A Broadcast-based Calculus for Communicating Systems

Cristian ENE[1], Traian MUNTEAN[1,2]

[1] University of Marseilles
Laboratoire d'Informatique de Marseille; Parc Scientifique de Luminy - Case 925
F-13288 Marseille, France
tel: 33 - 491 82 85 32, fax: 33 - 491 82 85 11
cene@esil.univ-mrs.fr, muntean@lim.univ-mrs.fr
[2] CNRS

This paper is a revised version of [5]

Abstract. This paper presents a process calculus for reconfigurable communicating systems which has broadcast as basic communication primitive, and we provide an operational semantics for this calculus. We illustrate the calculus through some examples, and we propose three behavioral equivalences for reasoning about systems of broadcasting processes, namely, barbed equivalence, step-equivalence and labelled bisimilarity. An important result, is that all these relations coincide, providing different ways to study the equivalence/non-equivalence of two systems. Then, we provide a direct characterization for the strong congruence relation induced by these equivalences. Finally, we give a complete axiomatisation for strong congruence.

1 Introduction

Communication between processes is the main aspect of concurrency when dealing with distributed and/or parallel computing. One can specify basic communications from several points of view; primitives interactions can be, for instance, synchronous or asynchronous, associated to point-to-point or broadcast (one-to-many) message exchange protocols. The theory behind point-to-point communication is today well-established in process algebra (e.g. started with Milner's CCS and Hoare's CSP pioneering works). On the other hand more complex and higher level communication schemes, like broadcast or multicast are encountered in many applications and programming models, but they remain nevertheless poorly represented in the algebraic theory of distributed systems. We emphasize here that group interactions shall be considered as a more appropriate exchange scheme for modelling and reasoning about many communicating systems and networking applications (e.g. multimedia, data and knowledge mining, mobile computing). Group communications are, in our opinion, a more abstract and higher level concept of interaction in distributed computing than the commonly used point-to-point communications, usually expressed by handshaking message-passing primitives or by remote invocations. Broadcast has been even chosen as a hardware exchange primitive for some local networks, and in this case point-to-point message passing (when needed) is to be implemented on top of it. Primitives for broadcast programming offer several advantages: processes may interact without having explicit knowledge of each other, receivers may be dynamically added or deleted without modifying the emitter, and activity of a process can be monitored without modifying the behaviour of the observed process (this is clearly not the case with the classical rendez-vous communications). Moreover, from a theoretical point of view, it appears difficult [3] to encode broadcast in calculi based on point-to-point communications.

Thus, developing an algebraic theory for models based on broadcast communication has its own interest. Hoare's CSP [9] is based on a multiway synchronisation mechanism, but it does not make any difference between input and output. Or, in a broadcast setting the anti-symmetrie between these two kinds of actions is particularly important (in a broadcast communication there is one sender, and an unbounded or possibly empty set of receivers; this is well represented in the I/O automata of Lynch and Tuttle ([10]) where outputs are non-blocking and locally controlled, whereas inputs are externally controlled and can not be refused). In [15], Prasad introduces and develops [16] a calculus of broadcasting systems, namely CBS. His calculus, inspired by Milner's CCS ([11]), has as main goal to provide a formal model for packets broadcast in Ethernet-like communication media. It is based on broadcast, but its main limitation is that it does not allow to model reconfigurable finer topologies of networks of processes which communicate by broadcast (as dynamic group communications). It is up to the receiver to use the received value or to discard it. In [8], Hennessy and Rathke present a process calculus based on broadcast with a more restrictive input ($x \in S?p$), but the continuation process p, do not change dynamically his restrictions on further inputs; so it cannot model reconfigurable systems based on broadcast. To summarise, it seems that there is not a framework which try to analyse (at least at theoretical level) what it happens if we combine mobility and broadcast (as it is the case for processes which use group communications *à la PVM* [6], buses-based reconfigurable architectures or Packet Radio Networks).

The aim of this paper is to introduce a new process calculus, whose unique and basic communication primitive is broadcast, and which permits to model reconfigurable group communication systems.

The rest of the paper is as follows. In section 2 we present the $b\pi$-calculus (already briefly introduced in [3]) as a variant of a broadcast calculus (inspired from [16]) together with some examples. Section 3 presents three equivalences between processes, and a proof of their similar discriminative power. The section 4 is devoted to the congruence induced by the already defined equivalences. In section 5 we provide for the congruence a complete axiomatisation. Section 6, discusses related works and presents future directions of research. Due to the limited length of this paper, the proofs of presented results have been omitted; they are included in the full version of this paper [4].

2 Preliminaries

2.1 The $b\pi$-calculus

The $b\pi$-calculus is a process calculus in which broadcast is the fundamental communication paradigm. It is derived from the broadcast calculus proposed by Prasad [16], and the π-calculus proposed by Milner, Parrow and Walker [12]. It differs from the broadcast calculus, in that communications are made on channels or ports (and transmitted values are channels too), and from the π-calculus in the manner the channels are used: for broadcast communications only. Let Ch_b be a countable set of *channels*. Processes are defined by the grammar of Table 1.

where π belongs to the set of prefixes $\pi ::= x(\tilde{y}) \mid \bar{x}\tilde{y} \mid \tau$, and $\tilde{x}, \tilde{y} \subseteq Ch_b,\ x, \in Ch_b$.

Prefixes denote the basic actions of processes: τ is a silent action (which corresponds to an internal transition), $x(\tilde{y})$ is the *input* of the names $\tilde{y}$ on the channel x, and $\bar{x}\tilde{y}$ is

$\mathcal{P}_b \ni p ::=$	nil	$\pi.p$	$\nu x p$	$\langle x = y \rangle p, q$	$p_1 + p_2$	$p_1 \parallel p_2$	$A\langle \tilde{x} \rangle$	$(rec\ A\langle \tilde{x} \rangle.p)\langle \tilde{y} \rangle$

Table 1. Processes in $b\pi$-calculus

the *output* of the names $\tilde{y}$ on the channel x. nil is a process which does nothing. $\pi.p$ is the process which realize the action denoted by π and next behaves like p. $p_1 + p_2$ denotes choice, it behaves like p_1 or p_2. $\nu x p$ is the creation of a new local channel x (whose initial scope is the process p). $\langle x = y \rangle p_1, p_2$ is a process which behaves like p_1 or p_2 depending on the relation between x and y. $p_1 \parallel p_2$ is the parallel composition of p_1 and p_2. X is a process identifier whose arity is satisfied by $\langle \tilde{x} \rangle$ and $(recX\langle \tilde{x} \rangle.p)\langle \tilde{y} \rangle$ is a recursive process (this allows to represent processes with infinite behaviour), with $\tilde{x}$ containing all the free names which appear in p. In this article, we assume that X occurs guarded in any recursive definition (underneath a prefix).

The operators νx and $y(\tilde{x})$, are $x-binders$, i.e. in $\nu x p$ and $y(\tilde{x}).p$, x and $\tilde{x}$ are bound, and $bn(p)$ denotes the set of bound names of p. The free names of p are those that do not occur in the scope of any binder, and are denoted by $fn(p)$. The set of names of p is denoted by $n(p)$. *Alpha-conversion* is defined as usual.

Definition 1. Actions, *ranged over α,β are defined by the following grammar:*

$$\alpha ::= a\langle \tilde{x} \rangle \ \mid \ \nu \tilde{y} \bar{a} \tilde{x} \ \mid \ \tau \ \mid \ a :$$

where $a, x \in Ch_b$, $\tilde{x}, \tilde{y} \subseteq Ch_b$. An action is either a reception, a (possibly bound) output, or the silent action τ, denoting an internal transition. In $a\langle \tilde{x} \rangle$ and $\nu \tilde{y} \bar{a} \tilde{x}$, a is the subject of the communication and $\tilde{x}$ is its object. By extension $n(\alpha)$ ($fn(\alpha)$, $bn(\alpha)$) denotes the names (respectively free names, bound names) used in the action α ($(fn(\tau) = \emptyset$, $fn(a\langle \tilde{x} \rangle) = \{a\} \cup \tilde{x}$, $fn(\nu \tilde{y} \bar{a} \tilde{x}) = \{a\} \cup \tilde{x} \setminus \tilde{y}$, $fn(a :) = \{a\}$, $bn(\tau) = \emptyset$, $bn(a\langle \tilde{x} \rangle) = \emptyset$, $bn(\nu \tilde{y} \bar{a} \tilde{x}) = \tilde{y}$, $bn(a :) = \emptyset$, $n(\alpha) = fn(\alpha) \cup bn(\alpha))$.

We give an operational semantics for our calculus in terms of transitions over the set $\mathcal{P}_b$ of processes. Before, we define, similarly to [15], a relation $\longrightarrow \subseteq \mathcal{P}_b \times Ch_b$ denoted $p \xrightarrow{a:}$ and which can be read "p discards all outputs made on the channel a " (see Table 2).

(1) $\frac{}{nil \xrightarrow{a:}}$	(2) $\frac{}{\tau.p \xrightarrow{a:}}$	(3) $\frac{}{\bar{b}\tilde{y}.p \xrightarrow{a:}}$
(4) $\frac{b \neq a}{b(\tilde{x}).p \xrightarrow{a:}}$	(5) $\frac{p \xrightarrow{a:} \ \vee \ x = a}{\nu x p \xrightarrow{a:}}$	(6) $\frac{p_1 \xrightarrow{a:} \wedge p_2 \xrightarrow{a:}}{p_1 + p_2 \xrightarrow{a:}}$
(7) $\frac{p_1 \xrightarrow{a:}}{\langle x = x \rangle p_1, p_2 \xrightarrow{a:}}$	(8) $\frac{x \neq y \wedge \ p_2 \xrightarrow{a:}}{\langle x = y \rangle p_1, p_2 \xrightarrow{a:}}$	
(9) $\frac{p_1 \xrightarrow{a:} \ \wedge \ p_2 \xrightarrow{a:}}{p_1 \parallel p_2 \xrightarrow{a:}}$	(10) $\frac{p[(rec\ X\langle \tilde{x} \rangle.p)/X, \tilde{y}/\tilde{x}] \xrightarrow{a:}}{(rec\ X\langle \tilde{x} \rangle.p)\langle \tilde{y} \rangle \xrightarrow{a:}}$	

Table 2. The "discard" relation

Intuitively, a process ignores all the communications made on the channels it is not listening. nil, $\tau.p$ or $\bar{b}\tilde{y}.p$ discard any communication. A process waiting for a message on a channel b, discards actions on the other channels a with $a \neq b$. In rule (5) a the condition $x = a$ expres the possibility that a does not occur free in p. Rules (6) to (10) follow the structure of the term.

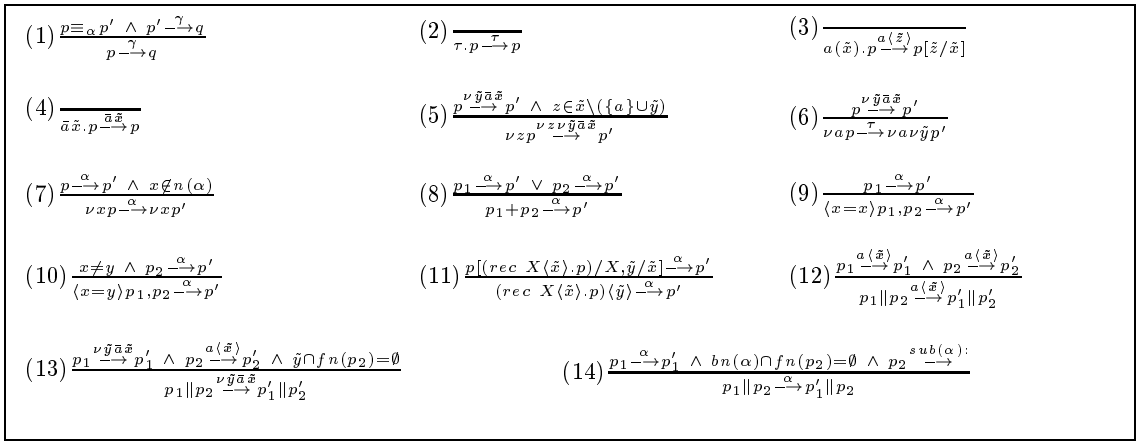

Table 3. Operational semantics of $b\pi$-calculus

To simplify the presentation, we extend sub and we denote $sub(\tau) = obj(\tau) = \tau$, and $p \xrightarrow{\tau:}$ for any process p.

Definition 2. Transition system *The operational semantics of $b\pi$-calculus is defined as a labelled transition system defined over the set $\mathcal{P}_b$ of processes. The judgement $p \xrightarrow{\alpha} p'$ means that the process p is able to perform action α and to evolve next to p'. The operational semantics is given in Table 3 (we omitted the symmetric versions of rules* (11) *and* (12)*).*

A communication between processes is performed through unbuffered broadcast. Compared to π-calculus, outputs are non-blocking, i.e. there is no need of a receiving process. One of the processes broadcasts an output and the remaining processes either receive or ignore the sending, according to whether they are "listening" or not on the channel which serves as support for the output. A process which "listens" to a channel a, cannot ignore any value sent on this channel.

The operational semantics is an early one, i.e. the bound names of an input are instantiated as soon as possible, in the rule for input.

Rule (1) allows to identify process which are alpha - convertible. Rules (2) to (4) are straightforward and they have the same signification as in π-calculus. Rule (5) states that when a local channel name is emitted, the related output has to be bound. The extrusion is more complex that in π -calculus, as more processes could learn the existence of a fresh name in a single comunication. Rule (6) does not exist in π -calculus; it establish the scope of the names exported on the channel a. Rules (7) to (11) have the same meaning as in π -calculus. Rules (12) and (13) are specific to broadcast; the same message can be received

by more processes in a single comunication. In rule (14), a proces which does not listen on a channel a, remains unchanged during a communcation made on this channel. As usual we shall use the following notations:

- $\stackrel{\epsilon}{\Longrightarrow} \stackrel{def}{=} (\stackrel{\tau}{\longrightarrow})^*, \quad \stackrel{\alpha}{\Longrightarrow} \stackrel{def}{=} \stackrel{\epsilon}{\Longrightarrow} \stackrel{\alpha}{\longrightarrow} \stackrel{\epsilon}{\Longrightarrow}$, if $\alpha \neq \tau$,
- $\stackrel{\oslash}{\longrightarrow} \stackrel{def}{=} \left\{ \stackrel{\alpha}{\longrightarrow} \middle| \alpha \textit{ is an output or } \alpha = \tau \right\}$,
- $\stackrel{\oslash}{\Longrightarrow} \stackrel{def}{=} (\stackrel{\oslash}{\longrightarrow})^*$.

Sometimes, we shall use, as in [8], we use $p \stackrel{\alpha:}{\longrightarrow} p$ instead of $p \stackrel{a:}{\longrightarrow}$, if $sub(\alpha) = a$ and we denote $p \stackrel{a(b)?}{\longrightarrow} p'$ to stand for either $p \stackrel{a(b)}{\longrightarrow} p'$ or $p \stackrel{a(b):}{\longrightarrow} p'$. Also, we shall omit the trail nil. We shall also use the notation $\bar{a}(x)$ to stand for the bound output $\nu x \bar{a} x$.

We shall use the following results in the next sections.

Lemma 1.
1. *If* $p \stackrel{\nu \tilde{y} \bar{a} \tilde{x}}{\longrightarrow} p'$, *then* $fn(p') \subseteq fn(p) \cup \{\tilde{y}\}$ *et* $(\tilde{x} \setminus \tilde{y}) \subseteq fn(p)$.
2. *If* $p \stackrel{a\langle \tilde{z} \rangle}{\longrightarrow} p'$, *then* $fn(p') \subseteq fn(p) \cup \{\tilde{z}\}$.
3. *If* $p \stackrel{\tau}{\longrightarrow} p'$, *then* $fn(p') \subseteq fn(p)$.

Proof
The proof is by a simultaneous induction on the inferrence of the transition $p \stackrel{\alpha}{\longrightarrow} p'$.
□ *Lemma* 1

Corollary 1. *If* $p \stackrel{\epsilon}{\Longrightarrow} p'$, *then* $fn(p') \subseteq fn(p)$.

Proof The corollary is a consequence by induction on the number of transitions in $p \stackrel{\epsilon}{\Longrightarrow} p'$.

2.2 Examples

In this subsection we give some examples where broadcast is the basic communication primitive. The ability to send and receive "names on names" as in π -calculus is very useful, and allows us in Example 2 (Detecting inconsistencies for transactions systems) to give a description which does not depend on the number of copies present in the system.

Example 1 A distributed algorithm for cycle-detection

We present a distributed algorithm for cycle-detection in a directed graph. *Detector* is a process which listens to new edges of the graph on a channel i, and spawns for each received pair of names (source and destination of edge) a new "edge manager" *Edge_manager*. An edge manager, broadcasts a personal token v (using the mechanism of name-generation). Next, for each received token w, if it receives the own token v, a cycle is detected and a signal is sent on o, otherwise, it propagates the token further (along the paths of the graph). This is done by using the channels a and b, which denote an edge (a, b) in the graph. Hence, any token received on a, and different from v is transmitted on b.

$$Detector\langle i, o \rangle \stackrel{def}{=}$$

$$i(x).i(y).(Detector\langle i,o\rangle \parallel Edge\_manager\langle o,x,y\rangle))$$

$$Edge\_manager\langle o,a,b\rangle \stackrel{def}{=}$$

$$\nu u((recY\langle b,u\rangle.\{\bar{b}u.Y\langle b,u\rangle\})\langle b,u\rangle \parallel$$

$$(recX\langle o,a,b,u\rangle.\{a(w).$$

$$(\langle u=w\rangle\bar{o}.nil,(\bar{b}w.nil \parallel X\langle o,a,b,u\rangle\})\langle o,a,b,u\rangle))))$$

Example 2 Detecting inconsistencies for transactions systems

We extend the previous example to give an implementation in $b\pi$-calculus of a fully distributed algorithm for detecting inconsistencies in partitioned distributed databases (our implementation is inspired from [1]).

In a replicated database, there exist several copies of each data item, with copies located at distinct sites in the system. We suppose that the database becomes partitioned (partitions p_j with $j = 1,\ldots,n$).

We allow transactions to continue to execute, but when the network is reconnected (simulated in our implementation by a broadcast on the channel "$unif$") , we have to check for inconsistencies. The idea is to construct a precedence graph that captures the temporal partial order between transactions. Then, the database is consistent iff the precedence graph contains no cycle. The vertices of the graph are all the transactions. An edge $< t,p >\rightarrow< t_1,p_1 >$ indicates that transaction t occurred before transaction t_1, where p,p_1 indicate the partition in which the transactions were executed.

Such an edge exists iff one of the following holds:

1. t read a data item i that was later written by t_1 and $p = p_1$
2. t write a data item i that was later read or written by t_1 and $p = p_1$
3. t read a data item i that was written by t_1 and $p \neq p_1$

The database can be simulated by:

$$Database \stackrel{def}{=}$$

$$\prod_{j=1,k}[\prod_{l=1,n(j)} Item\langle j_1,j_2,p_{j_l},unif,Val\rangle]$$

where j_1 and j_2 are channels associated with data j, each data item j having $n(j)$ copies, and $p_{j_l} \in \{p_1,\ldots,p_{n(j)}\}$ is the corresponding partition.

An item manager waits for transactions; for each new transaction, it forks a new transaction manager, and serves the user which was making the request. A transaction for a data item i, is an output on the channel i_1, and contains the transaction identifier t_1, the type (read or write), the partition affected, the return channel, and a value (which make sense for a write transaction).

$Item\langle i_1,i_2,p,unif,Val,w\rangle \stackrel{def}{=}$
$unif(p_1).Item\langle i_1,i_2,p_1,unif,Val\rangle +$
$i_1(t_1,type,p_1,req,V).\langle p_1 = p\rangle$
$\{\langle type = w\rangle$
$[Item\langle i_1,i_2,p,unif,V,w\rangle \parallel Tr\_Man_w\langle i_1,i_2,p,unif,t_1\rangle],$

$[Item\langle i_1, i_2, p, unif, Val, w\rangle \parallel Tr\_Man_r\langle i_1, i_2, p, unif, t_1\rangle \parallel \overline{req}Val + req(V)]\},$
$Item\langle i_1, i_2, p, unif, Val, w\rangle$

A transaction manager generates a new edge manager for each new ongoing transaction which affects the same data item on the same partition (cases 1. or 2.).

$Tr\_Man_w\langle i_1, i_2, p, unif, t\rangle \stackrel{def}{=}$
$i_1(t_1, type, p_1, req, V).\{\langle p_1 = p\rangle$
$[Tr\_Man_w\langle i_1, i_2, p, unif, t\rangle \parallel unif(p).Edge\_manager\langle error, t, t_1\rangle],$
$Tr\_Man_w\langle i_1, i_2, p, unif, t\rangle\}+$
$unif(p_1).STr\_Man_w\langle i_2, p, t, w\rangle$

$Tr\_Man_r\langle i_1, i_2, p, unif, t\rangle \stackrel{def}{=}$
$i_1(t_1, type, p_1, req, V).\{\langle p_1 = p\rangle$
$[Tr\_Man_r\langle i_1, i_2, p, unif, t\rangle \parallel \langle type = w\rangle unif(p).Edge\_manager\langle error, t, t_1\rangle, nil],$
$Tr\_Man_r\langle i_1, i_2, p, unif, t\rangle\}+$
$unif(p_1).STr\_Man_r\langle i_2, p, t, r\rangle$

Once the network reconnected, transaction managers change their behaviour, trying to detect edges of the third kind. In addition, if there are two transactions which have written the same data item in two distinct partitions, then an error is detected (two contrary edges between two vertices). $STr\_Man_w\langle i_2, p, t\rangle \stackrel{def}{=}$
$i_2(t_1, type, p_1).\{\langle p_1 = p\rangle$
$STr\_Man_w\langle i_2, p, t\rangle,$
$\langle type = w\rangle\overline{error}, [STr\_Man_w\langle i_2, p, t\rangle \parallel Edge\_manager\langle error, t_1, t\rangle]\}+$
$\bar{i}_2[t, w, p].STr\_Man_w\langle i_2, p, t\rangle$

$STr\_Man_r\langle i_2, p, t\rangle \stackrel{def}{=}$
$i_2(t_1, type, p_1).\{\langle p_1 = p\rangle$
$STr\_Man_r\langle i_2, p, t\rangle,$
$STr\_Man_r\langle i_2, p, t\rangle \parallel \langle type = r\rangle nil, Edge\_manager\langle error, t, t_1\rangle]\}+$
$\bar{i}_2[t, r, p].STr\_Man_r\langle i_2, p, t\rangle$

We note that this example uses the entire expressiveness power of our calculus. The same data item can be replicated (for reliability or efficiency reasons) and a transaction t can affect several data items; in this case broadcast is a quite natural communication primitive. In the same time, the ability to send and receive channel names across channels is used by item managers to fork new transaction managers corresponding to the received identifier.

Example 3 Semantics of group communication primitives

$b\pi$-calculus provide a framework to specify and analyse systems which interact by a broadcast (or multicast) mechanism combined with mobility of processes (names, addresses). We take here, as an example programs which use communication primitives of PVM-like libraries ([6]). PVM is a software system that permits a network of heterogeneous computers to be used as a single parallel computer (the *virtual machine*). Thus large computational problems may be solved using the power of many computers. PVM supplies functions to automatically start up tasks on the virtual machine, and allows task to communicate (by point-to-point or group communications) and synchronize with each-other.

The interesting part is the simple simulation of group communication primitives, which seems difficult to express in a process algebra based on point-to-point communications ([3]). Moreover, even in CBS ([16]) it is unclear how one can implement the primitives which permit to have dynamic groups (processes can freely join or leave a group, once they have knowledge of the name of the group). We present just a few communication primitives specific to concurrent applications (for the interpretation of imperative features in process algebra, see for example [17]):

$I ::= send(a,m) \mid bcast(g,m) \mid x = receive() \mid x = newgroup() \mid joingroup(g) \mid leavegroup(g) \mid x = spawn(Q)$

$P ::= I \mid I; P$

A process (or a task) is a sequence of actions. An action is either an output of a message m (to another process a or to a group g of processes), or an input (from the own buffer) of a message (stored after in variable x), or a creation of a new group g, or a joining to a group g, or a leaving of a group g or a spawning of a child Q. Once a process become member of a group g, it receives any message sent to that group. Communications are asynchronous, in that outputs are non-blocking (messages being stored in the buffers of receivers). For the sake of simplicity, we suppose that there is no guarantee in what concerns the order of messages' arrival.

Then, a possible encoding $\{P\}_a$ of a process P of address (pid) a is given below:

$$\{P\}_a \stackrel{def}{=} \nu r_a \nu k_a (Pool\langle a, r_a, k_a\rangle \parallel [\![P]\!]_{r_a,\emptyset})$$

$$Pool\langle a,r,k\rangle \stackrel{def}{=} k + a(x).(Pool\langle a,r,k\rangle \parallel Cell\langle r,x\rangle)$$

$$Cell\langle r,x\rangle \stackrel{def}{=} r(c).(\bar{c}x + c(y).Cell\langle r,x\rangle)$$

$$[\![x = receive(); P]\!]_{r,M} \stackrel{def}{=} \nu t(\bar{r}t \parallel t(x).[\![P]\!]_{r,M})$$

$$[\![send(a,m); P]\!]_{r,M} \stackrel{def}{=} \nu t(\bar{a}m.\bar{t} \parallel t.[\![P]\!]_{r,M})$$

$$[\![bcast(g,m); P]\!]_{r,M} \stackrel{def}{=} \nu t(\bar{g}m.\bar{t} \parallel t.[\![P]\!]_{r,M})$$

$$[\![joingroup(g); P]\!]_{r,M} \stackrel{def}{=} \nu t \nu k_g(\bar{t}k_g.Pool\langle g,r,k_g\rangle \parallel k_g).[\![P]\!]_{r,M\cup\{(g,k_g)\}})$$

$$[\![g = newgroup(); P]\!]_{r,M} \stackrel{def}{=} \nu t \nu g \nu k_g(\bar{t}g.\bar{t}k_g.Pool\langle g,r,k_g\rangle \parallel t(g).t(k_g).[\![P]\!]_{r,M\cup\{(g,k_g)\}})$$

$$[\![leavegroup(g); P]\!]_{r,M\cup\{(g,k_g)\}} \stackrel{def}{=} \nu t(\overline{k_g}.\bar{t}.nil \parallel t.[\![P]\!]_{r,M})$$

$$[\![x = spawn(Q); P]\!]_{r,M} \stackrel{def}{=} \nu a \nu t(\{Q\}_a \parallel \bar{t}a \parallel t(x).[\![P]\!]_{r,M})$$

$$[\![STOP]\!]_{r,\{(g_1,k_{g_1}),\ldots,(g_n,k_{g_n})\}} \stackrel{def}{=} \overline{k_{g_1}}.\ldots.\overline{k_{g_n}}.\bar{r}.nil)$$

The translations of "send" and "bcast" primitives are similar, but actually, for a channel a which appears as the argument of the "send" primitive, there is exactly one receiver, whereas for a channel g which appears as argument for the "bcast" primitive, there can be zero or many receivers (depending on the actual number of members in group g).

3 Bisimulations

We define now appropriate tools allowing to reason about processes. Bisimulations have been successfully used in processes algebra to compare two systems according to their operational ability to simulate each other. For the rest of the paper, we shall use the term process to stand for a closed process (which does not contain free identifiers), and we shall specify explicitly open processes.

3.1 Barbed Equivalences

Barbed bisimulation was firstly introduced by Sangiorgi and Milner (in [13], [18]) for the π-calculus. It is a natural relation which is easy definable in various calculi (or rewriting systems): it suffices to define a observability predicate and then to distinguish between two processes whenever they are not similarly observable, or their similar observability is not preserved by reduction (or rewriting). Barbed bisimulations were already used for calculi based on broadcast [8], but in a calculus where communications are made on a global *ether* rather than explicit channels.

Barbed bisimulation describes a relation between processes which can make the same "visible" actions, and whenever one can silently progress, so can the other, evolving to a process preserving the relation. It remains to precise some parts of this informal definition. In a broadcast calculus, outputs are visible (if we are listening a process on a channel, we receive any value it sends on it), while inputs are not (as sending is not blocking, we do not know if the observed process was receiving or discarding our value). We write $p \downarrow_a$ ($p \Downarrow_a$) if $p \xrightarrow{\alpha} p'$ (and respectively $p \stackrel{\alpha}{\Longrightarrow} p'$) for some output α of subject a.

Definition 3. Barbed Bisimulation
A symmetric relation S over the set $\mathcal{P}_b$ (of processes) is a **weak barbed bisimulation** *if whenever $(p,q) \in S$, then*

- *if $p \xrightarrow{\tau} p'$, then $\exists q'$ such that $q \stackrel{\epsilon}{\Longrightarrow} q'$ and $(p',q') \in S$,*
- *$\forall a \in Ch_b$, if $p \downarrow_a$ then $q \Downarrow_a$.*

Two processes p and q are **weak barbed bisimilar**, *noted $p \approx_b q$, if $(p,q) \in S$ for some barbed bisimulation S. The* **strong barbed bisimilarity** *$\sim_b$ is defined similar by replacing $\Longrightarrow$ by $\longrightarrow$ and $\Downarrow$ by $\downarrow$.*

We prove some equivalences between processes considered syntactical congruent in some classical presentation of π-calculus.

Next lemma resumes a few properties of the relation $\sim_b$.

Lemma 2. *$\sim_b$ satisfies the following properties:*

(a) $p \equiv_\alpha q$ *implies* $p \sim_b q$
(b) $p \parallel nil \sim_b p$
(c) $p \parallel q \sim_b q \parallel p$
(d) $(p \parallel q) \parallel r \sim_b p \parallel (q \parallel r)$
(e) $p + nil \sim_b p$

(f) $p + q \sim_b q + p$
(g) $(p + q) + r \sim_b p + (q + r)$
(h) $\nu x p \sim_b p$ *if* $x \notin fn(p)$
(i) $\nu y \nu x p \sim_b \nu x \nu y p$ *si* $x \neq y$
(j) $(\nu x p) \parallel q \sim_b \nu x(p \parallel q)$ *if* $x \notin fn(q)$
(k) $(\nu x p) + q \sim_b \nu x(p + q)$ *if* $x \notin fn(q)$
(l) $\langle y = z \rangle (\nu x p), q \sim_b \nu x(\langle y = z \rangle p, q)$ *if* $x \notin fn(q) \cup \{y, z\}$

Proof
Consequence of Lemma 6 et 10.

□ *Lemma* 2

Next lemma proves an interesting property of barbed bisimilarity: barbed bisimilarity is preserved by parallelism.

Lemma 3. $\sim_b$ *and* $\approx_b$ *are preserved by parallelism:*

- $p \sim_b q$ *implies* $\forall r,\ p \parallel r \sim_b q \parallel r$,
- $p \approx_b q$ *implies* $\forall r,\ p \parallel r \approx_b q \parallel r$,

Proof
We prove (for example), just the weak case. It suffices to prove that the relation $\mathcal{R}$ where
$\mathcal{R} \stackrel{def}{=} \{(p \parallel r, q \parallel r) \mid p, q, r \in \mathcal{P}_b,\ p \approx_b q\}$
is a weak barbed bisimulation.

- Let $p \parallel r \downarrow_a$.
  Then we have either $p \downarrow_a$, or $r \downarrow_a$.
  If $p \downarrow_a$, since $p \approx_b q$ we obtain $q \Downarrow_a$ and by successive applications of rule (14), we obtain $(q \parallel r) \Downarrow_a$. If $r \downarrow_a$, then obviously $(q \parallel r) \downarrow_a$.
- Let $p \parallel r \stackrel{\tau}{\longrightarrow} s$ be a silently transition of p.
  Then, the last applied rule used to infer this transition is either the rule (14) or its symmetrical.
  If the last applied rule is the rule (14), then $s \equiv p' \parallel r$ and $p \stackrel{\tau}{\longrightarrow} p'$. Since $p \approx_b q$, we obtain $q \stackrel{\epsilon}{\Longrightarrow} q'$ with $p' \approx_b q'$. By successive applications of rule (14), we obtain $q \parallel r \stackrel{\epsilon}{\Longrightarrow} q' \parallel r$.
  If the last applied rule is the symmetrical of the rule (14), then $s \equiv p \parallel r'$ and $r \stackrel{\tau}{\longrightarrow} r'$. By an application of the symmetrical of the rule (14) (14) we obtain $q \parallel r \stackrel{\tau}{\longrightarrow} q \parallel r'$.

□ *Lemma* 3

Remark 1. Contrary to π-calculus, in the $b\pi$-calculus it is no more true that $p \approx_b q$ $(p \sim_b q)$, implies $\nu a p \approx_b \nu a q$ $(\nu a p \sim_b \nu a q)$ for any channel a. For example, it suffices to take $p_0 \stackrel{def}{=} \bar{a}b$ and $q_0 \stackrel{def}{=} \bar{a}b.\bar{c}d$ which are strongly barbed bisimilar, whilst $\nu a p_0$ et $\nu a q_0$ are not even barbed bisimilar.

Lemma 3 and Remark 1 present interesting differencies between $b\pi$-calculus and π-calculus. In our model, barbed bisimilarity is preserved by parallelism, but it is not preserved by restriction. In π-calculus, barbed bisimilarity is preserved by restriction but it is not preserved by parallelism.

Since barbed bisimilarity it is not preserved by restriction and can not make any distinction between $\alpha.p$ and $\alpha.q$ for any $p, q, \alpha \neq \tau$, it is necessary to close it over various classes of contexts. A *context* (defined in Table 4) is a term containing a single "hole", such that placing another term in the hole defines a valid term. A *static context* (defined in Table 5) is a context which is build using just "the hole", constants, parallelism and name creation.

$$C ::= \bullet \mid \alpha.C \mid \nu x C \mid \langle x = y \rangle C, p \mid \langle x = y \rangle p, C \mid C + p \mid$$
$$p + C \mid C \parallel p \mid p \parallel C \mid (rec\ A\langle \tilde{x} \rangle.C)\langle \tilde{y} \rangle$$
where $p \in \mathcal{P}_b$ et $\alpha ::= x(\tilde{y}) \mid \bar{x}\tilde{y} \mid \tau$ with $x \in Ch_b, \tilde{y} \subseteq Ch_b$.

Table 4. Family of contexts $\mathcal{C}on$

$$C ::= \bullet \mid \nu x C \mid C \parallel p \mid p \parallel C$$
where $p \in \mathcal{P}_b$ et $\alpha ::= x(\tilde{y}) \mid \bar{x}\tilde{y} \mid \tau$ with $x \in Ch_b, \tilde{y} \subseteq Ch_b$.

Table 5. Static contexts $\mathcal{C}on_s$

Definition 4. Barbed Equivalence

- *Two processes p and q are* **weak barbed equivalent**, *shortly $p \overset{e}{\approx}_b q$, if for every static context $C \in \mathcal{C}on_s$, $C[p] \approx_b C[q]$.*
- **Strong barbed equivalence**$(\overset{e}{\sim}_b)$ *is obtained similarly from strong barbed bisimilarity over processes.*

By a classical reasoning, we can prove that $\approx_b$, $\sim_b$, $\overset{e}{\approx}_b$ et $\overset{e}{\sim}_b$ are equivalence relations.

Remark 1 implies that barbed equivalence does no more coincide with the closure of barbed bisimilarity with respect to arbitrary testers (as is the case in π-calculus).

3.2 Step Equivalences

In our calculus $\overset{\oslash}{\Longrightarrow}$ is the rewriting and not $\xrightarrow{\tau}$. This is the reason which push us to define another characterization of barbed equivalence, still based on testers.

We denote $p \Downarrow_a^{\phi}$ if $p \overset{\oslash}{\Longrightarrow} \xrightarrow{\alpha} p'$ for some output α of subject a.

Definition 5. Step-Bisimulation

A symmetric relation S over the set $\mathcal{P}_b$ (of processes) is a **weak step-bisimulation** *if whenever $(p,q) \in S$, then*

- *if $p \xrightarrow{\oslash} p'$, then $\exists q'$ such that $q \stackrel{\oslash}{\Longrightarrow} q'$ and $(p',q') \in S$,*
- *$\forall a \in Ch_b$, if $p \downarrow_a$ then $q \Downarrow_a^{\phi}$.*

Two processes p and q are **weak step-bisimilar**, *noted $p \approx_\phi q$, if $(p,q) \in S$ for some step-bisimulation S. The* **strong step-bisimilarity** *$\sim_\phi$ is defined similar, but replacing $\stackrel{\oslash}{\Longrightarrow}$ by $\xrightarrow{\oslash}$ and $\Downarrow^{\phi}$ by $\downarrow$.*

Intuitively, step-bisimilarity identifies processes which can broadcast messages on the same set of channels (immediately or after some autonomous transitions), and whenever one can progress without implicating the environment, so can the other, evolving to a process preserving the relation. In fact, for our calculus, step-bisimilarity is more natural than barbed bisimilarity since the real reduction is $\xrightarrow{\oslash}$ ($\stackrel{\oslash}{\Longrightarrow}$) and not $\xrightarrow{\tau}$ (or $\xrightarrow{\epsilon}$) as in π-calculus. Like barbed bisimilarity, step-bisimilarity is a very weak relation (it is not preserved by parallel composition), so it is necessary to close it with respect to observers.

From now on, we shall also use ϕ - bisimilarity to stand for step-bisimilarity.

As for barbed bisimilarity, we have the following step-bisimilarities.

Lemma 4. *$\sim_\phi$ satisfies the following properties:*

(a) $p \equiv_\alpha q$ *implies* $p \sim_\phi q$
(b) $p \parallel nil \sim_\phi p$
(c) $p \parallel q \sim_\phi q \parallel p$
(d) $(p \parallel q) \parallel r \sim_\phi p \parallel (q \parallel r)$
(e) $p + nil \sim_\phi p$
(f) $p + q \sim_\phi q + p$
(g) $(p+q)+r \sim_\phi p + (q+r)$
(h) $\nu x p \sim_\phi p$ *if* $x \notin fn(p)$
(i) $\nu y \nu x p \sim_\phi \nu x \nu y p$ *if* $x \neq y$
(j) $(\nu x p) \parallel q \sim_\phi \nu x (p \parallel q)$ *if* $x \notin fn(q)$
(k) $(\nu x p) + q \sim_\phi \nu x (p+q)$ *if* $x \notin fn(q)$
(l) $\langle y = z \rangle (\nu x p), q \sim_\phi \nu x (\langle y = z \rangle p, q)$ *if* $x \notin fn(q) \cup \{y, z\}$

Proof

Consequence of Lemma 6 and 11.

□ *Lemma* 4

As for barbed bisimilarity, step-bisimilarity is a very weak relation (it is not preserved by parallelism, nor by restriction).

Remark 2.

1. $\approx_\phi$ ($\sim_\phi$) is not preserved by $\parallel$;
2. $\approx_\phi$ ($\sim_\phi$) is not preserved by ν;
3. $\sim_b$ ($\approx_b$) et $\sim_\phi$ ($\approx_\phi$) are incomparable.

Proof

1. Let $p_1 \stackrel{def}{=} \bar{b} + \tau.\bar{c}$, $q_1 \stackrel{def}{=} \bar{b} + \bar{b}.\bar{c}$ and $r_1 \stackrel{def}{=} b + \bar{a}$.
   Then $p_1 \sim_\phi q_1$. On the contrary $(p_1 \parallel r_1) \not\approx_\phi (q_1 \parallel r_1)$ because $q_1 \parallel r_1$ can not simulate the transition $(p_1 \parallel r_1) \stackrel{\tau}{\longrightarrow} (\bar{c} \parallel b + \bar{a})$.
2. Let $p_2 \stackrel{def}{=} \bar{b}a.\bar{a}$, $q_2 \stackrel{def}{=} \bar{b}c.\bar{a}$.
   Then $p_2 \sim_\phi q_2$. On the contrary $\nu a p_2 \not\approx_\phi \nu a q_2$, because $\nu a q_2$ can not simulate the transition $\nu a p_2 \stackrel{\nu a \bar{b} a}{\longrightarrow} \bar{a}$.
3. Since $\nu a p_2 \sim_b \nu a q_2$ and $\nu a p_2 \not\approx_\phi \nu a q_2$, we obtain

   $$\sim_b \not\subseteq \approx_\phi .$$

   We have $p_1 \not\approx_b q_1$ since q_1 can not match the transition $p_1 \stackrel{\tau}{\longrightarrow} \bar{c}$. Since $p_1 \sim_\phi q_1$ we obtain

   $$\sim_\phi \not\subseteq \approx_b .$$

   To obtain the assertion 3. it suffices to use the inequalities $\sim_b \subseteq \approx_b$ and $\sim_\phi \subseteq \approx_\phi$.

□ *Remark* 2

Remark 2 justifies the definition of the closure of step-bisimilarity with respect to static contexts.

Definition 6. Step-Equivalence

- *Two processes p and q are* **weak step-equivalent**, *shortly $p \stackrel{e}{\approx}_\phi q$, if for every static context C, $C[p] \approx_\phi C[q]$.*
- **Strong step-equivalence**$(\stackrel{e}{\sim}_\phi)$ *is obtained similarly from strong step-bisimilarity over processes.*

Using some convenient contexts, we can prove that step-equivalence implies barbed equivalence.

Lemma 5.
1) $p \stackrel{e}{\sim}_\phi q$ implies $p \sim_b q$
2) $p \stackrel{e}{\approx}_\phi q$ implies $p \approx_b q$.

Proof

We proof only the weak case.

Let $p \stackrel{e}{\approx}_\phi q$ and let $M \stackrel{def}{=} \{a' \mid a \in fn(p,q)\}$ a set of channels such that $M \cap fn(p,q) = \emptyset$ and c a new channel such that $c \notin M \cup fn(p,q)$. Let $T \stackrel{def}{=} \Sigma_{a \in fn(p,q)} a(x).\bar{a'} + \bar{c}$.

We shall prove that the relation

$$\mathcal{R}^T \stackrel{def}{=} \left\{(p',q') \;\middle|\; fn(p',q') \subseteq fn(p,q),\ p' \parallel T \approx_\phi q' \parallel T\right\}$$

is a weak barbed bisimilarity.

Let $(p',q') \in \mathcal{R}^T$. Then $fn(p',q') \subseteq fn(p,q),\ p' \parallel T \approx_\phi q' \parallel T$.

- Let $p' \downarrow_d$.
  It follows that $d \neq c$, and $p' \stackrel{\alpha}{\longrightarrow}$ for an output α of subject d. We obtain $p' \parallel T \stackrel{\alpha}{\longrightarrow} (p'' \parallel \bar{d'})$. Since $p' \parallel T \approx_\phi q' \parallel T$, we obtain that $q' \parallel T \stackrel{\emptyset}{\Longrightarrow} r$ where $(p'' \parallel \bar{d'}) \approx_\phi r$. Then $r = (q''' \parallel \bar{d'})$, where $q' \stackrel{\epsilon}{\Longrightarrow} \stackrel{\gamma}{\longrightarrow} q'' \stackrel{\emptyset}{\Longrightarrow} q'''$, for an output γ of subject d.

– Let $p' \xrightarrow{\tau} p''$.
Then $p' \parallel T \xrightarrow{\tau} p'' \parallel T$. Since $p' \parallel T \approx_\phi q' \parallel T$, it follows that $q' \parallel T \stackrel{\emptyset}{\Longrightarrow} r$, where $p'' \parallel T \approx_\phi r$. If $\phi \neq \epsilon$, it is easy to remark that T had to participate in the transition, and that $r \not\Downarrow_c^\phi$, whilst $(p'' \parallel T)\downarrow_c$, contradiction with $p'' \parallel T \approx_\phi r$. So $r = q'' \parallel T$ where $q' \stackrel{\epsilon}{\Longrightarrow} q''$. From the corollary 1 we obtain also that $fn(p'', q'') \subseteq fn(p', q')$.

□ *Lemma* 5

Corollary 2.
1) $p \overset{e}{\sim}_\phi q$ *implies* $p \overset{e}{\sim}_b q$
2) $p \overset{e}{\approx}_\phi q$ *implies* $p \overset{e}{\approx}_b q$.

Proof
We prove only the weak case.

Let $p \overset{e}{\approx}_\phi q$ and let $C_1[\bullet]$ an arbitrary static context. For any static context $C[\bullet]$, since $C[C_1[\bullet]]$ is a static context, by the Definition 6 we obtain that, $C[C_1[p]] \approx_\phi C[C_1[q]]$. We also obtain $C_1[p] \overset{e}{\approx}_\phi C_1[q]$.

By the Lemma 5, we obtain that $C_1[p] \approx_b C_1[q]$. By the Definition 4 we obtain that $p \overset{e}{\approx}_b q$.

□ *Corollary* 2

3.3 Labelled Bisimulations

In order to prove that two processes p and q are not (weak) step-equivalent (or weak barbed equivalent), it is enough to find a suitable static context C, such that $C[p]$ and $C[q]$ are not (weak) step bisimilar (respectively (weak) barbed bisimilar). But the converse (i.e. proving the step-equivalence or the barbed equivalence) is much harder to prove in general just using the given definitions; indeed, this requires a quantification over all the contexts. Thus, it is interesting to have another way to directly prove that two systems are equivalent.

Definition 7. Weak labelled bisimulations *A symmetric relation S over the set $\mathcal{P}_b$ (of processes) is a* **(weak) bisimulation** *if whenever $(p, q) \in S$, then*

1) if $p \xrightarrow{\tau} p'$, *then* $\exists q'$ *such that* $q \stackrel{\epsilon}{\Longrightarrow} q'$ *and* $(p', q') \in S$,
2) if $p \xrightarrow{\nu\tilde{b}\bar{a}\tilde{c}} p'$ *and* $\tilde{b} \cap fn(p, q) = \emptyset$, *then* $\exists q'$ *such that* $q \stackrel{\nu\tilde{b}\bar{a}\tilde{c}}{\Longrightarrow} q'$ *and* $(p', q') \in S$,
3) if $p \xrightarrow{a\langle\tilde{b}\rangle?} p'$ *then* $\exists q'$ *such that* $q \stackrel{\epsilon}{\Longrightarrow} \bullet \xrightarrow{a\langle\tilde{b}\rangle?} \bullet \stackrel{\epsilon}{\Longrightarrow} q'$ *and* $(p', q') \in S$.

Two processes p and q are **weak bisimilar**, *noted $p \approx q$, if $(p, q) \in S$ for some weak bisimulation S.*

The **strong bisimilarity** $\sim$ is defined similar, by substituting $\longrightarrow$ to $\Longrightarrow$ in conditions 1) and 2), and replacing condition 4) by "if $p \xrightarrow{a\langle\tilde{b}\rangle?} p'$ then $\exists q'$ such that $q \xrightarrow{a\langle\tilde{b}\rangle?} q'$ and $(p', q') \in S$".

Definition 8. Strong labelled bisimulations *A symmetric relation S over the set $\mathcal{P}_b$ (of processes) is a* **strong bisimulation** *if whenever $(p,q) \in S$, then*

1) if $p \xrightarrow{\tau} p'$, then $\exists q'$ such that $q \xrightarrow{\tau} q'$ and $(p',q') \in S$,

2) if $p \xrightarrow{\nu \tilde{b} \bar{a} \tilde{c}} p'$ and $\tilde{b} \cap fn(p,q) = \emptyset$, then $\exists q'$ such that $q \xrightarrow{\nu \tilde{b} \bar{a} \tilde{c}} q'$ and $(p',q') \in S$,

3) if $p \xrightarrow{a\langle \tilde{b} \rangle ?} p'$ then $\exists q'$ such that $q \xrightarrow{a\langle \tilde{b} \rangle ?} q'$ and $(p',q') \in S$.

Two processes p and q are **strong bisimilar**, *noted $p \approx q$, if $(p,q) \in S$ for some strong bisimulation S.*

By classical arguments [11], we can prove that $\sim$ and $\approx$ are equivalence relations.

As for barbed bisimilarity and for ϕ - bisimilarity, we have for the strong labelled bisimilarity the next lemma.

Lemma 6. *$\sim$ satisfies the following properties:*

(a) $p \equiv_\alpha q$ *implies* $p \sim q$
(b) $p \parallel nil \sim p$
(c) $p \parallel q \sim q \parallel p$
(d) $(p \parallel q) \parallel r \sim p \parallel (q \parallel r)$
(e) $p + nil \sim p$
(f) $p + q \sim q + p$
(g) $(p + q) + r \sim p + (q + r)$
(h) $\nu x p \sim p$ *if* $x \notin fn(p)$
(i) $\nu y \nu x p \sim \nu x \nu y p$ *if* $x \neq y$
(j) $(\nu x p) \parallel q \sim \nu x (p \parallel q)$ *if* $x \notin fn(q)$
(k) $(\nu x p) + q \sim \nu x (p + q)$ *if* $x \notin fn(q)$
(l) $\langle y = z \rangle (\nu x p), q \sim \nu x (\langle y = z \rangle p, q)$ *if* $x \notin fn(q) \cup \{y, z\}$.

Proof

a) $p \equiv_\alpha q$ implies $p \sim q$
The relation S^1 where
$\mathcal{S}^1 \stackrel{def}{=} \{(p,q) \mid p, q \in \mathcal{P}_b,\ p \equiv_\alpha q\}$
is a strong bisimulation. The result follows from the observation that $p \equiv q$ and $p \xrightarrow{\gamma} p'$ implies $q \xrightarrow{\gamma} p'$ (by the application of the rule (1) from the Table 3).

b) $p \parallel nil \sim p$
The relation
$\mathcal{S}^2 \stackrel{def}{=} \{(p \parallel nil, p) \mid p, \in \mathcal{P}_b\}$
is a strong bisimulation.
Let $(p,q) \in \mathcal{S}^2$. Then $p = q \parallel nil$.
If $p \xrightarrow{\alpha} p'$, then the last used rule to infer this transition is the rule (14). So $p' = q' \parallel nil$ et $q \xrightarrow{\alpha} q'$. Then $(q' \parallel nil, q') \in \mathcal{S}^2$.
If $q \xrightarrow{\alpha} q'$, by the application of the rule (14) we obtain $p \xrightarrow{\alpha} q' \parallel nil$ with $(q' \parallel nil, q') \in \mathcal{S}^2$.

c) $p \parallel q \sim q \parallel p$ The relation
$\mathcal{S}^3 \stackrel{def}{=} \{(p \parallel q, q \parallel p) \mid p, q \in \mathcal{P}_b\}$
is a strong bisimulation.
If $p \parallel q \xrightarrow{\alpha} r$, then the last used rule to infer this transition is one of the rules (12), (13) or (14) and $r = p' \parallel q'$. By the application of the rule (12), or one of the symmetrical of the rules (13) or (14), we obtain $q \parallel p \xrightarrow{\alpha} q' \parallel p'$.

d) $(p \parallel q) \parallel r \sim p \parallel (q \parallel r)$
The relation
$\mathcal{S}^4 \stackrel{def}{=} \{(p \parallel q) \parallel r, p \parallel (q \parallel r) \mid p, q, r \in \mathcal{P}_b\}$
is a strong bisimulation. The proof is similar to the previous case.

e) $p + nil \sim p$
The relation
$\mathcal{S}^5 \stackrel{def}{=} \{(p + nil, p) \mid p \in \mathcal{P}_b\} \bigcup \{(p, p) \mid p \in \mathcal{P}_b\}$
is a strong bisimulation.
Let $(p, q) \in \mathcal{S}^2$. Then $p = q + nil$ ou $p = q$.
The case $p = q$ is obvious.
Let $p = q + nil$.
If $q + nil \xrightarrow{\alpha} r$, then the last used rule to infer this transition is the rule (8), and $r = q'$, where $q \xrightarrow{\alpha} q'$. Obviously $(q', q') \in \mathcal{S}^2$.
If $q \xrightarrow{\alpha} r$, by the application of the rule (8), we obtain $q + nil \xrightarrow{\alpha} r$ and $(r, r) \in \mathcal{S}^2$.

f) $p + q \sim q + p$

g) $(p + q) + r \sim p + (q + r)$.
For cases f) and g), the proof that $\mathcal{S}^6$ et $\mathcal{S}^7$ are strong bisimulations is similar to case e).
$\mathcal{S}^6 \stackrel{def}{=} \{(p + q, q + p) \mid p, q \in \mathcal{P}_b\} \bigcup \{(p, p) \mid p \in \mathcal{P}_b\}$
$\mathcal{S}^7 \stackrel{def}{=} \{((p + q) + r, p + (q + r)) \mid p, q, r \in \mathcal{P}_b\} \bigcup \{(p, p) \mid p \in \mathcal{P}_b\}$

(h) $\nu x p \sim p$ si $x \notin fn(p)$
The relation
$\mathcal{S}^8 \stackrel{def}{=} \{(\nu x p, p) \mid p \in \mathcal{P}_b, x \notin fn(p)\}$
is a strong bisimulation.
Let $(p, q) \in \mathcal{S}^8$. Then $p = \nu x q$ with $x \notin fn(q)$.
If $\nu x q \xrightarrow{\alpha} r$, then the last used rule to infer this transition is the rule (7), and $r = \nu x q'$, where $q \xrightarrow{\alpha} q'$ and $x \notin fn(q')$. Obviously $(\nu x q', q') \in \mathcal{S}^8$.
If $q \xrightarrow{\alpha} r$, then by the application of the rule (7), we obtain that $\nu x q \xrightarrow{\alpha} \nu x r$ and $(\nu x r, r) \in \mathcal{S}^8$ (since $x \notin fn(r)$).

(i) $\nu y \nu x p \sim \nu x \nu y p$ if $x \neq y$
The relation
$\mathcal{S}^9 \stackrel{def}{=} \{(\nu x_1 \ldots \nu x_n p, \nu x_{\sigma(1)} \ldots \nu x_{\sigma(n)} p) \mid n \in \mathbb{N},\ \sigma \text{ permutation of } \{1, \ldots, n\},\ \text{for all } i, j,\ x_i \neq x_j, p \in \mathcal{P}_b\}\}$
is a strong bisimulation.
Let $(\nu x_1 \ldots \nu x_n p, \nu x_{\sigma(1)} \ldots \nu x_{\sigma(n)} p) \in \mathcal{S}^9$ and let $\nu x_1 \ldots \nu x_n p \xrightarrow{\alpha} q'$ be a transition.
By induction on n, we can prove that this transition is derived from a transition of p, $p \xrightarrow{\beta} p'$. We shall make an analysis by cases, depending on β.

- $\beta = \tau$.
  Then by n applications of the rule (7), we obtain $\nu x_1 \ldots \nu x_n p \xrightarrow{\tau} \nu x_1 \ldots \nu x_n p'$ and $\nu x_{\sigma(1)} \ldots \nu x_{\sigma(n)} p \xrightarrow{\tau} \nu x_{\sigma(1)} \ldots \nu x_{\sigma(n)} p'$.

- $\beta = a\langle\tilde{b}\rangle$.

  Then $a \notin \{x_1, \ldots, x_n\}$, and by n applications of the rule (7), we obtain $\nu x_1 \ldots \nu x_n p \xrightarrow{a\langle\tilde{b}\rangle} \nu x_1 \ldots \nu x_n p'$ and $\nu x_{\sigma(1)} \ldots \nu x_{\sigma(n)} p \xrightarrow{a\langle\tilde{b}\rangle} \nu x_{\sigma(1)} \ldots \nu x_{\sigma(n)} p'$.
- $\beta = \nu\tilde{b}\bar{a}\tilde{c}$, and $\tilde{c} \setminus \tilde{b} \cap \{x_1, \ldots, x_n\} = \{x_{i_1}, \ldots, x_{i_m}\}$ We have two cases again.
  - $a \notin \{x_1, \ldots, x_n\}$

    Then by $n - m$ applications of the rule (7) and by m applications of the rule (5), we obtain $\nu x_1 \ldots \nu x_n p \xrightarrow{\nu x_{i_1} \ldots \nu x_{i_m} \nu\tilde{b}\bar{a}\tilde{c}} \nu y_1 \ldots \nu y_{n-m} p'$ and $\nu x_{\sigma(1)} \ldots \nu x_{\sigma(n)} p \xrightarrow{\nu x_{i_1} \ldots \nu x_{i_m} \nu\tilde{b}\bar{a}\tilde{c}} \nu y_{\rho(1)} \ldots \nu y_{\rho(n-m)} p'$ for some permutation ρ of $\{1, \ldots, n-m\}$ and $\{y_1 \ldots y_{n-m}\} = \{y_{\rho(1)} \cdots y_{\rho(n-m)}\} = \{x_1, \ldots, x_n\} \backslash \{x_{i_1}, \ldots, x_{i_m}\}$.
  - $a \in \{x_1, \ldots, x_n\}$

    Then by several applications of the rules (5) and (7) and by an application of the rule (6), we obtain $\nu x_1 \ldots \nu x_n p \xrightarrow{\tau} \nu y_1 \ldots \nu y_{n+k} p'$ and $\nu x_{\sigma(1)} \ldots \nu x_{\sigma(n)} p \xrightarrow{\tau} \nu y_{\rho(1)} \ldots \nu y_{\rho(n+k)} p'$ for some permutation ρ of $\{1, \ldots, n+k\}$, $|\tilde{b}| = k$ and $\{y_1 \ldots y_{n+k}\} = \{y_{\rho(1)} \cdots y_{\rho(n+k)}\} = \{x_1, \ldots, x_n\} \cup \tilde{b}$.

(j) $(\nu x p) \parallel q \sim \nu x(p \parallel q)$ if $x \notin fn(q)$

The relation
$\mathcal{S}^{10} \stackrel{def}{=} \{((\nu x p) \parallel q, \nu x(p \parallel q)) \mid p \in \mathcal{P}_b, x \notin fn(q)\} \bigcup \{(p, p) \mid p \in \mathcal{P}_b\}$ *
is a strong bisimulation.
Let $((\nu x p) \parallel q, \nu x(p \parallel q)) \in \mathcal{S}^{10}$, with $x \notin fn(q)$.
If $(\nu x p) \parallel q \xrightarrow{\alpha} r$, then the last used rule to infer this transition is one of the rules (12), (13), (14) or the symmetrical of one of the rules (13), or (14).
If $\alpha = \nu\tilde{b}\bar{a}\tilde{c}$ et $x \in \tilde{b}$, then for some p' and q' we have $(\nu x p) \parallel q \xrightarrow{\nu\tilde{b}\bar{a}\tilde{c}} p' \parallel q'$ and $\nu x(p \parallel q) \xrightarrow{\nu\tilde{b}\bar{a}\tilde{c}} p' \parallel q'$ by the application of the rule (5).
For all other cases, we obtain $(\nu x p) \parallel q \xrightarrow{\alpha} (\nu y p') \parallel q'$ and $\nu x(p \parallel q) \xrightarrow{\alpha} \nu y(p' \parallel q')$ for some y, p' and q'.

(k) $(\nu x p) + q \sim \nu x(p + q)$ if $x \notin fn(q)$

(l) $\langle y = z\rangle(\nu x p), q \sim_b \nu x(\langle y = z\rangle p, q)$ if $x \notin fn(q) \cup \{y, z\}$

For the cases k) and l), the proof that $\mathcal{S}^{11}$ et $\mathcal{S}^{12}$ are strong bisimulations is similar to case e).
$\mathcal{S}^{11} \stackrel{def}{=} \{((\nu x p) + q, \nu x(p + q)) \mid p \in \mathcal{P}_b, x \notin fn(q)\} \bigcup \{(p, p) \mid p \in \mathcal{P}_b\}$
$\mathcal{S}^{12} \stackrel{def}{=} \{(\langle y = z\rangle(\nu x p), q, \nu x(\langle y = z\rangle p, q)) \mid p, q \in \mathcal{P}_b, x \notin fn(q) \cup \{y, z\}\} \bigcup \{(p, p) \mid p \in \mathcal{P}_b\}$

□ *Lemma* 6

Lemma 2, 4 and 6 allow us to omit the parenthesis in $p \parallel q \parallel r$ or $p + q + r$ in some contexts, and to omit "nil" in some terms.

Like in [11], we use "bisimulations up-to" to reduce the size of relations in the proofs. We shall use just the "bisimulations up-to" $\sim$.

Definition 9. *A symmetric relation S is a* **weak bisimulation up-to** $\sim$ *if $(p, q) \in S$, implies*

1) if $p \xrightarrow{\tau} p'$, then $\exists q'$ such that $q \stackrel{\epsilon}{\Longrightarrow} q'$ and $(p', q') \in \sim S \sim$,

2) if $p \xrightarrow{\nu\tilde{b}\bar{a}\tilde{c}} p'$ *and* $\tilde{b} \cap fn(p,q) = \emptyset$, *then* $\exists q'$ *such that* $q \stackrel{\nu\tilde{b}\bar{a}\tilde{c}}{\Longrightarrow} q'$ *and* $(p',q') \in \sim S \sim$,

3) if $p \xrightarrow{a\langle\tilde{b}\rangle?} p'$ *then* $\exists q'$ *scuh that* $q \stackrel{\epsilon}{\Longrightarrow} \bullet \xrightarrow{a\langle\tilde{b}\rangle?} \bullet \stackrel{\epsilon}{\Longrightarrow} q'$ *et* $(p',q') \in \sim S \sim$.

The definition of **strong bisimulation up-to** $\sim$ *follows as in the Definition 8.*

By classical arguments, we can prove the next Lemma.

Lemma 7. *If S is a weak bisimulation up-to $\sim$ (strong bisimulation up-to $\sim$) then $S \subseteq \approx$ ($S \subseteq \sim$).*

Proof

We give the proof only for the weak case.

If S is a weak bisimulation up-to $\sim$, we prove that

$$\mathcal{S}^u \stackrel{def}{=} \bigcup_{n<\infty} \mathcal{S}_n$$

is a weak bisimulation, where

$$\mathcal{S}_0 \stackrel{def}{=} \{(p,q) \mid p \sim S \sim q\}$$

$$\mathcal{S}_{n+1} \stackrel{def}{=} \{(p[\tilde{z}/\tilde{x}], q[\tilde{z}/\tilde{x}]) \mid (p,q) \in \mathcal{S}_n,\ \tilde{z} \cap fn(p,q) = \emptyset,\ [\tilde{z}/\tilde{x}] \text{ injective}\}$$

The proof is by induction on n.

- Let $(p,q) \in \mathcal{S}_0$. Then $\exists p_1, q_1$ such that $p \sim p_1\ S\ q_1 \sim q$.
  1. For any $p \xrightarrow{\alpha} p'$ such that $bn(\alpha) \cap fn(p,q,p_1,q_1) = \emptyset$, we can prove that we can fill the next diagram:

$$\begin{array}{ccccccc} p & \sim & p_1 & & S & & \\ \downarrow\alpha & & \downarrow\alpha & & & & \\ p' & \sim & p'_1 & \sim & S & \sim & q'_1 \end{array} \qquad \begin{array}{ccc} q_1 & \sim & q \\ \Downarrow\hat{\alpha} & & \Downarrow\hat{\alpha} \\ q'_1 & \sim & q' \end{array}$$

     We prove the assertion only for the case $\alpha = \nu\tilde{b}\bar{a}\tilde{c}$. The other cases are simpler.

     Let $p \xrightarrow{\nu\tilde{b}\bar{a}\tilde{c}} p'$, with $\tilde{b} \cap fn(p,q,p_1,q_1) = \emptyset$.

     Then $p \sim p_1$ implies $p_1 \xrightarrow{\nu\tilde{b}\bar{a}\tilde{c}} p'_1$.

     Since $p_1\ S\ q_1$, by the definition 9 we obtain $q_1 \stackrel{\nu\tilde{b}\bar{a}\tilde{c}}{\Longrightarrow} q'_1$ with $p'_1 \sim S \sim q'_1$.

     By induction on the length of $\stackrel{\nu\tilde{b}\bar{a}\tilde{c}}{\Longrightarrow}$ and using that $q_1 \sim q$, we obtain $q \stackrel{\nu\tilde{b}\bar{a}\tilde{c}}{\Longrightarrow} q'$ with $q' \sim q'_1$.

     We use after $\sim \circ \sim \subseteq \sim$ to obtain $(p_1,q_1) \in \mathcal{S}_0$.
  2. If $p \xrightarrow{\nu\tilde{b}\bar{a}\tilde{c}} p'$ such that $\tilde{b} \cap fn(p,q) = \emptyset$ and $\tilde{b} \cap fn(p_1,q_1) \neq \emptyset$, then let $\tilde{d}$ such that $\tilde{d} \cap fn(p,q,p_1,q_1) \neq \emptyset$. Repeating the reasoning of case 1., we obtain that $p \xrightarrow{\nu\tilde{d}\bar{a}\tilde{c}} p$", and $q \stackrel{\nu\tilde{d}\bar{a}\tilde{c}}{\Longrightarrow} q$" with $p" \sim S \sim q"$.

     We can prove after that $p' = p"[\tilde{b}/\tilde{d}]$ and $q \stackrel{\nu\tilde{b}\bar{a}\tilde{c}}{\Longrightarrow} q'$ with $q' = q"[\tilde{b}/\tilde{d}]$, and hence $(p',q') \in \mathcal{S}_1$.

– Let $(p,q) \in \mathcal{S}_{n+1}$. Then $\exists (p_1,q_1) \in \mathcal{S}_n$ such that $p = p_1[\tilde{z}/\tilde{x}]$, $q = q_1[\tilde{z}/\tilde{x}]$, $fn(p_1,q_1) \cap \tilde{z} = \emptyset$ et $[\tilde{z}/\tilde{x}]$ is an injective substitution.
let $p \xrightarrow{\alpha} r'$. We can prove that exists σ' injective, such that $p = p_1[\tilde{z}/\tilde{x}] = p_1\sigma'$, $q = q_1[\tilde{z}/\tilde{x}] = q_1\sigma'$, $r' = r\sigma'$, $\alpha = \sigma'(\beta)$ and $prcod(\sigma') \cap fn(p_1,q_1,\beta) = \emptyset$. It follows $p_1 \xrightarrow{\beta} r$, and since $(p_1,q_1) \in \mathcal{S}_n$, by induction hypothesis, we obtain $q_1 \stackrel{\beta}{\Longrightarrow} s$ with $(r,s) \in \mathcal{S}_m$ for some $m \in \mathbb{N}$.
Hence $q = q_1\sigma' \stackrel{\alpha}{\Longrightarrow} s\sigma'$ and $prcod(\sigma') \cap fn(r,s) = \emptyset$, so $(r\sigma', s\sigma') \in \mathcal{S}_{m+1}$.

The proof for the strong bisimulation up-to $\sim$ is similar.

□ *Lemma* 7

Labelled bisimulation is a congruence with respect to restriction and parallelism.

Lemma 8. *If $p \approx q$ ($p \sim q$) then $\nu ap \approx \nu aq$ ($\nu ap \sim \nu aq$).*

Proof
We shall prove that the relation

$$\mathcal{S} \stackrel{def}{=} \bigcup_{n=0}^{\infty} \mathcal{S}_n$$

where
$\mathcal{S}_0 \stackrel{def}{=} \{(p,q) \mid p \approx q\}$
$\mathcal{S}_{n+1} \stackrel{def}{=} \{(p,q) \mid \exists a \in Ch_b, \exists (p',q') \in \mathcal{S}_n,\ p = \nu ap',\ q = \nu aq'\}$
is a weak bisimulation.

We can prove that $z \notin fn(p,q)$ and $p \approx q$ imply $p[z/x] \approx q[z/x]$, and after, we can prove by induction on n, that $z \notin fn(p,q)$ and $(p,q) \in \mathcal{S}_n$ imply $(p[z/x], q[z/x]) \in \mathcal{S}_m$ with $m \leq n$.

Using this, we can also prove by induction on n, that if $(p,q) \in \mathcal{S}_n$ and $p \equiv_\alpha p_1$, then it exists

$$q_1 \text{ and } m \leq n \text{ such that } q \equiv_\alpha q_1 \text{ et } (p_1,q_1) \in \mathcal{S}_m. \tag{1}$$

Now, we prove by induction on n, that if $(p,q) \in \mathcal{S}_n$ and $p \xrightarrow{\alpha} p'$ where α satisfies the conditions in the Definition 7, then it exists $m \in \mathbb{N}$ and $q \stackrel{\hat{\alpha}}{\Longrightarrow} q'$, such that $(p',q') \in \mathcal{S}_m$.

The case $n = 0$ follows directly from the Definition 7.

Let $(p,q) \in \mathcal{S}_{n+1}$. Hence $p = \nu ap_1$ et $q = \nu aq_1$ with $(p_1,q_1) \in \mathcal{S}_n$.

Let $p \xrightarrow{\alpha} p'$.

If the last rule used is (1), with $p \equiv_\alpha p_1$ and $p_1 \xrightarrow{\alpha} p'$, we use 1 to obtain a corresponding q_1 and we repeat the reasoning for p_1 et q_1 (in this manner we reduce strictly the number of applications of the rule (1)). So we can suppose that the last rule used is not (1).

We have several case, depending on the action α.

– $\alpha = \tau$. Then the last applied rule is (6) or (7).

If the last applied rule is (6), then $p_1 \xrightarrow{\nu\tilde{b}\bar{a}\tilde{c}} p_1'$ and $p' = \nu a\nu\tilde{b}p_1'$. By induction, $\exists m$ such that $q_1 \stackrel{\nu\tilde{b}\bar{a}\tilde{c}}{\Longrightarrow} q_1'$ and $(p_1',q_1') \in \mathcal{S}_m$. We obtain by an application of the rule (6) and by several applications of the rule (7), that $q \stackrel{\tau}{\Longrightarrow} \nu a\nu\tilde{b}q_1'$ with $(\nu a\nu\tilde{b}p_1', \nu a\nu\tilde{b}q_1') \in \mathcal{S}_{m+1+|\tilde{b}|}$.

If the last applied rule is (7), then $p_1 \xrightarrow{\tau} p_1'$ and $p' = \nu a p_1'$. By induction $\exists m$ such that $q_1 \stackrel{\tau}{\Longrightarrow} q_1'$ and $(p_1', q_1') \in \mathcal{S}_m$. We obtain by several applications of the rule (7), that $q \stackrel{\tau}{\Longrightarrow} \nu a q_1'$, and $(\nu a p_1', \nu a q_1') \in \mathcal{S}_{m+1}$.

- $\alpha = \nu \tilde{b} \bar{d} \tilde{c}$. So $d \neq a$ and that last rule used is (5) or (7).

  If the last applied rule is (5), then $p_1 \xrightarrow{\nu \tilde{b} \bar{d} \tilde{c}} p'$ and $a \in \tilde{c} \setminus (\tilde{b} \cup \{d\})$. By induction $\exists m$ such that $q_1 \stackrel{\nu \tilde{b} \bar{d} \tilde{c}}{\Longrightarrow} q'$ and $(p', q') \in \mathcal{S}_m$. We obtain by an application of the rule (5) and by several applications of the rule (7), that $q \stackrel{\nu a \tilde{b} \bar{d} \tilde{c}}{\Longrightarrow} q'$.

  If the last applied rule is (7), then $p_1 \xrightarrow{\nu \tilde{b} \bar{d} \tilde{c}} p_1'$, $p' = \nu a p_1'$ and $a \notin (\tilde{c} \setminus \tilde{b}) \cup \{d\}$. By induction $\exists m$ such that $q_1 \stackrel{\tau}{\Longrightarrow} q_1'$ et $(p_1', q_1') \in \mathcal{S}_m$. We obtain by several applications of the rule (7), that $q \stackrel{\nu \tilde{b} \bar{d} \tilde{c}}{\Longrightarrow} \nu a q_1'$, and $(\nu a p_1', \nu a q_1') \in \mathcal{S}_{m+1}$.

- $\alpha = d\langle \tilde{b} \rangle ?$. So $d \neq a$ and the last rule used is (7). Then $p_1 \xrightarrow{d\langle \tilde{b} \rangle ?} p_1'$, and $p' = \nu a p_1'$. By induction $\exists m$ such that $q_1 \stackrel{\epsilon}{\Longrightarrow} \bullet \xrightarrow{d\langle \tilde{b} \rangle ?} \bullet \stackrel{\epsilon}{\Longrightarrow} q_1'$ and $(p_1', q_1') \in \mathcal{S}_m$. We obtain by several applications of the rule (7), that $q \stackrel{\epsilon}{\Longrightarrow} \bullet \xrightarrow{d\langle \tilde{b} \rangle ?} \bullet \stackrel{\epsilon}{\Longrightarrow} \nu a q_1'$, and $(\nu a p_1', \nu a q_1') \in \mathcal{S}_{m+1}$.

The case "strong bisimulation" is similar.

□ *Lemma* 8

Lemma 9.

1. *If* $p \sim q$ *then* $p \parallel r \sim q \parallel r$.
2. *If* $p \approx q$ *then* $p \parallel r \approx q \parallel r$.

Proof

For the case "weak bisimulation", we prove that the relation

$$\mathcal{T} \stackrel{def}{=} \{ (p \parallel r, q \parallel r) \mid p \approx q \}$$

is a weak bisimulation.

Let $p \parallel r \xrightarrow{\alpha} s$.

If the last applied rule is (1), with $p \parallel r \equiv_\alpha p_1 \parallel r_1$, we have $q \parallel r \equiv_\alpha q \parallel r_1$ and $p_1 \approx q$ (in this manner we strictly reduce the number of applications of the rue (1)). So we can suppose that the last rule used is not (1).

- $p \parallel r \xrightarrow{\tau} s$. There are two cases:
  - $s = p' \parallel r$ et $p \xrightarrow{\tau} p'$.
    Since $p \approx q$ then $\exists q'$ such that $q \stackrel{\epsilon}{\Longrightarrow} q'$ and $p' \approx q'$. Hence $q \parallel r \stackrel{\epsilon}{\Longrightarrow} q' \parallel r$, with $(p' \parallel r, q' \parallel r) \in \mathcal{T}$.
  - $s = p \parallel r'$ et $r \xrightarrow{\tau} r'$.
    Then $q \parallel r \xrightarrow{\tau} q \parallel r'$, with $(p \parallel r', q \parallel r') \in \mathcal{T}$.
- $p \parallel r \xrightarrow{\nu \tilde{b} \bar{a} \tilde{c}} s$ with $\tilde{b} \cap fn(p, q, r) = \emptyset$. There are two cases:
  - $s = p' \parallel r'$, $p \xrightarrow{\nu \tilde{b} \bar{a} \tilde{c}} p'$ and $r \xrightarrow{a\langle \tilde{c} \rangle ?} r'$.
    Since $p \approx q$ then $q \stackrel{\nu \tilde{b} \bar{a} \tilde{c}}{\Longrightarrow} q'$ and $p' \approx q'$, and so $q \parallel r \stackrel{\nu \tilde{b} \bar{a} \tilde{c}}{\Longrightarrow} q' \parallel r'$, with $(p' \parallel r', q' \parallel r') \in \mathcal{T}$.

- $s = p' \parallel r'$, $p \xrightarrow{a\langle\tilde{c}\rangle?} p'$ and $r \xrightarrow{\nu\tilde{b}\tilde{a}\tilde{c}} r'$.
  Since $p \approx q$ then $q \stackrel{\epsilon}{\Longrightarrow} q'' \xrightarrow{a\langle\tilde{c}\rangle?} q''' \stackrel{\epsilon}{\Longrightarrow} q'$ for some q'', q''' and $p' \approx q'$, and so $q \parallel r \stackrel{\nu\tilde{b}\tilde{a}\tilde{c}}{\Longrightarrow} q' \parallel r'$, with $(p' \parallel r', q' \parallel r') \in \mathcal{T}$.

– $p \parallel r \xrightarrow{a\langle\tilde{b}\rangle?} s$.
  $s = p' \parallel r'$, $p \xrightarrow{a\langle\tilde{b}\rangle?} p'$ and $r \xrightarrow{a\langle\tilde{b}\rangle?} r'$.
  Since $p \approx q$ then $q \stackrel{\epsilon}{\Longrightarrow} q'' \xrightarrow{a\langle\tilde{b}\rangle?} q''' \stackrel{\epsilon}{\Longrightarrow} q'$ for some q'', q''' et $p' \approx q'$, and hence $q \parallel r \stackrel{\epsilon}{\Longrightarrow} q'' \parallel r \xrightarrow{a\langle\tilde{b}\rangle?} q''' \parallel r' \stackrel{\epsilon}{\Longrightarrow} q' \parallel r'$, with $(p' \parallel r', q' \parallel r') \in \mathcal{T}$.

The case "strong bisimulation" is similar.

□ *Lemma* 9

The following lemmas prove that labelled bisimulation is a pretty strong relation.

Lemma 10.
1) $p \sim q$ *implies* $p \sim_b q$.
2) $p \approx q$ *implies* $p \approx_b q$.

Proof
We prove only the weak case. We prove that $\approx$ is a weak barbed bisimulation.
Let $(p, q) \in \approx$.

– Let $p \downarrow_c$. Then $p \xrightarrow{\alpha}$ for an output α of subject c. Since $p \approx q$, we obtain that $q \stackrel{\alpha}{\Longrightarrow}$ and so $q \Downarrow_c$.
– Let $p \xrightarrow{\tau} p'$. Since $p \approx q$, then $q \stackrel{\epsilon}{\Longrightarrow} q'$ with $(p', q') \in \approx$.

□ *Lemma* 10

Corollary 3.
1) $p \sim q$ *implies* $p \overset{e}{\sim}_b q$.
2) $p \approx q$ *implies* $p \overset{e}{\approx}_b q$.

Proof
We give the proof for the weak case.
Let $p \approx q$.
Let C be an arbitrary static context.
By induction on the structure of C, and using the Lemmas 8, and 9, we obtain that $C[p] \approx C[q]$.
By the Lemma 10, we obtain that $C[p] \approx_b C[q]$. So $p \overset{e}{\sim}_b q$.

□ *Corollary* 3

In the same manner, we can prove the next results.

Lemma 11.
1) $p \sim q$ *implies* $p \sim_\phi q$.
2) $p \approx q$ *implies* $p \approx_\phi q$.

Corollary 4.

1) $p \sim q$ *implies* $p \overset{e}{\sim}_{\phi} q$.
2) $p \approx q$ *implies* $p \overset{e}{\approx}_{\phi} q$.

We use similar arguments as in [18], to prove that labelled bisimulation, ϕ - equivalence and barbed equivalence coincide for an important class of processes, namely image finite processes.

A labelled transitional system $(\mathcal{P}, Act, \mapsto)$ is *image finite* if for any processes p and action α, the set $\{q \mid p \overset{\alpha}{\mapsto} q\}$ est finite (in our case $\overset{\alpha}{\mapsto}$ can stand either for the strong or for the weak reduction).

A processes p is image finite if for any action α, the set $\{q \mid p \overset{\alpha}{\mapsto} q\}$ is finite, and for any processes q and action α such that $p \overset{\alpha}{\mapsto} q$, q is image finite.

Remark that in what concerns the strong reduction, any processes is image finite (up-to alpha-conversion).

In order to keep the notations simple, we shall give the proof just for the monadic version of $b\pi$ -calculus (but our results can be easily extended to the polyadic version).

Lemma 12. *For all image finite processes* p *and* q,

1) $p \overset{e}{\sim}_b q$ *implies* $p \sim q$,
2) $p \overset{e}{\approx}_b q$ *implies* $p \approx q$.

Proof

We shall give the proof only for the weak case.

Let $\mathcal{F}$ a monotonic function defined on the set of relations between processes $\mathbf{P}(\mathcal{P}_b \times \mathcal{P}_b)$ associated to the definition of labelled bisimulations.

If S is a relation on $\mathcal{P}_b$, then $(p,q) \in \mathcal{F}(S)$, only if the following conditions are satisfied:

1) if $p \xrightarrow{\tau} p'$, then $\exists q'$ such that $q \overset{\epsilon}{\Longrightarrow} q'$ and $(p', q') \in S$,
2) if $p \xrightarrow{\nu\tilde{b}\bar{a}\tilde{c}} p'$ and $\tilde{b} \cap fn(p,q) = \emptyset$, then $\exists q'$ such that $q \overset{\nu\tilde{b}\bar{a}\tilde{c}}{\Longrightarrow} q'$ and $(p', q') \in S$,
3) if $p \xrightarrow{a\langle\tilde{b}\rangle?} p'$ then $\exists q'$ such that $q \overset{\epsilon}{\Longrightarrow} \bullet \xrightarrow{a\langle\tilde{b}\rangle?} \bullet \overset{\epsilon}{\Longrightarrow} q'$ and $(p', q') \in S$.

Let $\approx^0 = \mathcal{P}_b \times \mathcal{P}_b$, $\approx^{n+1} = \mathcal{F}(\approx^n)$, and $\approx^\omega = \bigcap_{k<\omega} \approx^k$. We call $\approx^m$ a weak m-bisimulation. It is well known [7], that for an image finite labelled transitional system $\approx^\omega = \approx$.

It suffices to prove that $\forall m$, $p \overset{e}{\approx}_b q \Longrightarrow p \approx^m q$.

For a set N of pairs of names, we shall denote N_i the projection on the i-Th element, $n(N) = N_1 \cup N_2$, and we shall require $N_1 \cap N_2 = \emptyset$. We use H^+ (Y^-) as an abbreviation for $H \cup \{(y_1, y_1')\}$ $(Y \setminus \{(y_1, y_1')\})$. H_1 plays the role of names which can be known by p or q, and Y plays the role of names which can be learned by p or q. To prove the implication, we build a contexts family $C^n_{M,H,Y}[\bullet]$ such that the relation

$$r^m = \Big\{(p,q) \mid \exists n, M, H, Y, \tilde{x} \text{ such that } n(H) \cap n(Y) = \emptyset,\ |Y| = m,\ fn(p \parallel q) \subseteq H_1 \subseteq M,$$

$$(H_1 \cup Y_1) \subseteq \{\tilde{x}\},\ (H_2 \cup Y_2) \cap \{\tilde{x}\} = \emptyset,\ \ \nu\tilde{x} C^n_{M,H,Y}[p] \approx_b \nu\tilde{x} C^n_{M,H,Y}[q]\Big\}$$

is a weak m-bisimulation.

Let *in*, *out*, *new* and $\{b_n | n \geq 0\}$ be some "fresh" names, such that $(\{\tilde{x}\} \cup n(H) \cup n(Y)) \cap (\{b_n | n \geq 0\} \cup \{in, out, new\}) = \emptyset$.

Let $C^n_{M,H,Y}[\bullet] \stackrel{def}{=} [\bullet] \parallel ASensor_{r,n}\langle M\rangle \parallel GSensor_{s,m}\langle H,Y\rangle$
where $|M| = r,\ |H| = s,\ |Y| = m$,

The role of $GSensor$ is to provide to p et q all the possible communications. The conditions $fn(p \parallel q) \subseteq H_1$, $fn(p \parallel q) \cap Y_1 = \emptyset$ and $|Y_1| = m$ assure that $GSensor_{s,m}\langle H,Y\rangle$ can "satisfy" all the possible communications of p or q during m transitions.

$$GSensor_{s,m} \stackrel{def}{=} (H,Y)$$

$$\sum_{((a,a'),(b,b'))\in H\times H^+} \bar{a}b.(\tau.GSensor_{s+1,m-1}\langle H^+,Y^-\rangle + \tau.W\langle a',b',in\rangle)$$

$$+ \sum_{a\in H_1} a(y).case\langle y,H_1,(R_z)_{z\in H_1},R'_y\rangle, \qquad \text{with } |H| = s,\ |Y| = m$$

$$W \stackrel{def}{=} (a,b,r)(\bar{a} + \tau.(\bar{b} + \bar{r})),$$

$$R_z \stackrel{def}{=} \tau.GSensor_{s,m-1}\langle H,Y^-\rangle + \tau.W\langle a',z',out\rangle$$

$$R'_y \stackrel{def}{=} \tau.GSensor_{s+1,m-1}\langle H\cup\{(y,y'_1)\},Y^-\rangle + \tau.W\langle a',new,out\rangle$$

$$case\langle y,H,(R_y)_{y\in H},R\rangle \stackrel{def}{=} \begin{cases} R & \text{si } H = \emptyset, \\ \langle y = z\rangle R_z, case\langle y,H',(R_y)_{y\in H'},R\rangle & \text{if } H = H'\cup\{z\}. \end{cases} \quad (2)$$

$ASensor$ counts the number of visible made by p, q or $GSensor_{s,m}$ (hence the condition $fn(p \parallel q) \subseteq H_1 \subseteq M$).

$$ASensor_{r,n} \stackrel{def}{=} (M)(\overline{b_n} + \sum_{a\in M} a(x).ASensor_{r+1,n+1}\langle M\cup\{x\}\rangle), \qquad \text{with } |M| = r$$

We prove that r^m is a weak m-bisimulation.

Let $(p,q) \in r^m$. Then $\exists n, M, H, Y, \tilde{x}$ such that $n(H)\cap n(Y) = \emptyset$, $|Y| = m$, $fn(p \parallel q) \subseteq H_1 \subseteq M$, $(H_1\cup Y_1) \subseteq \{\tilde{x}\}$, $(H_2\cup Y_2)\cap\{\tilde{x}\} = \emptyset$, $\nu\tilde{x}C^n_{M,H,Y}[p] \approx_b \nu\tilde{x}C^n_{M,H,Y}[q]$.

[1] Let suppose $p \xrightarrow{\tau} p'$.

Then $\nu\tilde{x}C^n_{M,H,Y}[p] \xrightarrow{\tau} \nu\tilde{x}C^n_{M,H,Y}[p']$. Then we must have a transition

$$\nu\tilde{x}C^n_{M,H,Y}[q] \stackrel{\epsilon}{\Longrightarrow} t \approx_b \nu\tilde{x}C^n_{M,H,Y}[p']$$

Since $\nu\tilde{x}C^n_{M,H,Y}[p'] \downarrow_{b_n}$, we can infer that $ASensor$ remains unchanged, and since $H_1 \subseteq M$, we obtain that $t \equiv \nu\tilde{x}C^n_{M,H,Y}[q']$ where $q \stackrel{\epsilon}{\Longrightarrow} q'$.

[2] Let suppose $p \xrightarrow{\bar{a}b} p'$.

Then

$$\nu\tilde{x}C^n_{M,H,Y}[p] \xrightarrow{\tau} \nu\tilde{x}(p' \parallel ASensor_{r+1,n+1}\langle M\cup\{b\}\rangle \parallel case\langle b,H_1,(R_z)_{z\in H_1},R'_b\rangle)$$

$$= t_1 \xrightarrow{\tau} \nu\tilde{x}(p' \parallel ASensor_{r+1,n+1}\langle M\cup\{b\}\rangle \parallel GSensor_{s,m-1}\langle H,Y^-\rangle)$$

$$= t_2 = \nu\tilde{x}C^{n+1}_{M\cup\{b\},H,Y^-}[p']$$

Then it should exist the processes u_1, u_2 derived from $\nu\tilde{x}C^n_{M,H,Y}[q]$ such that

$$\nu\tilde{x}C^n_{M,H,Y}[q] \stackrel{\epsilon}{\Longrightarrow} u_1 \approx_b t_1$$

$$u_1 \stackrel{\epsilon}{\Longrightarrow} u_2 \approx_b t_2$$

First step: $\nu\tilde{x}C^n_{M,H,Y}[p] \stackrel{\tau}{\longrightarrow} t_1$. The configuration $\nu\tilde{x}C^n_{M,H,Y}[q]$ must do at least one step, since $\nu\tilde{x}C^n_{M,H,Y}[q] \downarrow_{b_n}$ whilst $t_1 \not\Downarrow_{b_n}$. Because $t_1 \downarrow_{b_{n+1}}$, p and $GSensor$ can not make more than one visible step. We obtain that u_1 must have the following form:

$$\nu\tilde{x'}(q'' \parallel ASensor_{r+1,n+1}\langle M \cup \{b_1\}\rangle \parallel v'')$$

for some $b_1, q'', v'', \nu\tilde{x'}, q \stackrel{\alpha'}{\Longrightarrow} q'', GSensor_{s,m}\langle H, Y\rangle \stackrel{\alpha''}{\longrightarrow} v''$, where α'' and α' are complementary actions. The fact $t_1 \Longrightarrow\Downarrow_{\{a',b',out,b_{n+1}\}}$ implies that α' is an output, and α'' must be the corresponding input; moreover, α' can not be a bound output. Hence we obtain that $\alpha' = \overline{a_1}b_1$, where $\{a, b\} = \{a_1, b_1\}$. If we suppose $a_1 = b,\ b_1 = a$, we should obtain $t_1 \Longrightarrow\Downarrow_{\{b',out,b_{n+1}\}}$, transition that u_1 can not match. For the second step, we easily obtain that u_2 must have the following form $\nu\tilde{x}C^{n+1}_{M\cup\{b\},H,Y^-}[q']$ where $q'' \stackrel{\epsilon}{\Longrightarrow} q'$ et $v'' \stackrel{\tau}{\longrightarrow} GSensor_{s,m-1}\langle H, Y^-\rangle$

[3] Let suppose $p \stackrel{\nu b\bar{a}b}{\longrightarrow} p'$.

Then

$$\nu\tilde{x}C^n_{M,H,Y}[p] \stackrel{\tau}{\longrightarrow} \nu b\nu\tilde{x}(p' \parallel ASensor_{r+1,n+1}\langle M \cup \{b\}\rangle \parallel case\langle b, H_1, (R_z)_{z\in H_1}, R'_b\rangle)$$

$$= t_1 \stackrel{\tau}{\longrightarrow} \nu b\nu\tilde{x}(p' \parallel ASensor_{r+1,n+1}\langle M \cup \{b\}\rangle \parallel GSensor_{s+1,m-1}\langle H \cup \{(b, y'_1)\}, Y^-\rangle)$$

$$= t_2 = \nu b\nu\tilde{x}C^{n+1}_{M\cup\{b\},H\cup\{(b,y'_1)\},Y^-}[p']$$

Then it should exist the processes u_1, u_2 derived from $\nu\tilde{x}C^n_{M,H,Y}[q]$ such that

$$\nu\tilde{x}C^n_{M,H,Y}[q] \stackrel{\epsilon}{\Longrightarrow} u_1 \approx_b t_1$$

$$u_1 \stackrel{\epsilon}{\Longrightarrow} u_2 \approx_b t_2$$

The first step: $\nu\tilde{x}C^n_{M,H,Y}[p] \stackrel{\tau}{\longrightarrow} t_1$. The configuration $\nu\tilde{x}C^n_{M,H,Y}[q]$ must do at least a step, since $\nu\tilde{x}C^n_{M,H,Y}[q] \downarrow_{b_n}$ or $t_1 \not\Downarrow_{b_n}$. Since $t_1 \downarrow_{b_{n+1}}$, p and $GSensor$ can not make more than a single visible step. Hence we obtain that u_1 must have the following form:

$$\nu\tilde{x'}(q'' \parallel ASensor_{r+1,n+1}\langle M \cup \{b_1\}\rangle \parallel v'')$$

for some $b_1, q'', v'', \tilde{x'}, q \stackrel{\alpha'}{\Longrightarrow} q'', GSensor_{s,m}\langle H, Y\rangle \stackrel{\alpha''}{\longrightarrow} v''$, where α'' and α' are complementary actions. Since $t_1 \Longrightarrow\Downarrow_{\{a',new,out,b_{n+1}\}}$ we deduce that α' is an output, and α'' must be the corresponding input; moreover, α' must be a bound output. Then $\alpha' = \nu b\overline{a_1}b$, where necessarily $a = a_1$. For the second step, we easily prove that u_2 must have the form $\nu b\nu\tilde{x}C^{n+1}_{M\cup\{b\},H\cup\{(b,y'_1)\},Y^-}[q']$ where $q'' \stackrel{\epsilon}{\Longrightarrow} q'$ and $v'' \stackrel{\tau}{\longrightarrow} GSensor_{s+1,m-1}\langle H \cup \{(b, y'_1)\}, Y^-\rangle$

[4] Let suppose $p \stackrel{a\langle\tilde{b}\rangle?}{\longrightarrow} p'$.

Then

$$\nu\tilde{x}C^n_{M,H,Y}[p] \xrightarrow{\tau}$$

$$\nu\tilde{x}(p' \parallel ASensor_{r+1,n+1}\langle M \cup \{b\}\rangle \parallel (\tau.GSensor_{s+1,m-1}\langle H^+, Y^-\rangle + \tau.W\langle a', b', in\rangle))$$

$$= t_1 \xrightarrow{\tau} \nu\tilde{x}(p' \parallel ASensor_{r+1,n+1}\langle M \cup \{b\}\rangle \parallel GSensor_{s+1,m-1}\langle H^+, Y^-\rangle)$$

$$= t_2 = \nu\tilde{x}C^{n+1}_{M\cup\{b\},H^+,Y^-}[p']$$

Then it should exist the processes u_1, u_2 derived from $\nu\tilde{x}C^n_{M,H,Y}[q]$ such that

$$\nu\tilde{x}C^n_{M,H,Y}[q] \stackrel{\epsilon}{\Longrightarrow} u_1 \approx_b t_1$$

$$u_1 \stackrel{\epsilon}{\Longrightarrow} u_2 \approx_b t_2$$

The first step: $\nu\tilde{x}C^n_{M,H,Y}[p] \xrightarrow{\tau} t_1$. The configuration $\nu\tilde{x}C^n_{M,H,Y}[q]$ must do at least a step, since $\nu\tilde{x}C^n_{M,H,Y}[q] \downarrow_{b_n}$ and $t_1 \not\downarrow_{b_n}$. Since $t_1 \downarrow_{b_{n+1}}$, p or $GSensor$ can not make more than one single visible step. Hence we obtain that u_1 must have the following form:

$$\nu\tilde{x'}(q'' \parallel ASensor_{r+1,n+1}\langle M \cup \{b_1\}\rangle \parallel v'')$$

for some $b_1, q'',\ v'',\ \nu\tilde{x'},\ q \stackrel{\alpha'}{\Longrightarrow} q'',\ GSensor_{s,m}\langle H, Y\rangle \xrightarrow{\alpha''} v''$, where α'' et α' are complementary actions. Since $t_1 \Longrightarrow\Downarrow_{\{a',b',in,b_{n+1}\}}$ then α' must be either an input, either a discard, whilst α'' is the corresponding output. Hence $\alpha' = a_1\langle\tilde{b_1}\rangle?$, $\alpha'' = \overline{a_1}b_1$, where necessarily $\{a, b\} = \{a_1, b_1\}$. If we suppose $a_1 = b,\ b_1 = a$, we should obtain $t_1 \Longrightarrow\Downarrow_{\{b',in,b_{n+1}\}}$, transition that u_1 can not match. For the second step, we easily obtain that u_2 must have the form $\nu\tilde{x}C^{n+1}_{M\cup\{b\},H^+,Y^-}[q']$ with $q'' \stackrel{\epsilon}{\Longrightarrow} q'$ and $v'' \xrightarrow{\tau} GSensor_{s+1,m-1}\langle H^+, Y^-\rangle$

□ *Lemma* 12

Theorem 1. *For all image finite processes p and q,*

1) $p \stackrel{e}{\sim}_b q$ *iff* $p \stackrel{e}{\sim}_\phi q$ *iff* $p \sim q$

2) $p \stackrel{e}{\approx}_b q$ *iff* $p \stackrel{e}{\approx}_\phi q$ *iff* $p \approx q$.

Proof

The assertion of the theorem is an immediate consequence of Corollaries 2 and 4 and of Lemma 12.

□ *Theorem* 1

4 Congruences

In this section we focus on the congruence induced by barbed bisimilarity. Our goal will be to define a relation R over the set $\mathcal{P}_b$ of processes such that if $p\ R\ q$ then p and q are observable on the same set of channels, observability is preserved by reduction, and moreover, when placed in an arbitrary context C, p and q cannot be distinguished ($C[p]\ R\ C[q]$). In this paper we shall concentrate on the strong congruence, for the weak case (which abstracts for internal steps), we shall defer to future work.

Definition 10. Barbed Congruence

Two processes p and q are **barbed congruent**, *shortly $p \overset{c}{\sim}_b q$, if for any context C, $C[p] \sim_b C[q]$.*

Obviously, $\overset{c}{\sim}_b$ is by definition a congruence. But as for barbed equivalence, this definition is more appropriate to prove the non-congruence rather than the congruence of two processes. So we are interested in finding a direct characterization of this congruence based on labelled transitions.

Unfortunately, $\sim$ is not the target characterization, as it follows from the above remark.

Remark 3.
- $\sim$ is not preserved by choice. We have that $a \sim b$, but $a + \bar{c} \not\sim b + \bar{c}$.
- $\sim$ is not preserved by substitution. Let $p \overset{def}{=} \bar{x}.y.\bar{c} + y.(\bar{x} \parallel \bar{c})$ and $q \overset{def}{=} \bar{x} \parallel y.\bar{c}$. Then $p \sim q$, but $p[x/y] \not\sim q[x/y]$.
- $\sim$ is not preserved by prefixing. It is a direct consequence of the previous item.

Unlike strong bisimilarity from the π-calculus, $\sim$ is not preserved by choice. We borrow ideas from [8] and [14] to obtain a congruence relation which do not require closure with respect to contexts.

Definition 11. *Let $\sim_+$ be given by*

1) if $p \xrightarrow{\tau} p'$, then $\exists q'$ such that $q \xrightarrow{\tau} q'$ and $p' \sim q'$,

2) if $p \xrightarrow{\nu \tilde{b}\bar{a}\tilde{c}} p'$ and $\tilde{b} \cap fn(p,q) = \emptyset$, then $\exists q'$ such that $q \xrightarrow{\nu \tilde{b}\bar{a}\tilde{c}} q'$ and $p' \sim q'$,

3) if $p \xrightarrow{a\langle \tilde{b} \rangle} p'$, then $\exists q'$ such that $q \xrightarrow{a\langle \tilde{b} \rangle} q'$ and $p' \sim q'$,

Let $\sim_c$ be given by $p \sim_c q$ if $p\sigma \sim_+ q\sigma$ for all substitutions σ.

Remark 4.
- $\sim_c \subseteq \sim_+ \subseteq \sim$.
- the inclusions are strict.
- $\sim_+$ is preserved by $+$, ν and $\parallel$.

Proof

The inclusions follow directly from the definitions.

To prove that the first inclusion is strict, it suffices to take $p \overset{def}{=} \bar{x}.y.\bar{c} + y.(\bar{x} \parallel \bar{c})$ and $q \overset{def}{=} \bar{x} \parallel y.\bar{c}$. Then $p \sim_+ q$, but $p \not\sim_c q$.

To prove that the second inclusion is strict, it suffices to take $p \overset{def}{=} a$ et $q \overset{def}{=} b$. Then $p \sim q$, but $p \not\sim_+ q$.

The fact that $\sim_+$ is preserved by ν and $\parallel$ can be proved by an analysis by cases, in the same manner as for $\sim$ in Lemmas 8 and 9.

The fact that $\sim_+$ est preserved by $+$ is proved by an analysis by cases.

Let $p \sim_+ q$. We prove that $(p + r) \sim_+ (q + r)$.

If $p + r \xrightarrow{\alpha} s$, then this transition is inferred using the rule (8), from $p \xrightarrow{\alpha} s$ or $r \xrightarrow{\alpha} s$.

If $p \xrightarrow{\alpha} s$, since $p \sim_+ q$, using the Definition 11, we obtain $q \xrightarrow{\alpha} t$ with $s \sim t$. Using the rule (8), we obtain $q + r \xrightarrow{\alpha} t$.

If $r \xrightarrow{\alpha} s$, using the rule (8), we obtain $q + r \xrightarrow{\alpha} s$, and obviously $s \sim s$.

The case $q + r \xrightarrow{\alpha} s$ is similar.

□ *Remark* 4

Lemma 13. *$\sim_c$ is preserved by prefix, restriction, summation, matching and parallelism.*

Proof

- $p \sim_c q \Longrightarrow \alpha.p \sim_c \alpha.q$. for any prefix α.
  - $\alpha = \tau$.
    Let σ be a substitution. Then $p \sim_c q \Longrightarrow p\sigma \ \sim_+ \ q\sigma \Longrightarrow p\sigma \ \sim \ q\sigma \Longrightarrow \tau.(p\sigma) \ \sim_+ \ \tau.(q\sigma) \Longrightarrow (\tau.p)\sigma \ \sim_+ \ (\tau.q)\sigma$.
    Hence $\tau.p \ \sim_c \ \tau.q$.
  - $\alpha = \bar{a}b$.
    Let σ be a substitution. Then $p \sim_c q \Longrightarrow p\sigma \ \sim_+ \ q\sigma \Longrightarrow p\sigma \ \sim \ q\sigma \Longrightarrow (\bar{a}\tilde{b})\sigma.(p\sigma) \ \sim_+ \ (\bar{a}\tilde{b})\sigma.(q\sigma) \Longrightarrow (\bar{a}\tilde{b}.p)\sigma \ \sim_+ \ (\bar{a}\tilde{b}.q)\sigma$.
    We obtain $\bar{a}\tilde{b}.p \ \sim_c \ \bar{a}\tilde{b}.q$.
  - $\alpha = a(\tilde{b})$.
    Let σ be a substitution. Then $p \sim_c q \Longrightarrow p\sigma \ \sim_+ \ q\sigma \Longrightarrow p\sigma \ \sim \ q\sigma \Longrightarrow (a(\tilde{b}).p)\sigma \ \sim_+ \ (a(\tilde{b}).q)\sigma$.
    Hence $a(\tilde{b}).p \ \sim_c \ a(\tilde{b}).q$.
- $p \sim_c q \Longrightarrow \nu ap \sim_c \nu aq$ for any channel $a \in Ch_b$.
  Let σ be a substitution. We can suppose $a \notin (prdom(\sigma) \cup prcod(\sigma))$. Then $p \sim_c q \Longrightarrow p\sigma \ \sim_+ \ q\sigma \Longrightarrow$ (from the Remark 4) $(\nu ap)\sigma \ \sim_+ \ (\nu aq)\sigma$.
  Hence $\nu ap \ \sim_c \ \nu aq$.
- $p \sim_c q \Longrightarrow p + r \sim_c q + r$.
  Let σ be a substitution. Then $p \sim_c q \Longrightarrow p\sigma \ \sim_+ \ q\sigma \Longrightarrow$ (from the Remark 4) $p\sigma + r\sigma \ \sim_+ \ q\sigma + r\sigma \Longrightarrow (p + r)\sigma \ \sim_+ \ (q + r)\sigma$.
  We obtain $p + r \ \sim_c \ q + r$.
- $p \sim_c q \Longrightarrow \langle x = y\rangle p, r \sim_c \langle x = y\rangle q, r \wedge \langle x = y\rangle r, p \sim_c \langle x = y\rangle r, q$.
  Let σ be a substitution. We can prove the implications by an analysis by cases (depending on the relation between $\sigma(x)$ and $\sigma(y)$), using the fact that $\sim_+$ is reflexive.
- $p \sim_c q \Longrightarrow p \parallel r \sim_c q \parallel r$ for any processes $p \in \mathcal{P}_b$.
  Let σ be a substitution. Then $p \sim_c q \Longrightarrow p\sigma \ \sim_+ \ q\sigma \Longrightarrow$ (from the Remark 4) $p\sigma \parallel r\sigma \ \sim_+ \ q\sigma \parallel r\sigma \Longrightarrow (p \parallel r)\sigma \ \sim_+ \ (q \parallel r)\sigma$
  We obtain $p \parallel r \ \sim_c \ q \parallel r$.

□ *Lemma* 13

For a process E which contains a free identifier X, and a process p, we denote $E(p)$ for the process obtained from E by replacing $X\langle\tilde{y}\rangle$ by $p\langle\tilde{y}/\tilde{x}\rangle$ where $\tilde{x}$ denotes the free names of p. For example, if $p \stackrel{def}{=} (x_1, x_2)(\bar{x_1}.x_2 \parallel \bar{x_2})$ and $E \stackrel{def}{=} \bar{a}b.X\langle a, b\rangle + \nu c\bar{a}c.X\langle c, b\rangle$, then $E(p) = \bar{a}b.(\bar{a}.b \parallel \bar{b}) + \nu c\bar{a}c.(\bar{c}.b \parallel \bar{b})$.

Definition 12. *Let E and F be two processes which contain a free identifier X. Then $E \sim F$ (respectively $E \sim_+ F$, $E \sim_c F$) if $E(p) \sim F(p)$ (respectively $E(p) \sim_+ F(p)$, $E(p) \sim_c F(p)$) for any process p.*

We shall use the bisimulations up-to $\sim_+$.

Definition 13. *A symmetric relation S is a* **bisimulation up-to** $\sim_+$ *sif for any $(p, q) \in S$, we have*

1) if $p \xrightarrow{\tau} p'$, *then* $\exists q'$ *such that* $q \xrightarrow{\tau} q'$ *and* $(p', q') \in \sim S \sim$,

2) if $p \xrightarrow{\nu \tilde{b} \bar{a} \tilde{c}} p'$ *and* $\tilde{b} \cap fn(p, q) = \emptyset$, *then* $\exists q'$ *such that* $q \xrightarrow{\nu \tilde{b} \bar{a} \tilde{c}} q'$ *and* $(p', q') \in \sim S \sim$,

3) if $p \xrightarrow{a\langle \tilde{b} \rangle} p'$ *then* $\exists q'$ *such that* $q \xrightarrow{a\langle \tilde{b} \rangle} q'$ *and* $(p', q') \in \sim S \sim$.

Remark that in the Definition 13, we require $(p', q') \in \sim S \sim$ and not $(p', q') \in \sim_+ S \sim_+$.

Next lemma allow to reduce the size of relations used to prove for two processes p and q that $p \sim_+ q$.

Lemma 14. *If* S *is a bisimulation up-to* $\sim_+$ *then* $S \subseteq \sim_+$.

Proof

Let S be a bisimulation up-to $\sim_+$. Then S satisfies the conditions of the Definition 9, and hence $S \subseteq \sim$.

Now let suppose that $(p, q) \in S$ and let $p \xrightarrow{\alpha} p'$, where α satisfies the conditions of the Definition 11. Then by the Definition 13, we obtain that $q \xrightarrow{\alpha} q'$ and $(p', q') \in \sim S \sim$.

Since $S \subseteq \sim$ et $\sim$ is transitive, we obtain $(p', q') \in \sim$.

□ *Lemma* 14

Lemma 15. *Let* E *and* F *be open processes which contain* X *as free identifier. If* $E \sim_c F$, *then* $(rec\ X\langle \tilde{x} \rangle.E)\langle \tilde{x} \rangle \sim_c (rec\ X\langle \tilde{x} \rangle.F)\langle \tilde{x} \rangle$.

Proof

Let $p \stackrel{def}{=} (rec\ X\langle \tilde{x} \rangle.E)\langle \tilde{x} \rangle$, $q \stackrel{def}{=} (rec\ X\langle \tilde{x} \rangle.F)\langle \tilde{x} \rangle$.

We prove that the relation

$$\mathcal{C} \stackrel{def}{=} \left\{ (G(p), G(q)) \;\middle|\; G \text{ contains only the identifier } X \right\}$$

is a bisimulation up-to $\sim_+$. Using the Lemma 14 we obtain $\mathcal{C} \subseteq \sim_+$. Choosing $G \equiv X\langle \tilde{z} \rangle$, we obtain that $p\langle \tilde{z} \rangle \sim_+ q\langle \tilde{z} \rangle$ for any $\tilde{z}$, and hence $p \sim_c q$, which prove the assertion of lemma.

Let $G(p) \xrightarrow{\alpha} p'$ $(*)$.

We shall prove the existence of a corresponding transition $G(q) \xrightarrow{\alpha} q'$ which satisfies the conditions of the Definition 13 by structural induction on the inference of the transition $(*)$. We present only a few cases.

- $G(p) \xrightarrow{\alpha} p'$ using the rule (11).
  Then $G = X\langle \tilde{y} \rangle$, and by a shorter deduction, $E(p)[\tilde{y}/\tilde{x}] \equiv E[\tilde{y}/\tilde{x}](p) \xrightarrow{\alpha} p'$. By induction $E[\tilde{y}/\tilde{x}](q) \xrightarrow{\alpha} q''$ with $p' \sim \mathcal{C} \sim q''$. Since $E \sim_c F$, we obtain $E[\tilde{y}/\tilde{x}](q) \sim_+ F[\tilde{y}/\tilde{x}](q)$; so $F[\tilde{y}/\tilde{x}](q) \xrightarrow{\alpha} q' \sim q''$. Using the rule (11) $G(q) = q\langle \tilde{y} \rangle \xrightarrow{\alpha} q'$. The transitivity of $\sim$ implies $p' \sim \mathcal{C} \sim q'$.
- $G(p) \xrightarrow{\alpha} p'$ using one of the rules which concerns the parallelism (12), (13), (14) or one of theirs symmetric.
  If $G = G_1 \parallel G_2$, and by a shorter deduction, $G_i(p) \xrightarrow{\alpha_i} p'_i$ for some appropriate α_i. By induction $G_i(q) \xrightarrow{\alpha_i} q'_i$ where $p'_i \sim \mathcal{C} \sim q'_i$. Hence $p'_i \sim H_i(p)\ \mathcal{C}\ H_i(q) \sim q'_i$, and using the Lemma 9 we obtain that
  $(p'_1 \parallel p'_2) \sim (H_1(p) \parallel H_2(p))\ \mathcal{C}\ (H_1(q) \parallel H_2(q)) \sim (q'_1 \parallel q'_2)$.

□ *Lemma* 15

Corollary 5. *$\sim_c$ is preserved by recursion.*

Proof

Let $p \sim_c q$ and let $D[\bullet] \stackrel{def}{=} (recX\langle\tilde{x}\rangle.C[\bullet])\langle\tilde{y}\rangle$ be a context such that $D[p]$ and $D[q]$ are well formed ($fn(p \parallel q) \subseteq \tilde{x}$).

Using the Lemma 13 we obtain $C[p]\sim_c C[q]$ (using the Definition 12) , and using the Lemma 15 we obtain $(recX\langle\tilde{x}\rangle.C[p])\langle\tilde{y}\rangle \sim_c (recX\langle\tilde{x}\rangle.C[q])\langle\tilde{y}\rangle$.

□ *Corollary* 5

Using Lemma 13 and Corollary 5, we obtain that indeed $\sim_c$ is a congruence.

Theorem 2. *$\sim_c$ is a congruence.*

Using some convenient contexts, we can prove that $\sim_c$ and $\overset{c}{\sim}_b$ coincide.

Theorem 3. $\sim_c = \overset{c}{\sim}_b$

Proof

- $\sim_c \subseteq \overset{c}{\sim}_b$
  Let C be an arbitrary context and let $p\sim_c q$. By definition $C[p]\sim_c C[q]$. Hence $C[p]\sim C[q]$ and by the lemma 10 $C[p]\sim_b C[q]$.
  We obtain $p \overset{c}{\sim}_b q$.
- $\overset{c}{\sim}_b \subseteq \sim_c$
  Firstly, we remark that with respect to $\stackrel{\alpha}{\longrightarrow}$ any processes is image finite. So the first point 1) of the Theorem 1 can be read
  "for all processes p et q:

$$p \overset{e}{\sim}_b q \quad \text{iff} \quad p \overset{e}{\sim}_\phi q \quad \text{iff} \quad p \overset{e}{\sim} q." \tag{3}$$

  Let $p \overset{c}{\sim}_b q$ and let $C_1[\bullet] \stackrel{def}{=} u(x_1).u(x_2)\ldots u(x_n).([\bullet] + \sum_{i=1}^{n} x_i(x).\bar{v})$, where $fn(p \parallel q) = \{x_1, x_2, \ldots, x_n\}$, and $fn(p \parallel q) \cap \{u, v\} = \emptyset$. We prove that

$$C_1[p]\sim C_1[q] \Longrightarrow p\sim_c q. \tag{4}$$

  Let $\sigma \stackrel{def}{=} [y_i/x_i]$ be a substitution (in the Definition 11 it suffices to restrict us only to substitutions equal to identity anywhere but $fn(p \parallel q)$).
  Let us consider the derivation

$$C_1[p] \stackrel{u(y_1)}{\longrightarrow} R_1 \stackrel{u(y_2)}{\longrightarrow} R_2 \ldots \stackrel{u(y_n)}{\longrightarrow} R_n = p[y_i/x_i] + \sum_{i=1}^{n} y_i(x).\bar{v}$$

  Since $C_1[p]\sim C_1[q]$, it must exist a derivation

$$C_1[p] \stackrel{u(y_1)}{\longrightarrow} S_1 \stackrel{u(y_2)}{\longrightarrow} S_2 \ldots \stackrel{u(y_n)}{\longrightarrow} S_n = q[y_i/x_i] + \sum_{i=1}^{n} y_i(x).\bar{v},$$

  such that

$$p[y_i/x_i] + \sum_{i=1}^{n} y_i(x).\bar{v} \sim q[y_i/x_i] + \sum_{i=1}^{n} y_i(x).\bar{v} \qquad (5)$$

By an analysis on cases, it is easy to prove that the equivalence 5 implies $p[y_i/x_i] \sim_+ q[y_i/x_i]$ (any input of $p[y_i/x_i)$ must be matched by an input of $q[y_i/x_i)$ and conversely).
Since σ is an arbitrary substitution, we obtain the implication 4.
Using the assertions 3 and 4, we have the following implications:

$$(p \overset{c}{\sim}_b q) \implies (C_1[p] \overset{c}{\sim}_b C_1[q]) \implies (C_1[p] \overset{e}{\sim}_b C_1[q])$$

$$\implies (C_1[p] \sim C_1[q]) \implies (p \sim_c q)$$

□ *Theorem* 3

If we denote by $\overset{c}{\sim}_\phi$ the congruence induced by $\sim_\phi$, then it is easy to prove that $\sim_c = \overset{c}{\sim}_b = \overset{c}{\sim}_\phi$.

As for the strong congruences, we obtain similar results for the weak congruences.

Definition 14. Weak barbed congruence

Two processes p and q are **weakly barbed congruent**, *denoted $p \overset{c}{\approx}_b q$, if for any context C, $C[p] \approx_b C[q]$.*

Definition 15. *Let $\approx_+$ be a symmetric relation which satisfies the following conditions:*

1) if $p \xrightarrow{\tau} p'$, then $\exists q'$ such that $q \overset{\tau}{\Longrightarrow} q'$ and $p' \approx q'$,
2) if $p \xrightarrow{\nu \tilde{b} \bar{a} \tilde{c}} p'$ and $\tilde{b} \cap fn(p,q) = \emptyset$, then $\exists q'$ such that $q \overset{\nu \tilde{b} \bar{a} \tilde{c}}{\Longrightarrow} q'$ and $p' \approx q'$,
3) if $p \xrightarrow{a\langle \tilde{b} \rangle} p'$ then $\exists q'$ such that $q \xrightarrow{a\langle \tilde{b} \rangle} q'$ where $q \overset{\tau}{\Longrightarrow} \xrightarrow{a\langle \tilde{b} \rangle :} q'$ and $p' \approx q'$,
4) if $p \xrightarrow{a:}$, then $q \overset{a:}{\Longrightarrow}$.

Let $\approx_c$ defined by: $p \approx_c q$ if $p\sigma \approx_+ q\sigma$ for all substitutions σ.

Theorem 4. *$\approx_c$ is a congruence.*

Theorem 5. *For all image finite processes p and q*
$p \approx_c q$ iff $p \overset{c}{\approx}_b q$.

5 Axiomatisation of strong congruence

In this section we give a complete axiomatisation of strong congruence for finite processes (without recursion). Our axiomatisation is derived from those given for the π-calculus by Parrow and Sangiorgi in [14]. But we need to take care of the fact that strong congruence $\sim_c$ is not directly obtained form strong bisimilarity $\sim$ by closure with respect to all substitutions (as for the π-calculus), but from a strictly stronger relation $\sim_+$. The gap between $\sim$ and $\sim_+$ is filled by the new axiom (H) (which does not hold for strong congruence in π-calculus), which corresponds to the axiom $P - Noisy$ given for CBS in [8]. To keep the syntax simple, we use the monadic version of our calculus.

5.1 Characterizing strong congruence over simple processes

In this subsection we restrict our attention to processes given by

$$p \quad ::= \quad nil \mid \pi.p \mid p_1 + p_2 \mid \phi p_1, p_2$$

where π belongs to the set of prefixes $\pi ::= \tau \mid x(y) \mid \bar{x}y,\ x, y \in Ch_b$.

Following [14] we use the more general form $\phi p_1, p_2$ with

$$\phi \quad ::= \quad \langle x = y \rangle \mid \neg\phi \mid \phi_1 \wedge \phi_2$$

with $x, y \in Ch_b$, and we use the shortcut ϕp to stand for $\phi\ p, nil$ and $\langle x \neq y \rangle p, q$ for $\neg\langle x = y \rangle p, q$. We denote by $In(p)$ the set of all input ports of p (the set of names a such that $p \xrightarrow{a(x)} p'$ for some p').

The axiom system $\mathcal{A}$ for strong congruence $\sim_c$ is given in Table 6.

(A)	if p and q are alpha-equivalent, then $p = q$
(IP)	if $p = q$ then $\alpha.p = \alpha.q$
(IC)	if $p = q$ then $\phi p = \phi q$
(IS)	if $p = q$ then $p + r = q + r$
(H)	if $x \notin fn(p)$ and $\forall b \in In(p) \phi \Rightarrow \langle a \neq b \rangle$ then $\alpha.p = \alpha.(p + \phi a(x).p)$
$(S1)$	$p + nil = p$
$(S2)$	$p + p = p$
$(S3)$	$p + q = q + p$
$(S4)$	$(p + q) + r = p + (q + r)$
$(C3)$	if $\phi \iff \psi$ then $\phi p = \psi p$
$(C4)$	$False\ p = False\ q$
$(C5)$	$\phi p, p = p$
$(C6)$	$\phi p, q = \neg\phi q, p$
$(CC1)$	$\phi(\psi p) = [\phi \wedge \psi]p$
$(SC1)$	$\phi(p_1 + p_2), (q_1 + q_2) = \phi p_1, q_1 + \phi p_2, q_2$
$(CP1)$	if $bn(\alpha) \cap n(\phi) = \emptyset$ then $\phi(\alpha.p) = \phi(\alpha.\phi p)$
$(CP2)$	$\langle x = y \rangle \alpha.p = \langle x = y \rangle (\alpha\{x/y\}).p$
(SP)	$a(x).p + a(x).q = a(x).p + a(x).q + a(x).(\langle x = y \rangle p, q)$

Table 6. Axiom system $\mathcal{A}$ for strong congruence.

We write $\mathcal{A} \vdash p = q$ whenever $p = q$ can be proved using the rules of the Table 6. The following theorem is easy to prove:

Theorem 6. (soundness of $\mathcal{A}$ for $\sim_c$) *If $\mathcal{A} \vdash p = q$ then $p \sim_c q$.*

Proof

It suffices to prove the correction of rules given in the Table 6. The assertion follows then by induction on the length of the inference of $\mathcal{A} \vdash p = q$.

We justify only the correction of the axioms (H) and (SP). The other cases are simpler.

– (H) if $x \notin fn(p)$ and $\forall b \in In(p)\big(\phi \Rightarrow \langle a \neq b\rangle\big)$ then $\alpha.p = \alpha.(p + \phi a(x).p)$
Let σ be a substitution. We can suppose that $x \notin (prdom(\sigma) \cup prcod(\sigma)$.
We must prove that $(\alpha.p)\sigma \sim_+ (\alpha.(p + \phi a(x).p))\sigma$. Using the Definition 11, it suffices to prove $p\sigma \sim (p + \phi a(x).p)\sigma$, i.e. $p\sigma \sim p\sigma + (\phi\sigma)\sigma(a)(x).p\sigma$.
If σ does not agree with ϕ (Definition 18), then $\phi\sigma \iff False$, and hence $(p\sigma + (\phi\sigma)\sigma(a)(x).p\sigma) \xrightarrow{\alpha} q$ if and only if $p\sigma \xrightarrow{\alpha} q$.
Let suppose that σ agree with ϕ. If $p\sigma \xrightarrow{\alpha} q$, obviously $(p\sigma + (\phi\sigma)\sigma(a)(x).p\sigma) \xrightarrow{\alpha} q$.
Let suppose $(p\sigma + (\phi\sigma)\sigma(a)(x).p\sigma) \xrightarrow{\alpha} q$.
Then the last rule used is the rule (8) and either $p\sigma \xrightarrow{\alpha} q$, either $(\phi\sigma)\sigma(a)(x).p\sigma \xrightarrow{\alpha} q$.
The first case est trivial. Let suppose $(\phi\sigma)\sigma(a)(x).p\sigma \xrightarrow{\alpha} q$.
Then, for some $c \in Ch_b$, $\alpha = \sigma(a)\langle c\rangle$ and since $x \notin fn(p)$, we obtain $q = p\sigma[c/x] = p\sigma$.
Let $b \in In(p)$. Then $\phi \Rightarrow \langle a \neq b\rangle$. Since σ agree with ϕ, using the Definition 18, we obtain $\sigma(b) \neq \sigma(a)$. Using that $In(p\sigma) = \{\sigma(b) | b \in In(p)\}$, we obtain $\sigma(a) \notin In(p\sigma)$, and hence $p\sigma \xrightarrow{\sigma(a):} p\sigma$.
– (SP) $a(x).p + a(x).q = a(x).p + a(x).q + a(x).(\langle x = y\rangle p, q)$
Let σ be a substitution. We can suppose that $x \notin (prdom(\sigma) \cup prcod(\sigma)$.
We must prove that $(a(x).p + a(x).q)\sigma \sim_+ (a(x).p + a(x).q + a(x).(\langle x = y\rangle p, q))\sigma$.
If $(a(x).p + a(x).q)\sigma \xrightarrow{\alpha} q$, obviously $(a(x).p + a(x).q + a(x).(\langle x = y\rangle p, q))\sigma \xrightarrow{\alpha} q$.
Let suppose $(a(x).p + a(x).q + a(x).(\langle x = y\rangle p, q))\sigma \xrightarrow{\alpha} q$.
Then the last used rule is the rule (8) and either $(a(x).p + a(x).q)\sigma \xrightarrow{\alpha} q$, either $(a(x).(\langle x = y\rangle p, q))\sigma \xrightarrow{\alpha} q$. The first case est trivial. Let suppose $(a(x).(\langle x = y\rangle p, q))\sigma \xrightarrow{\alpha} q$. The only interesting case is $\alpha = \sigma(a)\langle c\rangle$ for some $c \in Ch_b$. Hence $q = \langle c = \sigma(y)\rangle(p\sigma)[c/x], (q\sigma)[c/x]$.
If $\sigma(y) = c$, then using the rules (3) and (8), we obtain $(a(x).p + a(x).q)\sigma \xrightarrow{\sigma(a)\langle c\rangle} (p\sigma)[c/x]$, and it is easy to prove that $\langle c = \sigma(y)\rangle(p\sigma)[c/x], (q\sigma)[c/x] \sim (p\sigma)[c/x]$.
If $\sigma(y) \neq c$, then using the rules (3) and (8), we obtain $(a(x).p + a(x).q)\sigma \xrightarrow{\sigma(a)\langle c\rangle} (q\sigma)[c/x]$, and it is easy to prove that $\langle c = \sigma(y)\rangle(p\sigma)[c/x], (q\sigma)[c/x] \sim (q\sigma)[c/x]$.

□ *Theorem* 6

As in [14], it can be proved that for every process, there exists one equivalent process (in the system of axioms $\mathcal{A}$) which is in "normal form", and that congruent processes in normal form can be proved equal in our system of axioms.

Definition 16. *[14] Let V be a set of names; a condition ϕ is complete on V if for some equivalence relation $\mathcal{R}$ on V, it holds that $\phi \Rightarrow \langle x = y\rangle$ iff $x\mathcal{R}y$, and $\phi \Rightarrow \langle x \neq y\rangle$ iff $\neg(x\mathcal{R}y)$.*

Definition 17. (head normal form) *Let V be a set of names. p is in head normal form on V, if it is of the form $\Sigma_{i\in I}\phi_i\alpha_i.\phi_i p_i$, where for all i,*

1 $bn(\alpha_i) \notin V$;
2 ϕ_i is complete on V.

Lemma 16. *For each process p, and for each finite set of names V with $fn(p) \subseteq V$, there is a process h of no greater depth than p and in hnf on V, such that $\mathcal{A} \vdash p = h$.*

See [14] for the proof of Lemma 4.8.

Definition 18. *A substitution σ agrees with a condition ϕ, and ϕ agrees with σ, if for any x, y which occur in ϕ, we have $\sigma(x) = \sigma(y)$ iff $\phi \Rightarrow [x = y]$.*

Lemma 17. *[14] Let V be a set of names and let ϕ be complete on V.*

1. *If σ and σ' are substitution V which agree with ϕ, then $\sigma = \sigma'\rho$ for some injective substitution ρ.*
2. *If ψ is another condition on V, either $\phi \wedge \psi$ is not satisfiable, either $\phi \wedge \psi \iff \phi$.*
3. *If ψ is another condition complete on V such that ϕ and ψ agrees with the same substitution σ then $\phi \iff \psi$.*

By an analysis by cases we can easily prove the next lemma.

Lemma 18. *Let suppose that $p \sim_+ q$ and that σ is injective on $fn(p, q)$. Then $p\sigma \sim_+ q\sigma$.*

Lemma 19. *Let $p \equiv \phi p'$ and $q \equiv \phi q'$, with ϕ complete on a set of names V_1. Let $V_2 = fn(p, q) - V_1$ and σ_1 a substitution such that:*

1 $prdom(\sigma_1) \subseteq V_1$ and σ_1 agrees with ϕ;
2 $prcod(\sigma_1) \cap V_2 = \emptyset$;
3 for any σ_2 with $prdom(\sigma_2) \subseteq V_2$, we have $p\sigma_1\sigma_2 \sim_+ q\sigma_1\sigma_2$

Then $p \sim_c q$.

Proof Similar to the proof of Lemma 4.5 in [14].

□ *Lemma* 19

Using a similar reasoning as in [14] for the proof of the Theorems 4.9 and 4.11, and using the axiom (H) when needed, we can prove the following result:

Theorem 7. (completeness of $\mathcal{A}$ for $\sim_c$) *If $p \sim_c q$ then $\mathcal{A} \vdash p = q$.*

Proof

The proof of Theorem 7 inherits from the proofs of Theorems 4.9 and 4.11 from [14], but we have to take account of the fact the strong congruence $\sim_c$ it is not obtained directly from the strong bisimulation $\sim$ by a closure with respect to substitutions (as in the π-calculus), but from a stronger relation $\sim_+$. The gap between $\sim$ and $\sim_+$ is filled by the new axiom (H).

Using the Lemma 16, it suffices to prove the assertion when p and q are in head normal form on $fn(p, q)$. The proof is by induction on the sum of the depths of p and q.

Let p_{out} be the "output part" of p (the sum of all output prefix summands plus the sum of all τ prefix summands in p), and $p_{\phi,a}$ the sum of summands $\phi_i\alpha_i.p_i$ of p such that ϕ_i is equivalent to ϕ and such that the prefix α_i is the same as $a(x)$ (by taking into account the alpha-conversion and the identification of names implied by ϕ_i).

Using several times $(S3)$ et $(S4)$ we can rewrite p into

$$\mathcal{A} \vdash p = p_{out} + \sum_{a \in In(p), \phi \text{ complete on } V} p_{\phi,a}$$

[1.] Firstly we prove that for any summand of p_{out}, it exists a summand of q which is equal in the axioms system $\mathcal{A}$.

Let $\phi\alpha.p'$ be a summand of p_{out}, and let σ be a substitution which agrees with ϕ; we can suppose that σ coincide with identity anywhere but $fn(p,q)$.

We have $p\sigma \xrightarrow{\alpha\sigma} p'\sigma$. Since $p \sim_c q$, we obtain that $p\sigma \sim_+ q\sigma$. Let $\psi\beta.q'$ be the summand of q used to match the transition of $p\sigma$. By alpha-conversion we can suppose that all the names bound (if any) in α and β are the same. By the definition of $\sim_+$ we have

- σ agrees with ψ
- $\alpha\sigma = \beta\sigma$
- $p'\sigma \sim q'\sigma$.

Since ϕ and ψ are complete on $fn(p,q)$ and that they agree with σ, by the Lemma 17, we obtain $\phi \iff \psi$ and by $(C3)$ $\mathcal{A} \vdash \psi\beta.q' = \phi\beta.q'$. If α and β differ on the names a and b ($\alpha\{a/b\} = \beta\{a/b\}$), since $\alpha\sigma = \beta\sigma$, we obtain $\sigma(a) = \sigma(b)$. Since ϕ agrees with σ, we obtain $\psi \Rightarrow \langle a = b\rangle$ and hence

$$\phi\beta.q' = \phi(\langle a = b\rangle\beta.q') = \phi(\langle a = b\rangle(\beta\{a/b\}).q') \equiv \phi(\langle a = b\rangle(\alpha\{a/b\}).q') = \phi\alpha.q'$$

We can try to prove directly that $p' \sim_+ q'$, because some inputs of one of the processes can be matched by a "discard" from the other processes. But we shall saturate p' and q' such that neither of them can no more discard the inputs of the other.

Let $p' \equiv \phi p''$, $q' \equiv \phi q''$ and let

$$s' \equiv (\phi p'' + \sum\{\phi a(x).\phi p'' \mid a \in In(q''), \forall b \in In(p'')\ \phi \Rightarrow \langle a \neq b\rangle\})$$

with $x \notin fn(p'')$, and

$$t' \equiv (\phi q'' + \sum\{\phi a(x).\phi q'' \mid a \in In(p''), \forall b \in In(q'')\ \phi \Rightarrow \langle a \neq b\rangle\})$$

with $x \notin fn(q'')$.

Using repeatedly the axiom (H) and finally the axiom (IC), we can prove $\mathcal{A} \vdash \phi\alpha.p' = \phi\alpha.s'$ and $\mathcal{A} \vdash \phi\alpha.t' = \phi\alpha.q'$. We can not prove directly by induction $\mathcal{A} \vdash \phi\alpha.s' = \phi\alpha.t'$ using $s' \sim_c t'$ since the sum of the depths of s' and t' is the same as for p and q, so we shall use the intermediary process $p' + q'$.

We prove that $s' \sim_c (p' + q')$ et $(p' + q') \sim_c t'$.

We can prove that $s'\sigma \sim_+ (p' + q')\sigma$ by using the fact that $(\phi p'')\sigma \sim (\phi q'')\sigma$ and that any input of $(\phi p'')\sigma$, which was matched by a "discard", is now matched by a summand $(\phi a(x).\phi q'')\sigma$. In the same manner we prove that $(p' + q')\sigma \sim_+ t'\sigma$. Then, we obtain that $s' \sim_c (p' + q')$ and $(p' + q') \sim_c t'$ using the Lemma 19 (by taking $V_1 = fn(p,q)$ and $\sigma_1 = \sigma$).

Using the induction hypothesis, we get $\mathcal{A} \vdash s' = p' + q'$ and $\mathcal{A} \vdash p' + q' = t'$.

Consequently, $\mathcal{A} \vdash s' = t'$ and using (IP) and (IC) we obtain $\mathcal{A} \vdash \phi\alpha.s' = \phi\alpha.t'$, and hence $\mathcal{A} \vdash \phi\alpha.p' = \psi\beta.q'$.

[2.] Let $p_{\phi,a} \equiv \sum_{i=1}^{n} \phi a(x).p_i$ and $q_{\phi,a} \equiv \sum_{j=1}^{m} \phi a(x).q_i$

For any $i \in [1,n]$ we shall build the processes u_i and r_i such that

$\mathcal{A} \vdash \phi a(x).p_i = \phi a(x).u_i$, $\mathcal{A} \vdash q_{\phi,a} = q_{\phi,a} + \phi a(x).r_i$, and $\mathcal{A} \vdash u_i = r_i$.

Let $V = \{y_1, ..., y_k\}$ be the set of free names of $p_{\phi,a}$ and $q_{\phi,a}$ and let x be a name such that $x \notin fn(p_{\phi,a}, q_{\phi,a})$. Let σ be a substitution which agrees with ϕ (we can suppose that $\forall z \notin \{y_1, \ldots, y_k\}\ \sigma(z) = z$).

$p\sigma \sim_+ q\sigma$ implies $p_{\phi,a}\sigma \sim_+ q_{\phi,a}\sigma$. From $p_{\phi,a}\sigma \stackrel{a\langle x\rangle\sigma}{\longrightarrow} p_i\sigma$, using the definition of $\sim_+$, for any $y \in fn(p_{\phi,a}, q_{\phi,a}) \cup \{x\}$, it exists $J(i,y)$ such that $q_{\phi,a}\sigma \stackrel{a\langle x\rangle\sigma}{\longrightarrow} q_{J(i,y)}\sigma$ et $p_i\sigma\{y/x\} \sim q_{J(i,y)}\sigma\{y/x\}$.

If M is a set of names, we use the notation $[x \notin M]$ to stand for $[\wedge_{z\in M}\langle x \neq z\rangle]$. Let $p_i \equiv \phi p'_i$, $q_j \equiv \phi q'_j$ and let

$$u_i \equiv (\phi p'_i + \sum_{z\in\{y_1,\ldots,y_k\}} \sum_{d\in A_z} [x = z] \wedge \phi d(y).\phi p'_i$$

$$+ \sum_{d\in A_x} [x \notin \{y_1,\ldots,y_k\}] \wedge \phi d(y).\phi p'_i)$$

with $y \notin fn(p'_i)$, $A_z = \{b \mid b \in In(q'_{J(i,z)}), \forall a \in In(p'_i), [x = z] \wedge \phi \Longrightarrow a \neq b\}$,

$A_x = \{b \mid b \in In(q'_{J(i,x)}), \forall a \in In(p'_i), [x \notin \{y_1,\ldots,y_k\}] \wedge \phi \Longrightarrow a \neq b\}$, and let

$$t_j \equiv (\phi q'_j + \sum_{z\in\{y_1,\ldots,y_k\}} \sum_{d\in B_z} [x = z] \wedge \phi d(y).\phi q'_j$$

$$+ \sum_{d\in B_x} [x \notin \{y_1,\ldots,y_k\}] \wedge \phi d(y).\phi q'_j)$$

with $y \notin fn(q'_j)$, $B_z = \{b \mid b \in In(p'_i), \forall a \in In(q'_j), [x = z] \wedge \phi \Longrightarrow a \neq b\}$,

and $B_x = \{b \mid b \in In(p'_i), \forall a \in In(q'_j), [x \notin \{y_1,\ldots,y_k\}] \wedge \phi \Longrightarrow a \neq b\}$.

Let $s_{i,0} \equiv t_{J(i,x)}$, $s_{i,l} \equiv \langle x = y_l\rangle t_{J(i,y_l)}, s_{i,l-1}$, and $r_i \equiv s_{i,k}$.

Using repeatedly the axiom (H) and after the axiom (IC) , we can prove that

$$\mathcal{A} \vdash \phi a(x).p_i = \phi a(x).u_i \tag{6}$$

and $\mathcal{A} \vdash \phi a(x).q_j = \phi a(x).t_j$.

We prove now that $\mathcal{A} \vdash q_{\phi,a} = q_{\phi,a} + \phi a(x).s_{i,l}$ by induction on l.

From $\mathcal{A} \vdash \phi a(x).q_j = \phi a(x).t_j$ and $\mathcal{A} \vdash q_{\phi,a} = q_{\phi,a} + \phi a(x).q_j$ (since $\phi a(x).q_j$ is a summand of $q_{\phi,a}$), we obtain that $\mathcal{A} \vdash q_{\phi,a} = q_{\phi,a} + \phi a(x).t_j$ If $l = 0$, then $\mathcal{A} \vdash q_{\phi,a} = q_{\phi,a} + \phi a(x).t_{J(i,x)}$ follows immediately from the equalities proved previously. By induction let suppose that

$$\mathcal{A} \vdash q_{\phi,a} = q_{\phi,a} + \phi a(x).s_{i,l-1}$$

Since $\mathcal{A} \vdash q_{\phi,a} = q_{\phi,a} + \phi a(x).t_{J(i,y_l)}$ we obtain that $\mathcal{A} \vdash q_{\phi,a} = q_{\phi,a} + \phi a(x).s_{i,l-1} + \phi a(x).t_{J(i,y_l)}$, and by using (SP) we get that $\mathcal{A} \vdash q_{\phi,a} = q_{\phi,a} + \phi a(x).s_{i,l-1} + \phi a(x).t_{J(i,y_l)} + a(x).\langle x = y_l\rangle t_{J(i,y_l)}, s_{i,l-1}$ and after $\mathcal{A} \vdash q_{\phi,a} = q_{\phi,a} + \phi a(x).\langle x = y_l\rangle t_{J(i,y_l)}, s_{i,l-1}$. Making $l = k$ we obtain

$$\mathcal{A} \vdash q_{\phi,a} = q_{\phi,a} + \phi a(x).r_i. \tag{7}$$

Now we prove that $\mathcal{A} \vdash u_i = r_i$.

We can not apply directly the hypothesis of induction on u_i and t_j since the sum of the depths of u_i and t_j is the same as for p and q. So we shall pass by the intermediary processes $p_i + q_j$ with $j \in \{1,\ldots,m\}$. From $p_i\sigma\{y/x\} \sim q_{J(i,y)}\sigma\{y/x\}$, by an analysis by cases, we obtain

$$u_i\sigma\{y/x\} \sim_+ (p_i + q_{J(i,y)})\sigma\{y/x\} \sim_+ t_{J(i,y)}\sigma\{y/x\} \tag{8}$$

(the inputs previously matched by discards, are now matched by "inoffensive" inputs $\phi b(y).\phi p'_i$ or $\phi b(y).\phi q'_{J(i,y)}$).

The condition ϕ does not mention x, and it is possible that it is not complete on $fn(p_i, q_{J(i,y)}) = fn(p_{\phi,a}, q_{\phi,a}) \cup \{x\}$; so we can not use directly the Lemma 19; but as in [14] we can complete it by adding a conditional which agrees with $\{y/x\}$. From the equation 8 we obtain $\langle x = y\rangle u_i\sigma\{y/x\} \sim_+ \langle x = y\rangle(p_i + q_{J(i,y)})\sigma\{y/x\} \sim_+ \langle x = y\rangle t_{J(i,y)}\sigma\{y/x\}$ for $y \in V$ and $\langle x \neq y\rangle u_i\sigma\{y/x\} \sim_+ \langle x \neq y\rangle(p_i + q_{J(i,y)})\sigma\{y/x\} \sim_+ \langle x \neq y\rangle t_{J(i,y)}\sigma\{y/x\}$ for $y \notin V$.

Now we apply the Lemma 19 , where $V_1 = V \cup \{x\}$, $\sigma_1 = \sigma\{y/x\}$ and $V_2 = \emptyset$ in the Lemma, and we get

$$\langle x = y\rangle u_i\sigma\{y/x\} \sim_c \langle x = y\rangle(p_i + q_{J(i,y)})\sigma\{y/x\} \sim_c \langle x = y\rangle t_{J(i,y)}\sigma\{y/x\} \text{ for } y \in V \quad (9)$$

and

$$\langle x \neq y\rangle u_i\sigma\{y/x\} \sim_c \langle x \neq y\rangle(p_i + q_{J(i,y)})\sigma\{y/x\} \sim_c \langle x \neq y\rangle t_{J(i,y)}\sigma\{y/x\} \text{ for } y \notin V \quad (10)$$

and by using the hypothesis of induction, we obtain

$$\mathcal{A} \vdash \langle x = y\rangle u_i = \langle x = y\rangle(p_i + q_{J(i,y)}) = \langle x = y\rangle t_{J(i,y)} \text{ for } y \in \{y_1, \ldots, y_k\}$$

and

$$\mathcal{A} \vdash [x \notin V]u_i = [x \notin V](p_i + q_{J(i,y)}) = [x \notin V]t_{J(i,y)} \text{ for } y \notin \{y_1, \ldots, y_k\}.$$

As in the proof of the Theorem 4.11 in [14], using 9 10 we can prove $\mathcal{A} \vdash u_i = r_i$ and then

$$\mathcal{A} \vdash \phi a(x).u_i = \phi a(x).r_i. \quad (11)$$

From 6, 7 et 11 we obtain that for any $i \in [1, n]$ $\mathcal{A} \vdash q_{\phi,a} = q_{\phi,a} + \phi a(x).p_i$, and then $\mathcal{A} \vdash q_{\phi,a} = q_{\phi,a} + p_{\phi,a}$. By a symmetrical argument, we obtain $\mathcal{A} \vdash p_{\phi,a} = p_{\phi,a} + q_{\phi,a}$ and hence $\mathcal{A} \vdash q_{\phi,a} = p_{\phi,a}$

□ *Theorem* 7

Moreover, our axioms are independent (this follows from the fact that in [14] it is proved that all axioms, but (H), are independent, and that (H) cannot be proved from the others).

5.2 Adding restriction operator

To the grammar given in the previous section, we add the restriction operator:

$$p ::= \ldots \mid \nu x p$$

The axioms to deal with restriction are given in Table 7.

The only axiom which is new (and which does not hold in the π-calculus) is $(RP2)$. The soundness of all axioms is easy to prove. For the completeness, the axioms from Table 7 . are used to push a restriction inside a term until either it disappears or it gives rise to a bound output. The definition for the normal form changes slightly: $\Sigma_{i\in I}\phi_i\alpha_i.\phi'_i p_i$, where for all i,

1 $bn(\alpha_i) \notin V$,
2 ϕ_i is complete on V,
3 $\phi_i = \phi_i'$ if α is τ, an input or a free output,
4 $\phi_i' = \phi_i \wedge (\wedge_{z \in V} \langle x \neq z \rangle)$.

The proof of the completeness is then similar as for Theorem 7.

(IR) if $p = q$ then $\nu x p = \nu x q$
(R) $\nu x\ nil = nil$
(RR) $\nu x \nu y p = \nu y \nu x p$
(RS) $\nu x(p + q) = \nu x p + \nu x q$
$(RP1)$ if $x \notin n(\alpha)$ then $\nu x \alpha.p = \alpha.\nu x p$
$(RP2)$ $\nu x\ \bar{x}y.p = \tau.\nu x p$
$(RP3)$ $\nu x\ x(y).p = nil$
$(RC1)$ if $x \neq y$ then $\nu x \langle x = y \rangle p = nil$
$(RC2)$ if $x \neq y, z$ then $\nu x \langle z = y \rangle p = \langle z = y \rangle \nu x p$

Table 7. The axioms for restriction.

5.3 Adding parallelism

To the grammar given in the previous section, we add the parallel operator:

$$p ::= \ldots \mid p_1 \parallel p_2$$

The axioms needed to deal with parallelism are the expansion axiom given in Table 8. plus the axiom $(P1)$ $p \parallel nil = p$.

In the Table 8., the first summand corresponds to the situation where both processes makes an input. The second and the third summands to the situation where one process makes an output, and the other an input. The fourth and the fifth summands to the situation where one process makes an output and the other a discard. The sixth and the seventh to the situation where one process makes an input and the other a discard. And finally, the eighth and the ninth to the situation where one process makes a silent step τ. To prove the completeness it suffices to eliminate the operator $\parallel$ using the expansion axiom and the axiom $(P1)$.

6 Related work and conclusions

Closest related work to this paper concerns the work on CBS by Prasad [15], [16], and the work by Hennessy and Rathke [8]. In [8], the authors present a process calculus based on broadcast, together with an operational semantics. They also provide simpler characterizations of the congruence induced by barbed bisimilarity, together with complete axiomatisation for congruences (for finite processes). Our bisimilarities are following ideas

Assume

$$p \equiv \sum_{i_1 \in M_1} \phi_{i_1} \overline{x_{i_1}}[v].p_{i_1} + \sum_{i_2 \in M_2} \phi_{i_2} x_{i_2}(v).p_{i_2} + \sum_{i_3 \in M_3} \phi_{i_3} \tau.p_{i_3}$$

and

$$q \equiv \sum_{j_1 \in N_1} \phi_{j_1} \overline{x_{j_1}}[v].q_{j_1} + \sum_{j_2 \in N_2} \phi_{j_2} x_{j_2}(v).q_{j_2} + \sum_{j_3 \in N_3} \phi_{j_3} \tau.q_{j_3}$$

where $[v]$ stand for v (free output) or (v) (bound output). Let $S = \{x_i \mid i \in M_2\}$ and $T = \{x_i \mid i \in N_2\}$. Then:

$p \parallel q =$ $\sum_{(i_2,j_2)}[\phi_{i_2} \wedge \phi_{j_2} \wedge \langle x_{i_2} = x_{j_2}\rangle]x_{i_2}(v).(p_{i_2} \parallel q_{j_2}) +$

$\sum_{i_2}[\phi_{i_2} \wedge [x_{i_2} \notin T]]x_{i_2}(v).(p_{i_2} \parallel q) +$

$\sum_{j_2}[\phi_{j_2} \wedge [x_{j_2} \notin S]]x_{j_2}(v).(p \parallel q_{j_2})$

$\sum_{(i_1,j_2)}[\phi_{i_1} \wedge \phi_{j_2} \wedge \langle x_{i_1} = x_{j_2}\rangle]\overline{x_{i_1}}[v].(p_{i_1} \parallel q_{j_2}) +$

$\sum_{(i_2,j_1)}[\phi_{i_2} \wedge \phi_{j_1} \wedge \langle x_{i_2} = x_{j_1}\rangle]\overline{x_{i_2}}[v].(p_{i_2} \parallel q_{j_1}) +$

$\sum_{i_1}[\phi_{i_1} \wedge [x_{i_1} \notin T]]\overline{x_{i_1}}[v].(p_{i_1} \parallel q) + \sum_{j_1}[\phi_{j_1} \wedge [x_{j_1} \notin S]]\overline{x_{j_1}}[v].(p \parallel q_{j_1}) +$

$\sum_{i_3} \phi_{i_3} \tau.(p_{i_3} \parallel q) + \sum_{j_3} \phi_{j_3} \tau.(p \parallel q_{j_3})$

Table 8. The expansion axiom.

borrowed from their work. However, our calculus focus mainly on the influence of received values (names) by a process on his further possible communications by using a syntax closer to the π-calculus. Our axiomatisation is thus closer to the one given by Parrow and Sangiorgi [14]. The main difference with existing broadcast calculus is the presence of dynamic scoping (versus static scoping of CBS). It is common in concurrent programming to have several groups of processes participating in the same protocol concurrently (using different "channels"). It is then essential that communications be kept separate so that there is no risk of interference between the multiple instances of a protocol executed simultaneously. This is achieved by lexical scoping. Dynamic scoping is then obtained by the combination of local scoping and the ability to send channels along channels.

Concerning the expressiveness of our calculus, it is easy ([4]) to give an implementation (very similar to those given in [2] for a process algebraic approach of Linda) of a Random Access Machine. Also, it is interesting to compare the $b\pi$-calculus with the π-calculus. In [3], we have already proved that "there is no uniform encoding of the $b\pi$-calculus into the π-calculus". The existence of a "good" (compositional) encoding of the $b\pi$-calculus into the π-calculus remains an open question. Conversely, we can give an "uniform" encoding adequate with respect to barbed equivalence of the π-calculus into the $b\pi$-calculus.

Also, even if bisimulations provide a nice method to prove the relation which holds between two equivalents systems (just looking at their states, without building the whole traces set), we can ask if they are not too restrictive? For example, $\bar{a}.(\bar{b} + \bar{c})$ and $\bar{a}.\bar{b} + \bar{a}.\bar{c}$ are not barbed equivalents. This seems surprising, as in our calculus an observer can not influence the behavior of the two processes, nor it can distinguish them; indeed, this is the case in processes algebra based on point-to-point communications (CCS, π-calculus),

where an observer provide to tested process the necessary "co-actions". In a forthcoming paper we analyse the preorders induced by "may testing" in calculi based on broadcast.

Acknowledgements. We have gratefully appreciated the useful comments received from the referees.

References

[1] B. Bayerdorffer. Distributed programming with associative broadcast. In *Proceedings of the 27th Annual Hawaii International Conference on System Sciences. Volume 2: Software Technology (HICSS94-2), Wailea, HW, USA*, pages 353–362, 1994.

[2] N. Busi, R. Gorrieri, and G. Zavattaro. A process algebraic view of linda coordination primitives. *Theoretical Computer Science*, 192(2):167–199, 1998.

[3] C. Ene and T. Muntean. Expressiveness of point-to-point versus broadcast communications. In *Fundamentals of Computation Theory, 12th International Symposium, Lecture Notes in Computer Science*, volume 1684. Springer Verlag, 1999.

[4] C. Ene and T. Muntean. A broadcast-based calculus for communicating systems. Technical report, Laboratoire d'Informatique de Marseille, 2000.

[5] C. Ene and T. Muntean. A broadcast-based calculus for communicating systems. In *6th International Workshop on Formal Methods for Parallel Programming: Theory and Applications, San Francisco*, 2001.

[6] A. Geist, A. Beguelin, J. Dongarra, R. Manchek, W. Jiang, and V. Sunderam. *PVM: A Users' Guide and Tutorial for Networked Parallel Computing*. MIT Press, 1994.

[7] M. Hennessy and R. Milne. Algebraic laws for nondeterminism and concurrency. *Journal of the Association for Computing Machinery*, 32(1):137–161, January 1985.

[8] M. Hennessy and J. Rathke. Bisimulations for a calculus of broadcasting systems. In *CONCUR 95, Lecture Notes in Computer Science*, volume 962. Springer Verlag, 1995.

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

[10] N. Lynch and M. Tuttle. An introduction to input/output automata. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands. Also, Technical Memo MIT/LCS/TM-373, Laboratory for Computer Science, Massachusetts Institute ofTechnology, 2000.

[11] R. Milner. *Communication and concurrency*. Prentice-Hall, 1989.

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

[13] R. Milner and D. Sangiorgi. Barbed bisimulation. In *Proc. of 19-the International Colloquium on Automata, Languages and Programming (ICALP '92), Lecture Notes in Computer Science*, volume 623. Springer Verlag, 1992.

[14] J. Parrow and D. Sangiorgi. Algebraic theories for name-passing calculi. *Information and Computation*, 120(2):174–197, 1995.

[15] K. Prasad. A calculus of broadcasting systems. In *In TAPSOFT'91, Volume 1: CAAP, Lecture Notes in Computer Science*, volume 493. Springer Verlag, 1991.

[16] K. V. S. Prasad. A calculus of broadcasting systems. *Science of Computer Programming*, 25, 1995.

[17] C. Rockl and D. Sangiorgi. A pi-calculus semantics of concurrent idealised algol. In *In Proceedings of Fossacs'99, Lecture Notes in Computer Science*, volume 1578. Springer Verlag, 1999.

[18] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigm*. PhD thesis, University of Edinburgh, 1992.