Fair Testing

Ed Brinksma[1], Arend Rensink[2] and Walter Vogler[3]

[1] University of Twente; partially supported by the Esprit BRA 6021 'REACT'
[2] University of Hildesheim; partially supported by the HCM network 'EXPRESS'
[3] University of Augsburg

Abstract. We investigate the notion of *fair testing*, a formal testing theory in the style of De Nicola and Hennessy, where divergences are disregarded as long as there are visible outgoing transitions. The usual testing theories, such as the standard model of failure pre-order, do not allow such fair interpretations because of the way in which they ensure their compositionality with respect to *abstraction* from observable actions. This feature is usually present in the form of a hiding-operator (CSP, ACP, LOTOS) or part of parallel composition (CCS). Its application can introduce new divergences causing semantic complications. In this paper we present a testing scenario that captures the intended notion of fairness and induces a pre-congruence for abstraction. In the presence of a sufficiently strong synchronisation feature it is shown to be the *coarsest* pre-congruence contained in the (non-congruent) fair version of failure preorder. We also give a denotational characterisation.

1 Introduction

The usefulness of formalisms for the description and analysis of reactive and distributed systems is closely related to the underlying formal notions of behavioural equivalence. In a given application the formal equivalence should ideally both identify behaviours that are informally indistinguishable and distinguish between behaviours that are informally different. Of course, other criteria apply as well, such as for example the availability of a mathematically tractable and well-understood theory, so that in practice a compromise between the various requirements must be made.

In the past decade research in the field of transition systems has led to the discovery of a wealth of equivalences that can be used to formalise behavioural equivalence (the reader may consult [10] for an overview). Two important families of equivalences are those that employ the notion of *bisimulation* [18, 20], and those that are induced by a formalised notion of testing, the so-called *testing equivalences* [9, 14, 6]. Bisimulations provide the finer equivalences that keep track of the branching structure of behaviours, and have a rather elegant proof theory based of the construction of bisimulation relations. Abramsky has shown in [1] that bisimulation equivalences are also induced by a notion of testing, but only in the presence of a very strong notion of observability. Testing equivalences that can be characterised following the recipe of De Nicola and Hennessy [9] are generally coarser and distinguish mainly on the basis of difference in *deadlock* behaviour, which is in practical cases often sufficient. The higher resolution

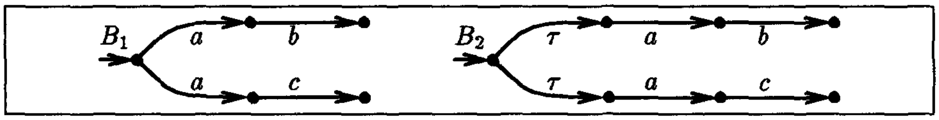

Fig. 1. Shifting nondeterminism

power of bisimulations that is based on the branching structure of processes is, in fact, often undesirable. The transition systems B_1 and B_2 in Fig. 1 are not weak bisimulation equivalent, but are testing equivalent.

In practice, we would sometimes like to implement behaviour B_1 by B_2, see for example [11], by resolving the choice between the two a-actions internally (hence the internal τ-actions in B_2), and not in the interaction with the environment. As the environment cannot influence the choice in either case, this should make no difference to the observable behaviour. A second advantage of testing equivalences is that they are generated by *pre-orders* that can be practically interpreted as *implementation relations*. They usually express that the implementation is some sort of deterministic reduction of the specified behaviour.

A feature of weak bisimulation equivalence is that it incorporates a particular notion of *fairness*. The two behaviours shown in Fig. 2 are weak bisimulation equivalent. Weak bisimulation works on the principle that the τ-loop of B_4 is executed an arbitrary but only finite number of times, in this case implying that eventually action b will be enabled. Such identification of behaviour can be very useful in practice, for example, when proving properties of systems that work with fair communication media. In such cases τ-loops, or *divergences*, represent the unbounded but finite number of message losses. Interesting proofs of protocol correctness that have been shown in this way can be found in [16, 5].

It is not difficult to define a testing pre-order that shares this fairness property with weak bisimulation, see for example [4]. The reason that this so-called fair failure pre-order is not very popular as a basis for developing an algebraic theory of behaviour is that it is not a pre-congruence with respect to the *abstraction* or hiding operation, which internalises observable actions and may thus produce new divergences. We give two examples showing that the fair failure preorder is not a pre-congruence with respect to abstraction.

Fig. 3 is taken from Bergstra et al. [3]; it shows two failure equivalent systems that differ when a is hidden. According to the standard testing scenario, the only observable fact is that after an arbitrary nonempty sequence of a's, either b is refused or c is refused; the difference between the two systems in Fig. 3 is that the left-hand system alternates between allowing b and allowing c, whereas the right-hand side keeps on offering the same action after the initial choice. After hiding a this difference becomes testable, at least if one takes a fair interpretation. Then the left-hand system cannot refuse to do b or c, whereas the right-hand system

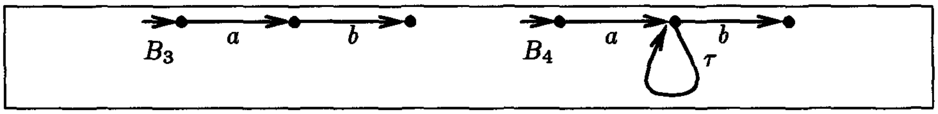

Fig. 2. Fair τ-loop

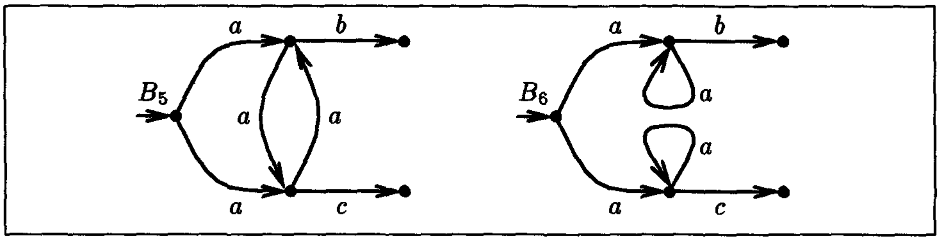

Fig. 3. Failure equivalent systems which are different after hiding *a*

will always refuse either *b* or *c*.

Note that this example is invalidated in some of the stronger notions of testing, such as *refusal testing* (cf. Phillips [21] or Langerak [15]) where testing may continue after a refusal has been observed. Consider, however, the behaviours in Fig. 4. The left hand system has strictly fewer failures than the right hand system, or in other words passes strictly more tests. After hiding *a* however, the failure inclusion no longer holds: in a fair testing scenario, B_6 will always succeed in performing a *b*-action whereas B_5 may refuse to do this.

It can be argued that this example shows that the removal of nondeterminism (taking away the *a*-*b* branch in Fig. 4) interferes with the congruence w.r.t. hiding we are after. In this paper we will show that this is true only if the nondeterministic option that is taken away is somehow the only remaining possibility for the system to terminate. Based on this insight we will develop a theory of fair testing that does possess the desired compositionality with respect to abstraction.

The rest of the paper is structured as follows. Sect. 2 contains the definitions of the basic concepts and notation that we use. In Sect. 3 we introduce a new notion of testing, viz. *should-testing*, and define the induced pre-order on behaviours. In Sect. 4 we study the congruence properties and in Sect. 5 the fairness properties of the should pre-order. In Sect. 6 we give an alternative characterisation of the should pre-order, based on a generalisation of the concept of failure pairs (cf. [6]). Finally, in Sect. 7 we draw our conclusions comparing our work to and draw our conclusions existing approaches.

2 Definitions

We assume a set **A** of actions, ranged over by a, b, c. Apart from **A** we use two special action symbols: the *invisible action* τ and the *success action* ω. The latter is used for the purpose of testing, to denote the successful conclusion of a test. In contrast, the actions in **A** are sometimes called *visible actions*. We use α to

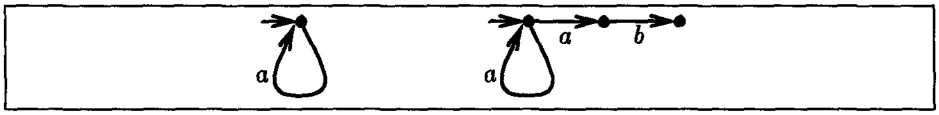

Fig. 4. Failure included systems which are incomparable after hiding *a*

Table 1. Structural operational semantics of $\mathbf{L}$

$$\frac{}{\mathbf{succ} \xrightarrow{\omega} \mathbf{stop}} \qquad \frac{}{\alpha; B \xrightarrow{\alpha} B} \qquad \frac{B \xrightarrow{\alpha} B'}{\Sigma(\mathbf{B} \cup \{B\}) \xrightarrow{\alpha} B'}$$

$$\frac{B \xrightarrow{\alpha} B' \quad \alpha \notin A}{B \parallel_A C \xrightarrow{\alpha} B' \parallel_A C} \qquad \frac{C \xrightarrow{\alpha} C' \quad \alpha \notin A}{B \parallel_A C \xrightarrow{\alpha} B \parallel_A C'} \qquad \frac{B \xrightarrow{\alpha} B' \quad C \xrightarrow{\alpha} C' \quad \alpha \in A}{B \parallel_A C \xrightarrow{\alpha} B' \parallel_A C'}$$

$$\frac{B \xrightarrow{\alpha} B'}{B[\phi] \xrightarrow{\phi(\alpha)} B'[\phi]} \qquad \frac{B \xrightarrow{\alpha} B' \quad \alpha \in A}{B/A \xrightarrow{\tau} B'/A} \qquad \frac{B \xrightarrow{\alpha} B' \quad \alpha \notin A}{B/A \xrightarrow{\alpha} B'/A} \qquad \frac{\theta(X) \xrightarrow{\alpha} B}{X \xrightarrow{\alpha} B}$$

range over $\mathbf{A} \cup \{\tau\}$. Furthermore we assume a set $\mathbf{X}$ of *process names*, ranged over by X. Process names are used in recursive process equations to specify infinite behaviour. We assume a *process environment* $\theta: \mathbf{X} \rightarrow \mathbf{L}$ containing the process definitions. We will use $X =_\theta B$ to denote $\theta(X) = B$. On this basis we define a language $\mathbf{L}$, inspired by CCS (see Milner [18]) and CSP (see Brookes, Hoare and Roscoe [6]), with the following grammar:

$$B ::= \mathbf{succ} \mid \alpha; B \mid \Sigma \mathbf{setof}\ B \mid B \parallel_A B \mid B[\phi] \mid B/A \mid X \ .$$

Furthermore, we use abbreviated forms of summation and synchronisation:

$$\begin{aligned} \mathbf{stop} &= \Sigma\varnothing \\ B_1 + B_2 &= \Sigma\{B_1, B_2\} \\ B_1 \mathbin{|||} B_2 &= B_1 \parallel_\varnothing B_2 \\ B_1 \parallel B_2 &= B_1 \parallel_{\mathbf{A}} B_2 \ . \end{aligned}$$

The constant **succ** may only do an ω-action, after which it reduces to **stop**. In addition, the language features a family of action prefix operators ($\alpha \in \mathbf{A} \cup \{\tau\}$ arbitrary), a CCS-like infinitary summation operator Σ, a CSP-like parallel composition operator indexed by the set of synchronisation actions ($A \subseteq \mathbf{A}$ arbitrary), a renaming operator indexed by a function ($\phi: \mathbf{A} \rightarrow \mathbf{A}$ arbitrary, extended with $\tau \mapsto \tau$), a hiding operator indexed by the set of actions to be abstracted away from ($A \subseteq \mathbf{A}$ arbitrary) and process invocation. We take these operators to be sufficiently familiar to make an extensive discussion superfluous. Note that we have not included a restriction operator, on the grounds that the form of synchronisation we have chosen already allows restriction. As usual, we define a transition relation over terms, consisting of triples $B \xrightarrow{\alpha} C$ denoting that the term B evolves into the term C by doing the action α. This relation is defined inductively by way of the SOS rules in Table 1. The simple transition relation $\rightarrow$ gives rise to another, string-labelled relation defined as follows: for all $\sigma = a_1 \cdots a_n \in \mathbf{A}^*$

$$B \overset{\sigma}{\Longrightarrow} C :\Leftrightarrow B \xrightarrow{\tau}{}^* \xrightarrow{a_1} \xrightarrow{\tau}{}^* \cdots \xrightarrow{\tau}{}^* \xrightarrow{a_n} \xrightarrow{\tau}{}^* C$$

We also frequently use $B \overset{\sigma}{\Longrightarrow}$ to denote $\exists C. B \overset{\sigma}{\Longrightarrow} C$. We furthermore use the *label set* of a term, defined inductively in Table 2. We also briefly recall the standard notions of *observation equivalence* and *congruence* (cf. e.g. Milner [18]):

Table 2. Label set of a term

$L(\mathbf{succ})$	$:=$	$\varnothing$
$L(a;B)$	$:=$	$\{a\} \cup L(B)$
$L(\Sigma \mathbf{B})$	$:=$	$\bigcup\{L(B) \mid B \in \mathbf{B}\}$
$L(B_1 \parallel_A C)$	$:=$	$L(B_1) \cup L(C)$
$L(B[\phi])$	$:=$	$\phi(L(B))$
$L(B/A)$	$:=$	$L(B) - \{A\}$
$L(X)$	$:=$	$\bigcup\{L(X^i) \mid i \in \mathbb{N}\}$ where $X^0 := \mathbf{stop}$, $X^{i+1} := \theta(X)\{^{X^i}/_X\}$

Definition 1 observation congruence. *Observation equivalence* is the largest equivalence relation $\approx \subseteq \mathbf{L} \times \mathbf{L}$ such that $B_1 \approx C$ implies that for all $B_1 \xrightarrow{\alpha} B_1'$ there is a $C' \approx B_1'$ such that $C \overset{\alpha}{\Longrightarrow} C'$. *Observation congruence* is the largest relation $\approx^c \subseteq \approx$ that is a congruence for the operators of $\mathbf{L}$.

Now we recall the testing scenario presented by De Nicola and Hennessy in [9], and a variation studied by Brinksma in [4]. For this purpose we distinguish *system descriptions* and *tests* for those systems, all of which are represented formally as terms of $\mathbf{L}$. The constant **succ** is allowed only in tests. A test $t \in \mathbf{L}$ is *applied* to a system $B \in \mathbf{L}$ by letting the two synchronise, as in $B \parallel t$. This test application is then judged to be either successful or unsuccessful; the verdict relies on the presence of sufficiently many ω-transitions in strategic places. De Nicola and Hennessy consider two kinds of evaluation, called *may-testing* and *must-testing*, respectively. We define the latter through a binary relation between systems and tests. A *maximal run* is a sequence $B_0 \xrightarrow{\alpha_0} B_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{n-1}} B_n \cdots$, which can be finite or infinite; in the former case, the final term should have no outgoing transitions.

$$B \;\mathbf{must}\; t \;\;:\Leftrightarrow\;\; \forall \text{ maximal runs } (B \parallel t) = B_0 \xrightarrow{\alpha_0} B_1 \xrightarrow{\alpha_1} \cdots .\, \exists i.\, B_i \xrightarrow{\omega} \;.$$

(May-testing may be defined in a like manner; however, we do not pursue this notion here.) A must-test, according to this definition, tests if every maximal run of B passes through a successful state. In particular, the presence of divergence in B (in the form of an infinite τ-path) may ruin a test. Brinksma has defined a "fair" variation on must-testing which concentrates on *finite, visible* runs; the effect is that divergence is ignored as long as there is a visible action available. The advantage of this notion is that it is compatible with observation congruence; an important disadvantage is that it is not a congruence for hiding. See [4] for a more extensive discussion. Recall that $\omega \notin \mathbf{A}$.

$$B \;\mathbf{fmust}\; t \;\;:\Leftrightarrow\;\; \forall \sigma \in \mathbf{A}^*, B' \in \mathbf{L}.\, B \parallel t \overset{\sigma}{\Longrightarrow} B' \text{ implies } (\exists a \in \mathbf{A} \cup \{\omega\}.\, B' \overset{a}{\Longrightarrow}) \;.$$

On the basis of binary relations $\rho \subseteq \mathbf{L} \times \mathbf{L}$ such as **must** and **fmust**, one may induce an *implementation relation* and a corresponding *equivalence*:

$$I \sqsubseteq_\rho S \;:\Leftrightarrow\; \forall t \in \mathbf{L}.\, (S \;\rho\; t) \Longrightarrow (I \;\rho\; t)$$
$$I \simeq_\rho S \;:\Leftrightarrow\; I \sqsubseteq_\rho S \wedge S \sqsubseteq_\rho I \;.$$

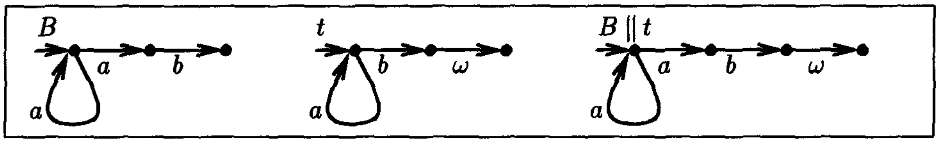

Fig. 5. Should-testing assumes fairness

Relations such as $\approx^c$ and $\sqsubseteq_{\mathbf{fmust}}$ are in a sense *fair* because of the nonchalant way they deal with divergences: essentially, since the only observations taken into account are visible transitions, τ-loops in a system are ignored. This kind of fairness can be expressed algebraically by the so-called *Koomen's Fair Abstraction Rule* (KFAR$_n$); see e.g. Baeten and Weijland [2]:

$$\frac{X_i = a_i; X_{i+1} + Y_i \quad a_i \in A}{X_i/A = \tau; \Sigma_{i \in \mathbb{N}_n}(Y_i/A)} \; (i \in \mathbb{N}_n) \tag{1}$$

where $X_i, Y_i \in \mathbf{L}$ are arbitrary and $\mathbb{N}_n$ denotes the natural numbers modulo n. It is a standard result that $\approx^c$ satisfies (1). For $\sqsubseteq_{\mathbf{fmust}}$ the situation is slightly more subtle, but since it is compatible with observation congruence ($\approx^c \subseteq \simeq_{\mathbf{fmust}}$) it is easily seen that $\simeq_{\mathbf{fmust}}$ satisfies the weaker property that if the X_i are *defined* as $a_i; X_{i+1} + Y_i$ in the binding environment θ, then certainly $X_i \approx^c a_i; X_{i+1} + Y_i$ and hence the conclusion of (1) holds. See also Sect. 5 below. Even this weaker property, however, does *not* hold for $\simeq_{\mathbf{must}}$; as remarked above, this was a major reason to investigate **fmust**. (Note that $\sqsubseteq_{\mathbf{must}}$ and $\sqsubseteq_{\mathbf{fmust}}$ are incomparable. See [4] for more details.)

3 Should-testing

To repair the non-congruence of fair must-testing for hiding, we introduce a new kind of test evaluation, which we call *should-testing*.

$$B \;\mathbf{should}\; t \;:\Leftrightarrow\; \forall \sigma \in \mathbf{A}^*, B' \in \mathbf{L}. \,(B \,||\, t \overset{\sigma}{\Longrightarrow} B' \text{ implies } \exists \sigma' \in \mathbf{A}^*. \, B' \overset{\sigma'\omega}{\Longrightarrow}) \;.$$

The idea behind should-testing is that there is always a reachable successful state, and hence if choices are made fairly, a successful state should eventually indeed be reached. For instance, if B and t are as in Fig. 5, then $B \,||\, t$ can in principle avoid ω forever by staying in the loop; nevertheless B **should** t because it is assumed that the other branch is not ignored forever.

The fairness properties of $\sqsubseteq_{\mathbf{should}}$ are studied in more detail in Sect. 5. Note the similarity of should-testing to fair must-testing. Fair must-testing states that a system may not deadlock unless a success action occurs first. Should-testing on the other hand requires something stronger: a success action must be reachable from every state in the system unless one has occurred already. For instance, the left hand system B in Fig. 4 passes the fair must-test $t = X$ where $X =_\theta a; X$ (there is no deadlock at all in $B \,||\, t$), but it does not pass t as a *should*-test (there is no reachable ω-transition). In fact it is easy to verify that for all B and t

$$B \;\mathbf{should}\; t \Longrightarrow B \;\mathbf{fmust}\; t \;.$$

Furthermore it is easy to see that the difference between should- and fair must-testing only lies in the treatment of infinite behaviour. If there are no infinite visible paths in $B \,\|\, t$ then every failure to pass a should-test can be reduced to the failure of the corresponding fair must-test. To a certain degree we can control the occurrence of infinite paths of $B \,\|\, t$, by selecting t appropriately: it follows that for every B and every t with only finite visible runs

$$B \;\mathbf{should}\; t \iff B \;\mathbf{fmust}\; t \ .$$

This is in particular interesting because it is well known that $\sqsubseteq_{\mathbf{fmust}}$ can be decided on the basis of finite tests only; in fact, for deciding this relation the subclass of *failure tests* suffices.

Definition 2 failures and failure tests. A *failure* is a pair (σ, A), where $\sigma \in \mathbf{A}^*$ is a trace attempted by a system and $A \subseteq \mathbf{A}$ a set of actions that can subsequently be *refused*. To every failure there corresponds a *failure test*, denoted $t_{\sigma,A}$; these are defined inductively by

$$\begin{aligned} t_{\varepsilon,A} &:= \Sigma\{a; \mathbf{succ} \mid a \in A\} \\ t_{a\sigma,A} &:= \mathbf{succ} + a; t_{\sigma,A} \ . \end{aligned}$$

The characterisation result mentioned above is as follows: $I \sqsubseteq_{\mathbf{fmust}} S$ iff for all σ and A, $S \;\mathbf{fmust}\; t_{\sigma,A} \Rightarrow I \;\mathbf{fmust}\; t_{\sigma,A}$. We may conclude the following:

Corollary 3. $\sqsubseteq_{\mathbf{should}} \subseteq \sqsubseteq_{\mathbf{fmust}}$.

4 Congruence Properties of Should-Testing

We first prove that $\sqsubseteq_{\mathbf{should}}$ is a pre-congruence for hiding. This depends on an intermediate lemma. An auxiliary definition first: for all $A \subseteq \mathbf{A}$, let

$$R_A =_\theta \Sigma\{a; R_A \mid a \in A\} \ .$$

Theorem 4. $\sqsubseteq_{\mathbf{should}}$ *is a pre-congruence for hiding.*

Proof. First note that for all $B, t \in \mathbf{L}$ and $A \subseteq \mathbf{A}$ such that $A \cap L(t) = \varnothing$

$$B \;\mathbf{should}\; (t \,|||\, R_A) \iff (B/A) \;\mathbf{should}\; t \ .$$

This follows by observing that for all such B and t the following holds:

$$(B/A) \,\|\, t \overset{\sigma}{\Longrightarrow} (B'/A) \,\|\, t' \quad \text{iff} \quad \exists \rho.\, \rho/A = \sigma \wedge B \,\|\, (t \,|||\, R_A) \overset{\rho}{\Longrightarrow} B' \,\|\, (t' \,|||\, R_A)$$

where ρ/A denotes the string obtained from ρ by removing all occurrences of actions from A. Using this fact, any failure of B w.r.t. $t \,|||\, R_A$ can be converted to a failure of B/A w.r.t. t, and vice versa.

Now let $I \sqsubseteq_{\mathbf{should}} S$, and let $A \subseteq \mathbf{A}$ be arbitrary. If (S/A) **should** t for some arbitrary t then without loss of generality we may assume that t does not contain actions from A (because these are prevented from occurring anyway, due

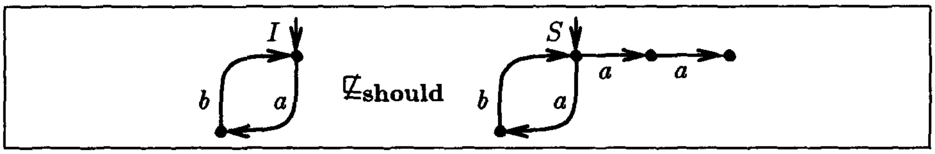

Fig. 6. $\sqsubseteq_{\mathbf{fmust}}$-related systems after arbitrary abstraction, but not $\sqsubseteq_{\mathbf{should}}$-related

to synchronisation); it follows that S **should** $(t \mid\mid\mid R_A)$, hence I **should** $(t \mid\mid\mid R_A)$, hence (I/A) **should** t. We may conclude $I/A \sqsubseteq_{\mathbf{should}} S/A$. □

For the other operators the situation is as for fair must-testing: we have congruence with respect to all operators except choice; to obtain congruence with respect to the latter, a straightforward side condition has to be added, stating that instability is preserved.

Theorem 5. $\sqsubseteq_{\mathbf{should}}$ *is a pre-congruence for prefixing, synchronisation and renaming.*

Proof sketch. By manipulating tests as in the proof of Theorem 4. The case of synchronisation is the most complex. Assume $I \sqsubseteq_{\mathbf{should}} S$; to be proved is $I \parallel_A B \sqsubseteq_{\mathbf{should}} S \parallel_A B$ for arbitrary $B \in \mathbf{L}$ and $A \subseteq \mathbf{A}$. We show the subcase that S is incapable of performing actions in $A' = L(B) - A$ (i.e., $S \overset{\sigma}{\Longrightarrow}$ implies $\sigma \in (\mathbf{A} - A')^*$); then it can be proved that in any interleaving semantics

$$(S \parallel_A B) \parallel t = S \parallel_{\mathbf{A}-A'} (B \parallel t) \ .$$

In turn, for the purpose of should-testing, the right hand term has the same failure capabilities as $S \parallel ((B \parallel t)/A')$, and hence we may conclude

$$(S \parallel_A B) \ \mathbf{should}\, t \iff S \ \mathbf{should}\, ((B \parallel t)/A') \ .$$

A similar property holds for $I \parallel_A B$; this essentially concludes the proof. □

Having established this, we return to the comparison of $\sqsubseteq_{\mathbf{should}}$ and $\sqsubseteq_{\mathbf{fmust}}$. We have that the former is a congruence for the majority of the operators in $\mathbf{L}$ and is contained in the latter. It is therefore natural to investigate the relation to the *coarsest congruence* in $\sqsubseteq_{\mathbf{fmust}}$. The initial observation is discouraging: $\sqsubseteq_{\mathbf{should}}$ is *not* the coarsest congruence for hiding within $\sqsubseteq_{\mathbf{fmust}}$. Consider the behaviour in Fig. 6. Here $I \not\sqsubseteq_{\mathbf{should}} S$; take for instance $t = X$ where $X =_\theta a; (b; X + a; \omega)$. On the other hand, it is easily seen that $I/A \sqsubseteq_{\mathbf{fmust}} S/A$ for arbitrary A.

As soon as we take more operators into consideration than just hiding, however, the situation suddenly changes for the better. It turns out that the coarsest congruence for hiding is *not* a congruence for parallel composition; and taking both operators into account at the same time *does* force the coarsest congruence down to $\sqsubseteq_{\mathbf{should}}$.

Theorem 6. $\sqsubseteq_{\mathbf{should}}$ *is the largest relation contained in* $\sqsubseteq_{\mathbf{fmust}}$ *that is a pre-congruence for both synchronisation and hiding.*

For the proof idea, consider once more Fig. 6. If we put both systems into the context $C[] = ([] \|_A Y)/A$ where $Y =_\theta a;(b;Y + a;c;\mathbf{stop})$ and $A = \{a, b\}$, then the right hand system satisfies the fair must-test $c;\mathbf{succ}$ whereas the left hand system does not. Note that the process Y in this context is very similar to the *should*-test t that was used to differentiate these systems in the first place: where t does the special success action ω, Y does an ordinary, but fresh action c; it then synchronises with the system on all actions except c and subsequently abstracts away from all actions except c. It is easy to see that B **should** t whenever $C[B]$ **fmust** $c;\mathbf{succ}$. The same pattern applies in general.

As mentioned above, the reason why $\sqsubseteq_{\mathbf{should}}$ fails to be a congruence with respect to choice is standard, as is the repair. We define

$$I \sqsubseteq^c_{\mathbf{should}} S \;:\Leftrightarrow\; I \sqsubseteq_{\mathbf{should}} S \wedge (I \xrightarrow{\tau} \text{ implies } S \xrightarrow{\tau})$$

This brings us to one of the main results of this paper. The proof is standard, combining the results achieved above.

Theorem 7. *$\sqsubseteq^c_{\mathbf{should}}$ is the largest relation contained in $\sqsubseteq_{\mathbf{fmust}}$ that is pre-congruent with respect to the operators of* $\mathbf{L}$.

5 Fairness Properties of Should-Testing

An important issue in introducing a new behavioural relation is to compare it to existing relations. Above we have done this for $\sqsubseteq^c_{\mathbf{should}}$ by showing it to be a coarsest congruence contained in a known relation; a further property is that its symmetric closure, $\simeq^c_{\mathbf{should}}$, contains observation congruence.

Theorem 8. $\approx^c \subseteq \simeq^c_{\mathbf{should}}$.

The proof has to be omitted for lack of space. However, this has an immediate consequence for the fairness properties of $\simeq^c_{\mathbf{should}}$, which "inherits" fairness from $\approx^c$ in the manner discussed in Sect. 2.

Corollary 9. *For all $X_i \in \mathbf{X}$ and $Y_i \in \mathbf{L}$ the following "weak KFAR" holds:*

$$\frac{X_i =_\theta a_i;X_{i+1} + Y_i \quad a_i \in A}{X_i/A \simeq^c_{\mathbf{should}} \tau;\Sigma(Y_i/A)} \; (i \in \mathbb{N}_n)$$

A natural question is if the full KFAR (Eq. 1) also holds. Unfortunately, this is not the case. Fig. 7 shows a counterexample with $n = 1$: $X \simeq_{\mathbf{should}} a;X + B$ but $X/a \not\simeq_{\mathbf{should}} B/a$ since (B/a) **should** $b;\mathbf{succ}$ but (X/a) $\cancel{\mathbf{should}}$ $b;\mathbf{succ}$.

The built-in fairness assumption of should-testing can also be expressed in another, more classical way. Loosely speaking, if a state is encountered infinitely often, then all its outgoing transitions will eventually be taken. To make this precise we define for $B_0 \in \mathbf{L}$: $B_0 \xrightarrow{\alpha_0} B_1 \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_{n-1}} B_n \dots$ is a *fair run* of B_0 if it is maximal and contains infinitely often each transition $B \xrightarrow{\alpha} B'$ for which B occurs infinitely often. Moreover, we call a process B *finite state* if there are only finitely many reachable B' (i.e., with $\exists \sigma \in \mathbf{A}^*.\, B \overset{\sigma}{\Longrightarrow} B'$).

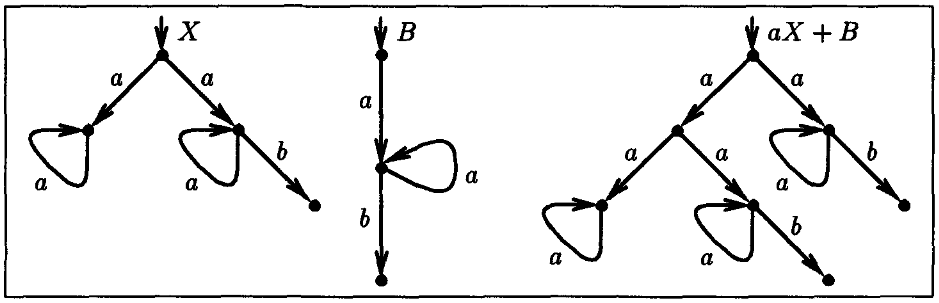

Fig. 7. A counterexample to Koomen's Fair Abstraction Rule for $\simeq^c_{\mathbf{should}}$

Lemma 10. *Let $B \in \mathbf{L}$ be a finite state process. If for every B' reachable from B there is some $\sigma \in \mathbf{A}^*$ with $B' \overset{\sigma\omega}{\Longrightarrow}$ then every fair run of B contains an ω-transition.*

The proof is straightforward and omitted here. The condition of the lemma is obviously connected to the **should**-relation. The following makes the connection explicit.

Corollary 11. *Let $B, t \in \mathbf{L}$ be finite state processes. B* **should** *t if and only if every fair run of $B \,||\, t$ contains an ω-transition.*

6 An internal characterisation for should-testing

As all test-equivalences, $\sqsubseteq_{\mathbf{should}}$ is defined externally by referring to arbitrary test environments. We now present a failure-type semantics which allows to characterise $\sqsubseteq_{\mathbf{should}}$ internally; this semantics was first developed some time ago, and independently of the testing framework, in [23] to deal with liveness in the sense of Petri net theory.

Consider again the example of Fig. 4. In their initial states, both systems can only perform a as an immediate next action; they can refuse all other actions. This information is insufficient to determine the behaviour after hiding a; here it is important that the right-hand system can perform ab initially while the left-hand system cannot. Hence, one can get the idea to study refusals of sequences instead of single actions.

As a first step, we define a variant of failure semantics where the refusal sets lie in $\mathbf{A}^+$ instead of $\mathbf{A}$. For a term $B \in \mathbf{L}$, define $F^+(B)$ as the set of all (σ, A) with $\sigma \in \mathbf{A}^*$ and $A \subseteq \mathbf{A}^+$ such that

$$\exists B' \in \mathbf{L}.\, B \overset{\sigma}{\Longrightarrow} B' \wedge \forall \sigma' \in A.\, B' \overset{\sigma'}{\not\Longrightarrow} \ .$$

The systems in Fig. 8 demonstrate that this easily defined semantics is too discriminating for our purpose. These systems have the same failure semantics, hence they are fmust-equivalent and —since their behaviour is finite— also should-equivalent. But the left-hand system can perform a and refuse $\{aa, b\}$,

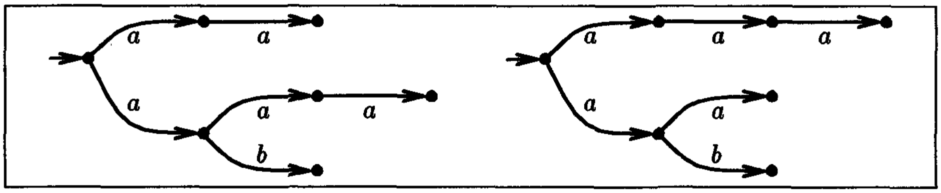

Fig. 8. F^+-semantics is too fine

while this is not possible for the right-hand system. It follows that we have to "saturate" the model somehow, so that this difference becomes unobservable.

Let us think back on the testing framework defined in Sect. 3. Just as the failure tests suffice to decide $\sqsubseteq_{\mathbf{fmust}}$, leading to the failure model, we might look for a minimal set of "essential tests" to decide $\sqsubseteq_{\mathbf{should}}$, and derive the denotational model from those. An immediate observation is that the deterministic tests suffice: for arbitrary t, the set T of deterministic tests obtained by resolving all nondeterministic choices in t arbitrarily, has the property that for arbitrary B, B **should** t iff $\forall t' \in T.\ B$ **should** t'. Indeed, the set of essential should-tests will be approximately *all deterministic, possibly infinite tests.*

Denotationally, rather than a pair of initial trace and refusal set, as in standard failures, every essential should-test can be represented by a pair of initial trace and *refusal tree*, which is a deterministic, possibly infinite tree whose maximal nodes correspond to success. A system refuses such a tree if it can do some initial prefix but then gets stuck, i.e., cannot reach a successful node any more. (From this point of view, a refusal set is a simple tree whose branches are single actions.) A refusal tree can be represented as the set of traces leading to successful nodes. The "tree failures" of a system B are those pairs (σ, T) with $T \subseteq \mathbf{A}^+$ such that $B \overset{\sigma\rho}{\Longrightarrow} B'$ where ρ is a prefix of some trace in T, and $\nexists \rho v \in T.\ B' \overset{v}{\Longrightarrow}$.

A set of nonempty traces T can be interpreted as a deterministic tree with nodes corresponding to prefixes of elements of T and "success nodes" corresponding to the elements of T. We denote

$$\begin{aligned} \downarrow T &:= \{\varepsilon\} \cup \{\rho \in \mathbf{A}^* \mid \exists v \in \mathbf{A}^*.\ \rho v \in T\} \\ \rho^{-1}T &:= \{v \mid \rho v \in T\}\ . \end{aligned}$$

for, respectively, the *node set* of T and the *remainder of T after ρ*. Note that even if T is empty, $\downarrow T$ contains the element ε, corresponding to the initial node of the tree. Now define

$$F^{++}(B) := \{(\sigma, T) \in \mathbf{A}^* \times \mathcal{P}(\mathbf{A}^+) \mid \exists \rho \in \downarrow T.\ (\sigma\rho, \rho^{-1}T) \in F^+(B)\}$$

Hence F^{++} is indeed a saturation of the model F^+ proposed and rejected earlier, since we can choose $\rho = \varepsilon$ in the above definition. The definition of F^{++} requires nothing for elements of T that do not have ρ as a prefix; e.g. $(a, \{aa, b\})$ is in the F^{++}-semantics of the right-hand system in Fig. 8, since $(aa, \{a\})$ is in its F^+-semantics. We now come to the fully abstract model for $\sqsubseteq_{\mathbf{should}}$.

Theorem 12. *For all $I, S \in \mathbf{L}$, $I \sqsubseteq_{\mathbf{should}} S$ if and only if $F^{++}(I) \subseteq F^{++}(S)$.*

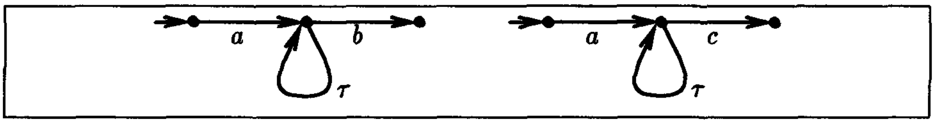

Fig. 9. Two divergent processes

The proof has to be omitted, due to lack of space. Note that the fact that the F^{++}-model is fully abstract does not imply that it is "optimal" in the sense of including no redundant test. This brings us back to the question of "essential tests" discussed above. For instance, the subset of (σ, T) where either $\sigma = \varepsilon$ or $T = \emptyset$ already suffices to establish full F^{++}-inclusion. The issue is an important one, because it concerns the question to what degree $\sqsubseteq_{\textbf{should}}$ can be effectively proved. A detailed investigation, however, is outside the scope of this paper.

7 Concluding Remarks

To evaluate the contribution of this paper it is useful to summarize the main points of other existing work on testing pre-orders and divergences first.

Existing work. We start our comparison with the 'unfair' varieties. In the work on CSP, congruence with respect to hiding is obtained by a catastrophic interpretation of divergences. In the presence of a divergence all information is lost and a process may subsequently show any behaviour. This means that the behaviours of Fig. 9 are failure-equivalent, whereas the the transition systems even contain different actions. Technically, [6] achieves this by inserting the behaviour of the maximally nondeterministic process CHAOS whenever a divergence is encountered. To be able to decide in the semantic model whether maximal behaviour was specified explicitly or introduced by divergences, a refined model that explicitly keeps track of the divergences is presented in [7].

In the must-testing approach followed by De Nicola and Hennessy [9], divergences are also treated explicitly, represented by the constant Ω. Algebraically, Ω plays the role of the underspecified process that can be refined by arbitrary other behaviour, avoiding the drawback of [6]. The related model of (strong) acceptance trees [12, 14] is isomorphic to [7], and therefore also identifies the processes of Fig. 9. An overview of many other characterizations of unfair must-equivalence for transition systems can be found in [8]. Another approach to fairness is described in [13], where fairness is modelled as a structural property of the operator for parallel composition. This interpretation of fairness is compatible with the unfair interpretation of divergences of the underlying semantic model. In this framework a notion of testing is presented that can distinguish between fair and unfair forms of parallel composition.

The unfair interpretation of divergence is useful when one wants to distinguish between livelock and deadlock. This is the case if one, for example, wishes to analyse a specification for the presence of busy waiting-loops and other forms of improductive behaviour. As we have argued in the introduction, however,

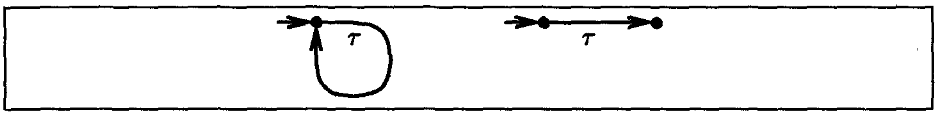

Fig. 10. livelock and (unstable) deadlock

there exist also a number of good reasons where a fair interpretation of divergence is useful. A first study of a fair interpretation of divergence was formulated by Bergstra, Klop, and Olderog in [3]. They make use of the concept of a *stable failure*. This is a failure that occurs in a stable state, i.e. a state without outgoing τ-transitions. The related equivalence *FS* is characterized equationally and a denotational model is constructed that consists of attributes of transition systems (traces, stable failures, and stability of the initial state). Syntactically, divergences are denoted by the constant Δ and the essential congruence is shown to be $\Delta.\tau =_{-} \tau$. This equation requires an outgoing τ-transition in order to abstract away from a divergence, and is therefore referred to as abstraction from *unstable* divergence. It is not sufficiently strong to conclude the equivalence of the behaviours in Fig. 10, which is sometimes paraphrased as *livelock* = (unstable) *deadlock*. These two processes are equated by $\sqsubseteq_{\mathbf{should}}$, as can be easily checked. The authors nevertheless show their equivalence to be fair in a reduced sense, viz. they show the following weaker version of KFAR to hold:

$$\frac{X = a;X + \tau;Y}{X/a = \tau;(Y/a)} \quad (\mathrm{KFAR}^{-})$$

In comparison to KFAR an extra τ appears as the guard of Y. ($\sqsubseteq_{\mathbf{should}}$ fails KFAR^{-}, for the same reason that it did KFAR; see however below.)

Valmari revisits in [22] the *FS*-equivalence of [3], and shows it to be the weakest deadlock-preserving congruence for the LOTOS operators $||_G$ and [>. He also analyses two weaker equivalences that are congruences for other operator sets. Here deadlock is understood in the strong sense, viz. that a deadlock state has no outgoing transitions, including τ-transitions. A reformulation of the conformance testing theory of [4], which introduced $\sqsubseteq_{\mathbf{fmust}}$, using the pre-orders that generate the equivalences mentioned in [22] can be found in [17].

In the very recent [19], Natarajan and Cleaveland independently develop the **should**-testing scenario. They also present a denotational characterisation, and moreover give a topological argument that the difference with **must**-testing is small. Since they do not consider a language, they have no congruence results.

Contributions of this paper. We have introduced a testing pre-order $\sqsubseteq_{\mathbf{should}}$ that is fair in the sense that it ignores divergences that can always be exited. This was done by proposing a new evaluation criterion for tests in the style of De Nicola and Hennessy [9], leading to the definition of a **should**-test. We have shown that with respect to finite behaviours $\sqsubseteq_{\mathbf{should}}$ coincides with $\sqsubseteq_{\mathbf{fmust}}$, the fair version of the must-testing pre-order of De Nicola and Hennessy (and of the failure pre-order of Brookes et al. [6]). Whereas $\sqsubseteq_{\mathbf{fmust}}$, however, is not a pre-congruence with respect to operations that allow abstraction from observable behaviour, such as the hiding operation, we have shown $\sqsubseteq_{\mathbf{should}}$ to be pre-congruent with respect to abstraction. Moreover, we have shown $\sqsubseteq_{\mathbf{should}}$ to be coarsest pre-congruence contained in $\sqsubseteq_{\mathbf{fmust}}$ for abstraction *and* parallel composition with a sufficiently rich synchronisation mechanism. This condition is met by the parallel composition operators of most process algebraic formalisms, such as CCS, CSP, ACP, and LOTOS. Finally, to obtain congruence also with respect to the choice operator $+$ we have introduced the pre-congruence $\sqsubseteq^c_{\mathbf{should}}$, using the standard additional requirement that the instability of the left-hand argument implies that of the right-hand argument. This is also sufficient to obtain congruence with respect to the LOTOS disruption operator [>.

We have demonstrated the fairness properties of $\sqsubseteq^c_{\mathbf{should}}$ in two ways, viz. by showing its compatibility with observation (or weak bisimulation) congruence $\approx^c$, and, more directly, by proving that every fair run of a should-test of a behaviour terminates successfully. The former result is of great interest because $\approx^c$ satisfies KFAR, which represents a very strong notion of fairness. This means that the results of applications of KFAR (or indeed any other sound rule) for $\approx^c$, are inherited by $\sqsubseteq^c_{\mathbf{should}}$. Unfortunately, the combination of fairness with an abstraction-congruent testing pre-order comes at a price: we have also shown that $\sqsubseteq^c_{\mathbf{should}}$ itself does not satisfy KFAR. The premise of KFAR for $\sqsubseteq^c_{\mathbf{should}}$, $X \sqsubseteq^c_{\mathbf{should}} a; X + Y$, equates more behaviours than can be identified by applying fair abstraction when hiding a. This generosity of $\sqsubseteq^c_{\mathbf{should}}$ is also apparent in another way: it does not satisfy the *Recursive Specification Principle* (RSP), i.e. (observationally) guarded equations generally do not have unique solutions modulo $\sqsubseteq^c_{\mathbf{should}}$. (This in fact follows directly from the above results: if $X \sqsubseteq^c_{\mathbf{should}} a; X + Y$ had a unique solution, it would have to be identical (modulo $\sqsubseteq^c_{\mathbf{should}}$) to X in $X =_\theta a; X + Y$, which has been shown to be equivalent to Y/a after hiding a. In other words, RSP would imply KFAR.)

Summarising, $\sqsubseteq^c_{\mathbf{should}}$ answers the long standing question for a fair testing pre-order that is congruent with respect to a standard set of process algebraic operators. This combination, however, implies the loss of the unique solvability of guarded equations. The compatibility with observation congruence, on the other hand, makes this loss less acute. Proofs that require the application of KFAR or RSP should first be carried out in the context of the finer congruence $\approx^c$, only after which the coarser laws of $\sqsubseteq^c_{\mathbf{should}}$ or $\sqsubseteq^c_{\mathbf{should}}$ should be applied.

Future work. The denotational characterisation of $\sqsubseteq_{\mathbf{should}}$ that we have presented is still quite involved, and should be investigated for further possible simplification. This is of some importance as it affects the development of a proof theory for $\sqsubseteq_{\mathbf{should}}$. It also remains to find an axiomatic characterisation for sufficiently well-behaved cases, such as for example regular processes. On the practical side a larger application example should be elaborated that capitalises on the fairness features of $\sqsubseteq_{\mathbf{should}}$ and that cannot be carried out by using only finer equivalences with fair abstraction, such as observation congruence.

Acknowledgement. The first two authors gladly acknowledge fruitful discussions on the topic of this paper with Rom Langerak and Rob van Glabbeek (who put us on the right track towards the coarsest congruence property).

References

1. S. Abramsky. Observation equivalence as a testing equivalence. *Theoretical Comput. Sci.*, 53(3):225–241, 1987.
2. J.C.M. Baeten and W.P. Weijland. *Process Algebra.* Cambridge Univ. Press 1990.
3. J. A. Bergstra, J. W. Klop, and E.-R. Olderog. Failures without chaos: A new process semantics for fair abstraction. In M. Wirsing, ed., *Formal Description of Programming Concepts — III*, pp. 77–103. IFIP, Elsevier, 1987.
4. E. Brinksma. A theory for the derivation of tests. In Aggarwal and Sabnani, eds., *Protocol Specification, Testing, and Verification VIII*, pp. 63–74. Elsevier, 1988.
5. E. Brinksma. Cache consistency by design. In Vuong and Chanson [24], pp. 53–67.
6. S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3):560–599, July 1984.
7. S. D. Brookes and A. W. Roscoe. An improved failures model for communicating processes. In Brookes, Roscoe and Winskel, eds., *Seminar on Concurrency*, vol. 197 of *LNCS*, pp. 281–305. Springer, 1985.
8. R. De Nicola. Extensional equivalences for transition systems. *Acta Inf.*, 24:211–237, 1987.
9. R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Comput. Sci.*, 34:83–133, 1984.
10. R. J. van Glabbeek. The linear time – branching time spectrum II: The semantics of sequential systems with silent moves. In Best, ed., *Concur '93*, vol. 715 of *LNCS*, pp. 66–81. Springer, 1993. Extended abstract.
11. J. F. Groote. Implementation of events in LOTOS-specifications. Master's thesis, University of Twente, 1998. Technical Report, Philips CAM Centre C.F.T.
12. M. Hennessy. Acceptance trees. *J. ACM*, 32(4):896–928, Oct. 1985.
13. M. Hennessy. An algebraic theory of fair asynchronous communicating processes. *Theoretical Comput. Sci.*, 49:121–143, 1987.
14. M. Hennessy. *Algebraic Theory of Processes.* Foundations of Computing Series. MIT Press, Boston, 1988.
15. R. Langerak. A testing theory for LOTOS using deadlock detection. In Brinksma, Scollo and Vissers, eds., *Protocol Specification, Testing and Verification IX*, pp. 87–98. North-Holland 1989.
16. K. G. Larsen and R. Milner. Verifying a protocol using relativized bisimulation. In T. Ottman, ed., *Automata, Languages and Programming*, vol. 267 of *LNCS*, pp. 126–135. Springer, 1987.
17. G. Leduc. Failure-based congruences, unfair divergences, and new testing theory. In Vuong and Chanson [24].
18. R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.
19. V. Natarajan and R. Cleaveland. Divergence and fair testing. To be published in the proceedings of ICALP '95, 1995.
20. D. Park. Concurrency and automata on infinite sequences. In P. Deussen, ed., *Proc. 5th GI Conference*, vol. 104 of *LNCS*, pp. 167–183. Springer, 1981.
21. I. Phillips. Refusal testing. *Theoretical Comput. Sci.*, 50(2):241–284, 1987.
22. A. Valmari. The weakest deadlock-preserving congruence, 1995. To appear in *Information Processing Letters.*
23. W. Vogler. *Modular Construction and Partial Order Semantics of Petri Nets*, vol. 625 of *LNCS.* Springer, 1992.
24. S. Vuong and S. Chanson, eds. *Protocol Specification, Testing, and Verification, XIV*, IFIP Series. Chapman & Hall, 1995.