full treatment we refer to [18, 19].

2.2 The bπ-Calculus

The bπ-Calculus [14] is a variant of the π-Calculus where the point-to-point synchronisation mechanism is replaced by broadcast communication. For example, while the π-Calculus program

$$S \equiv \overline{a}\langle b\rangle.P \mid a(x).Q \mid a(y).R \mid a(z).T$$

can evolve in one step to a system like S_1

$$S \to S_1 \equiv P \mid Q\{b/x\} \mid a(y).R \mid a(z).T$$

where only one of Q, R, S is affected by the performed communication, in bπ-Calculus the system S directly evolves to S_2

$$S \to S_2 \equiv P \mid Q\{b/x\} \mid R\{b/y\} \mid T\{b/z\}$$

where all the processes listening on channel a receive the broadcasted message.

Definition 3 *Let $\mathcal{N}$ be a set of names on a finite alphabet, $x, y, \ldots \in \mathcal{N}$. The syntax of the b$\pi$-Calculus is defined in terms of the following grammar* [1]

$$P \quad ::= \quad \mathbf{0} \;\Big|\; A\langle\tilde{x}\rangle \;\Big|\; \sum_{i\in I}\alpha_i.P_i \;\Big|\; P_1 \mid P_2 \;\Big|\; \nu x P \;\Big|\; (\mathrm{rec}\, A\langle\tilde{x}\rangle.P)\langle\tilde{y}\rangle$$

where

$$\alpha_i \quad ::= \quad x(y) \;\Big|\; \overline{x}y \;\Big|\; \nu y\overline{x}y \;\Big|\; \tau$$

As for the π-Calculus, we define the notion of barb in order to express the observable actions a bπ process is ready to perform.

Definition 4 *Let P be a bπ-Calculus process. P exhibits barb α, written $P \downarrow \alpha$, iff one of*

- $P \equiv \sum_{i\in I}\pi_i.P_i$, $\pi_i = x(y)$ *for some i and* $\alpha = x$;
- $P \equiv \sum_{i\in I}\pi_i.P_i$, $\pi_i = \overline{x}\langle y\rangle$ *for some i and* $\alpha = \overline{x}$;
- $P \equiv P_1 \mid P_2$ *and* $P_1 \downarrow \alpha \vee P_2 \downarrow \alpha$;
- $P \equiv \nu a P'$ *and* $P' \downarrow \alpha$, *with* $\alpha, \overline{\alpha} \neq a$;
- $P \equiv (recA\langle\tilde{x}\rangle.P')\langle\tilde{y}\rangle$ *and* $P'[(recA\langle\tilde{x}\rangle.P')/A, \tilde{y}/\tilde{x}] \downarrow \alpha$;

is satisfied.

The original semantics is given in terms of a labelled transition systems: in order to simplify and uniform the notation, from now on we say that $P \to Q \iff P \xrightarrow{\overline{x}y} Q \vee P \xrightarrow{\nu y\overline{x}y} Q \vee P \xrightarrow{\tau} Q$. For a detailed description of bπ-Calculus syntax and its semantics we refer to [14] for lack of space.

[1] For sake of clarity we maintain bπ original syntax since for its semantics we refer entirely to [14].

2.3 The CPG Language

The CPG language [5] derives from CCS with the addition of a local notion of priority over actions. Each action is guarded by a set of names representing the actions whose availability may prevent its execution. For example, the system S

$$S \equiv a.P \mid \overline{a}.Q \mid (a : b.R_1 + c : d.R_2) \mid \overline{b}.T \mid \overline{d}.V$$

may perform a synchronisation on channel a, but also on channel d because no action $\overline{c}$ is available. Instead a communication on b is not possible because it is prevented by the presence of the action $\overline{a}$.

Definition 5 *Let $\mathcal{N}$ be a set of names on a finite alphabet, $x, y, \ldots \in \mathcal{N}$. CPG syntax is defined in terms of the following grammar*

$$P \quad ::= \quad \mathbf{0} \;\Big|\; A\langle \tilde{x} \rangle \;\Big|\; \sum_{i \in I} S_i : \alpha_i.P_i \;\Big|\; P_1 \mid P_2 \;\Big|\; \nu x P \;\Big|\; A(x_1, \ldots, x_n) \stackrel{def}{=} P$$

where

$$\alpha_i \quad ::= \quad x \;\Big|\; \overline{x} \;\Big|\; \tau$$

and $S_i \subseteq \mathcal{N}$ represents the set of actions whose co-actions availability prevents the execution of α_i.

We report the definition of barb for CPG in [17].

Definition 6 *Let P be a CPG process. P exhibits barb α, written $P \downarrow \alpha$, iff*

- $P \equiv (\nu \tilde{y})(S : x.Q + R \mid T)$, *with* $\alpha = x$, $x \notin \tilde{y}$ *or*
- $P \equiv (\nu \tilde{y})(S : \overline{x}.Q + R \mid T)$, *with* $\alpha = \overline{x}$, $x \notin \tilde{y}$.

For a full treatment of the CPG language we refer to [5, 17].

2.4 The FAP language

As previously outlined, the FAP language is a slight variant of a minimal CCS^{sg} fragment, that is CCS added with static, global priority [4]: by keeping FAP minimal the expressive power of global priority can be better isolated. Only two operators are present in FAP: parallel composition and prefix. The prefix operation is allowed only after input actions, so that the output can be considered asynchronous as for the asynchronous π-Calculus (see e.g. [16]). Output actions are characterised by two priority levels, meaning that high priority output synchronisations are guaranteed to happen before low priority ones. As an example, consider the system

$$S \equiv a.P \mid a.Q \mid b.R \mid \overline{a} \mid \underline{\overline{a}} \mid \overline{b}$$

The processes $\overline{a}, \underline{\overline{a}}, \overline{b}$ model messages which must be delivered to the processes listening on the appropriate channels: The message $\underline{\overline{a}}$ has higher priority w.r.t. any other message in S and hence must be delivered before. Consequently the only possible transitions of S are

$$S \to P \mid a.Q \mid b.R \mid \overline{a} \mid \overline{b} \qquad\qquad S \to a.P \mid Q \mid b.R \mid \overline{a} \mid \overline{b}$$

where the process receiving the message is nondeterministically chosen. After this transition, either $\overline{a}$ or $\overline{b}$ can finally be delivered. In order to simplify the notation, inputs lack any denotation of priority, but the results presented in this paper are completely independent of this design choice.

Definition 7 *Let $\mathcal{N}$ be a set of names on a finite alphabet, $x, \ldots \in \mathcal{N}$. FAP syntax is defined in terms of the following grammar*

$$P \quad ::= \quad \mathbf{0} \;\Big|\; x.P \;\Big|\; \overline{x} \;\Big|\; \underline{\overline{x}} \;\Big|\; P \mid Q$$

In order to keep it simple as well, we define FAP semantics in terms of a reduction system in the style of [18].

Definition 8 *Structural congruence for FAP is the congruence $\equiv$ generated by the following equations:*

$$P \mid \mathbf{0} \equiv P, \qquad P \mid Q \equiv Q \mid P, \qquad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$$

Definition 9 *FAP operational semantics is given in terms of the reduction system described by the following rules:*

$$\frac{}{x.P \mid \overline{x} \mapsto P} \qquad \frac{}{x.P \mid \underline{\overline{x}} \twoheadrightarrow P}$$

$$\frac{P \twoheadrightarrow P'}{P \mid Q \twoheadrightarrow P' \mid Q} \qquad \frac{P \mapsto P' \qquad P \mid Q \not\twoheadrightarrow R}{P \mid Q \mapsto P' \mid Q}$$

$$\frac{P \equiv Q \quad P \mapsto P' \quad P' \equiv Q'}{Q \mapsto Q'} \qquad \frac{P \equiv Q \quad P \twoheadrightarrow P' \quad P' \equiv Q'}{Q \twoheadrightarrow Q'}$$

We say that $P \to Q \iff P \mapsto Q \vee P \twoheadrightarrow Q$.

Definition 10 *For any process in FAP, the function* fn *is defined as*

$$\mathrm{fn}(\mathbf{0}) = \emptyset \qquad \mathrm{fn}(\overline{x}) = \{x\} \qquad \mathrm{fn}(x.P) = \{x\} \cup \mathrm{fn}(P)$$
$$\mathrm{fn}(\underline{\overline{x}}) = \{x\} \qquad \mathrm{fn}(P \mid Q) = \mathrm{fn}(P) \cup \mathrm{fn}(Q)$$

As for previous languages, we define the notion of barb.

Definition 11 *A FAP process P exhibits barb α, written as $P \downarrow \alpha$, iff*

- $P \equiv x.Q \mid R$, $\alpha = x$, *or*
- $P \equiv \overline{x} \mid R$, $\alpha = \overline{x}$, *or*
- $P \equiv \underline{\overline{x}} \mid R$, $\alpha = \underline{\overline{x}}$.

Proposition 12 *FAP is not Turing-complete.*

Proof. Every process $P \in$ FAP terminates — FAP lacks any loop operator such as bang or recursion.

3 Encodings

In order to provide the results previously outlined, we now formalise the encoding conditions relevant for the expressiveness separation between languages.

We first formalise the notion of *observables* of a program computation, in the style of [17].

Definition 13 *Let L be a process language and processes $P, P_0, \ldots \in L$. A* computation $\mathcal{C}$ of P *is a finite or infinite sequence $P = P_0 \to P_1 \to \cdots$. $\mathcal{C}$ is* maximal *if it cannot be extended.*

A computation of a process P is the sequence of states P can reach during its execution. Each process P may present many different computations due to the nondeterminism intrinsic in concurrent calculi.

Definition 14 *Let L be a process language with names in $\mathcal{N}$ and processes $P_0, \ldots, P_i \in L$. Let $\mathcal{C}$ be a computation $P_0 \to \cdots \to P_i \cdots$. Given a set of* intended observables $\mathrm{Obs} \subseteq \mathcal{N}$*, the observables of $\mathcal{C}$ are* $\mathrm{Obs}(\mathcal{C}) = \{x \in \mathrm{Obs} : \exists i\, P_i \downarrow x\}$.

The observables of a computation C are the set of all the external interactions the process may perform in the states reached during the computation.

Some of the separation results are based on the topology of the network of processes: for example, the encoding impossibility of the π-Calculus into value-passing CCS [16] is based on the hypothesis that the encoding does not increase the connection of the network, that is all the processes independent (not sharing free names) in the source language must remain independent after the encoding. The same criterion will be necessary to separate FAP and CPG.

Definition 15 *Let L be a process language. Two processes $P, Q \in L$ are* independent *if they do not share any free names, that is* $\mathrm{fn}(P) \cap \mathrm{fn}(Q) = \emptyset$.

We now define the conditions an encoding may preserve, in the style of [17].

Definition 16 *Let L, L' be process languages. An encoding $[\![\cdot]\!] : L \to L'$ is*

1. observation-respecting *if* $\forall P \in L$,
   - *for every maximal computation $\mathcal{C}$ of P there exists a maximal computation $\mathcal{C}'$ of $[\![P]\!]$ such that* $\mathrm{Obs}(\mathcal{C}) = \mathrm{Obs}(\mathcal{C}')$;
   - *for every maximal computation $\mathcal{C}$ of $[\![P]\!]$ there exists a maximal computation $\mathcal{C}'$ of P such that* $\mathrm{Obs}(\mathcal{C}) = \mathrm{Obs}(\mathcal{C}')$;
2. weakly-observation-respecting *if* $\forall P \in L$,
   - *for every maximal computation $\mathcal{C}$ of P there exists a maximal computation $\mathcal{C}'$ of $[\![P]\!]$ such that* $\mathrm{Obs}(\mathcal{C}) = \mathrm{Obs}(\mathcal{C}')$;
   - *for every maximal computation $\mathcal{C}$ of $[\![P]\!]$ there exists a maximal computation $\mathcal{C}'$ of P such that* $\mathrm{Obs}(\mathcal{C}) \subseteq \mathrm{Obs}(\mathcal{C}')$;
3. distribution-preserving *if* $\forall P_1, P_2 \in L, \quad [\![P_1 \mid P_2]\!] = [\![P_1]\!] \mid [\![P_2]\!]$;
4. renaming-preserving *if for any permutation σ of the source names in L there exists a permutation θ in L' such that $[\![\sigma(P)]\!] = \theta([\![P]\!])$ and the permutations are compatible on observables, that is* $\sigma|_{\mathrm{Obs}} = \theta|_{\mathrm{Obs}}$;
5. independence-preserving *if $\forall P, Q \in L$, if P and Q are independent then $[\![P]\!]$ and $[\![Q]\!]$ are also independent.*

The observation-respecting property is the minimal requirement that any reasonable encoding should preserve: it ensures that some intended observable behaviour is preserved by the translation. The weak variant of this condition admits encodings which introduce divergence or failure deriving from livelocks or deadlocks. Under certain conditions, like fairness or other hypotheses on scheduling or execution, the introduction of divergence or failure by the encoding may be tolerated [20, 21] because it would be guaranteed not (or very unlikely) to happen anyway.

The distribution-preserving is a very important feature of an encoding in a concurrent framework: it implies its compositionality and above all it guarantees that the degree of parallelism of the translated system does not decrease.

The renaming-preserving property states that the encoding should not introduce asymmetries in the system, essential condition to preserve when impossibility results on problems such as the leader election are completely based on the symmetric topology of the network.

Independence-preserving represents the property of not increasing the connection of the network.

According to [16, 17], a distribution- and renaming-preserving encoding is called *uniform*, and *reasonable* if it preserves the intended observables over maximal computations. We call *sincere* a weakly-observation-respecting encoding, which is complete but only weakly correct, in the meaning that it admits divergence or premature termination.

3.1 The leader election problem

An electoral system represents the situation of a bunch of processes (modelling for example a set of workstations in a network) aiming at reaching a common agreement, i.e. deciding which one of them (an no other) is the leader. The modelling of the election problem in process algebras requires that the system composed of the network of processes will sooner or later signal unequivocally the result of the election on some channels ω_i, which become consequently the observables of interest on the computations of the system.

Definition 17 *Let L be a process language, and processes $P_1, \ldots, P_k \in L$. A network* Net *of size k, with $k \geq 1$, is a system* $\mathsf{Net} = P_1 \mid \ldots \mid P_k$.

Definition 18 *A network* Net *of size* k *is an* electoral system *if for every maximal computation* $\mathcal{C}$ *of* Net $\exists i \leq k : \mathrm{Obs}(\mathcal{C}) = \{\omega_i\}$*, where* $\mathrm{Obs} = \{\omega_i : i \in \mathbb{N}\}$.

In order to keep notation simple, the definition of electoral system reflects the design choices kept in [16, 14, 5] which are based on the hypothesis that the system will never perform external interactions on channel which are not intended to be observable. As in [5, 17], the winner process (the leader) is supposed to signal the outcome of the election, while all the other processes simply do nothing.

Some additional definitions are given in order to formalise the idea of connected network, in the style of [16].

Definition 19 *A* hypergraph *of size* k *is a tuple* $H = \langle N, X, t\rangle$ *where* N *is a finite set of* nodes, $|N| = k$, X *a finite set of* edges, *and* $t : X \to 2^N$ *is a function which assigns to each* $x \in X$ *a set of nodes,* $t(x) = \{n_1, \ldots, n_i\}$. *Two nodes* $n_1, n_2 \in N$ *are neighbours if* $\exists x \in X : \{n_1, n_2\} \subseteq t(x)$.

Definition 20 *Given a network* $\mathsf{Net} = P_1 \mid \ldots \mid P_k$, *the* hypergraph associated *to* Net *is* $H(\mathsf{Net}) = \langle N, X, t\rangle$ *with* $N = \{1, \ldots, k\}$, $X = \mathrm{fn}(P_1 \mid \ldots \mid P_k)$, *and* $\forall x \in X, t(x) = \{n : x \in \mathrm{fn}(P_n)\}$.

Definition 21 *A hypergraph* $H = \langle N, X, t\rangle$ *is* connected *if* $|N| = 1$ *or* $\exists n \in N, x \in X : \{n\} \subset t(x)$ *and* $H' = \langle N', X, t'\rangle$ *is connected, where* $N' = N \setminus \{n\}$ *and* $t'(x) = t(x) \setminus \{n\} \forall x$. H *is fully connected if* $|N| = 1$ *or* $\forall n_1, n_2 \in N\ \exists x \in X :\ \{n_1, n_2\} \subset t(x)$.

In a connected hypergraph each pair of nodes is connected by a sequence of hyperedges: Def. 21 concisely represents this condition. Nodes are directly connected by an edge in a fully connected hypergraph. We say that a network is (fully) connected if the associated hypergraph is (fully) connected.

3.2 The last man standing problem

The last man standing represents a very simple situation where a bunch of n processes in a fully connected network must realise if $n = 1$ or $n > 1$ in a distributed way. The possibility or impossibility to solve the LMS problem is based on the idea that in a given language L a process P *may* or *may not* know if another process Q is ready to perform some intended action on a given channel. Usually the only way P has to know the presence of Q is to *try* a synchronisation on it. Since the input (and often also the output) is blocking, P results blocked if the condition does not hold, or it follows another computation without knowledge on the presence of Q. The definition of LMS system follows.

Definition 22 *A network* Net_k *of size* k *is a* LMS system *if for every maximal computation* $\mathcal{C}$ *of* Net_k

- $\mathrm{Obs}(\mathcal{C}) = \{y\}$ *if* $k = 1$,
- $\mathrm{Obs}(\mathcal{C}) = \{n\}$ *if* $k > 1$,

where $\mathrm{Obs} = \{y, n\}$.

The above definition implies that any LMS system is a fully connected network. It is possible to adapt the definition for not fully connected networks, in order to better exploit the scoping of local priorities and still the separation results based on LMS would hold.

4 Separation results

The separation results previously outlined follow. We first give those based on the leader election, and then those based on the LMS problem.

4.1 Leader-election-based separation results

The bπ-Calculus and CPG were proved capable of solving the leader election in a fully connected network without knowledge of the number of processes. Here we show that in FAP this is possible in any (not only fully) connected network.

Theorem 23 *Let $P_1, \ldots, P_k$ be FAP processes, $\mathsf{Net} = P_1 \mid \cdots \mid P_k$ and $H = \langle N, X, t\rangle$ be the hypergraph of size k associated to Net. Let*

$$\begin{aligned} P_n = \overline{m}_n \mid \underline{s}_n \mid m_n.s_n.(\omega_n \mid \overline{\underline{d}}_{n1} \mid \cdots \mid \overline{\underline{d}}_{nz_n}) \\ \mid d_{n1}.(s_n \mid \overline{\underline{d}}_{n1} \mid \cdots \mid \overline{\underline{d}}_{nz_n}) \\ \vdots \\ \mid d_{nz_n}.(s_n \mid \overline{\underline{d}}_{n1} \mid \cdots \mid \overline{\underline{d}}_{nz_n}) \end{aligned}$$

where P_i and P_h are neighbours iff $\exists j, l : d_{ij} = d_{hl}$, with ω_i, s_j, m_h distinct and $\omega_i \neq s_j \neq m_h \neq d_{pq}, \forall i, j, h, p, q$. If H is connected then Net is an electoral system.

The following lemma [17] is needed to prove that the above results cannot be obtained without knowledge of the number of processes in the electoral system and also to show that the LMS problem is undecidable in the π-Calculus. As noted in [17], it would also hold for any language having comparable semantics of the parallel operator, such as Mobile Ambients [22].

Lemma 24 *1. For any π-Calculus processes P_1, P_2, if P_i has a maximal computation with observables $O_i (i = 1, 2)$ then $P_1 \mid P_2$ has a maximal computation with observables O such that $O_1 \cup O_2 \subseteq O$.*

Next we state a similar but weaker result for the bπ-Calculus, which is also needed for the separation from FAP and CPG based on the LMS problem.

Lemma 25 *1. For any bπ-Calculus processes P_1, P_2, if P_i has a maximal computation with observables $O_i (i = 1, 2)$ then $P_1 \mid P_2$ has two maximal computations $\mathcal{C}_1, \mathcal{C}_2$ (not necessarily distinct) with respective observables O'_1, O'_2 such that $O_i \subseteq O'_i$.*

The next theorem follows the idea in [14, 5]. As discussed for the definition of sincere semantics, here the condition on the preserved observables is weaker than those considered in [14, 5] but it is possible to relax them as well.

Theorem 26 *There is no distribution-preserving and weakly-observation-respecting encoding of FAP into the π-Calculus.*

Proof. Consider $P_n = \overline{m}_n \mid \underline{s}_n \mid m_n.s_n.(\omega_n \mid \overline{\underline{d}}) \mid d.(s_n \mid \overline{\underline{d}})$ for $n = 1, 2$. By theorem 23, $\mathsf{Net}_1 = P_1$, $\mathsf{Net}_2 = P_2$ and $\mathsf{Net}_{12} = P_1 \mid P_2$ are electoral systems. Let $[\![\cdot]\!]$ be a weakly-observation-respecting and distribution-preserving encoding of FAP into the π-Calculus. Hence $[\![P_1]\!]$ has a maximal computation $\mathcal{C}_1$ with observables $\mathrm{Obs}(\mathcal{C}_1) = \omega_1$, because of the weakly-observation-respecting condition. But also $[\![P_2]\!]$ has a maximal computation $\mathcal{C}_2$ with observables $\mathrm{Obs}(\mathcal{C}_2) = \omega_2$.

So we have, for the distribution-preserving property,

$$[\![\mathsf{Net}_{12}]\!] = [\![P_1 \mid P_2]\!] = [\![P_1]\!] \mid [\![P_2]\!]$$

By lemma 24, $[\![\mathsf{Net}_{12}]\!]$ has a maximal computation $\mathcal{C}_{12}$ with observables $\{\omega_1, \omega_2\} \subseteq \mathrm{Obs}(\mathcal{C}_{12})$, not included in the set of observables of any maximal computation of Net_{12}, which contradicts $[\![\cdot]\!]$ being weakly-observation-preserving.

Theorem 27 *There is no uniform, observation-respecting and independence-preserving encoding of FAP into CPG.*

Proof. (sketch) By theorem 23, it is possible to solve in FAP the leader election in any symmetric ring. The impossibility result is then the same proved for the encoding of the π-Calculus into CPG [17]. It is worth remarking that while the separation between π-Calculus and CPG derives from the capability of communicating new names proper of the π-Calculus, the separation between FAP and CPG is a strict consequence of the different scope of priority in the two languages.

4.2 LMS-based separation results

In this section the separation results based on the last man standing problem are formalised. First we show that both in FAP and CPG the LMS can be solved, and then from this expressive capability we derive the impossibility of encoding FAP or CPG into π-Calculus or bπ under distribution and weak preservation of observables hypotheses.

Lemma 28 *Let P be a FAP process,*

$$P = \overline{m} \mid \underline{s} \mid \underline{q} \mid k.(s \mid q \mid \overline{\underline{k}}) \mid m.s.(s.q.(\overline{\underline{k}} \mid n) \mid \overline{l} \mid l.q.y)$$

Then $\mathsf{Net}_k = \underbrace{P \mid \cdots \mid P}_{k}$ *is a LMS system of size k, $\forall k \geq 1$.*

Lemma 29 *Let P be a CPG process,*

$$P = a : b.\overline{a} \mid \overline{b}.(b : \tau.y \mid z : b.(\overline{b} \mid n \mid \overline{z}))$$

Then $\mathsf{Net}_k = \underbrace{P \mid \cdots \mid P}_{k}$ *is a LMS system of size k, $\forall k \geq 1$.*

The following is an alternative way w.r.t. theorem 26 to prove the separation between FAP and π-Calculus and act as a template for the next theorems.

Theorem 30 *There is no distribution-preserving and weakly-observation-respecting encoding $[\![\cdot]\!]$ of FAP into the π-Calculus.*

Proof. Suppose $[\![\cdot]\!]$ is distribution-preserving and weakly-observation-respecting. By lemma 28 $\exists P$: Net_k is a LMS system for any $k \geq 1$, where $\mathsf{Net}_k = P \mid \ldots \mid P$. By the weakly-observation-respecting condition, $[\![\mathsf{Net}_1]\!]$ has a computation $\mathcal{C}_1$ with observables $\mathrm{Obs}(\mathcal{C}_1) = \{y\}$. By the distribution-preserving condition

$$[\![\mathsf{Net}_k]\!] = [\![P \mid \ldots \mid P]\!] = [\![P]\!] \mid \ldots \mid [\![P]\!] = [\![\mathsf{Net}_1]\!] \mid \ldots \mid [\![\mathsf{Net}_1]\!]$$

By lemma 24 then there exists a maximal computation $\mathcal{C}_k$ of $[\![\mathsf{Net}_k]\!]$ such that $\mathrm{Obs}(\mathcal{C}_1) = \{y\} \subseteq \mathrm{Obs}(\mathcal{C}_k)$, while no computation of Net_k contains observable y for $k \geq 2$, which contradicts the weakly-observation-respecting property of the encoding function $[\![\cdot]\!]$.

Theorem 31 *There is no distribution-preserving and weakly-observation-respecting encoding of CPG into the π-Calculus.*

Proof. By lemma 29 exactly like for theorem 30.

Theorem 32 *There is no distribution-preserving and weakly-observation-respecting encoding of FAP into the bπ-Calculus.*

Proof. By lemmas 25 and 28, exactly like for theorem 30.

Theorem 33 *There is no distribution-preserving and weakly-observation-respecting encoding of CPG into the bπ-Calculus.*

Proof. By lemmas 25 and 29, exactly like for theorem 30.

5 Conclusion

We have considered FAP, a finite fragment of CCS added with global priority, and we have proved, by means of leader-election-based separation results, that it is not possible to encode it into CPG under uniformity and independence-preserving conditions on the encoding, thus providing the first expressiveness separation result between global and local priority within a process algebra framework. We have then proved that FAP cannot be distributively translated into the π-Calculus even if allowing partially correct implementations, i.e. encodings which may introduce divergence in computations or also premature termination caused by deadlock.

We have then analysed another setting, called last man standing (LMS) problem which allows to considerably strengthen the separation between prioritised (with both global or local priority) languages and non prioritised ones, by showing that even if we equip the language with broadcast-based primitives like the bπ-Calculus, the expressiveness of priority cannot be obtained under parallel-preserving conditions.

In conclusion we have showed that, within the context of the process algebras considered here, it is not possible to have a distribution-preserving encoding of neither global nor local priority in non-prioritised languages even if we admit asymmetric translations or divergence/failure in the computation as a consequence of livelocks or deadlocks. This impossibility result does not depend on the capability of communication of names or values, synchrony or asynchrony of the output, scope extrusion, choice on the available input/outputs, recursion or replication, point-to-point or broadcast communication/synchronisation type, but depends only on the power of instantaneous preemption characteristic of the prioritised languages considered in this paper. As a consequence we can see that it is not possible any purely distributed implementation of such kinds of priority on top of standard process calculi even if admitting good or randomised encodings like those considered in [20, 21] for the implementation of the choice operator. The strength of the separation suggests also that any encoding trying to preserve some relaxed condition on the distribution may be affected by severe performance issues due to the further synchronisations needed to preserve the constraint of instantaneous preemption.

As future work we plan to analyse process algebras equipped with non-instantaneous priority (in the style of the expressiveness study on PrioLinCa [23]) i.e. languages where the effect of preemption is not immediate, in order to better characterise the expressive power of preemption and to identify prioritised constructs easier to implement in a parallel, if not distributed, framework.

References

[1] Baeten, J.C.M., Bergstra, J.A., Klop, J.W.: Ready-trace semantics for concrete process algebra with the priority operator. Comput. J. **30**(6) (1987) 498–506

[2] Camilleri, J., Winskel, G.: Ccs with priority choice. Inf. Comput. **116**(1) (1995) 26–37

[3] Cleaveland, R., Hennessy, M.: Priorities in process algebras. Inf. Comput. **87**(1/2) (1990) 58–77

[4] Cleaveland, R., Lüttgen, G., Natarajan, V.: Priority in process algebra. In Bergstra, J., Ponse, A., Smolka, S., eds.: Handbook of Process Algebra. Elsevier Science Publishers (February 2001) 711–765

[5] Phillips, I.: Ccs with priority guards. In Larsen, K.G., Nielsen, M., eds.: CONCUR. Volume 2154 of Lecture Notes in Computer Science., Springer (2001) 305–320

[6] Bernardo, M., Gorrieri, R.: Extended markovian process algebra. In Montanari, U., Sassone, V., eds.: CONCUR. Volume 1119 of Lecture Notes in Computer Science., Springer (1996) 315–330

[7] Hermanns, H.: Interactive Markov Chains: The Quest for Quantified Quality. Volume 2428 of Lecture Notes in Computer Science. Springer (2002)

[8] Bravetti, M., Gorrieri, R.: The theory of interactive generalized semi-markov processes. Theor. Comput. Sci. **282**(1) (2002) 5–32

[9] Bernardo, M., Gorrieri, R.: A tutorial on empa: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. Theor. Comput. Sci. **202**(1-2) (1998) 1–54

[10] Baeten, J., Bergstra, J., Klop, J.: Syntax and defining equations for an interrupt mechanism in process algebra. Fundamenta Informaticae **IX**(2) (1986) 127–168

[11] Milner, R.: A Calculus of Communicating Systems. Volume 92 of Lecture Notes in Computer Science. Springer (1980)

[12] Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, i. Inf. Comput. **100**(1) (1992) 1–40

[13] Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, ii. Inf. Comput. **100**(1) (1992) 41–77

[14] Ene, C., Muntean, T.: Expressiveness of point-to-point versus broadcast communications. In Ciobanu, G., Paun, G., eds.: FCT. Volume 1684 of Lecture Notes in Computer Science., Springer (1999) 258–268

[15] Bougé, L.: On the existence of symmetric algorithms to find leaders in networks of communicating sequential processes. Acta Inf. **25**(2) (1988) 179–201

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

[17] Phillips, I.: Ccs with priority guards. Available at http://wwwhomes.doc.ic.ac.uk/∼iccp/papers/ccspgfullrevised.pdf

[18] Milner, R.: The polyadic pi-calculus: a tutorial. In Bauer, F.L., Brauer, W., Schwichtenberg, H., eds.: Logic and Algebra of Specification. Springer-Verlag (1993) 203–246

[19] Milner, R.: Communicating and mobile systems: the π-calculus. Cambridge University Press, New York, NY, USA (1999)

[20] Nestmann, U.: What is a "good" encoding of guarded choice? Inf. Comput. **156**(1-2) (2000) 287–319

[21] Palamidessi, C., Herescu, O.M.: A randomized encoding of the pi-calculus with mixed choice. Theor. Comput. Sci. **335**(2-3) (2005) 373–404

[22] Cardelli, L., Gordon, A.D.: Mobile ambients. In Nivat, M., ed.: FoSSaCS. Volume 1378 of Lecture Notes in Computer Science., Springer (1998) 140–155

[23] Bravetti, M., Gorrieri, R., Lucchi, R., Zavattaro, G.: Quantitative information in the tuple space coordination model. Theor. Comput. Sci. **346**(1) (2005) 28–57