ELSEVIER Theoretical Computer Science 195 (1998) 291–324

Theoretical Computer Science

On bisimulations for the asynchronous π-calculus[1]

Roberto M. Amadio[a], Ilaria Castellani[b,*], Davide Sangiorgi[b]

[a] *Université de Provence, CMI, rue Joliot-Curie, F-13453, Marseille, France*
[b] *INRIA, 2004 Route des Lucioles, BP 93, Sophia-Antipolis, F-06902, France*

Abstract

The *asynchronous π-calculus* is a variant of the π-calculus where message emission is non-blocking. Honda and Tokoro have studied a semantics for this calculus based on bisimulation. Their bisimulation relies on a modified transition system where, at any moment, a process can perform any input action.

In this paper we propose a new notion of bisimulation for the asynchronous π-calculus, defined on top of the standard labelled transition system. We give several characterizations of this equivalence including one in terms of Honda and Tokoro's bisimulation, and one in terms of *barbed equivalence*. We show that this bisimulation is preserved by name substitutions, hence by input prefix. Finally, we give a complete axiomatization of the (strong) bisimulation for finite terms. © 1998—Elsevier Science B.V. All rights reserved.

Keywords: Asynchronous communication; π-calculus; Bisimulation

1. Introduction

Process interaction in a distributed system is usually modelled by message passing. In this context, one often distinguishes between *synchronous* and *asynchronous* message passing. In the former, the send and receive events can be regarded as happening at the same time. In the latter, one can imagine that messages are sent and travel in the ether till they reach their destination, while the sending process accomplishes other tasks.

In the design of *distributed algorithms* the distinction synchronous vs. asynchronous communication is not considered a very important issue. For instance [19, p. 44], says:

> Messages in distributed systems can be passed either synchronously or asynchronously. (...) For many purposes synchronous message passing can be regarded as a special case of asynchronous message passing (...)

Indeed one can simulate a synchronous communication with two asynchronous ones. On the other hand, in the *language design* community the distinction is brought to

[1] An extended abstract of this paper appears in Proc. CONCUR '96.
* Corresponding author.

the limelight. Basically, asynchronous communication is easier to implement than the synchronous one as it is closer to the communication primitives offered by available distributed systems. In particular, asynchronous communication has become a popular choice in the design of languages for the programming of distributed applications. An early proposal is Agha's actors model [1], while more recent contributions based on the theory of the π-calculus include Pict [16] and the join calculus [6].

A second community where the distinction synchronous vs. asynchronous is gaining momentum is that concerned with the *semantics of programs*. In this community one is often interested in comparing calculi. Certain translations turn out to be fully abstract in an asynchronous setting, where the observer has less power. Examples include the encoding of input-guarded choice [15] into the asynchronous π-calculus, and the encoding of the asynchronous π-calculus into the join calculus [6].

A way to restrict a process calculus to asynchronous communications is to remove output prefixing. In other terms, an asynchronous output $\bar{a}$ followed by a process P is the same as the parallel composition $\bar{a} \mid P$. If the calculus has a non-deterministic sum, then we also disallow output guards. We can justify this decision as follows: (i) An output on a choice point forces synchronizations at the implementation level, this seems to contradict the very essence of asynchronous communication (we are not aware of any programming language which allows this). (ii) At the semantic level a calculus with output guards is more discriminating, in particular certain desirable equations such as (2) in Section 5 fail to hold.

The resulting calculus is still quite expressive when working in a framework where channel names are transmissible values, e.g. the π-calculus [13]. Indeed it is quite easy to simulate the synchronous π-calculus in the asynchronous one: the sending process waits for an acknowledgment from the receiving process on a private channel. Basic results on the expressiveness of the asynchronous π-calculus can be found in the works by Honda and Tokoro [8] and Boudol [4], where the asynchronous π-calculus was first proposed.

When communications are asynchronous, the sender of an output message does not know when the message is actually consumed. In other words, an asynchronous observer, as opposed to a synchronous one, cannot directly detect the input actions of the observed process. Consequently, the asynchronous calculus requires the development of an appropriate semantic framework, as observed by [8].

In this paper we develop a theory of bisimulation for the asynchronous π-calculus both in the strong and in the weak case. Our starting point is an original notion of asynchronous bisimulation over the standard labelled transition system. As a first contribution, we provide several characterizations of this bisimulation, and in particular we study under which conditions it coincides with *barbed equivalence*. We also show that our asynchronous bisimulation coincides with that proposed by Honda and Tokoro, which is based on a modified transition system for the π-calculus, on the sublanguage that they consider. As a second result, we observe that asynchronous bisimulation is preserved by the input prefix of the π-calculus (a similar property is proved in [9]) and coincides with *ground* bisimulation (a bisimulation where only *one* fresh name is considered in the input clause). Finally, we give a complete axiomatization of asynchronous bisimulation in the strong case for finite terms.

Insensitivity to name instantiation (and hence the possibility of using ground forms of bisimulation) appears to depend on having no output prefixing. It does not depend on having asynchronous, rather than synchronous, bisimulation (see [3] for a study of insensitivity to name instantiation for various forms of synchronous bisimulations).

Forms of asynchronous π-calculus have also been studied in [7], but the bisimilarity used is the standard (synchronous) one. Part of our theory, in particular axioms and normal forms, is based on that in [7]. Our formulation of asynchronous bisimulation has been recently used by Nestmann and Pierce [15] to prove the full abstraction of the above-mentioned encoding of input-guarded choice. The paper is organized as follows. In Section 2 we provide the basic definitions. In Section 3 we present various characterizations and properties of *strong* asynchronous bisimulation. In Section 4 we offer a detailed comparison of our work with that of Honda and Tokoro. In Section 5 we study an equational theory which characterizes strong asynchronous bisimulation for finite terms. In Section 6 we adapt some of the results in Section 3 to the *weak* case. Appendix contains longer proofs.

2. Asynchronous π-calculus

The *asynchronous* π-calculus is defined as a subset of the π-calculus where: (i) There is no output prefixing, and (ii) outputs cannot be on a choice point (formally sums are allowed only on input prefixes and τ's). Our language differs from the one proposed in [8, 4] for the presence of a form of choice. This will be important in the axiomatisation (Section 5).

We assume a countable collection *Ch* of channel names, say $a, b, \ldots$ We distinguish between general processes $P, Q, \ldots$ and guards $G, H, \ldots$ as specified in the following grammars:

$$P ::= \bar{a}b \,\big|\, P \mid P \,\big|\, \nu a\, P \,\big|\, !G \,\big|\, G \qquad G ::= \mathbf{0} \,\big|\, a(b).P \,\big|\, \tau.P \,\big|\, G + G \tag{1}$$

In Fig. 1 we define a labelled transition system with early instantiation (rule (*in*)). The actions α are specified as follows: $\alpha ::= \tau \,\big|\, \bar{a}b \,\big|\, \bar{a}(b) \,\big|\, ab$. Conventionally we set $n(\alpha) = fn(\alpha) \cup bn(\alpha)$ where

$$fn(\tau) = \emptyset \qquad fn(\bar{a}(b)) = \{a\} \qquad fn(\bar{a}b) = fn(ab) = \{a, b\},$$

$$bn(\tau) = \emptyset \qquad bn(\bar{a}(b)) = \{b\} \qquad bn(\bar{a}b) = bn(ab) = \emptyset.$$

The rules (*sync*), (*sync*$_{ex}$), (*comp*), and (*sum*) have a symmetric version which is omitted. Indeed, parallel composition and sum should be understood as commutative operators. We denote with $\equiv$ syntactic identity modulo α-renaming and with $fn(P)$ the names free in P.

$(cong)$	$\dfrac{P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q' \equiv Q}{P \xrightarrow{\alpha} Q}$	(τ)	$\dfrac{\cdot}{\tau.P \xrightarrow{\tau} P}$
(in)	$\dfrac{\cdot}{a(b).P \xrightarrow{ac} [c/b]P}$	(out)	$\dfrac{\cdot}{\bar{a}b \xrightarrow{\bar{a}b} \mathbf{0}}$
(out_{ex})	$\dfrac{P \xrightarrow{\bar{a}b} P' \quad a \neq b}{\nu b P \xrightarrow{\bar{a}(b)} P'}$	(ν)	$\dfrac{P \xrightarrow{\alpha} P' \quad a \notin n(\alpha)}{\nu a P \xrightarrow{\alpha} \nu a P'}$
$(sync)$	$\dfrac{P \xrightarrow{\bar{a}b} P' \quad Q \xrightarrow{ab} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$	$(sync_{ex})$	$\dfrac{P \xrightarrow{\bar{a}(b)} P' \quad Q \xrightarrow{ab} Q' \quad b \notin fn(Q)}{P \mid Q \xrightarrow{\tau} \nu b(P' \mid Q')}$
$(comp)$	$\dfrac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$	(sum)	$\dfrac{G \xrightarrow{\alpha} P}{G + G' \xrightarrow{\alpha} P}$
(rep)	$\dfrac{G \xrightarrow{\alpha} P}{!G \xrightarrow{\alpha} P \mid !G}$		

Fig. 1. Labelled transition system with early instantiation.

The notion of *weak* transition is defined as usual:

$$P \overset{\tau}{\Rightarrow} P' \quad \text{iff } P(\xrightarrow{\tau})^{*}P'$$

$$P \overset{\alpha}{\Rightarrow} P' \quad \text{iff } P \overset{\tau}{\Rightarrow} \cdot \xrightarrow{\alpha} \cdot \overset{\tau}{\Rightarrow} P' \quad (\text{for } \alpha \neq \tau)$$

where, e.g., the notation $P \xrightarrow{\alpha} \cdot \xrightarrow{\alpha'} P'$ stands for $\exists P''(P \xrightarrow{\alpha} P''$ and $P'' \xrightarrow{\alpha'} P')$. We write $\rightarrow$ and $\Rightarrow$ as abbreviations for $\xrightarrow{\tau}$ and $\overset{\tau}{\Rightarrow}$, respectively. The relations $\rightarrow$ and $\Rightarrow$ are often called *reduction* relations.

The first important technical point arises in the definition of *commitment*. In the asynchronous case it seems natural to restrict the observation to the *output* commitments. The intuition is that an observer has no direct way of knowing if the message he has sent has been received. All the sender can do is to introduce an output particle in the system, unless there is an explicitly programmed acknowledgment mechanism there is no way for him to know when the particle is actually consumed.

Definition 1 (*Commitment*). The strong commitment of a process on a channel expresses the fact that the process is ready to send a message on that channel. Formally, $P \downarrow \bar{a}$ if P can make an output action whose subject is a, that is if there exist P', b such that $P \xrightarrow{\bar{a}b} P'$ or $P \xrightarrow{\bar{a}(b)} P'$. The weak commitment is then defined as:

$$P \Downarrow \bar{a} \text{ if } \exists P'(P \Rightarrow P' \text{ and } P' \downarrow \bar{a})$$

From the definition of reduction and commitment the notion of barbed bisimulation is derived in a canonical way. Note that in the following we keep implicit the universal and existential quantifications which are formally necessary in the definition of bisimulation: (for any move of the first process, there is a corresponding move of the second process such that the following is satisfied.)

Definition 2 (*Barbed bisimulation*). A symmetric relation S on π-terms is a (strong) barbed bisimulation if whenever PSQ the following holds:

(1) If $P \downarrow \bar{a}$ then $Q \downarrow \bar{a}$.
(2) If $P \to P'$ then $Q \to Q'$ and $P'SQ'$.

Let $\overset{\bullet}{\sim}$ be the largest barbed bisimulation. The notion of weak barbed simulation is obtained by replacing everywhere the commitment $\downarrow$ with $\Downarrow$, and the transition $\to$ with $\Rightarrow$. We denote with $\overset{\bullet}{\approx}$ the largest weak barbed bisimulation.

A more refined notion of bisimulation can be obtained if we also allow observation of output transitions.

Definition 3 ($o\tau$*-bisimulation*). A symmetric relation S on π-terms is a (strong) $o\tau$-bisimulation if PSQ, $P \xrightarrow{\alpha} P'$, α is not an input action, and $bn(\alpha) \cap fn(Q) = \emptyset$ implies $Q \xrightarrow{\alpha} Q'$ and $P'SQ'$. Let $\sim_{o\tau}$ be the largest $o\tau$-bisimulation. Again, the notion of weak $o\tau$-bisimulation is obtained by replacing strong transitions with weak transitions. We denote with $\approx_{o\tau}$ the largest weak $o\tau$-bisimulation.

Both barbed bisimulation and $o\tau$-bisimulation are too rough to distinguish processes such as $a(b).\bar{c}b$ and $a(b).\bar{d}b$. Clearly these processes exhibit different behaviours when they are put in parallel with a process $\bar{a}b$. It is then natural to refine barbed bisimulation to an equivalence which is preserved by parallel composition. Following [14], we call it barbed equivalence.

Definition 4 (*Barbed equivalence*). The relations of strong and weak barbed equivalence are defined as follows:

$$P \sim_b Q \quad \text{if } \forall R (P \mid R \overset{\bullet}{\sim} Q \mid R)$$

$$P \approx_b Q \quad \text{if } \forall R (P \mid R \overset{\bullet}{\approx} Q \mid R)$$

Another approach consists in looking for a variant of the input clause. This leads to the following notion of asynchronous bisimulation. We will see later (Definition 12) that several other equivalent definitions are possible.

Definition 5 (*Asynchronous bisimulation*). A relation S is an asynchronous bisimulation if it is an $o\tau$-bisimulation and whenever PSQ and $P \xrightarrow{ab} P'$ the following holds:

- either $Q \xrightarrow{ab} Q'$ and $P'SQ'$
- or $Q \xrightarrow{\tau} Q'$ and $P'S(Q' \mid \bar{a}b)$.

Let $\sim_a$ be the largest asynchronous bisimulation. The definition of weak asynchronous bisimulation is obtained by replacing the strong labelled transitions with the weak labelled transitions everywhere. We denote with $\approx_a$ the largest weak asynchronous bisimulation.

Since asynchronous bisimulation is the basic bisimulation considered in this paper, we will call it simply bisimulation in what follows. The following properties are specific to the asynchronous π-calculus (Properties 1 and 2 also depend on the absence of outputs on choice points):

Lemma 6. (1) *If* $P \xrightarrow{\bar{a}b} P'$ *then* $P \sim_a P' \mid \bar{a}b$.
(2) *If* $P \xrightarrow{\bar{a}(b)} P'$ *then* $P \sim_a \nu b(P' \mid \bar{a}b)$.
(3) *If* $P \xrightarrow{\bar{a}b} \cdot \xrightarrow{\alpha} P'$ *then* $P \xrightarrow{\alpha} \cdot \xrightarrow{\bar{a}b} P'$.
(4) *If* $P \xrightarrow{\bar{a}(b)} \cdot \xrightarrow{\alpha} P'$ *and* $b \notin n(\alpha)$ *then* $P \xrightarrow{\alpha} \cdot \xrightarrow{\bar{a}(b)} P'$.
(5) *If* $P \xrightarrow{\bar{a}b} \cdot \xrightarrow{ac} P'$ *and c is fresh, then* $P \xrightarrow{\tau} [b/c]P'$.
(6) *If* $P \xrightarrow{\bar{a}(b)} \cdot \xrightarrow{ac} P'$ *and c is fresh, then* $P \xrightarrow{\tau} \nu b([b/c]P')$.

3. Asynchronous bisimulation, strong case

In this section, we study some properties of strong asynchronous bisimulation (Definition 5). In Section 6, we will discuss how these results can be lifted to the weak case, and relate them to previous work. Since most proofs for the weak case can be trivially adapted to the strong case we delay all proofs to that section. The contributions of the present section can be summarized as follows: (1) We show that bisimulation is preserved by name substitution; (2) We provide several equivalent definitions of bisimulation; (3) We prove that bisimulation and barbed equivalence coincide.

The definition of bisimulation has been given in an *early* style, and thus contemplates the substitution of the bound name of an input with all possible names. In the *ground*[2] style [18], on the other hand, *no* name instantiation is needed in the input clause.

Definition 7 (*Ground bisimulation*). A relation S is a ground bisimulation if it is an $o\tau$-bisimulation and whenever PSQ, $P \xrightarrow{ab} P'$, and $b \notin fn(P \mid Q)$ the following holds:

- either $Q \xrightarrow{ab} Q'$ and $P'SQ'$
- or $Q \xrightarrow{\tau} Q'$ and $P'S(Q' \mid \bar{a}b)$.

We denote with $\sim_g$ the largest ground bisimulation. Weak ground bisimulation is obtained by replacing transitions with weak transitions. We denote with $\approx_g$ the largest weak ground bisimulation.

Theorem 8. *Strong ground bisimulation is preserved by name substitutions.*

An important corollary is that bisimulation and ground bisimulation coincide.

Corollary 9. *Strong bisimulation and strong ground bisimulation coincide:* $\sim_a = \sim_g$.

[2] We use the adjective ground to emphasize the fact that in this bisimulation the formal parameter of an input prefix is treated as a fresh constant. Note that the terminology ground equivalence was used in [13, p. 28], with quite a different meaning.

A second corollary is that bisimulation is preserved by input prefix (a property which fails in the synchronous calculus). We can then easily conclude as follows.

Corollary 10. *Strong bisimulation is a congruence.*

Besides early and ground, other variants of bisimulation which have been studied in the literature are *late* and *open*. The difference among all these variants is in the requirements on closure under name instantiations. Late bisimulation requires that matching input transitions should be adequate for all instantiations of the bound name. In open bisimulation [17] the only constraints on equalities among names are those imposed by name extrusion and are recorded as a distinction in the bisimulation clauses. Moreover, in the synchronous π-calculus strong late and early bisimulations are not congruences because they are not preserved by input prefixes, hence the induced congruences, called late and early congruences, have been introduced. In the asynchronous π-calculus, bisimulation is preserved by name instantiations, and therefore all the above forms of bisimulation coincide. We omit the definitions of late and open (which are best defined on a late transition system) and we simply state the result.

Corollary 11. *Late and open variants of strong (asynchronous) bisimulation coincide with the early strong (asynchronous) bisimulation.*

We have thus demonstrated some interesting mathematical properties of our notion of bisimulation. Our next task will be to give an intuitive justification of this notion. First, we introduce three further definitions of bisimulation, which differ in the formulation of the input clause, and we show them all equivalent to Definition 5. Roughly, 1-bisimulation requires preservation under parallel composition with an output, while 2,3-bisimulations propose variants of the diagram chasing in the input clause (cf. Definition 5).

Definition 12 (*Variants of bisimulation*). An i-bisimulation $(i = 1,2,3)$ is an $o\tau$-bisimulation S such that:

- (*1-bisimulation*) PSQ implies $(P \mid \bar{a}b) S (Q \mid \bar{a}b)$, for all $\bar{a}b$.
- (*2-bisimulation*) PSQ and $P \xrightarrow{ab} P'$ implies
  - either $Q \xrightarrow{ab} Q'$ and $P'SQ'$
  - or $Q \xrightarrow{\tau} Q'$ and there is P'' s.t. $P' \xrightarrow{\bar{a}b} P''$ and $P''SQ'$.
- (*3-bisimulation*) PSQ and $P \xrightarrow{ab} P'$ implies
  - either $Q \xrightarrow{ab} Q'$ and $P'SQ'$
  - or there are P'', P''' s.t. $P' \xrightarrow{\bar{a}b} P''$, $P \xrightarrow{\tau} P'''$ and $P''SP'''$.

We denote with $\sim_i$ the largest i-bisimulation, for $i = 1,2,3$.

Theorem 13 (Characterization). *All definitions of bisimulation are equivalent. That is:* $\sim_a = \sim_1 = \sim_2 = \sim_3$.

Our last result connects bisimulation with barbed equivalence. This result will be discussed in detail for the weak case in Section 6.

Theorem 14. *Let P, Q be processes. Then $P \sim_b Q$ iff $P \sim_a Q$.*

4. Comparison with Honda and Tokoro's bisimulation

Our definition of asynchronous bisimulation (Definition 5) relies on a standard labelled transition system. A different approach is taken by Honda and Tokoro [8] and this is subsequently developed by Honda and Yoshida [10]. They use ordinary bisimulation on a modified labelled transition system where, essentially, every process can do any input action at any time.

The language considered in [8, 10] is an asynchronous π-calculus without sum. We will show that on this restricted language Honda and Tokoro's bisimulation coincides with our asynchronous bisimulation. Let us first recall the rules of Honda and Tokoro's transition system (HT-transition system, for short). Note that since there is no sum in their language, guarded sums G are reduced to guarded processes of the form $\tau.P$ or $a(b).P$, and replication is limited to such processes.

In the HT-transition system the transition relations, which we denote by $\xrightarrow{\alpha}_{HT}$, are defined up to a *structural equivalence* $\equiv_{HT}$ following [2, 12]. This is the smallest equivalence such that:[3]

$$P \equiv Q \;\Rightarrow\; P \equiv_{HT} Q \quad (\equiv \text{ is syntactic identity modulo } \alpha\text{-conversion}),$$

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

$$\nu a\, \nu b\, P \equiv_{HT} \nu b\, \nu a\, P,$$

$$\nu a\, (P \mid Q) \equiv_{HT} P \mid \nu a\, Q \quad \text{if } a \notin fn(P),$$

$$!G \equiv_{HT} G \mid !G,$$

$$P \equiv_{HT} Q \;\Rightarrow\; P \mid R \equiv_{HT} Q \mid R \text{ and } \nu a\, P \equiv_{HT} \nu a\, Q.$$

Then the transitions $\xrightarrow{\alpha}_{HT}$ are inferred using the system of rules in Fig. 1 (without the rules (sum) and $(sync_{ex})$) and with the following changes:

(1) The congruence rule $(cong)$ is replaced by the rule:

$$(cong_{HT}) \quad \frac{P \equiv_{HT} P' \quad P' \xrightarrow{\alpha}_{HT} Q' \quad Q' \equiv_{HT} Q}{P \xrightarrow{\alpha}_{HT} Q}$$

(2) The input rule (in) is replaced by an input rule for the $\mathbf{0}$ process:

$$(in_{HT}) \quad \frac{\cdot}{\mathbf{0} \xrightarrow{ab}_{HT} \bar{a}b}$$

[3] We take here a slightly simpler equivalence than that used in [8], keeping only the clauses that are necessary to infer transitions.

(3) The communication rule (*sync*) is replaced by the rule:

$$(sync_{HT})\quad \frac{\cdot}{\bar{a}c \mid a(b).P \xrightarrow{\tau}_{HT} [c/b]P}$$

The (strong) bisimulation equivalence[4] based on this transition system, denoted $\asymp_{HT}$, is defined as the largest HT-bisimulation.

Definition 15 (*HT-bisimulation*). A relation S is a *HT-bisimulation* if it is an $o\tau$-bisimulation and whenever PSQ and $P \xrightarrow{ab}_{HT} P'$ then $Q \xrightarrow{ab}_{HT} Q'$ and $P'SQ'$.

Note the rather special role played by input transitions in the HT-transition system: the transitions $\xrightarrow{ab}_{HT}$ are never consumed in communications; they are only used in the bisimulation to create contexts $[\ ] \mid \bar{a}b$ for testing processes. In fact, every process can perform any input and it is easy to show the following.

Lemma 16. $P \xrightarrow{ab}_{HT} P' \;\Leftrightarrow\; P' \equiv_{HT} P \mid \bar{a}b.$

This property will be the basis for an alternative definition of the HT-transition system, where there is no recourse to a structural equivalence. This new transition system, which we call *direct* HT-transition system, will be easier to compare with ours. It includes two kinds of input transitions:

- Those generated by $\mathbf{0}$ processes, noted $\overset{ab}{\longmapsto}_0$, which are only used in the bisimulation to create contexts $[\ ] \mid \bar{a}b$.
- Those corresponding to input guards $a(b).P$, noted $\overset{ab}{\longmapsto}_1$, which are only used in communications and never tested directly by the bisimulation.

We will use $\overset{\alpha}{\longmapsto}$ to denote a generic transition in the direct HT-transition system. The transition relations $\overset{\alpha}{\longmapsto}$ are defined by the system of rules in Fig. 2, where the symmetric rules for $(in_0), (sync'), (sync'_{ex})$ and $(comp)$ are omitted and in rules $(cong), (\nu), (comp)$ and (rep) we use $\overset{ab}{\longmapsto}$ to denote either kind of input transition. Note that the communication rules $(sync')$ and $(sync'_{ex})$ are based uniquely on the input transitions $\overset{ab}{\longmapsto}_1$ corresponding to input guards. The input transitions $\overset{ab}{\longmapsto}_0$ satisfy a slightly weaker property than that expressed by Lemma 16, namely:

Lemma 17. *The input transitions* $\overset{ab}{\longmapsto}_0$ *satisfy the following*:
- $P \overset{ab}{\longmapsto}_0 P' \Rightarrow P' \equiv_{HT} P \mid \bar{a}b,$
- $P' \equiv_{HT} P \mid \bar{a}b \Rightarrow \exists P''\ (P'' \equiv_{HT} P'$ *and* $P \overset{ab}{\longmapsto}_0 P'').$

Moreover the transitions $\overset{\alpha}{\longmapsto}$ preserve the structural equivalence $\equiv_{HT}$.

[4] In fact Honda and Tokoro define directly the weak bisimulation.

The transition relations $\overset{\alpha}{\mapsto}$ are the smallest relations such that:

$$(cong)\quad \frac{P \equiv P' \quad P' \overset{\alpha}{\mapsto} Q' \quad Q' \equiv Q}{P \overset{\alpha}{\mapsto} Q}$$

$$(in_0)\quad \frac{\cdot}{P \overset{ab}{\mapsto}_0 P \mid \bar{a}b} \qquad\qquad (in_1)\quad \frac{\cdot}{a(b).P \overset{ac}{\mapsto}_1 [c/b]P}$$

$$(\tau)\quad \frac{\cdot}{\tau.P \overset{\tau}{\mapsto} P} \qquad\qquad (out)\quad \frac{\cdot}{\bar{a}b \overset{\bar{a}b}{\mapsto} \mathbf{0}}$$

$$(out_{ex})\quad \frac{P \overset{\bar{a}b}{\mapsto} P' \quad a \neq b}{\nu b P \overset{\bar{a}(b)}{\mapsto} P'} \qquad\qquad (\nu)\quad \frac{P \overset{\alpha}{\mapsto} P' \quad a \notin n(\alpha)}{\nu a P \overset{\alpha}{\mapsto} \nu a P'}$$

$$(sync')\quad \frac{P \overset{\bar{a}b}{\mapsto} P' \quad Q \overset{ab}{\mapsto}_1 Q'}{P \mid Q \overset{\tau}{\mapsto} P' \mid Q'} \qquad\qquad (sync'_{ex})\quad \frac{P \overset{\bar{a}(b)}{\mapsto} P' \quad Q \overset{ab}{\mapsto}_1 Q' \quad b \notin fn(Q)}{P \mid Q \overset{\tau}{\mapsto} \nu b(P' \mid Q')}$$

$$(comp)\quad \frac{P \overset{\alpha}{\mapsto} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P \mid Q \overset{\alpha}{\mapsto} P' \mid Q} \qquad\qquad (rep)\quad \frac{G \overset{\alpha}{\mapsto} P}{!G \overset{\alpha}{\mapsto} P \mid !G}$$

Fig. 2. Direct HT labelled transition system.

Lemma 18. *The transitions* $\overset{\alpha}{\mapsto}$ *satisfy the property*:

$$P \equiv_{HT} Q \overset{\alpha}{\mapsto} Q' \Rightarrow \exists P' \ (P \overset{\alpha}{\mapsto} P' \equiv_{HT} Q').$$

We establish now the correspondence between the two HT-transition systems.

Lemma 19. *The two HT-transition systems are related as follows*:

(1) *If α is an output or τ action, then*
  (i) $P \xrightarrow{\alpha}_{HT} P' \Rightarrow \exists P'' \ (P \overset{\alpha}{\mapsto} P'' \equiv_{HT} P')$,
  (ii) $P \overset{\alpha}{\mapsto} P' \Rightarrow P \xrightarrow{\alpha}_{HT} P'$.

(2) *Moreover, for output transitions $P \overset{\bar{a}b}{\mapsto} P'$ or $P \overset{\bar{a}(b)}{\mapsto} P'$ we have*
  (i) $P \overset{\bar{a}b}{\mapsto} P' \Rightarrow P \equiv_{HT} \nu\vec{u}(\bar{a}b \mid R)$, *$a, b \notin \vec{u}$ and $P' \equiv_{HT} \nu\vec{u} R$*,
  (ii) $P \overset{\bar{a}(b)}{\mapsto} P' \Rightarrow P \equiv_{HT} \nu\vec{u}(\bar{a}b \mid R)$, *$a \notin \vec{u}$, $b \in \vec{u}$ and $P' \equiv_{HT} \nu(\vec{u} \backslash b)R$.*

(3) *Case of input transitions $P \overset{ab}{\mapsto}_0 P'$*:
  (i) $P \xrightarrow{ab}_{HT} P' \Rightarrow \exists P'' \ (P \overset{ab}{\mapsto}_0 P'' \equiv_{HT} P')$,
  (ii) $P \overset{ab}{\mapsto}_0 P' \Rightarrow P \xrightarrow{ab}_{HT} P'$.

(4) *Case of input transitions $P \overset{ab}{\mapsto}_1 P'$*:
  (i) *Let $a, b, c \notin \vec{u}$. Then $\nu\vec{u}(a(c).Q \mid R) \overset{ab}{\mapsto}_1 \nu\vec{u}([b/c]\, Q \mid R)$,*
  (ii) $P \overset{ab}{\mapsto}_1 P' \Rightarrow P \equiv_{HT} \nu\vec{u}(a(c).Q \mid R)$, *$a, b, c \notin \vec{u}$, $P' \equiv_{HT} \nu\vec{u}([b/c]\, Q \mid R)$.*

Proof. Lemma 18 is used in all cases to care for the fact that the transitions $\xrightarrow{\alpha}_{HT}$ are defined up to the structural equivalence $\equiv_{HT}$; then the proof for output transitions is straightforward. Point (2) is an easy consequence of Lemma 17. The proof of (3i) is immediate. Point (3ii) is shown by induction on the proof of $P \overset{ab}{\mapsto}_1 P'$. We give here the proof of point (1) for τ-transitions, which relies on (3).

- We show first that $P \xrightarrow{\tau}_{HT} P' \Rightarrow \exists P''\ (P \overset{\tau}{\mapsto} P'' \equiv_{HT} P')$.
- *Basis*: there are two cases to consider, $\tau.P \xrightarrow{\tau}_{HT} P$ and $\bar{a}c \mid a(b).P \xrightarrow{\tau}_{HT} [c/b]P$. The first case is immediate, since the defining rule is the same in the direct transition system. For the communication case, using rules (*out*), (in_1) and (*sync'*) we can deduce $\bar{a}c \mid a(b).P \overset{\tau}{\mapsto} \mathbf{0} \mid [c/b]P \equiv_{HT} [c/b]P$.
- *Inductive step*: the cases where the last rule used is one of (*comp*), (ν), (*rep*) are straightforward, since the rules are the same in the two transition systems. Suppose now the last rule used is $(cong)_{HT}$. This means that $P \xrightarrow{\tau}_{HT} P'$ is inferred from $P \equiv_{HT} Q \xrightarrow{\tau}_{HT} Q' \equiv_{HT} P'$. By induction we have $Q \overset{\tau}{\mapsto} Q'$. Then by Lemma 18 there exists P'' such that $P \overset{\tau}{\mapsto} P'' \equiv_{HT} Q' \equiv_{HT} P'$.
- We show now that $P \overset{\tau}{\mapsto} P' \Rightarrow P \xrightarrow{\tau}_{HT} P'$.
- *Basis*: there is only one case to consider, $\tau.P \overset{\tau}{\mapsto} P$, which is immediate.
- *Inductive step*: cases where the last rule used is one of (*comp*), (ν), (*rep*), (*sync'*), ($sync'_{ex}$). We only examine the case of (*sync'*): suppose $P \mid Q \overset{\tau}{\mapsto} P' \mid Q'$ because $P \overset{\bar{a}b}{\mapsto} P'$ and $Q \overset{ab}{\mapsto}_1 Q'$. By point (2i) we have $P \equiv_{HT} \nu\vec{u}(\bar{a}b \mid R)$, $a, b \notin \vec{u}$ and $P' \equiv_{HT} \nu\vec{u}R$. Similarly, by point (4ii) $Q \equiv_{HT} \nu\vec{v}(a(c).S \mid S')$, $a, b, c \notin \vec{v}, Q' \equiv_{HT} \nu\vec{v}([b/c]S \mid S')$. Then, supposing $\vec{u} \cap \vec{v} = \emptyset$ and $\vec{u} \cap fn(Q) = \emptyset = \vec{v} \cap fn(P)$, we have, by rule ($sync_{HT}$):

$$\begin{aligned} P \mid Q &\equiv_{HT} \nu\vec{u}\vec{v}(R \mid \bar{a}b \mid a(c).S \mid S') \xrightarrow{\tau}_{HT} \nu\vec{u}\vec{v}(R \mid [b/c]S \mid S') \\ &\equiv_{HT} \nu\vec{u}R \mid \nu\vec{v}([b/c]S \mid S') \equiv_{HT} P' \mid Q' \end{aligned}$$

whence, by rule ($cong_{HT}$), $P \mid Q \xrightarrow{\tau}_{HT} P' \mid Q'$. □

The bisimulation equivalence based on the direct transitions $\overset{\alpha}{\mapsto}$, denoted $\sim_{HT}$, is defined as may be expected.

Definition 20 (*Direct HT-bisimulation*). A relation S is a direct HT-bisimulation if it is an $o\tau$-bisimulation and whenever PSQ and $P \overset{ab}{\mapsto}_0 P'$ then $Q \overset{ab}{\mapsto}_0 Q'$ and $P'SQ'$.

Using Lemma 19 one can show the following.

Proposition 21. $\asymp_{HT} = \sim_{HT}$.

We prove now the coincidence of $\sim_{HT}$ with our asynchronous bisimulation $\sim_a$. The correspondence between the HT-transition system $\overset{\alpha}{\mapsto}$ and ours is quite direct (note that there is no counterpart for the transitions $\overset{ab}{\mapsto}_0$ in our system):

Lemma 22. *The direct HT-transition system and the lts in Fig.* 1 *are related as follows*:

(1) $P \overset{\bar{a}b}{\mapsto} P' \Leftrightarrow P \xrightarrow{\bar{a}b} P'$,
(2) $P \overset{ab}{\mapsto}_1 P' \Leftrightarrow P \xrightarrow{ab} P'$,
(3) $P \overset{\tau}{\mapsto} P' \Leftrightarrow P \xrightarrow{\tau} P'$.

We are now ready to show that $\sim_{HT}$ coincides with $\sim_a$. The proof is rather straightforward if we take the characterization of $\sim_a$ as $\sim_1$. In fact the coincidence of $\sim_{HT}$ (in its original formulation $\asymp_{HT}$) with $\sim_1$ (precisely the corresponding of $\sim_1$ on the *HT*-transition system) was already stated in [9] for the weak versions of the bisimulations.

Proposition 23. $\sim_{HT} = \sim_1$.

Proof. (i) $\sim_1 \subseteq \sim_{HT}$. We show that $\sim_1$ is a direct *HT*-bisimulation. Suppose $P \sim_1 Q$. We only have to check the input clause, so let $P \overset{ab}{\mapsto}_0 P'$. By Lemma 17 $P' \equiv_{HT} P \mid \bar{a}b$. By rule (in_0) we have $Q \overset{ab}{\mapsto}_0 Q \mid \bar{a}b$. Since $P \sim_1 Q$, by definition also $P \mid \bar{a}b \sim_1 Q \mid \bar{a}b$, and thus, since $\equiv_{HT} \subseteq \sim_1$, we conclude $P' \sim_1 P \mid \bar{a}b \sim_1 Q \mid \bar{a}b$.

(ii) $\sim_{HT} \subseteq \sim_1$. Suppose $P \sim_{HT} Q$. We want to show that $P \mid \bar{a}b \sim_{HT} Q \mid \bar{a}b$. But this is immediate because $P \overset{ab}{\mapsto}_0 P \mid \bar{a}b$, and since $P \sim_{HT} Q$, there exists Q' such that $Q \overset{ab}{\mapsto}_0 Q'$ and $P \mid \bar{a}b \sim_{HT} Q'$. By Lemma 17 we have $Q' \equiv_{HT} Q \mid \bar{a}b$. Thus, since $\equiv_{HT} \subseteq \sim_{HT}$, $P \mid \bar{a}b \sim_{HT} Q' \sim_{HT} Q \mid \bar{a}b$. □

We conclude this section with some remarks to support our alternative formulation of asynchronous bisimulation. We have seen that in the *HT*-transition system any process P can perform any input ab. Although rule (in_{HT}) directly represents the notion of asynchronous observer, which (quoted from [8]) "just sends asynchronous messages to the process and – possibly continuing to send further messages – waits for output messages from the process", we think it not so appealing because: (i) it introduces an infinite branching, and therefore makes it harder to prove process bisimilarities (for instance, *all* bisimulation relations are infinite) (ii) it is not obviously compatible with a calculus including choice or other dynamic operators (in particular $\mathbf{0}$ fails to be a unit for the choice operator, at least with the usual rule for choice), and (iii) it reflects the notion of observation rather than the computational content of processes.

5. Equational theory, strong case

We present now an equational theory which characterizes strong asynchronous bisimulation on finite terms. In the rest of this section we shall concentrate on the restricted language without replication. In this case, the following equation summarizes the differences between the synchronous and the asynchronous bisimulations:

$$a(b).(\bar{a}b \mid P) + \tau.P = \tau.P \quad b \notin fn(P). \tag{2}$$

(A) (*α-conversion*) $P \equiv Q \Rightarrow P = Q$

(S1) $G + \mathbf{0} = G$ (P1) $P \mid \mathbf{0} = P$
(S2) $G + G' = G' + G$ (P2) $P \mid Q = Q \mid P$
(S3) $G + (G' + G'') = (G + G') + G''$ (P3) $P \mid (Q \mid R) = (P \mid Q) \mid R$
(S4) $G + G = G$

(R1) $\nu a(\sum_{i=1}^{n} \alpha_i . P_i) = \sum\{\alpha_i \nu a P_i \mid i \in I,\ a \notin fn(\alpha_i)\} \quad \forall i\ a \notin bn(\alpha_i)$
(R2) $\nu a(p \mid Q) = P \mid \nu a Q$ if $a \notin fn(P)$
(R3) $\nu a\, \nu b\, P = \nu b\, \nu a\, P$

(EXP) (*Expansion Theorem*) *Let* $J \cap K = \emptyset = L \cap M$, $b \notin fn(Q)$, $d \notin fn(P)$, *and*

$$P = \left(\sum_{j \in J} \tau . P_j + \sum_{k \in K} a_k(b) . P_k \right) \quad \textit{and} \quad Q = \left(\sum_{\ell \in L} \tau . Q_\ell + \sum_{m \in M} c_m(d) . Q_m \right).$$

Then

$$P \mid Q = \sum_{j \in J} \tau . (P_j \mid Q) + \sum_{k \in K} a_k(b) . (P_k \mid Q) + \sum_{\ell \in L} \tau . (P \mid Q_\ell) + \sum_{m \in M} c_m(d) . (P \mid Q_m).$$

(OABS) (*Output Absorption*) Let I, J, K be disjoint, $h \in I \backslash Fire(\nu \vec{u} \prod_{i \in I} \overline{a_i} b_i)$ and $b \notin \{a_h, b_h\}$. Then

$$\nu \vec{u} \left(\prod_{i \in I} \overline{a_i} b_i \middle| \left(\sum_{j \in J} \tau . P_j + \sum_{k \in K} a_k(b) . P_k \right) \right)$$
$$= \nu \vec{u} \left(\prod_{i \in I \backslash \{h\}} \overline{a_i} b_i \middle| \left(\sum_{j \in J} \tau . (\overline{a_h} b_h \mid P_j) + \sum_{k \in K} a_k(b) . (\overline{a_h} b_h \mid P_k) + \sum_{\substack{k \in K \\ a_k = a_h}} \tau . [b_h / b] P_k \right) \right).$$

(IABS) (*Input Absorption*) $a(b) . (\overline{a} b \mid P) + \tau . P = \tau . P \qquad b \notin fn(P)$

Fig. 3. Axioms $\mathscr{A}$.

The reader should pause to formally verify this equation according to Definition 5. A particular instance of Eq. (2) is $a(b) . \overline{a} b + \tau = \tau$ which intuitively says that the process that emits what it has just received can be "absorbed" in an internal action. Our axiom system is reported in Fig. 3. We recall that $\equiv$ denotes syntactic identity modulo α-renaming.

The proof of completeness relies on a non-standard notion of normal form. Let us first observe that, due to the absence of output prefix in the syntax, the parallel operator cannot be completely eliminated via an expansion theorem. Unrestricted outputs will continue to be present as parallel components in normal forms, and their possible communications with the rest of the process will remain potential (that is, they will not give rise to an explicit τ-action in the normal form). Our notion of normal form originates from that introduced in [7] in a different semantic context. In that work the equational theory captures strong synchronous, rather than asynchronous, bisimulation; the axiom system is essentially the same as that in Fig. 3 but without Eq. (2).

We introduce some notation. Let $\prod_{i\in I}\overline{a_i}b_i$ denote a product of outputs, defined up to the laws (P1–3) in Fig. 3 (monoid laws for $|$). We shall use $\vec{c}$ to denote a sequence of names $c_1,\dots,c_m$. If $\vec{c}=c_1,\dots,c_m$, we let $v\vec{c}P$ stand for $vc_1\dots vc_mP$. If $\vec{c}=\varepsilon$ (the empty sequence), we let by convention $v\varepsilon P\equiv P$. With a slight abuse of notation, we will sometimes use $\vec{c}$ also to represent the set $\{c_1,\dots,c_m\}$ (this relies on axioms (R1–3)). We define now the set $\mathit{Fire}(v\vec{c}\prod_{i\in I}\overline{a_i}b_i)$ of indices of firable outputs of $v\vec{c}\prod_{i\in I}\overline{a_i}b_i$.

Definition 24. Let $P\equiv v\vec{c}\prod_{i\in I}\overline{a_i}b_i$. Then $\mathit{Fire}(P)=\bigcup_{n\in\omega}\mathit{Fire}_n(P)$, where $\mathit{Fire}_n(P)$ is the set of indices of outputs that can be fired after exactly n steps, given by:

$$\begin{aligned}\mathit{Fire}_0(P)&=\{i\mid a_i\notin\vec{c}\},\\ \mathit{Fire}_{n+1}(P)&=\{i\mid\exists k\in\mathit{Fire}_n(P)\ b_k=a_i\}\setminus\bigcup_{m\leqslant n}\mathit{Fire}_m(P).\end{aligned}$$

Example 25. Let $P=vb\,vc\prod_{i\in I}\overline{a_i}b_i$ with $I=\{1,2,3,4\}$ and $\overline{a_1}b_1=\overline{a}b$, $\overline{a_2}b_2=\overline{a}c$, $\overline{a_3}b_3=\overline{b}c$, and $\overline{a_4}b_4=\overline{c}b$. Then $\mathit{Fire}_0(P)=\{1,2\}$, $\mathit{Fire}_1(P)=\{3,4\}$, and $\mathit{Fire}_n(P)=\emptyset$ for $n\geqslant 2$. Hence $\mathit{Fire}(P)=I$. Note that by construction $\mathit{Fire}_n(P)\cap\mathit{Fire}_m(P)=\emptyset$ if $n\neq m$.

Let $=_{SP}$ be the congruence induced by the laws (S1)–(S4), (P1)–(P3) in Fig. 3 (commutative monoid laws and idempotence for $+$, and commutative monoid laws for $|$).

Definition 26. A *normal form* is a term defined up to (S1)–(S3) and (P1)–(P3) of the form:

$$v\vec{c}\left(\prod_{i\in I}\overline{a_i}b_i\,\middle|\,\left(\sum_{j\in J}\tau.P_j+\sum_{k\in K}a_k(b).P_k\right)\right)$$

where the sets I,J,K are pairwise disjoint, each P_j,P_k is a normal form, and supposing $\vec{c}=c_1,\dots,c_m$, the following conditions are satisfied:
(1) (All restricted names are emitted) $\forall\ell\in\{1,\dots,m\}\ \exists i\in I\ b_i=c_\ell$.
(2) (All outputs are firable) $\mathit{Fire}(v\vec{c}\prod_{i\in I}\overline{a_i}b_i)=I$.
(3) (*Non-redundancy*) $\forall k\forall j\ P_k\neq_{SP}(\overline{a_k}b\mid P_j)$.
By convention $\prod_{i\in I}\overline{a_i}b_i\equiv\mathbf{0}$ if $I=\emptyset$ (and similarly for the sums $\sum_{j\in J}\tau.P_j$ and $\sum_{k\in K}a_k(b).P_k$). Thus $\mathbf{0}$ is a normal form, when $\vec{c}=\varepsilon$ and $I=J=K=\emptyset$. A *guarded normal form* is a normal form such that $\vec{c}=\varepsilon$ and $I=\emptyset$.

We will show that each term P can be reduced to a normal form using axioms $\mathscr{A}$ in Fig. 3. Most axioms are standard: (EXP) is an instance of the expansion theorem applied to guards, (OABS) is a form of expansion in which the output particles which are not firable are forced to synchronize or to be postponed. Let $=_{\mathscr{A}}$ denote the congruence induced by these axioms. The proof of normalisation uses nested induction on the depth and on the structure of P.

Definition 27. The *depth* of a finite process P is denoted by $d(P)$, and is defined inductively by:

$$d(\mathbf{0})=0; \quad d(\bar{a}b)=1;$$

$$d(a(b).P)=d(\tau.P)=1+d(P);$$

$$d(P\,|\,Q)=d(P)+d(Q);$$

$$d(\nu a\,P)=d(P);$$

$$d(G+F)=max\{\,d(G),d(F)\}.$$

Remark 28. The depth $d(P)$ is an upper bound on the length of the transition sequences of P. It is easy to see that if P' is a subterm of P then $d(P')\leqslant d(P)$.

Lemma 29 (Normalization). *For any finite process P there exists a normal form*:

$$\lceil P\rceil \equiv \nu\vec{c}\left(\prod_{i\in I}\bar{a}_i b_i\,\middle|\,\left(\sum_{j\in J}\tau.P_j+\sum_{k\in K}a_k(b).P_k\right)\right)$$

such that $P=_{\mathscr{A}}\lceil P\rceil$ *and* $d(\lceil P\rceil)\leqslant d(P)$. *In particular, every guarded sum G can be reduced to a guarded normal form* $\lceil G\rceil \equiv \sum_{j\in J}\tau.P_j+\sum_{k\in K}a_k(b).P_k$.

In the proof of our completeness result, we shall use also the following lemma.

Lemma 30 (Separation). *Let P and Q be two normal forms*:

$$P\equiv\nu\vec{u}\left(\prod_{i\in I}\bar{a}_i b_i\,|\,P_\Sigma\right) \quad \textit{and} \quad Q\equiv\nu\vec{v}\left(\prod_{h\in H}\bar{c}_h d_h\,|\,Q_\Sigma\right)$$

where

$$P_\Sigma\equiv\left(\sum_{j\in J}\tau.P_j+\sum_{k\in K}a_k(b).P_k\right) \quad \textit{and} \quad Q_\Sigma\equiv\left(\sum_{\ell\in L}\tau.Q_\ell+\sum_{m\in M}c_m(d).Q_m\right).$$

If $P\sim_a Q$ *then there exists an injective substitution* σ *that renames the set* $\vec{v}$ *into* $\vec{u}$ *and acts as the identity otherwise, such that*:

$$\prod_{i\in I}\bar{a}_i b_i\equiv\sigma\prod_{h\in H}\bar{c}_h d_h \quad \text{and} \quad P_\Sigma\sim_a\sigma Q_\Sigma$$

Theorem 31. *On finite terms, the equivalence* $\sim_a$ *is the congruence generated by the axioms* $\mathscr{A}$.

Proof. *Soundness*: $P =_{\mathscr{A}} Q \Rightarrow P \sim_a Q$. This is the easy part: it is proved by exhibiting appropriate bisimulations for each axiom.

Completeness: $P \sim_a Q \Rightarrow P =_{\mathscr{A}} Q$. Given the normalization lemma and the soundness of the axioms, it is enough to prove the statement for normal forms. So assume $P \equiv \nu\vec{u}(\prod_{i\in I} \overline{a_i}b_i \mid P_\Sigma)$ and $Q \equiv \nu\vec{v}(\prod_{h\in H} \overline{c_h}d_h \mid Q_\Sigma)$, where as usual P_Σ and Q_Σ are the guarded parts of P and Q:

$$P_\Sigma \equiv \left(\sum_{j\in J} \tau.P_j + \sum_{k\in K} a_k(b).P_k\right), \qquad Q_\Sigma \equiv \left(\sum_{\ell\in L} \tau.Q_\ell + \sum_{m\in M} c_m(d).Q_m\right).$$

By the separation lemma we know that there exists a substitution σ such that $\sigma\vec{v} = \vec{u}$, $\sigma w = w$ if $w \notin \vec{v}$, and

$$\prod_{i\in I} \overline{a_i}b_i \equiv \sigma \prod_{h\in H} \overline{c_h}d_h \qquad P_\Sigma \sim_a \sigma Q_\Sigma.$$

We will show, by induction on the sum of depths of P and Q, that $P_\Sigma =_{S2} \sigma Q_\Sigma$. This will imply the required result, namely

$$P =_{S2} \nu\vec{u}\left(\sigma \prod_{h\in H} \overline{c_h}d_h \mid \sigma Q_\Sigma\right) \equiv Q.$$

Note that, if P is a normal form and $P \xrightarrow{\alpha} P'$ (where α is any action), then P' is a normal form such that $d(P') < d(P)$.[5] We will show that

$(*)$ $P_\Sigma =_{S2} P_\Sigma + \sigma Q_\Sigma =_{S2} \sigma Q_\Sigma$.

To this end it is enough to prove

(i) $P_\Sigma =_{S2} P_\Sigma + \tau.\sigma Q_\ell$,

(ii) $P_\Sigma =_{S2} P_\Sigma + \sigma c_m(d).Q_m$.

Then $(*)$ will follow by iteration and by symmetry.

(i) Suppose $P_\Sigma \xrightarrow{\tau} P_j$. Since $P_\Sigma \sim_a \sigma Q_\Sigma$, there exists $\ell \in L$ such that $\sigma Q_\Sigma \xrightarrow{\tau} \sigma Q_\ell$ and $P_j \sim_a \sigma Q_\ell$. By induction $P_j =_{S2} \sigma Q_\ell$ and thus also $\tau.P_j =_{S2} \tau.\sigma Q_\ell$. Then $P_\Sigma =_{S2} P_\Sigma + \tau.\sigma Q_\ell$.

(ii) Let now $P_\Sigma \xrightarrow{a_k b_k} [b_k/b]P_k$. We show first that σQ_Σ is forced to match this move by a transition of the form $\sigma Q_\Sigma \xrightarrow{a_k b_k} [b_k/d]\sigma Q_m$ for some m such that $\sigma c_m d_m = a_k b_k$. For suppose σQ_Σ responds with a transition $\sigma Q_\Sigma \xrightarrow{\tau} \sigma Q_\ell$ for some Q_ℓ such that $[b_k/b]P_k \sim_a \overline{a_k}b_k \mid \sigma Q_\ell$. Since $d(P_k) < d(P)$ and $d(\overline{a_k}b_k \mid \sigma Q_\ell) \leqslant d(Q)$, we have by induction that $P_k =_{S2} \overline{a_k}b_k \mid \sigma Q_\ell$. But then, since $P_\Sigma \sim_a \sigma Q_\Sigma$, there must be $j \in J$ such that $P_\Sigma \xrightarrow{\tau} P_j$ and $P_j \sim_a \sigma Q_\ell$. By induction this implies $P_j =_{S2} \sigma Q_\ell$ and hence $P_k =_{S2} \overline{a_k}b_k \mid P_j$, contradicting the hypothesis that P_Σ is a normal form.

Thus a transition $P_\Sigma \xrightarrow{a_k b_k} [b_k/b]P_k$ is always matched by a transition $\sigma Q_\Sigma \xrightarrow{a_k b_k} [b_k/d]\sigma Q_m$ such that $\sigma c_m d_m = a_k b_k$ and $[b_k/b]P_k \sim_a [b_k/d]\sigma Q_m$. By induction

[5] On the other hand, P' is not in general a subterm of P, so we could not use structural induction on normal forms.

$[b_k/b]P_k =_{S2} [b_k/d]\sigma Q_m$ and therefore also $a_k(b).P_k =_{S2} \sigma c_m(d).Q_m$. Then $P_\Sigma =_{S2} P_\Sigma + \sigma c_m(d).Q_m$. □

6. Asynchronous bisimulation, weak case

In an asynchronous world a process can make an input and then emit it again on the same channel without changing the overall behaviour of the system. Some interesting equations that hold in the weak semantics and that further motivate its study are the following:

$$!(a(b).\bar{a}b) = \mathbf{0},$$

$$a(b).(\bar{a}b \mid a(b).P) = a(b).P,$$

$$a(b).(\bar{a}b \mid G) + G = G.$$

We present the weak versions of Theorems 8 and 13. Our first task is to show that (weak) bisimulation is preserved by substitutions and coincides with ground bisimulation. To this end we first establish some elementary properties whose proof is not completely standard, in particular some work needs to be done to prove transitivity of $\approx_a$ (cf. the Appendix). In the following, $P, Q, R\ldots$ denote processes.

Lemma 32. *Bisimulation is preserved by parallel composition, restriction, replication and guarded sum, and it is included in ground bisimulation*:

(1) *If* $P \approx_a Q$ *then* $P \mid R \approx_a Q \mid R$, $\nu a P \approx_a \nu a Q$, $\alpha.P + R \approx_a \alpha.Q + R$, *and* $!P \approx_a !Q$.
(2) *If* $P \approx_a Q$ *then* $P \approx_g Q$.

Let σ denote a name substitution which is almost everywhere the identity. Whenever we apply a substitution to a process or an action we suppose that the bound names have been renamed so that no conflict can arise, in particular σ acts as an identity on bound names and if $\sigma(c) \neq c$ then $\sigma(c)$ is not a bound name either.

Lemma 33. *The transitions of P and σP can be related as follows*:
(1) *If* $P \xrightarrow{\alpha} P'$ *then* $\sigma P \xrightarrow{\sigma\alpha} \sigma P'$.
(2) *If* $\sigma P \xrightarrow{\alpha'} P''$ *and* α' *is either an output action, or an input action where the received name is fresh, then for some* P', $P \xrightarrow{\alpha} P'$, $\sigma P' \equiv P''$, *and* $\sigma\alpha = \alpha'$.
(3) *If* $\sigma P \xrightarrow{\tau} P''$ *then*
 (a) *either* $P \xrightarrow{\tau} P'$ *and* $\sigma P' \equiv P''$.
 (b) *or* $\sigma a = \sigma d$, $P \xrightarrow{\bar{a}b} \cdot \xrightarrow{dc} P'$ *and* $[b/c]\sigma P' \sim_a P''$ (*c fresh*).
 (c) *or* $\sigma a = \sigma d$, $P \xrightarrow{\bar{a}(b)} \cdot \xrightarrow{dc} P'$ *and* $\nu b([b/c]\sigma P') \sim_a P''$ (*c fresh*).

We are now ready to prove the crucial lemma.

Lemma 34. *If* $P \approx_g Q$ *then* $\sigma P \approx_a \sigma Q$.

Proof. We show that the following relation is a bisimulation *up to* $\sim_a$ *and restriction*:

$$S = \{(\sigma P, \sigma Q) \mid P \approx_g Q, \sigma \text{ substitution}\}. \tag{3}$$

Suppose $\sigma P \xrightarrow{\alpha} P'$. If α is a τ or output action then the "up to" means that there are $\vec{d}$, P'', Q'', Q' such that $\sigma Q \stackrel{\hat{\alpha}}{\Rightarrow} Q'$ and

$$P' \sim_a \nu\vec{d}\,P'' \quad P'' S Q'' \quad \nu\vec{d}\,Q'' \sim_a Q'. \tag{4}$$

If $\alpha \equiv ab$ is an input action then the "up to" means that there are $\vec{d}$, P'', Q'', Q' such that

- either $\sigma Q \stackrel{ab}{\Rightarrow} Q'$ and condition 4 holds.
- or $\sigma Q \stackrel{\tau}{\Rightarrow} Q'$ and

$$P' \sim_a \nu\vec{d}\,P'' \quad P'' S Q'' \quad \nu\vec{d}\,Q'' \sim_a (Q' \mid \bar{a}b). \tag{5}$$

We consider the various cases.

- The case when α is a τ or output action is simple.
- Suppose $\sigma P \xrightarrow{\tau} P'$ and we are not in the previous case. According to Lemma 33(3) we have to consider two cases:

output: Suppose $P \xrightarrow{\bar{a}b} \cdot \xrightarrow{dc} P_1$, where $P' \sim_a [b/c]\sigma P_1, c$ is fresh and $\sigma a = \sigma d$.

We have to consider two subcases:

input: Suppose $Q \stackrel{\bar{a}b}{\Rightarrow} \cdot \stackrel{dc}{\Rightarrow} Q_1$ and $P_1 \approx_g Q_1$. This means that $Q \stackrel{\tau}{\Rightarrow} \cdot \xrightarrow{\bar{a}b} \cdot \stackrel{\tau}{\Rightarrow} \cdot \xrightarrow{dc} \cdot \stackrel{\tau}{\Rightarrow} Q_1$. By Lemma 6(3) we have then $Q \stackrel{\tau}{\Rightarrow} \cdot \xrightarrow{\bar{a}b} \cdot \xrightarrow{dc} \cdot \stackrel{\tau}{\Rightarrow} Q_1$, whence, by Lemma 33(1), $\sigma Q \stackrel{\tau}{\Rightarrow} \cdot \xrightarrow{\sigma\bar{a}b} \cdot \xrightarrow{\sigma dc} \cdot \stackrel{\tau}{\Rightarrow} \sigma Q_1$. Then, by Lemma 6(5) we conclude that $\sigma Q \stackrel{\tau}{\Rightarrow} \cdot \sim_a [b/c]\sigma Q_1$.

τ: Let $Q \stackrel{\bar{a}b}{\Rightarrow} \cdot \stackrel{\tau}{\Rightarrow} Q_1$ and $P_1 \approx_g (Q_1 \mid \bar{d}c)$. By Lemma 6(3) we have $Q \stackrel{\tau}{\Rightarrow} \cdot \xrightarrow{\bar{a}b} Q_1$, and then by Lemma 33(1) there exists S such that $\sigma Q \stackrel{\tau}{\Rightarrow} S \xrightarrow{\sigma\bar{a}b} \sigma Q_1$. By Lemma 6(1) we know that $S \sim_a (\sigma Q_1 \mid \sigma\bar{a}b) \equiv [b/c]\sigma(Q_1 \mid \bar{d}c)$. Then $\sigma Q \stackrel{\tau}{\Rightarrow} \cdot \sim_a [b/c]\sigma(Q_1 \mid \bar{d}c)$ is the matching move.

bound output: Similar to above.

- The last case to consider is when $\sigma P \xrightarrow{ab} P'$. Then we have $P \xrightarrow{a'c} P_1$ where c is a fresh name, $\sigma a' = a$ and $[b/c]\sigma P_1 \equiv P'$. Again there are two cases:

input: If $Q \stackrel{a'c}{\Rightarrow} Q_1$ and $P_1 \approx_g Q_1$ then $\sigma Q \stackrel{ab}{\Rightarrow} [b/c]\sigma Q_1$.

τ: $Q \stackrel{\tau}{\Rightarrow} Q_1$ and $P_1 \approx_g (Q_1 \mid \bar{a'}c)$. Then the matching move is $\sigma Q \stackrel{\tau}{\Rightarrow} \sigma Q_1$, since $\sigma Q_1 \mid \bar{a}b \equiv [b/c]\sigma(Q_1 \mid \bar{a'}c)$. □

Theorem 35. *Weak ground bisimulation and weak bisimulation coincide and they are preserved by substitution.*

Proof. From Lemma 32(2) and Lemma 34 applied with the identity substitution we know that $P \approx_g Q$ iff $P \approx_a Q$. From Lemma 34 we can conclude that both bisimulations are preserved by substitution. □

It follows that weak bisimulation is preserved by all operators but sum (as usual) and that late and open uniform variants of the weak bisimulation coincide with the early bisimulation studied here.

Corollary 36. *If $P \approx_a Q$ then $a(b).P \approx_a a(b).Q$.*

We can generalize the characterization of asynchronous bisimulation in terms of 1-bisimulation to the weak case.

Definition 37. Let S be a weak $o\tau$-bisimulation. We say that S is a *weak 1-bisimulation* if PSQ implies $(P \mid \bar{a}b)S(Q \mid \bar{a}b)$. We denote with $\approx_1$ the largest weak 1-bisimulation.

Theorem 38 (Characterization). *The 1-bisimulation coincides with (asynchronous) bisimulation. That is: $\approx_a = \approx_1$.*

We now relate barbed equivalence and bisimulation. In the weak case our results rely crucially on the matching operator which we introduce next (in the strong case matching is not needed). We suppose that the grammar of the calculus specified in 1, Section 2, is extended by the clause: $P ::= \cdots \mid [a = b]P$. The rule associated to matching in the labelled transition system is:

$$(match) \quad \frac{P \xrightarrow{\alpha} P'}{[c = c]P \xrightarrow{\alpha} P'}$$

We will concentrate on the weak case first. In Remark 41 we indicate how to eliminate matching in the strong case (hence providing a proof for Theorem 14).

Proposition 39. *Let P, Q, R be processes. Then*
(1) *If σ is an injective substitution on $fn(P \mid Q)$ then $P \approx_a Q$ iff $\sigma P \approx_a \sigma Q$.*
(2) *If $P \approx_a Q$ then $P \mid R \approx_a Q \mid R$, for any process R.*
(3) *If $P \approx_a Q$ then $P \approx_b Q$.*

Proof. The proof of (1) is standard. The proof of (2) is shaped upon the one for Lemma 32 (we cannot use directly this lemma because we have extended the calculus with matching). The proof of (3) follows by

$$\begin{aligned} P \approx_a Q &\Rightarrow \forall R (P \mid R \approx_a Q \mid R) \\ &\Rightarrow \forall R (P \mid R \stackrel{\bullet}{\approx} Q \mid R) \\ &\Rightarrow P \approx_b Q \quad \square \end{aligned}$$

We recall that a lts $(Pr, Act, \mapsto)$ is *image finite* if for any process P and action α the set $\{P' \mid P \overset{\alpha}{\mapsto} P'\}$ is finite. We say that a process P is image finite if the lts generated by P is image finite. Image finite processes form an interesting class: w.r.t. strong reduction all processes are image finite (up to renaming of bound names), and w.r.t. weak reduction all finite control processes (cf. [5]) are image finite modulo the equation $\nu a P = P$ for $a \notin fn(P)$.

Theorem 40. *If P and Q are image finite processes (with respect to weak reduction), and $P \approx_b Q$ then $P \approx_a Q$.*

Proof. Let $\mathscr{F}$ be the monotone operator over $\mathscr{P}(Pr \times Pr)$ associated with the definition of asynchronous bisimulation. Suppose $\approx_a^0 = Pr \times Pr$, $\approx_a^{k+1} = \mathscr{F}(\approx_a^k)$, and $\approx_a^\omega = \bigcap_{k<\omega} \approx_a^k$. It is well-known that on an image finite lts the operator $\mathscr{F}$ preserves co-directed sets (the dual of directed sets). In particular, $\mathscr{F}(\approx_a^\omega) = \approx_a^\omega$. It follows that on image finite processes $\approx_a = \approx_a^\omega$. We show that $P \approx_b Q$ implies $P \approx_a^\omega Q$. From the previous remark the theorem follows.

More precisely, we define a collection of tests $R(n,L)$ depending on $n \in \omega$ and L finite set of channel names, and show by induction on n that

$$\exists L, L'(L \supseteq fn(P \mid Q), L' \subseteq L \text{ and } \nu L'(P \mid R(n,L)) \overset{\bullet}{\approx} \nu L'(Q \mid R(n,L))))$$
$$\text{implies } P \approx_a^n Q.$$

If the property above holds then we can conclude the proof by observing:

$$
\begin{aligned}
& P \approx_b Q \\
& \quad \Rightarrow \forall R(P \mid R \overset{\bullet}{\approx} Q \mid R) \\
& \quad \Rightarrow \forall n \in \omega(P \mid R(n,L) \overset{\bullet}{\approx} Q \mid R(n,L)) \quad \text{with } L = fn(P \mid Q),\ L' = \emptyset \\
& \quad \Rightarrow \forall n \in \omega(P \approx_a{}^n Q) \\
& \quad \Rightarrow P \approx_a^\omega Q.
\end{aligned}
$$

We define the tests $R(n,L)$. To this end we introduce an internal choice operator $\oplus$. This is a derived operator defined as follows:

$$P_1 \oplus \cdots \oplus P_n \equiv \nu a(a.P_1 \mid \cdots \mid a.P_n \mid \bar{a}) \quad a \notin fn(P_1 \mid \cdots \mid P_n).$$

When reducing an internal sum we implicitly garbage collect all dead branches. If $X = \{P_1, \ldots, P_n\}$ is a set of processes then $\bigoplus X$ is an abbreviation for $P_1 \oplus \cdots \oplus P_n$. We suppose that the collection of channel names Ch has been partitioned in two infinite well-ordered sets Ch' and Ch''. In the following we have $L' \subseteq L \subset_{finite} Ch''$. We also assume the following sequences of distinct names in Ch':

$$\{b_n, b'_n \mid n \in \omega\}$$
$$\{c_n^\beta \mid n \in \omega \text{ and } \beta \in \{\tau, aa', a, \bar{a}a', \bar{a} \mid a, a' \in Ch''\}\}$$
$$\{c_n'^\beta \mid n \in \omega \text{ and } \beta \in \{aa', a \mid a, a' \in Ch''\}\}$$
$$\{d_n^\beta \mid n \in \omega \text{ and } \beta \in \{a \mid a \in Ch''\}\}$$
$$\{e_n \mid n \in \omega\}$$

The test $R(n,L)$ is defined by induction on n as follows, where we pick a'' to be the first name in the well-ordered set $Ch''\backslash L$. When emitting or receiving a name which is not in L we work up to injective substitution to show that $P \approx_a^n Q$.

$$R(0,L) = \overline{b}_0 \oplus \overline{b'}_0$$

$$R(n,L) = \overline{b}_n \oplus \overline{b'}_n \oplus \quad (\text{for } n>0)$$

$$(\overline{c}_n^{\tau} \oplus R(n-1,L))$$

$$\oplus\{\overline{c}_n^{\bar{a}a'} \oplus (\bar{a}a' \mid R(n-1,L)) \mid a,a' \in L\}$$

$$\oplus\{\overline{c}_n^{\bar{a}} \oplus \nu a'' (\bar{a}a'' \mid R(n-1, L \cup \{a''\})) \mid a \in L\}$$

$$\oplus\{\overline{c}_n^{aa'} \oplus a(a'').(\overline{c'}_n^{aa'} \oplus ([a''=a']\overline{d}_n^{a'} \oplus R(n-1,L))) \mid a,a' \in L\}$$

$$\oplus\{\overline{c}_n^{a} \oplus a(a'').$$

$$(\overline{c'}_n^{a} \oplus (\oplus\{[a''=a']\overline{d}_n^{a'} \mid a' \in L\} \oplus \bar{e}_n \oplus R(n-1, L \cup \{a''\}))) \mid a \in L\}.$$

The base case is trivial, as $\approx_a^n$ is the full relation. We suppose $n>0$, $\nu L' (P \mid R(n,L)) \overset{\bullet}{\approx} \nu L' (Q \mid R(n,L))$, and $P \overset{\alpha}{\Rightarrow} P'$. We proceed by case analysis on the action α to show that Q can match the action α (in the asynchronous sense). We consider here the cases of a free input and a free output. The cases for τ, bound input and bound output are similar, and they are presented in Appendix A.

$\alpha \equiv aa'$ We suppose $a' \in L$. Then

$$\nu L' (P \mid R(n,L)) \overset{\tau}{\Rightarrow} \nu L' (P \mid (\overline{c}_n^{\bar{a}a'} \oplus (\bar{a}a' \mid R(n-1,L)))).$$

This has to be matched by

$$\nu L' (Q \mid R(n,L)) \overset{\tau}{\Rightarrow} \nu L' (Q_1 \mid (\overline{c}_n^{\bar{a}a'} \oplus (\bar{a}a' \mid R(n-1,L)))).$$

We make a further reduction on the lhs:

$$\nu L' (P \mid (\overline{c}_n^{\bar{a}a'} \oplus (\bar{a}a' \mid R(n-1,L)))) \overset{\tau}{\Rightarrow} \nu L' (P' \mid R(n-1,L)).$$

This is matched by

$$\nu L' (Q_1 \mid (\overline{c}_n^{\bar{a}a'} \oplus (\bar{a}a' \mid R(n-1,L)))) \overset{\tau}{\Rightarrow} Q''.$$

Now we have two possibilities:

- $Q_1 \overset{\tau}{\Rightarrow} Q'$ and $Q'' \equiv \nu L' (Q' \mid \bar{a}a' \mid R(n-1,L))$. Then $Q \overset{\tau}{\Rightarrow} Q_1 \overset{\tau}{\Rightarrow} Q'$ and $P' \approx_a^{n-1} Q' \mid \bar{a}a'$ by inductive hypothesis.
- $Q_1 \overset{aa'}{\Rightarrow} Q'$ and $Q'' \equiv \nu L' (Q' \mid R(n-1,L))$. Then $Q \overset{\tau}{\Rightarrow} Q_1 \overset{aa'}{\Rightarrow} Q'$ and $P' \approx_a^{n-1} Q'$ by inductive hypothesis.

$\alpha \equiv \bar{a}a'$ We may suppose $a' \in L$. Then

$$\nu L' (P \mid R(n,L)) \overset{\tau}{\Rightarrow} \nu L' (P \mid \overline{c}_n^{aa'} \oplus a(a'').(\overline{c'}_n^{aa'} \oplus ([a''=a']\overline{d}_n^{a'} \oplus R(n-1,L)))).$$

This has to be matched by

$$\nu L'(Q \mid R(n,L)) \overset{\tau}{\Rightarrow} \nu L'(Q_1 \mid \overline{c}_n^{aa'} \oplus a(a'').(\overline{c'}_n^{aa'} \oplus ([a''=a']\overline{d}_n^{a'} \oplus R(n-1,L)))).$$

We make a further move on the lhs:

$$\nu L'(P \mid \overline{c}_n^{aa'} \oplus a(a'').(\overline{c'}_n^{aa'} \oplus ([a''=a']\overline{d}_n^{a'} \oplus R(n-1,L))))$$
$$\overset{\tau}{\Rightarrow} \nu L'(P' \mid ([a'=a']\overline{d}_n^{a'} \oplus R(n-1,L))).$$

This has to be matched by (we have to lose the $\overline{c'}_n^{aa'}$ commitment while keeping the $\overline{d}_n^{a'}, \overline{b}_{n-1}, \overline{b'}_{n-1}$ commitments):

$$\nu L'(Q_1 \mid \overline{c}_n^{aa'} \oplus a(a'').(\overline{c'}_n^{aa'} \oplus ([a''=a']\overline{d}_n^{a'} \oplus R(n-1,L))))$$
$$\overset{\tau}{\Rightarrow} \nu L'(Q_2 \mid ([a'=a']\overline{d}_n^{a'} \oplus R(n-1,L))).$$

We note $Q_1 \overset{\overline{a}a'}{\Rightarrow} Q_2$. We take a further step on the lhs:

$$\nu L'(P' \mid ([a'=a']\overline{d}_n^{a'} \oplus R(n-1,L))) \overset{\tau}{\Rightarrow} \nu L'(P' \mid R(n-1,L)).$$

This has to be matched by

$$\nu L'(Q_2 \mid ([a'=a']\overline{d}_n^{a'} \oplus R(n-1,L))) \overset{\tau}{\Rightarrow} \nu L'(Q' \mid R(n-1,L)).$$

Now we observe $Q \overset{\tau}{\Rightarrow} Q_1 \overset{\overline{a}a'}{\Rightarrow} Q_2 \overset{\tau}{\Rightarrow} Q'$ and we apply the inductive hypothesis to conclude $P' \approx_a^{n-1} Q'$. □

Remark 41. (1) In the strong case we can simulate matching with synchronization by replacing $[a''=a']\overline{d}_n^{a'}$ with $\nu c(c.\overline{f_n} \mid \overline{a''}c \mid a'(c).(\overline{d}_n^{a'} \mid \overline{c}))$, where $\{f_n \mid n \in \omega\}$ is yet another sequence of names in Ch'. Suppose that in the process P on the lhs $a''=a'$, while in the process Q on the rhs $a' \neq a''$. Then P can perform two consecutive τ reductions, where we indicate with $C[\ ]$ a suitable context:

$$P \equiv C[\nu c(c.\overline{f_n} \mid \overline{a''}c \mid a'(c).(\overline{d}_n^{a'} \mid \overline{c}))] \to C[\nu c(c.\overline{f_n} \mid \overline{d}_n^{a'} \mid \overline{c})] \to C[\overline{f_n} \mid \overline{d}_n^{a'}].$$

To follow the first reduction Q is forced to communicate on a', leaving the message $\overline{a''}c$ idle (since $a' \neq a''$), but then Q cannot follow the second reduction, since it should in one step send out the private name c on a'' and activate the commitment on f_n. Therefore, in the strong case Theorem 40 holds also for a calculus without matching.

(2) In the weak case matching plays an essential role, for instance the terms $\overline{a}b$ and $\overline{a}c$ cannot be separated when put in parallel with the process $!(b(d).\overline{c}d) \mid !(c(d).\overline{b}d)$ (which is an equator in Honda-Yoshida terminology [10]).

(3) It should be noted that our definition of barbed equivalence follows [14]. Honda and Yoshida present a similar result in [10], however they rely on a stronger notion of barbed equivalence where the preservation under parallel composition with outputs is required at each step.

(4) The definition of the tests $R(n,L)$ does *not* involve the guarded sum. This implies that the characterization theorem still holds for an asynchronous calculus without guarded sum.

(5) In the asynchronous calculus *with matching* the various notions of bisimulation do not collapse. For instance consider $P \equiv a(c).\bar{b}e + a(c).\mathbf{0}$ and $Q \equiv P + a(c).[c = d]\bar{b}e$. The processes P and Q are early equivalent but late distinct. Moreover asynchronous bisimulation and barbed equivalence fail to be congruences. If we refine asynchronous bisimulation to an asynchronous congruence (by asking invariance under substitution) and if we refine barbed equivalence to barbed congruence (by considering contexts including the input prefix) then we can show that asynchronous congruence coincides with barbed congruence.

7. Conclusions

Our contributions are summarized in Fig. 4. We leave open the problem of finding an axiomatization of weak asynchronous bisimulation (with or without matching), and the problem of determining the counterpart in the weak case of the characterizations of strong asynchronous bisimulation in terms of $\sim_2$ and $\sim_3$. Axiomatizations of weak bisimulations of process calculi normally use some variant of Milner's tau-laws [11]; these laws are defined using the full guarded summation, that is not available in our asynchronous calculus. As for relations $\sim_2$ and $\sim_3$, we could not extend our characterization results for the strong case to the weak case.

In another direction, it would be worth investigating the applications of Theorem 35 (bisimulation equals ground bisimulation) to automatic verification. For instance, one may wonder if it is possible to speed up current verification techniques by compiling into the asynchronous π-calculus and applying ground bisimulation. To this end, it would be useful to find syntactic conditions under which asynchronous and synchronous bisimulations coincide.

Strong case (without matching)

- $\overset{\bullet}{\sim} \supset \sim_{o\tau} \supset \sim_a = \sim_g = \sim_1 = \sim_2 = \sim_3 = \sim_b$.
- $\sim_a$ is a congruence.
- Axiom which distinguishes asynchronous from synchronous bisimulation:

$$a(b).(\bar{a}b \mid P) + \tau.P = \tau.P \quad \text{if } b \notin fn(P).$$

Weak case

- $\overset{\bullet}{\approx} \supset \approx_{o\tau} \supset \approx_a = \approx_1$.
- Without matching: $\approx_g = \approx_a$ is a congruence, and $\approx_a \subset \approx_b$.
- With matching on image finite processes: $\approx_a = \approx_b$.

Fig. 4. Summary of results.

Appendix A. Proofs

A.1. Proofs of Section 3

Preliminaries to the proof of Theorem 13.

Lemma A.1. *The relations $\sim_a, \sim_1, \sim_2, \sim_3$ are equivalence relations.*

Proof. The only nontrivial property to show is transitivity. The transitivity of $\sim_1$ is immediate. That of $\sim_a$ is proved for the weak case, see Proposition A.7. We prove here the transitivity of $\sim_2$. The transitivity of $\sim_3$ is shown in a similar way.

Transitivity of $\sim_2$. We show that the relation $(\sim_2 \circ \sim_2)$ is a 2-bisimulation. Suppose that $P \sim_2 T \sim_2 Q$. The two interesting cases are:

- $P \xrightarrow{ab} P'$ and T answers by $T \xrightarrow{\tau} T'$ such that for some P'' we have $P' \xrightarrow{\bar{a}b} P''$ and $P'' \sim_2 T'$. Then Q must have a transition $Q \xrightarrow{\tau} Q'$ such that $T' \sim_2 Q'$. Therefore $P''(\sim_2 \circ \sim_2)\, Q'$ as required.
- $P \xrightarrow{ab} P'$ and $T \xrightarrow{ab} T'$ with $P' \sim_2 T'$. If $T \xrightarrow{ab} T'$ is matched by $Q \xrightarrow{ab} Q'$ we have finished. So suppose we are in the case where $Q \xrightarrow{\tau} Q'$ and for some T'' we have $T' \xrightarrow{\bar{a}b} T''$ and $T'' \sim_2 Q'$. Then P' must have a transition $P' \xrightarrow{\bar{a}b} P''$ such that $P'' \sim_2 T''$. Therefore $P''(\sim_2 \circ \sim_2)\, Q'$ and this concludes the proof. □

Let $\equiv_{HT}$ be the structural equivalence defined in p. 9. Clearly $\equiv_{HT}$ is included in all the equivalences $\sim_a, \sim_1, \sim_2, \sim_3$. The following property holds (it should be noted that this property depends on not having outputs on choice points).

Lemma A.2. *If $P \xrightarrow{\bar{a}b} P'$ then $P \equiv_{HT} P' \mid \bar{a}b$.*

Lemma A.3. *The relations $\sim_a$ and $\sim_2$ are preserved by parallel composition with outputs.*

Proof. The proof for $\sim_a$ is given in Lemma A.6 for the weak case. We give here the proof for $\sim_2$. We show that the relation:

$$R = \{(P \mid \bar{a}b,\ Q \mid \bar{a}b) \mid P \sim_2 Q\} \cup \sim_2$$

is a 2-bisimulation up to $\equiv_{HT}$. We check that the bisimulation condition is satisfied by the pairs $(P \mid \bar{a}b,\ Q \mid \bar{a}b)$. We only show the details for the case of *input actions*: here $P \mid \bar{a}b \xrightarrow{cd} P' \mid \bar{a}b$ is inferred from $P \xrightarrow{cd} P'$. Then Q can answer in two ways:

- $Q \xrightarrow{cd} Q'$ and $P' \sim_2 Q'$. In this case we have $Q \mid \bar{a}b \xrightarrow{cd} Q' \mid \bar{a}b$ and $(P' \mid \bar{a}b, Q' \mid \bar{a}b) \in R$.
- $Q \xrightarrow{\tau} Q'$ and there exists P'' such that $P' \xrightarrow{\bar{c}d} P''$ and $P'' \sim_2 Q'$. Then $Q \mid \bar{a}b \xrightarrow{\tau} Q' \mid \bar{a}b$ and $P' \mid \bar{a}b \xrightarrow{\bar{c}d} P'' \mid \bar{a}b$, where $(P'' \mid \bar{a}b,\ Q' \mid \bar{a}b) \in R$. □

Proof of Theorem 13. We show the three equalities: 1. $\sim_a = \sim_1$, 2. $\sim_a = \sim_2$, 3. $\sim_2 = \sim_3$. The proof of 1. is given in Appendix A.3 for the weak case: let us just mention that the direction $\sim_a \subseteq \sim_1$ uses the fact that $\sim_a$ is preserved by parallel composition with outputs, and the direction $\sim_1 \subseteq \sim_a$ uses transitivity of $\sim_1$. The proof of 3. is straightforward. We give here the proof of 2, which relies on Lemmas A.2 and A.3 and uses the transitivity of $\sim_2$.

Proof of 2. It is easy to show that $\sim_a$ is a 2-bisimulation. We now prove that $\sim_2$ is an asynchronous bisimulation. Let $P \sim_2 Q$. Suppose $P \xrightarrow{ab} P'$ and Q answers by a transition $Q \xrightarrow{\tau} Q'$ such that for some P'' we have $P' \xrightarrow{ab} P''$ and $P'' \sim_2 Q'$. By Lemma A.2 $P' \equiv_{HT} P'' \,|\, \bar{a}b$ and thus also $P' \sim_2 P'' \,|\, \bar{a}b$. By Lemma A.3, $P'' \sim_2 Q'$ implies $P'' \,|\, \bar{a}b \sim_2 Q' \,|\, \bar{a}b$. Whence, by transitivity of $\sim_2$, also $P' \sim_2 Q' \,|\, \bar{a}b$. □

A.2. Proofs of Section 5

Proof of Lemma 29. By lexicographic induction on the depth $d(P)$ and on the structure of P. For a given depth, we proceed by structural induction. Axioms S1, S2, S3 and P1, P2, P3 will be used implicitly in the proof, in particular the relation $\equiv$ should be intended as syntactic identity modulo α-renaming, and the axioms above. We shall concentrate on some interesting cases, leaving out the proof of $d(\lceil P \rceil) \leqslant d(P)$.

- Case $n = 0$. If $d(P) = 0$, P is built with the operators $\mathbf{0}, |$ and νa. If we define $\lceil P \rceil = \mathbf{0}$, then we have $P =_{\mathscr{A}} \lceil P \rceil$ by axioms (P1) and (R1).
- Case $n \geqslant 1$. We proceed by induction on the structure of P. We consider the cases of parallel composition and restriction.

(1) $P \equiv R \,|\, S$. By induction there exist normal forms $\lceil R \rceil, \lceil S \rceil$ such that $R =_{\mathscr{A}} \lceil R \rceil$, $S =_{\mathscr{A}} \lceil S \rceil$ and $d(\lceil R \rceil) \leqslant d(R), d(\lceil S \rceil) \leqslant d(S)$. Suppose that

$$\lceil R \rceil \equiv \nu\vec{u}\left(\prod_{i \in I} \bar{a_i} b_i \,|\, R_\Sigma\right), \qquad \lceil S \rceil \equiv \nu\vec{v}\left(\prod_{h \in H} \bar{c_h} d_h \,|\, S_\Sigma\right),$$

where

$$R_\Sigma \equiv \left(\sum_j \tau.R_j + \sum_k a_k(b).R_k\right), \qquad S_\Sigma \equiv \left(\sum_\ell \tau.S_\ell + \sum_m c_m(d).S_m\right)$$

are the guarded parts of $\lceil R \rceil$ and $\lceil S \rceil$. By induction on the depth, all the terms $(R_j \,|\, S_\Sigma)$, $(R_k \,|\, S_\Sigma)$, $(R_\Sigma \,|\, S_\ell)$ and $(R_\Sigma \,|\, S_m)$ have normal forms (induction can be applied because $d(\lceil R \rceil) \leqslant d(R)$, $d(\lceil S \rceil) \leqslant d(S)$). For instance $d(R_j \,|\, S_\Sigma) < (d(R) + d(S)) = d(P)$ follows from $d(R_j) < d(\lceil R \rceil) \leqslant d(R)$ and $d(S_\Sigma) \leqslant d(\lceil S \rceil) \leqslant d(S)$. Let now

$$K' = \{k \in K \mid \exists \ell \in L \cup J \;\; \lceil R_k \,|\, S_\Sigma \rceil =_{SP} (\bar{a_k} b \,|\, \lceil R_\Sigma \,|\, S_\ell \rceil)\},$$

$$M' = \{m \in M \mid \exists j \in L \cup J \;\; \lceil R_\Sigma \,|\, S_m \rceil =_{SP} (\bar{c_m} d \,|\, \lceil R_j \,|\, S_\Sigma \rceil)\}.$$

We can assume that $b \notin fn(\lceil S \rceil)$ and $d \notin fn(\lceil R \rceil)$, and also $\vec{u} \cap \vec{v} = \emptyset$ and $\vec{u} \cap fn(\lceil S \rceil) = \emptyset = \vec{v} \cap fn(\lceil R \rceil)$. We now define

$$\lceil P \rceil \equiv \nu\vec{u}\vec{v}\left(\prod_{i\in I}\overline{a_i}b_i \,\middle|\, \prod_{h\in H}\overline{c_h}d_h \,\middle|\, \left(\sum_{j\in J}\tau.\lceil R_j \mid S_\Sigma \rceil + \sum_{\ell\in L}\tau.\lceil R_\Sigma \mid S_\ell \rceil \right.\right.$$
$$\left.\left. + \sum_{k\in K\setminus K'} a_k(b).\lceil R_k \mid S_\Sigma \rceil + \sum_{m\in M\setminus M'} c_m(d).\lceil R_\Sigma \mid S_m \rceil \right)\right).$$

This is indeed a normal form. In particular, since $\vec{v} \cap fn(\prod_i \overline{a_i}b_i) = \emptyset = \vec{u} \cap fn(\prod_h \overline{c_h}d_h)$, we have

$$Fire\left(\nu\vec{u}\vec{v}\left(\prod_{i\in I}\overline{a_i}b_i \,\middle|\, \prod_{h\in H}\overline{c_h}d_h\right)\right) = Fire\left(\nu\vec{u}\prod_{i\in I}\overline{a_i}b_i\right) \cup Fire\left(\nu\vec{v}\prod_{h\in H}\overline{c_h}d_h\right)$$
$$= I \cup H.$$

Using laws (R2), (EXP) and (IABS), we can easily deduce that

$$P =_{\mathscr{A}} \lceil R \rceil \mid \lceil S \rceil =_{R2} \nu\vec{u}\vec{v}\left(\prod_{i\in I}\overline{a_i}b_i \,\middle|\, \prod_{h\in H}\overline{c_h}d_h \mid R_\Sigma \mid S_\Sigma\right)$$
$$=_{EXP} \nu\vec{u}\vec{v}\left(\prod_{i\in I}\overline{a_i}b_i \,\middle|\, \prod_{h\in H}\overline{c_h}d_h \,\middle|\, \left(\sum_{j\in J}\tau.(R_j \mid S_\Sigma) + \sum_{\ell\in L}\tau.(R_\Sigma \mid S_\ell)\right.\right.$$
$$\left.\left. + \sum_{k\in K} a_k(b).(R_k \mid S_\Sigma) + \sum_{m\in M} c_m(d).(R_\Sigma \mid S_m)\right)\right) =_{IABS} \lceil P \rceil.$$

(2) $P \equiv \nu a Q$. By induction $Q =_{\mathscr{A}} \lceil Q \rceil$ and $d(\lceil Q \rceil) \leqslant d(Q)$. Assume that

$$\lceil Q \rceil \equiv \nu\vec{u}\left(\prod_{i\in I}\overline{a_i}b_i \,\middle|\, \left(\sum_{j\in J}\tau.Q_j + \sum_{k\in K} a_k(b).Q_k\right)\right).$$

We consider separately the two cases where $a \notin fn(\nu\vec{u}\prod_{i\in I}\overline{a_i}b_i)$ and $a \in fn(\nu\vec{u}\prod_{i\in I}\overline{a_i}b_i)$. Note that we can assume $a \notin \vec{u}$, and in this case $a \in fn(\nu\vec{u}\prod_{i\in I}\overline{a_i}b_i) \Leftrightarrow a \in fn(\prod_{i\in I}\overline{a_i}b_i)$.

– If $a \notin fn(\prod_{i\in I}\overline{a_i}b_i)$, we set

$$\lceil P \rceil \equiv \nu\vec{u}\left(\prod_{i\in I}\overline{a_i}b_i \,\middle|\, \left(\sum_{j\in J}\tau.\lceil \nu a Q_j \rceil + \sum_{\substack{a_k \neq a \\ k\in K\setminus K'}} a_k(b).\lceil \nu a Q_k \rceil\right)\right)$$

where the normal forms $\lceil \nu a Q_j \rceil$, $\lceil \nu a Q_k \rceil$ exist by induction on the depth, and $K' = \{k \in K \mid \exists j \in J\ \lceil \nu a Q_k \rceil =_{SP} (\overline{a_k}b \mid \lceil \nu a Q_j \rceil)\}$. This is by definition a normal form. Suppose that both $J \neq \emptyset$ and $K \neq \emptyset$. Then

$$\begin{aligned}
P &=_{\mathscr{A}} \nu a\vec{u} \left(\prod_{i\in I} \overline{a_i}b_i \,\middle|\, \left(\sum_{j\in J} \tau.Q_j + \sum_{k\in K} a_k(b).Q_k \right)\right) \\
&=_{\substack{R3\\R2}} \nu\vec{u} \left(\prod_{i\in I} \overline{a_i}b_i \,\middle|\, \nu a \left(\sum_{j\in J} \tau.Q_j + \sum_{k\in K} a_k(b).Q_k \right)\right) \\
&=_{\substack{R1\\IABS}} \nu\vec{u} \left(\prod_{i\in I} \overline{a_i}b_i \,\middle|\, \left(\sum_{j\in J} \tau.(\nu a\, Q_j) + \sum_{\substack{a_k\neq a\\ k\in K\setminus K'}} a_k(b).(\nu a\, Q_k) \right)\right) \\
&=_{\mathscr{A}} \lceil P \rceil.
\end{aligned}$$

The cases where one or both of J,K are empty are simpler, since we do not need to apply (IABS). We have thus shown that $P =_{\mathscr{A}} \lceil P \rceil$ using laws (R1)–(R3) and (IABS).

– If $a \in fn(\prod_{i\in I} \overline{a_i}b_i)$, define $F = Fire(\nu a\vec{u} \prod_{i\in I} \overline{a_i}b_i)$, $\overline{F} = I\setminus F$, and let $\vec{v},\vec{w}$ be the projections of $a\vec{u}$ on the names that bind, respectively do not bind, some $\overline{a_i}b_i$ such that $i \in F$. Formally, if $\vec{u'} = \{u_\ell \mid \exists i \in F(a_i = u_\ell \vee b_i = u_\ell)\}$ and $\vec{u''} = \vec{u}\setminus\vec{u'}$, we define

$$\vec{v} = \begin{cases} a\vec{u'} & \text{if } \exists i \in F \ (a_i = a \vee b_i = a), \\ \vec{u'} & \text{otherwise.} \end{cases}$$

$$\vec{w} = \begin{cases} a\vec{u''} & \text{if } \nexists i \in F \ (a_i = a \vee b_i = a), \\ \vec{u''} & \text{otherwise.} \end{cases}$$

Supposing $b \notin \vec{v}\vec{w}$, let now

$$\begin{aligned}
\lceil P \rceil \equiv \nu\vec{v} \Bigg(\prod_{i\in F} \overline{a_i}b_i \,\Bigg|\, \Bigg(& \sum_{j\in J} \tau. \left\lceil \nu\vec{w} \left(\prod_{i\in\overline{F}} \overline{a_i}b_i \mid Q_j \right) \right\rceil \\
&+ \sum_{\substack{k\in K\setminus K'\\ a_k\notin\vec{w}}} a_k(b). \left\lceil \nu\vec{w} \left(\prod_{i\in\overline{F}} \overline{a_i}b_i \mid Q_k \right) \right\rceil \\
&+ \sum_{\substack{k\in K\\ a_k=a_h,\, h\in\overline{F}}} \tau. \left\lceil \nu\vec{w} \left(\prod_{\substack{i\in\overline{F}\\ i\neq h}} \overline{a_i}b_i \mid [b_h/b]Q_k \right) \right\rceil \Bigg)\Bigg)
\end{aligned}$$

where all the required normal forms exist by induction on depth and

$$K' = \left\{ k \in K \mid \exists j \in J \ \left\lceil \nu\vec{w}\left(\prod_{i\in\overline{F}} \overline{a_i}b_i \mid Q_k\right)\right\rceil =_{SP} \left(\overline{a_k}b \mid \left\lceil \nu\vec{w}\left(\prod_{i\in\overline{F}} \overline{a_i}b_i \mid Q_j\right)\right\rceil\right) \text{ or} \right.$$

$$\left. \exists k \in K \ \left\lceil \nu\vec{w}\left(\prod_{i\in\overline{F}} \overline{a_i}b_i \mid Q_k\right)\right\rceil =_{SP} \left(\overline{a_k}b \mid \left\lceil \nu\vec{w}\left(\prod_{\substack{i\in\overline{F}\\ i\neq h}} \overline{a_h}b_h \mid [b_r/b]Q_k\right)\right\rceil\right)\right\}.$$

Then, if $\lceil Q\rceil \equiv \nu\vec{u}\,Q'$ and $\lceil P\rceil \equiv \nu\vec{v}\,P'$ we have, by applying (OABS) until all unfirable outputs have been pushed under the guards, and then using (R2), (R1) to push under the guards the restrictions in $\vec{w}$:

$$P =_{\mathscr{A}} \nu a\vec{u}\ Q' =_{R3} \nu\vec{v}\vec{w}\ Q' =_{OABS,R2,R1,IABS} \nu\vec{v}\ P' \equiv \lceil P\rceil. \qquad \Box$$

Preliminaries to the Proof of Lemma 30. Let us look back at the definition of $Fire_n(P)$ for $P \equiv \nu\vec{c}\prod_{i\in I}\overline{a_i}b_i$. Note that the sets $Fire_n(P)$ partition $Fire(P)$. Note also that, since I is finite, there exists a minimal r such that $Fire_{r+1}(P) = \emptyset$. We then have $Fire(P) = \bigcup_{n=0}^{r} Fire_n(P)$. We shall use $\overline{a}\langle b\rangle$ to stand for either $\overline{a}b$ or $\overline{a}(b)$, and $P \xrightarrow{s} P'$ to denote a sequence of transitions $P \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_k} P'$ such that $\alpha_1, \ldots, \alpha_k = s$. The following fact can be easily proved by induction on n.

Remark A.4. Let $P \equiv \nu\vec{u}\prod_{i\in I}\overline{a_i}b_i$ be a normal form such that $I \neq \emptyset$, and define $I_n = Fire_n(P)$ and $N_n = |I_n|$. If $r = min\,\{n \mid Fire_{n+1}(P) = \emptyset\}$, then P has a transition sequence

$$P \equiv P_0 \xrightarrow{s_0} P_1 \cdots P_r \xrightarrow{s_r} P_{r+1} \equiv \mathbf{0}$$

such that for any $j = 1, \ldots, r+1$, $P_j \equiv \nu\vec{u}^j \prod_{i\in I\setminus I_0,\ldots,I_{j-1}} \overline{a_i}b_i$ where $\vec{u}^{j+1} = \vec{u}^j \setminus bn(s_j)$, and, letting $\vec{u}^0 = \vec{u}$, for any $j = 1, \ldots, r$ the sequence $s_j = \overline{a}_1^j\langle b_1^j\rangle, \ldots, \overline{a}_{N_j}^j\langle b_{N_j}^j\rangle$ is a sequentialisation of the outputs in $Fire_j(P)$ such that $\bigcup_{k=1}^{N_j}\{\overline{a_k}^j b_k^j\}$ and for any $k = 1, \ldots, N_j$:

$$\overline{a}_k^j\langle b_k^j\rangle = \begin{cases} \overline{a}_k^j(b_k^j) & \text{if } (b_k^j \in \vec{u}^j \text{ and } \forall \ell \leqslant k\ b_\ell^j \neq b_k^j), \\ \overline{a}_k^j b_k^j & \text{otherwise.} \end{cases}$$

Remark A.5. If $|I| = N$, we can assume w.l.o.g. that $I = \{1, \ldots, N\}$. Then we can build a canonical transition sequence $P \equiv P_0 \xrightarrow{s_0} P_1 \cdots P_r \xrightarrow{s_r} P_{r+1} \equiv \mathbf{0}$ where outputs within the same sequence s_i are sequentialised according to the ordering of I.

Proof of Lemma 30. Based on Remark A.4. Let the canonical transition sequence associated with P be

$$P \xrightarrow{\overline{a_1}\langle b_1\rangle} \nu(\vec{u}\setminus b_1)\left(\prod_{i\in I\setminus\{1\}} \overline{a_i}b_i \mid P_\Sigma\right) \cdots \xrightarrow{\overline{a_N}\langle b_N\rangle} \nu\varepsilon\left(\prod_{i\in\emptyset} \overline{a_i}b_i \mid P_\Sigma\right) \equiv P_\Sigma.$$

Since $P \sim_a Q$, we can find a matching sequence for Q, possibly using α-conversion. Let σ be a renaming on the names of $\vec{v}$ such that $\sigma\vec{v} = \vec{u}$ and the process

$$Q' \equiv \nu\vec{u}\left(\sigma\left(\prod_{h\in H} \overline{c_h}d_h \mid Q_\Sigma\right)\right)$$

has the following matching sequence, deduced without using α-conversion:

$$Q' \xrightarrow{\sigma\overline{c_{i_1}}\langle d_{i_1}\rangle} \nu\sigma(\vec{v}\backslash d_{i_1})\left(\sigma\left(\prod_{h\in H\backslash\{i_1\}} \overline{c_h}d_h \mid Q_\Sigma\right)\right) \cdots \xrightarrow{\sigma\overline{c_{i_N}}\langle d_{i_N}\rangle} \equiv \sigma Q_\Sigma$$

where for any $k = 1,\dots,i_N$, $\sigma\overline{c_{i_k}}\langle d_{i_k}\rangle = \overline{a_k}\langle b_k\rangle$. This shows that $P_\Sigma \sim_a \sigma Q_\Sigma$.

Let now $P_\Pi \equiv \prod_{i\in I} \overline{a_i}b_i$ and $Q_\Pi \equiv \prod_{h\in H} \overline{c_h}d_h$. To obtain $P_\Pi \equiv \sigma Q_\Pi$ it is enough to show that the two multisets of actions in P_Π and σQ_Π are the same. But this is an immediate consequence of the above and of the fact that $Fire(\nu\vec{u}P_\Pi) = I$ and $Fire(\nu\vec{u}\,\sigma Q_\Pi) = H$ (because P and Q are normal forms).

A.3. Proofs of Section 6

Preliminaries to the Proof of Theorem 38

Lemma A.6. *The relation $\approx_a$ is preserved by parallel composition with outputs*:

$$P \approx_a Q \Rightarrow P \mid \overline{a}b \approx_a Q \mid \overline{a}b.$$

Proof. Let $\equiv_P$ be the congruence induced by the commutativity and associativity laws for $\mid$ (laws (P2), (P3) of our axiom table). We show that the relation:

$$R = \{(P \mid \overline{a}b,\ Q \mid \overline{a}b) \mid (P \approx_a Q\} \cup \approx_a$$

is an asynchronous bisimulation up to $\equiv_P$. We check that the bisimulation condition is satisfied by the pairs $(P \mid \overline{a}b,\ Q \mid \overline{a}b)$. We consider the most interesting cases.

- Communication case: $P|\overline{a}b \overset{\tau}{\Rightarrow} P' \mid \mathbf{0}$ is inferred from $P \overset{\tau}{\Rightarrow} P_1 \xrightarrow{ab} P_2 \overset{\tau}{\Rightarrow} P'$ and $\overline{a}b \xrightarrow{\overline{a}b} \mathbf{0}$. There are two possibilities for Q to answer:
  - $Q \overset{\tau}{\Rightarrow} Q_1 \overset{\tau}{\Rightarrow} Q_1' \xrightarrow{ab} Q_2' \overset{\tau}{\Rightarrow} Q_2 \overset{\tau}{\Rightarrow} Q'$, with $P_i \approx_a Q_i$ and $P' \approx_a Q'$. Hence $Q \mid \overline{a}b \overset{\tau}{\Rightarrow} Q_1' \mid \overline{a}b \xrightarrow{\tau} Q_2' \mid \mathbf{0} \overset{\tau}{\Rightarrow} Q' \mid \mathbf{0}$, which is the required move since $P' \mid \mathbf{0} \approx_a Q' \mid \mathbf{0}$.
  - $Q \overset{\tau}{\Rightarrow} Q_2$ with $P_2 \approx_a Q_2 \mid \overline{a}b$. Hence $Q \mid \overline{a}b \overset{\tau}{\Rightarrow} Q' \mid \overline{a}b$ is the matching move, since $P' \mid \mathbf{0} \approx_a Q' \mid \overline{a}b$.
- Case of input action where P communicates with $\overline{a}b$. We only show the details for the case where the communication occurs later than the input (the case where the communication occurs earlier is simpler). Suppose $P \overset{cd}{\Rightarrow} P_1 \overset{ab}{\Rightarrow} P'$ and $P \mid \overline{a}b \overset{cd}{\Rightarrow} P_1 \mid \overline{a}b \overset{\tau}{\Rightarrow} P' \mid \mathbf{0}$. By the case where P moves alone we know that the input transition $P \mid \overline{a}b \overset{cd}{\Rightarrow} P_1 \mid \overline{a}b$ is matched either by $Q \mid \overline{a}b \overset{cd}{\Rightarrow} Q_1 \mid \overline{a}b$ for some Q_1 such that $P_1 \approx_a Q_1$ and $(P_1 \mid \overline{a}b, Q_1 \mid \overline{a}b) \in R$, or by $Q \mid \overline{a}b \overset{\tau}{\Rightarrow} Q_1 \mid \overline{a}b$ for some Q_1 such that $P_1 \approx_a Q_1 \mid \overline{c}d$.

- In the first case, by the communication case we know that $P_1 \mid \bar{a}b \overset{\tau}{\Rightarrow} P' \mid \mathbf{0}$ can be matched either by $Q_1 \mid \bar{a}b \overset{\tau}{\Rightarrow} Q' \mid \mathbf{0}$ such that $P' \mid \mathbf{0} \approx_a Q' \mid \mathbf{0}$, in which case $P \mid \bar{a}b \overset{cd}{\Rightarrow} P' \mid \mathbf{0}$ is matched by $Q \mid \bar{a}b \overset{cd}{\Rightarrow} Q' \mid \mathbf{0}$; or by $Q_1 \mid \bar{a}b \overset{\tau}{\Rightarrow} Q' \mid \bar{a}b$ such that $P' \mid \mathbf{0} \approx_a Q' \mid \bar{a}b$, in which case $P \mid \bar{a}b \overset{cd}{\Rightarrow} P' \mid \mathbf{0}$ is matched by $Q \mid \bar{a}b \overset{cd}{\Rightarrow} Q' \mid \bar{a}b$.
- In the second case, we have $P_1 \approx_a Q_1 \mid \bar{c}d$. Then $Q_1 \mid \bar{c}d$ can simulate the move $P_1 \overset{ab}{\Rightarrow} P'$ in two ways:
  (1) $Q_1 \mid \bar{c}d \overset{ab}{\Rightarrow} Q'$ for some Q' such that $P' \approx_a Q'$. Then there are again two possibilities:
    1a. $Q_1 \mid \bar{c}d \overset{ab}{\Rightarrow} Q'$ because $Q_1 \overset{ab}{\Rightarrow} Q''$ and $Q' = Q'' \mid \bar{c}d$. In this case $P_1 \mid \bar{a}b \overset{\tau}{\Rightarrow} P' \mid \mathbf{0}$ is matched by $Q_1 \mid \bar{a}b \overset{\tau}{\Rightarrow} Q'' \mid \mathbf{0}$, where $P' \approx_a (Q'' \mid \mathbf{0}) \mid \bar{c}d$ because $P' \mid \mathbf{0} \approx_a Q'$. Thus $P \mid \bar{a}b \overset{cd}{\Rightarrow} P' \mid \mathbf{0}$ is matched by $Q \mid \bar{a}b \overset{cd}{\Rightarrow} Q'' \mid \mathbf{0}$.
    1b. The transition $Q_1 \mid \bar{c}d \overset{ab}{\Rightarrow} Q'$ consumes the output $\bar{c}d$. This can be because $Q_1 \overset{ab}{\Rightarrow} Q_2 \overset{cd}{\Rightarrow} Q''$ or because $Q_1 \overset{cd}{\Rightarrow} Q_2 \overset{ab}{\Rightarrow} Q''$. In both cases we have $Q_1 \mid \bar{a}b \overset{cd}{\Rightarrow} Q'' \mid \mathbf{0} = Q'$ and thus $Q \mid \bar{a}b \overset{cd}{\Rightarrow} Q'$ is the matching move.
  (2) $Q_1 \mid \bar{c}d \overset{\tau}{\Rightarrow} Q'$ for some Q' such that $P' \approx_a Q' \mid \bar{a}b$. Again, there are two subcases:
    2a. $Q_1 \mid \bar{c}d \overset{\tau}{\Rightarrow} Q'$ because $Q_1 \overset{\tau}{\Rightarrow} Q''$ and $Q' = Q'' \mid \bar{c}d$. Then $Q_1 \mid \bar{a}b \overset{\tau}{\Rightarrow} Q'' \mid \bar{a}b$ and thus $Q \mid \bar{a}b \overset{\tau}{\Rightarrow} Q'' \mid \bar{a}b$ is the matching move, since $P' \approx_a Q' \mid \bar{a}b \approx_a (Q'' \mid \bar{a}b) \mid \bar{c}d$.
    2b. $Q_1 \mid \bar{c}d \overset{\tau}{\Rightarrow} Q'$ because $Q_1 \overset{cd}{\Rightarrow} Q''$ and $Q' = Q'' \mid \mathbf{0}$. Then $Q_1 \mid \bar{a}b \overset{cd}{\Rightarrow} Q'' \mid \bar{a}b$ and thus $Q \mid \bar{a}b \overset{cd}{\Rightarrow} Q'' \mid \bar{a}b$ is the matching move, since $P' \approx_a Q' \mid \bar{a}b \approx_a Q'' \mid \bar{a}b$. □

Proposition A.7. *The relation $\approx_a$ is an equivalence relation.*

Proof. The only nontrivial property is transitivity. We show that the relation $(\approx_a \circ \approx_a)$ is an asynchronous bisimulation. Suppose that $P \approx_a T \approx_a Q$. The two interesting cases are:

- $P \overset{ab}{\Rightarrow} P'$ and T answers by $T \overset{\tau}{\Rightarrow} T'$ with $P' \approx_a T' \mid \bar{a}b$. Then Q must have a transition $Q \overset{\tau}{\Rightarrow} Q'$ such that $T' \approx_a Q'$. By Lemma A.6 we have then $T' \mid \bar{a}b \approx_a Q' \mid \bar{a}b$ and thus $P' (\approx_a \circ \approx_a) Q' \mid \bar{a}b$ as required.
- $P \overset{ab}{\Rightarrow} P'$ and $T \overset{ab}{\Rightarrow} T'$ with $P' \approx_a T'$. Now if $T \overset{ab}{\Rightarrow} T'$ is matched by $Q \overset{ab}{\Rightarrow} Q'$ we are done. So suppose we are in the case where $Q \overset{\tau}{\Rightarrow} Q'$ and $T' \approx_a Q' \mid \bar{a}b$. Then we have $P' (\approx_a \circ \approx_a) Q' \mid \bar{a}b$ as required. □

Let $\approx_a^1$ be the variant of $\approx_a$ obtained by replacing $\overset{\alpha}{\Rightarrow}$ with $\overset{\alpha}{\rightarrow}$ in the hypothesis of the clauses of $\approx_a$ (that is, replacing $P \overset{\alpha}{\Rightarrow} P'$ with $P \overset{\alpha}{\rightarrow} P'$ in the weak version of Definition 5). We show that it is an equivalent formulation for $\approx_a$. Let us first prove some properties of $\approx_a^1$. It will be used to show that $\approx_a$ coincides with $\approx_1$ and thus with Honda and Tokoro's bisimulation.

Lemma A.8 (Simpler formulation of $\approx_a$). $\approx_a = \approx_a^1$.

Proof. It is clear that $\approx_a \subseteq \approx_a^1$, since $\xrightarrow{\alpha}$ is a particular case of $\overset{\alpha}{\Rightarrow}$. We show now $\approx_a^1 \subseteq \approx_a$. Let $P \approx_a^1 Q$ and suppose $P \overset{\alpha}{\Rightarrow} P'$. We consider the case where α is an input action:

- Let $P = P_0 \xrightarrow{\tau} \cdots P_i \xrightarrow{ab} P_{i+1} \cdots \xrightarrow{\tau} P_n = P'$. Then $Q = Q_0 \overset{\tau}{\Rightarrow} \cdots Q_i$ with $P_k \approx_a^1 Q_k$ for each $k = 0, \dots, i$. Now if $P_i \xrightarrow{ab} P_{i+1}$ is matched by $Q_i \overset{ab}{\Rightarrow} Q_{i+1}$ we proceed as above. So suppose we are in the case where $Q_i \overset{\tau}{\Rightarrow} Q_{i+1}$ and $P_{i+1} \approx_a^1 Q_{i+1} \mid \bar{a}b$. Then there are two ways in which $Q_{i+1} \mid \bar{a}b$ can match the move $P_{i+1} \overset{\tau}{\Rightarrow} P'$:
  - Q_{i+1} moves alone: $Q_{i+1} \mid \bar{a}b \overset{\tau}{\Rightarrow} Q' \mid \bar{a}b$ because $Q_{i+1} \overset{\tau}{\Rightarrow} Q'$. In this case we have $Q \overset{\tau}{\Rightarrow} Q'$ and $P' \approx_a^1 Q' \mid \bar{a}b$ as required.
  - Q_{i+1} consumes the output $\bar{a}b$ in a communication step. In this case the sequence $P_{i+1} \xrightarrow{\tau} \cdots P_j \xrightarrow{\tau} P_{j+1} \cdots \xrightarrow{\tau} P_n = P'$ is matched by $Q_{i+1} \mid \bar{a}b \overset{\tau}{\Rightarrow} \cdots Q_j \mid \bar{a}b \overset{\tau}{\Rightarrow} Q_{j+1} \mid \mathbf{0} \cdots \overset{\tau}{\Rightarrow} Q' \mid \mathbf{0}$ where $Q_j \overset{ab}{\Rightarrow} Q_{j+1}$ and $Q_{j+1} \overset{\tau}{\Rightarrow} Q'$. Then we have $Q \overset{ab}{\Rightarrow} Q'$ and $P' \approx_a^1 Q' \mid \mathbf{0} \approx_a^1 Q'$, which is the required matching transition. □

Lemma A.9 (Simpler formulation of $\approx_1$). $\approx_1 = \approx_1^1$.

Proof. The only difference between the two definitions is in the output and τ clauses, and the proof for this case goes exactly as for $\approx_a$. □

Proof of Theorem 38. We will use the characterizations of $\approx_a$ and $\approx_1$ as $\approx_a^1$ and $\approx_1^1$ respectively. For the sake of simplicity, we keep the notations $\approx_a$ and $\approx_1$. We will use implicitly the fact that $P \approx_1 Q \Leftrightarrow P \approx_1 (Q \mid \mathbf{0})$.

- $\approx_a \subseteq \approx_1$. It is immediate to see that $\approx_a$ is a 1-bisimulation.
- $\approx_1 \subseteq \approx_a$. We show that $\approx_1$ is an asynchronous bisimulation. Again, there is nothing to prove for the output and τ-clauses. As for the input clause, suppose that $P \xrightarrow{ab} P'$. Then $P \mid \bar{a}b \xrightarrow{\tau} P' \mid \mathbf{0}$. Since $P \approx_1 Q$, by definition of $\approx_1$ also $P \mid \bar{a}b \approx_1 Q \mid \bar{a}b$. Therefore there exists Q' such that $Q \mid \bar{a}b \overset{\tau}{\Rightarrow} Q'$ and $P' \mid \mathbf{0} \approx_1 Q'$. Then also $P' \approx_1 Q'$. Now there are three possibilities for the transition $Q \mid \bar{a}b \overset{\tau}{\Rightarrow} Q'$:

- $Q \mid \bar{a}b$ does not move: $Q' = Q \mid \bar{a}b$ and $P' \approx_1 Q \mid \bar{a}b$. In this case we just take $Q \overset{\tau}{\Rightarrow} Q$ and we are in the second case of the input clause of asynchronous bisimulation.
- Q consumes the output $\bar{a}b$: $Q \mid \bar{a}b \overset{\tau}{\Rightarrow} Q'$ because $Q \overset{\tau}{\Rightarrow} Q_1 \xrightarrow{ab} Q_2 \overset{\tau}{\Rightarrow} Q''$ and $Q' = Q'' \mid \mathbf{0}$. Whence $P' \approx_1 Q''$ as required.
- Q moves alone: $Q \mid \bar{a}b \overset{\tau}{\Rightarrow} Q'$ is inferred from $Q \overset{\tau}{\Rightarrow} Q_1 \xrightarrow{\tau} Q_2 \overset{\tau}{\Rightarrow} Q''$ and $Q' = Q'' \mid \bar{a}b$. Then $P' \approx_1 Q'' \mid \bar{a}b$, and we are again in the second case of the input clause of asynchronous bisimulation. □

Complement to the Proof of Theorem 40. We consider the three cases which were left out.

$\alpha \equiv \tau$ Then $\nu L' (P \mid R(n,L)) \overset{\tau}{\Rightarrow} \nu L' (P \mid (\bar{c}_n^{\tau} \oplus R(n-1,L)))$.

To match this reduction up to barbed bisimulation we have to have

$$\nu L' (Q \mid R(n,L)) \stackrel{\tau}{\Rightarrow} \nu L' (Q_1 \mid (\bar{c}_n^{\tau} \oplus R(n-1,L))).$$

We make a further reduction on the lhs:

$$\nu L' (P \mid (\bar{c}_n^{\tau} \oplus R(n-1,L))) \stackrel{\tau}{\Rightarrow} \nu L' (P' \mid R(n-1,L)).$$

Again this has to be matched by (note that we cannot run $R(n,L)$ without losing a commitment $\bar{b}_n$ or $\bar{b}'_n$):

$$\nu L' (Q_1 \mid (\bar{c}_n^{\tau} \oplus R(n-1,L))) \stackrel{\tau}{\Rightarrow} \nu L' (Q' \mid R(n-1,L)).$$

We observe $Q \stackrel{\tau}{\Rightarrow} Q_1 \stackrel{\tau}{\Rightarrow} Q'$. We can conclude by applying the inductive hypothesis.

$\alpha \equiv aa''$ We suppose $a'' \notin L$. Up to an injective substitution we may suppose a'' is the first name in $Ch''\backslash L$. Then

$$\nu L' (P \mid R(n,L)) \stackrel{\tau}{\Rightarrow} \nu L' (P \mid \bar{c}_n^{\bar{a}} \oplus \nu a'' (\bar{a}a'' \mid R(n-1, L \cup \{a''\}))).$$

This has to be matched by:

$$\nu L' (Q \mid R(n,L)) \stackrel{\tau}{\Rightarrow} \nu L' (Q_1 \mid \bar{c}_n^{\bar{a}} \oplus \nu a'' (\bar{a}a'' \mid R(n-1, L \cup \{a''\}))).$$

We make a further reduction on the lhs:

$$\nu L' (P \mid \bar{c}_n^{\bar{a}} \oplus \nu a'' (\bar{a}a'' \mid R(n-1, L \cup \{a''\}))) \stackrel{\tau}{\Rightarrow} \nu L' \cup \{a''\} (P' \mid R(n-1, L \cup \{a''\})).$$

This is matched by

$$\nu L' (Q_1 \mid \bar{c}_n^{\bar{a}} \oplus \nu a'' (\bar{a}a'' \mid R(n-1, L \cup \{a''\}))) \stackrel{\tau}{\Rightarrow} Q''.$$

We have two possibilities:

- $Q_1 \stackrel{\tau}{\Rightarrow} Q'$ and $Q'' \equiv \nu L' \cup \{a''\} (Q' \mid \bar{a}a'' \mid R(n-1, L \cup \{a''\}))$. Then $Q \stackrel{\tau}{\Rightarrow} Q_1 \stackrel{\tau}{\Rightarrow} Q'$ and $P' \approx_a{}^{n-1} Q' \mid \bar{a}a''$ by inductive hypothesis.
- $Q_1 \stackrel{aa''}{\Rightarrow} Q'$ and $Q'' \equiv \nu L' \cup \{a''\} (Q' \mid R(n-1, L \cup \{a''\}))$. Then $Q \stackrel{\tau}{\Rightarrow} Q_1 \stackrel{aa''}{\Rightarrow} Q'$ and $P' \approx_a{}^{n-1} Q'$ by inductive hypothesis.

$\alpha \equiv \bar{a}(a'')$ We may suppose a'' is the first element in $Ch''\backslash L$ (otherwise we rename and use an injective substitution). Then: $\nu L' (P \mid R(n,L)) \stackrel{\tau}{\Rightarrow}$
$\nu L' (P \mid \bar{c}_n^{a} \oplus a(a'').(\overline{c'}_n^{a} \oplus (\oplus \{[a''=a']\bar{d}_n^{a'} \mid a' \in L\} \oplus \bar{e}_n \oplus R(n-1, L \cup \{a''\})))).$
This has to be matched by (we abbreviate $R(n-1, L \cup \{a''\})$ with $R(..)$):

$$\nu L' (Q \mid R(n,L)) \stackrel{\tau}{\Rightarrow}$$

$$\nu L' (Q_1 \mid \bar{c}_n^{a} \oplus a(a'').(\overline{c'}_n^{a} \oplus (\oplus \{[a''=a']\bar{d}_n^{a'} \mid a' \in L\} \oplus \bar{e}_n \oplus R(..)))).$$

We take a further step on the lhs:

$$\nu L' (P \mid \bar{c}_n^{a} \oplus a(a'').(\overline{c'}_n^{a} \oplus (\oplus \{[a''=a']\bar{d}_n^{a'} \mid a' \in L\} \oplus \bar{e}_n \oplus R(..)))) \stackrel{\tau}{\Rightarrow}$$

$$\nu L' \cup \{a''\} (P' \mid \oplus \{[a''=a']\bar{d}_n^{a'} \mid a' \in L\} \oplus \bar{e}_n \oplus R(..)).$$

This has to be matched by (we reason as in the free output case and note that the name sent by Q cannot be in L):

$$\nu L' (Q_1 \mid \bar{c}_n^{\,a} \oplus a(a'').(\overline{c'}_n^{\,a} \oplus (\oplus \{[a''=a']\bar{d}_n^{a'} \mid a' \in L\} \oplus \bar{e}_n \oplus R(..)))) \overset{\tau}{\Rightarrow}$$
$$\nu L' \cup \{a''\} (Q_2 \mid (\oplus \{[a''=a']\bar{d}_n^{a'} \mid a' \in L\} \oplus \bar{e}_n \oplus R(..))).$$

We note $Q_1 \overset{\bar{a}(a'')}{\Rightarrow} Q_2$. We take a last step on the lhs:

$$\nu L' \cup \{a''\} (P' \mid \oplus \{[a''=a']\bar{d}_n^{a'} \mid a' \in L\} \oplus \bar{e}_n \oplus R(..)) \overset{\tau}{\Rightarrow}$$
$$\nu L' \cup \{a''\} (P' \mid R(..)).$$

This has to be matched by

$$\nu L' \cup \{a''\} (Q_2 \mid (\oplus \{[a''=a']\bar{d}_n^{a'} \mid a' \in L\} \oplus \bar{e}_n \oplus R(..))) \overset{\tau}{\Rightarrow}$$
$$\nu L' \cup \{a''\} (Q' \mid R(..)).$$

Conclude by observing that $Q \overset{\tau}{\Rightarrow} Q_1 \overset{\bar{a}(a'')}{\Rightarrow} Q_2 \overset{\tau}{\Rightarrow} Q'$ and $P' \approx_a^{n-1} Q'$ by inductive hypothesis. □

Acknowledgements

We would like to thank David N. Turner for interesting initial discussions and Marco Pistore for helpful comments. We thank also the anonymous referees, whose comments helped us in structuring our results, and in relating our work to contributions previously appeared in the literature. The authors were partially supported by France Télécom, CTI-CNET 95-1B-182 Modélisation de Systèmes Mobiles, and by the HCM Network EXPRESS.

References

[1] G. Agha, Actors: A Model of Concurrent Computation in Distributed Systems, MIT-Press, Cambridge, MA, 1986.
[2] G. Berry, G. Boudol, The chemical abstract machine, Theoret. Comput. Sci. 96 (1992) 217–248.
[3] M. Boreale, D. Sangiorgi, Some congruence properties for π-calculus bisimilarities, Research Report 2870, INRIA, Sophia-Antipolis, 1996.
[4] G. Boudol, Asynchrony and the π-calculus, Research Report 1702, INRIA, Sophia-Antipolis, 1992.
[5] M. Dam, Model checking mobile processes, in: Proc. CONCUR'93, Lecture Notes in Computer Science, vol. 715, 1993, pp. 22–36. Full version in SICS report RR94:1, 1994.
[6] C. Fournet, G. Gonthier, The reflexive CHAM and the join-calculus, Proc. POPL, 1996.
[7] M. Hansen, J. Kleist, H. Hüttel, Bisimulations for asynchronous mobile processes, in: Proc. Tbilisi Symp. on Language, Logic, and Computation, 1995. Research Paper HCRC/RP-72, Human Communication Research Centre, University of Edinburgh.
[8] K. Honda, M. Tokoro, An object calculus for asynchronous communication, Proc. ECOOP 91, Geneve, 1991.
[9] K. Honda, M. Tokoro, On asynchronous communication semantics, in: Object-based Concurrent Computing, Lecture Notes in Computer Science, vol. 612, Springer, Berlin, 1992.

[10] K. Honda, N. Yoshida, On reduction based process semantics, Theoret. Comput. Sci. 151 (1995) 437–486.
[11] R. Milner, Communication and concurrency, Lecture Notes in Computer Science, vol. 92, Springer, Berlin, 1980.
[12] R. Milner, Functions as processes, Math. Struct. Comput. Sci. 2(2) (1992) 119–141.
[13] R. Milner, J. Parrow, D. Walker, A Calculus of Mobile Process, Parts 1–2, Inform. and Comput. 100 (1) (1992) 1–77.
[14] R. Milner, D. Sangiorgi, Barbed bisimulation, in: Proc. ICALP 92, Lecture Notes in Computer Science, vol. 623, Springer, Berlin, 1992.
[15] U. Nestmann, B. Pierce, Decoding choice encodings, in: CONCUR 96, Lecture Notes in Computer Science, vol. 1119, Springer, Pisa, 1996.
[16] B. Pierce, D. Turner, Pict: a programming language based on the π-calculus, Tech. Report, Indiana University, 1997, to appear.
[17] D. Sangiorgi, A theory of bisimulation for the π-calculus, in: Best (Ed.) Proc. CONCUR93, 1993, Lecture Notes in Computer Science, vol. 715, Springer, Berlin.
[18] D. Sangiorgi, Lazy functions and mobile processes, Research Report RR-2515, INRIA, Sophia-Antipolis, 1995. Invited for Festschrift Volume in Honor of Robin Milner's 60th birthday, Cambridge Press, to appear.
[19] G. Tel, Introduction to Distributed Algorithms, Cambridge University Press, Cambridge, 1995.