Quiescence Management Improves Interoperability Testing

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

Introduction

- Conformance testing : *one implementation compared to its specification*
- Interoperability testing : *2 or more interacting implementations*
- Quiescence management : *quiescence (deadlock, outputlock or livelock) can be allowed in the specification(s)*

Formal definitions of interoperability testing

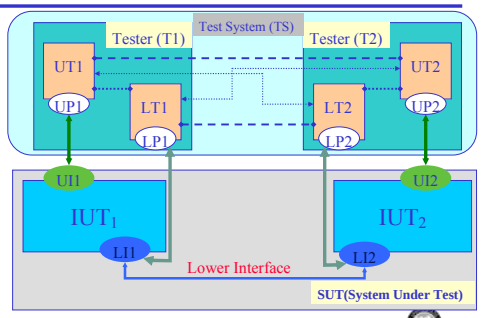
- Objective : *formal definitions of Interoperability*
- No precise characterization of interoperability at the moment.
- Quiescence management based on :

Towards a formal framework for interoperability testing, *César VIHO*, Sébastien BARBIN & Lénaïck TANGUY, FORTE 01.

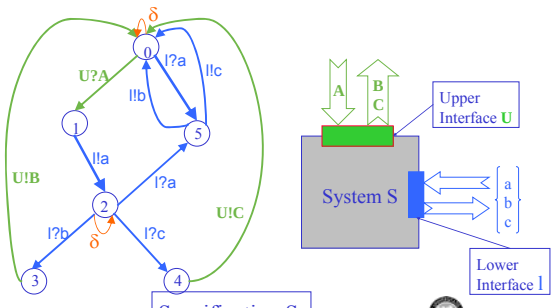
Plan

- Background
- Previous work on interoperability formalization
- Interoperability definitions with quiescence management
- Results
- Conclusion

Interoperability testing architecture



Model of IOLTS and notations



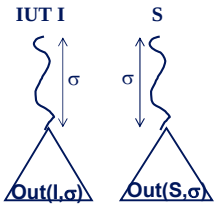
Some words about conformance testing

- Conformance relation *ioconf*

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

- Conformance relation *ioco*

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



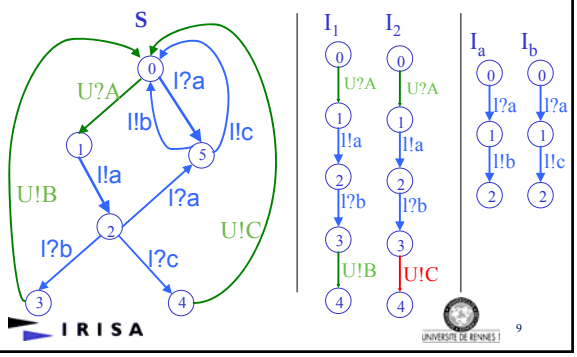
Interoperability testing without quiescence management : previous work

- Definition of *interoperability relations* based on *ioconf*. Examples:
  - $R_9(I_1,I_2)= \forall\sigma\in Traces(S_1||S_2) \Rightarrow Out(I_1||I_2,\sigma) \subseteq Out(S_1||S_2,\sigma)$

    R_9 equivalent to $I_1||I_2$ ioconf $S_1||S_2$
  - $R_7(I_1,I_2)= \forall\sigma_1\in Traces(S_1),\forall\sigma\in Traces(I_1||I_2)$ $\sigma/\Sigma^{I1}=\sigma_1 \Rightarrow Out((I_1||I_2)/\Sigma^{I1},\sigma) \subseteq Out(S_1,\sigma_1)$.

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

Interoperability testing without quiescence management : an example



Interoperability testing without quiescence management : different problems

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

Quiescence management

- Quiescence management in interoperability definitions based on conformance relation *ioco*

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

- Projection and interaction *(synchronous)* operations re-defined to keep information on quiescence.

Interoperability relations with quiescence management

- $R^{\delta}_9(I_1,I_2)= \forall\sigma\in Traces(S_1||_{\delta}S_2) \Rightarrow Out(I_1||_{\delta}I_2,\sigma) \subseteq Out(S_1||_{\delta}S_2,\sigma)$.
- $R^{\delta}_7(I_1,I_2)= \forall\sigma_1\in Traces(\Delta(S_1)),$ $\sigma\in Traces(I_1||_{\delta}I_2),\ \sigma/\Sigma^{I1}=\sigma_1 \Rightarrow$

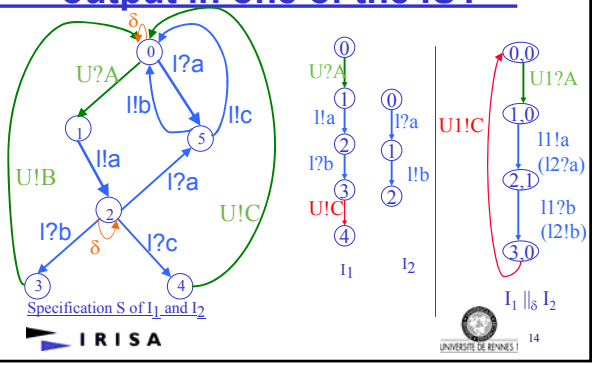
  $Out((I_1||_{\delta}I_2)/\Sigma^{I1},\sigma) \subseteq Out(\Delta(S_1),\sigma_1)$.

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

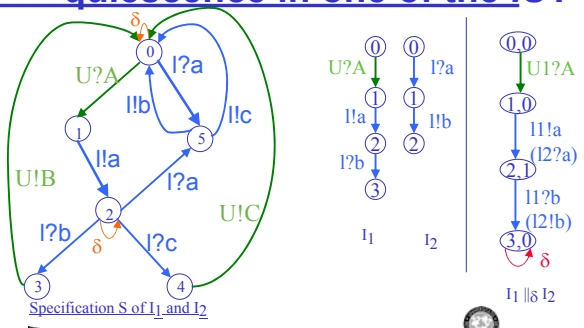
IRISA

Example 1 : non-allowed output in one of the IUT



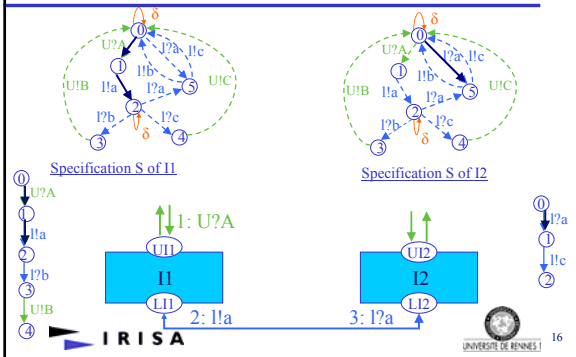
IRISA

Example 2 : non-allowed quiescence in one of the IUT



IRISA

Example 3 : incompatibility between output and input



IRISA

Conclusion

- Improvements of Interoperability testing with quiescence management:
  - More precise definitions of interoperability
  - More accurate verdicts using these definitions
    - Previous cases of non-interoperability conserved
    - More cases of non interoperability detected
- Future work:
  - Asynchronous context
  - Generalization to a context with N>2

IRISA