Information and Computation **149**, 77–98 (1999)
Article ID inco.1998.2755, available online at http://www.idealibrary.com on IDEAL®

Basic Observables for Processes*

Michele Boreale

Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza," Italy

and

Rocco De Nicola and Rosario Pugliese

Dipartimento di Sistemi e Informatica, Università di Firenze, Italy

A general approach for defining behavioral preorders over process terms as the maximal precongruences induced by basic observables is examined. Three different observables that provide information about the initial communication capabilities of processes and about the possibility that processes get engaged in divergent computations will be considered. We show that the precongruences induced by our basic observables coincide with intuitive and/or widely studied behavioral preorders. In particular, we retrieve in our setting the *must preorder* of De Nicola and Hennessy and the *fair/should preorder* introduced by Cleaveland and Natarajan and by Brinksma, Rensink, and Vogler. A new form of testing preorder, which we call *safe-must*, also emerges. The alternative characterizations we offer shed light on the differences between these preorders and on the role played in their definition by tests for divergence.

1. INTRODUCTION

In the classical theory of functional programming, the point of view is taken that executing a program expression corresponds to evaluating it. If we write $M \downarrow v$ to indicate that program M evaluates to value v, the problem of the equivalence of two programs, hence of their semantics, can be stated as follows:

> Two programs M and N are *observationally equivalent* if for every program context C such that both $C[M]$ and $C[N]$ are programs, and for every value v, we have $C[M] \downarrow v$ if and only if $C[N] \downarrow v$.

* The extended abstract of this paper has been presented at ICALP'97 and appears in LNCS 1256, at pages 482 –492. This work has been partially supported by EEC: HCM project EXPRESS, by CNR project "Specifica ad alto livello e verifica formale di sistemi digitali" and by Istituto di Elaborazione dell'Informazione CNR, Pisa.

0890-5401/99 $30.00

A similar approach, used, e.g., for the lazy lambda calculus [1], is that of defining equivalence relations in terms of reduction to normal forms. It leads to considering as equivalent any two programs that cannot be differentiated by considering the possibility of obtaining normal forms after plugging terms in any context.

In general, given a language equipped with a reduction relation, the paradigm for defining preorders (equivalences) over terms of the language can be traced back to Morris [20] and can be phrased as follows:

1. Define a set of observables (values, normal forms, ...) to which a program can evaluate by means of successive reductions.

2. Define a basic preorder over terms by stating that a term is less defined than another if it exhibits a smaller set of basic observables.

3. Consider the largest precongruence over the language induced by the basic preorder.

This paradigm has been the basis for assessing many semantics of sequential languages and is at the heart of the full abstraction problem, see, e.g. [26].

Here, we aim at taking advantage of this paradigm also for studying models of concurrent systems and their equivalences. In this case, the choice of the basic observables is less obvious. It is well known that input/output relations are not sufficient for describing the semantics of this class of systems; it would thus be limitative to use values as observables.

Also studying the evolution to some kind of normal forms under all possible contexts is not as inspective as in the case of lambda calculus. Indeed, while the interaction between a λ-term and the environment is circumscribed, that between a process and its environment is less clear. Suppose a λ-term M is plugged into a "context" N, to form an application MN; then, everywhere along a computation, we know *when* an interaction between M and N occurs, namely when M reduces to a λ-abstraction and a β-reduction takes place. Thus, observing reduction of M to a λ-abstraction is a sensible basic observable that permits understanding the overall behavior of a term. On the contrary, when considering concurrent systems, the internal evolution of single parallel components is freely intermingled with external communications. Understanding the semantics of concurrent components via their contextual behavior turns out to be much less obvious.

A first attempt at approaching the problem of process equivalences along the mentioned lines is described in [19]. Milner and Sangiorgi defined a new equivalence for CCS [18] based on *barbed bisimilarity*. This relation represents a uniform basis for defining sensible process equivalences for different languages, as it only relies on a reduction relation and an observation predicate that detects the communication capability at a given channel. Informally, two processes are considered as barbed equivalent if they have the same communication capabilities, and this property is preserved by internal reduction. The latter requires a coinductive definition. Milner and Sangiorgi showed that CCS and π-calculus context closures of barbed bisimilarity lead to alternative characterizations of bisimulation congruences.

It can be said that in [19] the classical approach, à la Morris outlined above, is not followed to the letter. Indeed, the basic observables are very simple, but the basic equivalence heavily relies on coinduction. In this paper we consider the impact of a simpler observation machinery that only relies on contexts and basic observables, avoiding the use of coinductive tests.

Like Milner and Sangiorgi, we are interested in testing for the communication capabilities of systems, but we shall look for *guaranteed* ones. When one is willing to infer the interactive behavior of a system from its "isolated" behavior, the knowledge of the system's *possibility* of accepting communications along specific channels is not sufficient. Indeed, considering just the possibility of communication and closing with respect to all contexts would lead to trace semantics (see, e.g., [8]) that totally ignores possible deadlocks and other liveness properties. Due to the inherent nondeterminism of concurrent computations, to get more inspective semantics it is necessary to know whether communications are guaranteed.

Moreover, we shall be interested in the risk a system has of getting involved in an infinite sequence of internal communications (*divergence*), because this could lead to ignoring all subsequent external stimuli. Finally, when considering divergence we find it interesting also to detect those external communications that can lead a process to a divergent state.

These considerations lead us to introducing three basic observables:

$P!\ell$ (P *guarantees* ℓ) asserts that, by internal actions, P can only reach states from which action ℓ can be eventually (after a sequence of internal actions) performed;

$P\downarrow$ (P *converges*) asserts that P cannot get involved in an infinite sequence of internal actions;

$P\downarrow\ell$ (P *converges along* ℓ) asserts that P converges and does so also after performing ℓ.

For processes equipped with a finite reduction relation, these observables are obviously decidable; but, in general, they are not. This is somehow expected whenever the base language is Turing powerful.

As base language, we shall consider a simple variant of CCS, named Tau-less CCS (*TCCS*, [10]), that replaces the operators for internal transitions and for choice with an operator for internal choice and an operator for purely external choice[1]. We have chosen TCCS for the sake of simplicity and for avoiding the well known congruence problem that arises in the presence of silent transitions and choice. All of our results are, however, easily extended to CCS and to other languages whose operational semantics enjoys some mild conditions (see the final section).

The three predicates described above naturally induce five contextual preorders, that are listed in Table 1 (on the left of $\Leftrightarrow$). There we represent a contextual preorder using the notation ${}_{s_1}\preccurlyeq^{c}_{s_2}$, where s_1 (if present) refers to the used convergence

[1] These choice operators were originally introduced by Hoare, see e.g., [15]; their operational semantics was described in [25].

TABLE 1

Contextual Preorder Characterizations

		Convergence requirements		
		No. req.	Convergence: $\downarrow$	Conv. after ℓ: $\downarrow\ell$
Communication requirements	No req.	$\mathcal{U}$	$\downarrow\preccurlyeq^{c} \Leftrightarrow \sqsubseteq_{CT}$	$\downarrow_{\mathscr{L}}\preccurlyeq^{c} \Leftrightarrow \sqsubseteq_{CT}$
	Guarantee ℓ: $!\ell$	$\preccurlyeq^{c}_{\mathscr{L}} \Leftrightarrow \sqsubseteq^{c}_{FS}$	$\downarrow\preccurlyeq^{c}_{\mathscr{L}} \Leftrightarrow \sqsubseteq_{M}$	$\downarrow_{\mathscr{L}}\preccurlyeq^{c}_{\mathscr{L}} \Leftrightarrow \sqsubseteq_{SM}$

predicate, and s_2 (if present) refers to the guarantees one. The universal relation is denoted by $\mathcal{U}$.

The main results of this paper are five full abstraction theorems that make it manifest that our contextual preorders do coincide with well-known and/or intuitive behavioral preorders over processes studied in the literature. More specifically, we will show that:

- $\preccurlyeq^{c}_{\mathscr{L}}$, the contextual preorder induced by $!\ell$, coincides with $\sqsubseteq^{c}_{FS}$, the maximal precongruence included in the *fair/should* preorder of [21] and [4].
- $\downarrow\preccurlyeq^{c}$ and $\downarrow_{\mathscr{L}}\preccurlyeq^{c}$, the contextual preorders induced by $\downarrow$ and $\downarrow\ell$, both coincide with $\sqsubseteq_{CT}$, the preorder given by reverse inclusion of *convergent traces*. This is the maximal refinement of trace semantics [8] that respects divergence.

Together with the impact of the three observables used in isolation we will also study the result of using them in pairs and shall show that:

- $\downarrow\preccurlyeq^{c}_{\mathscr{L}}$, the contextual preorder induced by $\downarrow$ and $!\ell$, coincides with $\sqsubseteq_{M}$, the original *must* preorder of [9, 13];
- $\downarrow_{\mathscr{L}}\preccurlyeq^{c}_{\mathscr{L}}$, the contextual preorder induced by $\downarrow\ell$ and $!\ell$, gives rise to a *new* preorder, the *safe-must* preorder $\sqsubseteq_{SM}$, which is also supported by a very intuitive testing scenario.

Table 1 provides a summary of the mentioned results.

The safe-must preorder has a direct characterization in terms of computations from pairs of observer and process: a computation is successful if a success state is reached *strictly before* a "catastrophic" (divergent) one (this explains the adjective "safe"). This condition is stronger than the one introduced by De Nicola and Hennessy [9] and is very closely related to the definition of Olderog's readiness semantics in [24].

The rest of the paper is organized as follows. In Section 2, we briefly recall syntax and transitional semantics of TCCS. In Section 3, we introduce the relevant notions of the observational semantics for TCCS. Moreover, we report some alternative characterizations of the testing preorders that will be useful in later proofs. In Section 4, we present the full abstraction theorems that relate our contextual preorders to the preorder given by the reverse inclusion of convergent traces, to the fair/should preorder, to the must preorder and to the safe-must preorder. Section 5 is devoted to studying the relationships among the different preorders we have considered, and to further investigating the safe-must preorder.

Section 6 contains a brief discussion on extensions and future work. The final section contains some concluding remarks and comments on related work.

2. TAU-LESS CCS: TCCS

In this section, we briefly present the syntax and the operational semantics of TCCS, (τ-less CCS [10, 13]). As mentioned in the introduction, we have preferred TCCS to CCS because the former allows us to avoid the "congruence problems" that arise when the CCS choice operator $(+)$ is used and silent actions are abstracted away. However, the very same results can be obtained by using CCS and its must precongruence. This can be obtained from the must preorder by imposing that whenever the "better" process can perform a silent move, so can the other [9].

We let

- $\mathcal{N}$, ranged over by a, b, ..., be an infinite set of *names* and $\bar{\mathcal{N}} = \{\bar{a} \mid a \in \mathcal{N}\}$, ranged over by $\bar{a}$, $\bar{b}$, ..., be the set of *conames*. $\mathcal{N}$ and $\bar{\mathcal{N}}$ are disjoint and are in bijection via the *complementation* function $(\bar{\cdot})$; we define: $\overline{(\bar{a})} = a$;
- $\mathcal{L} = \mathcal{N} \cup \bar{\mathcal{N}}$, ranged over by ℓ, ℓ', ..., be the set of *labels*; we shall use L, K, ..., to range over subsets of $\mathcal{L}$ and we define $\bar{L} = \{\bar{\ell} \mid \ell \in L\}$;
- $\mathcal{X}$, ranged over by X, Y, ..., be a countable set of *process variables*.

DEFINITION 2.1. The set of *TCCS terms* is generated by the grammar

$$E ::= \mathbf{0} \mid \Omega \mid \ell.E \mid E[]F \mid E \oplus F \mid E \mid F \mid E\backslash L \mid E\{f\} \mid X \mid recX.E,$$

where $f: \mathcal{L} \to \mathcal{L}$, called *relabeling function*, is such that $\{\ell \mid f(\ell) \neq \ell\}$ is finite, $f(a) \in \mathcal{N}$, and $f(\bar{\ell}) = \overline{f(\ell)}$. We let $\mathcal{P}roc$, ranged over by P, Q, etc., denote the set of *closed* terms or *processes* (i.e., those terms where every occurrence of any agent variable X lies within the scope of some $recX.\_$ operator).

The language has two basic processes ($\mathbf{0}$ and Ω) and a number of operators for building up terms from existing ones. The intuitive meaning of TCCS terms is:

— $\mathbf{0}$ (*inaction*) cannot perform any action;

— Ω (*divergence*) may only compute internally;

— $\ell.E$ (*action prefix*) executes action ℓ and then behaves like E;

— $E[]F$ (*external choice*) behaves either like E or like F and the choice is controlled by the environment;

— $E \oplus F$ (*internal choice*) may autonomously decide to behave either like E or like F;

— $E \mid F$ (*parallel composition*) denotes the concurrent execution of E and F;

— $E\backslash L$ (*restriction*) behaves like E except that it cannot execute actions in L;

— $E\{f\}$ (*relabeling*) behaves like E except that its actions are renamed by f;

— $recX.E$ (*recursive definition*) has the same meaning as the term defined by the equation $X = E$ and is used for describing recursive behaviors.

In the following, we often shall write ℓ instead of $\ell.\mathbf{0}$. We write $\_\{\ell'_1/\ell_1, ..., \ell'_n/\ell_n\}$ for the relabeling operator $\_\{f\}$ where $f(\ell)=\ell'_i$ if $\ell=\ell_i$, $i\in\{1, ..., n\}$, and $f(\ell)=\ell$ otherwise. As usual, we write $E[E_1/X_1, ..., E_n/X_n]$ for the term obtained by simultaneously substituting each occurrence of X_i in E with E_i (with renaming of bound process variables possibly involved). We use the notation $\sum_{i\in\{1, ..., n\}} E_i$ as a shorthand of $E_1[]\cdots[]E_n$ (the order in which the operands E_i are arranged is unimportant, as [] is associative and commutative in every semantics considered in the paper); when $n=0$, this term will by convention indicate $\mathbf{0}$. Similarly, the notation $\bigoplus_{i\in\{1, ..., n\}} E_i$ is used as a shorthand of $E_1\oplus\cdots\oplus E_n$ (also $\oplus$ is associative and commutative in every semantics considered in the paper).

The structural operational semantics of a TCCS term is defined via the two transition relations $\xrightarrow{\ell}$ (visible actions) and $\rightarrowtail$ (internal actions) induced by the inference rules in Tables 2 and 3, respectively.

As usual, we use $\Rightarrow$ or $\stackrel{\varepsilon}{\Longrightarrow}$ to denote the reflexive and transitive closure of $\rightarrowtail$ and use $\stackrel{s}{\Longrightarrow}$, with $s\in\mathscr{L}^+$, for $\Rightarrow\xrightarrow{\ell}\stackrel{s'}{\Longrightarrow}$ when $s=\ell s'$. Moreover, we write $P\stackrel{s}{\Longrightarrow}$ if there exists P' such that $P\stackrel{s}{\Longrightarrow}P'$ ($P\xrightarrow{l}$ and $P\rightarrowtail$ will be used similarly). We will call *sort of* P the set $sort(P)=\{\ell\in\mathscr{L}\mid P\stackrel{s\ell}{\Longrightarrow}$ for some $s\in\mathscr{L}^*\}$, *successors of* P the set $S(P)=\{\ell\in\mathscr{L}\mid P\Rightarrow\ell\}$, and *language* generated by P the set $L(P)=\{s\in\mathscr{L}^*\mid P\stackrel{s}{\Longrightarrow}\}$. Note that, since we only consider finite relabelling operators, every TCCS process has a finite sort.

We will write P^n to denote the nth *finite syntactical approximant* of P, obtained by first unfolding n times all the recursive subterms of P, and then replacing the recursive subterms with Ω (see, e.g., [13]).

DEFINITION 2.2. A *context* is a TCCS term C with one free occurrence of a process variable, usually denoted by $\_$. If C is a context, we write $C[P]$ instead of $C[P/\_]$.

The context *closure* $\mathscr{R}^c$ of a given binary relation $\mathscr{R}$ over processes, is defined as:

$$P\mathscr{R}^cQ \text{ if and only if for each context } C: C[P]\,\mathscr{R}C[Q].$$

$\mathscr{R}^c$ enjoys two important properties:

(a) $(\mathscr{R}^c)^c=\mathscr{R}^c$,

(b) $\mathscr{R}\subseteq\mathscr{R}'$ implies $\mathscr{R}^c\subseteq\mathscr{R}'^c$.

In the following, we will write $\not\mathscr{R}$ for the complement of $\mathscr{R}$.

TABLE 2

SOS Rules for Visible Actions (Symmetric of Rules AR4 and AR5 Omitted)

AR1	$\ell.P\xrightarrow{\ell}P$		
AR2	$\dfrac{P\xrightarrow{\ell}P'}{P\{f\}\xrightarrow{f(\ell)}P'\{f\}}$	AR3 $\dfrac{P\xrightarrow{\ell}P'}{P\backslash L\xrightarrow{\ell}P'\backslash L}$	if $\ell\notin L\cup\overline{L}$
AR4	$\dfrac{P\xrightarrow{\ell}P'}{P[]Q\xrightarrow{\ell}P'}$	AR5 $\dfrac{P\xrightarrow{\ell}P'}{P\mid Q\xrightarrow{\ell}P'\mid Q}$	

TABLE 3

SOS Rules for Internal Actions (Symmetric of Rules IR5, IR6, and IR7 Omitted)

IR1	$\Omega \rightarrowtail \Omega$	IR2	$recX.E \rightarrowtail E[recX.E/X]$
IR3	$\dfrac{P \rightarrowtail P'}{P\{f\} \rightarrowtail P'\{f\}}$	IR4	$\dfrac{P \rightarrowtail P'}{P\backslash L \rightarrowtail P'\backslash L}$
IR5	$P \oplus Q \rightarrowtail P$	IR6	$\dfrac{P \rightarrowtail P'}{P \,[\!]\, Q \rightarrowtail P' \,[\!]\, Q}$
IR7	$\dfrac{P \rightarrowtail P'}{P \mid Q \rightarrowtail P' \mid Q}$	IR8	$\dfrac{P \xrightarrow{\ell} P', \quad Q \xrightarrow{\bar{\ell}} Q}{P \mid Q \rightarrowtail P' \mid Q'}$

3. OBSERVATIONAL SEMANTICS

In this section, we introduce a number of observational semantics for TCCS; we follow two approaches. The first one relies on three basic observables (i.e., predicates over processes) which give rise to five significant preorders; the corresponding precongruences are obtained by closing these preorders over all possible TCCS contexts and determine five semantics for the language. The second approach relies on the classical *testing* scenario of [9, 13] or variants of it. We shall also introduce alternative characterizations of the obtained testing preorders that will be useful in later proofs.

3.1. Basic Observables and Observation Preorders

DEFINITION 3.1. Let P be a process and $\ell \in \mathscr{L}$. We define three basic *observation predicates* over processes as follows:

- $P!\ell$ (P *guarantees* ℓ) if and only if for each P', $P \Rightarrow P'$ implies $P' \stackrel{\ell}{\Longrightarrow}$;
- $P\downarrow$ (P *converges*) if and only if there is no infinite sequence of internal transitions $P \rightarrowtail P_1 \rightarrowtail \cdots$ starting from P;
- $P\downarrow\ell$ (P *converges along* ℓ) if and only if $P\downarrow$ and, for each P', $P \stackrel{\ell}{\Longrightarrow} P'$ implies $P'\downarrow$.

The above predicates can be combined in five sensible ways and used to define five basic *observation preorders* over processes, as stated in the following definition.

DEFINITION 3.2. Let P and Q be processes.

- $P \preccurlyeq_{\downarrow} Q$ if and only if $P\downarrow$ implies $Q\downarrow$;
- $P \preccurlyeq_{\downarrow\mathscr{L}} Q$ if and only if for each $\ell \in \mathscr{L}$: $P\downarrow\ell$ implies $Q\downarrow\ell$;
- $P \preccurlyeq_{\mathscr{L}} Q$ if and only if for each $\ell \in \mathscr{L}$: $P!\ell$ implies $Q!\ell$;
- $P \preccurlyeq_{\downarrow\mathscr{L}} Q$ if and only if for each $\ell \in \mathscr{L}$: ($P\downarrow$ and $P!\ell$) implies ($Q\downarrow$ and $Q!\ell$);
- $P \preccurlyeq_{\downarrow\mathscr{L}\mathscr{L}} Q$ if and only if for each $\ell \in \mathscr{L}$: ($P\downarrow\ell$ and $P!\ell$) implies ($Q\downarrow\ell$ and $Q!\ell$).

Of course, the basic observation preorders are very coarse. More refined relations can be obtained by closing the above preorders under all TCCS contexts. For each basic observation preorder, say $\preccurlyeq$, the *contextual preorder* generated by $\preccurlyeq$ is defined as its closure $\preccurlyeq^c$.

3.2. Testing Preorders and Alternative Characterizations

Like in the original theory of testing [9, 13], we have that:

— *observers*, ranged over by O, O', ..., are processes capable of possibly performing an additional distinct "success" action $w \notin \mathscr{L}$;

- *computations* from $P \mid O$ are sequences of internal transitions

$$P \mid O\ (= P_0 \mid O_0) \rightarrowtail P_1 \mid O_1 \rightarrowtail \cdots,$$

which are either infinite or such that there exists $k \geqslant 0$ with $P_k \mid O_k \not\rightarrowtail$.

DEFINITION 3.3. Let P be a process and O be an observer.

1. $P \underline{must}_M O$ if for each computation from $P \mid O$, say $P \mid O \rightarrowtail P_1 \mid O_1 \rightarrowtail \cdots$, there is some $i \geqslant 0$ such that $O_i \xrightarrow{w}$.
2. $P \underline{must}_{SM} O$ if for each computation from $P \mid O$, say $P \mid O \rightarrowtail P_1 \mid O_1 \rightarrowtail \cdots$, there is some $i \geqslant 0$ such that $O_i \xrightarrow{w}$ and $P_i \downarrow$.
3. $P \underline{must}_{FS} O$ if for each computation from $P \mid O$, say $P \mid O \rightarrowtail P_1 \mid O_1 \rightarrowtail \cdots$, it holds that $P_i \mid O_i \stackrel{w}{\Longrightarrow}$ for each $i \geqslant 0$.

The first definition of successful computation given above is exactly that of [9]. The second one considers successful only those computations that can report a success *strictly before* the observed process diverges. The third definition, which is essentially borrowed from [4], totally ignores the issue of divergence[2]. These three notions allow us to define three preorders: the first one ($\sqsubseteq_M$) is the original *must* preorder of [9, 13], the second one ($\sqsubseteq_{SM}$) is the new *safe-must* preorder and the third one ($\sqsubseteq_{FS}$) is the (reverse of the) *fair/should* preorder of [21] and [4].

DEFINITION 3.4. Let P and Q be processes and $X \in \{M, SM, FS\}$ then

$$P \sqsubseteq_X Q \text{ if and only if for every observer } O : P \underline{must}_X O \text{ implies } Q \underline{must}_X O.$$

$\sqsubseteq_M$, $\sqsubseteq_{SM}$, and $\sqsubseteq_{FS}$ are called *must*, *safe-must* and *fair/should* preorder, respectively.

Given a testing preorder $\sqsubseteq_X$, $X \in \{M, SM, FS\}$, the corresponding TCCS *precongruence* is defined as its closure $\sqsubseteq_X^c$ and the corresponding equivalence, $\simeq_X$, is defined as $\simeq_X = \sqsubseteq_X \cap (\sqsubseteq_X)^{-1}$.

We introduce below alternative characterizations of the preorders *must* and *safe-must*. They support simpler methods for proving (or disproving) that two processes are behaviorally related. For presenting the new characterizations, we need some additional notation.

[2] Although not explicitly present in the first definition, divergence is taken into account there: a divergent process would contain an unsuccessful computation.

DEFINITION 3.5. Let $s \in \mathscr{L}^*$, $B \subseteq_{\text{fin}} \mathscr{L}$ and $\mathscr{Q}$ be a set of processes.

- The *convergence* predicate, $\downarrow s$, is defined inductively as follows:
  - $P \downarrow \varepsilon$ if $P \downarrow$;
  - $P \downarrow \ell s'$ if $P \downarrow \varepsilon$ and for each P': $P \stackrel{\ell}{\Longrightarrow} P'$ implies $P' \downarrow s'$.

We write $P \uparrow s$ ($P \uparrow \varepsilon$ or $P \uparrow$) if $P \downarrow s$ ($P \downarrow \varepsilon$) does not hold.

- $(P \underline{after}\ s)$ is the set of processes $\{P' \mid P \stackrel{s}{\Longrightarrow} P'\}$.
- $P \downarrow B$ means $P \downarrow \ell$ for each $\ell \in B$.
- $\mathscr{Q} \downarrow B$ means $P \downarrow B$ for each $P \in \mathscr{Q}$.
- $P\ \underline{accepts}_M\ B$ means that there exists $\ell \in B$ such that $P \stackrel{\ell}{\Longrightarrow}$.
- $\mathscr{Q}\ \underline{accepts}_M\ B$ means $P\ \underline{accepts}_M\ B$ for each $P \in \mathscr{Q}$.
- $\mathscr{Q}\ \underline{accepts}_{SM}\ B$ means $\mathscr{Q} \downarrow B$ and $\mathscr{Q}\ \underline{accepts}_M\ B$.

DEFINITION 3.6. Let $X \in \{M, SM\}$. For processes P and Q, we write $P \ll_X Q$ if for each $s \in \mathscr{L}^*$ such that $P \downarrow s$, it holds that:

(a) $Q \downarrow s$, and

(b) for every $B \subseteq_{\text{fin}} \mathscr{L}$: $(P \underline{after}\ s)\ \underline{accepts}_X\ B$ implies $(Q \underline{after}\ s) \underline{accepts}_X B$.

The proof of the following result is reported in [9, 13].

THEOREM 3.7. *For all processes P and Q, $P \sqsubseteq_{\approx M} Q$ if and only if $P \ll_M Q$.*

THEOREM 3.8. *For all processes P and Q, $P \sqsubseteq_{\approx SM} Q$ if and only if $P \ll_{SM} Q$.*

Proof. Very similar to that of Theorem 3.7, reported, e.g., in [13]. Below, we outline the proof. We provide additional details for those points that differ from [13].

Part ($\Leftarrow$). Let O be any observer and suppose that $Q\ \underline{must}_{SM}\ O$: we show that $P\ \underline{must}_{SM}\ O$ as well. Let γ be any nonsuccessful computation, say $Q \mid O = (Q_0 \mid O_0) \rightarrowtail Q_1 \mid O_1 \rightarrowtail \cdots$, for $Q \mid O$. The case when γ is infinite is dealt with exactly like in [13] (it requires König's lemma for reducing to the finite case). If γ is finite, then there are k and s such that $Q_k \mid O_k \not\rightarrowtail$, and $Q \stackrel{s}{\Longrightarrow} Q_k$ and $O \stackrel{\bar{s}}{\Longrightarrow} O_k$. Furthermore, for each i, $0 \leqslant i \leqslant k$, such that $O_i \stackrel{w}{\longrightarrow}$, there is $j \leqslant i$ with $Q_j \uparrow$. Now, if $P \uparrow s$ a nonsuccessful computation for $P \mid O$ can be easily constructed. If $P \downarrow s$ then $Q \downarrow s$, by definition of $\ll_{SM}$: this implies that $O_i \not\stackrel{w}{\longrightarrow}$ for $0 \leqslant i \leqslant k$. Now, let $B \stackrel{\text{def}}{=} \overline{S(O_k)}$. Since $Q_k \mid O_k \not\rightarrowtail$, we deduce that $(Q \underline{after}\ s)\ \underline{accepts}_{SM}\ B$ does not hold. From this and $P \downarrow s$ we deduce that also $(P \underline{after}\ s)\ \underline{accepts}_{SM}\ B$ does not hold. That is, there is P' such that $P \stackrel{s}{\Longrightarrow} P'$ and either $S(P') \cap B = \varnothing$ or $P' \stackrel{\ell}{\Longrightarrow} P''$ and $P'' \uparrow$, for some $\ell \in B$: in both cases, a nonsuccessful computation for $P \mid O$ can be easily constructed.

The proof of part ($\Rightarrow$), similarly to the proof of Theorem 3.7 in [13], relies on two sets of observers. The first kind of observers tests for convergence along s ($P \downarrow s$) and is defined inductively on s as follows: $c(\varepsilon) = w$ and $c(\ell s') = (w \oplus \mathrm{w})\,[](\bar{\ell}.c(s'))$. The second kind tests for the sets of acceptance after a sequence of

actions $((P \underline{after}\ s)\ \underline{accepts}_{SM}\ B)$, and is defined inductively on s as follows: $a(\varepsilon, B) = \sum_{\ell \in B} \bar{\ell}.w$ and $a(\ell s', B) = (w \oplus w)[](\bar{\ell}.a(s', B))$. ∎

By taking advantage of the above alternative characterizations it is easy to prove that the must and the safe-must preorders are precongruences.

THEOREM 3.9. *For all processes P and Q and $X \in \{M, SM\}$, $P \sqsubseteq_X Q$ if and only if $P \sqsubseteq_X^c Q$.*

Proof. The proof for $\sqsubseteq_M$ relies on the alternative characterization and can be found, e.g., in [13]. The proof for $\sqsubseteq_{SM}$ can be done along the same lines. In particular, to show that the preorder is preserved also by the recursive contexts, the following property is used:

for any process P and any observer O,

$P\ \underline{must}_{SM}\ O$ implies that there exists $n \geqslant 0$ such that $P^n\ \underline{must}_{SM}\ O$. ∎

The fair/should preorder $\sqsubseteq_{FS}$ was not considered above because it is not preserved by recursive contexts. This can be easily seen by considering the following counterexample. Consider the processes $P = a.b[\,]\ a.c$ and $Q = a.b$ and the context $C = recX.(\_ \mid \bar{a}\bar{b}.X)\backslash\{a, b\}$. It obviously holds that $P \sqsubseteq_{FS} Q$, but $C[P] \not\sqsubseteq_{FS} C[Q]$ (just take $O = \bar{c}.w$); hence $P \not\sqsubseteq_{FS}^c Q$.

In [5], for a language slightly different from ours, the following alternative characterization of the closure of the fair/should preorder is conjectured: $P \sqsubseteq_{FS}^c Q$ if and only if ($P \sqsubseteq_{FS} Q$ and $L(P) \subseteq L(Q)$). If the conjecture were proved we would get a simple and natural characterization of the contextual preorder $\sqsubseteq_{FS}^c$.

4. FULL ABSTRACTION RESULTS

In this section, we present the full abstraction theorems that relate our contextual preorders to the (reverse) inclusion of convergent traces preorder, the fair/should testing, the must testing and the safe-must preorders.

From now onward, we shall adopt the following convention: an action declared *fresh* in a statement is assumed different from any other name and coname there mentioned.

4.1. Convergence Predicate and Convergent Traces

In this section, we deal with the first two contextual preorders, $\downarrow \preccurlyeq^c$ and $\downarrow_{\mathscr{L}} \preccurlyeq^c$, and prove that they have the same distinguishing power and coincide with the reverse inclusion of the convergent traces preorder.

DEFINITION 4.1. For all processes P and Q, we write $P \sqsubseteq_{CT} Q$ if for each $s \in \mathscr{L}^*$ such that $P \downarrow s$, it holds that:

(a) $Q \downarrow s$, and

(b) $s \in L(Q)$ implies $s \in L(P)$.

It is easy to show that $\underset{\sim}{\sqsubset}_{CT}$ is the largest preorder included in trace semantics (reverse trace inclusion) which includes the must preorder $\underset{\sim}{\sqsubset}_{M}$. Furthermore, $\underset{\sim}{\sqsubset}_{CT}$ is a congruence, as stated by the following theorem.

THEOREM 4.2. *For all processes P and Q, $P \underset{\sim}{\sqsubset}_{CT} Q$ if and only if $P \underset{\sim}{\sqsubset}^{c}_{CT} Q$.*

Proof. Obviously we have that $\underset{\sim}{\sqsubset}^{c}_{CT}$ is included in $\underset{\sim}{\sqsubset}_{CT}$. To establish the reverse inclusion we need a case analysis on the contexts. The only difficult case is when a recursive context is used. In this case the proof relies on the following facts, whose proofs are standard:

1. for any process P and sequence $s \in \mathscr{L}^*$, $P \downarrow s$ if and only if there exists $n \geqslant 0$ such that $P^n \downarrow s$;

2. for any process P and sequence $s \in \mathscr{L}^*$, $s \in L(P)$ if and only if there exists $n \geqslant 0$ such that $s \in L(P^n)$

where P^n denotes the nth finite syntactical approximant of P. ∎

We will use some special contexts for proving relationships between the preorders. If $s \in \mathscr{L}^*$, say $s = \ell_1 \cdots \ell_n$ $(n \geqslant 0)$, we define

$$C_1^s = \_ \mid \bar{\ell}_1.\cdots.\bar{\ell}_n.\mathbf{0} \qquad \text{and} \qquad C_2^s = \_ \mid \bar{\ell}_1.\cdots.\bar{\ell}_n.\Omega.$$

The following two lemmas will be useful for proving the coincidence of the preorders $\downarrow\preccurlyeq^{c}$, $\downarrow_{\mathscr{L}}\preccurlyeq^{c}$, and $\underset{\sim}{\sqsubset}_{CT}$. The proof of the first is straightforward.

LEMMA 4.3. *For any process P and $s \in \mathscr{L}^*$, $P \downarrow s$ if and only if $C_1^s[P] \downarrow$.*

LEMMA 4.4. *Consider a process P and $s \in \mathscr{L}^*$ such that $P \downarrow s$. Then $s \in L(P)$ if and only if $C_2^s[P] \uparrow$.*

Proof. Assume that $P \downarrow s$. If $P \overset{s}{\Longrightarrow} P'$ then we can construct the derivation $C_2^s[P] \Rightarrow P' \mid \Omega$: this implies that $C_2^s[P] \uparrow$. On the contrary, suppose that $C_2^s[P] \uparrow$; then, relying on the fact that $P \downarrow s$, we can easily show by induction on s that $P \overset{s}{\Longrightarrow}$. ∎

THEOREM 4.5. *For all processes P and Q, $P \underset{\sim}{\sqsubset}_{CT} Q$ if and only if $P \downarrow\preccurlyeq^{c} Q$.*

Proof. $(\Rightarrow)$ Since $\varepsilon \in L(P)$ for any process P then, by definition, $\underset{\sim}{\sqsubset}_{CT}$ is contained in $\downarrow\preccurlyeq$ from which the result follows by closing under contexts and by relying on Theorem 4.2.

$(\Leftarrow)$ Suppose that $P \downarrow\preccurlyeq^{c} Q$ and that $P \downarrow s$. Then we have:

$$\begin{aligned} P \downarrow s \quad & \text{implies (Lemma 4.3)} \\ C_1^s[P] \downarrow \quad & \text{implies (hypothesis } P \downarrow\preccurlyeq^{c} Q \text{ with } C = C_1^s) \\ C_1^s[Q] \downarrow \quad & \text{implies (Lemma 4.3)} \\ Q \downarrow s. & \end{aligned}$$

Now we can use the fact that $P \downarrow s$ and $Q \downarrow s$ for proving that $s \in L(Q)$ implies $s \in L(P)$. Indeed, suppose that $s \in L(Q)$; then we have

$$
\begin{aligned}
s \in L(Q) \quad & \text{implies (Lemma 4.4 and } Q \downarrow s) \\
C_2^s[Q] \uparrow \quad & \text{implies (hypothesis } P \;{}_{\downarrow}\preccurlyeq^{c} Q \text{ with } C = C_2^s) \\
C_2^s[P] \uparrow \quad & \text{implies (Lemma 4.4 and } P \downarrow s) \\
s \in L(P) &
\end{aligned}
$$

which proves that $P \sqsubseteq_{CT} Q$. ∎

THEOREM 4.6. *For all processes P and Q, $P \;{}_{\downarrow\mathscr{L}}\preccurlyeq^{c} Q$ if and only if $P \;{}_{\downarrow}\preccurlyeq^{c} Q$.*

Proof. ($\Rightarrow$) We prove that ${}_{\downarrow\mathscr{L}}\preccurlyeq^{c}$ is contained in ${}_{\downarrow}\preccurlyeq$, from which the result follows by closing under contexts. Suppose that $P \;{}_{\downarrow\mathscr{L}}\preccurlyeq^{c} Q$ and that $P \downarrow$. Fix a fresh $\ell \in \mathscr{L}$ (such an ℓ exists because $sort(P)$ and $sort(Q)$ are finite). Then we have:

$$
\begin{aligned}
P \downarrow \quad & \text{implies} \\
(P \mid \ell) \downarrow \ell \quad & \text{implies (hypothesis } P \;{}_{\downarrow\mathscr{L}}\preccurlyeq^{c} Q \text{ with } C = \_ \mid \ell) \\
(Q \mid \ell) \downarrow \ell \quad & \text{implies} \\
Q \downarrow &
\end{aligned}
$$

which proves that $P \;{}_{\downarrow}\preccurlyeq Q$.

($\Leftarrow$) It can be proved like Theorem 4.5, first part of case ($\Leftarrow$) with $s = \ell$. ∎

4.2. Guarantees and Fair Testing

To prove full abstraction for fair/should, we will use the following lemma.

LEMMA 4.7. *Let P be a process and O be an observer.*

1. *P $\underline{must}_{FS}$ O if and only if $P \mid O\{\ell/w\}!\ell$, where $\ell \in \mathscr{L}$ is a fresh action;*
2. *$P!\ell$ if and only if P $\underline{must}_{FS}$ $\bar{\ell}.w$.*

Proof.

1. Observe that, as $\{w, \bar{w}\} \cap sort(P) = \varnothing$ and $\{\ell, \bar{\ell}\} \cap (sort(P, O)) = \varnothing$, the renaming $\{\ell/w\}$ does not affect the interactions between P and O, therefore

(a) $P \mid O \Rightarrow P' \mid O'$ if and only if $P \mid O\{\ell/w\} \Rightarrow P' \mid O'\{\ell/w\}$

and

(b) $P' \mid O' \overset{w}{\Longrightarrow}$ if and only if $P' \mid O'\{\ell/w\} \overset{\ell}{\Longrightarrow}$.

Now we prove that P $\underline{must}_{FS}$ O implies $P \mid O\{\ell/w\}!\ell$ (the converse can be proved similarly). Let P' and O' be such that $P \mid O\{\ell/w\} \Rightarrow P' \mid O'\{\ell/w\}$. Facts (a) and (b) above and the hypothesis imply that $P \mid O \Rightarrow P' \mid O'$ and $P' \mid O' \overset{s}{\Longrightarrow}$. Using again (a) and (b) above, but in the opposite direction, we conclude that $P' \mid O'\{\ell/w\} \overset{\ell}{\Longrightarrow}$.

2. ($\Rightarrow$) Let P' and O be such that $P \mid \bar{\ell}.w \Rightarrow P' \mid O$; we must show that $P' \mid O \overset{w}{\Longrightarrow}$. If $O = w$ then $P' \mid O \xrightarrow{w}$, which implies the thesis. If $O = \bar{\ell}.w$ then $P \Rightarrow P'$. By hypothesis, $P' \overset{\ell}{\Longrightarrow}$. This implies that $P' \mid \bar{\ell}.w \overset{w}{\Longrightarrow}$.

($\Leftarrow$) Suppose that $P \Rightarrow P'$; we must show that $P \stackrel{\ell}{\Longrightarrow}$. Now $P \mid \bar{\ell}.w \Rightarrow P' \mid \bar{\ell}.w$. By hypothesis, $P' \mid \bar{\ell}.w \stackrel{w}{\Longrightarrow}$; since $w \notin sort(P)$, then $P' \stackrel{\ell}{\Longrightarrow}$. ∎

THEOREM 4.8. *For all processes P and Q, $P \sqsubseteq^{c}_{\widetilde{FS}} Q$ if and only if $P \preccurlyeq^{c}_{\mathscr{L}} Q$.*

Proof. ($\Leftarrow$) We prove that $\preccurlyeq^{c}_{\mathscr{L}}$ is contained in $\sqsubseteq_{\widetilde{FS}}$, and the claimed result will follow by closing under contexts. Suppose that $P \preccurlyeq^{c}_{\mathscr{L}} Q$ and that $P \underline{must}_{FS} O$; let ℓ be a fresh action. We have:

$$\begin{aligned} P \underline{must}_{FS} O \quad & \text{implies (Lemma 4.7(1))} \\ P \mid O\{\ell/w\} \,!\, \ell \quad & \text{implies (hypothesis } P \preccurlyeq^{c}_{\mathscr{L}} Q, \text{ with } C = \_ \mid O\{\ell/w\}) \\ Q \mid O\{\ell/w\} !\ell \quad & \text{implies (Lemma 4.7(1))} \\ Q \underline{must}_{FS} O. & \end{aligned}$$

($\Rightarrow$) We prove that $\sqsubseteq_{\widetilde{FS}}$ is contained in $\preccurlyeq_{\mathscr{L}}$, and the claimed result will follow by closing under contexts. Suppose that $P \sqsubseteq_{\widetilde{FS}} Q$ and that $P!\ell$, for any ℓ. We have:

$$\begin{aligned} P!\ell \quad & \text{implies (Lemma 4.7(2))} \\ P \underline{must}_{FS} \bar{\ell}.w \quad & \text{implies (hypothesis P } \sqsubseteq_{\widetilde{FS}} \text{ Q)} \\ Q \underline{must}_{FS} \bar{\ell}.w \quad & \text{implies (Lemma 4.7(2))} \\ Q!\ell. & \end{aligned}$$ ∎

4.3. Guarantees plus Convergence and Must Testing

To prove full abstraction for must, we will use the following special contexts.

DEFINITION 4.9. Let $s \in \mathscr{L}^*$, say $s = \ell_1 \cdots \ell_n$ $(n \geqslant 0)$, and $B \subseteq_{\text{fin}} \mathscr{L}$. Let f^B denote a function which maps each $\ell \in B$ to one and the same fresh c. Fix a bijective correspondence among $\ell_1, ..., \ell_n$ and n fresh actions $\alpha_1, ..., \alpha_n$. We define

$$C_3^s = \_ \mid Q_3^s \qquad \text{where} \qquad Q_3^{\varepsilon} \qquad \text{and} \qquad Q_3^{\ell s'} = \bar{\ell}.Q_3^{s'}[\,]c$$

and

$$C_4^{s,B} = (\_ \mid R^s)\{f^B\} \mid Q_4^s \qquad \text{where} \qquad R^s = \bar{\ell}_1.\alpha_1.\cdots\bar{\ell}_n.\alpha_n,$$
$$Q_4^{\varepsilon} = \mathbf{0} \qquad \text{and} \qquad Q_4^{\ell_1 s'} = \bar{\alpha}_1.Q_4^{s'}[\,]c.$$

To give a better intuition of these contexts, we report in Fig. 1 a pictorial representation of processes Q_3^s and Q_4^s, for $s = \ell_1 \cdots \ell_n$.

LEMMA 4.10. *Let $s \in \mathscr{L}^*$, $B \subseteq_{\text{fin}} \mathscr{L}$, and c be a fresh action.*

(a) *$P \downarrow s$ if and only if $C_3^s[P] \downarrow$ if and only if $C_3^s[P] \downarrow c$.*

(b) *$(P \underline{after}\ s) \underline{accepts}_M B$ if and only if $C_4^{s,B}[P]!c$.*

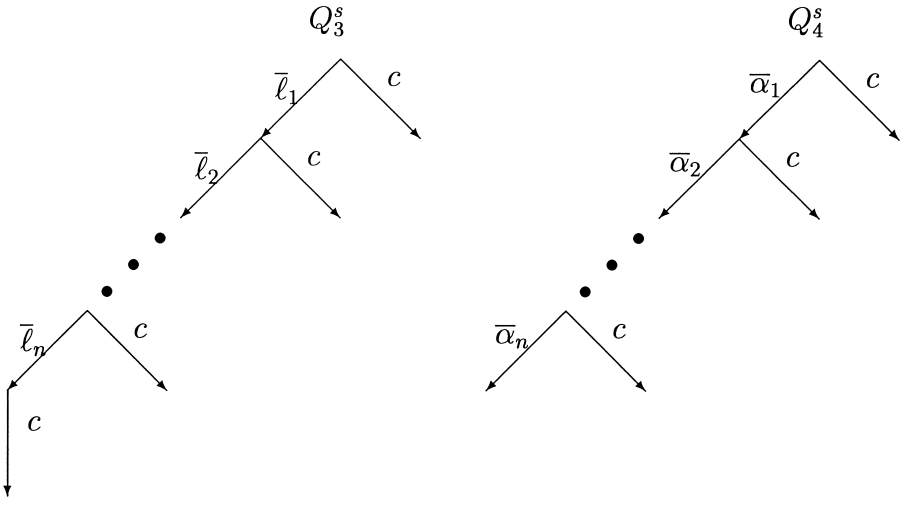

FIG. 1. Processes Q_3^s and Q_4^s used for contexts C_3^s and C_4^s.

Proof.

(a) We start with proving that $P\downarrow s$ implies $C_3^s[P]\downarrow$. The proof is by induction on s. The base case is trivial. Suppose now $s=\ell s'$, and assume by contradiction that $C_3^s[P]\uparrow$, i.e. assume there is an infinite sequence $C_3^s[P]=R_0 \rightarrowtail R_1 \rightarrowtail \cdots$. Due to the form of the context C_3^s and to the fact that $P\downarrow$, we deduce that, for some i, $R_i=C_3^{s'}[P']$, with $P \stackrel{\ell}{\Longrightarrow} P'$. Now, by hypothesis $P'\downarrow s'$, and thus by the induction hypothesis, we get $R_i=C_3^{s'}[P']\downarrow$, which is a contradiction.

Now, we prove that $C_3^s[P]\downarrow$ implies $C_3^s[P]\downarrow c$. We have just to show that $C_3^s[P] \stackrel{c}{\Longrightarrow} R$ implies $R\downarrow$. Due to the form of the context C_3^s and the fact that c is fresh, R is of the form $P'\mid \mathbf{0}$ where $P \stackrel{s'}{\Longrightarrow} P'$, for some s' prefix of s. Since $C_3^s[P]\Rightarrow P'\mid Q_3^{s''}$, with $s's''=s$, the hypothesis implies that $P'\downarrow$ which, in turn, implies $R\downarrow$.

We are left with proving that $C_3^s[P]\downarrow c$ implies $P\downarrow s$. Let s' be some prefix of s, i.e., $s=s's''$, and P' be such that $P \stackrel{s'}{\Longrightarrow} P'$. We must show that $P'\downarrow$. Since we can construct the sequence of internal moves $C_3^s[P]\Rightarrow P'\mid Q_3^{s''}$, the hypothesis $C_3^s[P]\downarrow c$ implies $P'\downarrow$.

(b) ($\Rightarrow$) Suppose that $C_4^{s,B}[P]\Rightarrow R$. We must show that $R \stackrel{c}{\Longrightarrow}$. Due to the form of the context $C_4^{s,B}$, there exist s', s'' and P' such that $s=s's''$, and $P \stackrel{s'}{\Longrightarrow} P'$ and either $R=(P'\mid \alpha_{i-1}.R^{s''})\{f^B\}\mid(\bar{\alpha}_{i-1}.Q_4^{s''}[\,]c)$ or $R=(P'\mid R^{s''})\{f^B\}\mid Q_4^{s''}$. In the first case, obviously, $R \stackrel{c}{\Longrightarrow}$. In the second case, if s' is a proper prefix of s then $Q_4^{s''} \stackrel{c}{\longrightarrow}$ otherwise there exists $\ell\in B$ such that $P' \stackrel{\ell}{\Longrightarrow}$ and then $R \stackrel{c}{\Longrightarrow}$.

($\Leftarrow$) Let $R\in(P \underline{after}\ s)$ (if $(P \underline{after}\ s)=\varnothing$, we are done). Due to the form of the context $C_4^{s,B}$, $R\in(P \underline{after}\ s)$ implies that the following sequence of internal transitions $C_4^{s,B}[P]\Rightarrow(R\mid\mathbf{0})\{f^B\}\mid\mathbf{0}$ is possible. Since, by hypothesis, $C_4^{s,B}[P]!c$ and c is fresh then there exists $\ell\in B$ such that $R \stackrel{\ell}{\Longrightarrow}$, i.e., $R \underline{accepts}_M B$, and the thesis is proved. ▮

THEOREM 4.11. *For all processes P and Q, $P \sqsubseteq_M Q$ if and only if $P \downarrow \preccurlyeq^{c}_{\mathscr{L}} Q$.*

Proof. ($\Rightarrow$) From the definition, it is easily seen that $\ll_M$ is contained in $_{\downarrow}\leqslant_{\mathscr{L}}$ (indeed $P!c$ if and only if $(P \underline{after}\ \varepsilon) \underline{accepts}_M \{c\}$). By applying Theorem 3.7, closing under contexts and recalling that $\sqsubseteq_M$ is a precongruence (Theorem 3.9), the thesis follows.

($\Leftarrow$) Here, we show that $_{\downarrow}\leqslant^{c}_{\mathscr{L}}$ is contained in $\ll_M$. This fact, Theorem 3.7, and Theorem 3.9 imply the thesis. Assume that $P\ {}_{\downarrow}\leqslant^{c}_{\mathscr{L}} Q$ and that $P\downarrow s$, for some $s\in\mathscr{L}^*$. We have to show that: (a) $Q\downarrow s$ and (b) $(P \underline{after}\ s) \underline{accepts}_M B$ implies $(Q \underline{after}\ s) \underline{accepts}_M B$, for any $B \subseteq_{\text{fin}} \mathscr{L}$.

As to part (a), from $P\downarrow s$ and Lemma 4.10(a), it follows that $C_3^s[P]\downarrow$. Obviously, for every process R, $C_3^s[R]!c$. From $C_3^s[P]\downarrow$, $C_3^s[P]!c$ and $P\ {}_{\downarrow}\leqslant^{c}_{\mathscr{L}} Q$ it follows that $C_3^s[Q]\downarrow$. By applying again Lemma 4.10(a), but in the opposite direction, we obtain $Q\downarrow s$.

As to part (b), suppose that $(P \underline{after}\ s) \underline{accepts}_M B$. This and Lemma 4.10(b) imply that $C_4^{s,B}[P]!c$. Moreover, it is easy to see that for every process R, $R\downarrow s$ implies $C_4^{s,B}[R]\downarrow$. From $C_4^{s,B}[P]\downarrow$, $C_4^{s,B}[P]!c$ and $P\ {}_{\downarrow}\leqslant^{c}_{\mathscr{L}} Q$, it follows that $C_4^{s,B}[Q]!c$. By applying again Lemma 4.10(b), but in the opposite direction, we obtain $(Q \underline{after}\ s) \underline{accepts}_M B$. ∎

4.4. Guarantees Plus Convergence and Safe-Must Testing

To prove full abstraction for safe-must, we will use another special context. Again, we assume that $c\in\mathscr{L}$ is always fresh.

DEFINITION 4.12. Let $s\in\mathscr{L}^*$, say $s=\ell_1\cdots\ell_n$ ($n\geqslant 0$), and $B\subseteq_{\text{fin}}\mathscr{L}$. We define the context

$$C_5^{s,B} = \_ \mid Q_5^{s,B} \qquad \text{where} \qquad Q_5^{\varepsilon,B} = \sum_{\ell\in B} \bar{\ell}.c \qquad \text{and} \qquad Q_5^{\ell s',B} = \bar{\ell}.Q_5^{s',B}[]\,c.$$

Again, to give a better intuition of context $C_5^{s,B}$, we report in Fig. 2 a pictorial representation of process $Q_5^{s,B}$, for $s=\ell_1\cdots\ell_n$ and $B=\{\ell'_1,\ \cdots,\ell'_m\}$.

LEMMA 4.13. *Let $s\in\mathscr{L}^*$, $B\subseteq_{\text{fin}}\mathscr{L}$ and c be a fresh action. If $P\downarrow s$ then $(P \underline{after}\ s) \underline{accepts}_{SM} B$ if and only if $(C_5^{s,B}[P]\downarrow c$ and $C_5^{s,B}[P]!c)$.*

Proof. ($\Rightarrow$) We must show that (a) $C_5^{s,B}[P]\downarrow$, (b) $C_5^{s,B}[P] \stackrel{c}{\Longrightarrow} R$ implies $R\downarrow$, and (c) $C_5^{s,B}[P]\Rightarrow R$ implies $R\stackrel{c}{\Longrightarrow}$.

The proof of (a) goes by induction on s and is similar to the proof of the first part of Lemma 4.10(a); the only difference is that now, in the base case, one relies on the fact that for each $\ell\in B$ and P' such that $P\Rightarrow P'$, we have $P'\downarrow\ell$, which is a consequence of $(P \underline{after}\ s) \underline{accepts}_{SM} B$.

As to (b), if $C_5^{s,B}[P] \stackrel{c}{\Longrightarrow} R$, due to the form of $C_5^{s,B}$ it must be $R=P'\mid \mathbf{0}$, with $P\stackrel{s'}{\Longrightarrow}P'$ for some prefix s' of $s\ell$, with $\ell\in B$. Since it must be $P\downarrow s\ell$ (from $P\downarrow s$ and $(P \underline{after}\ s) \underline{accepts}_{SM} B$), we get that $P'\downarrow$, and the claim follows.

As to (c), suppose that $C_5^{s,B}[P]\Rightarrow R$. Due to the form of $C_5^{s,B}$, it must be either $R=P'\mid c$ or $R=P'\mid Q_5^{s'',B}$ and $P\stackrel{s'}{\Longrightarrow}P'$ with $s's''=s$. In the first case, obviously $R\stackrel{c}{\longrightarrow}$. In the second case, if $s''\neq\varepsilon$, we have $Q_5^{s'',B}\stackrel{c}{\longrightarrow}$; otherwise, it is $s'=s$ and

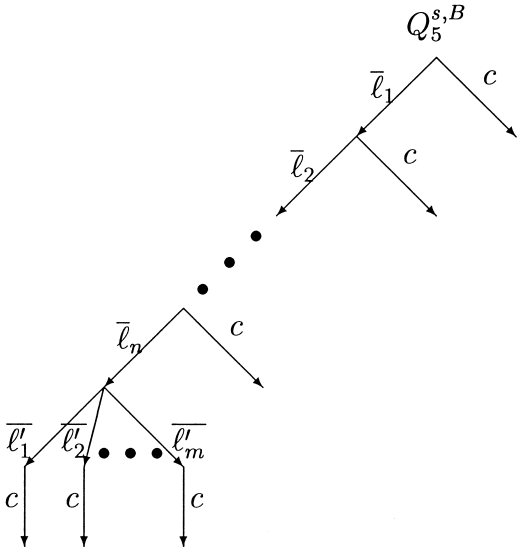

FIG. 2. Process $Q_5^{s,B}$ for context $C_5^{s,B}$.

from $(P \underline{after}\ s)\ \underline{accepts}_{SM}\ B$ we deduce that there is $\ell \in B$ such that $P' \stackrel{\ell}{\Longrightarrow}$; hence $R \stackrel{c}{\Longrightarrow}$.

($\Leftarrow$) Let $R \in (P \underline{after}\ s)$ (if $(P \underline{after}\ s) = \varnothing$, we are done). Due to the form of the context $C_5^{s,B}$, we have that $C_5^{s,B}[P] \Rightarrow R \mid \sum_{\ell \in B} \overline{\ell}.c$. Since c is fresh, the hypothesis $C_5^{s,B}[P]!c$ implies that there exists $\ell \in B$ such that $R \stackrel{\ell}{\Longrightarrow}$, i.e. $R\ \underline{accepts}_M\ B$; moreover, the hypothesis $C_5^{s,B}[P] \downarrow c$ implies that whenever $R \stackrel{\ell}{\Longrightarrow} R'$ with $\ell \in B$ then $R' \downarrow$, i.e., $R \downarrow B$, and the thesis follows. ∎

THEOREM 4.14. *For all processes P and Q, $P \sqsubseteq_{SM} Q$ if and only if $P \downarrow_{\mathscr{L}} \preccurlyeq^{c}_{\mathscr{L}} Q$.*

Proof. The proof can be done along the lines of Theorem 4.11, but relying on Theorem 3.8 and Lemmas 4.10(a) and 4.13 and on the fact that $\sqsubseteq_{SM}$ is a precongruence (Theorem 3.9). ∎

Remark 4.15. It is worthwhile to point out that the context $C_5^{s,B}$ cannot be used in place of the (more complex) context $C_4^{s,B}$ to prove full abstraction for the must preorder (Theorem 4.11). In fact, $P \downarrow s$ does not imply that $C_5^{s,B}[P] \downarrow$ (for instance $a.b.\Omega \downarrow a$ but $C_5^{a,\{b\}}[a.b.\Omega] \uparrow$). This would invalidate the proof of the "if" part of Theorem 4.11.

Indeed, the use of a context very similar to our $C_5^{s,B}$ invalidates a proof in a paper by Main ([17], Lemma 4.2), where the relationships between the must and the maximal trace preorders are studied.

5. AN ASSESSMENT OF THE PREORDERS

In this section we explore the relationships among the preorders we have considered: the uniform setting we have used makes this task relatively simple. Moreover, we comment on the safe-must preorder.

THEOREM 5.1. *For all processes P and Q, $P \sqsubseteq_M Q$ implies $P \sqsubseteq_{SM} Q$, but not vice-versa.*

Proof. We show that $\downarrow \preccurlyeq^{c}_{\mathscr{L}}$ is contained in $\ll_{SM}$, from which the result will follow by applying Theorems 4.11 and 3.8. Suppose that $P \downarrow \preccurlyeq^{c}_{\mathscr{L}} Q$ and that $P \downarrow s$, for some $s \in \mathscr{L}^*$. We show that (a) $Q \downarrow s$ and that (b) $(P \underline{after}\ s)\ \underline{accepts}_{SM} B$ implies $(Q \underline{after}\ s)\ \underline{accepts}_{SM} B$, for any $B \subseteq_{\text{fin}} \mathscr{L}$. Let s be $\ell_1 \cdots \ell_n$. As to (a), just apply Lemma 4.10(a), like in the proof of Theorem 4.11. As to (b), suppose that $(P \underline{after}\ s)\ \underline{accepts}_{SM} B$. It is easy to show (paralleling the proof of Lemmas 4.10 and 4.13 part (b)) that for any P' it holds that $C^{s,B}_5[P'] \downarrow$ and $C^{s,B}_5[P'] ! c$ if and only if $(P' \underline{after}\ s)\ \underline{accepts}_{SM} B$. Applying this result to P and Q it follows $(Q \underline{after}\ s)\ \underline{accepts}_{SM} B$ (just parallel the proofs of Theorems 4.11 and 4.14).

This proves that $P \downarrow \preccurlyeq^{c}_{\mathscr{L}} Q$ implies $P \sqsubseteq_{SM} Q$. To show that the vice-versa does not hold, we exhibit a counterexample. Consider $P \stackrel{\text{def}}{=} a.b.\Omega$ and $Q \stackrel{\text{def}}{=} a$. It is easy to see that $P \sqsubseteq_{SM} Q$, but $P \downarrow \not\preccurlyeq^{c}_{\mathscr{L}} Q$ (just consider the context $\_ \mid \bar{a}$). ∎

The following theorem summarizes the relationships among the precongruences considered in the paper.

THEOREM 5.2.

1. $\sqsubseteq_M \subset \sqsubseteq_{SM} \subset \sqsubseteq_{CT}$.
2. $\sqsubseteq^{c}_{FS}$ *is not comparable with* $\sqsubseteq_M$, $\sqsubseteq_{SM}$, *and* $\sqsubseteq_{CT}$.

Proof.

1. The result follows from Theorems 4.5, 4.11, 4.14 and 5.1. By definition, it is easily seen that $\downarrow_{\mathscr{L}} \preccurlyeq^{c}_{\mathscr{L}}$ is included in $\downarrow_{\mathscr{L}} \preccurlyeq^{*}$. The inclusion is strict: $a \downarrow_{\mathscr{L}} \preccurlyeq^{*} \mathbf{0}$ but $a \downarrow_{\mathscr{L}} \not\preccurlyeq_{\mathscr{L}} \mathbf{0}$.

2. To see that neither of $\sqsubseteq_M$, $\sqsubseteq_{SM}$, and $\sqsubseteq_{CT}$ is included in $\sqsubseteq_{FS}$ (hence in $\sqsubseteq^{c}_{FS}$), consider the processes $P \stackrel{\text{def}}{=} recX.(a.X[\,]a.b)$ and $Q \stackrel{\text{def}}{=} recX.a.X$. Clearly, $P \sqsubseteq_M Q$; hence, $P \sqsubseteq_{SM} Q$ and $P \sqsubseteq_{CT} Q$. However, $P \not\sqsubseteq_{FS} Q$ (because $P\ \underline{must}_{FS}\ O$ and $Q\ \underline{mu\not{s}t}_{FS}\ O$, when $O \stackrel{\text{def}}{=} recX.(\bar{a}.X[\,]\bar{b}.w)$). To see the converse, observe that $\mathbf{0} \sqsubseteq^{c}_{FS} \Omega$, but $\mathbf{0} \not\sqsubseteq_{CT} \Omega$, hence $\mathbf{0} \not\sqsubseteq_{SM} \Omega$ and $\mathbf{0} \not\sqsubseteq_{M} \Omega$. ∎

The mutual relationships among the precongruences are simpler if we move to *strongly convergent* processes. We say that a process P is strongly convergent if $P \downarrow s$ for every $s \in \mathscr{L}^*$.

THEOREM 5.3. *For strongly convergent processes, it holds that*

$$\sqsubseteq^{c}_{FS} \subset \sqsubseteq_M = \sqsubseteq_{SM} \subset \sqsubseteq_{CT}.$$

Proof. The alternative characterizations of $\sqsubseteq_M$ and $\sqsubseteq_{SM}$, show that they coincide for strongly convergent processes: this accounts for the equality.

We show now that if P and Q are strongly convergent then $P \sqsubseteq^{c}_{FS} Q$ implies $P \sqsubseteq_M Q$. For this we prove that $\preccurlyeq^{c}_{\mathscr{L}}$, restricted to strongly convergent processes, is contained in $\ll_M$, from which the result follows by Theorems 4.8 and 3.7. Indeed, assume that $P \preccurlyeq^{c}_{\mathscr{L}} Q$ and that $(P \underline{after}\ s)\ \underline{accepts}_M B$, for some $s \in \mathscr{L}^*$ and

$B \subseteq \mathscr{L}$. We prove that $(Q \underline{after}\ s)\ \underline{accepts}_M B$ as well, which, since the considered processes are strongly convergent, is sufficient to prove the thesis. By Lemma 4.10(b), it follows that $C_4^{s,B}[P]!c$, which in turn implies $C_4^{s,B}[Q]!c$, which in turn, in virtue of Lemma 4.10(b), implies the wanted $(Q \underline{after}\ s)\ \underline{accepts}_M B$. This proves that $P \ll_M Q$ and concludes the proof that $P \sqsubseteq_M Q$. The same counterexamples used in the proof of Theorem 5.2 show that the inclusions $\sqsubseteq_{FS}^{c} \subset \sqsubseteq_M$ and $\sqsubseteq_{SM} \subset \sqsubseteq_{CT}$ are strict. ∎

We conclude the section with a discussion on the safe–must preorder. First, we outline an axiomatization for it.

A key axiom is the law

$$\ell.\Omega = \ell.\Omega \oplus \mathbf{0}. \tag{S}$$

The intuition behind (S) is that if action ℓ is "unsafe" (leads to divergence), ℓ cannot even be guaranteed. It is not difficult to prove that any axiomatization for $\sqsubseteq_M$ (e.g., those in [9, 13]) augmented with the law (S) yields a sound and complete axiomatization for $\sqsubseteq_{SM}$. The idea is to show that must and safe-must coincide over normal forms (in the sense of [9, 13]) which have additionally been "saturated" with respect to left-to-right applications of the law (S). These *safe* normal forms can be defined inductively as follows:

- Ω is a safe normal form;
- $\sum_{A \in \mathscr{L}} \sum_{\ell \in A} \ell.P_\ell$ is a safe normal form if P_ℓ is a safe normal form for each $\ell \in \cup\{A : A \in \mathscr{L}\}$, and whenever $P_{\ell_0} \Rightarrow \Omega$, for some $\ell_0 \in A$ with $A \in \mathscr{L}$, then $A \backslash \{l_0\} \in \mathscr{L}$.

We omit the details of the proof, which develops along the same lines of [9, 13,], and is based on transforming via (S) the normal forms there introduced.

Safe-must is closely related to a variant of readiness semantics considered by Olderog [24]. In particular, the *must engage* relation that is used in [24] to define this variant is very similar to our $\underline{must}_{SM}$, modulo the fact that we consider observers, whereas there traces are considered. Indeed, it is easily seen that $\sqsubseteq_{SM}$ is included in Olderog's semantics, while vice-versa is not true, as shown by the following counterexample. Consider the processes $P = a.\Omega[]b$ and $Q = (a.\Omega[]b) \oplus \mathbf{0}$. They have the same readiness semantics in the sense of Olderog but, by considering the observer $O \stackrel{\text{def}}{=} b.w$, we have $P\ \underline{must}_{SM}\ O$ and $Q\ \underline{\not must}_{SM}\ O$.

However, a slight variation of Olderog's readiness semantics coincides with our safe-must. It is sufficient to modify the third set of the *close* operation used by Olderog, Definition 4.4.1, p. 126 of [24], where $\Gamma(P)$ is, essentially, the set of ready pairs and divergence points of P:

$$\{(s, F') \mid \text{there exists } \ell \text{ such that } (s\ell, \uparrow) \in \Gamma(P) \text{ and } F' \subseteq succ(s, P)\}$$

with the following:

$$\{(s, F) \mid F \subseteq succ(s, P) \text{ and there exists } F' \text{ such that } s, F' \in \Gamma(P),\ F' \backslash F \neq 0 \text{ and for each } \ell \in F' \backslash F\colon (s\ell, \uparrow) \in \Gamma(P)\}.$$

6. GENERALIZATIONS AND FUTURE WORK

The full abstraction results for TCCS (Theorems 4.5, 4.8, 4.11, and 4.14) can be easily extended to many other process languages, provided the theorems are restated to replace each of the (must and safe-must) testing preorders with the induced precongruences. This is necessary to cope with operators that might not preserve the preorders, such as the CCS choice operator $+$.

More precisely, the full abstraction results can be easily established for any process language that satisfies the following two requirements:

1. The set of operators contains *inaction*, *action prefix*, *external choice*, *parallel composition*, and *relabeling*;
2. The associated labelled transition system is *finitely branching*.

Indeed, once observables and testing preorders for such a language have been defined, the wanted results can be obtained by simply noticing that:

— the proofs of the full abstraction theorems only rely on contexts that can be built using the operators listed above and, for must and safe-must, on the existence of alternative characterizations preorders $\ll_M$ and $\ll_{SM}$;

— on the other hand, the alternative characterizations $\ll_M$ and $\ll_{SM}$ are (almost) language-independent, as they only rely on the two previously mentioned requirements (see also [13], Chap. 4.4).

The observables we have considered capture natural communications and convergence capabilities of reactive systems. It is not difficult to strengthen the corresponding predicates to get more inspective observables, some of which require considering a richer base language than TCCS. Below, we discuss a few possible generalizations.

- *Efficiency* can be taken into consideration by counting the number of internal actions, in the spirit of [2, 22]. To capture the above within our setting, we can refine the guarantee predicate with information about the number of $\rightarrowtail$-reductions needed to reach a state capable of the visible action. More precisely, we could define $P!^m\ell$, $m \geqslant 0$, as:

$$\text{whenever } P \rightarrowtail {}^iP' \text{ then there is } j \text{ such that } i+j \leqslant m \text{ and } P' \rightarrowtail {}^j\xrightarrow{\ell}.$$

We strongly conjecture that this observable (plus convergence) exactly captures the must *efficiency testing* preorder of Natarajan and Cleaveland [22], provided that the language is extended with the parallel composition operator $|_k$ of [22], that depends on a natural number k which decreases whenever an internal action takes place.

- *Distribution* of systems can be observed by tagging visible actions with some information about the location where they take place, see, e.g., [3, 7]. A natural choice for us would be to consider a guarantee predicate of the form $P!(\ell@u)$, where u is a *locality* in the sense of [3]. If the language is extended with a parallel composition operator which requires two synchronizing actions to be tagged with

the same locality, it should not be difficult to establish full abstraction results for the resulting testing theories.

• *Timing* aspects of processes behavior are elegantly modeled in [23, 29, 30] by adopting a *two-phase* operational semantics. One phase models occurrence of usual atomic actions, the other models time passing via "idling" transitions, of the form $P \xrightarrow{d} P'$ (d nonnegative real). A parallel composition of processes can idle only if both the components can; as a consequence, in a testing scenario, observers can be used to exactly detect how long processes can idle. Full abstraction results with respect to testing could be obtained by extending the guarantee predicate to idling actions: $P!d$ if and only if whenever $P \stackrel{d'}{\Longrightarrow} P'$, $d' < d$, then $P' \stackrel{d-d'}{\Longrightarrow}$ (here $Q \stackrel{d}{\Longrightarrow} Q'$ means $Q \Rightarrow \xrightarrow{d_1} \Rightarrow \cdots \Rightarrow \xrightarrow{d_n}$, for some $d_1, ..., d_n$ such that $\sum_{i=1}^{n} d_i = d$).

In much the same vein, basic observables could be devised to deal with other features of concurrent systems, such as priority, probability and causality, and to assess the different proposals for tackling these issues.

7. CONCLUDING REMARKS AND RELATED WORK

We have advocated the general approach of defining behavioral preorders for processes as the maximal precongruences induced by basic observables. As a case study we have considered a simple process algebra (TCCS) and three observables that check the communication capabilities of processes and the possibility that processes have of getting engaged in infinite internal computations. Our standpoint is vindicated by the fact that all but one of the obtained precongruences for TCCS do correspond to preorders long studied in the literature [4, 9, 21]. If we had to sum up the main achievements of our approach, we could say that it represents a uniform basis for defining testing–based observational preorders.

Our results would still hold for languages that significantly differ from the process algebra considered in this paper. For example, our approach can be used for defining behavioral equivalences for asynchronous models of parallelism, once one has fixed what are the important facets of such systems. Also, it can be used to capture other semantics aiming to describe other aspects (efficiency, location, duration, ...) of concurrent systems. Of course, relations defined in terms of context closure might turn out to be of little use for practical applications, but they can definitely be used prescriptively to assess alternative characterization more amenable to automatic checking.

Beside Milner and Sangiorgi [19], notions of observables in the same spirit as ours have been proposed by Main [17], Vogler [28], Hennessy [14], Ferreira [11], and Laneve [16].

In [17], it was shown that the precongruence induced by the preorder based on inclusion of maximal traces coincides, both for CCS and CSP, with the must precongruence of [9]; another characterization is given by only considering the inclusion of the maximal ε-trace; i.e., a sequence of invisible moves leading to a divergent state or to a deadlocked one. These basic observables hinder the role played by the convergence test, which is somehow included in that for maximality, and this prevents the capture of different notions, such as fair testing.

In [28], two Petri nets are called *d-equivalent* if they both can reach a deadlocked state or if they both cannot do so. Then it is proved that the variant of failure semantics [6] that ignores divergence is obtained by closing d-equivalence with respect to parallel composition.

In [14], a series of variants of the testing framework are proposed and results are listed that show how, by changing the expressive power of observers, a number of equivalences ranging from bisimulation to testing can be captured. One of the considered family of observers consists just of agents of the form $\ell.w.\mathbf{0}$ that somehow resemble our $!\ell$ predicates. It is claimed that for strongly convergent processes the precongruence induced by this family of observers coincides with the must preorder.

Ferreira [11] and Laneve [16] dealt with languages different from classical process algebras. In particular, Ferreira used a predicate which resembles very much the conjunction of our $\downarrow$ and $!\ell$ (based on production of values rather than on communication capabilities) to define a testing preorder for Concurrent ML [27]; this seems to be strongly related to our safe–must preorder. He also conjectured that if one considers pure CCS (and observes communication capabilities instead of value productions) the obtained preorder coincides with the *must* precongruence of [9]; here we have proved this conjecture. Laneve discusses the impact of an observables-based testing scenario on the Join Calculus, a language with elaborate synchronization schemata [12].

ACKNOWLEDGMENTS

We are grateful to L. Aceto, F. van Breugel, W. Ferreira, E.-R. Olderog, A. Rensink, and W. Vogler for interesting discussions and suggestions and to F. Focardi for a first debugging of the ideas presented in the paper. Two anonymous referees provided helpful suggestions for improving the presentation.

Received October 6, 1997; final manuscript received August 26, 1998.

REFERENCES

1. Abramsky, S. (1990), The lazy lambda calculus, *in* "Research Topics in Functional Programming" (David Turner, Ed.), Addison–Wesley, Reading, MA.
2. Arun-Kumar, S., and Hennessy, M. (1992), An efficiency preorder for processes, *Acta Inform.* **29**(8), 737–760.
3. Boudol, G., Castellani, I., Hennessy, M., and Kiehn, A. (1993), Observing localities, *Theoret. Comput. Sci.* **114**, 31–61.
4. Brinksma, E., Rensink, A., and Vogler, W. (1995), Fair testing, "Proceedings of CONCUR'95" (I. Lee and S. A. Smolka, Eds.), Lecture Notes in Computer Science, Vol. 962, pp. 313–327, Springer-Verlag, Berlin/New York.
5. Brinksma, E., Rensink, A., and Vogler, W. (1996), Applications of fair testing, *in* "Formal Description Techniques IX, Theory, Applications and Tools" (R. Gotzhein and J. Bredereke, Eds.), IFIP, Chapman & Hall, London/New York.
6. Brookes, S. D., Hoare, C. A. R., and Roscoe, A. W. (1984), A theory of communicating sequential processes, *J. Assoc. Comput. Mach.* **31**(3), 560–599.
7. Castellani, I., and Hennessy, M. (1989), Distributed bisimulations, *J. Assoc. Comput. Mach.* **10**, 887–911.
8. De Nicola, R. (1987), Extensional equivalences for transition systems, *Acta Inform.* **24**, 211–237.

9. De Nicola, R., and Hennessy, M. (1984), Testing equivalence for processes, *Theoret. Comput. Sci.* **34**, 83–133.
10. De Nicola, R., and Hennessy, M. (1987), CCS without τ's, "Proceedings of TAPSOFT'87" (H. Ehrig *et al.*, Eds.), Lecture Notes in Computer Science, Vol. 249, pp. 138–152, Springer-Verlag, Berlin/New York.
11. Ferreira, W. (1996), "Semantic Theories for Concurrent ML," Ph.D. Thesis, University of Sussex.
12. Fournet, C., Gonthier, G., Lévy, J.-L., Maranget, L., and Rémy, D. (1966), A calculus of mobile agents, "Proceedings of CONCUR'96" (U. Montanari and V. Sassone, Eds.), Lecture Notes in Computer Science, Vol. 1119, pp. 406–421, Springer-Verlag, Berlin/New York.
13. Hennessy, M. (1988), "Algebraic Theory of Processes," MIT Press, Cambridge, MA .
14. Hennessy, M. (1989), Observing processes, *in* "Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency" (J. de Bakker *et al.*, Eds.), Lecture Notes in Computer Science, Vol. 354, Springer-Verlag, Berlin/New York.
15. Hoare, C. A. R. (1985), "Communicating Sequential Processes," Prentice–Hall, Englewood Cliffs, NJ.
16. Laneve, C. (1996), "May and Must Testing in the Join-Calculus," Technical Report UBLCS-96-4, Università di Bologna, Dept. of Computer Science, Bologna.
17. Main, M. G. (1987), Trace, failure and testing equivalences for communicating processes, *Int. J. Parallel Progr.* **16**(5), 383–400.
18. Milner, R. (1989), "Communication and Concurrency," Prentice–Hall, Englewood Cliffs, NJ.
19. Milner, R., and Sangiorgi, D. (1992), Barbed bisimulation, "Proceedings of ICALP'92" (W. Kuich, Ed.), Lecture Notes in Computer Science, Vol. 623, Springer-Verlag, Berlin/New York.
20. Morris, J.-H. (1968), "Lambda Calculus Models of Programming Languages," Ph.D. Thesis, MIT, Cambridge, MA.
21. Natarajan, V., and Cleaveland, R. (1995), Divergence and fair testing, "Proceedings of ICALP'95" (Z. F'ul'op and F. Gécseg, Eds.), Lecture Notes in Computer Science, Vol. 944, pp. 648–659, Springer-Verlag, Berlin/New York.
22. Natarajan, V., and Cleaveland, R. (1995), An algebraic theory of process efficiency, "Proceedings of LICS'96" (Los Alamitos, California), pp. 63–72, IEEE Comput. Soc. Press, New York.
23. Nicollin, X., and Sifakis, J. (1991), An overview and synthesis on timed process algebras, "Proceedings of CAV'91" (K. G. Larsen and A. Shou, Eds.), Lecture Notes in Computer Science, Vol. 575, pp. 376–398, Springer-Verlag, Berlin/New York.
24. Olderog, E.-R. (1991), "Nets, Terms and Formulas," Cambridge University Press, Cambridge, UK.
25. Olderog, E.-R., and Hoare, C. A. R. (1986), Specification-oriented semantics for communicating processes, *Acta Inform.* **23**, 9–66.
26. Ong, C.-H.L. (1995), Correspondence between operational and denotational semantics: the full abstraction problem for PCF, "Handbook of Logic in Computer Science" (S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, Eds.), Vol. 4, Oxford Science, Oxford.
27. Reppy, J. H. (1993), Concurrent ML: Design, application and semantics, "Proceedings of Functional Programming, Concurrency, Simulation and Automata Reasoning" (P. E. Lauer, Ed.), Lecture Notes in Computer Science, Vol. 693, pp. 165–198, Springer-Verlag, Berlin/New York.
28. Vogler, W. (1989), Failures semantics and deadlocking of modular Petri nets, *Acta Inform.* **26**, 333–348.
29. Wang, Y. (1990), Real-time behaviour of asynchronous agents, "Proceedings of CONCUR'90" (J. C. M. Baeten and J. W. Klop, Eds.), Lecture Notes in Computer Science, Vol. 458, pp. 502–520, Springer-Verlag, Berlin/New York.
30. Wang, Y. (1991), CCS + Time = an interleaving model for real time systems, "Proceedings of ICALP'91" (J. Leach Albert *et al.*, Eds.), Lecture Notes in Computer Science, Vol. 510, Springer-Verlag, Berlin/New York.

Printed in Belgium