On Bisimulations for the Asynchronous π-calculus *

Roberto M. Amadio
CNRS, Sophia-Antipolis

Ilaria Castellani
INRIA, Sophia-Antipolis

Davide Sangiorgi
INRIA, Sophia-Antipolis

June 7, 1996

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.

1 Introduction

Process interaction in a distributed system without global clock 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 *distributed algorithms* community the distinction synchronous vs. asynchronous communication is not considered a very important issue. For instance [Tel95], pp 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 seems

*The authors were partially supported by France Télécom, CTI-CNET 95-1B-182 Modélisation de Systèmes Mobiles.

to be quite relevant. 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 [Agh86], while more recent contributions based on the theory of the π-calculus include Pict [PT96] and the join calculus [FG96].

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 [NP96] into the asynchronous π-calculus and the encoding of the asynchronous π-calculus into the join calculus [FG96].

A way to restrict a process calculus to asynchronous communications is to remove output prefixing. In other terms, an asynchronous output $\overline{a}$ followed by a process P is the same as the parallel composition $\overline{a} \mid P$. 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 4 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 [MPW92]. 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, and Boudol [HT91, Bou91], 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.

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 [HT92]) 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 [BS96] for a study of insensitivity to name instantiation for various forms of synchronous bisimulations).

Forms of asynchronous π-calculus have also been studied in [HKH95], but the bisimilarity used is the standard (synchronous) one. Part of our theory, in particular axioms and normal forms, is related to that in [HKH95]. Our formulation of asynchronous bisimulation has been recently used by Nestmann and Pierce [NP96] 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 study an equational theory which characterizes strong asynchronous bisimulation for finite terms. In section 5 we adapt some of the results in section 3 to the *weak* case. In this short version of the paper, most of the proofs are omitted; they can be found in [ACS96].

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 [HT91, Bou91] for the presence of a form of choice. This will be important in the axiomatisation (section 4).

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:

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

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

$$\begin{array}{lll} fn(\tau) = \emptyset & fn(\overline{a}(b)) = \{a\} & fn(\overline{a}b) = fn(ab) = \{a, b\} \\ bn(\tau) = \emptyset & bn(\overline{a}(b)) = \{b\} & bn(\overline{a}b) = bn(ab) = \emptyset \end{array}$$

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.

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

$$\begin{array}{lll} P \stackrel{\tau}{\Rightarrow} P' & \text{iff} & P(\stackrel{\tau}{\rightarrow})^* P' \\ P \stackrel{\alpha}{\Rightarrow} P' & \text{iff} & P \stackrel{\tau}{\Rightarrow} \cdot \stackrel{\alpha}{\rightarrow} \cdot \stackrel{\tau}{\Rightarrow} P' \quad (\text{for } \alpha \neq \tau) \end{array}$$

We write $\rightarrow$ and $\Rightarrow$ as abbreviations for $\stackrel{\tau}{\rightarrow}$ and $\stackrel{\tau}{\Rightarrow}$, respectively. The relations $\rightarrow$ and $\Rightarrow$ are often called *reduction* relations.

$$(cong)\quad \frac{P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q' \equiv Q}{P \xrightarrow{\alpha} Q} \qquad (\tau)\quad \frac{\cdot}{\tau.P \xrightarrow{\tau} P}$$

$$(in)\quad \frac{\cdot}{a(b).P \xrightarrow{ac} [c/b]P} \qquad (out)\quad \frac{\cdot}{\overline{a}b \xrightarrow{\overline{a}b} \mathbf{0}}$$

$$(out_{ex})\quad \frac{P \xrightarrow{\overline{a}b} P' \quad a \neq b}{\nu b\, P \xrightarrow{\overline{a}(b)} P'} \qquad (\nu)\quad \frac{P \xrightarrow{\alpha} P' \quad a \notin n(\alpha)}{\nu a\, P \xrightarrow{\alpha} \nu a\, P'}$$

$$(sync)\quad \frac{P \xrightarrow{\overline{a}b} P' \quad Q \xrightarrow{ab} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad (sync_{ex})\quad \frac{P \xrightarrow{\overline{a}(b)} P' \quad Q \xrightarrow{ab} Q' \quad b \notin fn(Q)}{P \mid Q \xrightarrow{\tau} \nu b\,(P' \mid Q')}$$

$$(comp)\quad \frac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \qquad (sum)\quad \frac{G \xrightarrow{\alpha} P}{G + G' \xrightarrow{\alpha} P}$$

$$(rep)\quad \frac{G \xrightarrow{\alpha} P}{!G \xrightarrow{\alpha} P \mid !G}$$

Figure 1: Labelled transition system with early instantiation

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, $P \downarrow \overline{a}$ if P can make an output action whose subject is a, that is if there exist P', b such that $P \xrightarrow{\overline{a}b} P'$ or $P \xrightarrow{\overline{a}(b)} P'$. The weak commitment is then defined as:*

$$P \Downarrow \overline{a} \text{ if } P \Rightarrow P' \text{ and } P' \downarrow \overline{a}$$

From the definition of reduction and commitment the notion of barbed bisimulation is derived in a canonical way.

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

1. *If $P \downarrow \overline{a}$ then $Q \downarrow \overline{a}$.*
2. *If $P \to P'$ then $Q \to Q'$ and $P'SQ'$.*

Let $\stackrel{\bullet}{\sim}$ 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 $\stackrel{\bullet}{\approx}$ 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 $a(b).\overline{c}b$ and $a(b).\overline{d}b$. Clearly these processes exhibit different behaviours when they are put in parallel with a process $\overline{a}b$. It is then natural to refine barbed bisimulation to an equivalence which is preserved by parallel composition. Following [MS92], we call it barbed equivalence.

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

$$
\begin{array}{lll}
P \sim_b Q & \textit{if} & \forall R\, (P \mid R \stackrel{\bullet}{\sim} Q \mid R) \\
P \approx_b Q & \textit{if} & \forall R\, (P \mid R \stackrel{\bullet}{\approx} Q \mid R)
\end{array}
$$

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 7) 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 $P'S(Q' \mid \overline{a}b)$.*

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.

Remark 1 (comparison with [HT91]) *Definition 5 relies on a standard labelled transition system. Honda and Tokoro [HT91] take a different approach. They modify the labelled transition system by replacing the input rule with the following rule for the $\mathbf{0}$ process (which to some extent allows one to observe the behaviour of a process after an input):*

$$
\frac{\cdot}{\mathbf{0} \xrightarrow{ab} \overline{a}b} \tag{1}
$$

Since rules in [HT91] are applied modulo a structural equivalence $\equiv_{HT}$, and $P \equiv_{HT} P \mid \mathbf{0}$, this implies that any process P can perform any input ab.

We think that rule 1 is not so appealing because: (i) it introduces an infinite branching, (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 does not reflect the computational content of processes.

Honda and Tokoro's bisimulation coincides with ours; the proof is easy, using the characterisation of our asynchronous bisimulation as 1-bisimulation (definition 7).

The following properties are specific to the asynchronous π-calculus:

Lemma 1

1. *If* $P \stackrel{\overline{a}b}{\rightarrow} P'$ *then* $P \sim_a \overline{a}b \mid P'$.

2. *If* $P \stackrel{\overline{a}(b)}{\rightarrow} P'$ *then* $P \sim_a \nu b\,(\overline{a}b \mid P')$.

3. *If* $P \stackrel{\overline{a}b}{\rightarrow} \cdot \stackrel{\alpha}{\rightarrow} P'$ *then* $P \stackrel{\alpha}{\rightarrow} \cdot \stackrel{\overline{a}b}{\rightarrow} P'$.

4. *If* $P \stackrel{\overline{a}(b)}{\rightarrow} \cdot \stackrel{\alpha}{\rightarrow} P'$ *and* $b \notin n(\alpha)$ *then* $P \stackrel{\alpha}{\rightarrow} \cdot \stackrel{\overline{a}(b)}{\rightarrow} P'$.

3 Asynchronous bisimulation, strong case

In this section we study some properties of strong asynchronous bisimulation (definition 5). In section 5 we will discuss how these results can be lifted to the weak case. 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* [1] style [San95], on the other hand, *no* name instantiation is needed in the input clause.

Definition 6 (ground bisimulation) *A relation S is a ground bisimulation if it is an $o\tau$-bisimulation and whenever PSQ, $P \stackrel{ab}{\rightarrow} P'$ and $b \notin fn(P \mid Q)$ the following holds:*

[1] 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 [MPW92], pp 28, with quite a different meaning.

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

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 1 *Strong ground bisimulation is preserved by name substitutions.*

An important corollary of theorem 1 is that bisimulation and ground bisimulation coincide.

Corollary 1 *Strong bisimulation and strong ground bisimulation coincide:* $\sim_a = \sim_g$.

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 2 *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 [San93] bisimulation 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 3 *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 7 (variants of bisimulation) *An i-bisimulation* ($i = 1, 2, 3$) *is an* $o\tau$*-bisimulation* S *such that:*

- (1-bisimulation) PSQ *implies* $(P \mid \overline{a}b)\, S\, (Q \mid \overline{a}b)$, *for all* $\overline{a}b$.

- (2-bisimulation) PSQ *and* $P \stackrel{ab}{\rightarrow} P'$ *implies*
  - *either* $Q \stackrel{ab}{\rightarrow} Q'$ *and* $P'SQ'$
  - *or* $Q \stackrel{\tau}{\rightarrow} Q'$ *and there is* P'' *s.t.* $P' \stackrel{\overline{a}b}{\rightarrow} P''$ *and* $P''SQ'$*.*

- (3-bisimulation) PSQ *and* $P \stackrel{ab}{\rightarrow} P'$ *implies*
  - *either* $Q \stackrel{ab}{\rightarrow} Q'$ *and* $P'SQ'$
  - *or there are* P'', P''' *s.t.* $P' \stackrel{\overline{a}b}{\rightarrow} P''$, $P \stackrel{\tau}{\rightarrow} P'''$ *and* $P''SP'''$*.*

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

Theorem 2 (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. It should be noted that our definition of barbed equivalence follows [MS92]. Honda and Yoshida [HY95] rely on a stronger notion of barbed equivalence, where the preservation under parallel composition with outputs is required at each step.

Theorem 3 *Let* P, Q *be processes. Then* $P \sim_b Q$ *iff* $P \sim_a Q$*.*

4 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:

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

The reader should pause to formally verify this equation according to definition 5. A particular instance of equation 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 figure 2. 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). A related notion of normal form is introduced in [HKH95]. In this work the equational theory captures strong synchronous, rather than *a*synchronous, bisimulation; the axiom system is essentially the same as that in figure 2 but without equation 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)–(P3) in figure 2 (monoid laws for $\mid$). We shall use $\vec{c}$ to denote a sequence of names $c_1, \ldots, c_m$. If $\vec{c} = c_1, \ldots, c_m$, we let $\nu\vec{c}\,P$ stand for $\nu c_1 \ldots \nu c_m\,P$. If $\vec{c} = \varepsilon$ (the empty sequence), we let by convention $\nu\varepsilon\,P \equiv P$. With a slight abuse of notation, we will sometimes use $\vec{c}$ also to represent the set $\{c_1, \ldots, c_m\}$. We define now the set $\mathit{Fire}\,(\nu\vec{c}\prod_{i\in I}\overline{a_i}b_i)$ of indices of firable outputs of $\prod_{i\in I}\overline{a_i}b_i$ when all names in $\vec{c}$ are restricted.

Definition 8 *Let* $P \equiv \nu\vec{c}\prod_{i\in I}\overline{a_i}b_i$*. Then* $\mathit{Fire}\,(P) = \bigcup_n \mathit{Fire}_{n\in\omega}\,(P)$*, where* $\mathit{Fire}_n(P)$ *is the set of indices of outputs that can be fired after exactly n steps, given by:*

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

Example 1 *Let* $P = \nu b\,\nu c\,\Pi_{i\in I}\overline{a_i}b_i$ *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* $n \geq 2$*. 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 figure 2 (commutative monoid laws and idempotence for $+$, and commutative monoid laws for $\mid$).

Definition 9 *A* normal form *is a term defined up to (S1)–(S3) and (P1)–(P3) of the form:*

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

where the sets I*,* J*,* K *are pairwise disjoint, each* P_j*,* P_k *is a normal form, and supposing* $\vec{c} = c_1, \ldots, c_m$*, the following conditions are satisfied:*

1. *(All restricted names are emitted)* $\forall \ell \in \{1, \ldots, m\}\;\exists\, i \in I\;\; b_i = c_\ell$
2. *(All outputs are firable)* $\mathit{Fire}\,(\nu\vec{c}\prod_{i\in I}\overline{a_i}b_i) \;=\; I$
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 $\mathcal{A}$ in figure 2. 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 $=_\mathcal{A}$ denote the congruence induced by these axioms. The proof of normalisation uses nested induction on the depth and on the structure of P.

Definition 10 *The* depth *of a process* P*,* $d(P)$*, is defined inductively by:*

$$
\begin{aligned}
&d(\mathbf{0}) \;=\; 0; \quad d(\overline{a}b) \;=\; 1;\\
&d(a(b).P) \;=\; d(\tau.P) \;=\; 1 \,+\, d(P);\\
&d(P \mid Q) \;= d(P) \,+\, d(Q);\\
&d(\nu a\, P) \;= d(P);\\
&d(G \,+\, F) \;=\; max\,\{\, d(G),\, d(F)\,\}.
\end{aligned}
$$

Remark 2 *$d(P)$ 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 $d(P') \leq d(P)$.*

Lemma 2 (normalisation lemma) *For any process P there exists a normal form:*

$$\lceil P \rceil \;\equiv\; \nu\vec{c}\,\left(\prod_{i\in I} \overline{a_i} b_i \;\mid\; (\sum_{j\in J} \tau.P_j \;+\; \sum_{k\in K} a_k(b).P_k\,) \right)$$

such that $P =_{\mathcal{A}} \lceil P \rceil$ and $d(\lceil P \rceil) \leq d(P)$. 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 3 (separation) *Let P and Q be two normal forms:*

$$P \;\equiv\; \nu\vec{u}\,(\prod_{i\in I}\; \overline{a_i} b_i \;\mid\; P_\Sigma\,) \qquad \text{and} \qquad Q \;\equiv\; \nu\vec{v}\,(\prod_{h\in H} \overline{c_h} d_h \;\mid\; Q_\Sigma\,)$$

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

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

Theorem 4 *On the finite terms, the equivalence $\sim_a$ is the congruence generated by the axioms $\mathcal{A}$.*

5 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:

$$
\begin{aligned}
&!(a(b).\overline{a}b) && = \mathbf{0}\\
&a(b).(\overline{a}b \mid a(b).P) && = a(b).(P)\\
&a(b).(\overline{a}b \mid G) + G && = a(b).G
\end{aligned}
$$

We present the weak versions of theorems 1 and 2. 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 $\approx_a$. In the following $P, Q, R \ldots$ denote processes.

(S1) $G + \mathbf{0} \; = \; G$
(S2) $G + G' \; = \; G' + G$
(S3) $G + (G' + G'') \; = \; (G + G') + G''$
(S4) $G + G \; = \; G$

(P1) $P \mid \mathbf{0} \; = \; P$
(P2) $P \mid Q \; = \; Q \mid P$
(P3) $P \mid (Q \mid R) \; = \; (P \mid Q) \mid R$

(R1) $\nu a\, (\sum_{i \in I} \alpha_i . P_i) = \sum \{ \alpha_i . \nu a\, P_i \mid i \in I,\; a \notin n(\alpha_i) \}$

(R2) $\nu a\, (P \mid Q) = P \mid \nu a\, Q \quad$ if $a \notin fn(P)$

(R3) $\nu a\, \nu b\; P = \nu b\, \nu a\; P$

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

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

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

(OABS) (*Output Absorption*) Let I, J, K be disjoint, $F \;=\; \mathit{Fire}\, (\nu \vec{u} \prod_{i \in I} \overline{a_i} b_i\,)$, $\overline{F} = I \backslash F$, $\vec{v} = \{ u_l \mid \exists i \in F\, (a_i = u_l \text{ or } b_i = u_l) \}$ and $\vec{w} = \vec{u} \backslash \vec{v}$. If $b \notin \vec{w}$, then:

$$\nu \vec{u} \left(\prod_{i \in I} \overline{a_i} b_i \;\mid\; (\sum_{j \in J} \tau . P_j \;+\; \sum_{k \in K} a_k(b) . P_k\,) \right) \;=\; \nu \vec{v} \left(\prod_{i \in F} \overline{a_i} b_i \;\mid\; S \right) \quad \text{where:}$$

$$S \;=\; \sum_{j \in J} \tau . \nu \vec{w}\, (\prod_{i \in \overline{F}} \overline{a_i} b_i \mid P_j) \;+\; \sum_{\substack{k \in K \\ a_k \notin \vec{w}}} a_k(b) . \nu \vec{w}\, (\prod_{i \in \overline{F}} \overline{a_i} b_i \mid P_k) \;+\; \sum_{\substack{k \in K \\ a_k = a_h, h \in \overline{F}}} \tau . \nu \vec{w}\, (\prod_{\substack{i \in \overline{F} \\ i \neq h}} \overline{a_i} b_i \mid [b_h/b] P_k)$$

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

Figure 2: Axioms $\mathcal{A}$

Lemma 4 *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 5 *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* $\alpha' \neq \tau$ *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 there is* P', $P \xrightarrow{\tau} P'$ *and* $\sigma P' \equiv P''$.

   (b) *or* $P \xrightarrow{\overline{a}b} \cdot \xrightarrow{dc} P'$ *and* $[b/c]\sigma P' \sim_a P''$ *(c fresh).*

   (c) *or* $P \xrightarrow{\overline{a}(b)} \cdot \xrightarrow{dc} P'$ *and* $\nu b\,([b/c]\sigma P') \sim_a P''$ *(c fresh).*

We are now ready to prove the crucial lemma.

Lemma 6 *If* $P \approx_g Q$ *then* $\sigma P \approx_a \sigma Q$.

Proof. One shows 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 $\sigma Q \stackrel{\alpha}{\Rightarrow} Q'$ 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

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

$\square$

Theorem 5 *Weak ground bisimulation and weak bisimulation coincide and they are preserved by substitution.*

PROOF. From lemma 4(2) and lemma 6 applied with the identity substitution we know that $P \approx_g Q$ iff $P \approx_a Q$. From lemma 6 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 variants of the weak bisimulation coincide with the early bisimulation studied here.

Corollary 4 *If $P \approx_a Q$ then $a(b).P \approx_a a(b).Q$.*

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

Definition 11 *Let S be a weak $o\tau$-bisimulation. We say that S is a* weak 1-bisimulation *if PSQ implies $(P \mid \overline{a}b)\, S\, (Q \mid \overline{a}b)$. We denote with $\approx_1$ the largest weak 1-bisimulation.*

Theorem 6 (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 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 concentrate on the weak case first; the strong case is easier.

Proposition 1 *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 4 (we cannot use directly this lemma because we have extended the calculus with matching). The proof of (3) follows by:

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

□

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. [Dam93]) are image finite modulo the equation $\nu a\, P = P$ for $a \notin fn(P)$.

Theorem 7 *If P and Q are image finite processes, and $P \approx_b Q$ then $P \approx_a Q$.*

PROOF. Let $\mathcal{F}$ be the monotone operator over $\mathcal{P}(Pr \times Pr)$ associated with the definition of asynchronous bisimulation. Suppose $\approx_a^0 = Pr \times Pr$, $\approx_a^{k+1} = \mathcal{F}(\approx_a^k)$, and $\approx_a^\omega = \bigcap_{k<\omega} \approx_a^k$. It is well-known that on an image finite lts the operator $\mathcal{F}$ preserves co-directed sets. In particular, $\mathcal{F}(\approx_a^\omega) = \approx_a^\omega$. 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:

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

Full definitions of the tests $R(n, L)$ can be found in [ACS96] □

Remark 3 *(1) In the proof for the strong case one can achieve the effect of matching with synchronization. Therefore theorem 7 holds also for a calculus without matching. 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 $!(b(d).\overline{c}d) \, |!(c(d).\overline{b}d)$ (which is an equalizer in Honda-Tokoro terminology).*

(2) 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.*

(3) In the asynchronous calculus with matching *the various notions of bisimulation do not collapse. For instance consider $P \equiv a(c).\overline{b}e + a(c).\mathbf{0}$ and $Q \equiv P + a(c).[c = d]\overline{b}e$. 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.*

Strong case (without matching):

- $\stackrel{\bullet}{\sim} \supset \sim_{o\tau} \supset \sim_a = \sim_g = \sim_1 = \sim_2 = \sim_3 = \sim_b$.
- $\sim_a$ is a congruence.
- Axiom which distinguishes asynchronous from synchronous bisimulation:
  $a(b).(\overline{a}b \mid P) + \tau.P \;=\; \tau.P, \quad \text{if } b \notin fn(P)$.

Weak case:

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

Figure 3: Summary of results

6 Conclusion

Our contributions are summarized in figure 3. 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 characterisations of strong asynchronous bisimulation in terms of $\sim_2$ and $\sim_3$. In another direction, it would be worth investigating the applications of theorem 5 (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. For this, it would be useful to find syntactic conditions under which asynchronous and synchronous bisimulations coincide.

Acknowledgements

We would like to thank David N. Turner for interesting initial discussions and Marco Pistore and the anonymous referees for helpful comments.

References

[ACS96] R. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous π-calculus. Research Report To appear, INRIA, Sophia-Antipolis, 1996. Available at `http://wwwi3s.unice.fr/~amadio/`.

[Agh86] G. Agha. *Actors: a model of concurrent computation in distributed systems*. MIT-Press, 1986.

[Bou91] G. Boudol. Asynchrony and the π-calculus. Research Report 1702, INRIA, Sophia-Antipolis, 1991.

[BS96] M. Boreale and D. Sangiorgi. Some congruence properties for π-calculus bisimilarities. Research Report 2870, INRIA, Sophia-Antipolis, 1996.

[Dam93] M. Dam. Model checking mobile processes. In *Proc. CONCUR'93,* Lecture Notes in Computer Science, 715:22–36, 1993. Full version in SICS report RR94:1, 1994.

[FG96] C. Fournet and G. Gonthier. The reflexive CHAM and the join-calculus. *Proc. ACM-POPL*, 1996.

[HKH95] M. Hansen, J. Kleist, and H. Hüttel. Bisimulations for asynchronous mobile processes. In *Proceedings of the Tbilisi Symposium on Language, Logic, and Computation*, 1995. Research paper HCRC/RP-72, Human Communication Research Centre, University of Edinburgh.

[HT91] K. Honda and M. Tokoro. An object calculus for asynchronous communication. *Proc. ECOOP 91, Geneve*, 1991.

[HT92] K. Honda and M. Tokoro. On asynchronous communication semantics. *Object-based concurrent computing, SLNCS 612*, 1992.

[HY95] K. Honda and N. Yoshida. On reduction based process semantics. *Theoretical Computer Science*, 151:437–486, 1995.

[MPW92] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Process, Parts 1-2. *Information and Computation*, 100(1):1–77, 1992.

[MS92] R. Milner and D. Sangiorgi. Barbed bisimulation. In *Proc. ICALP 92, SLNCS 623*, 1992.

[NP96] U. Nestmann and B. Pierce. Decoding choice encodings. In *CONCUR 96, SLNCS to appear*, Pisa, 1996.

[PT96] B. Pierce and D. Turner. Pict: a programming language based on the π-calculus. U. Cambridge, 1996.

[San93] D. Sangiorgi. A theory of bisimulation for the π-calculus. in *Proc. CONCUR'93* Lecture Notes in Computer Science, 715:127–142, 1993.

[San95] D. Sangiorgi. Lazy functions and mobile processes. Research Report RR-2515, INRIA, Sophia-Antipolis, 1995. Available via anonymous ftp from `cma.cma.fr` as `pub/papers/davide/RR-2515.ps`.

[Tel95] G. Tel. *Introduction to distributed algorithms.* Cambridge University Press, 1995.