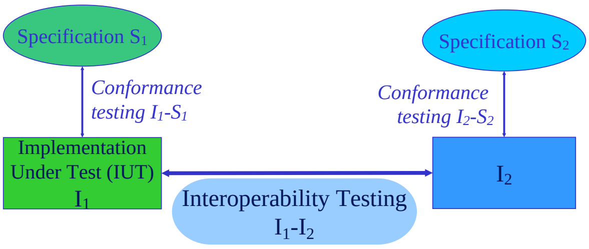
Formalizing Interoperability for Test Case Generation Purpose

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

Formalizing Interoperability for Test Case Generation Purpose: Plan

- Generality on interoperability testing and test architectures
- Interoperability formal definitions and comparison
- Interoperability test case generation :
  - Classical approach
  - Our approach
  - Comparison
- Conclusion and future work

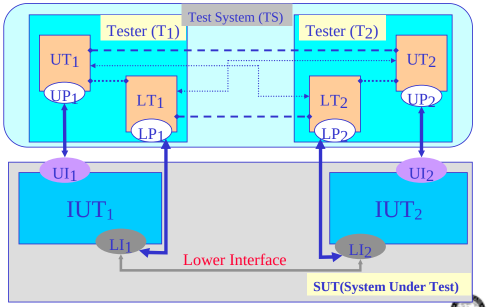
Interoperability testing in a context one-to-one


Specification S_1
Specification S_2
Conformance testing I_1-S_1
Conformance testing I_2-S_2
Implementation Under Test (IUT) I_1
I_2
Interoperability Testing I_1-I_2

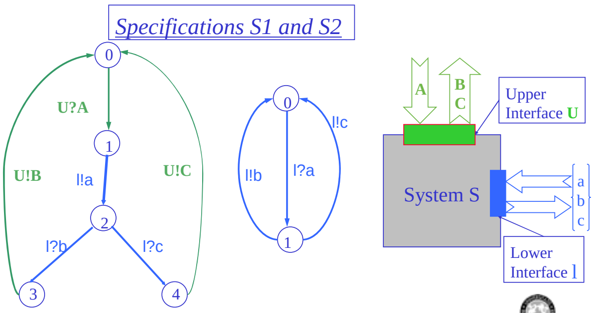
Formal definitions of interoperability testing and test generation

- Interoperability : verification of the communication between the IUTs *and* of the provided services
- Formal definitions improves conformance testing
- Some existing attempts to formalize interoperability but no precise characterization.
- Interoperability test generation : no method based on formal definitions

Interoperability testing architecture



Model of IOLTS and notations



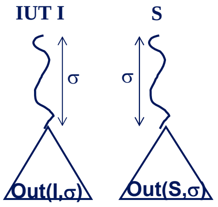
Interoperability definitions

- Based on conformance notions: Conformance relation *ioco*

I *ioco* $S = \forall\sigma\in Traces(\Delta(S))$

$\Rightarrow Out(\Delta(I),\sigma)\subseteq Out(\Delta(S),\sigma)$

- Different definitions of interoperability based on the different architectures are possible



Global interoperability criteria iop_G:

- iop_G = two implementations are considered interoperable iff, after a trace of the asynchronous interaction of the specifications (and in $I_1||_{\mathcal{A}}I_2$), all outputs and quiescence observed during the (asynchronous) interaction of the implementations must be foreseen in the asynchronous interaction of the specifications.
- Formally : $iop_G(I_1, I_2)$=

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

Bilateral interoperability criteria iop_B:

- iop_B = after a trace of S_1 observed during the asynchronous interaction of the implementations, all outputs and quiescence observed in I_1 must be foreseen in S_1, and the same in the point of view of I_2 implementing the specification S_2.
- Formally : $iop_B(I_1, I_2)$=

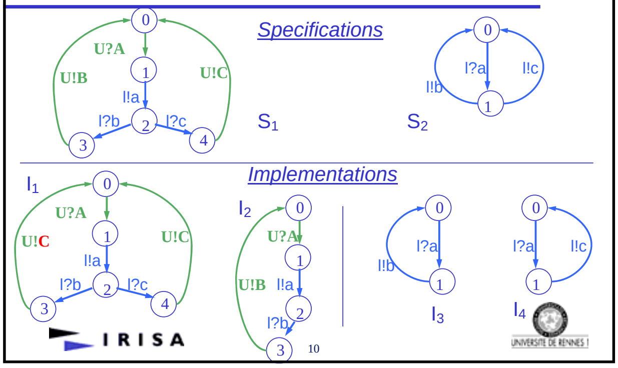
$\forall\sigma_1\in Traces(\Delta(S_1)), \sigma\in Traces(I_1||_{\mathcal{A}}I_2), \sigma/\Sigma^{I1} = \sigma_1 \Rightarrow$

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

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

$Out((I_1||_{\mathcal{A}} I_2)/\Sigma^{I1}, \sigma) \subseteq Out(\Delta(S_1),\sigma_1)$.

Example



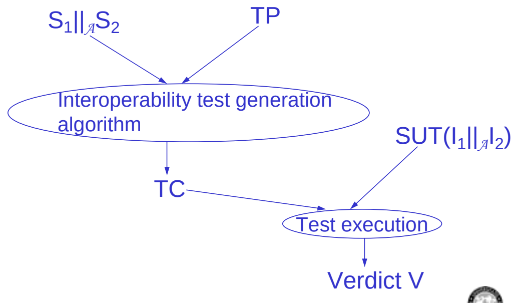
Comparison of iop_B and iop_G

- $iop_B \equiv iop_G$ (same power of non-interoperability detection)

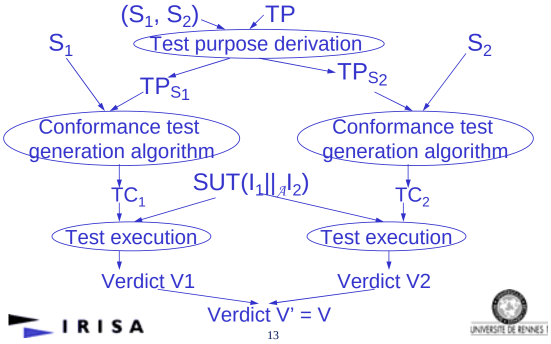
- $S_1||_{\mathcal{A}}S_2$ calculated in iop_G, not in iop_B

$\Rightarrow$ Interoperability test generation based on iop_B whithout the calculation of the interaction

Interoperability test generation based on iop_G



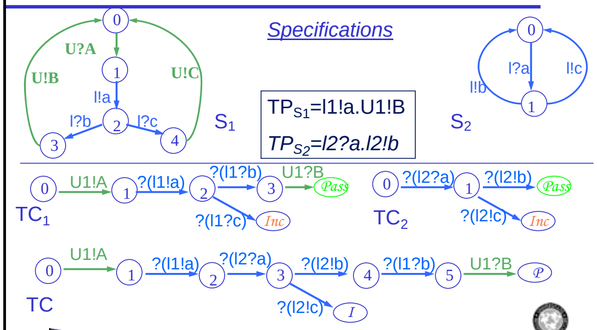
Interoperability test generation based on iop_B



Comparison of the two methods

- **Test Purpose Derivation : separating in TP events from S_1 and S_2 to obtain "unilateral" test purposes.**
- **No calculation of $S_1||_{\mathcal{A}}S_2$ with method based on $iop_B \Rightarrow$ no state-space explosion problem.**
- **Same verdicts with the two methods**
- **Test cases obtained with our method are unbiaised**

Exemple of test cases : TP=I1!a.U1!B



Conclusion and future work

- **Proof of equivalence between global interoperability criterion iop_G and bilateral interoperability criterion iop_B**
- **Proposition of a method to generate interoperability test cases from iop_B avoiding state-space explosion problem**
- **Future work :**
  - **Experimenting the method on real protocols**
  - **Studying the case of (N>2) implementations**