Fair Pi

Diletta Cacciagrano, Flavio Corradini, Catuscia Palamidessi

Fair $\Pi^{\star}$

Diletta Cacciagrano [1], Flavio Corradini [2]

Dipartimento di Matematica e Informatica
Università degli Studi di Camerino, Italy

Catuscia Palamidessi [3]

INRIA Futurs, LIX École Polytechnique, France

Abstract

In this paper, we define fair computations in the π-calculus [18]. We follow Costa and Stirling's approach for CCS-like languages [9,10] but exploit a more natural labeling method of process actions to filter out unfair process executions. The new labeling allows us to prove all the significant properties of the original one, such as unicity, persistence and disappearance of labels. It also turns out that the labeled π-calculus is a conservative extension of the standard one. We contrast the existing fair testing [3,19] with those that naturally arise by imposing weak and strong fairness as defined by Costa and Stirling. This comparison provides the expressiveness of the various fair testing-based semantics and emphasizes the discriminating power of the one already proposed in the literature.

Key words: Pi-Calculus, Testing Semantics, Strong Fairness, Weak Fairness.

1 Introduction

In the theory and practice of parallel systems, fairness plays an important role when describing the system dynamics. Several notions have been proposed in the literature, as in [9,10], where Costa and Stirling distinguish between fairness of actions in [9] (for a CCS-like language without restriction), and fairness of components in [10]. In both cases they distinguish between weak fairness and strong fairness. Weak fairness requires that if an action (a component,

* This work was supported by the Investment Funds for Basic Research (MIUR-FIRB) project Laboratory of Interdisciplinary Technologies in Bioinformatics (LITBIO) and by Halley Informatica.

[1] Email: diletta.cacciagrano@unicam.it
[2] Email: flavio.corradini@unicam.it
[3] Email: catuscia@lix.polytechnique.fr

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

resp.) can *almost always* proceed, then it must eventually do so, while strong fairness requires that if an action (a component, resp.) can proceed *infinitely often*, then it must proceed infinitely often. The main ingredients of the theory of fairness in [9] and [10] are:

- *A labeling method for process terms.* This allows to detect the action performed during a transition and the component responsible for it. Labels are strings in $\{0,1\}^*$, associated systematically with operators and basic actions inside a process. Along a computation, labels are unique and, once a label disappears, it does not reappear in the system anymore (unicity, persistence and disappearance properties).
- *Live actions (components, resp.).* An action (a component, resp.) of a process term is live if it can currently be performed (perform an action, resp.). In a term like $(\nu z)(x(y).\bar{z}w.0 \,|\, z(u).0)$, only an input action on x can be performed while no action on z can, momentarily.

In this paper, we adapt to the π-calculus [18] the approach to fairness which has been proposed in [9,10] for CCS-like languages [17]. A difference with [9,10] is that our labels are pairs $\langle w, n\rangle \in (\{0,1\}^* \times \mathbb{N})$. The first element, w, represents the position of the component (in the term structure) and depends only on the static operators (parallel and restriction). This element ensures the unicity of a label. The second element, n, provides information about the dynamics of the component, more precisely, it indicates how many actions that component has already executed since the beginning of the computation, and it depends only on the dynamic operator (prefix). This second element serves to ensure the disappearence property of a label. So, we have the unicity and disappearence properties of labels like in [9,10] but, differently from the latter, we keep separated the information about the static and dynamic operators. We believe that this new labeling method represents more faithfully the structure of a process and makes more intuitive the role of the label in the notion of fairness.

The proposed labeling technique allows to define weak and strong fair computations. At the top of them we introduce must testing semantics [1], to obtain the so-called weak-fair must semantics and strong-fair must semantics. These two fair testing semantics are compared with an existing one in the literature - the fair testing [3,19] - that does not need any labeling of actions. We present a comparison between fair testing and weak and strong-fair must semantics as well as with standard must testing. This comparison emphasizes the expressiveness of the different fair testing semantics, especially for what it concerns fair testing. We show interesting side-effects when the must testing is imposed over weak and strong-fair computations. In particular, any strong-fair computation is weak-fair too, while it turns out that the weak-fair must semantics is strictly finer than the strong-fair must one.

The rest of the paper is organized as follows. Section 2 presents the π-calculus. Section 3 defines must testing [1] and fair testing semantics [3,19].

Section 4 shows the labeling method and its main properties. Weak and strong-fair must semantics are defined in Section 5 and compared in Section 6. Finally, in Section 7 we investigate why strong and weak fairness notions are not enough to characterize fair testing semantics. As usual, Section 8 gathers several related work and Section 9 contains a few concluding remarks and further work. All of the proofs omitted in the body of the paper are in the appendixes.

2 The π-calculus

We now briefly recall the basic notions about the (choiceless) π-calculus. Let $\mathcal{N}$ (ranged over by $x, y, z, \ldots$) be a set of names. The set $\mathcal{P}$ (ranged over by $P, Q, R, \ldots$) of processes is generated by the following grammar:

$$P ::= \quad 0 \;\Big|\; x(y).P \;\Big|\; \tau.P \;\Big|\; \bar{x}y.P \;\Big|\; P \mid P \;\Big|\; (\nu x)P \;\Big|\; !x(y).P$$

The input prefix $y(x).P$, and the restriction $(\nu x)P$, act as name binders for the name x in P. The free names $fn(P)$ and the bound names $bn(P)$ of P are defined as usual. The set of names of P is defined as $n(P) = fn(P) \cup bn(P)$. Only input guarded terms can be in the scope of the bang operator, but this is not a real shortcoming, since this kind of replicator is as expressive as the full bang operator [14].

The operational semantics of processes is given via a labeled transition system, whose states are the process themselves. The labels (ranged over by $\mu, \gamma, \ldots$) "correspond" to prefixes, input xy, output $\bar{x}y$ and tau τ, and to the bound output $\bar{x}(y)$ (which models scope extrusion). If $\mu = xy$ or $\mu = \bar{x}y$ or $\mu = \bar{x}(y)$ we define $sub(\mu) = x$ and $obj(\mu) = y$. The functions fn, bn and n are extended to cope with labels as follows:

$$bn(xy) = \emptyset \qquad bn(\bar{x}(y)) = \{y\} \qquad bn(\bar{x}y) = \emptyset \qquad bn(\tau) = \emptyset$$
$$fn(xy) = \{x, y\} \qquad fn(\bar{x}(y)) = \{x\} \qquad fn(\bar{x}y) = \{x, y\} \qquad fn(\tau) = \emptyset$$

The transition relation is given in Table 1. We omit symmetric rules of Par, Com and Close for lake of space. We also assume alpha-conversion to avoid collision of free and bound names.

Definition 2.1 (*Weak transitions*) Let P and Q be $\mathcal{P}$ processes. Then:

- $P \stackrel{\varepsilon}{\Longrightarrow} Q$ iff $\exists\, P_0, ..., P_n \in \mathcal{P}$, $n \geq 0$, s.t. $P = P_0 \stackrel{\tau}{\longrightarrow} ... \stackrel{\tau}{\longrightarrow} P_n = Q$;
- $P \stackrel{\mu}{\Longrightarrow} Q$ iff $\exists\, P_1, P_2 \in \mathcal{P}$ s.t. $P \stackrel{\varepsilon}{\Longrightarrow} P_1 \stackrel{\mu}{\longrightarrow} P_2 \stackrel{\varepsilon}{\Longrightarrow} Q$.

Notation 2.1 For convenience, we write $x(y)$ and $\bar{x}y$ instead of $x(y).0$ and $\bar{x}y.0$, respectively. Furthermore, we write $P \stackrel{\mu}{\longrightarrow}$ (respectively $P \stackrel{\mu}{\Longrightarrow}$) to mean that there exists P' such that $P \stackrel{\mu}{\longrightarrow} P'$ (respectively $P \stackrel{\mu}{\Longrightarrow} P'$) and we write $P \stackrel{\varepsilon}{\Longrightarrow} \stackrel{\mu}{\longrightarrow}$ to mean that there are P' and Q such that $P \stackrel{\varepsilon}{\Longrightarrow} P'$ and $P' \stackrel{\mu}{\longrightarrow} Q$.

$$\text{Input}\ \ x(y).P \xrightarrow{xz} P\{z/y\}$$

$$\text{Output/Tau}\ \ \alpha.P \xrightarrow{\alpha} P \ \text{ where } \alpha = \bar{x}y \text{ or } \alpha = \tau$$

$$\text{Open}\ \frac{P \xrightarrow{\bar{x}y} P'}{(\nu y)P \xrightarrow{\bar{x}(y)} P'}\, x \neq y \qquad \text{Res}\ \frac{P \xrightarrow{\mu} P'}{(\nu y)P \xrightarrow{\mu} (\nu y)P'}\, y \notin n(\mu)$$

$$\text{Par}\ \frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q}\ bn(\mu) \cap fn(Q) = \emptyset$$

$$\text{Com}\ \frac{P \xrightarrow{xy} P',\ \ Q \xrightarrow{\bar{x}y} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \text{Close}\ \frac{P \xrightarrow{xy} P',\ \ Q \xrightarrow{\bar{x}(y)} Q'}{P \mid Q \xrightarrow{\tau} (\nu y)(P' \mid Q')}$$

$$\text{Bang}\ \ !x(y).P \xrightarrow{xz} P\{z/y\} \mid !x(y).P$$

Table 1
Early operational semantics for $\mathcal{P}$ terms.

3 Testing semantics

In this section we briefly summarize the basic definitions behind the testing machinery for the π-calculus.

Definition 3.1 (*Observers*)

- Let $\mathcal{N}' = \mathcal{N} \cup \{\omega\}$ be the set of names, assuming $\omega \notin \mathcal{N}$. By convention $fn(\omega) = bn(\omega) = \emptyset$. ω is used to report success.
- The set $\mathcal{O}$ (ranged over by $o, o', o'', \ldots$) of observers is defined like $\mathcal{P}$, where the grammar is extended with the production $P ::= \omega.P$.
- The operational semantics of $\mathcal{P}$ is extended to $\mathcal{O}$ by adding $\omega.P \xrightarrow{\omega} P$.

Definition 3.2 (*Experiments*) $\mathcal{E}$ denotes the set

$$\{\ (P \mid o) \mid\ P \in \mathcal{P} \text{ and } o \in \mathcal{O}\}$$

of experiments in $\mathcal{P}$.

Definition 3.3 (*Maximal Computations*) Given $P \in \mathcal{P}$ and $o \in \mathcal{O}$, a maximal computation from $P \mid o$ is either an infinite sequence of the form

$$P \mid o = T_0 \stackrel{\tau}{\longrightarrow} T_1 \stackrel{\tau}{\longrightarrow} T_2 \stackrel{\tau}{\longrightarrow} \ldots$$

or a finite sequence of the form

$$P \mid o = T_0 \stackrel{\tau}{\longrightarrow} T_1 \stackrel{\tau}{\longrightarrow} \ldots \stackrel{\tau}{\longrightarrow} T_n \stackrel{\tau}{\not\longrightarrow} .$$

We are now ready to define must and fair testing semantics.

Definition 3.4 (*Must and Fair Testing Semantics*) Given a process $P \in \mathcal{P}$ and an observer $o \in \mathcal{O}$, define:

- P *must* o if and only if *for every* maximal computation from $P \mid o$

$$P \mid o = T_0 \stackrel{\tau}{\longrightarrow} T_1 \stackrel{\tau}{\longrightarrow} \ldots T_i \, [\stackrel{\tau}{\longrightarrow} \ldots]$$

there exists $i \geq 0$ such that $T_i \stackrel{\omega}{\longrightarrow}$;

- P *fair* o if and only if *for every* maximal computation from $P \mid o$

$$P \mid o = T_0 \stackrel{\tau}{\longrightarrow} T_1 \stackrel{\tau}{\longrightarrow} \ldots T_i \, [\stackrel{\tau}{\longrightarrow} \ldots]$$

$T_i \stackrel{\omega}{\Longrightarrow}$, for every $i \geq 0$.

4 A labeled version of the π-calculus

Fairness imposes that concurrent subprocesses always eventually proceed unless they are deadlock processes or have terminated. Such a constraint will affect the behavior of processes. Consider the process $P \mid P$, where

$$P = (\nu a)(!a.\bar{a} \mid \bar{a})$$

and the following maximal computation

$$P \mid P \stackrel{\tau}{\longrightarrow} P \mid P \stackrel{\tau}{\longrightarrow} P \mid P \stackrel{\tau}{\longrightarrow} ..$$

We can not know whether the computation is fair or not, since we do not know which component (either on the right hand or on the left one of $\mid$), performs a synchronization at each step: we need to distinguish unambiguously actions of a concurrent system and to monitor them along its computations.

For this purpose, we extend to the π-calculus the label-based approach proposed in [10]. As explained in the introduction, however, we depart from [10] in the way we define the labels. In our case, labels are pairs whose first and second elements represent, respectively, the position of the component in the term and the number of actions already executed.

We proceed by defining $\mathcal{L}(B)$, as the language generated by the grammar

$$B ::= 0 \;\Big|\; \mu_{\langle s,n\rangle}.B \;\Big|\; (\nu x)B \;\Big|\; B \mid B \;\Big|\; !_{\langle s,n\rangle}x(y).P$$

where $s \in \{0,1\}^*$, $n \in \mathbb{N}$, $P \in \mathcal{P}$ and $\mu \in \{x(y), \bar{x}y, \tau\}$.

Then we define a binary relation $\Re$ over sets of labels and two functions, *top* and *lab*, allowing to obtain all labels appearing at the top of a labeled term and the whole labels set, respectively.

Definition 4.1 Let L_1, $L_2 \subseteq (\{0,1\}^* \times \mathbb{N})$. We define $L_1 \,\Re\, L_2$ if and only if $\forall\langle s_1, n_1\rangle \in L_1$, $\forall\langle s_2, n_2\rangle \in L_2$, $s_1 \not\leq s_2$ and $s_2 \not\leq s_1$, where $\leq$ is the usual prefix relation between strings.

Definition 4.2 Let $E \in \mathcal{L}(B)$. $top(E)$ and $lab(E)$ are defined by structural induction as follows:

$E = 0$:	$top(E) = \emptyset$	$lab(E) = \emptyset$
$E = \mu_{\langle s,n\rangle}.E'$:	$top(E) = \{\langle s, n\rangle\}$	$lab(E) = \{\langle s, n\rangle\} \cup lab(E')$
$E = (\nu x)E'$:	$top(E) = top(E')$	$lab(E) = lab(E')$
$E = E_1 \mid E_2$:	$top(E) = top(E_1) \cup top(E_2)$	$lab(E) = lab(E_1) \cup lab(E_2)$
$E = !_{\langle s,n\rangle}x(y).P$:	$top(E) = \{\langle s, n\rangle\}$	$lab(E) = \{\langle s, n\rangle\}$

Hence, we define a specific labeling function.

Definition 4.3 Let $P \in \mathcal{P}$. Define $L_{\langle s,n\rangle}(P)$, where $s \in \{0,1\}^*$ and $n \in \mathbb{N}$, inductively as follows:

$$\begin{array}{lcll}
L_{\langle s,n\rangle}(0) & = & 0 & \\
L_{\langle s,n\rangle}(\mu.P) & = & \mu_{\langle s,n\rangle}.L_{\langle s,n+1\rangle}(P) & (\mu \in \{x(y), \bar{x}y, \tau\}) \\
L_{\langle s,n\rangle}(P \mid Q) & = & L_{\langle s0,n\rangle}(P) \mid L_{\langle s1,n\rangle}(Q) & \\
L_{\langle s,n\rangle}((\nu x)P) & = & (\nu x)L_{\langle s,n\rangle}(P) & \\
L_{\langle s,n\rangle}(!x(y).P) & = & !_{\langle s,n\rangle}x(y).P &
\end{array}$$

Now, we are ready to define $\mathcal{P}^e$, the set of labeled π-calculus terms.

Definition 4.4 The labeled π-calculus, denoted by $\mathcal{P}^e$, is the set

$$\{E \in \mathcal{L}(B) \mid wf(E)\}$$

where $wf(E)$ is defined in Table 2.

$$\text{Nil}\ \frac{}{wf(0)} \qquad \text{Pref}\ \frac{\mu.P \in \mathcal{P}}{wf(L_{\langle s,n\rangle}(\mu.P))}$$

$$\text{Par}\ \frac{wf(E_1),\quad wf(E_2),\quad top(E_1)\ \Re\ top(E_2)}{wf(E_1 \mid E_2)}$$

$$\text{Res}\ \frac{wf(E)}{wf((\nu x)E)} \qquad \text{Bang}\ \frac{x(y).P \in \mathcal{P}}{wf(!_{\langle s,n\rangle}x(y).P)}$$

Table 2
Well formed terms.

4.1 Some properties of the labeled π-calculus

The operational semantics of $\mathcal{P}^e$ is similar to the one in Table 1; we simply ignore labels in order to derive a transition. As expected, the only rule that needs attention regards bang processes, because the unfolding generates new components and we must ensure unicity of labels. Since the unfolding puts two components in parallel, we exploit a proper dynamic labeling of the parallel components (Table 3). The intuition behind this rule follows by viewing $!_{\langle s,n\rangle}x(y).P$ as $L_{\langle s,n\rangle}(x(y).(P \mid !x(y).P))$.

$$\text{Bang}(\mathcal{P})\quad !x(y).P \xrightarrow{xz} P\{z/y\} \mid !x(y).P$$

$$\text{Bang}(\mathcal{P}^e)\quad !_{\langle s,n\rangle}x(y).P \xrightarrow{xz} L_{\langle s0,n+1\rangle}(P\{z/y\}) \mid !_{\langle s1,n+1\rangle}x(y).P$$

Table 3
Bang rules.

To give some more intuition, consider $S = x(y).(z(k).0 \mid \bar{z}h).0 \mid f.0$ and its labeled version $S' = x(y)_{\langle 0,0\rangle}.(z(k)_{\langle 00,1\rangle}.0 \mid \bar{z}h_{\langle 01,1\rangle}).0 \mid f_{\langle 1,0\rangle}.0$ [4].

Prefixes $x(y)$ and f in S are both top level prefixes. For this reason, they get labels of length 1; though the one on the left hand side of the parallel

[4] According to Costa and Stirling, we have: $S' = x(y)_0.(z(k)_{010}.0_{0101} |_{01} \bar{z}h_{011}.0_{0111})|_\varepsilon f_1.0_{11}$.

composition has been labeled 0, while the one on the right hand side has been labeled 1, just to distinguish the two prefixes. On the other hand, $z(k)$ and $\bar{z}h$ within the scope of $x(y)$ are both second level prefixes composed in parallel, so that they get 00 and 01 as different parallel subcomponents, respectively. However, as second action of the source component, they have the same index (i.e. 1). The significance of the second element of the labels is, of course, more evident when we consider more sequential processes.

It is possible to verify that $\forall E \in \mathcal{P}^e$, $top(E) \subseteq lab(E)$. $\mathcal{P}^e$ enjoys closure properties under any renamings σ, since σ does not change labels. Hence, it is closed under the execution of basic actions. Furthermore, no label occurs more than once in a labeled term (*unicity of labels*) and once a label disappears (it happens when the action related to such a label is performed) along a computation, it does not appear in the system anymore (*persistence and disappearance of labels*).

Lemma 4.5 Let $E \in \mathcal{P}^e$. Then:

1. No label $\langle s, n\rangle$ occurs more than once in E;
2. If $E \xrightarrow{\mu} E'$ then $\exists \langle s, n\rangle \in lab(E) : \langle s, n\rangle \notin lab(E')$;
3. $\forall k \geq 1 : E \xrightarrow{\mu_1} E_1 \xrightarrow{\mu_2} E_2 \xrightarrow{\mu_3} \ldots \xrightarrow{\mu_k} E_k$, if $\langle s, n\rangle \in lab(E) \cap lab(E_k)$ then $\langle s, n\rangle \in \bigcap_i lab(E_i)$, where $i \in [1..(k-1)]$.

As expected, the labeled language is also a *conservative extension* of the unlabeled one. To prove the statement, we have to formally define the π-calculus process obtained by deleting all the labels appearing within a labeled term.

Definition 4.6 Let $E \in \mathcal{P}^e$. Define $Unl(E)$ as the $\mathcal{P}$ process obtained by removing all the labels in E. It can be defined by induction as follows:

$$
\begin{array}{lcll}
Unl(0) & = & 0 & \\
Unl(\mu_{\langle s,n\rangle}.E) & = & \mu.Unl(E) & (\mu \in \{x(y), \bar{x}y, \tau\}) \\
Unl(E_1 \mid E_2) & = & Unl(E_1) \mid Unl(E_2) & \\
Unl((\nu x)E) & = & (\nu x)Unl(E) & \\
Unl(!_{\langle s,n\rangle}x(y).P) & = & !x(y).P &
\end{array}
$$

Then, we can prove the result, stated in the following lemma.

Lemma 4.7 Let $E \in \mathcal{P}^e$. Then:

1. $E \xrightarrow{\mu} E'$ implies $Unl(E) \xrightarrow{\mu} Unl(E')$;
2. $Unl(E) \xrightarrow{\mu} P'$ implies $\exists E' \in \mathcal{P}^e$ such that $E \xrightarrow{\mu} E'$ and $Unl(E') = P'$.

5 Strong and weak fairness

The labeling method proposed in the previous section can be extended in a natural way over experiments, adding $B ::= \omega.B$ in the grammar of $\mathcal{L}(B)$, $\omega.o \xrightarrow{\omega} o$ in the operational semantics and extending the functions $L_{\langle s,n\rangle}$, top, lab, Unl and the predicate wf as shown in Table 4.

$(L_{\langle s,n\rangle}/Unl)$	$L_{\langle s,n\rangle}(\omega.o)$	$=$	$Unl(L_{\langle s,n\rangle}(\omega.o))$	$=$	$\omega.o$
(top/lab)	$top(\omega.o)$	$=$	$lab(\omega.o)$	$=$	$\emptyset$
(wf)	$\dfrac{\omega.o \in \mathcal{O}}{wf(\omega.o)}$				

Table 4
Labeling method extension over experiments.

The definition of *live label* is crucial in every fairness notion. Given a labeled experiment $S \in \mathcal{E}^e$, a *live label* is a label associated to a top-level action which can immediately be performed, i.e. either a τ prefix or a input/output prefix able to synchronize. Table 5 defines live labels for a labeled experiment $S \in \mathcal{E}^e$, according to the labeling method proposed in Section 4. Since ω is a special action without complementary version (i.e. $\bar{\omega}$ is not defined), it is correct to assume that ω is not live. Furthemore, ω occurrences do not need to be observed: consequently, no label is associated to them.

Given a labeled experiment S, its set of live labels is denoted by $Lp(S)$. Notice that, by definition of liveness, if S can not perform any reduction (either an explicit τ action or a synchronization) then $Lp(S) = \emptyset$.

Definition 5.1 Let $S \in \mathcal{E}^e$, let $\langle s,n\rangle \in (\{0,1\}^* \times \mathbb{N})$.

$$Lp(S) = \{\langle s,n\rangle \in (\{0,1\}^* \times \mathbb{N}) \mid \mathit{live}(\langle s,n\rangle, \tau, S)\}$$

is the set of live labels associated to initial τ actions.

Since $top(S)$ is defined as the set of any labels appearing at the top of S, $Lp(S) \subseteq top(S)$ follows immediately by the definition of live actions.

In the following, labels will be denoted by $v, v_1, v_2, .. \in (\{0,1\}^* \times \mathbb{N})$ for convenience. $\mathcal{O}^e$ (ranged over by $\rho, \rho', ..$) denotes the set of observers and $\mathcal{E}^e$ denotes the set of labeled experiments in $\mathcal{P}^e$, as expected. Now, we can formally define two well-known notions of fairness.

Definition 5.2 (*Weak-fair Computations*) Given $S \in \mathcal{E}^e$, a *weak-fair computation* from S is a maximal computation,

$$S = S_0 \xrightarrow{\tau} S_1 \xrightarrow{\tau} S_2 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S_i \,[\xrightarrow{\tau} \ldots]$$

where $\forall v \in (\{0,1\}^* \times \mathbb{N})$, $\forall i \geq 0$, $\exists j \geq i$ such that $v \notin Lp(S_j)$.

$$\text{Tau}\ \frac{}{live(\langle s,n\rangle, \tau, \tau_{\langle s,n\rangle}.S)} \qquad \text{Input}\ \frac{x,y,z \in \mathcal{N}}{live(\langle s,n\rangle, xz, x(y)_{\langle s,n\rangle}.S)}$$

$$\text{Output}\ \frac{x,y \in \mathcal{N}}{live(\langle s,n\rangle, \bar{x}y, \bar{x}y_{\langle s,n\rangle}.S)} \qquad \text{Res}\ \frac{live(\langle s,n\rangle, \mu, S) \quad y \notin n(\mu)}{live(\langle s,n\rangle, \mu, (\nu y)S)}$$

$$\text{Open}\ \frac{live(\langle s,n\rangle, \bar{x}y, S) \quad x \neq y}{live(\langle s,n\rangle, \bar{x}(y), (\nu y)S)} \qquad \text{Bang}\ \frac{z \in \mathcal{N}}{live(\langle s,n\rangle, xz, !_{\langle s,n\rangle}x(y).S}$$

$$\text{Par}\ \frac{live(\langle s,n\rangle, \mu, S_1) \quad bn(\mu) \cap fn(S_2) = \emptyset}{live(\langle s,n\rangle, \mu, (S_1 \mid S_2))}$$

$$\text{Com}\ \frac{live(\langle s,n\rangle, xy, S_1), \quad live(\langle r,m\rangle, \bar{x}y, S_2)}{live(\langle s,n\rangle, \tau, S_1 \mid S_2), \ live(\langle r,m\rangle, \tau, S_1 \mid S_2)}$$

$$\text{Close}\ \frac{live(\langle s,n\rangle, xy, S_1), \quad live(\langle r,m\rangle, \bar{x}(y), S_2)}{live(\langle s,n\rangle, \tau, (\nu y)(S_1 \mid S_2)), \ live(\langle r,m\rangle, \tau, (\nu y)(S_1 \mid S_2))}$$

Table 5
Live labels.

Definition 5.3 (*Strong-fair Computations*) Given $S \in \mathcal{E}^e$, a *strong-fair computation* from S is a maximal computation,

$$S = S_0 \xrightarrow{\tau} S_1 \xrightarrow{\tau} S_2 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S_i \,[\xrightarrow{\tau} \ldots]$$

where $\forall v \in (\{0,1\}^* \times \mathbb{N})$, $\exists i \geq 0$ such that $\forall j \geq i$, $v \notin Lp(S_j)$.

A *weak-fair* computation is a maximal computation such that no label becomes live and then keeps on being live forever.

A *strong fair* computation is a maximal computation such that no label is live infinitely often, i.e. no label can become live, lose its liveness, become live again, etc. forever. Formally, strong fairness imposes that for every label there

is some point beyond which it never becomes live. Any finite computation is strong fair because all the actions, corresponding to live labels, are performed, and the computation stops when there is no reduction at all. Some useful results follow:

Theorem 5.4 For every labeled experiment $S \in \mathcal{E}^e$, then

1. every strong-fair computation from S is weak-fair, but not the vice versa;
2. there always is a strong-fair computation out of S.

Proof. (Sketch of:) Consider item (1). To prove the positive result it suffices to notice that a strong-fair computation is a special case of weak-fair computation. To prove the negative result, let $S :=!_{v_1} a \mid (\nu b)(\bar{b}_{v_2} \mid !_{v_3} b.(\bar{a} \mid \bar{b})) \mid a_{v_4}.\omega$ be an experiment: it is not difficult to check that there exists a maximal computation from S, along which a_{v_4} is never performed. It is weak-fair but not strong-fair.

Now consider item (2). It suffices to prove that $\forall S \in \mathcal{E}^e$,

(a) $Lp(S)$ is a finite set;

(b) $S \stackrel{\tau}{\not\longrightarrow}$ implies $Lp(S) = \emptyset$;

(c) $v \in Lp(S)$ implies $\exists S' \in \mathcal{E}^e$ such that $S \stackrel{\mu}{\longrightarrow} S'$ and for any S'' such that $S' \stackrel{\varepsilon}{\Longrightarrow} S''$, $v \notin Lp(S'')$;

(d) $\exists S' \in \mathcal{E}^e$ such that $S \stackrel{\varepsilon}{\Longrightarrow} S'$, $Lp(S) \cap Lp(S') = \emptyset$ and for any S'' such that $S' \stackrel{\varepsilon}{\Longrightarrow} S''$, $Lp(S) \cap Lp(S'') = \emptyset$.

□

6 Comparing fair semantics

In this section we provide a comparison among two different notions of fairness and the must semantics. It is easy to prove that $\forall P \in \mathcal{P}, \forall o \in \mathcal{O}$, P *must* o implies P *fair* o, but not the vice versa: it suffices to consider the process $P ::= (\nu a)(\bar{a} \mid !a.\bar{a}) \mid \bar{b}$ and the observer $o ::= b.\omega$.

Now, we try to add fairness in the must testing semantics and investigate the resulting semantic relations.

Definition 6.1 (*Strong/Weak-fair Must Semantics*) Let $E \in \mathcal{P}^e$ and $\rho \in \mathcal{O}^e$. Define E *sfmust* ρ (E *wfmust* ρ) if and only if *for every* strong (weak)- fair computation from $(E \mid \rho)$

$$E \mid \rho = S_0 \stackrel{\tau}{\longrightarrow} S_1 \stackrel{\tau}{\longrightarrow} \ldots \stackrel{\tau}{\longrightarrow} S_i \,[\stackrel{\tau}{\longrightarrow} \ldots]$$

$\exists i \geq 0$ such that $S_i \stackrel{\omega}{\longrightarrow}$.

6.1 Weak fairness and strong fairness in a must testing scenario

The following proposition states a very interesting result regarding weak and strong-fair must semantics. Notice that the positive implication follows by the fact that an unsuccessful strong-fair computation from an experiment $S = E|\rho$ is weak-fair too. This result seems to go against a well-established notion stating strong fairness a special case of weak fairness. More in details, it is well-known that strong fairness implies weak fairness, in the sense that a strong-fair computation is obviously weak-fair too. However, this implication is reversed when the must testing semantics is embedded in this fairness scenario: in the case that every weak-fair computation from an experiment is successful, then every strong-fair computation from the same experiment is successful.

Theorem 6.2 For every $E \in \mathcal{P}^e$ and $\rho \in \mathcal{O}^e$, then E *wfmust* ρ implies E *sfmust* ρ, but not the vice versa.

Must semantics imposes the success on any computation from a given experiment; that being so, any action leading to success in a weak-fair computation, can alternatively be live and lose its liveness only a finite number of steps, since its execution is surely forced to reach the success. It follows that a successful weak-fair computation collapses in a successful strong-fair computation. To prove the negative result, consider $E := !_{v_1}a \mid (\nu b)(\bar{b}_{v_2} \mid !_{v_3}b.(\bar{a} \mid \bar{b}))$ and $\rho := a_{v_4}.\omega$.

From $E \mid \rho$ there exists a maximal computation along which every live label different from v_4 is performed, while v_4 becomes live, loses its liveness, becomes live again, etc., without being performed: this computation is weak-fair by definition and unsuccessful. Notice that v_4 should be always performed in a strong-fair computation, determining the success of it.

Theorem 6.3 shows some interesting results by comparing weak/strong-fair must and must semantics.

Theorem 6.3 For every $E \in \mathcal{P}^e$ and $\rho \in \mathcal{O}^e$, then

1. $Unl(E)$ *must* $Unl(\rho)$ implies E *wfmust* ρ, but not the vice versa.
2. $Unl(E)$ *must* $Unl(\rho)$ implies E *sfmust* ρ, but not the vice versa.

Proof. (Sketch of:) Consider item (1): the positive result is trivial, since a successful weak-fair computation is a successful maximal computation. To prove the negative result, consider $E := (\nu a)(\bar{a}_{v_1} \mid !_{v_2}a.\bar{a}) \mid \bar{b}_{v_3}$ and $\rho := b_{v_4}.\omega$.

It is easy to check that $Unl(E)$ *mustUnl*(ρ). E *wfmust* ρ holds since, given a weak-fair computation from $E \mid \rho$, there has to exist a term performing ω, being v_4 already live since the beginning of the computation and having to lose its liveness at least once, by definition of weak fairness. In this case, losing liveness implies that b_{v_4} is performed. Item (2) is just a corollary of item (1) and Theorem 6.2. □

6.2 Weak and strong fairness vs fair testing semantics

Since weak-fair must semantics is strictly finer than strong-fair must one, the latter would look suitable to express fair testing semantics. However, Theorem 6.4 shows that not only the former but also the latter does not suffice to characterize fair testing semantics.

Theorem 6.4 For every $E \in \mathcal{P}^e$ and $\rho \in \mathcal{O}^e$, then

1. E *wfmust* ρ implies $Unl(E)$ *fair* $Unl(\rho)$, but not the vice versa.
2. E *sfmust* ρ implies $Unl(E)$ *fair* $Unl(\rho)$, but not the vice versa.

Proof. (Sketch of:) Consider item (2). Regarding the positive result, it is crucial to show that, given $S, S' \in \mathcal{E}^e$ such that $S' \stackrel{\varepsilon}{\Longrightarrow} S$, and a strong-fair computation $\mathcal{C}$ from S, then the computation obtained by prefixing $\mathcal{C}$ with $S' \stackrel{\varepsilon}{\Longrightarrow} S$ keeps on being strong-fair. Regarding the negative result of item (2), it is enough to consider $E := \bar{c}_{v_1} | !_{v_2} c.(\nu a)(\bar{a}|a.\bar{c}|a.\bar{b})$ and $\rho := b_{v_3}.\omega$. It easy to check that $Unl(E)$ *fair* $Unl(\rho)$, but there exists a strong-fair computation where v_3 never becomes live. Since v_3 prefixes the only ω occurrence along the given computation, the success will never be reached. Item (1) is just a corollary of item (2) and Theorem 6.2. □

7 Strong fairness and fair testing semantics

A more detailed interpretation of *live action* in the strong and weak fairness scenarios is crucial for both the negative results of Theorem 6.4. An action corresponding to a live label is not required to be performed to lose its liveness. Of course, when such an action is performed, then its label disappears forever. However, the label of an action may be present but no longer be live if, for example, a complementary action, which determines its liveness, is consumed in another synchronization.

We sketch why strong-fair must semantics (and, consequently, weak-fair must semantics) fails in attempt to characterize fair testing. For convenience, we say that a state performing ω is successful. P *fair* o means that, from every state in any maximal computation from $P \mid o$, a successful state can always be reached after finitely many interactions of live actions. It follows that, whenever there is a maximal computation from $P \mid o$ where a state T_i cannot lead to success at all (P *fair* o), any fair scheduling policy will always fail in attempt to obtain a successful state from T_i.

Indeed, there also exist experiments that satisfy the fair testing predicate and can perform some maximal unsuccessful computations. Consider, for instance, $P := \bar{c}|!c.(\nu a)(\bar{a}|a.\bar{c}|a.\bar{b})$ and $o := b.\omega$. Denote $Q_2 := (\nu a)(\bar{a}|a.\bar{c}|a.\bar{b})$. In the following (infinite) unsuccessful computation

$$P \mid o = \bar{c} \mid !c.Q_2 \mid \rho \stackrel{\tau}{\longrightarrow} Q_2 \mid !c.Q_2 \mid \rho \stackrel{\tau}{\longrightarrow} (\nu a)(a.\bar{b}) \mid \bar{c} \mid !c.Q_2 \mid \rho \stackrel{\tau}{\longrightarrow} \ldots$$
$$\stackrel{\tau}{\longrightarrow} (\nu a)(a.\bar{b}) \mid \ldots \mid (\nu a)(a.\bar{b}) \mid \bar{c} \mid !c.Q_2 \stackrel{\tau}{\longrightarrow} \ldots$$

ω is always prefixed and its prefix will never be performed, since any occurrence of $\bar{b}$ is prefixed in a deadlock term $(\nu a)(a.\bar{b})$. Notice that this computation is strong-fair, since strong fairness imposes that, after finitely many interactions, any action has to be either performed or disabled forever. Since the prefix b in $b.\omega$ is initially disabled, and keeps on being disabled forever, the computation is also unsuccessful, even if every state could perform ω after finitely many interactions. So, strong fairness gives to each live action only a finite number of chance to be performed and strong-fair must semantics does not admit unsuccessful maximal computations, while fair testing also admits unfair and unsuccessful maximal computations.

The following result emphasizes the reason behind the impossibility of characterizing strong-fair and weak-fair must semantics in terms of a fair testing-like semantics on the basis of the transition tree only.

Theorem 7.1 It is not possible to characterize *sfmust* and *wfmust* in terms of a fair testing-like semantics on the basis of the transition tree only.

Proof. Given

$$P := (\nu c)(\bar{c} \mid !c.(\bar{c} \mid \bar{a})) \mid (\nu c)(\bar{c} \mid !c.\bar{c})$$

and

$$Q := (\nu x)(\bar{x}a \mid (\nu b)(\bar{b} \mid \bar{x}b) \mid !x(y).(\bar{y} \mid \bar{x}y)) \mid (\nu c)(\bar{c} \mid !c.\bar{c}),$$

fairness assumptions distinguish P and Q: in fact, every strong (weak)-fair computation from P forces the execution of $\bar{a}$, sooner or later. This is not the case of some strong (weak)-fair computations from Q: occurrences of $\bar{x}a$ and $\bar{x}b$ compete to be performed infinitely often and, denoting by $\bar{b}^i$ $(i \geq 1)$ the parallel composition of i occurrences of $\bar{b}$, $(\nu b)(\bar{b}^{i+1} \mid \bar{x}b)$ is generated instead of $(\bar{a} \mid \bar{x}a)$ whenever one occurrence of $\bar{x}b$ in $(\nu b)(\bar{b}^i | \bar{x}b)$ is performed. That is the fairness constraint has not effect anymore. It follows that P and Q are neither *sfmust* nor *wfmust* equivalent, i.e. there exists some observer o that distinguishes P and Q w.r.t. both *sfmust* and *wfmust*. However, if we only consider transitions out of the terms P and Q, they are even strong bisimilar. It follows that $(P \mid o)$ e $(Q \mid o)$ are strong bisimilar, for every observer o. We conclude that a fair testing definition can not distinguish P and Q. □

8 Related work

Fairness is a key concept in systems modeling and verification. Different kinds of fairness have been proposed in process algebras (see, for instance, [12]). In this paper we adopt the definitions of weak and strong fairness proposed for CCS-like languages by Costa and Stirling in [9,10], to the π-calculus. An important result stated in [9,10] characterizes fair computations as the concatenation of certain finite sequences, called LP-steps that permits to think of fairness in terms of a 'localizable property' and not as a property of complete maximal executions. Almost simultaneously, two groups of authors [19], [3]

have come up with the so-called *fair testing*. They proposed two equivalent testing semantics with the property of abstracting from 'certain' divergences in contrast to the classical must testing. The idea is to modify the classical definition of must testing in such a way that the success can always be reached after finitely many steps. Both groups of authors present alternative characterizations of the new fair testing semantics. In [4], the framework described in [3] is extended to consider a set of sound axioms for fair testing and with more examples showing the usefulness of the new semantics. Another interesting paper is [8], where the authors generate a natural hierarchy of equivalences for asynchronous name-passing process calculi based on variations of Milner and Sangiorgi's weak barbed bisimulation. The considered calculi (based on π-calculus and join calculus) are asynchronous in the sense of [13]. After defining a particular class of contexts, called *evaluation contexts* - contexts with only one hole and unguarded - they prove that barbed congruence coincides with Honda and Yoshida's reduction equivalence and, when the calculus includes name matching, with asynchronous labeled bisimulation. They also show that barbed congruence is coarser than reduction equivalence when only one barb is tested. By combining simulation coupling and barbed properties, they prove that every coupled barbed equivalence strictly implies fair testing equivalence. They show that both relations coincide in the join calculus and on a restricted version of the π-calculus where reception occurs only on names bound by a restriction (not on free names and not on received names). In [15], Koomen explains fairness with probabilistic arguments: Fair Abstraction Rule says that no matter how small the probability of success, if you try often enough you will eventually succeed. The probabilistic intuitions motivating this rule are formalized in [20], where the authors define a probabilistic testing semantics which can be used to alternatively characterize *fair testing*. The key idea is to define this new semantics in such a way that two non-probabilistic processes are fair-equivalent if and only if any probabilistic version of both processes are equivalent in the probabilistic testing semantics. In order to get this result, the authors define a simple probabilistic must semantics, by saying that a probabilistic process *must* satisfy a test if and only if the probability with which the process satisfies the test equals 1. The subject of fairness in probabilistic systems has been widely discussed in the literature; Pnueli [21] introduces the notion of *extreme fairness* and α-*fairness*, to abstract from the precise values of probabilities.

9 Conclusion and future work

In this paper, we define a labeled version of the π-calculus [18], importing techniques in [9,10] for CCS-like languages. We compare weak and strong fairness and prove that both notions of fairness are not enough to characterize fair testing semantics and we state the main reason of this failure. The results scale to the asynchronous π-calculus [2] and do not depend on the proposed

labeling method. As a future work, we plan to investigate on the existence of alternative characterizations of the investigated fairness notions, allowing simple and finite representations of fair computations such as the use of regular expressions as in [6,7]. It is also interesting to investigate on the impact that these different notions of fairness have on the encodings from the π-calculus into the asynchronous π-calculus [5].

References

[1] Boreale, M. & De Nicola, R., *Testing Equivalence for Mobile Processes*, Information and Computation, **120** (1995), 279-303.

[2] Boudol, G., "Asynchrony and the π-calculus", Technical Report 1702, INRIA, Sophia-Antipolis (1992).

[3] Brinksma, E., Rensink, A. & Vogler, W., *Fair Testing*, Proc. of CONCUR'95, LNCS, **962** (1995), 313-327.

[4] Brinksma. E., Rensink, A. & Vogler, W., "Applications of Fair Testing", In "Protocols Specification, Testing and Verification" (XVI), Chapman & Hall (1996), 145-160.

[5] Cacciagrano, D., Corradini, F. & Palamidessi, C., *Separation of Synchronous and Asynchronous Communication Via Testing*, Proc. of EXPRESS'05, ENTCS, **154**(3)(2006), 95-108. To appear in Theoretical Computer Science.

[6] Corradini, F., Di Berardini, M.R. & Vogler, W., *Relating Fairness and Timing in Process Algebra*, Proc. of Concur'03, LNCS, **2761** (2003), 438-452.

[7] Corradini, F., Di Berardini, M.R. & Vogler, W., *Fairness of Components in System Computations*, ENTCS, **128**(2) (2005), 35-52.

[8] Fournet, C. & Gonthier, G., *A Hierarchy of Equivalences for Asynchronous Calculi*, Proc. of ICALP'98 (1998), 844-855.

[9] Costa, G. & Stirling, C., *A Fair Calculus of Communicating Systems*, Acta Informatica, **21** (1984), 417-441.

[10] Costa, G. & Stirling, C., *Weak and Strong Fairness in CCS*, Information and Computation, **73** (1987), 207-244.

[11] Francez, N., "Fairness", Springer-Verlag (1986).

[12] Hennessy, M., *An Algebraic Theory of Fair Asynchronous Communicating Processes*, Theoretical Computer Science, **49** (1987), 121-143.

[13] Honda, K. & Tokoro, M., *An Object calculus for Asynchronous Communication*, Proc. of ECOOP '91, LNCS, **512** (1991), 133-147.

[14] Honda, K. & Yoshida, N., *Replication in Concurrent Combinators*, Proc. of TACS '94, LNCS, **789** (1994).

[15] Koomen, C., *Albegraic Specification and Verification of Communications protocols*, Science of Computer Programming, **5** (1985), 1-36.

[16] Lehmann, D., Pnueli, A. & Stavi, J., *Impartiality, justice and Fairness:the Ethics of Concurrent Termination*, Proc. of 8th Int. Colloq. Aut. Lang. Prog., LNCS, **115** (1981), 264-277.

[17] Milner, R., "Communication and Concurrency", Prentice-Hall International (1989).

[18] Milner, R., Parrow, J. & Walker, D., *A Calculus of Mobile Processes*, Part I and II, Information and Computation, **100** (1992), 1-78.

[19] Natarajan, V. & Cleaveland, R., *Divergence and Fair Testing*, Proc. of ICALP '95, LNCS, **944** (1995), 648-659.

[20] Núñez, M. & Rupérez, D., *Fair testing through probabilistic testing*, Acta Informatica, **19** (1983), 195-210 . Protocol Specification, Testing, and Verification, **19** (1999), Kluwer Academic Publishers, 135-150.

[21] Pnueli, A., *On the Extremely Fair Treatment of Probabilistic Algorithms*, Proc. of ACM Symph. Theory of Comp. (1983), 278-290 .

[22] Queille, J.P. & Sifakis, J., *Fairness and Related Properties in Transition Systems-A Temporal Logic to Deal with Fairness*, Acta Informatica, **19** (1983), 195-210.

Appendix A: a labeled version of the π-calculus

This appendix section contains intermediate results and proofs of the statements omitted in Section 4. Several proofs follow the same lines as the corresponding results in [10].

Lemma 9.1 Let $E \in \mathcal{P}^e$. Then $top(E) \subseteq lab(E)$.

Proof. By induction on the structure of E.

- $E = 0$: $top(0) = \emptyset$ and $lab(0) = \emptyset$;
- $E = L_{\langle s,n\rangle}(\mu.P)$: $top(E) = \{\langle s, n\rangle\}$ and $lab(E) = \{\langle s, n\rangle\} \cup lab(L_{\langle s,n+1\rangle}(P))$;
- $E = (E_1 \mid E_2)$: then $top(E_1 \mid E_2) = top(E_1) \cup top(E_2)$ and $lab(E_1 \mid E_2) = lab(E_1) \cup lab(E_2)$. By induction $top(E_1) \subseteq lab(E_1)$ and $top(E_2) \subseteq lab(E_2)$. Hence $top(E_1 \mid E_2) \subseteq lab(E_1 \mid E_2)$;
- $E =!_{\langle s,n\rangle} x(y).P$: then $top(E) = \{\langle s, n\rangle\} = lab(E)$;
- Case $E = (\nu x)E'$ can be proven similarly.

□

Lemma 9.2 Let $E = L_{\langle r,m\rangle}(P)$, for some $P \in \mathcal{P}$. Then $\forall \langle s, n\rangle \in lab(E)$, $r \leq s$ and $m \leq n$.

Proof. By induction on the structure of P.

- $E = 0$: then $lab(0) = \emptyset$;
- $E = L_{\langle r,m\rangle}(\mu.P)$: then $lab(E) = \{\langle r, m\rangle\} \cup lab(L_{\langle r,m+1\rangle}(P))$;
- $E = L_{\langle r,m\rangle}(P_1 \mid P_2)$: $lab(L_{\langle r,m\rangle}(P_1 \mid P_2)) = lab(L_{\langle r0,m\rangle}(P_1)) \cup lab(L_{\langle r1,m\rangle}(P_2))$. By induction, $\forall\langle s_1, n_1\rangle \in lab(L_{\langle r0,m\rangle}(P_1))$, $r \leq r0 \leq s_1$ and $m \leq n_1$. Analogously, $\forall\langle s_2, n_2\rangle \in lab(L_{\langle r1,m\rangle}(P_2))$, $r \leq r1 \leq s_2$ and $m \leq n_2$.
- $E = !_{\langle r,m\rangle}x(y).P$: then $lab(E) = \{\langle r, m\rangle\}$;
- Case $E = L_{\langle r,m\rangle}((\nu x)P)$ can be proven similarly.

□

Lemma 9.3 $\forall P \in \mathcal{P}, \forall r \in \{0,1\}^*$ and $\forall n \in \mathbb{N}$, $wf(L_{\langle r,m\rangle}(P))$.

Proof. By induction on the structure of P.

- $P = 0, \mu.P', !x(y).P'$: these cases are trivial;
- $P = P_0 \mid P_1$: then $L_{\langle r,m\rangle}(P_0 \mid P_1) = L_{\langle r0,m\rangle}(P_0) \mid L_{\langle r1,m\rangle}(P_1)$ and by Lemma 9.2 on $top(L_{\langle ri,m\rangle}(P_i))$ we have that $\forall\langle s_i, n_i\rangle \in top(L_{\langle ri,m\rangle}(P_i))$, $ri \leq s_i$ and $m \leq n_i$ $(i \in \{0,1\})$. Hence $top(L_{\langle r0,m\rangle}(P_0)) \Re\, top(L_{\langle r1,m\rangle}(P_1))$;
- $P = (\nu x)P'$: then $L_{\langle r,m\rangle}(P) = (\nu x)L_{\langle r,m\rangle}(P')$, where $wf(L_{\langle r,m\rangle}(P'))$. Hence $wf(L_{\langle r,m\rangle}(P))$.

□

Lemma 9.4 Let $E \in \mathcal{P}^e$. Then $\forall\langle s, n\rangle \in lab(E)$, $\exists\langle r, m\rangle \in top(E)$ such that $r \leq s$ and $m \leq n$.

Proof. By induction on the structure of E.

- $E = 0$: $top(0) = \emptyset$ and $lab(0) = \emptyset$;
- $E = L_{\langle s,n\rangle}(\mu.P')$: then $top(E) = \{\langle s, n\rangle\}$. It is enough to apply Lemma 9.2;
- $E = (E_1 \mid E_2)$: then $top(E_1 \mid E_2) = top(E_1) \cup top(E_2)$ and $lab(E_1 \mid E_2) = lab(E_1) \cup lab(E_2)$. By induction, $\forall\langle s_1, n_1\rangle \in lab(E_1)$, $\exists\langle r_1, m_1\rangle \in top(E_1)$ s.t. $r_1 \leq s_1$ and $m_1 \leq n_1$; analogously $\forall\langle s_2, n_2\rangle \in lab(E_2)$, $\exists\langle r_2, m_2\rangle \in top(E_2)$ s.t $r_2 \leq s_2$ and $m_2 \leq n_2$;
- Case $E = (\nu x)E'$ can be proven similarly;
- $E = !_{\langle s,n\rangle}x(y).P$: then $top(E) = \{\langle s, n\rangle\} = lab(E)$.

□

Lemma 9.5 Let $E \in \mathcal{P}^e$ such that $E \xrightarrow{\mu} E'$. Then:

1. $\forall\langle r', m'\rangle \in top(E')$, $\exists\langle r, m\rangle \in top(E)$ such that $r \leq r'$ and $m < m'$;
2. $\forall\langle s', n'\rangle \in lab(E')$, $\exists\langle s, n\rangle \in lab(E)$ such that $s \leq s'$ and $n < n'$.
3. $E' \in \mathcal{P}^e$;

Proof.

(1) By induction on the depth of $E \xrightarrow{\mu} E'$.

Rule Input/Output/Tau: $E = L_{\langle s,n\rangle}(\mu.P') \xrightarrow{\mu} E'' = L_{\langle s,n+1\rangle}(P'')$ (either

$P'' = P'$ or $P'' = P'\{z/y\}$). It suffices to notice that $top(L_{\langle s,n\rangle}(\mu.P')) = \{\langle s,n\rangle\}$ and to apply Lemma 9.2 on $top(E'')$;

Rule Par: $E = (E_1 \mid E_2) \xrightarrow{\mu} (E_1' \mid E_2)$, where $bn(\mu) \cap fn(E_2) = \emptyset$. Since $wf(E_1 \mid E_2)$, then $wf(E_1)$, $wf(E_2)$ and $top(E_1)\Re top(E_2)$, then $E_1 \xrightarrow{\mu} E_1'$ and, by induction, $\forall\langle r_1', m_1'\rangle \in top(E_1')$, $\exists\langle r_1, m_1\rangle \in top(E_1)$ such that $r_1 \leq r_1'$ and $m_1 < m_1'$. Since $top(E_1' \mid E_2) = top(E_1') \cup top(E_2)$, then $\forall\langle r'', m''\rangle \in top(E_1' \mid E_2)$, $\exists\langle \tilde{r}, \tilde{m}\rangle \in top(E)$ such that $\tilde{r} \leq r''$ and $\tilde{m} < m''$;

Rule Open/Res/Com/Close: These cases can be proven similarly.

Rule Bang: $!_{\langle s,n\rangle}x(y)P' \xrightarrow{xz} L_{\langle s0,n+1\rangle}(P'\{z/y\}) \mid !_{\langle s1,n+1\rangle}x(y).P'$. Then we have $top(!_{\langle s,n\rangle}x(y).P') = \{\langle s,n\rangle\}$ and $top(L_{\langle s0,n+1\rangle}(P'\{z/y\})|!_{\langle s1,n+1\rangle}x(y).P') = \{\langle s1, n+1\rangle\} \cup top(L_{\langle s0,n+1\rangle}(P'\{z/y\}))$. It suffices to apply Lemma 9.2 on $top(L_{\langle s0,n+1\rangle}(P'\{z/y\}))$.

(2) $\forall\langle r', m'\rangle \in top(E')$, $\exists\langle s,n\rangle \in top(E)$ such that $s \leq r'$ and $n < m'$; since $top(E') \subseteq lab(E')$ and $\forall\langle s', n'\rangle \in lab(E')$, $\exists\langle r', m'\rangle \in top(E')$ such that $r' \leq s'$ and $m' < n'$ (Lemma 9.4), it follows that $\forall\langle s', n'\rangle \in lab(E')$, $\exists\langle s,n\rangle \in top(E)$ such that $s \leq s'$ and $n < n'$. Hence, $\forall\langle s', n'\rangle \in lab(E')$ $\exists\langle s,n\rangle \in lab(E)$ such that $s \leq s'$ and $n < n'$.

(3) We prove that $wf(E')$ holds, by induction on the depth of $E \xrightarrow{\mu} E'$.

Rule Input/Output/Tau: $E = L_{\langle s,n\rangle}(\mu.P') \xrightarrow{\mu} E'' = L_{\langle s,n+1\rangle}(P'')$ (either $P'' = P'$ or $P'' = P'\{z/y\}$). By Lemma 9.3, $wf(L_{\langle s,n+1\rangle}(P''))$;

Rule Par: $E = (E_1 \mid E_2) \xrightarrow{\mu} (E_1' \mid E_2)$, where $bn(\mu) \cap fn(E_2) = \emptyset$. Since $wf(E_1 \mid E_2)$, then $top(E_1)\Re top(E_2)$. Then $E_1 \xrightarrow{\mu} E_1'$ and, by induction, $wf(E_1')$; by (i), $\forall\langle r_1', m_1'\rangle \in top(E_1')$, $\exists\langle r_1, m_1\rangle \in top(E_1)$ such that $r_1 \leq r_1'$ and $m_1 \leq m_1'$. Since $\forall\langle r_1, m_1\rangle \in top(E_1)$, $\forall\langle r_2, m_2\rangle \in top(E_2)$ we have $r_1 \not\leq r_2$ and $r_2 \not\leq r_1$, then $\forall\langle r_1', m_1'\rangle \in top(E_1')$, $\forall\langle r_2, m_2\rangle \in top(E_2)$ we have $r_1' \not\leq r_2$ and $r_2 \not\leq r_1'$, that is $top(E_1')\ \Re\ top(E_2)$. Hence $wf(E_1' \mid E_2)$;

Rule Open/Res/Com/Close: These cases can be proven similarly.

Rule Bang: it suffices to recall that $top(L_{\langle s0,n+1\rangle}(P'\{z/y\})|!_{\langle s1,n+1\rangle}x(y).P') = \{\langle s1, n+1\rangle\} \cup top(L_{\langle s0,n+1\rangle}(P'\{z/y\}))$.

□

Lemma 4.5 Let $E \in \mathcal{P}^e$. Then:

1. No label $\langle s,n\rangle$ occurs more than once in E;
2. If $E \xrightarrow{\mu} E'$ then $\exists\langle s,n\rangle \in lab(E) : \langle s,n\rangle \notin lab(E')$;
3. $\forall k \geq 1 : E \xrightarrow{\mu_1} E_1 \xrightarrow{\mu_2} E_2 \xrightarrow{\mu_3} \ldots \xrightarrow{\mu_k} E_k$, if $\langle s,n\rangle \in lab(E) \cap lab(E_k)$ then $\langle s,n\rangle \in \bigcap_i lab(E_i)$, where $i \in [1..(k-1)]$.

Proof.

(1) By induction on the structure of E.

- $E = 0$: then $lab(0) = \emptyset$;
- $E = L_{\langle s,n\rangle}(\mu.P')$: then $lab(E) = \{\langle s,n\rangle\} \cup lab(L_{\langle s,n+1\rangle}(P'))$. By induction $\forall \langle s',n'\rangle \in lab(L_{\langle s,n+1\rangle}(P'))$, $\langle s',n'\rangle$ does not occur more than once in $lab(L_{\langle s,n+1\rangle}(P'))$. Moreover, by Lemma 9.2 $\forall \langle s',n'\rangle \in lab(L_{\langle s,n+1\rangle}(P'))$, $s \leq s'$ and $n+1 \leq n'$. Hence $\langle s,n\rangle \notin lab(L_{\langle s,n+1\rangle}(P'))$;
- $E = (E_1 \mid E_2)$: then $lab(E) = lab(E_1) \cup lab(E_2)$. By induction, $\forall i \in \{\langle 1,2\rangle\} \forall \langle s_i,n_i\rangle \in lab(E_i)$, $\langle s_i,n_i\rangle$ does not occur more than once in $lab(E_i)$. Since $\forall i \in \{\langle 1,2\rangle\}$, $\forall \langle s_i,n_i\rangle \in lab(E_i)$, $\exists \langle r_i,m_i\rangle \in top(E_i)$ such that $r_i \leq s_i$ and $m_i \leq n_i$ and $top(E_1)\ \Re\ top(E_2)$, then $\forall \langle s_1,n_1\rangle \in lab(E_1), \forall \langle s_2,n_2\rangle \in lab(E_2)$, $\langle s_1,n_1\rangle \neq \langle s_2,n_2\rangle$. Hence $\forall i \in \{\langle 1,2\rangle\}$, $\forall \langle s_i,n_i\rangle \in lab(E_i)$, $\langle s_i,n_i\rangle$ does not occur more than once in $lab(E)$;
- Cases $E = (\nu x)E'$ and $E = !_{\langle s,n\rangle}x(y).P'$ can be proven similarly.

(2) It suffices to prove that $E \xrightarrow{\mu} E'$ implies $\exists \langle s,n\rangle \in top(E)$ s.t. $\langle s,n\rangle \notin top(E')$. By induction on the depth of $E \xrightarrow{\mu} E'$.

Rule Input/Output/Tau: $E = L_{\langle s,n\rangle}(\mu.P') \xrightarrow{\mu} E'' = L_{\langle s,n+1\rangle}(P'')$ (either $P'' = P'$ or $P'' = P'\{z/y\}$). Since $top(L_{\langle s,n\rangle}(\mu.P')) = \{\langle s,n\rangle\}$ and, by Lemma 9.2, $\forall \langle s',n'\rangle \in top(L_{\langle s,n+1\rangle}(P''))$, $s \leq s'$ and $n+1 \leq n'$, we have that $\langle s,n\rangle \notin top(L_{\langle s,n+1\rangle}(P''))$;

Rule Par: $E = (E_1 \mid E_2) \xrightarrow{\mu} (E_1' \mid E_2)$, where $bn(\mu) \cap fn(E_2) = \emptyset$. Then $E_1 \xrightarrow{\mu} E_1'$ and, by induction, $\exists \langle r_1,m_1\rangle \in top(E_1) : \langle r_1,m_1\rangle \notin top(E_1')$. Since $\langle r_1,m_1\rangle \notin top(E_2)$, then $\langle r_1,m_1\rangle \notin top(E_1' \mid E_2)$;

Rule Open/Res/Com/Close/Bang: These cases can be proven similarly.

(3) The statement can be proven by induction on k. If $k = 1$, the statement holds by definition. If $k = 2$, the proof proceeds by induction on the depth of the derivation $E_1 \xrightarrow{\mu_2} E_2$, applying item (2) of Lemma 9.5 on $E_1 \xrightarrow{\mu_2} E_2$, and applying item (2) of the current lemma on $E \xrightarrow{\mu_1} E_1$. □

Appendix B: comparing testing semantics and fairness policies

This appendix section contains intermediate results and proofs of the statements omitted in Section 5.

9.1 Weak fairness and strong fairness

Proposition 9.6 For every labeled experiment $S \in \mathcal{E}^e$, every strong-fair computation from S is weak-fair, but not the vice versa.

Proof. The positive result is trivial, since strong fairness is a special case of weak fairness. To prove the negative result, consider $E = !_{v_1^0}a|(\nu b)(\bar{b}_{v_2^0}|!_{v_3^0}b.(\bar{a}|\bar{b}))$, $\rho = a_{v_4^0}.\omega$ and the maximal computation (we omit 0 term by convenience)

$$\mathcal{C} ::= E \mid \rho = S_0 \xrightarrow{\tau} S_1 \xrightarrow{\tau} S_2 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S_i \xrightarrow{\tau}$$

where $\forall j \geq 0, Q_2(v^j_{2,3}) := (\nu b)(\bar{b}_{v^j_2} \mid !_{v^j_3} b.(\ \bar{a}\ \mid\ \bar{b}))$ and

$$\begin{array}{ll}
S_0 = !_{v^0_1} a \mid Q_2(v^0_{2,3}) \mid a_{v^0_4}.\omega & \ldots \\
S_1 = !_{v^0_1} a \mid \bar{a}_{v^1_5} \mid Q_2(v^1_{2,3}) \mid a_{v^0_4}.\omega & S_i = !_{v^i_1} a \mid Q_2(v^{i-1}_{2,3}) \mid a_{v^0_4}.\omega \\
S_2 = !_{v^2_1} a \mid Q_2(v^1_{2,3}) \mid a_{v^0_4}.\omega & S_{i+1} = !_{v^i_1} a \mid \bar{a}_{v^{i+1}_5} \mid Q_2(v^{i+1}_{2,3}) \mid a_{v^0_4}.\omega \\
S_3 = !_{v^2_1} a \mid \bar{a}_{v^3_5} \mid Q_2(v^3_{2,3}) \mid a_{v^0_4}.\omega & S_{i+2} = !_{v^{i+2}_1}\ a \mid Q_2(v^{i+1}_{2,3}) \mid a_{v^0_4}.\omega \\
S_4 = !_{v^4_1} a \mid Q_2(v^3_{2,3}) \mid a_{v^0_4}.\omega & \ldots
\end{array}$$

Notice that, in $\mathcal{C}$, we have $v^0_4 \notin Lp(S_0), v^0_4 \in Lp(S_1), v^0_4 \notin Lp(S_2), v^0_4 \in Lp(S_3)$, $\ldots, v^0_4 \notin Lp(S_i), v^0_4 \in Lp(S_{i+1}), v^0_4 \notin Lp(S_{i+2}), \ldots$ and so on. Moreover $\forall v \in Lp(S_j)$, where $v \neq v^0_4$, there exists $k > j$ such that $v \notin Lp(S_k)$. It follows that $\mathcal{C}$ is weak-fair but it is not strong-fair. □

Proposition 9.7 For any labeled experiment S there is a strong-fair computation out of S.

Proof. It suffices to prove that $\forall S \in \mathcal{E}^e$

(a) $Lp(S)$ is a finite set;

(b) $S \xrightarrow{\tau}\!\!\!\!\!\!/\ \ $ implies $Lp(S) = \emptyset$;

(c) $v \in Lp(S)$ implies $\exists S' \in \mathcal{E}^e$ such that $S \xrightarrow{\mu} S'$ and for any S'' such that $S' \stackrel{\varepsilon}{\Longrightarrow} S''$, $v \notin Lp(S'')$;

(d) $\exists S' \in \mathcal{E}^e$ such that $S \stackrel{\varepsilon}{\Longrightarrow} S'$, $Lp(S) \cap Lp(S') = \emptyset$ and $\forall S''$ such that $S' \stackrel{\varepsilon}{\Longrightarrow} S''$, $Lp(S) \cap Lp(S'') = \emptyset$.

We recall that $\forall S \in \mathcal{E}^e, Lp(S) \subseteq top(S) \subseteq lab(S)$. Items (a) and (b) are trivial. Consider Item (c). S' is the term obtained from S by performing the action labeled by v: by Lemma 4.5 , $v \notin lab(S')$ and $\forall S''$ such that $S' \stackrel{\varepsilon}{\Longrightarrow} S''$, $v \notin lab(S'')$. Hence $v \notin Lp(S')$ and $\forall S''$ such that $S' \stackrel{\varepsilon}{\Longrightarrow} S''$, $v \notin Lp(S'')$.

To prove item (d) it suffices to apply the previous item, where $\mu = \tau$. S' is the term obtained from S by performing any $v \in Lp(S)$ and such that $\forall v \in Lp(S), \forall S'' : S' \stackrel{\varepsilon}{\Longrightarrow} S''$ either $v \notin lab(S')$ (following that $v \notin lab(S'')$) or $v \notin Lp(S'')$ and $v \in lab(S')$. In both cases, $Lp(S) \cap Lp(S') = \emptyset$ and $Lp(S) \cap Lp(S'') = \emptyset$. Since $Lp(S)$ is finite, such S' exists.

Now, we can prove the main statement. If $S \xrightarrow{\tau}\!\!\!\!\!\!/\ \ $, then the empty computation is strong-fair, since $Lp(S) = \emptyset$. Otherwise, there exists a maximal computation $\mathcal{C}$

$$S = S_0 \xrightarrow{\tau} S^1_0 \xrightarrow{\tau} .. \xrightarrow{\tau} S^{n_0}_0 \xrightarrow{\tau} S_1 \,[\xrightarrow{\tau} S^1_1 \xrightarrow{\tau} .. \xrightarrow{\tau} S^{n_1}_1 \xrightarrow{\tau} S_2 \xrightarrow{\tau} \ldots]$$

where $\forall i \geq 0, Lp(S_i) \cap Lp(S_{i+1}) = \emptyset$ and $\forall j \geq i, Lp(S_i) \cap Lp(S_j) = \emptyset$. Suppose, by contradiction, that $\mathcal{C}$ is not strong-fair: then there exists a label v such that $\forall i \geq 0, \exists j \geq i : v \in Lp(\tilde{S})$, where either $\tilde{S} = S_j$ or $\tilde{S} = S^k_j$, contradicting the hypothesis on $\mathcal{C}$. □

9.2 Must and fair testing semantics

Proposition 9.8 Let $P \in \mathcal{P}$ and $o \in \mathcal{O}$. Then P *must* o implies P *fair* o.

Proof. By contradiction, suppose P *f̸air* o, that is there exists a maximal computation from $P \mid o$

$$\mathcal{C} ::= P \mid o = T_0 \xrightarrow{\tau} T_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} T_i \,[\xrightarrow{\tau} \ldots]$$

such that $T_i \stackrel{\omega}{\Longrightarrow}$ for some $i \geq 0$, i.e. $\forall T' : T_i \stackrel{\varepsilon}{\Longrightarrow} T'$ it holds that $T' \stackrel{\omega}{\not\longrightarrow}$. It follows that $T_i \stackrel{\omega}{\not\longrightarrow}$, $\forall j \in [0..(i-1)], T_j \stackrel{\omega}{\not\longrightarrow}$ and $\forall h \geq i, T_h \stackrel{\omega}{\not\longrightarrow}$, by hypothesis on T_i. In fact, since ω does not appear in a choice operator and can not synchronize, it does not disappear once it is at the top level of a term. It follows that the above computation $\mathcal{C}$ is such that $\forall j \geq 0, T_j \stackrel{\omega}{\not\longrightarrow}$, i.e. P *m̸ust* o. □

Proposition 9.9 There exist $P \in \mathcal{P}$ and $o \in \mathcal{O}$ s.t. P *fair* o but P *m̸ust* o.

Proof. Consider $P ::= (\nu a)(\bar{a} \mid !a.\bar{a}) \mid \bar{b}$ and $o ::= b.\omega$. Since $(\nu a)(\bar{a} \mid !a.\bar{a}) \xrightarrow{\tau} (\nu a)(\bar{a} \mid !a.\bar{a}) \xrightarrow{\tau} \ldots$ (we omit 0 term by convenience), there is an unsuccessful max computation from $P \mid o$, i.e. P *m̸ust* o. However, P *fair* o, since every max computation from $P \mid o$

$$\mathcal{C} ::= P \mid o = T_0 \xrightarrow{\tau} T_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} T_i \xrightarrow{\tau} \ldots$$

is such that either $\forall i \geq 0, T_i = (\nu a)(\bar{a} \mid !a.\bar{a}) \mid \bar{b} \mid b.\omega$ or $\exists j \geq 1$ such that $T_j = (\nu a)(\bar{a} \mid !a.\bar{a}) \mid \omega \xrightarrow{\omega}$ and $\forall i \in [0..(j-1)], T_i = (\nu a)(\bar{a} \mid !a.\bar{a}) \mid \bar{b} \mid b.\omega$ and $T_i \stackrel{\varepsilon}{\Longrightarrow} T_j$. □

9.3 Weak-fair must and strong-fair must testing semantics

Theorem 6.2 For every $E \in \mathcal{P}^e$ and $\rho \in \mathcal{O}^e$, then

E *wfmust* ρ implies E *sfmust* ρ, but

there is $E \in \mathcal{P}^e$ and $\rho \in \mathcal{O}^e$, such that E *sfmust* ρ and E *wf̸must* ρ.

Proof. Consider the first item. Suppose, by contradiction, that there exists a strong-fair computation

$$\mathcal{C} ::= E \mid \rho = S_0 \xrightarrow{\tau} S_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S_i \,[\xrightarrow{\tau} \ldots]$$

such that $\forall i \geq 0, S_i \stackrel{\omega}{\not\longrightarrow}$. Since a strong-fair computation is weak-fair too, then $\mathcal{C}$ is weak-fair. It follows that E *wf̸must* ρ, contradicting the hypothesis.

Consider the second item. Consider again $E = !_{v_1^0} a \mid Q_2(v_{2,3}^0)$ and $\rho = a_{v_4^0}.\omega$, where $Q_2(v_{2,3}^j) := (\nu b)(\bar{b}_{v_2^j} \mid !_{v_3^j} b.(\bar{a} \mid \bar{b}))$.

Notice that the computation where $v_4^0 \notin Lp(S_0), v_4^0 \in Lp(S_1), v_4^0 \notin Lp(S_2), v_4^0 \in Lp(S_3), .., v_4^0 \notin Lp(S_i), v_4^0 \in Lp(S_{i+1}),\ v_4^0 \notin Lp(S_{i+2}), ..$ and so on, is unsuccessful, since v_4^0 loses its liveness without being performed: in such a case $\forall i \geq 0, S_i \stackrel{\omega}{\not\longrightarrow}$. It follows that E *wf̸must* ρ.

To prove E *sfmust* ρ, it suffices to notice that $\forall v_{2,3}^i \in (\{0,1\}^* \times \mathbb{N})$,

a. $Q_2(v^i_{2,3}) \xrightarrow{\tau} \bar{a}_{v_5^{i+1}} \mid Q_2(v^{i+1}_{2,3})$; that is $Q_2(v^i_{2,3})$ can perform infinite sequences of τ steps, becoming itself (equipped by new labels) in parallel with a component $\bar{a}_{v_5^{i+1}}$;

b. $Q_2(v^i_{2,3})$ can not synchronize with any parallel component;

c. for every maximal computation from $E \mid \rho$

$$\mathcal{C}' ::= E \mid \rho = S_0 \xrightarrow{\tau} S_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S_i \,[\xrightarrow{\tau} \ldots]$$

there always exists $S_1 = !_{v_1^0} a \mid \bar{a}_{v_5^1} \mid Q_2(v^1_{2,3}) \mid a_{v_4^0}.\omega$;

d. $v_4^0 \notin Lp(S_0)$, $v_4^0 \in Lp(S_1)$ and $v_4^0 \in Lp(S_i)$ for every S_i in $\mathcal{C}$ where there exists a $\bar{a}_{v_5^i}$ component in parallel.

By $Q_2(v^i_{2,3})$ properties, there exist infinite indexes $i, j, \ldots$ such that an output $\bar{a}_{v_5^k}$ is available in $S_i, S_j, \ldots$; it follows that v_4^0 can be live infinitely often. But this is not possible if $\mathcal{C}'$ is a strong-fair computation: in fact, by definition, v_4^0 will lose its liveness forever, i.e. v_4^0 will be performed. In such a case there will be $j \geq 0$ in $\mathcal{C}'$ such that $S_j \xrightarrow{\omega}$. □

9.4 Weak-fair must, strong-fair must and must testing semantics

The following propositions prove item (i) of Theorem 6.3.

Proposition 9.10 Let $E \in \mathcal{P}^e$, $\rho \in \mathcal{O}^e$. Then $Unl(E)$ *must* $Unl(\rho)$ implies E *wfmust* ρ.

Proof. By contradiction, suppose there is a weak-fair computation from $E \mid \rho$

$$\mathcal{C} ::= E \mid \rho = S_0 \xrightarrow{\tau} S_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S_i \,[\xrightarrow{\tau} \ldots]$$

such that $\forall i \geq 0$, $S_i \not\xrightarrow{\omega}$. Then there exists the following maximal computation

$$\mathcal{C}' ::= Unl(E \mid \rho) = Unl(S_0) \xrightarrow{\tau} Unl(S_1) \xrightarrow{\tau} \ldots \xrightarrow{\tau} Unl(S_i) \,[\xrightarrow{\tau} \ldots]$$

where $\forall i \geq 0, Unl(S_i) \not\xrightarrow{\omega}$, i.e. $Unl(E)$ *m/ust* $Unl(\rho)$. □

Proposition 9.11 There exist $E \in \mathcal{P}^e$ and $\rho \in \mathcal{O}^e$ such that E *wfmust* ρ but $Unl(E)$ *m/ust* $Unl(\rho)$.

Proof. Consider $E ::= (\nu a)(\bar{a}_{v_1^0} \mid !_{v_2^0} a.\bar{a}) \mid \bar{b}_{v_3^0}$ and $\rho ::= b_{v_4^0}.\omega$. We omit 0 terms by convenience. Notice that E *wfmust* ρ, since in every weak-fair computation from $E \mid \rho$

$$\mathcal{C} ::= E \mid \rho = S_0 \xrightarrow{\tau} S_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S_i \xrightarrow{\tau} \ldots$$

there has to exist $j \geq 1$ such that $S_j = (\nu a)(\bar{a}_{v_1^j} \mid !_{v_2^j} a.\bar{a}) \mid \omega \xrightarrow{\omega}$ and $\forall i \in [0..(j-1)], S_i = (\nu a)(\bar{a}_{v_1^i} \mid !_{v_2^i} a.\bar{a}) \mid \bar{b}_{v_3^i} \mid b_{v_4^0}.\omega$. It follows by the fact that $\forall i \in [0..(j-1)], v_4^0 \in Lp(S_i)$ and there has to exist $j \geq i$ such that $v_4^0 \notin Lp(S_j)$. It is possible only in the case $b_{v_4^0}.\omega$ synchronizes with $\bar{b}_{v_3^{j-1}}$ in S_{j-1}. However, $Unl(E)$ *m/ust* $Unl(\rho)$ (see Proposition 9.9). □

The following corollary proves item (ii) of Theorem 6.3.

Corollary 9.12 Let $E \in \mathcal{P}^e$, $\rho \in \mathcal{O}^e$. Then $Unl(E)$ *must* $Unl(\rho)$ implies E *sfmust* ρ, but not the viceversa.

Proof. The positive result follows by Proposition 9.10 and by Theorem 6.2. The negative result follows by Proposition 9.11 and by Theorem 6.2 □

9.5 Weak-fair must, strong-fair must and fair testing semantics

The following propositions prove item (ii) of Theorem 6.4. We give a preliminary lemma for proving Proposition 9.14.

Lemma 9.13 Let $S \in \mathcal{E}^e$ and $S = S_0 \xrightarrow{\tau} S_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S_i \,[\xrightarrow{\tau} \ldots]$ be a strong-fair computation from S. If $\exists S'_0, S'_1, S'_2, \ldots, S'_n \in \mathcal{E}^e$ such that

$$S' = S'_0 \xrightarrow{\tau} S'_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S'_n = S, \text{ then}$$
$$S' \xrightarrow{\tau} S'_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S'_n \xrightarrow{\tau} S_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S_i \,[\xrightarrow{\tau} \ldots]$$

is a strong-fair computation from S'.

Proof. Consider $\mathcal{C} ::= S' \xrightarrow{\tau} S'_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S'_n \xrightarrow{\tau} S'_{n+1} \xrightarrow{\tau} \ldots \xrightarrow{\tau} S'_{n+i} \,[\xrightarrow{\tau} \ldots]$, where $\forall j \geq 0, S'_{n+j} ::= S_j$. Obviously $\mathcal{C}$ is a maximal computation from S'. To prove that $\mathcal{C}$ is also strong-fair, it suffices to prove that $\forall \langle s, n\rangle \in (\{0,1\}^* \times \mathbb{N})\ \exists h \geq 0$ such that $\forall k \geq h, \langle s, n\rangle \notin Lp(S'_k)$. Since $S'_n \xrightarrow{\tau} S'_{n+1} \xrightarrow{\tau} \ldots \xrightarrow{\tau} S'_{n+i} \,[\xrightarrow{\tau} \ldots]$ is a strong fair computation from S'_n, then $\forall \langle s, n\rangle \in (\{0,1\}^* \times \mathbb{N})\ \exists h \geq n$ such that $\forall k \geq h, \langle s, n\rangle \notin Lp(S'_k)$. Since $n \geq 0$, $\forall \langle s, n\rangle \in (\{0,1\}^* \times \mathbb{N})\ \exists h \geq 0$ such that $\forall k \geq h, \langle s, n\rangle \notin Lp(S'_k)$. It follows that $\mathcal{C}$ is a strong-fair computation from S'. □

Proposition 9.14 Let $E \in \mathcal{P}^e$ and $\rho \in \mathcal{O}^e$. Then E *sfmust* ρ implies $Unl(E)$ *fair* $Unl(\rho)$.

Proof. By contradiction, suppose there exists a maximal computation from $Unl(E) \,|\, Unl(\rho)$

$$\mathcal{C} ::= Unl(E) \,|\, Unl(\rho) = T_0 \xrightarrow{\tau} T_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} T_i \,[\xrightarrow{\tau} \ldots]$$

and there exists $i \geq 0$ such that $T_i \stackrel{\omega}{\not\Longrightarrow}$, i.e. $\forall T'$ such that $T_i \stackrel{\varepsilon}{\Longrightarrow} T'$, we have $T' \stackrel{\omega}{\not\longrightarrow}$. It follows that for every maximal computation from T_i

$$T_i = T'_0 \xrightarrow{\tau} T'_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} T'_j \,[\xrightarrow{\tau} \ldots]$$

$T'_j \stackrel{\omega}{\not\longrightarrow}$ for every j. Moreover, by ω's properties, $\forall j \in [0..(i-1)], T_j \stackrel{\omega}{\not\longrightarrow}$. Now, consider

$$\mathcal{C}^e ::= E \,|\, \rho = S_0 \xrightarrow{\tau} S_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S_i \,[\xrightarrow{\tau} \ldots]$$

where $\forall k \geq 0$ we have $T_k = Unl(S_k)$. Then there exists $i \geq 0$ such that $S_i \stackrel{\omega}{\not\Longrightarrow}$, i.e. $\forall S'$ such that $S_i \stackrel{\varepsilon}{\Longrightarrow} S'$, we have $S' \stackrel{\omega}{\not\longrightarrow}$. It follows that for every maximal computation from S_i

$$S_i = S'_0 \xrightarrow{\tau} S'_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S'_j \,[\xrightarrow{\tau} \ldots]$$

$S'_j \not\xrightarrow{\omega}$ for every j. Hence for every strong-fair computation from S_i, that always exists and it is trivially a maximal computation from S_i,

$$S_i = S'_0 \xrightarrow{\tau} S'_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S'_j \,[\xrightarrow{\tau} \ldots]$$

$S'_j \not\xrightarrow{\omega}$ for every j. It follows that, given a strong-fair computation from S_i

$$S_i = S''_0 \xrightarrow{\tau} S''_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S''_j \,[\xrightarrow{\tau} \ldots]$$

where $S''_j \not\xrightarrow{\omega}$ for every j, by Lemma 9.13

$$E \mid \rho = S_0 \xrightarrow{\tau} S_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S_i = S'_0 \xrightarrow{\tau} S''_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} S''_j \,[\xrightarrow{\tau} \ldots]$$

is a strong fair computation from $E \mid \rho$, and both $\forall k \in [0..(i-1)], S_k \not\xrightarrow{\omega}$ and $\forall j \geq 0, S''_j \not\xrightarrow{\omega}$. It follows that E *sf∉must* ρ, contradicting the hypothesis. □

Proposition 9.15 There exist $E \in \mathcal{P}^e$, $\rho \in \mathcal{O}^e$ such that $Unl(E)$ *fair* $Unl(\rho)$ but E *sf∉must* ρ.

Proof. By simplicity, we consider the unlabeled terms obtained from $E = \bar{c}_{v_1^0} \mid !_{v_2^0} c.Q_2$ and $\rho = b_{v_3^0}.\omega$, where Q_2 denotes $(\nu a)(\bar{a} \mid a.\bar{c} \mid a.\bar{b})$. Then we have $Unl(E)$ *fair* $Unl(\rho)$, but there exists the following maximal computation

$Unl(E \mid \rho) = \bar{c} \mid !c.Q_2 \mid \rho \xrightarrow{\tau} Q_2 \mid !c.Q_2 \mid \rho \xrightarrow{\tau} (\nu a)(a.\bar{b}) \mid \bar{c} \mid !c.Q_2 \mid \rho \xrightarrow{\tau} \ldots$
$\xrightarrow{\tau} (\nu a)(a.\bar{b}) \mid \ldots \mid (\nu a)(a.\bar{b}) \mid \bar{c} \mid !c.Q_2 \xrightarrow{\tau} \ldots$

where every term does not perform ω: ω is always prefixed in ρ and its prefix will never be consumed, since every occurrence of $\bar{b}$ is prefixed in a deadlock term $(\nu a)(a.\bar{b})$. Notice that this computation is strong fair: the prefix of omega is not performed because it is always disabled. □

The following corollary proves item (i) of Theorem 6.4.

Corollary 9.16 Let $E \in \mathcal{P}^e$ and $\rho \in \mathcal{O}^e$. Then E *wfmust* ρ implies $Unl(E)$ *fair* $Unl(\rho)$ but not the vice versa.

Proof. The positive result follows by Theorem 6.2 and by Proposition 9.14. The negative result follows by Proposition 9.15 and by Theorem 6.2. □