Preserving Contexts for Soft Conformance Relation

David de Frutos Escrig[⋆] and Carlos Gregorio Rodríguez[⋆⋆]

Department of Sistemas Informáticos y Programación,
Universidad Complutense de Madrid
{defrutos, cgr}@sip.ucm.es

Abstract. This paper addresses the study of bisimulation based conformance relations in which input and output actions not presented in the specification are added to the implementation. A new definition, that we called *soft conformance*, is given. Then, we concentrate on the study of the conditions under which a context preserves the soft conformance relation of two agents. These conditions depend both on the specification and the implementation in the conformance relation and also on the context. Since the addition of extraneous actions to the implementation allows to define malicious contexts that would not preserve the conformance relation, such a characterisation of the family of contexts preserving each individual pair (implementation and specification) in the conformance relation is the best result that can be expected in this direction.

1 Introduction

Conformance relations have been introduced and studied since late eighties, providing a testing methodology for communicating systems. Conformance relations look for the adequate way to check when a concrete system should be considered a correct implementation of a given specification. The most popular conformance relations are based on traces and refusals [Hoa85], and probably that called `conf` [BSS86, Bri88] is the most widely spread and accepted.

First definitions on the subject were quite informal and tried to capture by means of some simple, but sometimes vague, conditions those *reasonable* requirements to get a correct implementation of a given specification. Fortunately, it was not too difficult to obtain formal definitions which captured the intuitive ideas supporting the original proposals, as the relation `conf` cited above.

The bad news were that although these formal definitions where rather simple and elegant they did not satisfy some also simple and clearly desirable properties, such as transitivity and substitutivity, and therefore they were far from being precongruencies.

In [Led91, Led92] an extensive and careful study of the subject can be found. There the relation `conf-eq` is introduced and proved to be the biggest equivalence relation contained in the nucleus $\texttt{conf} \cap \texttt{conf}^{-1}$ of the conformance relation, while $\texttt{conf}^* = \texttt{conf} \circ \texttt{conf}$ is proved to be its transitive closure.

⋆ Partially supported by the MCyT project TERMAS TIC2003-07848-C02-01, JCCM project PAC-03-001 and MRTN-CT-2003-505121/TAROT.

⋆⋆ Partially supported by the MCyT project TERMAS TIC2003-07848-C02-01.

Since traces and failures are strongly related with the semantic information given by testing formalisms [Hen88], several works have studied this relation. For instance, in [dFLN97] it was proved that `conf`* can be characterised by means of an special kind of testing mechanisms, the so called *friendly testing*, which is thoroughly studied in [dFLN98].

Together with the testing school, there are other approaches to define the equivalence between concurrent processes in process algebras. Equivalences based on *bisimulation* [Mil80, Mil89] are also widely used. It is well known that bisimulation equivalences are stronger than testing equivalences, but also much easier to decide, which seem to be two important reasons to prefer them to the others. Clearly, if it is possible to prove bisimilarity of two processes, then they would be also testing equivalent. But this strong power of bisimulation can also became a weakness, since there are not clear reasons to consider that two processes which are testing equivalent, but not bisimilar, should not be considered to be equivalent. Besides, weak bisimulation is not a congruence for languages such as CSP [Hoa85], where there exists an external choice operator (see [dFLN99]).

In [Ste94] a bisimulation based conformance, called *logical conformance*, is presented where classical bisimulation rules are relaxed and asymmetrical conditions related to the specification and the implementation are introduced. In [BS02] a new version of this conformance relation is given. In this relation it is allowed for the process describing the implementation system to execute both input and output new actions. Similar ideas have also been followed, in conformance relations defined by testing semantics, for instance [Bri88, dFLN97].

As it happens in the conformance relation based on testing semantics, the addition in [BS02] of new input and output actions to the implementation yields to a conformance relation that it is neither transitive nor preserved by most of the algebraic operators in CCS. To overcome these problems was the main goal of [BS02], and their authors concluded the paper by asserting that they have defined the *congruent weak conformance* induced by their *weak conformance* relation. Unfortunately, even if in that paper there are several interesting ideas, and some useful partial results, we have to present here some criticisms because there are several technical mistakes in that work, as we will show by means of some counterexamples later.

However, our main intended goal in this paper it is mainly to continue the research in conformance relations in which input and output actions not presented in the specification are added to the implementation, looking for the adequate way to get preservation results in order to make the conformance relation useful.

To be more concrete, what indeed is done in [BS02] is to find a collection of properties which have to be satisfied in order to preserve the presented conformance relation. Most of these conditions would restrict the containing context and not the relationship between the given implementation and the corresponding specification. Therefore, it is not possible to use those conditions to try to define a precongruence which would preserve the conformation relation.

Instead, what we propose is to characterise which are the contexts that would preserve each particular pair in the conformation relation. In fact, these contexts would be different for each pair in the relation, and therefore, out of some trivial cases, we cannot look for a family of contexts totally preserving the conformance relation. As a consequence, the weaker precongruence relation stronger than the conformance relation would be just the weak bisimulation equivalence, where we have no possibility to add any new action when implementing a given specification.

We address this goal in the next sections organised as follows: in Section 2 classic definition of agents and previously bisimulation based conformance relations are introduced, besides, definition of weak conformance [BS02] is discussed and some flaws of that relation are shown; in Section 3 we present our own definition of bisimulation like conformance relation, that we call *soft conformance*; Section 4 presents the results of the paper: we prove that the soft conformance relation can be preserved by contexts under some conditions related to a given pair of specification and implementation agents; finally, in Section 5 we present our conclusions.

2 Basic Definitions and Bisimulation Based Conformances

In this paper we will mainly use the operators from CCS [Mil80, Mil89], whose syntax and semantics we will briefly recall below.

We have a set of action names, called $\mathcal{A}$, from which we obtain the set of barred actions, $\overline{\mathcal{A}} = \{\overline{a} \mid a \in \mathcal{A}\}$. Following [BS02], we will assume that plain names represent input actions, while barred names would correspond to output actions. Finally, we have an internal action $\tau \notin \mathcal{A} \cup \overline{\mathcal{A}}$, and we define the alphabet $Act = \mathcal{A} \cup \overline{\mathcal{A}} \cup \{\tau\}$.

Definition 1 ([Mil89]). *Given a set of actions Act, as described above, the set of CCS agents is defined by the following BNF-expression:*

$$E ::= \mathbf{0} \;\mid\; \alpha.E \;\mid\; E+E \;\mid\; E|E \;\mid\; E[f] \;\mid\; E\backslash L$$

where $\alpha \in Act$, L denotes a finite subset of $\mathcal{A}$ and $f : \mathcal{A} \longrightarrow \mathcal{A}$ denotes a relabelling function.

The inactive agent, represented by $\mathbf{0}$, is not capable of executing any action; prefix operator defines the execution of sequential actions; choice operator introduces into the language a choice between two alternative behaviours; parallel operator represents the parallel execution of two independent agents, but allowing the synchronisation between them by the execution of a pair of conjugated actions, a and $\overline{a}$, thus producing the internal action denoted by τ; relabelling operator, by means of a function $f : \mathcal{A} \longrightarrow \mathcal{A}$, produces a change in the name of the executed actions, by executing $f(a)$ instead of a and $\overline{f(a)}$ instead of $\overline{a}$; and finally the restriction of the actions in a set L would disallow the execution of actions in $L \cup \overline{L}$.

The operational semantics of CCS formalise the ideas above and can be found in [Mil80, Mil89]. From the operational semantics of processes we can construct the bisimulations and the definition of bisimulation equivalences. Semantic equivalences, and in particular weak bisimulation equivalence [Mil89], have been proposed as a way to formalise the implementation relations, but it seems too strong to use an equivalence relation to accomplish such a task, even if we can abstract away from internal details of the implementation, as allowed by the weak character of that equivalence relation. Instead, conformance relations allow the introduction of new actions in the implementation, when they do not interfere with the rest of the behaviour of the system. This idea has been developed in [Ste94], where the author proposed his *logic conformance* (see definition 2 below).

The classical notation on computation of agents is used: The ability of an agent P to perform some action $\alpha \in Act$ and to evolve into an agent Q is denoted by $P \xrightarrow{\alpha} Q$. Similarly, $P \overset{\alpha}{\Longrightarrow} Q$ is used to denote the ability of P to evolve into Q through the execution of α and any number of additional τ actions. Considering sequences of actions, $s \in Act^*$, the transition relations are naturally extended to get $\xrightarrow{s}$ and $\overset{s}{\Longrightarrow}$, which describe the evolution of an agent when executing a sequence of actions. For the empty sequence we have only the second of this transitions which in this case it is just denoted by $\Longrightarrow$. The hat operator over a sequence of actions, $\hat{s}$, denotes its projection over the set of visible (input and output) actions, so that we have $\hat{s} \in (\mathcal{A} \cup \overline{\mathcal{A}})^*$.

Definition 2 ([Ste94]: Definition 30). *Implementation I* logically conforms *to specification S, written $I \succeq_l S$, iff $\forall \alpha \in Act, \forall \beta \in \overline{\mathcal{A}} \cup \{\tau\}$ and $\forall \gamma \in \mathcal{A}$:*

(1) Whenever $S \xrightarrow{\alpha} S'$ then $\exists I' : I \overset{\hat{\alpha}}{\Longrightarrow} I'$ and $I' \succeq_l S'$.

(2) Whenever $I \xrightarrow{\beta} I'$ then $\exists S' : S \overset{\hat{\beta}}{\Longrightarrow} S'$ and $I' \succeq_l S'$.

(3) Whenever $I \xrightarrow{\gamma} I'$ and $S \overset{\gamma}{\Longrightarrow}$ then $\exists S'$ such that $S \overset{\gamma}{\Longrightarrow} S'$ and $I' \succeq_l S'$.

If we compare this definition with that of plain weak bisimulation we find that the difference comes only from the third clause that allow the implementation to accept additional input actions which are not imposed by the specification.

In [BS02] this definition is considered too strong, and two reasons are argued: (1) an implementation must implement every specified output action, even when there is output concurrency, that is, when multiple output events are produced without interleaving with any input action, and the order of output events is unimportant; (2) it is not possible for the implementation to generate output signals not in the specification.

Then, in order to allow even more flexible implementations a new relation called *weak conformance* is introduced. To define it, they first introduce the *weak conformation* relations defined as follows:

Definition 3 ([BS02]: Definition 7). *A binary process relation $\mathcal{W}$ is a* weak conformation *if $\forall \alpha \in \mathcal{A}(S) \cup \{\tau\}$, $\forall \beta \in \overline{\mathcal{A}}(I) \cup \{\tau\}$, $\forall \gamma \in \mathcal{A}(S)$: $I\, \mathcal{W}\, S$ implies the following four laws:*

Law of Specified Input or Tau (LSIT). *If $S \xrightarrow{\alpha} S'$ then $\exists t \in (\mathcal{A}(S) \cup \overline{Extr}(I,S))^*$ such that*

(1) $I \overset{t}{\Longrightarrow} I'$ (2) $t \upharpoonright \mathcal{A}(S) = \hat{\alpha}$ (3) $I'\, \mathcal{W}\, S'$

Law of Specified Output (LSO). *Let X be a maxoctset of S. $\exists s \in X$ and $\exists t \in \overline{\mathcal{A}}(I)^+$ such that*

(1) $S \overset{s}{\Longrightarrow} S'$ (2) $I \overset{t}{\Longrightarrow} I'$ (3) $t \upharpoonright \overline{\mathcal{A}}(S) = s$ (4) $I'\, \mathcal{W}\, S'$

Law of Implemented Input (LII). *Whenever $I \xrightarrow{\gamma} I'$ and $S \overset{\gamma}{\Longrightarrow}$ then*

(1) $S \overset{\gamma}{\Longrightarrow} S'$ (2) $I'\, \mathcal{W}\, S'$

Law of Implemented Output or Tau (LIOT). *If $I \xrightarrow{\beta} I'$ and $\delta \equiv \beta \upharpoonright \overline{\mathcal{A}}(S)$ then*

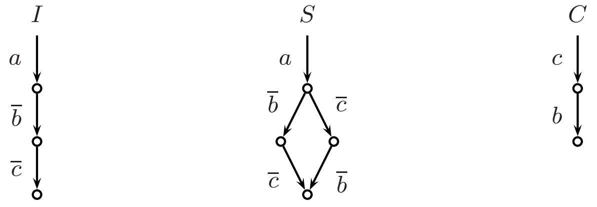
(1) $S \overset{\delta}{\Longrightarrow} S'$ (2) $I'\, \mathcal{W}\, S'$

Where $\mathcal{A}(P)$ and $\overline{\mathcal{A}}(P)$ define the input and output sorts of an agent P, respectively; the binary operator $\upharpoonright$ applies to a sequence s of actions and a set of *actions A, $s\lceil A$, projecting the actions in s over the set A. Besides, $Extr(I,S) = \mathcal{A}(I) - \mathcal{A}(S)$ is called the extraneous input sort and $\overline{Extr}(I,S) = \overline{\mathcal{A}}(I) - \overline{\mathcal{A}}(S)$ is called the extraneous output sort.*

Definition 4 ([BS02]: Definition 9). *The* weak conformance *relation, written $\succeq_W$, is the union of all the weak conformations.*

To formally define the condition capturing their intention of getting a more flexible implementation of output concurrency, a rather complex concept of *maxoctset* (maximal output confluent transition set) is defined in [BS02] and used in LSO rule. The concept of maxoctset tried to capture those maximal partial behaviours of a system which correspond to the parallel execution of several output actions.

But to reduce the output concurrency in the implementation is not compatible with the goal of getting a precongruence from the conformance relation. Let us consider the specification $S = a.(\overline{b}|\overline{c})$. To Reduce the output concurrency implies not to force any implementation of S to implement all the specified output sequences, but just some of them. So, $I = a.\overline{b}.\overline{c}$ would be an adequate implementation of S. But then, we cannot expect this conformance relation to be a precongruence: if we take the agent $C = c.b$ and put it in parallel with the specification S and the implementation I, then we have that the agent $S|C$ can execute the trace $t = ab\overline{b}$, because after executing a action in S, C and S can synchronise and arrive to a state in which they can interleave the actions b and $\overline{b}$. On the contrary, $I|C$ cannot execute such a trace. All this is illustrated in figure 1.



$I \succeq_W S$ but surprisingly $C|I \not\succeq_W C|S$

Fig. 1. Not implementation of output concurrency do not allow $\succeq_W$ to be a congruence

But even if we would not mind this lack of substitutivity, using the definition of maxoctset in [BS02] in order to allow the reduction of output concurrency yields to undesirable implementations. Maxoctsets are maximal traces which correspond to a locally confluent behaviour. A trace t corresponds to a locally confluent behaviour if $P \overset{t}{\Longrightarrow}$ and for each s that is a permutation of t with $P \overset{s}{\Longrightarrow}$, if we have $P \overset{t}{\Longrightarrow} P'$ and $P \overset{s}{\Longrightarrow} P''$, P' and P'' are weak bisimulation equivalent. The authors of [BS02] where too generous allowing that not any permutation of s would be a trace of P. Let us consider the agents in figure 2. If we take $S = \overline{a}.\overline{b}.\overline{c}$, we have that the trace $\overline{a}\overline{b}\overline{c}$ would be a maxoctset of S. But this would be also the case for a specification such as $S' = \overline{a}.\overline{b}.\overline{c} + \overline{a}.\overline{b}.d + \overline{b}.\overline{a}.e$ that has added behaviour. It is clear that the trace $\overline{a}\overline{b}$ is not a locally confluent behaviour of S', but under

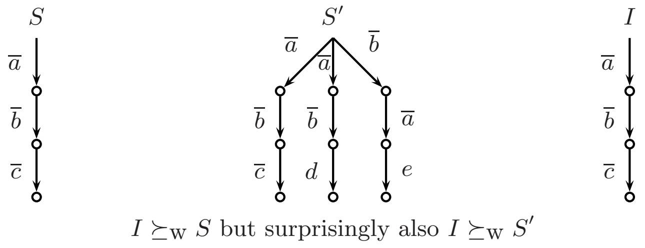


Fig. 2. Maxoctset definition yields to improper implementations

the definition in [BS02] the trace $\overline{a}\overline{b}\overline{c}$ would still be a locally confluent behaviour of S' and then a maxoctset of it. As a consequence the behaviours of S' after the execution of the traces $\overline{a}\overline{b}$ and $\overline{b}\overline{a}$ would not be considered when checking the weak conformance of any implementation. Then, $I = \overline{a}.\overline{b}.\overline{c}$, where neither d nor e can be executed, would be considered to be an admissible implementation of S', which does not seem reasonable at all.

Finally, to conclude with the comments on [BS02], we show one more flaw on the definition of the weak conformance relation arising from the way rule LII is asserted (definition 3). Transitivity of $\succeq_W$ is not guaranteed just by imposing that an implementation would not execute any extraneous output action in the beginning.

Let us consider the agents in figure 3: $Q = a.\overline{v}.b.c$ is an implementation of $R = a.b.c$, because rule LSIT allows the introduction of output actions, provided that they do not appear in the specification. Rule LII also allows to introduce new input actions into the implementation, but it is too generous since it is just imposed that these actions could not be executed by the initial state of the specification. So, agent $P = a.(\overline{v}.b.c + b)$ is an implementation of Q. But the added choice executing the action b makes P not to be an implementation of R, thus spoiling transitivity of the conformance relation.

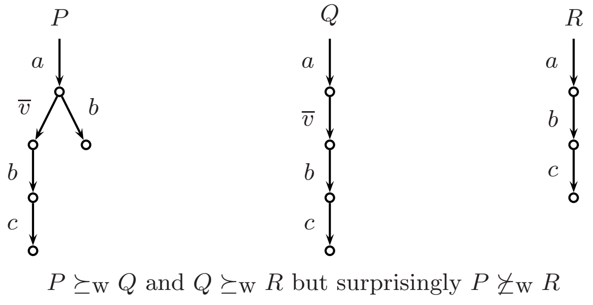

P
Q
R
$P \succeq_W Q$ and $Q \succeq_W R$ but surprisingly $P \not\succeq_W R$

Fig. 3. $\succeq_W$ relation is not transitive

3 Soft Conformance Relation

In this section we present a new variant of conformation relations, that we call *soft conformations.* The union of all soft conformation relations define the *soft* *conformance*, denoted by $\succeq_S$. This new notion gathers the desired conditions discussed above, namely the capability of the implementation to introduce new input and output actions in its behaviour.

In order to give the definition of soft conformations some new notation has to be introduced. As usual $\longrightarrow$ and $\Longrightarrow$ relations denote the capability of an agent to evolve through a single action or an action preceded and followed by any number of τ actions, respectively. Besides, we introduce the new transition relation $\Longmapsto$ that gathers the idea that once a specification is fixed, the extraneous output actions in the implementations play the same role as τ actions. That is, given the specification S and the implementation I, $I \overset{\alpha}{\Longmapsto}_S I'$ indicates that I evolves to I' after executing the action α preceded and followed by any number of transitions executing either τ actions or output actions $\overline{b} \in \overline{Extr}(I, S)$. For the sake of simplicity, if S is clear from the context where the $\Longmapsto$ relation is used, we will avoid the subscript S, writing just $I \overset{\alpha}{\Longmapsto} I'$.

In order to define soft conformations relations, the sort of an agent, that is, the set of actions that it could possibly execute, has to be introduced:

Definition 5. *The* set of executable actions *of an agent E, denoted by $Exec(E)$, is inductively defined as follows:*

- $Exec(\boldsymbol{0}) = \emptyset$
- $Exec(\alpha.E) = \{\alpha\} \cup Exec(E)$
- $Exec(E_1 + E_2) = Exec(E_1|E_2) = Exec(E_1) \cup Exec(E_2)$
- $Exec(E[f]) = f(Exec(E))$
- $Exec(E\backslash L) = Exec(E) - L$

Definition 6. *Binary process relation $\mathcal{V}$ is a* soft conformation *if $\forall\alpha \in Act$, $\forall a \in \mathcal{A}(S)$, $\forall\beta \in \overline{\mathcal{A}}(I) \cup \{\tau\}$: $I\,\mathcal{V}\,S$ implies that $Exec(S) \subseteq Exec(I)$ and the following laws are satisfied:*

Law of Specified Behaviour (LSB)

If $S \xrightarrow{\alpha} S'$ then $\exists I' : I \overset{\hat{\alpha}}{\Longmapsto} I'$ and $I'\,\mathcal{V}\,S'$.

Law of Implemented Input (LII)

If $I \xrightarrow{a} I'$ and $a \in Exec(S)$ then $\exists S' : S \overset{a}{\Longrightarrow} S'$ and $I'\,\mathcal{V}\,S'$.

Law of Implemented Output or Tau (LIOT)

If $I \xrightarrow{\beta} I'$ and $\beta \in Exec(S)$ then $\exists S' : S \overset{\beta}{\Longrightarrow} S'$ and $I'\,\mathcal{V}\,S'$.

If $I \xrightarrow{\beta} I'$ and $\beta \notin Exec(S)$ then $\exists S' : S \Longrightarrow S'$ and $I'\,\mathcal{V}\,S'$.

Therefore the differences between our soft conformations and the weak conformations in definition 3 are that we have drop out the considerations about output concurrency and then the two laws of specified input and output have became a single law; besides in LII rule we only allow the additional execution by the implementation of extraneous input actions.

Proposition 1. *Let $\mathcal{V}$ and $\mathcal{V}'$ be soft conformation relations, then*

(1) The identity relation is a soft conformation relation.
(2) The composition $\mathcal{V}\ \mathcal{V}'$ is a soft conformation relation.
(3) The union $\mathcal{V} \cup \mathcal{V}'$ is a soft conformation relation.

Definition 7. *The implementation I is said to* softly conform *to the specification S, denoted by $I \succeq_S S$, if there exists some soft conformation relation $\mathcal{V}$ with $I\,\mathcal{V}\,S$. That is, the* soft conformance *relation, denoted by $\succeq_S$, is the union of all the soft conformation relations.*

4 Contexts That Preserve Soft Conformance

In this section we address the main goal of our paper: given a specification S and an a soft conformance implementation of it, $I \succeq_{\mathrm{S}} S$, to determine the properties that a context $C(X)$ should verify in order to get $C(I) \succeq_{\mathrm{S}} C(S)$. We will start by formalising the concept of context. In order to get a simpler presentation, we will first just consider contexts with a single hole.

Definition 8. *Given a set of actions Act, the set of contexts is defined by the following BNF-expression:*

$$C ::= \mathbf{0} \;\mid\; X \;\mid\; \alpha.C \;\mid\; E+C \;\mid\; C+E \;\mid\; E|C \;\mid\; C|E \;\mid\; C[f] \;\mid\; C\backslash L$$

where X represents a single (hole) variable, E represents CCS agents (definition 1), $\alpha \in Act$, L denotes a finite subset of $\mathcal{A}$ and $f : \mathcal{A} \longrightarrow \mathcal{A}$ denotes a relabelling function.

The operational semantics of contexts is defined in the same way as agents, since there is no rule for the hole X.

To define the conditions that contexts have to satisfy in order to preserve the soft conformance relation we need to use a collection of auxiliary functions and predicates that we will define below. All of them are defined by structural induction.

Definition 9. *The following functions are defined over both contexts and agents:*

Exec() *computes the set of executable actions of a context.*

$$\begin{aligned}
Exec(X) &= Exec(\mathbf{0}) = \emptyset\\
Exec(\alpha.C) &= \{\alpha\} \cup Exec(C)\\
Exec(E+C) &= Exec(C+E) = Exec(E|C) = Exec(C|E) = Exec(C) \cup Exec(E)\\
Exec(C[f]) &= f(Exec(C))\\
Exec(C\backslash L) &= Exec(C) - L
\end{aligned}$$

Init() *computes the set of initials actions that a context can execute.*

$$\begin{aligned}
Init(X) &= Init(\mathbf{0}) = \emptyset\\
Init(\alpha.C) &= \{\alpha\}\\
Init(E+C) &= Init(C+E) = Init(C) \cup Init(E)\\
Init(E|C) &= Init(C|E) = Init(C) \cup Init(E) \cup \{\tau \mid \mathit{if}\ \exists \alpha \in Init(E), \overline{\alpha} \in Init(C)\}\\
Init(C[f]) &= f(Init(C))\\
Init(C\backslash L) &= Init(C) - L
\end{aligned}$$

Guar() *defines a boolean function that indicates whether a context has its hole guarded by an action.*

$$\begin{aligned}
Guar(X) &= \mathit{false}\\
Guar(\mathbf{0}) &= \mathit{true}\\
Guar(\alpha.C) &= \mathit{true}\\
Guar(E+C) &= Guar(C+E) = Guar(E|C) = Guar(C|E) =\\
&\ Guar(C[f]) = Guar(C\backslash L) = Guar(C)
\end{aligned}$$

Choice-app() *defines a boolean function that indicates if a choice operator applies directly on the hole of a context.*

$$\begin{aligned} \textit{Choice-app}(X) &= \textit{Choice-app}(\boldsymbol{0}) &&= \textit{false} \\ \textit{Choice-app}(\alpha.C) &= \textit{Choice-app}(C) \\ \textit{Choice-app}(E+C) &= \textit{Choice-app}(C+E) &&= \textit{Choice-app}(C) \vee \neg \textit{Guar}(C) \\ \textit{Choice-app}(E|C) &= \textit{Choice-app}(C|E) &&= \textit{Choice-app}(C[f]) \quad = \\ &\ \textit{Choice-app}(C\backslash L) &&= \textit{Choice-app}(C) \end{aligned}$$

Exec-par() *defines the set of actions that can be executed in parallel with the hole and the context.*

$$\begin{aligned} \textit{Exec-par}(X) &= \textit{Exec-par}(\boldsymbol{0}) &&= \emptyset \\ \textit{Exec-par}(\alpha.C) &= \textit{Exec-par}(C) \\ \textit{Exec-par}(E+C) &= \textit{Exec-par}(C+E) &&= \textit{Exec-par}(C) \cup \textit{Exec-par}(E) \\ \textit{Exec-par}(E|C) &= \textit{Exec-par}(C|E) &&= \textit{Init}(C|E) \\ \textit{Exec-par}(C[f]) &= f(\textit{Exec-par}(C)) \\ \textit{Exec-par}(C\backslash L) &= \textit{Exec-par}(C) - L \end{aligned}$$

Rest() *defines the set of restricted actions over the hole in the context.*

$$\begin{aligned} \textit{Rest}(X) &= \textit{Rest}(\boldsymbol{0}) \quad = \emptyset \\ \textit{Rest}(\alpha.C) &= \textit{Rest}(C) \\ \textit{Rest}(E+C) &= \textit{Rest}(C+E) = \textit{Rest}(E|C) = \textit{Rest}(C|E) = \textit{Rest}(C) \\ \textit{Rest}(C[f]) &= f(\textit{Rest}(C)) \\ \textit{Rest}(C\backslash L) &= \textit{Rest}(C) \cup L \end{aligned}$$

Renamed() *defines the set of actions that are either renamed or renamed to over the hole in the context.*

$$\begin{aligned} \textit{Renamed}(X) &= \textit{Renamed}(\boldsymbol{0}) &&= \emptyset \\ \textit{Renamed}(\alpha.C) &= \textit{Renamed}(C) \\ \textit{Renamed}(E+C) &= \textit{Renamed}(C+E) &&= \textit{Renamed}(E|C) = \\ &\ \textit{Renamed}(C|E) &&= \textit{Renamed}(C) \\ \textit{Renamed}(C[f]) &= \textit{Renamed}(C) \cup \{a \mid f(a) \neq a \vee \exists b \neq a : f(b) = a\} \\ \textit{Renamed}(C\backslash L) &= \textit{Renamed}(C) - L \end{aligned}$$

From the previous functions, we define the following ones:

$\overline{\textbf{\textit{Init}}}$() *defines the initial output or τ actions that a context can execute.*

$$\overline{\textit{Init}}(C) = \textit{Init}(C) \cap (\overline{A} \cup \{\tau\})$$

$\overline{\textbf{\textit{Init-extr}}}$() *defines the initial output extraneous actions of I with respect to S.*

$$\overline{\textit{Init-extr}}(I,S) = \overline{\textit{Init}}(I) \cap (\overline{\textit{Extr}}(I,S) \cup \{\tau\})$$

$\dot{\textbf{\textit{IOExtr}}}$(,) *defines the union of extraneous input and output actions.*

$$\textit{IOExtr}(I,S) = \textit{Extr}(I,S) \cup \overline{\textit{Extr}}(I,S)$$

$\overline{\textbf{\textit{Exec-par}}}()$ *defines the* complementary *set of actions with respect to the set* *Exec-par*().

$$\overline{\textit{Exec-par}(C)} = \{\overline{\alpha} \mid \alpha \in \textit{Exec-par}(C)\}$$

where $\overline{\overline{\alpha}} = \alpha$ *and* $\overline{\tau} = \tau$.

The following proposition explains which is the relation between the syntactically defined functions and predicates introduced in definition 9 and the semantic behaviour of the involved elements.

Proposition 2. *The functions declared in definition 9 verify the following characteristic properties:*

1. *For any action* α *in any trace* s *such that* $C \overset{s}{\Longrightarrow}$ *it holds that* $\alpha \in \textit{Exec}(C)$.
2. *For any action* $\beta \in \overline{\textit{Extr}}(I,S) \cup \{\tau\}$ *such that* $C \xrightarrow{\beta}$ *then* $\beta \in \overline{\textit{Init-extr}}(I,S)$.
3. *If there exists some computation* $C(X) \overset{s}{\Longrightarrow} C'(X)$ *such that* $C'(\omega.\omega') \xrightarrow{\omega} C''(\omega')$ *with* $C' \neq C''$ *then* *Choice-app*(C) *is true.*
4. *If there exists some computation* $C(X) \overset{s}{\Longrightarrow} C'(X)$ *such that* $C'(X) \xrightarrow{\overline{\alpha}} C''(X)$ *and* $C'(\alpha.\omega) \xrightarrow{\tau} C''(\omega)$, *by the synchronisation of the first transition with the execution of action* α *in the hole, then* $\overline{\alpha} \in \textit{Exec-par}(C)$.
5. *If there exists some computation* $C(X) \overset{s}{\Longrightarrow} C'(X)$ *such that* $C'(\omega.\omega') \xrightarrow{\omega} C''(\omega')$ *but* $C'(\alpha.\omega') \overset{\alpha}{\not\longrightarrow}$, *then* $\alpha \in \textit{Rest}(C)$.
6. *If* $a \notin \textit{Renamed}(C(X)) \wedge \overline{a} \notin \textit{Renamed}(C(X))$ *and there exists some computation* $C(X) \overset{s}{\Longrightarrow} C'(X)$ *such that* $C'(a.\omega') \xrightarrow{\alpha} C''(\omega')$ *then* $a = \alpha$.

Where ω *and* ω' *denote* fresh *actions that are not in* $\textit{Exec}(C) \cup \textit{Exec}(I)$.

We can now give the conditions that determine when a context preserve the soft conformance relation that holds between two agents.

Definition 10. *Given two agents* I *and* S, $I \succeq_S S$, *and a context* C, *it is said that* C *is a* preserving context *with respect to the pair* (I,S) *if the following five conditions are fulfilled:*

i. $\textit{IOExtr}(I,S) \cap \textit{Exec}(C) = \emptyset$
ii. $\neg \textit{Choice-app}(C) \vee (\overline{\textit{Init-extr}}(I,S) = \emptyset)$
iii. $\textit{IOExtr}(I,S) \cap \overline{\textit{Exec-par}(C)} = \emptyset$
iv. $\overline{\textit{Extr}}(I,S) \cap \textit{Rest}(C) = \emptyset$
v. $\textit{Renamed}(C) \cap \textit{IOExtr}(I,S) = \emptyset$

Next proposition tie together some properties that will be useful when proving the main result of our paper: the preservation theorem.

Proposition 3. *If* C *is a preserving context with respect to the pair* (I,S), *where the* X *appears, then the following properties are satisfied:*

1. $\textit{IOExtr}(C(I), C(S)) = \textit{IOExtr}(I,S)$
2. *If* $C(X) \overset{\alpha}{\Longrightarrow} C'(X)$, *then* C' *is also a preserving context with respect to the pair* (I,S).
3. *If* $C(\omega.\omega') \xrightarrow{\omega} C'(\omega')$ *then for any* $P \overset{s}{\Longrightarrow} P'$ *with* $s = \alpha_1 \ldots \alpha_n$, *and* $\forall i\ \alpha_i \notin \textit{Renamed}(C)$ *and* $\alpha_i \notin \textit{Rest}(C)$ *we have also* $C(P) \overset{s}{\Longrightarrow} C'(P')$.

4. *If $I \stackrel{\hat{\alpha}}{\Longmapsto} I'$ and $C(\omega.\omega') \xrightarrow{\omega} C'(\omega')$ and $C(S) \xrightarrow{\alpha} C'(S')$ then $C(I) \stackrel{\hat{\alpha}}{\Longmapsto} C'(I')$ and C' is a preserving context with respect to (I', S').*
5. *If $S \xrightarrow{\beta} S'$ and $C(X) \xrightarrow{\overline{\beta}} C'(X)$ and it is possible to get a synchronisation step $C(S) \xrightarrow{\tau} C'(S')$, then $C(I) \xrightarrow{\tau} C'(I')$ and C' is a preserving context with respect to (I', S').*

Proof. *Let us prove the previous statements:*

1. *By a simple structural induction over the form of the contexts.*
2. *If C' is such a derived context from C then:*

$$\begin{aligned} Exec(C') &\subseteq Exec(C) \\ Choice\text{-}app(C') &\Rightarrow Choice\text{-}app(C) \\ Exec\text{-}par(C') &\subseteq Exec\text{-}par(C) \\ Rest(C') &= Rest(C) \\ Renamed(C') &= Renamed(C) \end{aligned}$$

   and therefore C' verify the conditions (i)... (v) with respect to (I, S).
3. *By structured induction over the form of C:*
   - *$C = a.C''$. This cannot be the case, since then $C(\omega.\omega') \stackrel{\omega}{\not\rightarrow}$*
   - *$C = Q + C''$. If $C(\omega.\omega') \xrightarrow{\omega} C'(\omega')$ then we should have $C''(\omega.\omega') \xrightarrow{\omega} C'(\omega')$, and by induction hypothesis $C''(P) \stackrel{s}{\Longrightarrow} C'(P')$ and therefore $C(P) \stackrel{s}{\Longrightarrow} C'(P')$.*
   - *$C = Q|C''$. If $C(\omega.\omega') \xrightarrow{\omega} C'(\omega')$ we have $C' = Q|C'''$ with $C''(\omega.\omega') \xrightarrow{\omega} C'''(\omega')$. Then we have $C''(P) \stackrel{s}{\Longrightarrow} C'''(P')$ and $C(P) \stackrel{s}{\Longrightarrow} Q|C'''(P') = C'(P')$.*
   - *$C = C''[f]$. If $C(\omega.\omega') \xrightarrow{\omega} C'(\omega')$ we have also $C''(\omega.\omega') \xrightarrow{\omega} C'''(\omega')$ with $C' = C'''[f]$. Then we have $C''(P) \stackrel{s}{\Longrightarrow} C'''(P')$ and since f does not rename any action in s we have also $C(P) \stackrel{s}{\Longrightarrow} C'(P')$.*
   - *$C = C''\backslash L$. If $C(\omega.\omega') \xrightarrow{\omega} C'(\omega')$ we have also $C''(\omega.\omega') \xrightarrow{\omega} C'''(\omega')$ with $C' = C'''\backslash L$. Then we have $C''(P) \stackrel{s}{\Longrightarrow} C'''(P')$ and since the actions in s are not in $Rest(C)$ in particular the are not in L, and therefore $C(P) \stackrel{s}{\Longrightarrow} C'(P')$.*
4. *Let us consider the sequence of visible actions s which corresponds to the computation $I \stackrel{\hat{\alpha}}{\Longmapsto}{}'_{I}$, then since $C(S) \xrightarrow{\alpha} C'(S')$ and the rest of the actions in s are also in $IOExtr(I, S)$, by proposition 3(1), $C(I) \stackrel{\hat{\alpha}}{\Longmapsto} C'(I')$.*
   The proof that C' is a preserving context with respect to (I', S') is similar to that of proposition 3(2) considering that $Choice\text{-}app(C')$ is always false. This statement can be proved by structural induction as before.
5. *Similar to the previous one.*

□

Theorem 1 (Preservation theorem). *If $I \succeq_S S$ and C is a preserving context of the pair (I, S) then $C(I) \succeq_S C(S)$.*

Proof. We have to prove that there exists a soft conformation relation $\mathcal{V}$ containing the pair $\langle C(I), C(S)\rangle$. We define

$$\mathcal{V} = \{\langle C(I), C(S)\rangle \ \mid \ I \succeq_S S \text{ and } C \text{ is a preserving context w.r.t. } (I, S)\}$$

and we will check that $\mathcal{V}$ verifies the laws in definition 6.

LSB. *Let us suppose that $C(S) \xrightarrow{\alpha} T$. There are three different possibilities:*

1. *$T = C'(S)$ and $C(X) \xrightarrow{\alpha} C'(X)$, then $C(I) \xrightarrow{\alpha} C'(I)$ and, by proposition 3, C' is a preserving context with respect to (I, S) and therefore $C'(I)$ and $C'(S)$ are in the conformation relation $\mathcal{V}$.*
2. *$S \xrightarrow{\alpha} S'$ and $C(S) \xrightarrow{\alpha} C'(S')$, where the context C' is derived from C by means of the execution of an action of its hole. From $I \succeq_S S$ we know that $I \overset{\hat{\alpha}}{\Longmapsto}_S I'$ and $I' \succeq_S S'$ and then, by proposition 3(4) $C(I) \overset{\hat{\alpha}}{\Longmapsto} C(I')$ and C' is a preserving context with respect to (I', S') and therefore $C'(I') \,\mathcal{V}\, C'(S')$.*
3. *Finally, if $\alpha = \tau$ and there exists some β such that $S \xrightarrow{\beta} S'$, and $C(X) \xrightarrow{\overline{\beta}} C'(X)$ and there is a synchronisation step of these two complementary actions which produces $C(S) \xrightarrow{\tau} C'(S')$. Then, $I \overset{\beta}{\Longrightarrow} I'$ and $I' \succeq_S S'$, considering proposition 3(5), we have $C(I) \xrightarrow{\tau} C'(I')$, and combining the arguments in the two previous cases we get that C' is a preserving context with respect to (I', S'), and therefore $C'(I') \,\mathcal{V}\, C'(S')$.*

LII. *Let us suppose that $C(I) \xrightarrow{a} T$ and $a \in Exec(C(S))$, then two cases should be considered:*

1. *$C(X) \xrightarrow{a} C'(X)$ and $T = C'(I)$ then, by proposition 3(2), C' is a preserving context with respect to (I, S) and therefore $C'(I) \,\mathcal{V}\, C'(S)$.*
2. *$I \xrightarrow{a'} I'$ and $C(I) \xrightarrow{a} C'(I')$ where the context C' is derived from C by means of the execution of an action of its hole, and $a \in Renamed(C)$. By definition 10(i), $a' \in Exec(S)$, by proposition 2(1), $a' \in Exec(I)$ and if it were the case that $a' \notin Exec(S)$ then $a' \in Extr(I, S)$ and by using conditions (i), (iv) and (v) of definition 10, we would conclude that $a \in Extr(C(I), C(S))$ and then $a \notin Exec(C(S))$ against the hypothesis.*

   Then, since $I \succeq_S S$ we have that $S \overset{a'}{\Longrightarrow} S'$ with $I' \succeq_S S'$ and, reasoning as in LSB rule, we get that $C(S) \overset{a}{\Longrightarrow} C'(S')$ with C' a preserving context with respect to (I', S'), so that $C'(I') \,\mathcal{V}\, C'(S')$.

LIOT. *We consider two cases:*

1. *$C(I) \xrightarrow{\overline{a}} T$ with $\overline{a} \in Exec(C(S))$. Once again we consider two cases:*
   - *(a) $C(X) \xrightarrow{\overline{a}} C'(X)$ and $T = C'(I)$, then by proposition 3(2), C' is a preserving context with respect to (I, S) and therefore $C'(I) \,\mathcal{V}\, C'(S)$.*
   - *(b) $I \xrightarrow{\overline{a'}} I'$ and $C(I) \xrightarrow{\overline{a}} C'(I')$ for C' derived from C by means of the execution of an action in its hole, and $a \in Renamed(C)$. Then, we have that $S \overset{\overline{a'}}{\Longrightarrow} S'$ with $I' \succeq_S S'$ and therefore, as in the previous case, we conclude that $C'(I') \,\mathcal{V}\, C'(S')$.*
2. *$C(I) \xrightarrow{\beta} T$ were $\beta = \tau$ or $\beta = \overline{a} \notin Exec(C(I))$. Now there are three possible cases:*

(a) $C(X) \xrightarrow{\beta} C'(X)$ *and* $T = C'(I)$*, as in the corresponding case above,* C' *is a preserving context with respect to* (I, S) *and as a consequence* $C'(I)\,\mathcal{V}\,C'(S)$*.*

(b) $I \xrightarrow{\beta'} I'$ *and* $C(I) \xrightarrow{\beta} C'(I')$ *with* $\beta \in Renamed(C)$*. Due to condition (ii) in definition 10 this case is only possible if* $C' = C$*. Then, by proposition 2(2), the hole* X *in* C *cannot be under a choice operator and so* $S \overset{\beta'}{\Longrightarrow} S'$ *and* $C(S) \overset{\beta}{\Longrightarrow} C(S')$*, with* C *preserving* (I', S')*, and therefore* $C(I')\,\mathcal{V}\,C(S')$*.*

(c) $\beta = \tau$ *and there exists some* γ' *such that* $I \xrightarrow{\gamma} I'$*, and* $C(X) \xrightarrow{\overline{\gamma}} C'(X)$ *after a renaming of* γ' *into* γ *and there is a synchronisation step of these two actions which produces* $C(I) \xrightarrow{\tau} C'(I)$*. Then* $\gamma' \in Exec(S)$ *because* C *verifies condition (iii) in definition 10, and we can proceed either as in the previous case (1) of the LIOT rule, or as for the law LII to conclude that* $S \overset{\gamma'}{\Longrightarrow} S'$ *and* $C(S) \overset{\gamma}{\Longrightarrow} C'(S')$ *where* C' *is a preserving context with respect to the pair* (I', S')*, thus concluding that* $C'(I')\,\mathcal{V}\,C'(S')$*.*

$\square$

In definition 8, contexts with a single hole were defined and the preservation theorem proves that preserving contexts (definition 10) allow the substitutivity of agents that are in soft conformation getting a new pair of agents in soft conformation. We next generalised the results to contexts with a finite set of variable names.

Definition 11. *Let us consider a (finite) set of* hole *variables* $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_k\}$*. We define* generalised contexts *exactly as simple contexts (definition 8) but changing the unique symbol* X *by a representative element of the set* $\mathcal{X}$*, and replacing all the metavariables* C *in that definition by that corresponding to a generalised context* $\mathcal{C}$*.*

We do not want to have contexts with repeated appearances of the same hole, therefore, we forbid such possibility and concentrate on what we call *valid generalised contexts.*

Definition 12. *The following function and predicate are defined over generalised contexts:*

Holes() *computes the set of hole variables of a generalised context.*

$$\begin{aligned} Holes(\mathcal{X}_i) &= \{\mathcal{X}_i\} \\ Holes(\alpha.\mathcal{C}) &= Holes(\mathcal{C}[f]) = Holes(\mathcal{C}\backslash L) = Holes(\mathcal{C}) \\ Holes(\mathcal{C}_1 + \mathcal{C}_2) &= Holes(\mathcal{C}_1|\mathcal{C}_2) = Holes(\mathcal{C}_1) \cup Holes(\mathcal{C}_2) \end{aligned}$$

Valid() *indicates if a generalised context has no hole names repeated.*

$$\begin{aligned} Valid(\mathcal{X}_i) &= true \\ Valid(\alpha.\mathcal{C}) &= Valid(\mathcal{C}[f]) = Valid(\mathcal{C}\backslash L) = Valid(\mathcal{C}) \\ Valid(\mathcal{C}_1 + \mathcal{C}_2) &= Valid(\mathcal{C}_1|\mathcal{C}_2) = \\ &\quad Valid(\mathcal{C}_1) \wedge Valid(\mathcal{C}_2) \wedge (Holes(\mathcal{C}_1) \cap Holes(\mathcal{C}_2) = \emptyset) \end{aligned}$$

Par-holes() *a binary function that applies on valid generalised contexts, that is $Valid(\mathcal{C}) = true$ and $\mathcal{X}_i \in \mathcal{X}$, and computes the set of hole names that are in the context $\mathcal{C}$ in parallel with the given hole name X_i.*

$$\begin{aligned}
\mathit{Par\text{-}holes}(\mathcal{X}_j, \mathcal{X}_i) &= \emptyset \\
\mathit{Par\text{-}holes}(\alpha.\mathcal{C}, \mathcal{X}_i) &= \mathit{Par\text{-}holes}(\mathcal{C}[f], \mathcal{X}_i) = \mathit{Par\text{-}holes}(\mathcal{C}\backslash L, \mathcal{X}_i) = \\
& \mathit{Par\text{-}holes}(\mathcal{C}, \mathcal{X}_i) \\
\mathit{Par\text{-}holes}(\mathcal{C}_1 + \mathcal{C}_2, \mathcal{X}_i) &= \mathit{Par\text{-}holes}(\mathcal{C}_1, \mathcal{X}_i) \cup \mathit{Par\text{-}holes}(\mathcal{C}_2, \mathcal{X}_i) \\
\mathit{Par\text{-}holes}(\mathcal{C}_1 | \mathcal{C}_2, \mathcal{X}_i) &= \begin{cases} \mathit{Par\text{-}holes}(\mathcal{C}_1, \mathcal{X}_i) \cup \mathit{Holes}(\mathcal{C}_2) & \mathit{if}\ X_i \in \mathit{Holes}(\mathcal{C}_1) \\ \mathit{Par\text{-}holes}(\mathcal{C}_2, \mathcal{X}_i) \cup \mathit{Holes}(\mathcal{C}_1) & \mathit{if}\ X_i \in \mathit{Holes}(\mathcal{C}_2) \\ \emptyset \qquad \mathit{if}\ \mathcal{X}_j \notin \mathit{Holes}(\mathcal{C}_1) \cup \mathit{Holes}(\mathcal{C}_2) \end{cases}
\end{aligned}$$

Definition 13. *Given a set of hole variables $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_k\}$ and a family of pairs of agents $\mathcal{F} = \{(I_i, S_i)\}_{i \in 1..k}$ and a valid generalised context $\mathcal{C}$, we say that it is a* preserving generalised context *with respect to the family $\mathcal{F}$ if, besides the conditions in definition 10, for each pair (I_i, S_i) with $\mathcal{X}_i \in \mathit{Holes}(\mathcal{C})$ we have also*

vi. For each i, j with $\mathcal{X}_i \in \mathit{Holes}(\mathcal{C})$

$$\mathit{IOExtr}(I_i, S_i) \cap \mathit{Exec}(S_j) = \emptyset$$

vii. For each $\mathcal{X}_i \in \mathit{Holes}(\mathcal{C})$ and $\mathcal{X}_j \in \mathit{Par\text{-}holes}(\mathcal{C}, \mathcal{X}_i)$

$$\mathit{IOExtr}(I_i, S_i) \cap \overline{\mathit{Exec}(I_j)} = \emptyset$$

Theorem 2. *If $\mathcal{C}$ is a preserving generalised context with respect to a family $\mathcal{F} = \{(I_i, S_i)\}_{i \in 1..k}$ and for each $i \in 1..k$ we have that $I_i \succeq_S S_i$ then $\mathcal{C}(\overline{I}) \succeq_S \mathcal{C}(\overline{S})$ where, as usual, $\mathcal{C}(\overline{E})$ denotes the substitution of the hole variables $\mathcal{X}_i$ in $\mathcal{C}$ by the corresponding agent E_i.*

5 Conclusions and Future Work

Conformance relations define when a communicating system should be considered a correct implementation of a given specification. In this paper we have studied the conditions under which a context preserves a bisimulation based conformance. It is clear that as soon as we allow extraneous actions in an admissible implementation then there exists a malicious context that would not preserve that conformance relation, and then the only preorder stronger than it being a precongruence would be weak bisimulation, that does not allow the introduction of any extraneous actions with respect to the given specification.

Therefore such a characterisation of the family of contexts preserving each individual pair in the conformance relation is the best result that we can expect in this direction.

In order to get a clearer exposition, and simpler proofs, we have not considered either recursive agents or contexts containing recursive components (without hole variables involved), but it would not be difficult to extend our results to cover also these recursive sceneries. Instead, we think it would be more complicated to extend the results to cover the case in which it is the recursive construction itself that we want to preserve the conformance relation. In such a case, it is necessary to decide how the conformance relation should be extended to the case in which we have higher order agents where the free variables are introduced to be instantiated by first order agents. This question is far from being simple, as studied in detail in [Ren00].

In [BdFMM00] it is shown that tile bisimulation, where weak bisimulation is extended to contextualized processes in a very algebraic way, is not always a congruence, and it is also discussed under which conditions it is possible to get the preservation of that relation. We are interested on a more thorough study of the relations between our paper and this mentioned work.

Besides, [dFLN99] studied several notions of global bisimulation, where weak bisimulation is relaxed by allowing more flexible moves when playing the bisimulation game. Again, there were problems when trying to get a congruence, and therefore it would be also interesting to compare that work with the results and ideas in the current paper.

Also into the testing based conformances, the problem of getting precongruences should be more intensively studied, as appointed in [Led91, dFLN97]. The relations between testing based and bisimulation based conformances deserves, in our opinion, a deeper study, not only in the [Abr87] style, where the testing semantics is presented as a bisimulation semantics, but also on the opposite way, as in [dFLN99], where bisimulation semantics are presented as testing semantics.

References

[Abr87] Samson Abramsky. Observational equivalence as a testing equivalence. *Theoretical Computer Science*, 53(3):225–241, 1987.

[BdFMM00] Roberto Bruni, David de Frutos-Escrig, Narciso Martí-Oliet, and Ugo Montanari. Bisimilarity congruences for open terms and term graphs via tile logic. In *CONCUR*, volume 1877 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2000.

[Bri88] E. Brinksma. A theory for the derivation of tests. In *Protocol Specification, Testing and Verification VIII*, pages 63–74. North Holland, 1988.

[BS02] Ronald W. Brower and Kenneth S. Stevens. Congruent weak conformance, a partial order among processes. In *Formal Techniques for Networked and Distributed Systems-FORTE 2002*, volume 2529 of *Lecture Notes in Computer Science*, pages 34–49. Springer, 2002.

[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. North Holland, 1986.

[dFLN97] D. de Frutos-Escrig, L.F. Llana-Díaz, and M. Núñez. Friendly testing as a conformance relation. In *Formal Description Techniques and Protocol Specification, Testing, and Verification FORTE X/ PSTV XVII*, pages 283–298. Chapman & Hall, 1997.

[dFLN98] D. de Frutos-Escrig, L.F. Llana-Díaz, and M. Núñez. An invitation to friendly testing. *Journal of Computer Science and Technology*, 13(6):531–545, 1998.

[dFLN99] David de Frutos-Escrig, Natalia López, and Manuel Núñez. Global timed bisimulation: An introduction. In *Formal Methods for Protocol Engineering and Distributed Systems, FORTE XII / PSTV XIX*, pages 401–416. Kluwer Academic Publishers, 1999.

[Hen88] Matthew 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. North Holland, 1991.

[Led92] G. Leduc. A framework based on implementation relations for implementing LOTOS specifications. *Computer Networks and ISDN Systems*, 25(1):23–41, 1992.

[Mil80] Robin Milner. *A Calculus of Communicating Systems.* LNCS 92. Springer, 1980.

[Mil89] Robin Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[Ren00] Arend Rensink. Bisimilarity of open terms. *Information and Computation*, 156(1–2):345–385, 2000.

[Ste94] Kenneth S. Stevens. *Practical Verification and Synthesis of Low Latency Asynchronous Systems.* PhD thesis, University of Calgary, 1994.