18
Friendly Testing as a Conformance Relation

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

Abstract
In this paper we present a new kind of testing, namely *friendly testing*, which has been developed to obtain a satisfactory conformance relation sharing the good properties of the more popular conformance relations, that is *must-testing* and **conf** , while avoiding their respective problems. In particular, our friendly tests cannot punish a process when it is able to execute some action, while classical testing did it. This was a clear drawback of must testing when considered as a conformance relation. Finally, We prove that the preorder induced by friendly testing is just the transitive closure of **conf** . As a consequence we obtain an interesting characterization of this closure, from which we derive several its properties.

Keywords
Semantical foundations, Conformance testing, Formal methods.

1 INTRODUCTION AND RELATED WORK

Conformance is the term used by system analyzers to describe the situation in which an implementation is adequate with respect to a given specification. In order to properly define this notion, and thus to have the formal basis for the process of testing, there has been a considerable effort, that in particular has been the seed for the joint ISO/ITU-T working group on "Formal Methods in Conformance Testing". In a recent paper [Cavalli, Favreau & Phalippou 1996], some members of this group have presented a short, but nice, summary of the work carried out by the group, that was included in their working

Research supported in part by the CICYT project TIC 97-0669-C03-01.

documents [JTC1/SC21/WG1/54.1 1995*b*, JTC1/SC21/WG1/54.1 1995*a*]. In the same special issue of "Computer Networks and ISDN Systems" devoted to testing where [Cavalli et al. 1996] appeared, there is a longer paper on conformance testing which will be our main reference for definitions and main results [Tretmans 1996]. There, the reader is pointed out for a clear and rather intuitive presentation of the subject. In addition, a complete list of references is provided and therefore we will omit in this paper most of them.

In order to formalize the notion of conformance, two are the most extended methods: by means of an *implementation relation* or by *requirements*. We will concentrate ourselves on the first approach, that is the one in which more work has been developed. An implementation relation relates implementations from a given set $\texttt{Imp}$ with specifications from another set $\texttt{Spec}$. We are interested in the case in which both sets are somehow formalized, and more specifically, in the case in which both sets are the same. Thus, we will explore relations $\texttt{imp} \subseteq \texttt{Proc} \times \texttt{Proc}$, for some classes of processes $\texttt{Proc}$. The most known implementation relation is $\texttt{conf}$ [Brinksma, Scollo & Steenbergen 1986, Brinksma 1988]. This relation is defined from traces and refusals of processes in the following way:

$$i \; \texttt{conf} \; s \text{ iff } \forall\, t \in \texttt{Tr}(s) : \texttt{Ref}(i,t) \subseteq \texttt{Ref}(s,t)$$

This relation is derived from the plain refusal ordering [Hoare 1985], which is obtained by removing the constraint $\texttt{Tr}(s)$ in the universal quantification above. As it is well known, when restricted to non-divergent processes, the refusal ordering is an alternative characterization of the must testing preorder [Hennessy 1988]. Once the (must) passing of tests is defined (a detailed definition can be found at the beginning of Section 2), we can define the preorder $\sqsubseteq_{\mathrm{must}}$ as follows: *

$$s \sqsubseteq_{\mathrm{must}} i \;\text{ iff }\; \forall\, T \; (s \; must \; T \Rightarrow i \; must \; T)$$

Since conformance relations are defined to establish the framework in which to formally define testing, it seems that to define one of them by means of passing of tests is a very natural choice. Unfortunately, and even if the (must) testing relation has many pleasant properties, it proves to be too strong to adequately formalize the implementation process. For instance, if we use the notation in [Hennessy 1988] (where there are two choice operators: external, denoted by $+$, and internal, denoted by $\oplus$.), we have $a \not\sqsubseteq_{\mathrm{must}} a + b$. This does not seem very reasonable, because if we are restricted to execute only the

*There is some notation disagreement between the testing [Hennessy 1988] and the conformance communities. We have adopted the conventions of the first one to define $\sqsubseteq_{\mathrm{must}}$, since in a testing scenario it seems natural to consider that a process is *better* than another one when it passes more tests, and it is usual to read greater than relations as *better than*. On the contrary, in [Tretmans 1996] testing ordering $\leq_{\mathbf{te}}$ is represented by $i \leq_{\mathbf{te}} s$, probably to maintain the left to right convention between the implementation and the specification in the conformance relation. Finally, it is easy to check that the reduction relation $\texttt{red}$ [Brinksma et al. 1986, Leduc 1992], which can be defined by $i \; \texttt{red} \; s$ iff $i \; \texttt{conf} \; s$ and $\texttt{Tr}(i) \subseteq \texttt{Tr}(s)$, is equal to $\leq_{\mathbf{te}}$ above. In fact, this could be seen as a more convincing justification of the use of the left to right notation for the ordering, since in this alternative definition of the relation there is no reference to tests, and instead the stress is put on the conformance relation $\texttt{conf}$.

action a, it should not matter if we are also able to execute the action b. It is the case that the relation `conf` solves this problem; actually, $a + b$ `conf` a. But this relation does not possess good formal properties. Probably, its most important weakness is that `conf` is not transitive, and thus neither an order relation. For example, it is easy to check that we also have a `conf` $a \oplus (b\,;c)$ but not $a + b$ `conf` $a \oplus (b\,;c)$, since after the possible execution of b by the specification, the implementation cannot execute the expected c, while this b plays no role when comparing a and $a \oplus (b\,;c)$.

G. Leduc has thoroughly worked on the theoretical study of conformance relations [Leduc 1991, Leduc 1992]. He has studied the equivalence induced by an implementation relation, which is defined by:

$$s_1 \ \mathbf{imp\text{–}eq} \ s_2 \ \text{ iff } \ \forall\, i : \ (i \ \mathbf{imp} \ s_1 \Longleftrightarrow i \ \mathbf{imp} \ s_2)$$

Whenever `imp` is an order relation, it is immediate to prove that `imp–eq` is the usual equivalence relation induced by it, thus we have $\mathbf{imp\text{–}eq} = \mathbf{imp} \cap \mathbf{imp}^{-1}$. But if `imp` is not an order relation, we only have $\mathbf{imp\text{–}eq} \subseteq \mathbf{imp} \cap \mathbf{imp}^{-1}$. This is the case for `conf`, for which we have

$$\begin{aligned} s_1 \ \mathbf{conf\text{–}eq}\, s_2 \ \text{ iff } \ s_1 \ \mathbf{conf} \ s_2 \wedge s_2 \ \mathbf{conf} \ s_1 \ & \wedge \forall\, t \in \mathtt{Tr}(s_1) - \mathtt{Tr}(s_2) : L \in \mathtt{Ref}(s_1, t) \\ & \wedge \forall\, t \in \mathtt{Tr}(s_2) - \mathtt{Tr}(s_1) : L \in \mathtt{Ref}(s_2, t) \end{aligned}$$

where L denotes the full alphabet of observable actions. The last two conditions above are necessary indeed, as the following example shows: Let $s_1 = a$ and $s_2 = a\,; (\mathtt{STOP} \oplus (b\,; c))$; we have s_1 `conf` s_2 and s_2 `conf` s_1, but not s_1 `conf–eq` s_2. As a matter of fact, $\mathbf{conf} \cap \mathbf{conf}^{-1}$ is not an equivalence relation, as the following example shows: Let $s_3 = a\,; (\mathtt{STOP} \oplus (b\,; d))$; we have s_1 `conf` s_3 and s_3 `conf` s_1, but neither s_2 `conf` s_3 nor s_3 `conf` s_2. Finally, `conf–eq` is weaker than must–equivalence, since for $s_4 = a\,; (\mathtt{STOP} \oplus b)$ we have s_1 `conf–eq` s_4, but $s_1 \not\sqsubseteq_{\mathrm{must}} s_4$.

Since `conf` is not an order relation, we need a stronger relation if we want to follow a refinement process to obtain implementations from specifications. Thus, `confrestr` is introduced, which is the strongest order relation weaker than `conf` which preserves that relation, that is, $\mathbf{conf} \circ \mathbf{confrestr} = \mathbf{conf}$. The relation `confrestr` can be defined in any of the following alternative ways:

- s_1 `confrestr` s_2 iff $\forall\, i : \ (i \ \mathbf{conf} \ s_1 \Rightarrow i \ \mathbf{conf} \ s_2)$.
- s_1 `confrestr` s_2 iff s_1 `conf` s_2 $\wedge \ \forall\, t \in \mathtt{Tr}(s_2) - \mathtt{Tr}(s_1) : L \in \mathtt{Ref}(s_2, t)$.

It is easy to check that $\mathbf{conf\text{–}eq} = \mathbf{confrestr} \cap \mathbf{confrestr}^{-1}$.

Based on these somehow negative facts about the two most extended relations, that is $\sqsubseteq_{\mathrm{must}}$ and `conf`, we have looked for a compromise between them which could inherit the good properties of both, while avoiding their problems. As we already said, we think that to maintain a testing interpretation for a relation that will be the basis for the testing framework seems to be very desirable. So, we have tried to find a new notion of test, and of the passing of tests mechanism, by means of which the desired ordering could be defined following the usual testing way: an implementation is better than (or adequate with respect to) a specification, relatively to our desired conformance relation, if it passes more tests than this last one.

It is clear that any relation defined in this way is an ordering. As an immediate consequence we have that the conformance relation cannot be characterized in this way. Thus we concluded that the adequate starting point for our new notion of testing was not the conformance relation **conf**, but the classical testing scenario defining $\sqsubseteq_{\text{must}}$. So we concentrated ourselves on how tests, and the passing of tests, are defined.

As we will define in detail in the following section, tests are just processes over the alphabet of actions **Act** extended with a special action ω to express successful passing of tests. We apply a test T to a process P by considering the system $P \,||\, T$; then, a computation succeeds whenever it reaches a point where the action ω can be executed. If we consider must passing of tests as defined in [de Nicola & Hennessy 1984, Hennessy 1988], we have found that tests have the power to ***punish*** processes being able to execute actions. So, $a \not\sqsubseteq a + b$, since the former process passes the test $(1 \,;\, \omega) + (b \,;\, \mathtt{STOP})$, while the latter does not.* We are interested on a testing framework in which tests cannot punish processes when they are able to execute some action. This is why we call ***friendly testing*** to our new testing scenario, and we denote by $\sqsubseteq_{\text{fr}}$ the induced preorder. 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 tested process. 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.

We could think that if we are working with a language including an internal choice operator, as it is $\oplus$ in [Hennessy 1988], then internal actions are not needed 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. Then it could be thought that all our problems would be solved just by 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, $a \,;\, \mathtt{STOP} \sqsubseteq_{\text{fr}} a \,;\, P$ and so on; but unfortunately when there are choices among several observable actions, the problem still remains. For instance, $a + b$ would not be ***better*** than a because the test $(a \,;\, \omega) + (b \,;\, \textit{reject} \,;\, \omega)$ is passed by the latter process but not by the former. This means that we cannot just restrict the family of tests to reach our goal,

*In [Hennessy 1988] the symbol 1 is used to denote internal actions. Other alternative notations are τ and i.

but also the definition of test passing must change, if we desire to obtain $a \sqsubseteq_{\text{fr}} a + b$.

In the following section we will show the adequate changes leading to our new notion of testing. Moreover, we will show that $\sqsubseteq_{\text{fr}}$ is indeed related with $\texttt{conf}$, as it was our intention. Actually, we have proved that $\sqsubseteq_{\text{fr}}$ is just the transitive closure of $\texttt{conf}$, namely $\texttt{conf}^*$, which is the strongest order relation weaker than $\texttt{conf}$ where, as usual, we say that $\sqsubseteq_1$ is stronger that $\sqsubseteq_2$ iff $S_1 \sqsubseteq_1 S_2$ implies $S_1 \sqsubseteq_2 S_2$. This is, in our opinion, a nice alternative to the original conformance relation $\texttt{conf}$, which solves most of its problems, and thus represents the searched compromise between $\texttt{conf}$ and $\sqsubseteq_{\text{must}}$, even if it is not somewhere between them, but instead it is weaker than both. Thus, we have followed the opposite direction that led to $\texttt{confrestr}$. The reason is very simple: when we studied the examples showing that $\texttt{conf}$ is not an order relation we found no problem on taking $\texttt{conf}^*$ instead of $\texttt{conf}$, thus having, for instance, $a + b \ \texttt{conf}^* \ a \oplus (b;c)$. We think that the only reason because this is not allowed under $\texttt{conf}$ is that $\texttt{conf}$ was defined looking for an elegant solution in terms of traces and refusals, even if the obtained relation did not posses some good properties (for instance, being an order relation).

The relation $\texttt{conf}^*$ was introduced in [Leduc 1992] where also some of its properties were studied. In particular, two interesting results are: $\texttt{conf}^* = \texttt{conf} \circ \texttt{conf}$ and $\texttt{conf}^* = \texttt{ext} \circ \texttt{red}$. Both results will be somehow used in our proof of the fact that $\sqsubseteq_{\text{fr}}$ is equal to $\texttt{conf}^*$. From this characterization one can find many interesting properties of this relation, which would be more difficult to obtain directly using its definition. For example, by means of this characterization we have defined a complete axiomatization which is obtained by adding a single axiom to that for must testing. Moreover, we also have obtained an explicit characterization based on acceptance sets (or equivalently on refusals). All these results contribute to get a justification and support of friendly testing (equivalently $\texttt{conf}^*$) as a satisfactory conformance relation.

The rest of the paper is structured as follows. In Section 2 we present our new notion of testing. We first introduce *friendly testing* for a particular class of processes that we call *normal forms*. Next, we define friendly testing for arbitrary processes in our language. In Section 3, an alternative characterization of the friendly testing relation, based on a modification of acceptance sets, is defined. Section 4 is devoted to prove that $\sqsubseteq_{\text{fr}}$ is equal to $\texttt{conf}^*$. Finally, in Section 5 we present our conclusions and sketch some of the results on friendly testing that we have obtained, including the complete axiomatization announced above.

2 FRIENDLY TESTING: BASIC DEFINITIONS

Since its introduction in [de Nicola & Hennessy 1984, Hennessy 1988] *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 experimental 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 whose computations are all unsuccessful, those for which some computations are successful, and those whose computations are all 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.

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.

In order to present our proposal, we will concentrate ourselves on a syntactic definition of processes, considering a process algebra, instead of using arbitrary transition systems.

Besides we will follow a step by step approach, considering incrementally more general languages, because we think that this process contributes to a better understanding of the definition itself, and also of its properties. To make the comparison with classical testing semantics easier, we will work with the same signature considered in [Hennessy 1988], which is defined by:

Definition 2.1 The set of *finite processes*, denoted by *Proc*, is defined as the set of expressions given by the following BNF-expression:

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

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

Operator + corresponds with the external choice operator □ in CSP, and also with the same operator + in CCS when internal actions are not involved in the choice. Besides, $\oplus$ corresponds with the internal choice operator $\sqcap$ in CSP. All the actions in **Act** are assumed to be visible.

In order to introduce friendly testing, we will first 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 $\bigoplus_{A \in \{\emptyset\}}$ represents the process STOP.

As usually, tests will be just processes over the alphabet $\mathbf{Act} \cup \{\omega\}$. For the same reasons that for processes, we will consider a restricted version of tests: those finite deterministic tests with acceptance actions at the end of each trace. We will show that this family of deterministic tests is a set of *essential* tests, in the sense that whenever two processes are not friendly equivalent then there exists a deterministic test distinguishing them. *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 2.2 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 friendly passes 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 this definition, although recursive, 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 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, 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. Next, we compare our definition with the plain must testing by means of an illustrative example.

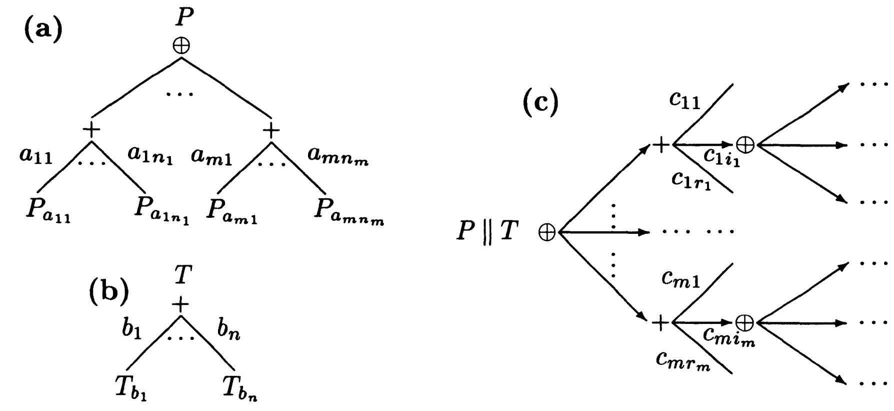

Figure 1 Normal Forms, Deterministic Tests, and P *friendly passes* T.

Example 2.3 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 not P_2 *must* T. The reason for this is that in order to get P_2 *must* T all the computations of $P_2 \| T$ must be successful. In particular, this must be true for the computations of $P' \| T$. 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 or external. So, for this kind of tests we have P' *must* T iff P'' *must* T.* Such a behavior could be justified by the assumption of testing being the only way to observe the behavior of the tested process. As a matter of fact, and even if that would have no effect in its definition of passing tests, [Hennessy 1988] does not label the transitions of experimental systems of the form $P \| T$. As a consequence, the computations tree corresponding to both $P' \| T$ and $P'' \| T$ 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 $P' \| T$ and $P'' \| T$. This is indeed the case, because if we apply the classic (expansion) axioms for the parallel operator we obtain on the one hand $P' \|_{\mathbf{Act}} T \approx (a\,;(P_a \|_{\mathbf{Act}} \omega)) + (b\,;(P_b \|_{\mathbf{Act}} \mathtt{STOP}))$, while on the other hand $P'' \|_{\mathbf{Act}} T \approx (a\,;(P_a \|_{\mathbf{Act}} \omega)) \oplus (b\,;(P_b \|_{\mathbf{Act}} \mathtt{STOP}))$. So, under our notion of friendly testing we have that P' and P'' can be distinguished by the

*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.

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 friendly passed.

The reader could think this new notion of passing tests is much more involved than the classic one, but we advocate that this is not the case. Actually, if we consider a recursive definition of the classical notion of must test passing for normal form processes, we see that it can be obtained from our definition of friendly test passing just by changing the existential quantification in the third condition of Definition 2.2 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, but 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 2.4

1. $P \oplus Q \sqsubseteq_{\text{fr}} P$. This is because we already had $P \oplus Q \sqsubseteq_{\text{must}} P$, and in general we have $P \sqsubseteq_{\text{must}} Q \implies P \sqsubseteq_{\text{fr}} Q$. As a particular case we have $a \oplus (a+b) \sqsubseteq_{\text{fr}} a+b$. On the contrary, we have $(a;c) \oplus (b;c) \not\sqsubseteq_{\text{fr}} a+b$, since the test $(a;c;\omega)+(b;c;\omega)$ is friendly passed by the former process but not by the latter.
2. $a \sqsubseteq_{\text{fr}} a + b$. 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 $P \sqsubseteq_{\text{fr}} P + Q$ whenever the sets of actions that can be executed by P and Q in their first steps are disjoint.
3. $a \oplus (a + b) \approx_{\text{fr}} a$, because on the one hand we have $a \oplus (a + b) \sqsubseteq_{\text{fr}} a$, again as a particular case of the property asserted in 1. On the other hand, note that $a \oplus a \approx_{\text{fr}} a$ and then we can apply the fact that all the operators of the language are monotonic with respect to the friendly testing relation.

□

2.1 Friendly Testing for arbitrary finite processes and tests

In this section we will consider arbitrary finite processes and tests generated by the syntax given in Definition 2.1. The operational semantics of the language is defined as in [Hennessy 1988]:

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

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

The following conventions will be used:

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

Moreover, for $s = a_1, \ldots, a_n$ we write $P \overset{s}{\Longrightarrow} P'$ 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 **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. 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* if $P \not\rightarrowtail$. Moreover, given a test T we say that a configuration $P \parallel T$ is *stable* if $P \parallel T \not\rightarrowtail$.

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

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

- If $P \parallel T$ is stable, then either $T \xrightarrow{\omega}$, or there exists some $a \in$ **Act** such that $P \parallel T \xrightarrow{a}$ and (P/a) *friendly passes* (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' *friendly passes* 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 $P', T', a \in$* **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'' friendly passes T''.* Thus it is easy to check that the definition above is an extension of the one for normal forms.

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

Proposition 2.7 Let P, P_1, P_2 be processes, and T, T_1, T_2 tests. We have

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

Definition 2.8 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 state a result showing that deterministic tests constitute indeed a set of *essential* tests.

Proposition 2.9 Let P, Q be processes. Then we have $P \sqsubseteq_{\text{fr}} Q$ iff for any deterministic test T whenever P *friendly passes T* we also have that Q *friendly passes T.*

3 ALTERNATIVE CHARACTERIZATION OF $\sqsubseteq_{\text{fr}}$

In this section we provide an alternative characterization of the friendly testing preorder given in Definition 2.8. This characterization is based on a modification of acceptance sets [Hennessy 1988]. 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. We define the following concepts:

- *Initial actions* of P: $S(P) = \{a \mid P \overset{a}{\Longrightarrow}\}$.
- *Acceptance sets* of P *after* s: $\mathcal{A}(P, s) = \{S(P') \mid P \overset{s}{\Longrightarrow} P'\}$.

- *friendly acceptance sets* of P: $\mathcal{F}(P) = \{A \in \mathcal{A}(P,\epsilon) \mid \nexists A' \in \mathcal{A}(P,\epsilon) : A' \subsetneq A\}$ □

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 $((P/a_1)\cdots)/a_n$. 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 $P \ll_{\mathrm{fr}} P'$ if for all $A' \in \mathcal{F}(P')$ there exists $A \in \mathcal{F}(P)$ such that $A \subseteq A'$, and for all $a \in A$, $P/a \ll_{\mathrm{fr}} P'/a$. □

Now we will prove that the preorders $\sqsubseteq_{\mathrm{fr}}$ and $\ll_{\mathrm{fr}}$ coincide. We split the proof in two parts.

Theorem 3.3 Given P and P' be processes, we have $P \sqsubseteq_{\mathrm{fr}} P'$ implies $P \ll_{\mathrm{fr}} P'$.
Proof: The proof will be done by the contrapositive, and structural induction. 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:

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

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

In the first case we construct a set S including for each $A \in \mathcal{F}(P)$ one action in $A - A'$. Then, if we consider the deterministic test $T = \sum_{a \in S} a\,;\omega$, we get P *friendly passes* 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 *friendly passes* 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 *friendly passes* T but P' does not, since each P/a_A does not *friendly pass* the test T_{a_A}. □

Theorem 3.4 Given P and P' processes, we have $P \ll_{\mathrm{fr}} P'$ implies $P \sqsubseteq_{\mathrm{fr}} P'$.
Proof: Let T be a deterministic test such that P *friendly passes* T. We will prove, by induction on the depth of T, that P' also *friendly passes* T.

If depth$(T) = 1$ then $T = \omega$ and the result is trivial. Otherwise we have $T = \sum_{i \in I} a_i \, ; T_i$. Then, in order to check that P' *friendly passes* 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' *friendly passes* T_i. Since 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 $P \ll_{\text{fr}} P'$, 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 $a \in A$: $P/a \ll_{\text{fr}} P'/a$. By hypothesis P *friendly passes* T, and thus there exists $a \in A$, with $a = a_i$ for some i, such that P/a *friendly passes* T_i. Therefore we can take $a' = a = a_i$, and by applying the induction hypothesis we obtain P'/a' *friendly passes* T_i, and thus we conclude P' *friendly passes* T. □

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

4 RELATION BETWEEN conf* AND $\sqsubseteq_{\text{fr}}$

In this section we will prove that the relations **conf*** and $\sqsubseteq_{\text{fr}}$ are the same.

First, to make easier the comparison with the conformance relation **conf**, we give a characterization of $\sqsubseteq_{\text{fr}}$ in terms of refusals. We have obtained an explicit non–recursive characterization by introducing the notion of *friendly admissible sets of traces* which gathers the information about the traces that must be taken into account to friendly compare two given processes.

Definition 4.1 Given two processes P, P' we define the family of *friendly admissible sets of traces* for them, denoted by $\mathcal{F}at(P, P')$, as the class of sets $\mathcal{S}$ verifying the following conditions:

- $\epsilon \in \mathcal{S}$
- $t \in \mathcal{S} \Longrightarrow \forall R' \in \mathtt{Ref}(P', t) \; \exists R \in \mathtt{Ref}(P, t) : \; (R' \subseteq R \; \wedge \; \forall a \notin R : \; ta \in \mathcal{S})$

□

Theorem 4.2 Given two processes P, P' we have:

$$P \sqsubseteq_{\text{fr}} P' \text{ iff } \exists \, \mathcal{S} \in \mathcal{F}at(P, P') \; \forall t \in \mathcal{S} : \; \mathtt{Ref}(P', t) \subseteq \mathtt{Ref}(P, t)$$

Let us remark that the condition on the traces of $\mathcal{S}$ in the formula above is already coded in the definition of friendly admissible sets of traces and thus could be removed here, but we include it in order to make easier the comparison with **conf**.

Corollary 4.3 P' **conf** $P \Rightarrow P \sqsubseteq_{\text{fr}} P'$.
Proof: We only have to notice that for any $\mathcal{S} \in \mathcal{F}at(P, P')$ whenever we have $t \in \mathcal{S}$ we also have $t \in \mathtt{Tr}(P')$. □

Let us note that $t \in \mathtt{Tr}(P)$, too. This means that only common traces have to be explored. This makes possible P' being friendly better than P when the former has either more or less traces than the latter.

The following two theorems prove the desired equivalence between $\mathtt{conf}^*$ and $\sqsubseteq_{\mathrm{fr}}$.

Theorem 4.4 Let P, P' be processes. We have $P\ \mathtt{conf}^*\ P' \Longrightarrow P' \sqsubseteq_{\mathrm{fr}} P$.
Proof: Trivial, just noticing that $\mathtt{conf}^*$ is the transitive closure of the relation $\mathtt{conf}$, that $P\ \mathtt{conf}\ P' \Longrightarrow P' \sqsubseteq_{\mathrm{fr}} P$ (Corollary 4.3), and that $\sqsubseteq_{\mathrm{fr}}$ is an order relation. □

Theorem 4.5 Let P, P' be processes. We have $P \sqsubseteq_{\mathrm{fr}} P' \Longrightarrow P'\ \mathtt{conf}^*\ P$.
Proof: We will present the proof for normal form processes. In order to extend it to arbitrary processes, we would use the result in [Hennessy 1988] saying that any finite process can be transformed into normal form up to must–testing equivalence, and the fact that $\sqsubseteq_{\mathrm{must}}$ is stronger than $\sqsubseteq_{\mathrm{fr}}$.

Let P_1, P_2 be normal form processes such that $P_1 \sqsubseteq_{\mathrm{fr}} P_2$. We will prove by induction on the depth of P_1 that we also have $P_2\ \mathtt{conf}^*\ P_1$.

If $\mathrm{depth}(P_1) = 0$ we have $P_1 = \mathtt{STOP}$, and so we trivially get $P_2\ \mathtt{conf}^*\ P_1$.

Let $\mathrm{depth}(P_1) = n + 1$ with $P_1 = \bigoplus_{A\in\mathcal{A}} P^1_A$ and $P_2 = \bigoplus_{B\in\mathcal{B}} P^2_B$ such that $P_1 \sqsubseteq_{\mathrm{fr}} P_2$. Then we have that for any $B \in \mathcal{B}$ there exists some $A_B \in \mathcal{A}$ such that $A_B \subseteq B$ and for all $a \in A_B$ we have $P_1/a \sqsubseteq_{\mathrm{fr}} P_2/a$. Then, if we take $\mathcal{A}' = \{A_B \mid B \in \mathcal{B}\}$ we have that for any $a \in A'$ with $A' \in \mathcal{A}'$, we also have $P_1/a \sqsubseteq_{\mathrm{fr}} P_2/a$. This means that if we consider $P_1' = \bigoplus_{A'\in\mathcal{A}'} P^1_{A'}$, and we define $P_2' = \bigoplus_{B\in\mathcal{B}} P'_B$, where $P'_B = \sum_{b\in B} P'^2_b$ and

$$P'^2_b = \begin{cases} P^1_b & \text{if } \exists\, A' \in \mathcal{A}' : \ b \in A' \\ P^2_b & \text{otherwise} \end{cases}$$

we have $P_2'\ \mathtt{conf}\ P_1'$. Besides, by applying induction hypothesis, we have that $\forall A' \in \mathcal{A}', a \in A'$: $P_2/a\ \mathtt{conf}^*\ P_1/a$. Given that $\mathtt{conf}$ is substitutive in the context of the arguments of normal forms, if we recover the original continuations of P_2, by substituting those from P_1 in P_2' by those from P_2, we conclude $P_2\ \mathtt{conf}^*\ P_1'$, and since obviously we have $P_1'\ \mathtt{conf}\ P_1$, we finally obtain $P_2\ \mathtt{conf}^*\ P_1$. □

It is interesting to observe that it is just this final step of the proof which makes (in general) not possible to conclude $P_2\ \mathtt{conf}\ P_1$, since when relating P_2 and P_1 by using an intermediate process P_1', we have that P_1' is a restriction of P_1 (i.e. $P_1'\ \mathtt{red}\ P_1$) while P_2 could extend P_1' (i.e. $P_2\ \mathtt{ext}\ P_1'$), and if we eliminate this intermediate process we could obtain some common traces that $\mathtt{conf}$ must explore, what $\sqsubseteq_{\mathrm{fr}}$ only partially does. Let us note that we could make a more detailed proof to directly conclude $\sqsubseteq_{\mathrm{fr}} \equiv \mathtt{ext} \circ \mathtt{red}$, but given that $\mathtt{conf}^* \equiv \mathtt{ext} \circ \mathtt{red}$ [Leduc 1992], it is enough to prove $\sqsubseteq_{\mathrm{fr}} \equiv \mathtt{conf}^*$ even if we were interested in the final characterization $\sqsubseteq_{\mathrm{fr}} \equiv \mathtt{ext} \circ \mathtt{red}$.

Corollary 4.6 Let P, P' be processes. We have $P \sqsubseteq_{\rm fr} P' \iff P'$ $\mathtt{conf}^*$ P.

5 CONCLUSIONS AND FURTHER WORK

We have presented a new kind of testing, ***friendly testing***, which proves to behave as a conformance relation better than the classical must testing does. This is because we reduce the power of tests in such a way that processes cannot be punished when they are able to execute more actions than others. More exactly, we have proved that the order relation induced by friendly testing is just the transitive closure of the conformance relation, **conf**. As a consequence we have obtained an interesting characterization of this relation, from which many properties of it can be derived.

In [Frutos-Escrig, Llana-Díaz & Núñez 1997] we have developed a full theory of friendly testing similar to that for classical testing [Hennessy 1988]. First, we have adapted the results in this paper to deal with general labeled transitions systems which in particular cover the case of recursive processes. Moreover, we have provided both a denotational model and a complete axiomatization. This axiomatization is obtained by adding to the set of axioms for must testing in [Hennessy 1988] the following one

$$\sum_{a\in A} a\,;P_a \leq_{\rm fr} \sum_{a\in A'} a\,;P_a \text{ whenever } A \subseteq A'$$

In order to obtain both, the denotational model and the complete axiomatization we have found an important technical problem: as it was the case for $\mathtt{conf}$, $\mathtt{conf}^*$ is not substitutive for arbitrary contexts. More exactly, we have that $\sqsubseteq_{\rm fr}$ is not a pre–congruence with respect to the external choice operator, as the following example shows:

$$\begin{array}{l} \mathtt{STOP} \oplus b\,;P \approx_{\rm fr} \mathtt{STOP} \\ (\mathtt{STOP} \oplus b\,;P) + b\,;Q \approx_{\rm fr} b\,;(P \oplus Q) \not\approx_{\rm fr} b\,;Q \approx_{\rm fr} \mathtt{STOP} + b\,;Q \end{array}$$

The problem disappears if there are no interferences between the offerings of the two involved processes.

In the axiomatization we only have to substitute the external choice substitutivity axiom for a more restrictive version covering the case where the involved processes do not offer any common action, to obtain a sound system for friendly testing which can be proved to be also complete by adequating the concept of normal form to the new framework, by means of the characterization by friendly acceptance sets.

Concerning the denotational semantics, it is obvious that we cannot obtain a fully abstract model, since the friendly testing equivalence is not substitutive. This leads to study the weaker pre–congruence $\sqsubseteq_{\rm frext}$ stronger than $\sqsubseteq_{\rm fr}$.

We have seen that the relation $\sqsubseteq_{\rm frext}$, which is the pre–congruence induced by $\sqsubseteq_{\rm fr}$, is somewhere between $\sqsubseteq_{\rm fr}$ and $\sqsubseteq_{\rm must}$, but closer to the first than to the last. In fact, we still have that $\sqsubseteq_{\rm frext}$ is not stronger than $\mathtt{conf}$ (it is not weaker either).Thus if we work under $\sqsubseteq_{\rm frext}$ we still have a rather satisfactory behavior as expected for a conformance relation. Besides $\sqsubseteq_{\rm fr}$ is *almost* a pre–congruence and so we can, in most of the contexts, substitute a process by another related by that relation, with the guarantee that the relation will be preserved.

REFERENCES

Brinksma, E. [1988], A theory for the derivation of tests, *in* 'Protocol Specification, Testing and Verification VIII', pp. 63–74.

Brinksma, E., Scollo, G. & Steenbergen, C. [1986], LOTOS specifications, their implementations and their tests, *in* 'Protocol Specification, Testing and Verification VI', pp. 349–360.

Cavalli, A., Favreau, J. & Phalippou, M. [1996], 'Standardization of formal methods in conformance testing of communication protocols', *Computer Networks and ISDN Systems* **29**, 3–14.

Frutos-Escrig, D., Llana-Díaz, L. & Núñez, M. [1997], Introducing friendly testing, Technical Report DIA 53/97, Dept. Informática y Automática. Universidad Complutense de Madrid.

de Nicola, R. & Hennessy, M. [1984], 'Testing equivalences for processes', *Theoretical Computer Science* **34**, 83–133.

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

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

JTC1/SC21/WG1/Project 54.1 [1995*a*], 'FMCT guidelines on Test Generation Methods from Formal Descriptions'.

JTC1/SC21/WG1/Project 54.1 [1995*b*], 'Working Draft on "Framework: Formal Methods in Conformance Testing"'.

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

Leduc, G. [1992], 'A framework based on implementation relations for implementing LOTOS specifications', *Computer Networks and ISDN Systems* **25**(1), 23–41.

Tretmans, J. [1996], 'Conformance testing with labelled transition systems: Implementation relations and test generation', *Computer Networks and ISDN Systems* **29**, 49–79.