Generating interoperability test cases from conformance test case generation tools

Ismail Berrada, Richard Castanet, Patrick Felix

Generating Interoperability Test Cases from Conformance Test Case Generation Tools

Ismail Berrada, Richard Castanet and Patrick Félix

LaBRI - CNRS - UMR 5800 Université Bordeaux 1
33405 Talence cedex, France
{ berrada | castanet | felix}@labri.fr

Abstract. This paper focuses on studing efficient solutions for deriving interoperability tests. In this work, we propose first a coherent formal framework for interoperability testing based on a parametrized writing of the interoperability relations. An approach and associated algorithm is then introduced for this framework. As a consequence, adaptation of our results to conformance testing tools for generating interoperability tests is fully automatic. By the way, we show that the proposed approach applies for testing in context.

1 Introduction

Protocol specifications are used to develop products and services. To ensure correctness of such products, testing, the process of checking that a system possesses a set of desired properties and/or behaviors, is the one of the used validation techniques. Various testing methodologies have been proposed and applied. One approach is conformance testing, in which a single implementation is compared to the standard to be sure that the implementation does what the standard specifies; the theory behind conformance testing is that all conform implementations to the abstract standard should interoperate with each other, although in practice this is not necessarily the case. The other approach is interoperability testing, in which two or more implementations are tested directly against each other, with the standard used as a primarily as a reference to adjudicate problems and incompatibilities, and secondarily as a guide to the functions to be tested and the general behavior to be expected.

Conformance testing inspires little confidence in the user community. In contrast, conformance testing potentially provides benefits to an implementor during the early stages of development as a simple check. A limitation of conformance testing become from the generality of the protocols being tested, particularly at the OSI model applications layer, protocols are typically very complex and general, offering many options and choices.

In order to decide the correctness of an implementation, a clear criterion is needed. In the context of conformance testing, many proposals for such correctness in the form of implementation relations have been made [1, 2]. Unfortunately, interoperability testing did not know such formalization. Another

weakness of interoperability testing is the lack of tools and precise algorithms as discussed bellow.

Interoperability Definitions. While a precise definition of interoperability is somewhat elusive, functionally the meaning is clear: components communicating with one another correctly and providing the expected services. Some definitions of interoperability are: The ability of two or more systems to exchange information and mutually use the information that has been exchanged [4]. The ability of a distributed system to interchange PDUs via the communicating platform [5]. Thus, the activity of interoperability testing consists in checking correct operations of a system of components. According to [21], this activity should check only external messages that lead to inter-component communication. This is not the case of [6] in which this activity should check an interoperability relation. The authors gave nine interoperability relations depending on the architecture used.

Interoperability Architectures. The degree of observability and controllability, and the configuration of the communicating systems have a great influence on the interoperability architecture to adopt. The Astride architecture of [7] uses upper testers and a test coordination procedure. The ATM Forum [8] proposed a test configuration connecting testers to the different Implementations Under Test (IUTs), and places Points of observation (PO) and Points of Control and Observation (PCO) between IUTs and between testers and IUTs. [9] proposed a generic architecture designed as toolbox whose components can be configured as needed.

Interoperability Testing. The recent research works on interoperability testing are related to systematic test suite generation. [7] was one of the first papers in the field. The approach adopted is based on reachability analysis. Kang, Kim et al. have proposed a whole of works relating to interoperability testing [13–21]. [13, 14, 17] proposed a test generation technique for symmetric protocols based on system composition. [21] deal with derivation of interoperability tests for the control and the data part of protocols. A skeleton test suite for control part is derived. Each test case is then parameterized. The test suite is completed by assigning values to parameters. The approach adopted by [22] is so called *one against N* in which a single entity interoperates with the rest of the integrated communicating systems. The approach is then reported on the VoIP system. In order to avoid the state space explosion, [3] proposed a test purpose oriented method. The principle was the definition of synchronization rules for parallel composition. Tretmans et al. [23] showed that, under some assumptions, the **ioco**-test theory for conformance is suitable for compositional systems.

Our Approach. This paper proposes a framework for testing interoperability inspired by work of [6]. Some parameterized relations, taking into account

available interfaces, are then introduced. Based on this framework, we propose an approach and associated algorithm for checking the correctness of communicating systems. The approach is based on a lazy composition in order to avoid the state space explosion. The originality of our approach is the possibility to interface some conformance test tools for generating interoperability tests.

The remainder of the paper is organized as follows: Section 2 presents the formal model and notations used. Section 3 introduces and explains some parameterized interoperability relations. Based on the results of section 3, section 4 shows first, how to generate interoperability tests for this framework and then how our approach can be used to generate interoperability tests from some conformance test tools. Conclusion is in section 5, and some equivalence proofs are presented in section 6.

2 Modeling with IOLTS

The behavior of communicating components can be described by means of formal models such as Input Output Labeled Transition System (IOLTS). As usual in the testing theory, we need to model implementations, even if their behaviors are unknown. An IUT will also be represented by an IOLTS. Section 2.1 introduces the IOLTS model and some notions used in the rest of the paper. Section 2.2 defines the IOLTS composition.

2.1 IOLTS Model

Let S be a system composed of a set of communicating components (M_i). Every component M_i defines a set of actions (events) A^{M_i} performed by M_i, and a set of interactions points (ports or interfaces) P^{M_i} through which M_i communicates with the other components and with its environment. To define formally a component M_i, we will use the model of Input Output Labelled Transition System (IOLTS) which is an adaptation of the usual LTS model. In this model, every action is associated to an interface through which the system receives or sends the event.

Definition 1. *An IOLTS is a tuple* $M = (Q^M, \Sigma^M, \rightarrow_M, q_0^M)$ *where:*

- Q^M *is a set of states and* $q_0^M \in Q^M$ *is the initial state.*
- $\Sigma^M \subseteq P^M \times A^M$ *where* P^M *is a finite set of interfaces (ports) through which* M *communicates with its environment and* A^M *is the alphabet of actions exchanged by* M *through* P^M*. It is partitioned into three sets:* $\Sigma^M = \Sigma_i^M \cup \Sigma_o^M \cup I^M$*: for* $(p,a) \in \Sigma^M$*, then* $(p,a) \in \Sigma_i^M$ *if* a *is input action,* $(p,a) \in \Sigma_o^M$ *if* a *is an output action, and* $(p,a) \in I^M$ *if* a *is an internal action of* M*.*
- $\rightarrow_M \subseteq Q^M \times \Sigma^M \times Q^M$ *is the transition relation. We note* $q \xrightarrow{\alpha}_M q'$ *for* $(q,\alpha,q') \in \rightarrow_M$ *and* $q \xrightarrow{\alpha}_M$ *for* $\exists q' : q \xrightarrow{\alpha}_M q'$*.*

Next, $\mathcal{IOLTS}$ *will denote the set of input output labeled transition systems.*

Note 1. $p?\alpha$ stands for $(p,\alpha) \in \Sigma_i^M$ and $p!\alpha$ for $(p,\alpha) \in \Sigma_o^M$. $\bar{\mu} = p!\alpha$ if $\mu = p?\alpha$ and $\bar{\mu} = p?\alpha$ if $\mu = p!\alpha$.

Example 1. Specification M_1 of the Fig. 1 communicates with specification M_2 through interfaces $C1$ and $C2$ and with its environment through interfaces $C3$ and $C4$. Specification M_2 communicates with its environment through interface $C5$. For M_1: $P^{M_1} = \{C1, C2, C3, C4\}$, $A^{M_1} = \{!start, ?ack, !init, ?status\}$, $\Sigma_o^{M_1} = \{C1!start, C4!init\}$, and $\Sigma_i^{M_1} = \{C2?ack, C3?status\}$.

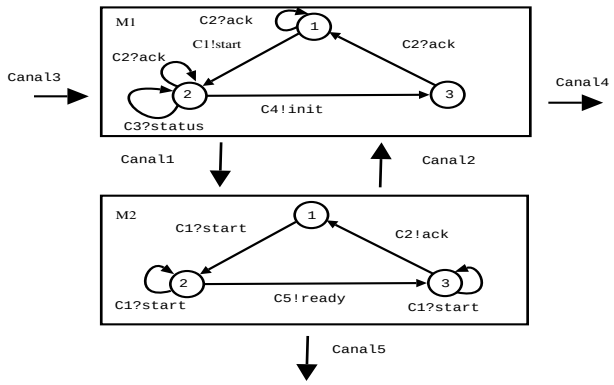

Fig. 1. Communicating specifications.

We use the following standard notation of IOLTS.

Notation 1 *Let* $M \in \mathcal{IOLTS}$, $\mu_{(i)} \in \Sigma^M$, $\alpha_{(i)} \in \Sigma^M \setminus I^M$, $\tau_{(i)} \in I^M$, $\sigma \in (\Sigma^M \setminus I^M)^*$, $S \subseteq \Sigma^M$, $P \subseteq Q^M$, *and* $q, q', q_i \in Q^M$.

- $q \stackrel{\mu_1 \ldots \mu_n}{\longrightarrow}_M q' \quad =_\Delta \quad \exists\, q_0 = q, q_1 ..., q_n = q', \forall i \in [1,n], q_{i-1} \stackrel{\mu_i}{\rightarrow}_M q_i.$
- $out(q) \quad =_\Delta \quad \{\alpha \in \Sigma_o^M \mid \exists\, q' \text{ and } q \stackrel{\alpha}{\rightarrow}_M q'\}.$
- $out(P) \quad =_\Delta \quad \{out(q) \mid q \in P\}.$

The visible behavior of M *is described with* $\Rightarrow$

- $q \stackrel{\epsilon}{\Rightarrow}_M q' \quad =_\Delta \quad q = q' \text{ or } q \stackrel{\tau \ldots \tau}{\longrightarrow}_M q'.$
- $q \stackrel{\alpha}{\Rightarrow}_M q' \quad =_\Delta \quad \exists\, q_1, q_2, q \stackrel{\epsilon}{\Rightarrow}_M q_1 \stackrel{\alpha}{\rightarrow}_M q_2 \stackrel{\epsilon}{\Rightarrow}_M q'.$
- $q \stackrel{\sigma}{\Rightarrow}_M q' \quad =_\Delta \quad \exists\, q_0 = q, q_1 \ldots, q_n = q', \forall i \in [1,n], q_{i-1} \stackrel{\mu_i}{\Rightarrow}_M q_i,$ $\sigma = \mu_1 \cdots \mu_n.$
- $q \stackrel{\sigma}{\Rightarrow}_M \quad =_\Delta \quad \exists\, q', q \stackrel{\sigma}{\Rightarrow}_M q'.$
- $q \; after \; \sigma \quad =_\Delta \quad \{q' \in Q^M \mid q \stackrel{\sigma}{\Rightarrow}_M q'\}.\; M \; after \; \sigma \; =_\Delta \; q_0^M \; after \; \sigma.$
- $Out(M, \sigma) \quad =_\Delta \quad out(M \; after \; \sigma).\; Out_S(M, \sigma) \; =_\Delta \; Out(M, \sigma) \cap S.$
- $Traces(q) \quad =_\Delta \quad \{\sigma \in (\Sigma^M \setminus I^M)^* \mid q \; after \; \sigma \neq \emptyset\}.\; Traces(M) \; =_\Delta \; Traces(q_0^M).$

Note 2. For $M \in \mathcal{IOLTS}$, $X \subseteq P^M$, $S \subseteq \Sigma^M$, $\mu \in \Sigma^M$, and $(\sigma, \sigma') \in Traces(M)$:

- $\sigma.\sigma'$ will denote the concatenation of the two traces.
- $\sigma_{/S}$ will denote the projection of σ on S defined by: $\epsilon_{/S} = \epsilon, (\alpha.\sigma)_{/S} = \sigma_{/S}\ if\ \alpha \notin S$, and $(\alpha.\sigma)_{/S} = \alpha.(\sigma_{/S})\ if\ \alpha \in S$.
- $\Sigma^{M/X} = \{(p,a) \in \Sigma^M | p \in X\}$ will denote the projection of Σ^M on the interface set X.
- $P_M(\sigma, X) = \{\sigma' \in Traces(M) \mid \sigma_{/\Sigma^{M/X}} = \sigma'_{/\Sigma^{M/X}}\ \}$ will denote the set of traces of M that correspond with σ on X.
- $card(S)$ will denote the number of elements of S.

An IOLTS M is input complete if it is completely specified for input actions.

Definition 2. *Let M be an IOLTS.*

- *M is input complete if $\forall q \in Q^M$, $\forall a \in \Sigma_i^M$, $q \stackrel{a}{\Rightarrow}_M$.*
- *M is linear if $\forall q \in Q^M$, $card(out(q)) \leq 1$. In this case, the projection of M on $S \subseteq \Sigma^M$ is the linear IOLTS noted by $M_{/S}$ restricted to S (thus $Traces(M_{/S}) = \{\sigma_{/S} \mid \sigma \in Traces(M)\}$).*

2.2 Synchronous Composition of IOLTS

Communication mechanism of two IOLTS can be modeled by an IOLTS.

Definition 3. *Let M_1 and M_2 be two IOLTS such that $\Sigma_i^{M_1} \cap \Sigma_i^{M_2} = \Sigma_o^{M_1} \cap \Sigma_o^{M_2} = \emptyset$. The synchronous composition of M_1 and M_2 is the IOLTS $M = (Q^M, \Sigma^M, \rightarrow_M, q_0^M)$ noted $M_1 \| M_2$, defined by:*

- *Ports of M is the union of ports of M_1 and M_2: $P^M = P^{M_1} \cup P^{M_2}$.*
- $\Sigma_o^M = \Sigma_o^{M_1} \cap \Sigma_o^{M_2}$.
- $\Sigma_i^M = (\Sigma_i^{M_1} \backslash \Sigma_o^{M_2}) \cup (\Sigma_i^{M_2} \backslash \Sigma_o^{M_1})$.
- $I^M = I^{M_1} \cup I^{M_2}$.
- $q_0^M = (q_0^{M_1}, q_0^{M_2})$.
- $Q^M = \{q_1 \| q_2 \mid q_1 \in Q^{M_1}, q_2 \in Q^{M_2}\}$.
- *$\rightarrow_M$ is the smallest relation defined by the following rules ($\mu \in \Sigma^M$):*
  1. $q_1 \stackrel{\mu}{\rightarrow}_{M_1} q_1'$, $\mu \notin \Sigma^{M_2} \backslash I^{M_2}$ $\Rightarrow$ $q_1 \| q_2 \stackrel{\mu}{\rightarrow}_M q_1' \| q_2$
  2. $q_2 \stackrel{\mu}{\rightarrow}_{M_2} q_2'$, $\mu \notin \Sigma^{M_1} \backslash I^{M_1}$ $\Rightarrow$ $q_1 \| q_2 \stackrel{\mu}{\rightarrow}_M q_1 \| q_2'$
  3. $q_1 \stackrel{\mu}{\rightarrow}_{M_1} q_1'$, $q_2 \stackrel{\mu}{\rightarrow}_{M_2} q_2'$, $\mu \notin I^M$ $\Rightarrow$ $q_1 \| q_2 \stackrel{\mu}{\rightarrow}_M q_1' \| q_2'$

In interoperability testing, we usually need to observe only some specific events among all possible events. That is, we define M **visible** X. The observable events of M **visible** X are those exchanged over X. The other events are considered internal actions τ.

Definition 4. *Let M be an IOLTS and $X \subseteq P^M$ a set of interfaces. M* **visible** *X is IOLTS defined by:*

- $Q^{M \textbf{ visible } X} = Q^M$, *and* $q_0^{M \textbf{ visible } X} = q_0^M$.
- $\Sigma^{M \textbf{ visible } X} = \Sigma^{M/X} \cup \{\tau\}$.
- *$\rightarrow_{M \textbf{ visible } X}$ is the smallest relations defined by ($\mu \in \Sigma^M$):*
  1. $q \stackrel{\mu}{\rightarrow}_M q'$, $\mu \notin \Sigma^{M/X}$ $\Rightarrow$ $q \stackrel{\tau}{\rightarrow}_{M \textbf{ visible } X} q'$
  2. $q \stackrel{\mu}{\rightarrow}_M q'$, $\mu \in \Sigma^{M/X}$ $\Rightarrow$ $q \stackrel{\mu}{\rightarrow}_{M \textbf{ visible } X} q'$

Note 3. Let M, M_1, and M_2 be three IOLTS. When $X = P^M$ then M **visible** $X = M$. $\mathbf{M_1 \|_X M_2}$ **will denote** $\mathbf{M_1 \| M_2}$ **visible** $\mathbf{X}$.

3 A Parameterized Writing of the Interoperability Relation

A system of communicating components defined a set of available interfaces. Events exchanged over this set are observable and then testable. Interoperability relations for such systems must take account these available interfaces. This section introduces and compares some parameterized interoperability relations.

3.1 Preliminaries

One of the used conformance relations is the relation **ioconf**. It states that an implementation I is conformant to its specification M if after a trace of M, outputs of I are foreseen in M.This relation allows partial specifications and allows implementations to add the underspecification.

Definition 5. *Let M and I be two IOLTS.*
I **ioconf** $M =_{\Delta} \forall \sigma \in Traces(M) \Rightarrow Out(I,\sigma) \subseteq Out(M,\sigma)$.

To compare binary relations, we use $\sqsubseteq$ and $\cong$. Let $\mathcal{R}_x$ and $\mathcal{R}_y$ be two binary relations over $\mathcal{IOLTS}$. $\mathcal{R}_y \sqsubseteq \mathcal{R}_x$ if for every $(I,M) \in \mathcal{IOLTS}$ such that $I\ \mathcal{R}_x\ M$ then $I\ \mathcal{R}_y\ M$. $\mathcal{R}_x \cong \mathcal{R}_y$ if $\mathcal{R}_y \sqsubseteq \mathcal{R}_x$ and $\mathcal{R}_x \sqsubseteq \mathcal{R}_y$.

3.2 A Parameterized Writing.

The starting point for interoperability testing is some given specifications, implementations for these specifications defining some available interfaces, and a criterion that implementations should satisfy. Thus, Two implementations interoperate if they check the interoperability criterion. The framework presented here, is based on the comparison between the outputs of the system to be tested and the outputs of one specification, modulo some interface projection. Let $I_{(i)}$ be an implementation of the specification $M_{(i)}$ with $i \in \{1,2\}$, and $X \subset P^{M_1} \cup P^{M_2}$ be an interface set. We recall that $\mathbf{M_1 \| _X M_2}$ is $\mathbf{M_1 \| M_2}$ **visible** $\mathbf{X}$.

Definition 6. $\mathbf{interop_X}(I_1,I_2) \stackrel{\triangle}{=} \forall \sigma \in trace(M_1 \|_X M_2), \Rightarrow$
$Out_{\Sigma^{I_1}}(I_1 \|_X I_2, \sigma) \subset \cup_{\sigma' \in P_{M_1 \| M_2}(\sigma,X)} Out_{\Sigma^{M_1/X}}(M_1, \sigma'_{/\Sigma^{M_1}})$

The first relation defined is the relation $\mathbf{interop_X}$. It states that during the interaction between I_1 and I_2, when a trace σ of $M_1 \|_X M_2$ is injected to the system to be tested, the observable behaviors of the system projected on the available interfaces of I_1 is foreseen in the union of observable behaviors (projected on the available interfaces) of M_1, while applying the projection, on the alphabet of M_1, of each trace $\sigma' \in Traces(M_1 \| M_2)$ (σ' coincides with σ on the interface set X).
Let $Y = P^{M_1} \cap P^{M_2}$ be the commun interfaces of M_1 and M_2.

Definition 7. $\mathbf{intercom_X}(I_1,I_2) \stackrel{\triangle}{=} \forall \sigma \in trace(M_1 \|_X M_2), \forall \sigma' \in P_{M_1 \| M_2}(\sigma,X),$
$\sigma'_{/Y} \neq \epsilon \Rightarrow Out_{\Sigma^{I_1}}(I_1 \|_X I_2, \sigma) \subset \cup_{\sigma' \in P_{M_1 \| M_2}(\sigma,X)} Out_{\Sigma^{M_1/X}}(M_1, \sigma'_{/\Sigma^{M_1}})$

$\mathbf{intercom_X}$ is similar to $\mathbf{interop_X}$, but only invokes inter-component communications. Generally, components are tested to conformance, and thus, $\mathbf{intercom_X}$ avoids the conformance testing, but only tests the inter-communication part.

Another criterion is to define a set $A \subseteq \Sigma^{M_1} \cup \Sigma^{M_2}$ to be covered. $\mathbf{intercover_X}$ covers all transitions which pass by a given event of A. $\mathbf{intercover'_X}$ covers some transitions which pass by a given event of A. The traces that these relations consider can be smaller comparing to $\mathbf{intercom_X}$ and $\mathbf{interop_X}$.

Definition 8. $\mathbf{intercover_X}(I_1, I_2) \stackrel{\triangle}{=} \forall a \in A,\ \forall \sigma \in trace(M_1 \|_X M_2),\ \forall \sigma' \in P_{M_1\|M_2}(\sigma, X),\ \sigma'_{/a} \neq \epsilon \Rightarrow$

$$Out_{\Sigma^{I_1}}(I_1 \|_X I_2, \sigma) \subset \cup_{\sigma' \in P_{M_1\|M_2}(\sigma,X)} Out_{\Sigma^{M_1/X}}(M_1, \sigma'_{/\Sigma^{M_1}})$$

Definition 9. $\mathbf{intercover'_X}(I_1, I_2) \stackrel{\triangle}{=} \forall a \in A,\ \exists \sigma \in trace(M_1 \|_X M_2),\ \forall \sigma' \in P_{M_1\|M_2}(\sigma, X),\ \sigma'_{/a} \neq \epsilon \Rightarrow$

$$Out_{\Sigma^{I_1}}(I_1 \|_X I_2, \sigma) \subset \cup_{\sigma' \in P_{M_1\|M_2}(\sigma,X)} Out_{\Sigma_{M_1/X}}(M_1, \sigma'_{/\Sigma^{M_1}})$$

The four relations defined here are suitable for any test architecture. The choice of a relation depends on the confidence that we have in an implementation and also on the size of components to be invoked.
As result, the next theorem shows the equivalence between $\mathbf{interop_X}$ and **ioconf**:

Theorem 2. *Let M_1, M_2, I_1 and I_2 be four IOLTS. We assume that M_1, M_2, I_1 and I_2 are input complete, $\Sigma_o^{M_1} \cap \Sigma_o^{M_2} = \emptyset$, $\Sigma_o^{M_1} \cap \Sigma_o^{I_2} = \emptyset$, $\Sigma_o^{I_1} \cap \Sigma_o^{M_2} = \emptyset$, and $\Sigma_o^{I_1} \cap \Sigma_o^{I_2} = \emptyset$. Then:*

$$\mathbf{interop_X}(I_1, I_2) \wedge \mathbf{interop_X}(I_2, I_1) \ \cong\ I_1 \|_X I_2 \ \mathbf{ioconf}\ M_1 \|_X M_2$$

An interesting case is when the whole interfaces are available: $X = P^{M_1} \cup P^{M_2}$. In this case, for $\sigma \in M_1 \| M_2$, we have $P_{M_1\|M_2}(\sigma, P^{M_1} \cup P^{M_2}) = \{\sigma\}$. Therefore, the relations $\mathbf{interop_X}$ and $\mathbf{intercom_X}$ can be written as follows (the subscript $_X$ will be omitted):

$\mathbf{interop}(I_1, I_2) \stackrel{\triangle}{=} \forall \sigma \in trace(M_1 \| M_2) \Rightarrow Out_{\Sigma^{I_1}}(I_1 \| I_2, \sigma) \subset Out(M_1, \sigma_{/\Sigma^{M_1}})$.

$\mathbf{intercom}(I_1, I_2) \stackrel{\triangle}{=} \forall \sigma \in trace(M_1 \| M_2)\ \sigma_{/Y} \neq \epsilon \Rightarrow Out_{\Sigma^{I_1}}(I_1 \| I_2, \sigma) \subset Out(M_1, \sigma_{/\Sigma^{M_1}})$.

The paper [6] defined nine interoperability relations based on a layer architecture for asynchronous environment. These relations are decomposed into three classes depending on which interfaces of the implementations they focus on. These relations can be obtained from $\mathbf{interop_X}$ by defining a suitable interface set X. For example, the relation **interop** corresponds to Unilateral Total Interoperability Relation defined in [6].
To conclude, comparison between relations is given bellow:

$\mathbf{interop_X}(I_1, I_2) \wedge \mathbf{interop_X}(I_2, I_1) \cong I_1 \|_X I_2$ **ioconf** $M_1 \|_X M_2$
$\mathbf{intercom_X}(I_1, I_2) \sqsubseteq \mathbf{interop_X}(I_1, I_2)$.
$\mathbf{intercover'_X}(I_1, I_2) \sqsubseteq \mathbf{intercover_X}(I_1, I_2) \sqsubseteq \mathbf{interop_X}(I_1, I_2)$.

4 Derivation of Interoperability Test Cases

Interoperability testing is a finite set of experiences, in which a set of test cases, usually derived from some specifications according to a given interoperability relation, is a applied by a tester to the system under test (SUT). Depending on the results of the execution of a test case, it can be concluded whether or not the implementations interoperate. In the preceding section, we have introduce a framework for interoperability testing. Relations presented there are based on the available interfaces of implementations, and on the comparison between outputs of implementations with outputs of one specification modulo some projections. In this section, we propose an approach to check the relation $\mathbf{interop_X}$. Section 4.1 introduces some definitions used in section 4.2 which presents our approach. Section 4.3 shows how to interface some conformance generation tools to generate interoperability tests.

4.1 Preliminaries

Definition 10 (Completion). *Let T be a linear IOLTS, and $X' \subseteq P^T$ a subset of interfaces of T. $Comp(T, X')$ is the linear IOLTS defined by:*

- $q_0^{Comp(T,X')} = q_0^T$.
- $\Sigma^{Comp(T,X')} \subseteq P^T \times A^T$.
- $Q^{Comp(T,X')}$, $\rightarrow_{Comp(T,X')}$, *and* $\Sigma^{Comp(T,X')}$ *are obtained by the following rules:*
  1. $\mu \notin \Sigma^{T/X'}$, $q \xrightarrow{\mu}_T q' \Rightarrow q, q' \in Q^{Comp(T,X')}$, $\mu \in \Sigma^{Comp(T,X')}$, *and* $q \xrightarrow{\mu}_{Comp(T,X')} q'$.
  2. $p!a \in \Sigma^{T/X'}$, $q \xrightarrow{p!a}_T q' \xrightarrow{p?a}_T q'' \Rightarrow q, q'q,'' \in Q^{Comp(T,X')}$, $p!a, p?a \in \Sigma^{Comp(T,X')}$, *and* $q \xrightarrow{p!a}_{Comp(T,X')} q' \xrightarrow{p?a}_{Comp(T,X')} q''$
  3. $p!a \in \Sigma^{T/X'}$, $q \xrightarrow{p!a}_T q'$, $q' \not\xrightarrow{p?a}_T \Rightarrow q, q'q,'' \in Q^{Comp(T,X')}$, $p!a, p?a \in \Sigma^{Comp(T,X')}$, *and* $q \xrightarrow{p!a}_{Comp(T,X')} q'' \xrightarrow{p?a}_{Comp(T,X')} q'$.
  4. $p?a \in \Sigma^{T/X'}$, $q \not\xrightarrow{p!a}_T q'$, $q' \xrightarrow{p?a}_T q'' \Rightarrow q'q,'' q''' \in Q^{Comp(T,X')}$, $p!a, p?a \in \Sigma^{Comp(T,X')}$, *and* $q' \xrightarrow{p!a}_{Comp(T,X')} q''' \xrightarrow{p?a}_{Comp(T,X')} q''$.

Indeed, $Comp(T, X')$ is T by following each emission, over X' interfaces, by the same reception and reciprocally. In Fig. 4.1, T_1 represents a linear ILOTS. It consists on the emission of *start* through port $C1$ and the reception of *ack* through port $C2$. The complemention of T_1 according to ports $X' = \{C1, C2\}$ is the linear IOLTS T_2 ($T_2 = Comp(T_1, \{C1, C2\})$).

Definition 11 (Transition Matching). *Let T be a linear IOLTS. We assume that T is without internal actions ($I^T = \emptyset$). A transition matching is a function $TM :\rightarrow_T \mapsto \mathbb{N}$ that assigns to every transition a naturel number such that:*

- $TM((q, a, q_1)) > TM((q_2, b, q_3))$ *If* $\exists \sigma \in (\Sigma^M)^* \mid q \xrightarrow{a}_T q_1 \xRightarrow{\sigma}_T q_2 \xrightarrow{b}_T q_3$.
- $TM((q, a, q_1)) = TM((q_1, b, q_2))$ *If* $a = \overline{b}$.

A transition matching is a function that assigns to every label a value depending on the order of its appearance on the linear IOLTS. The set of the transition matching over T is noted $\mathcal{TM}(T)$. For example, in Fig. 4.1, we associate the transition matching TM to T_2 to obtain T_3: $TM((0, C1!start, 1')) = TM((1', C1?start, 1)) = 2$, and $TM((1, C2!ack, 2')) = TM((2', C2?ack, 2)) = 1$.

Definition 12. *Let T and T' be two linear IOLTS without internal actions. T' is said to be a path of T if:*

- $Q^{T'} \subseteq Q^T$, $\Sigma^{T'} \subseteq \Sigma^T$, *and* $\rightarrow_{T'} \subseteq \rightarrow_T$.
- $\forall \sigma \in Traces(T) \Rightarrow \sigma_{/\Sigma^{T'}} \in Traces(T)$.

$L(T)$ will denote the set of all paths T' of T.

Definition 13 (Restricted Transition Matching). *Let T be a linear IOLTS without internal actions, $T' \in L(T)$ be a path of T, and $TM \in \mathcal{TM}(T)$ a transition matching of T. The transition matching TM restricted to T' is the matching noted $TM[T/T'] :\rightarrow_{T'} \mapsto \mathbb{N}$ defined by: $TM[T/T'](q, a, q_1)) = TM((q, a, q_1))$.*

Definition 14 (Extended Transition Matching). *Let T be a linear IOLTS without internal actions, $T' \in L(T)$ be a path of T, and $TM \in \mathcal{TM}(T')$ a transition matching of T'. We assume that if $q \in Q^{T'}$ such that $out(q) = \emptyset$ in T' then $out(q) = \emptyset$ in T. An extended transition matching of T' to T is the function $\widetilde{TM}(T/T') :\rightarrow_T \mapsto \mathbb{N}$ defined by:*

- $\widetilde{TM}(T/T')((q, a, q_1)) = TM((q, a, q_1))$ *If* $(q, a, q_1) \in \rightarrow_{T'}$.
- $\widetilde{TM}(T/T')((q, a, q_1)) = TM((q_2, b, q_3))$ *If* $\exists \sigma \in (\Sigma^T)^*$, $\sigma_{/\Sigma^{T'}} = \epsilon$, *and* $q \xrightarrow{a}_T q_1 \stackrel{\sigma}{\Rightarrow}_T q_2 \xrightarrow{b}_T q_3$.

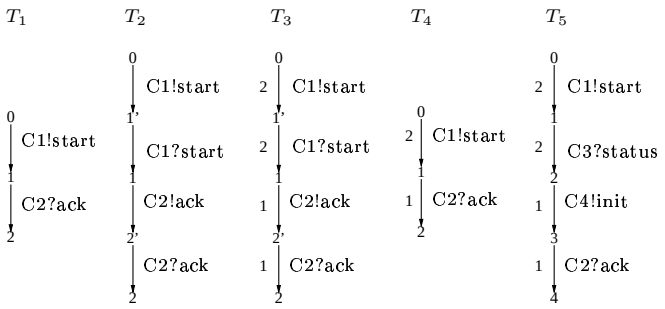

Fig. 2. Transition matching.

Example 2. In Fig; 4.1, we can remark that T_4 is a path of $T3$, and the restriected transition of TM to T_4 is defined by $TM[T_3/T_4]((0, C1!start, 1) = 2$,

$TM[T_3/T_4]((1, C2?ack, 2) = 1$. Also, T_4 is a pth of T_5, then the extended transition matching is defined by:
$\widetilde{TM}(T_5/T_4)((0, C1!start, 1)) = 2$, $\widetilde{TM}(T_5/T_4)((1, C3?status, 2)) = 1$,
$\widetilde{TM}(T_5/T_4)((2, C4!init, 3)) = 1$, and $\widetilde{TM}(T_5/T_4)((3, C2?ack, 4)) = 1$.

Definition 15 (Lazy Composition). *Let T_1 and T_2 be two linear IOLTS and $TM1 \in \mathcal{TM}(T_1)$, $TM2 \in \mathcal{TM}(T_2)$. The lazy composition of T_1 and T_2 noted $(T_1, TM1)$ LC $(T_2, TM2)$ is the IOLTS defined by:*

1. $q \xrightarrow{a}_{T_1} q'$, $q_1 \xrightarrow{b}_{T_2} q_1'$, $a \notin \Sigma^{T_2}$, $TM1((q, a, q')) \geq TM2((q_1, bq_2)) \Rightarrow (q, q_1) \xrightarrow{a}_{T_1\ LC\ T_2} (q', q_1)$.
2. $q \xrightarrow{a}_{T_1} q'$, $q_1 \xrightarrow{a}_{T_2} q_1' \Rightarrow (q, q_1) \xrightarrow{a}_{T_1\ LC\ T_2} (q', q_1)$.
3. *Symmetric rule of 1.*

The goal of the lazy composition is to avoid the construction of linear IOLTS product, but to build a product who preserves the event appearance order.

4.2 Our Approach

A Test Purpose (TP) is a property that one would like to observe/check on implementation behavior. It defines a temporal sequencing of observable actions.

Definition 16 (Test Purpose). *A test purpose TP is a deterministic and acyclic IOLTS. TP is associated with two distinguished non-empty sets of states $Accept(TP)$ and $Incon(TP)$ such as $Accept(TP) \subset Q^{TP}$ and $Incon(TP) \subset Q^{TP}$.*

To generate test case for the relation $\mathbf{interop_X}$, our approach is oriented test purpose. First, we start from σ_1 a M_1 trace, then we search the corresponding $M_1 \| M_2$ traces σ such as $\sigma_1 = \sigma_{/\Sigma^{M_1}}$. The same reasoning can be adopted for M_2 or $M_1 \| M_2$ traces.

Let us suppose, that TP is a test purpose of $M_1 \| M_2$[1] modeled by a linear IOLTS, such as $Incon(TP) = \emptyset$ and $\forall (q, a) \in (Q^{TP}, \Sigma^{TP})$, $q \stackrel{a}{\Rightarrow}_{TP}$ then $\forall q' \in Q^{TP}$, $q' \stackrel{a}{\not\Rightarrow}_{TP}$. Let $Y = P^{M_1} \cap P^{M_2}$ be the commun interfaces of M_1 and M_2. The algorithm for generating interoperability tests is described in figure 3.

First a matching transition is selected for TP. TP is then completed according to Y and devised into two test purposes: test purpose T_1 for M_1 and test purpose T_2 for M_2. M_1 is traversed to find a path T_3 that verifies T_1. The resulting path T_3 brings some M_2 events that is not in TP and by consequence is not in T_2. To add these new events to T_2, T_3 is first completed according to Y, then projected on M_2 alphabet (T_4 as result), and finally composed with T_2 according to the lazy composition defined above. This composition aim is to add new M_2 events to T_2 while preserving events order (T_5 as result). T_5 is projected on Σ^{M_2}. In every state of T_6, we prohibit synchronization from M_2 other than those reported in T_6. This is done by adding $Inconc$ states to T_6. M_2 is traversed

[1] TP contains events of M_1 or M_2 or both.

Input: Three IOLTS M_1, M_2 and TP.
Output: Test cases for $M_1 \| M_2$.

1. Selection of a transition matching TM for TP.
2. Completion of $TP \Rightarrow Comp(TP, Y)$.
3. Making projections: $T_1 = Comp(TP, Y)_{/\Sigma^{M_1}}$ and $T_2 = Comp(TP, Y)_{/\Sigma^{M_2}}$.
4. Computation of a test case verifying T_1 in $M_1 \Rightarrow T_3$.
5. Completion of $T_3 \Rightarrow Comp(T_3, Y)$.
6. Making projection: $T_4 = Comp(T_3, Y)_{/\Sigma^{M_2}}$.
7. Realization of the lazy composition: $(T_4, \widetilde{TM}(T_4/T_1))\ LC\ (T_2, TM[T_2]) \Rightarrow T_5$.
8. Making projection: $T_6 = T_{5/\Sigma^{M_2}}$.
9. Addition to T_6, in every state, synchronization events not envisaged with M_1 in T_6. These states are labeled $Incon$.
10. Computation of a test case verifying T_6 in $M_2 \Rightarrow T_7$.
11. Realization of the lazy composition: $T_8 = (T_7, \widetilde{TM}(T_7/T_2))\ LC\ (T_3, \widetilde{TM}(T_3/T_1))$.
12. Computation of T_8 **Visible** X.
13. Test Selection.

Fig. 3. Interoperability generation algorithm.

to find a path T_7 that verifies T_6. In this case, we are sure, that new events of M_1 brought by T_7 are envisaged by T_3. Then a lazy composition of T_3 and T_7 is performed and projected on X.

Method Complexity. The complexity of the algorithm depends on the size of the three entities: $O(size(TP) * (size(M_1) + size(M_2)) + size(TP) * size(TP))$.

Remark 1. In the case where the test purpose TP contains only events from one entity, for example M_1, the TP projection on M_2 is empty ie. $T_2 = \emptyset$. In this case $T_5 = T_4$ and the algorithm hold. The interest of our approach is the possibility to transform conformance test cases into interoperability test case.

4.3 Interoperability Testing using Conformance Test Tools

A number of tools for conformance testing exist, among which there are TVG [24], GenTest [26] and TorX [25]. To generate interoperability testing with these tools, for a system of communicating components S, one must compute the global specification of S by composing all its specification components. In this case, we generate conformance test of S rather then interoperability test. The inconvenient of this approach is the state space explosion. Interfacing this tools for generating interoperability tests that checks the relation $\mathbf{interop_X}$, without composing systems is showed bellow.

Oriented Test Purpose Generation Tools. Oriented test purpose generation tools take in entry an IOLTS M and a test purpose T and produce a

conformance test case. Interfacing these tools can be done through an external module. The external module interacts and controls the generating tool inputs and outputs. The algorithm given in 3 is implemented by this latter.

Figure 4 summarizes the functionalities of the external module for a given TP. Let us suppose that TP consists in checking the emission of $C1!start$ followed by the emission of $C2!ack$ for specifications of figure 1. The module selects the transition matching 1 and 2 for TP (1). TP is then completed (2). Two projections are carried out: T_1 and T_2 (3). T_1 and M_1 are injected into the generation tool who produces a test case T_3 of M_1 (T_3 brings new events $C3?statut$ and $c4!init$ (4)). This latter is completed and projected on Σ^{M_2} (5,6). At this time, the external module realizes the lazy composition according to the initial matching of TP (7). Non-envisaged events are added (thus a verdict $Inconc$ (9)), and the result is injected into the generation tool with M_2 to obtain a test case of M_2 (this latter brings event $C5!ready$ (10)). The last stage is a lazy construction of M_1 test case with M_2 test case (new events of T_3 and T_7 has the same priority (11)). The result is an IOLTS checking the test purpose TP.

Remark 2. To generate interoperability testing from conformance tools, we made the assumption that all components interfaces are accessible. In the case of a system having X accessible interfaces, hiding interfaces and eliminating redundant test cases can be done at the end of the test generation.

5 Discussion

In this paper, we have proposed a formal framework for the interoperability testing. We show that interoperability relations can be defined by taking account test architecture, and do not necessary need to consider the whole interfaces between components. The equivalence between the parameterized relation $\mathbf{interop_X}$ and conformance relation **ioconf** is proved. As results of our framework, we proposed an efficient approach and an associated algorithms for generating interoperability tests. Interfacing conformance generating tools is then presented. Depending in the goal of generating tools, we suggested an addition of an external module to interface these tools.

Testing in context. We was interested in this work in generating interoperability tests, but the framework, the approach and the algorithms we gave, is suitable to testing in context.

Synchronous vs Asynchronous We supposed here that the communications between entities are synchronous, but the results are the same in an asynchronous environment. Indeed, in the work [6], we studied this problem by consider an environment ξ between M_1 and M_2 and we have considered the composition $M_1 \| \xi \| M_2$. In this case, the completion in input of M_1 and M_2 is not necessary.

Future Work. Future works are oriented interoperability testing for real-time systems and interfacing conformance tools for interoperability testing.

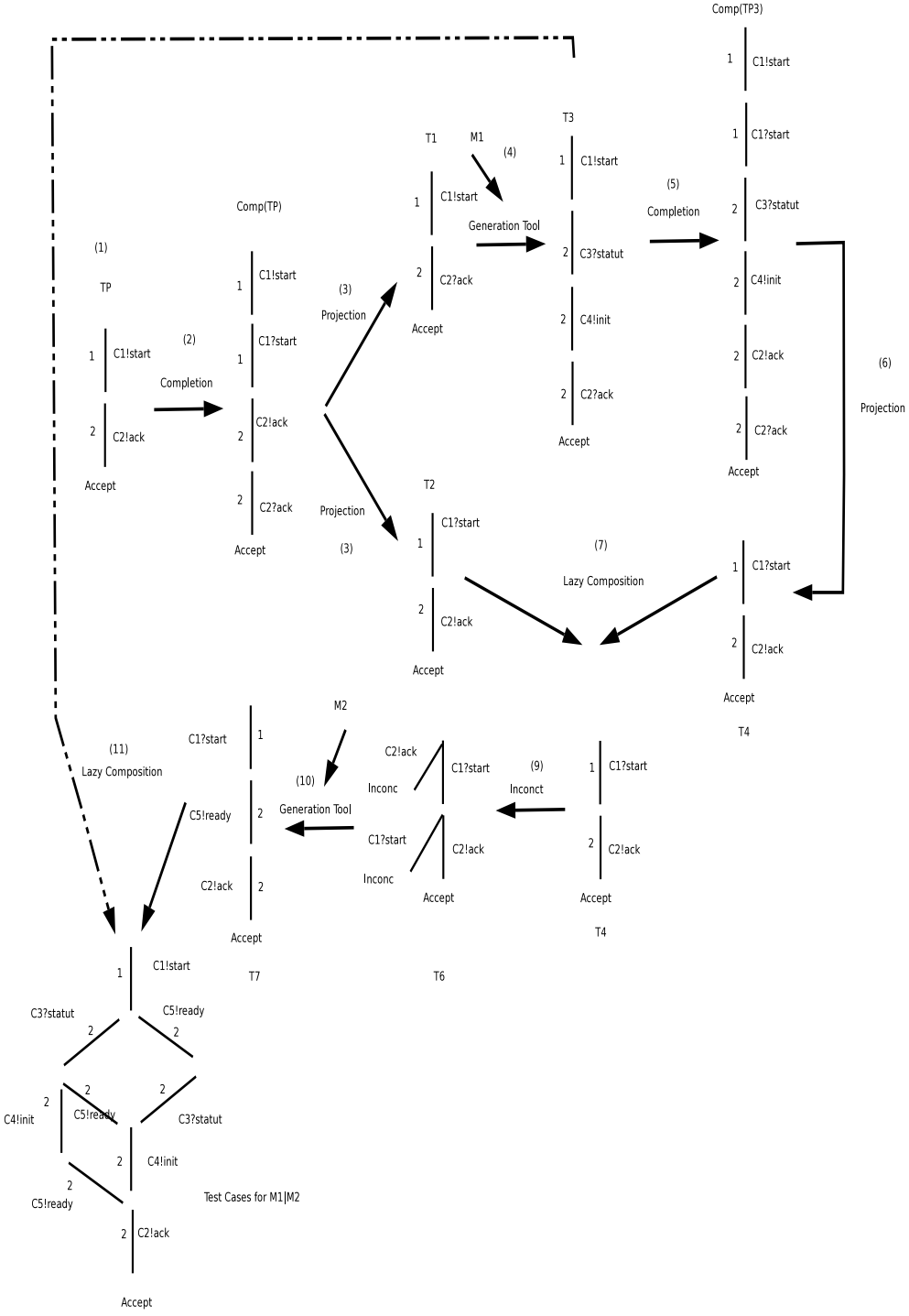

Fig. 4. An application example.

References

1. M. Philippou. Relations d'implantations et Hypothèses de test sur les automates à entréss sorties. PhD thesis, Université de Bordeaux 1, France 1994.
2. J. Tretmans. A formal approach to conformance testing. PhD thesis, University of Twente, Enschede, The Netherlands, 1992.
3. R. Castanet, O. Kone. Test generation for interworking systems. *Computer Communications*, 23 (2000), 642-652.
4. ISO/IEC JTC1 DTR-10000. Information Technology - Framework and Classification of International Standard Protocol. 1994.
5. ETSI ETR (ETSI Technical Report) 130. Methods for Testing and Specification (MTS). Interoperability and Conformance Testing/A Classification Scheme. April 1994.
6. S. Barbin, L. Tanguy and C. Viho, Towards a Formal Framework for Interoperability Testing. *Proceedings FORTE01*, Korea, August 28-31, 2001.
7. O. Rafiq and R. Castanet. From Conformance Testing to Interoperability Testing. *Proceedings of the 3rd IWPCS'90*, Oct-Nov, Washington DC, USA, 1990.
8. W. Buehler. Introduction to ATM Forum Test Specifications, Version 1.0. *ATM Forum Technical Committee*, Testing SUbworking Group, af-test-0022.000.
9. T. Walter, I. Schieferdecker and J. Grabowski. Test Architecture for Distributed Systems - State of Art and Beyond. *Proceedings of 11th IWTCS'98*, Tomsk, Russia, August 31 - September 2, 1998.
10. J. Gadre, C. Rohrer, C. Summers, and S. Stmington. A COS Study of OSI Interoperability. *Computer Standards and Interfaces*, 9:217-237, 1990.
11. G. S. Vermmeer and H. Blik. Interoperability Testing: Basis for the Accepting of Communicating Systems. *Protocol Test System VI*, 1994.
12. G. Bonnes. IBM OSI Interoperability Verifications Services. *The 3rd IWPTS*, 1990.
13. S. Kang and M. Kim. Test Sequence Generation for Adaptive Interoperability Testing. *In Proceedings of Protocol Test Systems VIII*, pp. 187 - 200, 1995.
14. S. Kang, M. Kim, Interoperability Test Suite Derivation for Symmetric Communication Protocols. *Proceedings of FORTE/PSTV'97*, p.57-72, November 18-21, 1997
15. J. Shin and S. kang. Interoperability Test Suite Derivation for the ATM/B-ISDN Signaling Protocol. *Proceedings of IWTCS'98*, Tomsk, Russia, August31-September 2, 1998.
16. S. Seol, M. Kim, S. Kang. Interoperability Test Suite Derivation for the TCP Protocol. *In FORTE/PSTV'99*, 1999.
17. S. Kang, J. Shin and M. Kim. Interoperability Test Suite Derivation for Communication Protocols. *Computer Network* 32 (2000) 347 - 364.
18. J. Ryu, M. Kim, S. Kang, S. Seol, Interoperability Test Suite Generation for the TCP Data Part Using Experimental Design Techniques. *Proceedings of TESTCOM'00*, p.127-142, August 29-September 01, 2000
19. S. Seol, M. Kim and S. T. Chanson. Interoperability Test Generation for communication Protocols based on Multiple Stimuli Principle. *Proceedings of TestCom'22)*, Berlin, Germany, March 19 - 22 , 2002.
20. V. Trenkaev, M. Kim, S. Seol, Interoperability Testing Based on a Fault Model for a System of Communicating FSMs. *Proceedings of TestCom'03*, Sophia Antipolis, France, May 26-28, 2003.
21. S. Seol, M. Kim, S. Kang, J. Ryu. Fully automated interoperability test suite derivation for communication protocols. *Computer Networks* Volume 43, Pages 735 - 759, December 2003.

22. N. Griffeth, R. Hao, D. Lee, R.K. Sinha, Integrated System Interoperability Testing with Applications to VoIP. *Proceedings of FORTE XIII/PSTV XX*, p.69-84, October 10-13, 2000
23. Machiel van der Bijl, Arend Rensink, Jan Tretmans, Compositional Testing with ioco. *Proceedings of the 3rd International Workshop on Formal Approaches to Testing of Software, FATES2003*, Montreal, Canada 2003.
24. J. Fernandez, C. Jard, T. Jéron, C Viho, An Experiment in Automatic Generation of Test Suites for Protocols with Verification Technology. *Sci. Comput. Program.* 29(1-2): 123-146 (1997).
25. A. Belinfante, J. Feenstra, R.G. de Vries, J. Tretmans, N. Goga, L. Feijs, S. Mauw, L. Heerink. Formal test automation, A simple experiment. Proceedings of 12th Int. Workshop on Testing of Communicating Systems, pages 179-196, 1999.
26. J. He and K. Turner. Protocol-Inspired Hardware Testing. *Proceedings of Testing Communicating Systems XII*, pages 131-147, September 1999, London, UK.