Concurrency Can't Be Observed, Asynchronously *

Paolo Baldan[1], Filippo Bonchi[2],
Fabio Gadducci[3], and Giacoma Valentina Monreale[3]

[1] Dipartimento di Matematica Pura e Applicata, Università Padova
[2] INRIA Saclay and LIX, École Polytechnique
[3] Dipartimento di Informatica, Università di Pisa

Abstract. The paper is devoted to an analysis of the concurrent features of asynchronous systems. A preliminary step is represented by the introduction of a non-interleaving extension of barbed equivalence. This notion is then exploited in order to prove that *concurrency cannot be observed* through asynchronous interactions, i.e., that the interleaving and concurrent versions of a suitable asynchronous weak equivalence actually coincide. The theory is validated on two case studies, related to nominal calculi (π-calculus) and visual specification formalisms (Petri nets).

1 Introduction

Since the introduction of process calculi, one of the richest sources of foundational investigations stemmed from the analysis of behavioural equivalences. The rationale is that in any formalism, specifications which are syntactically different may intuitively denote the same system, and it is then pivotal to be able to equate different specifications at the right level of abstraction.

By now classical, one of the most influential synthesis on the issue is offered by the taxonomy proposed in the so-called *linear time/branching time spectrum* [20]. Since then, a major dichotomy among equivalences was established between *interleaving* and *truly concurrent* semantics, according to the possibility of capturing the parallel composition of two systems by means of a non-deterministic selection. Concretely, adopting a CCS-like syntax, the system represented by the specification $a \mid b$ either coincides with (interleaving) or differs from (truly concurrent) the system represented by $a.b + b.a$.

Behavioural equivalences for process calculi often rely on *labelled transitions*: each evolution step of a system is tagged by some information aimed at capturing the possible interactions of the system with the environment. Nowadays, though, the tendency is to adopt operational semantics based on *unlabelled transitions*. This is due to the intricacies of the intended behaviour of a system, especially in the presence of topological or transactional features (see, e.g., foundational calculi such as Mobile Ambients [14] or Join [18]).

* Supported by the MIUR project `SisteR` and the University of Padova project `AVIAMO`.

This paradigmatic shift stimulated the adoption of *barbed congruence* [33], a behavioural equivalence based on a family of predicates over the states of a system, called *barbs*, intended to capture the ability of a system of performing an *interaction* with the environment. For instance, in the calculus of Mobile Ambients [14], barb n verifies the occurrence of an ambient named n at top level [30]; in CCS [31], a process satisfies barb a if it may input on channel a [33].

Assuming that systems interact with a form of synchronous communication, barbs can be explained by a scenario where a system is just a black box with several buttons, one for each possible interaction with the environment. An observer can push a button only if the system is able to perform the corresponding interaction. In this scenario, barbs check if buttons can be pushed. Similarly, an asynchronous system is a black box equipped with several bags (unordered buffers) that are used to exchange messages with the environment. At any time the observer can insert a message in a bag or remove one, whenever present. In this case, barbs check the presence of messages inside bags. Moreover, in order to properly capture the scenario outlined above, for an observer internal steps should not be visible: we thus focus on weak equivalences.

So far, barbed congruences included no concurrent feature, abstractly characterized as the possibility of performing *simultaneously* more than one single interaction. However, in the synchronous scenario, systems $a.b + b.a$ and $a \mid b$ could be distinguished by an observer able to push two buttons at the same time, since only $a \mid b$ allows for the simultaneous pressing of buttons a and b.

The situations is less clearly-cut for asynchronous systems. Indeed, one of the assumptions of this communication style is that message sending is non-blocking: a system may send a message with no agreement with the receiver, and then continue its execution. Hence, an observer interacting with a system by message exchanges cannot know if or when a message has been received and thus message reception is deemed unobservable. And since message sending is non-blocking, a system which may emit a sequence of messages can also hold them, proceed with internal computation and make them available at once at a later time. So, the simultaneous observation of many sendings seems to add no discriminating power to the observer. Concretely, systems $a.b + b.a$ and $a \mid b$ should be equated in an asynchronous setting, even if observing concurrent barbs.

Moving from this intuition, we propose a formal framework where the slogan *concurrency can't be observed, asynchronously* is formalised. We work in a setting where we only assume the availability of an operator for parallel composition, used for defining the notions of *concurrent barb* and concurrent barbed congruence: a system exhibits a concurrent barb $a_1 \otimes a_2$ if it is decomposable into two components exhibiting barbs a_1 and a_2, respectively. We then identify a set of axioms which are intended to capture essential features of asynchronous systems in a barbed setting, showing that for any formalism satisfying them barbed congruence and its concurrent variant coincide. The appropriateness of the axioms is checked by proving that they are satisfied by concrete asynchronous formalisms, like the asynchronous π-calculus [25, 9] and open Petri nets [27], as well as by the (output-buffered) asynchronous systems as characterised in [38].

Synopsis. Section 2 introduces our framework: the notion of concurrent barb, the corresponding behavioural equivalence and Theorem 1, stating the unobservability of concurrency through asynchronous interactions. Sections 3 and 4 show how our theory captures asynchronous π-calculus and open Petri nets, respectively. In the latter case the new concurrent equivalence is shown to coincide with standard step semantics. Section 5 proves that systems deemed as (output-buffered) asynchronous in [38] fall into our theory. Section 6 draws some conclusions, discusses related works and outlines directions for further research.

2 A Theory of Concurrent Barbs and Asynchrony

This section introduces a notion of equivalence based on *concurrent* barbs. It is then argued that, for a reasonable notion of asynchronous system, the possibility of observing concurrent barbs does not add any discriminating power.

2.1 Transition Systems and Barbs

Let $\mathcal{P}$ be a set of *systems* (ranged over by p, q ...) and $\rightarrow \subseteq \mathcal{P} \times \mathcal{P}$ a *transition relation*: we write $p \rightarrow q$ for $\langle p, q \rangle \in \rightarrow$, and we denote by $\rightarrow^*$ the reflexive and transitive closure of $\rightarrow$.

A *barb* is a predicate over the set $\mathcal{P}$ representing a minimal observation on any system. The set of barbs, ranged over by a, b, x, y ..., is denoted $\mathcal{B}$ and we write $p{\downarrow_a}$ if the system p *satisfies* the barb a. For each barb $a \in \mathcal{B}$, we say that p *weakly satisfy* a, written $p{\Downarrow_a}$, if $p \rightarrow^* p'$ and $p'{\downarrow_a}$. Moreover, we write $p\Box{\downarrow_a}$ if $p'{\downarrow_a}$ holds $\forall p'$ such that $p \rightarrow^* p'$. The weak version $p\Box{\Downarrow_a}$ is defined analogously.

We finally assume to have a commutative and associative *parallel composition* operator on systems $| : \mathcal{P} \times \mathcal{P} \rightarrow \mathcal{P}$, satisfying the axioms below

$$\text{(P1)} \quad \frac{p \rightarrow p'}{p|q \rightarrow p'|q} \qquad\qquad \text{(P2)} \quad \frac{p{\downarrow_a}}{p|r{\downarrow_a}}$$

In other terms, the parallel operator must preserve the barbs and the transition relation: the requirement concerning its associativity and commutativity would not be essential for our theory, but it simplifies the presentation.

With these ingredients we can define a behavioural equivalence which equates two systems if these cannot be distinguished by an observer that can add components in parallel and observe the barbs which are exposed. In the paper we focus only on weak equivalences, hence the qualification "weak" is omitted.

Definition 1 (saturated barbed bisimilarity). *A symmetric relation $R \subseteq \mathcal{P} \times \mathcal{P}$ is a saturated barbed bisimulation if whenever pRq then $\forall r \in \mathcal{P}$*

- *$\forall a \in \mathcal{B}$, if $p|r{\Downarrow_a}$ then $q|r{\Downarrow_a}$*
- *if $p|r \rightarrow^* p'$ then $q|r \rightarrow^* q'$ and $p'Rq'$*

We say that p and q are saturated barbed bisimilar (written $p \sim q$) if there exists a saturated barbed bisimulation relating them.

SYNCHRONOUS	ASYNCHRONOUS
$p \ ::= m, p_1 \mid p_2, \mathbf{0}$	$p \ ::= m, p_1 \mid p_2, \mathbf{0}, \bar{a}$
$m ::= \tau.p, a.p, \bar{a}.p, m_1 + m_2$	$m ::= \tau.p, a.p, m_1 + m_2$
(SYN) $a.p + m \mid \bar{a}.q + n \to p \mid q$	(ASYN) $a.p + m \mid \bar{a} \to p$

(TAU) $\tau.p + m \to p$ (PAR) $\dfrac{p \to q}{p \mid r \to q \mid r}$

$p \mid q \equiv q \mid p \quad p \mid (q \mid r) \equiv (p \mid q) \mid r \quad p \mid \mathbf{0} \equiv p$
$m + n \equiv n + m \quad m + (n + o) \equiv (m + n) + o$

Fig. 1. The syntax and the reduction semantics of SCCS and ACCS.

Note that $\sim$ is, by definition, a congruence with respect to the parallel composition operator[4]. It differs from *barbed congruence* [33] since in the latter the observer is allowed to add a parallel component only at the beginning of the computation and not at any step. Hence, in general, barbed congruence is coarser than saturated barbed bisimilarity, although in many cases the two definitions coincide (as e.g. in the asynchronous π-calculus [19]).

As a running example for illustrating our theory we use a finite fragment of CCS [31] and its asynchronous counterpart, with the reduction semantics in [32], but our considerations would trivially extend to the full calculus. A set of *names* $\mathcal{N}$ is fixed (ranged over by a, b ...) with $\tau \notin \mathcal{N}$. The syntax of synchronous CCS (SCCS) processes is defined by the grammar on the left of Figure 1, while asynchronous CCS (ACCS) processes are defined by the grammar on the right. In both cases processes are considered up to structural congruence $\equiv$. The transition relation $\to$ for SCCS is defined by rules SYN, TAU, and PAR. For ACCS, rule SYN is replaced by ASYN: the occurrence of an unguarded $\bar{a}$ indicates a message that is available on some communication media named a. The message disappears whenever it is received. Note that output prefixes $\bar{a}.p$ are absent in ACCS.

The definition of the "right" notion of barb is not a trivial task. For SCCS both input and output barbs are considered (see e.g. [33]). Intuitively, a process has an input (output) barb on a if it is ready to perform an input (output) on a. Formally, if $\alpha = a$ or $\alpha = \bar{a}$, then $p{\downarrow_\alpha}$ when $p \equiv \alpha.p_1 + m \mid p_2$ for processes p_1, p_2, m. Following [1], for ACCS only output barbs are considered, defined by $p{\downarrow_{\bar{a}}}$ when $p \equiv \bar{a} \mid p_1$ for a process p_1. The idea is that, since message sending is non-blocking, an external observer can just send messages without knowing if they will be received or not. Hence inputs are deemed unobservable.

Several works (e.g. [36, 26, 7]) have proposed abstract criteria for defining "good" barbs independently from the formalism at hand. Here, inspired by [36], we propose to formalise the intuition that barbs should capture the possibility of exhibiting an observable behaviour by introducing a notion of *test*.

[4] Requiring $\sim$ to be closed under all unary contexts, instead of just $-|r$ (see [26, 30]), would not substantially change our theory, yet make its presentation more complex.

Definition 2 (barbs witnessed by a test). *A* test *is a family t of systems indexed by barbs, i.e., $t = \{t_x \mid x \in \mathcal{B}\}$. Given a barb $a \in \mathcal{B}$ and a system $p \in \mathcal{P}$, an element $t_x \in t$ is called a* concrete test for a on p *if whenever $p \to^* p'$*

$$p'{\downarrow_a} \qquad \textit{iff} \qquad p'|t_x \to p'' \textit{ and } p''\Box{\downarrow_x}.$$

A barb a is witnessed *by a test t if for all systems $p, q \in \mathcal{P}$ there exists a barb $x \in \mathcal{B}$ such that $t_x \in t$ is a concrete test for a on both p and q.*

Intuitively, a concrete test for a barb a on a process p is a process t_x capable of exposing a barb x, which is instead never observable in the evolution of p. Process t_x releases a (permanent) barb x only after interacting with a process exposing barb a. Since x can never be generated by p, observing x in the evolution of $p' \mid t_x$, where p' is a reduct of p, witnesses that p' has exposed the barb a.

Note that the notion of witness is defined by considering pairs of processes: this is motivated by the fact that tests witnessing a barb will be used when comparing processes in the bisimulation game.

Hereafter, we assume that any barb is always witnessed by some test.

(B) For any $a \in \mathcal{B}$ there exists a test witnessing a. We denote t^a a chosen test that witnesses barb a.

The assumption above holds for any calculus endowed with reduction semantics and barbs that we are aware of (see e.g. [32, 1, 14, 18]). For instance, in ACCS each output barb $\bar{a}$ is witnessed by the test $t^{\bar{a}} = \{a.\bar{x} \mid x \in \mathcal{N}\}$. Indeed, for all processes p, q, a concrete test for $\bar{a}$ on p and q can be $t^{\bar{a}}_x = a.\bar{x}$, for $x \in \mathcal{N}$ a name that syntactically occurs neither in p nor in q. Note that input barbs cannot be witnessed by any test in ACCS, since there are no output prefixes. In SCCS, instead, for the presence of both input and output prefixes, an input barb a is witnessed by the test $\{\bar{a}.\bar{x} \mid x \in \mathcal{N}\}$.

Axiom (B) is pivotal in Section 2.3: the chosen witness for a barb is needed in the formulation of our axiom of asynchrony (AA), which abstractly characterizes a basic feature of asynchronous systems with reduction semantics and barbs.

2.2 Concurrent Barbs and Non-Interleaving Semantics

Most semantics for interactive systems are *interleaving*, meaning that parallelism is reduced to non-determinism, or, in terms of processes, $a.b + b.a \sim a|b$. Here we propose a non-interleaving semantics based on barbs. For this, we first need a *concurrent transitions relation* on systems $\rightsquigarrow \subseteq \mathcal{P} \times \mathcal{P}$, for which we assume

$$\text{(C)} \quad \to \subseteq \rightsquigarrow \subseteq \to^*$$

and thus $\rightsquigarrow^* = \to^*$. The assumption is quite natural: it just means that (1) each non-concurrent transition is also a concurrent one and (2) each concurrent transition $p \rightsquigarrow q$ is simulated by a sequence of non-concurrent ones $p \to \ldots \to q$.

$$\frac{p \to p'}{p \rightsquigarrow p'} \qquad \frac{p \rightsquigarrow p' \quad q \rightsquigarrow q'}{p \mid q \rightsquigarrow p' \mid q'}$$

Fig. 2. Parametric rules for a concurrent transition relation.

For both SCCS and ACCS the relation $\rightsquigarrow$ can be defined by the rules in Figure 2. Alternative definitions could be given, in order e.g. to avoid several concurrent communications on the same channel. This is irrelevant here as our theory abstracts from the actual definition of $\rightsquigarrow$ and only relies on property (C) above.

As a second ingredient, we introduce concurrent barbs. For a set X, let $X^{\otimes}$ denote the free commutative monoid over X, whose elements are called *multisets*.

Definition 3 (concurrent barbs). *The set of* concurrent barbs $\mathcal{CB}$ *is the free monoid* $\mathcal{B}^{\otimes}$*, ranged over by* A*,* B*,* X*,* $Y \ldots$ *We write* $p{\downarrow}^c_A$ *to mean that* p *satisfies the concurrent barb* $\downarrow^c_A$*. The satisfaction relation is defined by the rules*

$$\frac{p{\downarrow}_a}{p{\downarrow}^c_a} \qquad \frac{p{\downarrow}^c_A \text{ and } q{\downarrow}^c_B}{p|q{\downarrow}^c_{A\otimes B}}$$

Weak concurrent barbs are defined as $p{\Downarrow}^c_A$ *if* $p \rightsquigarrow^* p'$ *and* $p'{\downarrow}^c_A$*.*

A more abstract theory could be defined relying on general, non necessarily free monoids of barbs. We defer this proposal to the full version of the paper.

Definition 4 (concurrent saturated barbed bisimilarity). Concurrent saturated barbed bisimilarity*, denoted by* $\sim^c$*, is defined by replacing* $\to$ *with* $\rightsquigarrow$ *and* $\Downarrow_a$ *with* $\Downarrow^c_A$ *in Definition 1.*

Note that for general, possibly synchronous languages, the concurrent equivalence can distinguish processes that are identical in the interleaving semantics. For example, in SCCS $a.b+b.a \not\sim^c a \mid b$ since $a.b+b.a$ does not satisfy $\Downarrow^c_{a\otimes b}$, while $a|b$ does. Instead, if we consider ACCS, where only output barbs are available, it is easy to see that the two processes are equivalent with respect to $\sim^c$.

2.3 Concurrency Can't Be Observed, Asynchronously

This section focus on the observability of concurrency through asynchronous interactions, arguing that $\sim^c = \sim$ in formalisms with asynchronous communication.

As a first step we require that assumption (B) actually holds for *concurrent* barbs, a property denoted as (CB). Formally, the witness property is defined as in Definition 2 by replacing $\mathcal{B}$ with $\mathcal{CB}$, $\to$ with $\rightsquigarrow$ and $\downarrow_a$ with $\downarrow^c_A$.

A further assumption is now needed, relating concrete tests for concurrent barbs and reduction sequences. Since it is intended to capture an essential feature of asynchronous communication, it is referred to as the *Axiom of Asynchrony*

(AA) Let A be a concurrent barb, p a system, t^A_X a concrete test for A on p with $X = \bigotimes_{i=1}^n x_i$. If $p|t^A_X \to^* p_1{\downarrow}_{x_1} \to^* \ldots \to^* p_n{\downarrow}_{x_n}$ then $p{\Downarrow}^c_A$.

$p ::= \overline{a}b,\ p_1 \mid p_2,\ (\nu a)p,\ !m,\ m$	$m ::= \mathbf{0},\ \alpha.p,\ m_1 + m_2$	$\alpha ::= a(b),\ \tau$
$p \mid q \equiv q \mid p$	$(p \mid q) \mid r \equiv p \mid (q \mid r)$	$p \mid \mathbf{0} \equiv p$
$m + n \equiv n + m$	$(m + n) + o \equiv m + (n + o)$	$m + \mathbf{0} \equiv m$
$(\nu a)(\nu b)p \equiv (\nu b)(\nu a)p$	$(\nu a)(p \mid q) \equiv p \mid (\nu a)q$ if $a \notin fn(p)$	$(\nu a)\mathbf{0} \equiv \mathbf{0}$
$(\nu a)p \equiv (\nu b)(p\{^b/_a\})$ if $b \notin fn(p)$	$a(b).p \equiv a(c).(p\{^c/_b\})$ if $c \notin fn(p)$	$!p \equiv p \mid !p$

$$\overline{a}b \mid (a(c).p + m) \to p\{^b/_c\} \qquad \tau.p + m \to p \qquad \frac{p \to q}{(\nu a)p \to (\nu a)q} \qquad \frac{p \to q}{p \mid r \to q \mid r}$$

Fig. 3. Syntax, structural congruence and reduction relation of the asynchronous π.

Informally, the axiom can be explained as follows. We can think that A is a multiset of output messages. The fact that t_X^A is a concrete test for A on p and that $p|t_X^A \to^* p_1{\downarrow_{x_1}} \to^* \ldots \to^* p_n{\downarrow_{x_n}}$ means that p can emit the messages in A one after the other. Then the intuition is that, if the system is asynchronous and thus sending is non-blocking, the messages can be also kept internally and made all available concurrently at the end.

As for our running examples, axiom (AA) holds in ACCS, but not in SCCS. In fact, take the SCCS process $p = \bar{a}.\bar{b}$. A concrete test for the concurrent barb $A = \bar{a} \otimes \bar{b}$ could be $t_X^A = a.\bar{x_1} \mid b.\bar{x_2}$ with $X = \bar{x_1} \otimes \bar{x_2}$. Yet, $p|t_X^A \to \bar{b} \mid \bar{x_1} \mid b.\bar{x_2} \to \bar{x_1} \mid \bar{x_2}$ but $p \not\Downarrow_A^c$.

Relying on the assumptions made so far, we can prove the desired theorem.

Theorem 1 (concurrency can't be observed, asynchronously). *For any formalism satisfying axioms (P1), (P2), (CB), (C), and (AA), concurrent saturated barbed bisimilarity and saturated barbed bisimilarity coincide, i.e.,* $\sim = \sim^c$.

3 Asynchronous π-calculus

This section shows that the asynchronous π-calculus fits in the theory of Section 2, and thus saturated barbed congruence (which coincides with barbed congruence [1]) and its concurrent version coincide.

Asynchronous π-calculus has been introduced in [25] as a model of distributed systems interacting via asynchronous message passing. Its syntax is shown in Figure 3: we assume an infinite set $\mathcal{N}$ of *names*, ranged over by $a, b \ldots$, with $\tau \notin \mathcal{N}$, and we let $p, q \ldots$ range over the set $\mathcal{P}_\pi$ of processes. *Free names* of a process p (denoted by $fn(p)$) are defined as usual. Processes are taken up to a *structural congruence*, axiomatised in Figure 3 and denoted by $\equiv$. The *reduction relation*, denoted by $\to$, describes process evolution: it is the least relation $\to \subseteq \mathcal{P}_\pi \times \mathcal{P}_\pi$ closed under $\equiv$ and inductively generated by the axioms and rules in Figure 3.

As for ACCS (Section 2), barbs account only for outputs. So, for an output $\overline{a}$, $p{\downarrow_{\overline{a}}}$ if $p \equiv (\nu a_1) \ldots (\nu a_k)(\overline{a}b|q)$ and $\forall i,\ a \neq a_i$ [1]. Concurrent barbs are multisets of outputs, and they check the presence of several parallel outputs.

A non-interleaving semantics for the calculus is obtained by introducing a concurrent transition relation $\rightsquigarrow$, as defined in Figure 2. Multiple synchronizations over the same channel are thus allowed, as in the semantics proposed in [12, 34]. Different approaches are conceivable, see e.g. [28], yet they could still be accommodated in our theory.

Now, let $\sim_\pi$ denote saturated barbed bisimilarity for the asynchronous π-calculus and let $\sim_\pi^c$ denote the concurrent one. It is worth remarking that $\sim_\pi$ coincides with the standard semantics for the calculus, namely, *asynchronous bisimilarity* [1], as shown in [19]. Then we have the following result.

Corollary 1 (concurrency can't be observed in asynchr. π). $\sim_\pi = \sim_\pi^c$.

This follows from Theorem 1. Indeed, axioms (P1), (P2), and (C) clearly hold. Concerning (CB), a test witnessing the concurrent barb $A = \bigotimes_{i=1}^n \overline{a}_i$ is $t^A = \{t_C^A \mid C = \bigotimes_{i=1}^n \overline{c_i} \wedge t_C^A = a_1(b_1).\overline{c_1}d | \ldots | a_n(b_n).\overline{c_n}d \wedge \forall i.b_i \neq c_i\}$: for processes p, q, we obtain a concrete test t_C^A for A on p and q by taking a C containing only names syntactically occurring neither in p nor q. With the above definition, it is easy to prove that also the Axiom of Asynchrony (AA) holds.

4 Open Petri Nets

Open Petri nets [27, 37, 3] are a reactive extension of ordinary P/T nets, equipped with a distinguished set of *open places* that represent the interfaces through which the environment interacts with a net. This kind of interactions is inherently asynchronous (see e.g. [2]) and thus it represents an ideal testbed.

This section shows that indeed the interleaving and concurrent equivalences defined in the literature (see e.g. [3]) are instances of $\sim$ and $\sim^c$, respectively. Then, since all the axioms of our theory are satisfied, these equivalences coincide.

Definition 5 (open nets). *An* open net *is a tuple* $\hat{N} = (S, T, {}^\bullet(.), (.)^\bullet, O)$ *for* S *a set of places,* T *a set of transitions,* ${}^\bullet(.), (.)^\bullet : T \to S^\otimes$ *functions mapping each transition to its pre- and post-set, and* $O \subseteq S$ *a set of* open places. *A marked (open) net is pair* $N = \langle \hat{N}, m \rangle$ *for* $\hat{N}$ *an open net and* $m \in S^\otimes$ *a marking.*

Examples of marked nets can be found in Figure 4. As usual, circles represent places and rectangles transitions. Arrows from places to transitions represent function ${}^\bullet(.)$, arrows from transitions to places represent $(.)^\bullet$. An open net is enclosed in a box and open places are on the border of such a box.

We assume a fixed infinite set $\mathcal{S}$ of place names. The set of *interactions* (ranged over by i) is $\mathcal{I}_\mathcal{S} = \{s^+, s^- \mid s \in \mathcal{S}\}$. The set of *labels* (ranged over by l) consists in $\{0\} \uplus \mathcal{I}_\mathcal{S}$. The firing (interleaving) semantics of open nets is expressed by the rules on the top of Figure 5, where we write ${}^\bullet t$ and $t^\bullet$ instead of ${}^\bullet(t)$ and $(t)^\bullet$. The rule (TR) is the standard rule of P/T nets (seen as multiset rewriting) modelling internal transitions, which are labelled with 0 for subsequent use. The other two rules model interactions with the environment: at any moment a token can be inserted in (rule (IN)) or removed from (rule (OUT)) an open place.

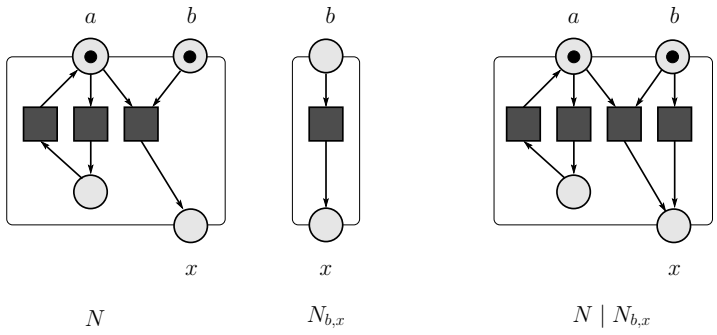

Fig. 4. Marked open nets and their parallel composition.

$$\text{(TR)}\ \frac{m = {}^\bullet t \otimes m' \quad t \in T}{m \xrightarrow{0} t^\bullet \otimes m'} \qquad \text{(IN)}\ \frac{s \in O}{m \xrightarrow{s^+} m \otimes s} \qquad \text{(OUT)}\ \frac{m = m' \otimes s \quad s \in O}{m \xrightarrow{s^-} m'}$$

$$\text{(CFIR)}\ \frac{m \xrightarrow{\ell} m'}{m \overset{\ell}{\rightsquigarrow} m'} \qquad \text{(CSTEP)}\ \frac{m = m_1 \otimes m_2 \quad m_1 \overset{c_1}{\rightsquigarrow} m_1' \quad m_2 \overset{c_2}{\rightsquigarrow} m_2'}{m \overset{c_1 \otimes c_2}{\rightsquigarrow} m_1' \otimes m_2'}$$

Fig. 5. Firing and step semantics for open nets.

Weak transitions are defined as usual, i.e., $\overset{0}{\Rightarrow}$ denotes the reflexive and transitive closure of $\xrightarrow{0}$ and $\overset{i}{\Rightarrow}$ denotes $\overset{0}{\Rightarrow}\xrightarrow{i}\overset{0}{\Rightarrow}$. We write $N \overset{i}{\Rightarrow} N'$ when $N = \langle \hat{N}, m \rangle$, $N' = \langle \hat{N}, m' \rangle$ and $m \overset{i}{\Rightarrow} m'$.

Definition 6 (firing bisimilarity). *A symmetric relation R over marked nets is a* firing bisimulation *if whenever $N_1 R N_2$, if $N_1 \overset{l}{\Rightarrow} N_1'$ then $N_2 \overset{l}{\Rightarrow} N_2'$ and $N_1' R N_2'$. We say that N_1 and N_2 are firing bisimilar (written $N_1 \approx N_2$) if there exists a firing bisimulation R such that $N_1 R N_2$.*

In order to ease the intuition, nets can be thought of as black boxes, where only the interfaces are visible. Two nets are bisimilar if they cannot be distinguished by an observer that may only insert and remove tokens in open places.

Steps of open nets ($\rightsquigarrow$) are defined in Figure 5, bottom. Step labels (ranged over by $c, c_1, c_2 \dots$) are multisets of interactions $\mathcal{I}_N$. By rule (CFIR), each firing is also a step and, in particular, the label 0 is interpreted as the empty multiset. Rule (CSTEP) allows to construct concurrent steps. *Weak transitions* are defined as usual: $\overset{0}{\mapsto}$ denotes the reflexive and transitive closure of $\overset{0}{\rightsquigarrow}$ and $\overset{c}{\mapsto}$ denotes $\overset{0}{\mapsto}\overset{c}{\rightsquigarrow}\overset{0}{\mapsto}$. *Step bisimilarity* ($\approx^c$) is defined by replacing $\Rightarrow$ with $\mapsto$ in Definition 6.

We now show that $\approx$ and $\approx^c$ are instances of $\sim$ and $\sim^c$, respectively. The *parallel composition* $N_1|N_2$ of open nets N_1, N_2 is obtained by gluing them on their open places. More precisely, $N_1|N_2$ is the marked net obtained by taking

the disjoint union of the nets, merging open places with the same name and summing the markings. An example of composition is shown in Figure 4.

Transitions $\xrightarrow{0}$ of marked nets correspond to transitions $\rightarrow$ in the theory of Section 2, and $\overset{0}{\rightsquigarrow}$ corresponds to $\rightsquigarrow$. *Barbs* check the presence of tokens in open places. Formally, if we write $m \subseteq n$ for $m, n \in S^{\otimes}$ whenever $m = n \otimes n'$ for some $n' \in S^{\otimes}$, the marked net $N = \langle \hat{N}, m \rangle$ satisfies the barb b, denoted $N{\downarrow_b}$, if $b \in O$ (i.e., b is an open place of $\hat{N}$) and $b \subseteq m$. *Concurrent barbs* check the presence of multisets of tokens: for $m' \in \mathcal{S}^{\otimes}$, $N{\downarrow^c_{m'}}$ if $m' \in O^{\otimes}$ and $m' \subseteq m$.

With these definitions it is possible to prove that firing (step) bisimilarity coincides with (concurrent) saturated barbed bisimilarity.

Proposition 1. *Let N_1, N_2 be two marked nets with the same set of open places. Then $N_1 \approx N_2$ iff $N_1 \sim N_2$ and $N_1 \approx^c N_2$ iff $N_1 \sim^c N_2$.*

In order to apply Theorem 1, we finally need to prove that all the axioms are satisfied. This is immediate for (P1), (P2), and (C). Instead, concerning (CB), a test witnessing a barb $b \in \mathcal{S}$ is given by $t^b = \{t^b_x \mid x \in \mathcal{S}\}$, where $t^b_x = N_{b,x}$ is the net in Figure 4, middle. For a concurrent barb $B = b_1 \otimes \cdots \otimes b_n \in \mathcal{S}^{\otimes}$ a test is given by $t^B = \{t^B_X \mid X = \bigotimes_{i=1}^n x_i \wedge t^B_X = t^{b_1}_{x_1} | \ldots | t^{b_n}_{x_n}\}$. With this definition of test, also the Axiom of Asynchrony (AA) can be easily shown to hold. Hence, as a corollary of Theorem 1 we get the following result.

Corollary 2 (concurrency can't be observed in open nets). $\approx = \approx^c$.

5 On Selinger's Axiomatization

An axiomatization of different classes of systems with asynchronous communication has been proposed in [38]. Roughly speaking, a system is said to be asynchronous if its observable behaviour is not changed by filtering its input and/or output through a suitable communication medium, which can store messages and release them later on. Different choices of the medium (queues, unordered buffers) are shown to lead to different notions of asynchrony, and suitable sets of axioms are then identified which are shown to precisely capture the various classes of asynchronous systems.

In order to further check the appropriateness of our framework, here we prove that the class of systems characterised as asynchronous in [38] satisfy the requirements in Section 2. More precisey, we focus on so-called *out-buffered asynchrony with feedback* [38, Section 3.2], where output is asynchronous, the order of messages is not preserved and the output of a process can be an input for the process itself (feedback). The corresponding axioms [38, Table 3] are listed in Figure 6. They are given for labelled transition systems, with labels *in* a, *out* a and τ denoting input, output and internal transitions, respectively.

In order to bring the correspondence to a formal level, we must overcome two problems. Firstly, the theory in [38] is developed for a labelled semantics, while we are concerned with barbed reduction semantics, and secondly, the theory in [38] does not consider concurrent transitions, which are pivotal in our setting.

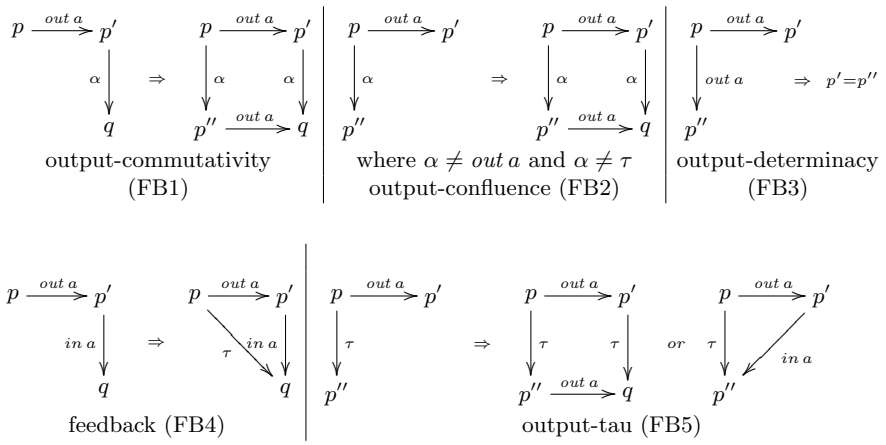

Fig. 6. Axioms for out-buffered agents with feedback.

The first issue is solved by taking as reductions $p \to p'$ the τ-transitions $p \xrightarrow{\tau} p'$ and by defining (output) barbs $p{\downarrow_a}$ if $p \xrightarrow{out\,a}$.

As a parallel operator for out-buffered agents with feedback, we use the parallel composition with interaction defined in [38, Section 3.1] and given by[5]

$$\frac{p \xrightarrow{\alpha} p'}{p|q \xrightarrow{\alpha} p'|q} \qquad \frac{q \xrightarrow{\alpha} q'}{p|q \xrightarrow{\alpha} p|q'} \qquad \frac{p|q \xrightarrow{out\,a} \xrightarrow{\tau}{}^{*} \xrightarrow{in\,a} r}{p|q \xrightarrow{\tau} r}$$

As far as concurrent barbs are concerned, we define $p{\downarrow^c_A}$, where $A = \bigotimes_{i=1}^{n} a_i$ whenever $p \xrightarrow{out\,a_1} \ldots \xrightarrow{out\,a_n}$. This is motivated by the fact that, by axiom (FB1), this implies that the same outputs can be performed by p in *any* order (in particular $p \xrightarrow{a_i}$ for any $i \in \{1, \ldots, n\}$). In words, although the labelled transition system does not provide any information on concurrency, we assume that outputs which can be observed in any order are generated concurrently.

Moreover, $\mathcal{A}(p)$ denotes the *acceptance set of* p defined as $\mathcal{A}(p) = \{a \mid \exists p' : p \to^{*} p' \text{ and } p'{\downarrow_a}\}$, and we stick to systems p such that the set $\mathcal{A}(p)$ is finite.[6]

With the above definitions, it is easy to see that axiom (B) holds.

Lemma 1. *A barb a is witnessed by test $t^a = \{t^a_x \mid t^a_x \xrightarrow{in\,a} \xrightarrow{out\,x}\}$. In particular, for p, q, if $x \notin \mathcal{A}(p) \cup \mathcal{A}(q)$, the system t^a_x is a concrete test for a on p and q.*

Concurrent reductions can now be defined as in Figure 2. With this definition it is not difficult to see that assumptions (P1), (P2), (C) hold, and that (CB) is

[5] Actually, this operator is associative and commutative only up-to isomorphism of the underlying transition space of the system, which is implicitly assumed here.

[6] This requirement is far from restrictive. For instance, it holds in the π-calculus since for all processes p, q such that $p \to^{*} q$ we have $\mathrm{fn}(q) \subseteq \mathrm{fn}(p)$.

an immediate consequence of (B). In fact the test witnessing a concurrent barb $A = \bigotimes_{i=1}^{n} a_i$ can be $t^A = \{t^A_X \mid X = \bigotimes_{i=1}^{n} x_i \wedge t^{a_i}_{x_i} \in t^{a_i} \wedge t^A_X = t^{a_1}_{x_1} | \ldots | t^{a_n}_{x_n}\}$.

With this set up we can finally prove that also the Axiom of Asynchrony (AA) holds for any out-buffered system p with feedback.

Lemma 2. *Let $A = \bigotimes_{i=1}^{n} a_i$ be a concurrent barb, let p be a system satisfying the axioms in Figure 6, and let t^A_X be a concrete test for A on p with $X = \bigotimes_{i=1}^{n} x_i$. If $p|t^A_X \rightarrow^* p_1{\downarrow_{x_1}} \rightarrow^* \ldots \rightarrow^* p_n{\downarrow_{x_n}}$ then $p{\Downarrow^c_A}$.*

6 Conclusions, Related and Future Works

Building on the notion of concurrent barbs, we introduced a non interleaving observational congruence for systems, and we proved that our slogan holds in a rather general fashion: whenever the observer is only able to check the possible interactions of a system with the environment, and the system can interact only through an unordered buffer (corresponding to the *out-buffered systems with feedback* of [38]), then concurrency cannot be observed, i.e., concurrent barbs add no observational power.

As case studies, we considered open Petri nets and the asynchronous π-calculus, showing that they fall in our framework. In particular, for nets we recovered the ordinary firing and step semantics (as defined in [3]); while for the π-calculus the well-known asynchronous bisimilarity [1]. Our result holds for other interesting formalisms as well, such as the Join calculus [18]. Indeed, the latter is an instance of [38] and thus, as proved in Section 5, our theory applies.

The non-interleaving equivalence we introduced intuitively corresponds to *step semantics*. This has been shown for the concrete case of open Petri nets, even if it seems hard to raise the correspondence at an abstract level. Some idea could come from the observation that steps naturally arise from the *theory of reactive systems* [29] when replacing $\rightarrow$ with $\rightsquigarrow$. Since $p \xrightarrow{a} q$ means that $-|\bar{a}$ is the smallest context $c[-]$ such that $c[p] \rightarrow q$, analogously the step $p \xrightarrow{a \otimes b} q$ would mean that $-|\bar{a}|\bar{b}$ is the smallest $c[-]$ such that $c[p] \rightsquigarrow q$. As a side remark, note that one of the compelling arguments against step semantics (i.e., that it is not preserved by *action refinement* [21]) looses its strength in the paradigm of reduction systems and barbed equivalences, since actions (labels) disappear.

As far as *ST-equivalences* [23] are concerned, it seems conceivable to develop an ST-operational semantics in an asynchronous setting, making production and consumption of messages (tokens) not instantaneous (see, e.g., [22] for a net model where token consumption is non-instantaneous and [13] for a similar study on Linda-like languages) and we conjecture that unobservability of concurrency would hold true also in this setting.

Close to our spirit are also *equivalences with localities* [10], that distinguish (interleaving equivalent) processes by observing the locations where interactions occur. We chose of not adopting this kind of equivalence for two main reasons: (1) localities are usually structured as trees, but this does not make much sense either in a calculus featuring joins (e.g. [18]) or in a graphical formalism such

as open Petri nets; (2) equivalence with localities have never been defined for reduction semantics and, more importantly, for asynchronous formalisms.

It can be shown that equivalences with localities are incomparable with ours. Still we conjecture that our slogan "concurrency can't be observed, asynchronously" still holds for equivalences with localities. Indeed, since in the asynchronous case inputs are not observable, also their locations should not be observable. Therefore, only the locations of outputs could be observed, but these are all independent (since outputs have no continuations). A formal study of equivalences with localities for asynchronous systems is left as future work.

Our proposal is quite far from other non-interleaving semantics, such as those proposed in e.g. [16, 17, 21]: these consider *causal properties* of the systems, either by direct inspection of the state structure or by suitably enriching the labels of the transition steps, thus being of a more extensional nature. For these semantics, the fact that the internals of the systems are directly inspected, clearly implies that the unobservability of concurrency will not hold.

It would be interesting to investigate the possibility of extending our results to other classes of languages. This could include asynchronous calculi with bounded capacity channels, where a bounded number of messages can be transmitted simultaneously along the same channel. We would also like to study notions of asynchrony based on buffers which are not just unordered bags, but ordered structures like queues (see e.g. [5, 6, 38, 4]). A preliminary investigation on the calculi $\pi_{\mathfrak{Q}}$ and $\pi_{\mathfrak{S}}$ in [4] (where buffers are, respectively, queues and stacks) seems to suggest that our results on the unobservability of concurrency should extend also to "ordered asynchrony". Intuitively, a key difference would be that in these calculi concurrent barbs should be *sets* of barbs instead of multisets, since these ordered buffers should not allow concurrent operations. Finally, another appealing case study could concern Linda-like languages, where the presence of test-and-check operators might allow an observer to verify not only the presence but also the absence of messages. In the same class would then end up also nets with inhibitor arcs.

The different distinguishing power of concurrent equivalences in the synchronous and asynchronous case could also be inspiring for the development of additional separation results between the two paradigms, along the style of [35]. In more general terms, integrating our framework with the one proposed in [24] seems to represent a promising direction for future investigations.

So far, few papers (such as e.g. [8, 15, 11]) tackled the study of the concurrency features of asynchronous systems. And to the best of our knowledge our result, albeit quite intuitive, has never been shown on any specific formalism, let alone for a general framework as in our paper. Indeed, besides the catchy slogan, we do believe that our work unearthed some inherent features of asynchronous systems that should hopefully shed some further light on the issue. That is, it should represent a further step towards a complete characterisation of the still fuzzy synchronous/asynchronous dichotomy.

Acknowledgments. The authors would like to thank Catuscia Palamidessi for the helpful discussions and the pointers to the literature.

References

1. Amadio, R.M., Castellani, I., Sangiorgi, D.: On bisimulations for the asynchronous π-calculus. In: Proc. of CONCUR'96. LNCS, vol. 1119, pp. 147–162. Springer (1996)
2. Baldan, P., Bonchi, F., Gadducci, F.: Encoding asynchronous interactions using open Petri nets. In: Proc. of CONCUR'09. LNCS, vol. 5710, pp. 99–114. Springer (2009)
3. Baldan, P., Corradini, A., Ehrig, H., Heckel, R.: Compositional semantics for open Petri nets based on deterministic processes. Math. Str. in Comp. Sci. 15(1), 1–35 (2005)
4. Beauxis, R., Palamidessi, C., Valencia, F.D.: On the asynchronous nature of the asynchronous π-calculus. In: Concurrency, Graphs and Models. LNCS, vol. 5065, pp. 473–492. Springer (2008)
5. Bergstra, J.A., Klop, J.W., Tucker, J.V.: Process algebra with asynchronous communication mechanisms. In: Seminar on Concurrency. LNCS, vol. 197, pp. 76–95. Springer (1984)
6. de Boer, F.S., Klop, J.W., Palamidessi, C.: Asynchronous communication in process algebra. In: Proc. of LICS'92. pp. 137–147. IEEE Computer Society (1992)
7. Bonchi, F., Gadducci, F., Monreale, G.V.: On barbs and labels in reactive systems. In: Proc. of SOS'09. EPTCS, vol. 18, pp. 46–61 (2010)
8. Boreale, M., Sangiorgi, D.: Some congruence properties for π-calculus bisimilarities. Theor. Comp. Sci. 198(1-2), 159–176 (1998)
9. Boudol, G.: Asynchrony and the π-calculus. Tech. Rep. 1702, INRIA (1992)
10. Boudol, G., Castellani, I., Hennessy, M., Kiehn, A.: Observing localities. In: Proc. of MFCS'91. LNCS, vol. 520, pp. 93–102 (1991)
11. Bruni, R., Melgratti, H.C., Montanari, U.: Event structure semantics for dynamic graph grammars. ECEASST 2 (2006)
12. Busi, N., Gorrieri, R.: A Petri net semantics for π-calculus. In: Proc. of CONCUR'95. LNCS, vol. 962, pp. 145–159. Springer (1995)
13. Busi, N., Gorrieri, R., Zavattaro, G.: Comparing three semantics for linda-like languages. Theor. Comp. Sci. 240(1), 49–90 (2000)
14. Cardelli, L., Gordon, A.D.: Mobile ambients. Theor. Comp. Sci. 240(1), 177–213 (2000)
15. Crafa, S., Varacca, D., Yoshida, N.: Compositional event structure semantics for the internal π-calculus. In: Proc. of CONCUR'07. LNCS, vol. 4703, pp. 317–332. Springer (2007)
16. Darondeau, P., Degano, P.: Causal trees. In: Proc. of ICALP'89. LNCS, vol. 372, pp. 234–248. Springer (1989)
17. Degano, P., Nicola, R.D., Montanari, U.: Partial orderings descriptions and observations of nondeterministic concurrent processes. In: Proc. of REX Workshop. LNCS, vol. 354, pp. 438–466. Springer (1988)
18. Fournet, C., Gonthier, G.: The reflexive CHAM and the Join-calculus. In: Proc. of POPL'96. pp. 372–385. ACM Press (1996)
19. Fournet, C., Gonthier, G.: A hierarchy of equivalences for asynchronous calculi. J. Log. Algebr. Program. 63(1), 131–173 (2005)

20. van Glabbeek, R.J.: The linear time-branching time spectrum. In: Proc. of CONCUR'90. LNCS, vol. 458, pp. 278–297. Springer (1990)
21. van Glabbeek, R.J., Goltz, U.: Equivalence notions for concurrent systems and refinement of actions. In: Proc. of MFCS'89. LNCS, vol. 379, pp. 237–248. Springer (1989)
22. van Glabbeek, R.J., Goltz, U., Schicke, J.W.: Symmetric and asymmetric asynchronous interaction. In: Proc. of ICE'08. ENTCS, vol. 229(3), pp. 77–95. Elsevier (2009)
23. van Glabbeek, R.J., Vaandrager, F.W.: Petri net models for algebraic theories of concurrency. In: Proc. of PARLE'87. LNCS, vol. 259, pp. 224–242. Springer (1987)
24. Gorla, D.: Towards a unified approach to encodability and separation results for process calculi. In: Proc. of CONCUR'08. LNCS, vol. 5201, pp. 492–507. Springer (2008)
25. Honda, K., Tokoro, M.: An object calculus for asynchronous communication. In: Proc. of ECOOP'91. LNCS, vol. 512, pp. 133–147. Springer (1991)
26. Honda, K., Yoshida, N.: On reduction-based process semantics. Theor. Comp. Sci. 151(2), 437–486 (1995)
27. Kindler, E.: A compositional partial order semantics for Petri net components. In: Proc. of ATPN'97. LNCS, vol. 1248, pp. 235–252 (1997)
28. Lanese, I.: Concurrent and located synchronizations in π-calculus. In: Proc. of SOFSEM'07. LNCS, vol. 4362, pp. 388–399. Springer (2007)
29. Leifer, J.J., Milner, R.: Deriving bisimulation congruences for reactive systems. In: Proc. of CONCUR'00. LNCS, vol. 1877, pp. 243–258. Springer (2000)
30. Merro, M., Nardelli, F.Z.: Bisimulation proof methods for mobile ambients. In: Proc. of ICALP'03. LNCS, vol. 2719, pp. 584–598. Springer (2003)
31. Milner, R.: Communication and Concurrency. Prentice Hall (1989)
32. Milner, R.: Communicating and Mobile Systems: the π-Calculus. Cambridge University Press (1999)
33. Milner, R., Sangiorgi, D.: Barbed bisimulation. In: Proc. of ICALP '92. LNCS, vol. 623, pp. 685–695. Springer (1992)
34. Montanari, U., Pistore, M.: Concurrent semantics for the π-calculus. In: Proc. of MFPS'95. ENTCS, vol. 1. Springer (1995)
35. Palamidessi, C.: Comparing the expressive power of the synchronous and asynchronous π-calculi. Math. Str. in Comp. Sci. 13(5), 685–719 (2003)
36. Rathke, J., Sassone, V., Sobociński, P.: Semantic barbs and biorthogonality. In: Proceedings of FoSSaCS'07. LNCS, vol. 4423, pp. 302–316. Springer (2007)
37. Sassone, V., Sobociński, P.: A congruence for Petri nets. In: Proc. of PNGT'04. ENTCS, vol. 127, pp. 107–120. Elsevier (2005)
38. Selinger, P.: First-order axioms for asynchrony. In: Proc. of CONCUR'97. LNCS, vol. 1243, pp. 376–390. Springer (1997)