Formalizing Interoperability Testing

Alexandra DESMOULIN
IRISA / Université de Rennes 1
adesmoul@irisa.fr, http://www.irisa.fr

Introduction

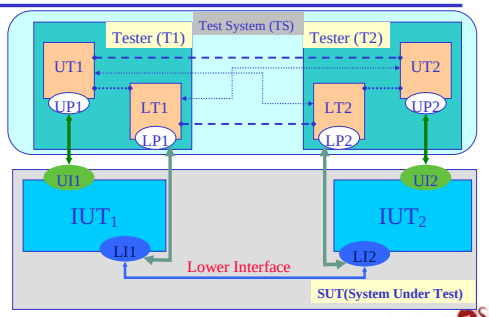
- *Interoperability test:* verification of the communication between the IUTs *and* of the provided services
- Some existing attempts to formalize interoperability but no precise characterization.
- Interoperability test generation : no method based on formal definitions
- Work presented here: one-to-one context
- Future work: N>2

Plan

- Introduction
- Interoperability testing architecture(s)
- Model for the formalization
- Interoperability definitions in the one-to-one context
- Interoperability test generation in the one-to-one context
- Conclusion and perspectives

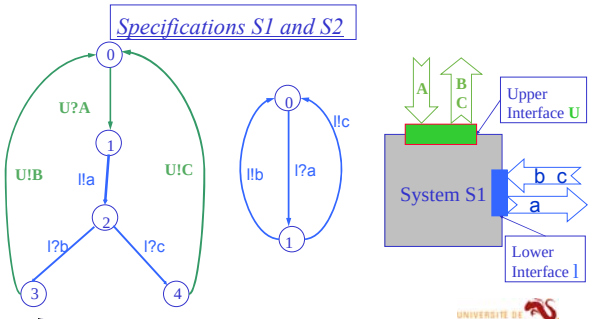
Test architectures

General interoperability testing architecture



Model for the formalization

Model of IOLTS and notations

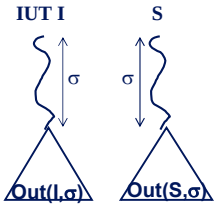


Model for the formalization

Some words about conformance testing

Conformance relation *ioconf*:
I *ioconf* S =∀σ∈Traces(S) ⇒ Out(I,σ)⊆ Out(S,σ)

Traces(S) **: succession of events of the specification S**
Out(S, σ) **: set of possible outputs (sent messages) after the succession of events σ**



Interoperability formal definitions — Interoperability testing : first definitions

- Definition of ***interoperability criteria*** based on *ioconf*. Examples:
  - ***"Global total Interoperability Criterion"*** $R_9(I_1,I_2)= \forall\sigma\in Traces(S_1||_{\mathcal{A}}S_2) \Rightarrow Out(I_1||_{\mathcal{A}}I_2,\sigma) \subseteq Out(S_1||_{\mathcal{A}}S_2,\sigma)$

    R_9 equivalent to $I_1||_{\mathcal{A}}I_2$ ioconf $S_1||_{\mathcal{A}}S_2$
  - ***"Unilateral total Interoperability Criterion"*** $R_7(I_1,I_2)= \forall\sigma_1\in Traces(S_1), \forall\sigma\in Traces(I_1||_{\mathcal{A}}I_2)$ $\sigma/\Sigma^{I1} = \sigma_1 \Rightarrow Out((I_1||_{\mathcal{A}}I_2)/\Sigma^{I1}, \sigma) \subseteq Out(S_1,\sigma_1)$.

    R_7 equivalent to I_1 ioconf S_1 *during the interaction of I_1 with I_2*
- Different criteria considering the different testing architectures.

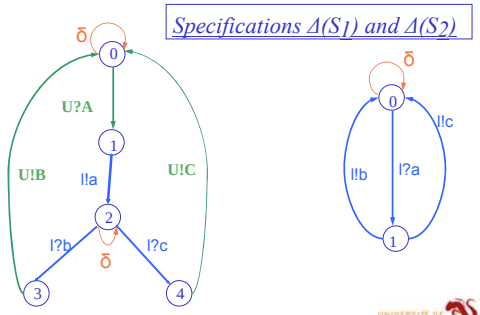
Interoperability formal definitions — Interoperability testing definitions: different problems

- Cases of non-interoperability not detected
- Incorrect verdicts during tests based on these relations due to the absence of ***quiescence management***.

⇒ ***Quiescence:*** deadlock, outputlock or livelock
  - Quiescence can be allowed in the specification(s)
  - Represented with δ
  - Considered as an ***observable output event***

Interoperability formal definitions — Specifications with quiescence modeled

Specifications $\Delta(S_1)$ and $\Delta(S_2)$



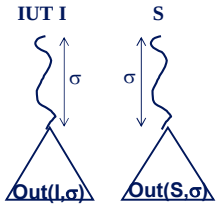
Interoperability formal definitions — Conformance testing

- Conformance relation ***ioconf***

I *ioconf* S =∀σ∈Traces(S) ⇒ Out(I,σ)⊆ Out(S,σ)

- Conformance relation ***ioco*** ***(with quiescence management)***

I *ioco* S =∀σ∈Traces(Δ(S)) ⇒ Out(Δ(I),σ)⊆ Out(Δ(S),σ)



Interoperability formal definitions — Interoperability criteria

- ***Global*** interoperability criterion:

$iop_G(I_1, I_2)= \forall\sigma \in Traces(S_1||_{\mathcal{A}}S_2) \Rightarrow Out(I_1||_{\mathcal{A}}I_2,\sigma) \subseteq Out(S_1||_{\mathcal{A}}S_2, \sigma)$

- ***Bilateral*** interoperability criterion: $iop_B(I_1, I_2)=$

$\forall\sigma_1 \in Traces(\Delta(S_1)), \sigma\in Traces(I_1||_{\mathcal{A}}I_2), \sigma/\Sigma^{I1} = \sigma_1 \Rightarrow Out((I_1||_{\mathcal{A}}I_2)/\Sigma^{I1}, \sigma) \subseteq Out(\Delta(S_1),\sigma_1)$ ***and***

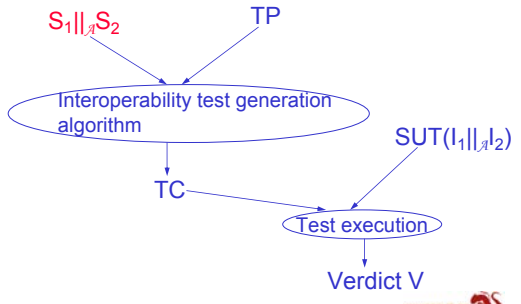
$\forall\sigma_2\in Traces(\Delta(S_2)), \sigma\in Traces(I_1||_{\mathcal{A}}I_2), \sigma/\Sigma^{I1} = \sigma_1 \Rightarrow Out((I_1||_{\mathcal{A}}I_2)/\Sigma^{I1}, \sigma) \subseteq Out(\Delta(S_1),\sigma_1)$.

Interoperability formal definitions — Quiescence Management on Interoperability : some results

- Non-interoperability due to a non-allowed output still detected
- More non-interoperability cases detected :
  - One due to non-allowed quiescence in one of the IUT
  - ***One due to incompatibility between output of one IUT and input of the other***

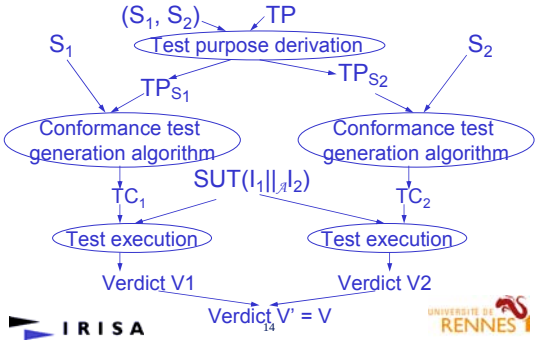
Interoperability test generation

Interoperability test generation based on iop_G (classical *method*)



Interoperability test generation

Interoperability test generation based on iop_B



Interoperability test generation

Input-output causal dependency based method

- Previous methods only verify that ***outputs (and quiescence)*** are foreseen in the specifications
- ***Problem:*** tester cannot verify if corresponding ***inputs*** are really executed (by the other IUT)
- ***Solution :*** use of ***causal dependencies*** between an input and the possible output to conclude
- Contributions of the method compared with the iop-criteria-based method:
  - Interoperability test purpose expressiveness
  - Pass and Inc verdict refinement
  - Interoperability notion more taken into account

Conclusion and perspectives

Conclusion on one-to-one interoperability context

- Formal definitions (including quiescence management)
- Interoperability test generation algorithms:
  - Based on the formal definitions
  - Input-output causal dependency based
- Future work:
  - Implementation of these algorithms and test on real protocols
  - Generalization to N>2 implementations

Conclusion and perspectives

Interoperability with N>2 IUT

- N>2 specifications (we can have $S_i=S_j$ $\forall$ (i,j) with $i\neq j$)
- Interconnexion topologies:
  - Depends on the number of IUT
  - Some topologies are equivalent
- Test architecture: depends on the topology