Introducing Friendly Testing*

David de Frutos-Escrig, Luis Llana-Díaz and Manuel Núñez
Dept. de Informática y Automática
Universidad Complutense de Madrid. E-28040 Madrid. Spain.
e-mail: {defrutos,llana,manuelnu}@dia.ucm.es

(Draft Version)

Abstract

We present a new testing semantics, called *friendly testing*, whose main property is that the induced preorder between processes $\sqsubseteq_{\rm fr}$ is consistent with the conformance relation, and so we have, for instance, $a \oplus b \sqsubseteq_{\rm fr} a \sqsubseteq_{\rm fr} a + b$. The new theory is strongly based on De Nicola & Hennessy's work on testing, and the structure of the paper closely follows that of Hennessy's book on the subject. Friendly tests are defined exactly as in his famous book, except that internal actions are not owed. However, this restriction is not enough and we also have to relax the conditions to pass a test in order to obtain the desired notion of friendly testing. Thus we obtain a new testing semantics and a new preorder between processes which is strictly weaker than the relation $\sqsubseteq_{\rm must}$. As a consequence, a fully abstract denotational semantics can be obtained as a quotient algebra of the corresponding construction for the *must* semantics, and the addition to the complete axiomatization of the latter of a single conformance axiom, gives us a complete axiomatization of our friendly testing semantics.

1 Introduction

Since its introduction in [dNH84, Hen88] *Testing Semantics* has been broadly studied and used as a natural way to define an observational semantics with a reasonable power to distinguish semantically different processes. It is defined by observing the operational semantics of processes by means of *tests*. Tests are just processes which may execute a new action ω reporting *success* of the test application. To define the application of a test to a process, we consider the different computations of the experiment system which is obtained by composing in parallel the test and the tested process. We say that a computation is *successful* if there exists a step in the computation such that the associated test can execute the action ω. Since it is possible that some, but not all of the computations may succeed, we can distinguish three families of tests for each process: those that all computations are unsuccessful, those that some computations are successful, and those that all computations are successful. From the last two classes of tests we define two different semantics which are called *may* and *must* semantics. A process P *may* pass a test T (in short P *may* T) if the composition of P and T has at least a successful computation, while P *must* pass T (in short P *must* T) if every computation is successful. By combining these two semantics, we can obtain a third one: the *may-must* semantics. Two processes are (*may*, *must*) equivalent iff they pass (in the corresponding sense) the same tests. In addition to these equivalences, we obtain respective partial orderings between processes: Q is *better* than P, if any test passed by P is also passed by Q. As a matter of fact, the previous equivalence notions are just the ones induced by the preorders, which could also be studied by themselves. This is usually done in two different ways: first, to define the notion of approximation used to build the semantics of recursive processes by applying the well known technique of minimal fixpoints; and second to formalize the notions of refinement and conformance (see e.g. [BSS86]). From the latter point of view we would expect $P \oplus Q \sqsubseteq P$, whenever the initial actions of P and Q are disjoint, $P \sqsubseteq P + Q$. It is at this level that the term *better* finds its justification: we would expect that $P \sqsubseteq Q$ exactly when P can be reasonably considered to be worse than Q. Usually the testing preorder is only used for the first of the two mentioned purposes. This is justified by the difficulties found when trying to interpretate the testing preorder as a *worse than* relation.

*Or how to test processes without punishing them when they are able to execute actions.

Research supported in part by the CICYT project TIC 94-0851-C02-02.

It is well known that, for divergence-free processes, the different testing preorders and equivalences are related in the following way:

- $P \sqsubseteq_{\text{must}} Q \implies Q \sqsubseteq_{\text{may}} P$
- $P \approx_{\text{may}-\text{must}} Q \iff P \approx_{\text{must}} Q$ and $P \approx_{\text{must}} Q \implies P \approx_{\text{may}} Q$

As a consequence, the axiom $P \sqsubseteq_{\text{must}} P + Q$ is not fulfilled at all, and so the testing preorder does not capture the notion of conformance.

We have investigated which are the different reasons implying this undesirable fact and the ways to define a new notion of testing equivalence correcting this situation. In the remainder of this introduction we will concentrate ourselves on the *must* preorder $\sqsubseteq_{\text{must}}$ over divergence-free processes. Actually, we will usually write $\sqsubseteq$ instead of $\sqsubseteq_{\text{must}}$.

On the one hand, we have concentrated ourselves on how tests, and the passing of tests, are defined. Thus, we have found that De Nicola & Hennessy's [dNH84] tests have the power to *punish* processes being able to execute actions. So, $a \not\sqsubseteq a+b$, since the former passes the test $(a;\omega)+(b;\texttt{STOP})$, while the latter does not. We are interested on a testing framework in which tests cannot punish the processes when they are able to execute some action. This is why we call friendly testing, and we will denote it by $\sqsubseteq_{\text{fr}}$ the new preorder that we are going to introduce, to our new testing scenario. Intuitively, friendly tests can just *reward* with success when the desired traces are executed, but not to *punish* with a failure when some other traces are executable by the process to be tested. A possible interpretation of this fact leads to the conclusion that the problem comes because we allow both successful (ω) and unsuccessful (STOP) terminations in tests, but this is not the case. In fact, if we restrict the set of tests to *always successful* tests, i.e. tests whose *leaves* are always labeled by ω, nothing is gained, since we could always assume the existence of a new action *reject*, to be read as failure, such that any STOP (failure) termination in the original tests could be simulated by a *reject*$;\omega$ termination. This means that we cannot just restrict the family of tests to reach our goal, but also the definition of test passing must be changed, if we desire to obtain $a \sqsubseteq_{\text{fr}} a + b$. In the following section we will show which is the adequate modification and how it can be intuitively justified.

There is also an additional modification whose significance at the intuitive level justifies an extended comment in this introduction. When defining the classic notion of test, and the interaction between processes and tests, Hennessy [Hen88, pg. 67] says:

> *... An obvious possibility is to let* $\rightarrow$ *be the least relation which for an arbitrary* $a \in \mathbf{Act}$*, satisfies*
>
> $$t \xrightarrow{a} t',\ p \xrightarrow{a} p' \ \text{ implies } \ t \parallel p \rightarrow t' \parallel p'$$
>
> *so that they interact by performing the same action. This gives the experimenter complete control over the process ... Unfortunately this control works both ways since the process also has the same control over the experimenter. To break this symmetry, and allow the experimenter a natural independence from the process it is investigating we allow it to use a special action* 1*, such that we have*
>
> $$t \xrightarrow{1} t' \ \text{ implies } \ t \parallel p \rightarrow t' \parallel p$$

We could rephrase this sentence by saying that tests may be nondeterministic in order to avoid to be always controlled by the tested process. Internal actions, labeled by 1, are a way to introduce such a nondeterministic behavior, and at the moment they appear in [Hen88] they are fully justified, since only the external choice operator had been considered. However, the internal choice operator is later introduced [Hen88, pg. 89] and once it appears we could think that internal actions are not needed any more in order to have nondeterministic choices in tests. But, even in the presence of internal choices, the possibility of having internal actions increases the discriminatory power of tests, which does not seem to have a clear intuitive justification. Thus, we will preclude the use of internal actions in tests eliminating tests like $t = (1\,;\omega) + (b\,;\texttt{STOP})$, which also verifies that a passes t while $a + b$ does not pass t.

It could be thought that all our problems would be solved just considering tests without internal actions, and indeed this was our first attempt, but this is far from being true. We would get that STOP is the minimum element (if we do not allow divergent processes) since STOP only passes trivial tests. Moreover, $a\,;\texttt{STOP} \sqsubseteq_{\text{fr}} a\,;P$ and so on; but unfortunately we would not solve the conformance problem. For instance, a would not be *worse* than $a+b$ because we still have the test $(a;\omega)+(b;f;\omega)$, which is passed by the former process but not by the latter.

Next we present a **real** *real life* example. *Matthew* and his new *friend* went to play tennis at the Concurrency World Center. After the match, they were really thirsty, so they went to the electronic bar. There they found two changes: first, besides the two old single product machines delivering orange juice and cola, that were donated by *C.A.R. Hoare* some years ago, there is a brand-new machine delivering both products; the second change is that there is a warning in the wall informing that because of a strike there are some problems to refill the machines in time. Due to their sport clothes, the two men only have one single coin to expend each.

After studying the situation Matthew told his friend: "Although I am rather thirsty, and so it does not matter what to drink, since I slightly prefer orange juice I will put my coin in the old machine. I do not trust too much these new machines." But his friend told him: "I do not understand why you do not try in the new machine. I have heared that it has new technology, so that you can be more confident on it". Matthew replied: "I assume that if they are out of some product, both machines will be out of it. Since I prefer orange juice it would have no sense to push just the cola button. Perhaps I could try to push both buttons simultaneously, but in this case I could lose my coin even if the machine has orange juice, if it decides to try to give me a cola, and finally, it has not colas[1]". Then, the friend said: "That is exactly what I will do, I've heard that these new machines have the ability to decide in the right way, whatever the situation is". They put the coins in the machines at the same time and pushed the buttons as they had decided. Afterwards, Matthew was reading the following message: "We are sorry, but the machine is out of bottles. Please, ask the company for you money", while his friend was drinking his cola with pleasure.

Let us now discuss the subject from the *denotational* semantics point of view. It is well known that *must* testing semantics coincides with *failure* semantics [Hoa85]. Both can be alternatively defined by means of *acceptance trees* [Hen85]. Since friendly testing will be weaker than *must* testing, we can build our denotational semantics from acceptance trees.

In order to define the labels of the nodes of a semantic process in the *must* case, one has to consider the *acceptance sets* of the process, and to apply convex and union closures. As a consequence all the traces of the process are preserved, which is a must since $\approx_{\text{must}} \implies \approx_{\text{may}}$. Instead in our case we will have, for instance, $\texttt{STOP} \approx_{\text{fr}} \texttt{STOP} \oplus b$, since to distinguish between both processes in the *must* testing framework we need to apply a test like $t = (1\,;\omega) + (b\,;\texttt{STOP})$ which punishes the process $\texttt{STOP} \oplus b$ due to its capability to execute the action b, and this kind of tests are not allowed in our friendly framework. Thus we need to apply a *minimality* condition over the *acceptance sets* of a process by removing those sets that are not minimal. Thus, the process whose only acceptance set is the empty one will be the bottom element of the semantic domain, as far as we restrict ourselves to divergence-free processes.

In Section 4 we will briefly present the new denotational semantics.

Finally, we discuss the algebraic characterization of our new semantics. In [Hen88, pg. 93], the axiomatization for finite processes under the *may-must* semantics is presented. In order to obtain the corresponding axiomatization for the *must* semantics, it is enough to add the axiom $P \oplus Q \leq P$ which indicates that nondeterministic processes are worse than those more deterministic. We were interested in the definition of a new testing semantics under which $a \sqsubseteq_{\text{fr}} a + b$ holds. This is satisfied by the *may* semantics, but it is not by the *must* semantics. Thus, it is necessary to add a new axiom $P \leq P + Q$, which is applicable whenever the sets of *initial* actions of P and Q are disjoint. With this addition we obtain a complete and sound axiomatization of friendly testing. Section 5 will be devoted to the study of this axiomatization.

2 Friendly Testing: Definitions and Justification

Following [Hen88] we will first concentrate on finite processes, over the signature $\Sigma^2 = \{\texttt{STOP}, +, \oplus\} \cup \{a\,; \mid a \in \mathbf{Act}\}$. Although in the following section we will give a general definition of friendly testing which closely follows the classic operational pattern, in order to avoid obscure technicalities we start by studying a particular case. By now, we will just consider finite processes in *normal form*. They are defined by the following BNF expression:

$$NFP ::= \bigoplus_{A \in \mathcal{A}} P_A, \qquad \text{where } P_A \in DP ::= \sum_{a \in A} (a\,; P_a), \text{ and } P_a \in NFP$$

where $\mathcal{A} \subseteq \mathcal{P}_f(\mathbf{Act})$, $\mathcal{A}$ is non-empty, and $\bigoplus$ and $\sum$ are the obvious generalizations of $\oplus$ and $+$ to an arbitrary (but finite) number of arguments. By convention $\bigoplus_{A \in \{\varnothing\}}$ represents the process

[1] Note that this would be indeed the situation in a must testing like scenario.

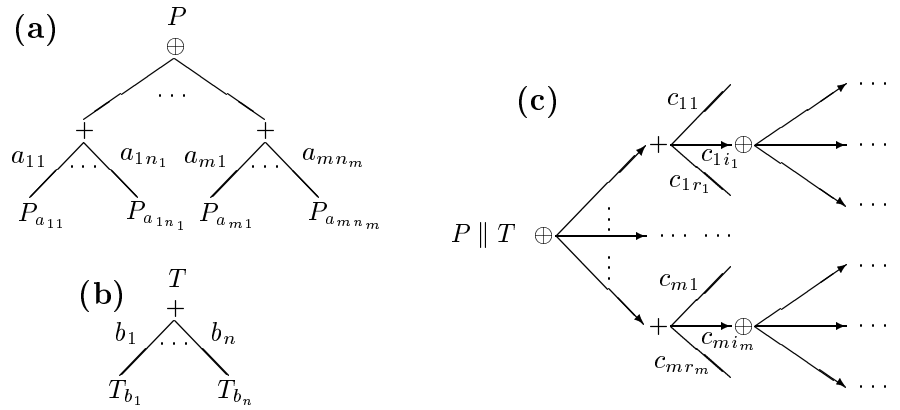

Figure 1: Normal Forms, Deterministic Tests, and P fr T.

STOP. We call *initially deterministic processes* those defined by the nonterminal symbol DP, while *deterministic processes* are those in which all the occurrences of $\bigoplus$ are trivial, which means that the indexing sets of actions are singletons. In order to simplify the notation, we will usually omit trivial ocurrences of operator $\bigoplus$. Thus, initially deterministic processes will be seen as a particular class of normal forms. Since we have used a regular expression to define our normal forms, it is not the case, like in the classical notion of normal form, that all the continuations after different occurrences of the same action a in several states $A \in \mathcal{A}$ must be the same.

As usually, tests will be just processes over the alphabet $\mathbf{Act} \cup \{\omega\}$. For the same reasons that for processes, by now we will consider a restricted version of tests: those finite deterministic tests with acceptance actions at the end of any trace. We will show that this family of deterministic tests is a basis for the full family of tests, in the sense that whenever two processes are not friendly equivalent then there exists a deterministic test distinguishing them. Thus, *deterministic tests* are defined by the BNF expression:

$$DT ::= \omega \mid \sum_{a \in A \subseteq \mathbf{Act}} a \,;\, DT$$

In Figure 1 we give a graphical representation of **(a)** normal forms and **(b)** deterministic tests. Note that we could see deterministic tests as a particular case of normal forms for which $|\mathcal{A}| = 1$.

Definition 2.1 Given a normal form process P and a deterministic test T, we say that P *friendly passes* T iff

1. $T = \omega$, or
2. $P = \bigoplus_{A \in \mathcal{A}} P_A$, and for each $A \in \mathcal{A}$, P_A fr T, or
3. $P = \sum_{a \in A}(a \,;\, P_a)$, $T = \sum_{b \in B}(b \,;\, T_b)$, and there exists some $a \in A \cap B$ such that P_a *friendly passes* T_a.

□

Note that, although this definition is recursive, it is sensible since it is well founded, as far as we only consider finite tests.

Let us note that the first two cases of this definition are equivalent to those for the classical must testing. The differences appear in the last case. If we are testing a generalized external choice, and the test offers several of the actions in the choice then we do not impose that *all* the possible computations must succeed; on the contrary, we only impose that the computations starting with one of the (common) offered actions succeed. In Figure 1 **(c)** we illustrate this definition. In order to friendly pass the test, it is enough that all the computations that are obtained by following the arrows, succeed.

Although we are more interested on the (good) properties of the induced semantics than on an intuitive justification of our notion of testing, we will try to justify the definition by itself.

For it we begin by comparing our definition with the plain must testing by means of an illustrative example.

Example 2.2 Let us consider the following processes $P_1 = a;P_a$, $P' = (a;P_a)+(b;P_b)$, $P_2 = P_1 \oplus P'$, and $P'' = (a\,;P_a) \oplus (b\,;P_b)$. Let $T = (a\,;\omega) + (b\,;\mathtt{STOP})$. It is easy to check that under the classic notion of testing we have P_1 *must* T but P_2 *must not* T. The reason for this is that in order to have P_2 *must* T we need all the computations of $P_2 \parallel T$ to be successful. In particular, this must be true for the computations of $P' \parallel T$. But when we apply a test like T, offering several actions that could be executed by the tested process, it does not matter if the involved choices in this process are either internal of external. So we have P' *must* T iff P'' *must* T. [2] Such a behavior could be justified by the assumption of testing being only a final way to observe the behavior of the tested process. As a matter of fact, and even if that would have no effect in the definition of passing tests, [Hen88] does not label the transitions of experimental systems of the form $P \parallel T$. As a consequence, the computations tree corresponding to both $P' \parallel T$ and $P'' \parallel T$ are equivalent. On the contrary, we consider that the test is not the final way to observe the behavior of the process. Thus, we do not hide the synchronization actions, and so we maintain some information which allows us to distinguish $P' \parallel T$ and $P'' \parallel T$. This is indeed the case, because if we apply the classic (expansion) axioms for the parallel operator we obtain on the one hand $P' \parallel_{\mathbf{Act}} T \approx (a\,;(P_a \parallel_{\mathbf{Act}} \omega)) + (b\,;(P_b \parallel_{\mathbf{Act}} \mathtt{STOP}))$, while on the other hand $P'' \parallel_{\mathbf{Act}} T \approx (a\,;(P_a \parallel_{\mathbf{Act}} \omega)) \oplus (b\,;(P_b \parallel_{\mathbf{Act}} \mathtt{STOP}))$. So, under our notion of friendly testing we have that P and P' can be distinguished by the test T. Thus we have P_2 *friendly passes* T, and in fact it is the case that for any test T' we have P_1 *friendly passes* T' iff P_2 *friendly passes* T'. □

Then, our justification of the way friendly test passing is defined is that the observer maintains the control, even after a test is applied, as far as external choices remains, as it is the case for process P' in the example above. In such a case the observer can select the action to be executed taking into account when a success (or more exactly, when a set of successful computations) will be reached. The existence of such an action is enough to pass the test. In this way the computations leading to a failure could possibly be avoided, and a test that is not passed in the classic way, could be passed in the friendly way.

The reader could think this new notion of passing tests is much more involved than the classic one, but we advocate that, at least for normal form processes, this is not the case. As a matter of fact, if we consider a recursive definition of the classical notion of must test passing for normal form processes, we see that it can obtained from our definition of friendly test passing just by changing the existential quantification in the third condition of Definition 2.1 by a universal quantification. Anyway, one could insist on the fact that to impose that all the computations have to be successful is simpler than to check our (apparently) more complicated condition. Actually, this is not the case. In order to check any of these notions we must (in the worst case) explore the full tree of computations; sometimes to check must testing will be faster (when the test fails), and sometimes it is faster to check friendly testing (when the test is successfully passed).

Next we present a collection of examples showing the strength and properties of our new notion of testing.

Example 2.3

1. $P \oplus Q \sqsubseteq_{\mathrm{fr}} P$. This is because we already had $P \oplus Q \sqsubseteq P$, and in general we have $P \sqsubseteq Q \implies P \sqsubseteq_{\mathrm{fr}} Q$. As a particular case we have $a \oplus (a+b) \sqsubseteq_{\mathrm{fr}} a+b$. On the contrary, we have $(a\,;c) \oplus (b\,;c) \not\sqsubseteq_{\mathrm{fr}} a+b$, since the test $(a\,;c\,;\omega) + (b\,;c\,;\omega)$ is friendly passed by the former process but it is not friendly passed by the latter.

2. $a \sqsubseteq_{\mathrm{fr}} a + b$. Note that under our notion of testing we cannot *punish* the second process when applying a test like $(a\,;\omega) + (b\,;c\,;\omega)$. Even if the computation executing b will not succeed, we can select instead the computation executing a, which immediately succeeds (note that this test is not passed by the second process in the must sense). Actually, we have $P \sqsubseteq_{\mathrm{fr}} P + Q$ whenever the sets of actions that can be executed by P and Q in their first steps are disjoint.

3. $a \oplus (a + b) \approx_{\mathrm{fr}} a$, because on the one hand we have $a \oplus (a + b) \sqsubseteq_{\mathrm{fr}} a$, again as a particular case of the property asserted in 1. On the other hand, note that $a \oplus a \approx_{\mathrm{fr}} a$ and then we apply the fact that all the operators of the language are monotonic with respect to the friendly testing relation.

□

[2] It is clear that P' and P'' can be distinguished under plain must testing by a test like $a\,;\omega$. In fact, if this would not be the case, they could neither be distinguished under friendly testing. However, it is interesting to observe that P' and P'' cannot be distinguished under must testing by a test like T that offers both a and b; on the contrary, under friendly testing we are able to distinguish P' and P'' by such a test.

2.1 Friendly Testing for arbitrary finite processes and tests

In this section we will consider arbitrary finite processes and tests. We will work on an arbitrary set of actions **Act**.

Definition 2.4 The set of *finite processes*, denoted by *Proc*, is defined as the set of expressions given by the following BNF-expression:

$$P ::= \mathtt{STOP} \mid a\,;P \mid P + P \mid P \oplus P$$

where $a \in \mathbf{Act}$. For the sake of clarity we will omit trailing occurrences of STOP. □

The operational semantics of the language is defined as in [Hen88]:

$$\frac{}{a;P \xrightarrow{a} P} \qquad \frac{}{P \oplus Q \rightarrowtail P} \qquad \frac{}{P \oplus Q \rightarrowtail Q}$$

$$\frac{P \xrightarrow{a} P'}{P+Q \xrightarrow{a} P'} \qquad \frac{Q \xrightarrow{a} Q'}{P+Q \xrightarrow{a} Q'} \qquad \frac{P \rightarrowtail P'}{P+Q \rightarrowtail P'+Q} \qquad \frac{Q \rightarrowtail Q'}{P+Q \rightarrowtail P+Q'}$$

The following conventions will be used:

$P \xrightarrow{a}$ stands for $\exists P' :\ P \xrightarrow{a} P'$, $\quad P \not\xrightarrow{a}$ stands for $\nexists P' :\ P \xrightarrow{a} P'$,
$P \not\longrightarrow$ stands for $\nexists P', a :\ P \xrightarrow{a} P'$,
$P \rightarrowtail$ stands for $\exists P' :\ P \rightarrowtail P'$, $\quad P \not\rightarrowtail$ stands for $\nexists P' :\ P \rightarrowtail P'$, and
$\rightarrowtail^*$ stands for the transitive and reflexive closure of $\rightarrowtail$.

Moreover, for $s = a_1, \ldots, a_n$ we write $P \stackrel{s}{\Longrightarrow} P'$ if there exist $P_1, \ldots, P_n, P'_1, \ldots, P'_n$ such that $P \rightarrowtail^* P_1 \xrightarrow{a_1} P'_1 \rightarrowtail^* P_2 \cdots P_n \xrightarrow{a_n} P'_n \rightarrowtail^* P'$.

Tests are just finite processes over the alphabet $\mathbf{Act} \cup \{\omega\}$, and the previous operational semantics is also valid for tests. We define the operational semantics of *experimental systems*, $P \parallel T$, by

$$\frac{P \xrightarrow{a} P' \ \wedge\ T \xrightarrow{a} T'}{P \parallel T \xrightarrow{a} P' \parallel T'} \qquad \frac{P \rightarrowtail P'}{P \parallel T \rightarrowtail P' \parallel T} \qquad \frac{T \rightarrowtail T'}{P \parallel T \rightarrowtail P \parallel T'}$$

Let us remark that, in contrast with the classical testing semantics, we do not hide the actions that experimental systems execute. Next we consider the tree of (complete) computations of $P \parallel T$, whose nodes are labeled by experimental systems $P' \parallel T'$. Now, we introduce some auxiliary concepts for the definition of *friendly* testing.

Definition 2.5 Let P be a process. We say that P is *stable* iff $P \not\rightarrowtail$. Moreover, given a test T we say that a configuration $P \parallel T$ is *stable* iff $P \parallel T \not\rightarrowtail$.

Given a process P and $a \in \mathbf{Act}$, we define the process P *after the execution of* the action a, denoted by P/a, as $P/a = \bigoplus\{P' \mid P \stackrel{a}{\Longrightarrow} P'\}$. □

Definition 2.6 (*Friendly Test Passing*).
Given a process P and a test T, we say that P fr T if the following conditions hold:

- If $P \parallel T$ is stable, then either $T \xrightarrow{\omega}$, or there exists some $a \in \mathbf{Act}$ such that $P \parallel T \xrightarrow{a}$ and (P/a) fr (T/a).
- If $P \parallel T$ is not stable, then for each P', T' such that $P \parallel T \rightarrowtail P' \parallel T'$ we have P' fr T'.

□

Let us remark that the first condition in the previous definition is equivalent to the following one: *If $P \parallel T$ is stable, then either $T \xrightarrow{\omega}$, or there exist $P', T', a \in \mathbf{Act}$ such that $P \parallel T \xrightarrow{a} P' \parallel T'$, and for all P'', T'' such that $P \parallel T \xrightarrow{a} P'' \parallel T''$, we have P'' fr T''.* It is easy to check that this definition is an extension of the one for normal forms (Definition 2.1).

Proposition 2.7 Let P be a normal form process and T be a deterministic test. We have P *friendly passes* T under Definition 2.1 iff P fr T under Definition 2.6.

Next we present some properties of the general definition of friendly testing. The proofs, by structural induction, are easy.

Proposition 2.8 Let P, P_1, P_2 be processes, and T, T_1, T_2 be tests. Then, the following properties hold:

1. P fr ω.
2. P fr $T_1 \oplus T_2$ iff Pfr both T_1 and T_2.
3. $P_1 \oplus P_2$ fr T iff both P_1 and P_2 *friendly pass* T.
4. If P_1, P_2 are stable, and $\{a \mid P_1 \xrightarrow{a}\} \cap \{b \mid P_2 \xrightarrow{b}\} = \varnothing$ then for any test T we have $P_1 + P_2$ fr T iff P_1 fr T or P_2 fr T.
5. If P *must* T then P fr T.

As usual, fr induces a preorder relation between processes:

Definition 2.9 Let P, Q be processes. We write $P \sqsubseteq_{\text{fr}} Q$ iff for all test T we have P *friendly passes* $T \implies Q$ *friendly passes* T. Besides, we write $P \approx_{\text{fr}} Q$ iff $P \sqsubseteq_{\text{fr}} Q$ and $Q \sqsubseteq_{\text{fr}} P$. □

Concluding this section we present a result showing that deterministic tests constitute indeed a set of *essential* tests.

Proposition 2.10 Let P, Q be processes. $P \sqsubseteq_{\text{fr}} Q$ iff for any deterministic test T we have P *friendly passes* $T \implies Q$ fr T.

3 Alternative Characterization of Friendly Testing

In this section we provide an alternative characterization of the friendly testing preorder given in Definition 2.9. This characterization is based on a modification of acceptance sets [Hen88]. These adapted acceptance sets are called *friendly acceptance sets*. The last result of the previous section will be very helpful in order to prove that the preorder induced by the alternative characterization is equivalent to $\sqsubseteq_{\text{fr}}$.

Definition 3.1 Let P be a process, and $s = a_1, \ldots, a_n$ a (possibly empty, denoted by ϵ) sequence of actions. Then we define the following concepts:

- The *set of initial actions* of P as $S(P) = \{a \mid P \stackrel{a}{\Longrightarrow}\}$.
- The *acceptance sets* of P *after* s as $\mathcal{A}(P, s) = \{S(P') \mid P \stackrel{s}{\Longrightarrow} P'\}$.
- The *friendly acceptance sets* of P as:

$$\mathcal{F}(P) = \{A \in \mathcal{A}(P, \epsilon) \mid \nexists A' \in \mathcal{A}(P, \epsilon) : \ A' \subsetneq A\}$$

□

Note that we have defined friendly acceptance sets of a process only for the empty trace. Anyway, friendly acceptance sets for each trace $s = a_1, \ldots, a_n$ could be defined as the friendly acceptance sets of the process $((P/a_1)/a_2) \cdots /a_n$. By comparing the friendly acceptance sets of processes we can obtain a new preorder. This preorder is obtained by adapting the preorder for acceptance sets to the new setting.

Definition 3.2 Let P, P' be processes. We write $P \ll_{\text{fr}} P'$ iff for all $A' \in \mathcal{F}(P')$ there exists $A \in \mathcal{F}(P)$ such that

- $A \subseteq A'$, and
- for all $a \in A$, $P/a \ll_{\text{fr}} P'/a$. □

Now we present the main result of this paper, where we prove that the preorders $\sqsubseteq_{\text{fr}}$ and $\ll_{\text{fr}}$ coincide. We split this theorem in two parts.

Theorem 3.3 Let P, P' be processes. Then $P \sqsubseteq_{\mathrm{fr}} P' \implies P \ll_{\mathrm{fr}} P'$.
Proof: The proof will be done by contraposition, and by structural induction over processes. So let us suppose $P \not\ll_{\mathrm{fr}} P'$. Then there exists some $A' \in \mathcal{F}(P')$ such that one of the following conditions hold:[3]

- $\forall\, A \in \mathcal{F}(P) :\ A \not\subseteq A'$, or
- $\forall\, A \in \mathcal{F}(P) : \Big(A \subseteq A' \implies \exists\, a_A \in A :\ P/a_A \not\ll_{\mathrm{fr}} P'/a_A \Big)$.

In the first case we construct a set S including for each $A \in \mathcal{F}(P)$ an action in $A - A'$. Then, if we consider the deterministic test $T = \sum_{a \in S} a\,;\omega$, we get $P' \;\mathsf{fr}\; T$ but P does not.

In the second case, by induction hypothesis we can assume that for each $A \subseteq A'$ there exists T_{a_A} such that $P'/a_A \;\mathsf{fr}\; T_{a_A}$, but P/a_A does not. Besides, for each $A'' \in \mathcal{F}(P)$ such that $A'' \not\subseteq A'$ we take $a_{A''} \in A'' - A'$, and we consider the deterministic test

$$T = \sum_{\substack{A \subseteq A' \\ A \in \mathcal{F}(P)}} a_A\,;T_{a_A} + \sum_{\substack{A'' \not\subseteq A' \\ A'' \in \mathcal{F}(P)}} a_{A''}\,;\omega$$

It is easy to check that $P \;\mathsf{fr}\; T$ but P' does not, since each P/a_A does not *friendly pass* the test T_{a_A}. □

Theorem 3.4 Let P, P' be processes. Then $P \ll_{\mathrm{fr}} P' \implies P \sqsubseteq_{\mathrm{fr}} P'$.
Proof: Let T be a deterministic test such that $P \;\mathsf{fr}\; T$. We will prove, by induction on the depth of T, that P' also$\mathsf{fr}T$.

If $\mathrm{depth}(T) = 1$ then $T = \omega$ and the result is trivial. Otherwise $T = \sum_{i \in I} a_i\,;T_i$. Then, in order to check that $P' \;\mathsf{fr}\; T$ we have to show that for each $A' \in \mathcal{A}(P', \epsilon)$ there exists some $a' \in A'$ with $a' = a_i$, for some i, and such that $P/a' \;\mathsf{fr}\; T_i$. Given that for any $A' \in \mathcal{A}(P', \epsilon)$ there exists $A'' \in \mathcal{F}(P')$ such that $A'' \subseteq A'$, it is enough to prove the previous property for the sets in $\mathcal{F}(P')$.

Given that $P \ll_{\mathrm{fr}} P'$, we have that for any $A' \in \mathcal{F}(P')$ there exists $A \in \mathcal{F}(P)$ with $A \subseteq A'$ such that for all $a \in A :\ P/a \ll_{\mathrm{fr}} P'/a$. By the hypothesis of the theorem we have $P \;\mathsf{fr}\; T$, and thus there exists $a \in A$, with $a = a_i$ for some i, such that $P/a \;\mathsf{fr}\; T_i$. Therefore we can take $a' = a = a_i$, and by applying induction hypothesis we obtain that $P'/a' \;\mathsf{fr}\; T_i$, and thus we conclude that $P' \;\mathsf{fr}\; T$. □

Note that in the previous two theorems we have used that deterministic tests have the same discriminatory power than the whole family of sets (i.e. Theorem 2.10). The combination of the previous results gives the final result.

Corollary 3.5 Let P, P' be processes. Then $P \ll_{\mathrm{fr}} P' \iff P \sqsubseteq_{\mathrm{fr}} P'$.

It is interesting to note the close relation between the alternative characterization of the must testing semantics (based on acceptance sets) and that of friendly testing semantics. For it, we begin by presenting an alternative characterization of the former derived from that of friendly testing.

Definition 3.6 We write $P \ll' P'$ iff

- $\forall\, A' \in \mathcal{F}(P')\ \exists\, A \in \mathcal{F}(P) : A \subseteq A' \ \wedge\ \forall\, a \in A' :\ P/a \ll' P'/a$,
- $S(P) \supseteq S(P')$, and
- $\forall\, a \in S(P') :\ P/a \ll' P'/a$. □

Next we present the previously announced result, showing that $\ll'$ alternatively characterizes the classical must testing equivalence.

Theorem 3.7 Let P, P' be processes. $P \sqsubseteq P'$ iff $P \ll' P'$.
Proof: It is enough to prove

$$P \ll' P' \quad \iff \quad \forall\, A' \in \mathcal{A}(P', s)\ \exists A \in \mathcal{A}(P, s) :\ A \subseteq A'$$

The left to right implication is straightforward by induction over the length of s; and for the right to left implication it is enough to notice

$$A \in \mathcal{A}(P/a, s) \iff A \in \mathcal{A}(P, as)$$

□

[3] As a matter of fact the first case is just a particular case of the second, but we think that by considering first that particular case we contribute to make the proof more easily understandable.

This means that the difference between must and friendly testing when comparing two processes P and P' comes from the more restrictive conditions in this last characterization of the must ordering. Thus, if $P \sqsubseteq_{\text{fr}} P'$ but $P \not\sqsubseteq P'$ we have either:

1. $\forall A'_n \in \mathcal{F}(P')\ \exists A_n \in \mathcal{F}(P) :\ A_n \subseteq A'_n\ \wedge\ \forall a \in A_n :\ P/a \ll_{\text{fr}} P'/a$ in such a way that there exists $a \in \left(\bigcup_{\mathcal{F}(P')} A'_n\right) - \left(\bigcup_{\mathcal{F}(P)} A_n\right)$ such that $P/a \not\ll' P'/a$.

   Example: $P = a \oplus (b\,;c)$, $P' = a + b$. $P/b = c$, $P'/b = \texttt{STOP}$. So, we have $P/b \not\ll' P'/b$ (note that $P/b \ll_{\text{fr}} P'/b$).

2. $S(P) \not\supseteq S(P')$.

   Example: $P = a$, $P' = a + b$.

3. $\exists a \in (S(P') - \bigcup_{\mathcal{F}(P)} A') :\ P/a \not\ll' P'/a$.

   Example: $P = a \oplus (b\,;c)$, $P' = a \oplus (a + b)$. $P/b = c$, $P'/b = \texttt{STOP}$.

It is easy to check that case 2 can be considered as a particular case of either 1 or 3. Besides, it is clear that several of these cases can be presented at the same time when comparing a pair of processes. Then, we can say that the stronger power of must testing comes from the possibility to check (and to punish for it) traces, as well as the continuations after those actions that are not informative for friendly testing.

4 Denotational Semantics

From the former characterization of the friendly ordering is easy to define a model of friendly acceptance trees. These can be considered as a restrictive class of acceptance trees. Besides, our new model can also be obtained (up to isomorphism) as a quotient algebra from that of acceptance trees, **fAT**, in [Hen88, pg. 78].

Definition 4.1 Let $\mathcal{A}$ be a set of acceptance sets.

1. Given $A \in \mathcal{A}$, we say that it is *minimal* in $\mathcal{A}$ if $\nexists A' \in \mathcal{A} : A' \subsetneq A$.
2. We say that $\mathcal{A}$ is *friendly* when it is *free of non-minimal actions*, that is, for any $a \in \bigcup_{\mathcal{A}} A$ there exists a minimal set $A \in \mathcal{A}$ with $a \in A$.
3. We define the set of *friendly acceptance trees*, denoted by **fFAT**, as the set of acceptance trees whose nodes are labeled by friendly acceptance sets.

□

The order relation $\leq_{\mathbf{fFAT}}$ between friendly acceptance trees is defined as follows:

Definition 4.2 Given $t, t' \in \mathbf{fFAT}$ we have $t \leq_{\mathbf{fFAT}} t'$ whenever

1. either $t = \bullet$, the trivial tree, or
2. for any $A' \in \mathcal{A}(t')$ there exists $A \in \mathcal{A}(t)$ satisfying:
   - $A \subseteq A'$, and
   - for any $a \in A$, $t(a) \leq_{\mathbf{fFAT}} t'(a)$.

□

In order to define the interpretations of all the operators, we first apply the interpretations over **fAT**, and then we *friendly normalize* the results by removing at any node all the states including non-minimal actions, and also the outgoing branches labeled by those actions. Some simple examples are presented in Figure 2.

Now it is not difficult to prove that the interpretations of all operators are monotonic functions (with respect to $\leq_{\mathbf{fFAT}}$) and thus $\langle \mathbf{fFAT}, \leq_{\mathbf{fFAT}} \rangle$ is a Σ-*po algebra*.

In order to directly construct **fFAT** as a quotient algebra from **fAT** we have just to apply friendly normalization over arbitrary acceptance trees. However, when we consider the corresponding ordered algebras we have that $(\mathbf{fFAT}, \leq_{\mathbf{fFAT}})$ is no more obtained by applying the quotient transformation.

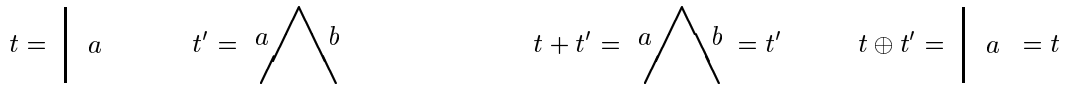

Figure 2: Examples of friendly normalization.

This is because $\leq_{\mathbf{fFAT}}$ refines the corresponding quotient ordering. Take for instance t and t' as described in Figure 2. So we have to explicitly define $\leq_{\mathbf{fFAT}}$ if we follow the quotient approach to define **fFAT**.

By comparing the definition of **fFAT** and the characterization of friendly testing, it is not difficult to prove the following

Theorem 4.3 (Full abstraction for fFAT.)
For any processes P and P', we have

$$P \sqsubseteq_{\mathrm{fr}} P' \qquad \text{iff} \qquad \mathbf{fFAT}[\![P]\!] \leq_{\mathbf{fFAT}} \mathbf{fFAT}[\![P']\!]$$

Moreover, since **fAT** is surjective, and **fFAT** can be constructed as a quotient algebra from it, we have that **fFAT** is also surjective.

5 Algebraic Characterization of fFAT

In this section we present an algebraic characterization of **fFAT**, or equivalently a sound and complete axiomatization of friendly testing. The set of inequations **FA2** is that corresponding to ordinary must testing, that is **A2** (see [Hen88, pg. 93]) extended with

$$(\mathbf{S}) \qquad P \oplus Q \leq P$$

to obtain **SA2** ([Hen88, pg. 105–106]), to which we add the axiom expressing that a deterministic external choice give us a (friendly) better process. This can be formalized as follows: $P \leq P + Q$ whenever $S(P) \cap S(Q) = \varnothing$. This condition can be captured in a pure syntactic way, taking instead

$$(\mathbf{F}) \qquad \sum_{a \in A} a\,;\, P_a \leq \sum_{a \in A'} a\,;\, P_a \qquad \text{whenever } A \subseteq A'.$$

Since $\sqsubseteq_{\mathrm{fr}}$ refines $\sqsubseteq$ we have that all the axioms in **SA2** are also correct with respect to $\sqsubseteq_{\mathrm{fr}}$. The new axiom is also correct, since the unique state of $\sum_{a \in A'} a\,;\, P_a$ is A', and we have $A \subseteq A'$ is a state of $\sum_{a \in A} a\,;\, P_a$, and all the continuations after the occurrences of the same action $a \in A$ are the same.

In order to prove completeness, we first define friendly normal forms. These are obtained from ordinary (may-must or must) normal forms, by including a condition of minimality for the states.

Definition 5.1 Friendly normal forms are those Hennessy's (see [Hen88, pg. 94]) normal forms

$$\bigoplus_{A \in \mathcal{A}} \sum_{a \in A} a\,;\, n(a)$$

such that for any $A \in \mathcal{A}$ and $a \in A$ there exists $A' \in \mathcal{A}$ which is minimal in $\mathcal{A}$ such that $a \in A'$. □

Next we prove that using **FA2** we can transform each syntactic process into friendly normal form. We already know that this is possible for plain normal forms. So we only need the following.

Theorem 5.2 For every normal form $n = \bigoplus_{A \in \mathcal{A}} \sum_{a \in A} a\,;\, n(a)$ there exists a friendly normal form $f(n)$ such that $n \leq_{\mathbf{FA2}} f(n)$ and $f(n) \leq_{\mathbf{FA2}} n$.
Proof: Let $F(\mathcal{A}) = \{a \mid \exists\, A \in \mathcal{A} :\ a \in A \in \mathcal{A} \ \wedge\ \nexists\, A' \in \mathcal{A} :\ a \notin A' \subseteq A\}$. We take

$$f^1(n) = \bigoplus_{\substack{A \in \mathcal{A} \\ A \subseteq F(\mathcal{A})}} \sum_{a \in A} a\,;\, n(a)$$

Then by applying axiom (**S**) we have $f^1(n) \leq_{\mathbf{FA2}} n$. Moreover, for each $A \in \mathcal{A}$ such that $A \not\subseteq F(\mathcal{A})$ there exists some $A' \in \mathcal{A}$, $A' \subseteq F(\mathcal{A})$, $A' \subseteq A$. Therefore by applying axiom (**F**) we obtain $\sum_{a \in A'} a \,;\, n(a) \leq_{\mathbf{FA2}} \sum_{a \in A} a \,;\, n(a)$ and then we conclude $n \leq_{\mathbf{FA2}} f^1(n)$. Now, by iterating the construction we can friendly normalize all the terms $n(a)$, and thus transform $f^1(n)$ into $f(n)$ in such a way that we have $n \approx_{\mathbf{FA2}} f(n)$. □

Finally we have to prove that friendly normal forms are related by $\leq_{\mathbf{FA2}}$ exactly as by $\sqsubseteq_{\mathrm{fr}}$.

Theorem 5.3 If f_1, f_2 are friendly normal forms we have $f_1 \leq_{\mathbf{FA2}} f_2 \iff f_1 \sqsubseteq_{\mathrm{fr}} f_2$

Proof: The left to right implication is an immediate consequence of the correctness of **FA2**. For the right to left implication, if $f_1 = \mathtt{STOP}$ we have just to apply (**F**) taking $\varnothing \subseteq A$. Otherwise we have

$$f_1 = \bigoplus_{A \in \mathcal{A}} \sum_{a \in A} a \,;\, f_1(a) \qquad \text{and} \qquad f_2 = \bigoplus_{A' \in \mathcal{A}'} \sum_{a \in A'} a \,;\, f_2(a)$$

with $\varnothing \notin \mathcal{A}$, and so, by applying the characterization of $\sqsubseteq_{\mathrm{fr}}$, $\varnothing \notin \mathcal{A}'$. Then, by applying again this characterization, we have

$$\forall\, A' \in \mathcal{A}'\ \exists A \in \mathcal{A} :\ \Big(A \subseteq A' \ \wedge\ \forall\, a \in A\ f_1(a) \sqsubseteq_{\mathrm{fr}} f_2(a)\Big)$$

Now, by induction hypothesis, we can assume that $f_1(a) \leq_{\mathbf{FA2}} f_2(a)$. Then, by applying (**F**), for each $A' \in \mathcal{A}'$ we obtain

$$\sum_{a \in \mathcal{A}(A')} a \,;\, f_1(a) \leq_{\mathbf{FA2}} \sum_{a \in A'} a \,;\, f_2(a)$$

where $\mathcal{A}(A') \in \mathcal{A}$ is the corresponding state of f_1 with $\mathcal{A}(A') \subseteq A'$. So we obtain

$$\bigoplus_{A' \in \mathcal{A}'} \sum_{a \in \mathcal{A}(A')} a \,;\, f_1(a) \leq_{\mathbf{FA2}} \bigoplus_{A' \in \mathcal{A}'} \sum_{a \in \mathcal{A}} a \,;\, f_2(a) = f_2$$

and by applying (**S**) we finally obtain

$$f_1 = \bigoplus_{A \in \mathcal{A}} \sum_{a \in A'} a \,;\, f_1(a) \leq_{\mathbf{FA2}} f_2$$

□

Thus we obtain the desired initiality and completeness theorem

Theorem 5.4

1. **fFAT** is initial in the class of algebras that satisfy **FA2**.
2. **FA2** is sound and complete for friendly testing.

6 Conclusions and Related Work

In this paper we have introduced a new kind of testing whose main property is to be consistent with the idea of design by refinement and conformance. Friendly testing results to be a smooth version of must testing. More exactly the induced testing preorder is weaker than the ordinary must preorder, and this is how the desired behavior covering the conformance relation is obtained. The conformance relation was introduced in [BSS86], where is defined by:

$$P_1 \ \mathbf{conf}\ P_2 \qquad \Longleftrightarrow \qquad \mathrm{Failures}(P_1) \cap \Big(\mathrm{Traces}(P_2) \times \mathcal{P}(\mathbf{Act})\Big) \subseteq \mathrm{Failures}(P_2)$$

Using the usual ordering notation we would write P_2 `conf` P_1. Unfortunately, it is the case that this relation is not an ordering, since for instance we have

$$a \oplus (b \,;\, c)\ \mathtt{conf}\ a\ \mathtt{conf}\ a + b, \qquad \text{but} \qquad a \oplus (b \,;\, c)\ \not\mathbf{conf}\ a + b$$

This is because under the definition of the conformance relation all the traces of P_2 have to be independently explored. Note that this is exactly the first example that we presented at the end of Section 3 when comparing the must and the friendly preorders.

G. Leduc [Led91, Led94] and A. Valmari & M. Tienari [VT91, VT95] have presented and studied a collection of semantic models based on the failures model. They are also interested on weakening the failures ordering, but they mainly concentrate on divergences, trying to avoid the catastrophic treatment of them in the classic model. More recently, E. Brinksma et. al [BRV95, BRV96] and V. Natarajan & R. Cleaveland [NC95] have studied the same subject, but directly in the testing framework.

Finally we will remark the relation between our friendly testing and bisimulation. This is suggested by the recursive definition of the friendly testing preorder. In fact, R. Cleaveland & M. Hennessy have proved [CH93] that the classical notion of testing equivalence can be seen as a bisimulation equivalence, once the classical notion of bisimulation is generalized in the adequate way. As a matter of fact, this generalization is an instance of a generalization framework that they present in the paper, and the authors claim that many others natural semantics equivalences could be obtained as instances of this generalization framework, and we are indeed convinced that this will be the case for our friendly testing semantics.

References

[BRV95] E. Brinksma, A. Rensink, and W. Vogler. Fair testing. In *CONCUR'95, LNCS 962*, pages 313–327, 1995.

[BRV96] E. Brinksma, A. Rensink, and W. Vogler. Applications of fair testing. In *Formal Description Techniques IX*, pages 145–160, 1996.

[BSS86] E. Brinksma, G. Scollo, and C. Steenbergen. LOTOS specifications, their implementations and their tests. In *Protocol Specification, Testing and Verification VI*, pages 349–360, 1986.

[CH93] R. Cleaveland and M. Hennessy. Testing equivalence as a bisimulation equivalence. *Formal Aspects of Computing*, 5:1–20, 1993.

[dNH84] R. de Nicola and M.C.B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[Hen85] M. Hennessy. Acceptance trees. *Journal of the ACM*, 32(4):896–928, 1985.

[Hen88] M. Hennessy. *Algebraic Theory of Processes.* MIT Press, 1988.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

[Led91] G. Leduc. Conformance relation, associated equivalence, and minimum canonical tester in LOTOS. In *Protocol Specification, Testing and Verification XI*, pages 249–264, 1991.

[Led94] G. Leduc. Failure-based congruences, unfair divergences and a new testing theory. In *Protocol Specification, Testing and Verification XIV*, pages 252–267, 1994.

[NC95] V. Natarajan and R. Cleaveland. Divergence and fair testing. In *22nd ICALP, LNCS 944*, pages 648–659, 1995.

[VT91] A. Valmari and M. Tienari. An improved failures equivalence for finite-state systems with a reduction algorithm. In *Protocol Specification, Testing and Verification XI*, pages 3–18, 1991.

[VT95] A. Valmari and M. Tienari. Compositional failure-based semantics models for basic LOTOS. *Formal Aspects of Computing*, 7:440–468, 1995.