Formalizing Interoperability Testing: Quiescence Management and Test Generation

Alexandra Desmoulin and César Viho

IRISA/Université de Rennes 1, Campus de Beaulieu,
35042 Rennes Cedex, France
{adesmoul, viho}@irisa.fr
http://www.irisa.fr/armor

Abstract. This paper gives formal definitions of the different existing interoperability notions called *interoperability criteria*. The equivalence between two of them leads to a method for interoperability test generation that avoids the state explosion problem of classical approaches.

1 Introduction

Despite a large literature on the interest of providing a formal approach for interoperability testing [1, 2], only few tentative have been proposed. Therefore, the aims of this study presented in this paper are double. First, we give formal definitions of interoperability testing called *interoperability criteria* (*iop criteria* for short in the following). The second contribution of this work is a new method to generate automatically interoperability test cases. It uses a theorem proving the equivalence between two iop criteria. It avoids the well-known state-explosion problem due to the classical construction of the specification composition. Thus, the proposed method is a real solution that provides an easy and efficient way to derive effectively interoperability test cases.

2 Interoperability Definitions

One-to-One Interoperability Testing Architecture. In this study, we consider the one-to-one interoperability context : the System Under Test (SUT) is composed of two Implementation Under Test (IUT). There are two kind of interfaces. The lower interfaces used for the interaction of the IUTs and the upper interfaces used for the communication with their environment. Depending on the access to the different interfaces, different architectures can be distinguished.

Formal Background. The well-known IOLTS (Input-Output Labeled Transition System) model will be used to model specifications and to define interoperability criteria. We note $p?m$ ($p!a$) for an input (output) of message m on the interface p. Figure 1 gives an example of two specifications using this model.

Quiescence and ioco. Three main situations lead to quiescence of a system : *deadlock* (a state after which no event is possible), *outputlock* (a state after which

F. Wang (Ed.): FORTE 2005, LNCS 3731, pp. 533–537, 2005.
© IFIP International Federation for Information Processing 2005

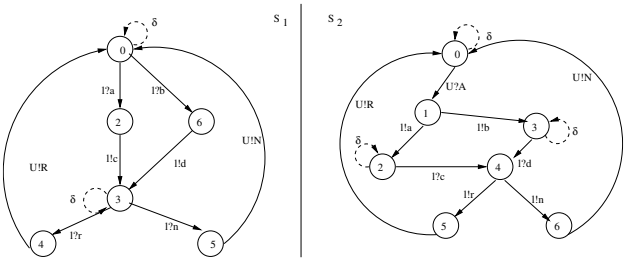

Fig. 1. Specifications S_1 and S_2

only transitions labeled with input exist) and *livelock* (a loop of internal events). Quiescence is modeled by δ and is treated as an observable output event. The obtained IOLTS is called suspensive IOLTS [3] and noted $\Delta(M)$. The **ioco** conformance relation [3] is used for the formal interoperability definitions. It says that an IUT I is **ioco**-conformant to its specification S if I can never produce an output that could not be produced by S after the same suspension trace.

Interaction. We need a model of the asynchronous interaction of the implementations. This is noted $M_1 \|_{\mathcal{A}} M_2$ and obtained as usual by a synchronous composition of $\Delta(M_1)$, $\Delta(M_2)$ and FIFO queues modeling the asynchronous environment. Quiescence is preserved and $\delta(i)$ corresponds to quiescence of M_i and δ of the two IOLTS.

Projection. In interoperability testing, we usually need to observe some specific events of an IUT. M/X represents the projection of the behavior of the implementation M reduced to a set X of expected messages.

Model of an Implementation: iop-Input Completion. In the context of interoperability testing, tester can only observe the events on the lower interfaces. But these testers can not differenciate events received by an IUT from events effectively treated. A completion is needed for inputs corresponding to the output alphabet of the other IUT specification. It is called the iop-input completion leading the IOLTS into an error deadlock state.

Formal Definition of Interoperability Criteria. According to the chosen testing architecture, different notions of interoperability can be used [4]. We will focus here on two *interoperability (iop) criteria*. The **global iop criterion** iop_G says that two implementations are considered interoperable if, after a suspensive trace of the asynchronous interaction of the specifications, all outputs and quiescence observed during the asynchronous interaction of the implementations are foreseen in the specifications. The **bilateral iop criterion** iop_B says that after a suspensive trace of S_1 observed during the asynchronous interaction of the implementations, all outputs and quiescence observed in I_1 are foreseen in S_1, and the same in the point of view of I_2 implementing the specification S_2.

The most important result is the following theorem 1 stating that iop_G is equivalent to the bilateral total iop criterion iop_B.

Theorem 1. $I_1\, iop_G\, I_2 \Leftrightarrow I_1\, iop_B\, I_2$

3 Interoperability Test Generation

The goal of an interoperability test generation algorithm is to generate interoperability Test Cases (TC) that can be executable on the SUT composed of the two IUT to be tested. The inputs of such algorithms are the specifications S_1 and S_2 on which the two IUT (I_1 and I_2) are based, and a Test Purpose (TP) which is a particular property (in the shape of incomplete sequences of actions that have to be observed or sent to the SUT) to be tested.

Interoperability Verdicts. The execution of an iop test case TC on $SUT(I_1 \| _{\mathcal{A}} I_2)$ gives a verdict : $verdict(TC, SUT) \in \{PASS, FAIL, INC\}$. The interoperability verdict PASS means that no interoperability error was detected, FAIL means that the iop criterion is not verified, and INC (for Inconclusive) means that the behavior of the SUT seems valid but it is not the purpose of the test case.

The Classical Approach and the State-Space Explosion Problem. In the classical approach based on a criteria like iop_G, the test generation algorithm begins with the construction of the asynchronous interaction $S_1 \|_{\mathcal{A}} S_2$. Then $S_1 \|_{\mathcal{A}} S_2$ is composed with the TP. The consistency of TP is checked in parallel and TC is generated. Yet, the construction of $S_1 \|_{\mathcal{A}} S_2$ can cause the well-known state-space explosion, as building $S_1 \|_{\mathcal{A}} S_2$ is exponential in the number of states of S_1 and S_2 and the FIFO queues size. Thus, interoperability test generation based on the global iop criterion may be impossible even for small specifications.

A New Method Based on the Bilateral iop Criterion iop_B. The equivalence of iop_B and iop_G (cf. therorem 1) suggests to study a method for iop test cases generation based on the bilateral iop criterion iop_B. The idea is to derive TP_{S_i} from an iop test purpose TP. Each TP_{S_i} represents TP in the point of view of S_i. This step is described in the following algorithm (see figure 2). The second step is to use a conformance test generation tool $\mathcal{F}$ such that $\mathcal{F} : (S_1, TP_{S_1}) \rightarrow TC_1$ and $\mathcal{F} : (S_2, TP_{S_2}) \rightarrow TC_2$. We obtain two unilateral iop test cases TC_1 and TC_2. The obtained test cases obtained are modified in order to take into account the differences between upper and lower interfaces in interoperability testing. For example, an event $l!m$ (resp. $l?m$) in the obtained test case will be replaced by $?(l?m)$ (resp. $?(l!m)$) in the interoperability test case. This means that the unilateral interoperability tester observes that a message m is received from (resp. sent to) the other IUT on the lower interface l. No changes are made on the test cases for events on the upper interfaces. According to the theorem 1, $verdict(TC, I_1 \|_{\mathcal{A}} I_2) = verdict(TC_1, I_1 \|_{\mathcal{A}} I_2) \wedge verdict(TC_2, I_1 \|_{\mathcal{A}} I_2)$. The rules for the combination of these two verdicts to obtain the final iop_B verdict are given by : $PASS \wedge PASS = PASS$, $PASS \wedge INC = INC$, $INC \wedge INC = INC$, and $FAIL \wedge (FAIL \vee INC \vee PASS) = FAIL$.

Applying the Method to an Example. Let us consider the two specifications S_1 and S_2 of figure 1 and the interoperability testing purpose $TP = l1?a.U2!N$. This test purpose is interesting because it contains events on both interfaces

Input: TP: test purpose; **Output:** $\{TP_{S_i}\}_{i=1,2}$;
Invariant: $S_k = S_{3-i}$ (* S_k is the other specification *); $TP = \mu_1 ... \mu_n$
Initialization: $\mu_0 = \epsilon$; $TP_{S_i} = \epsilon$;
for (i=0;$i \leq n$;i++) **do**
  if ($\mu_i \in \Sigma^{S_i}$) **then** $TP_{S_i} = TP_{S_i}.\mu_i$ (* just add *)
  if ($\mu_i \in \Sigma_L^{S_k}$) **then** $TP_{S_i} = TP_{S_i}.\bar{\mu_i}$ (* just add the mirror *)
  if ($\mu_i \in \Sigma_U^{S_k} \cup \{\tau\}$)
    $\sigma_1 := TP_{S_i}$; a_j =last_event(σ_1)
    while $a_j \in \Sigma_U^{S_k} \cup \{\tau\}$ **do** σ_1=remove_last_event(σ_1)
      a_{j-1} =last_event(σ_1) (* a_{j-1} is the last event added to TP_{S_i} and
    end a mirror event $\bar{a}_{j-1}$ may exist in S_k *)
    $M_{S_k} = \{q \in Q^{S_k}$ such that q $\stackrel{\bar{a}_{j-1}}{\rightarrow}$ and $\sigma = \bar{a}_{j-1}.\omega.\mu_i \in Traces(q)\}$
    if ($\forall q \in M_{S_k}$, $q \stackrel{\sigma}{\not\longrightarrow}$) **then** error(TP not valid : no path to μ_i)
    while (e=last_event(ω) $\notin \Sigma_L^{S_k} \cup \{\epsilon\}$) **do** ω=remove_last_event(ω) **end**
    if (e $\in \Sigma_L^{S_k}$) **then** $TP_{S_i} = TP_{S_i}.\bar{e}$
  else error(TP not valid : $\mu_i \notin \Sigma^{S_1} \cup \Sigma^{S_2}$)

Fig. 2. Algorithm to derive TP_{S_i} from TP

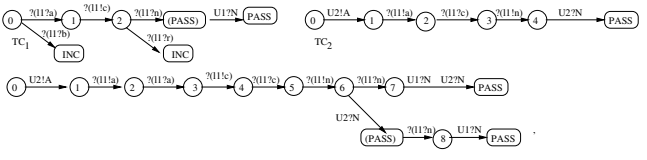

Fig. 3. Interoperability test cases obtained for $TP = l1?a.U2!N$

and both IUTs. Applying the algorithm of figure 2, we obtain : TP_{S_1}=$l1!a.l1?n$ and $TP_{S_2} = \bar{\mu}_1.\mu_2 = l2!a.U2!N$. The obtained test cases TC_1 and TC_2 are given in upper side of figure 3. For interoperability test case generation based on the global relation, the obtained TC (cf. the third test case in figure 3) comes from the composition of $S_1 \| _{\mathcal{A}} S_2$ with TP. According to the theorem 1, final interoperability verdicts obtained with TC_1 and TC_2 should be the same as the verdict obtained with TC. The proof is not given here but a look at glance to TC_1 and TC_2 shows the same paths and verdicts in TC.

4 Conclusion

In this paper, *interoperability criteria* taking quiescence into account are defined, describing the conditions under which two IUT can be considered interoperable. A theorem proving that two of them are equivalent allows a new method for interoperability test generation that avoids the classical state-explosion problem. Further studies will consider a distributed approach for interoperability testing of architectures composed of more than two implementations.

References

[1] O. Rafiq and R. Castanet. From conformance testing to interoperability testing. In *Protocol Test Systems*, volume III, pages 371–385, North-Holland, 1991. IFIP, Elsevier sciences publishers B. V.

[2] S. Seol, M. Kim, S. Kang, and J. Ryu. Fully automated interoperability test suite derivation for communication protocols. *Comp. Networks*, 43(6):735–759, 2003.

[3] J. Tretmans. Testing concurrent systems: A formal approach. In J.C.M Baeten and S. Mauw, editors, *CONCUR'99 – 10^{th} Int. Conference on Concurrency Theory*, volume 1664 of *LNCS*, pages 46–65. Springer-Verlag, 1999.

[4] S. Barbin, L. Tanguy, and C. Viho. Towards a formal framework for interoperability testing. In M. Kim, B. Chin, S. Kang, and D. Lee, editors, *FORTE' 2001*, pages 53–68, Cheju Island, Korea, 2001.