Institut de Recherche en Informatique de Toulouse

Université de Toulouse

Unambiguous UML Composite Structures: the OMEGA2 experience

Iulian OBER and Iulia DRAGOMIR

IRIT – University of Toulouse

France

Outline

- Overview of the OMEGA Profile
- Composite Structures
- Implementation and Evaluation
- Conclusions and Future Work

The OMEGA Language

- UML Profile for the specification and verification of real-time systems

- Consists of:

A large subset of UML

+

Model coherence constraints

+

A formal operational semantics

+

Real-time and verification extensions

The OMEGA Profile v1

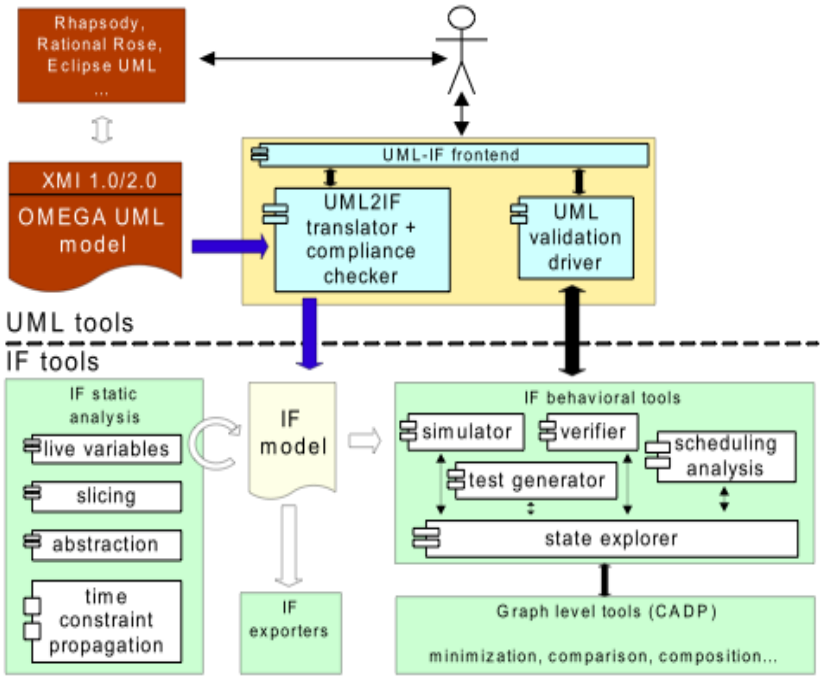
- Structure = UML Class Diagrams
  - Classes with attributes, operations and state machine
  - Relations: association, composition and generalization
- Behavior
  - State machines
  - Communication through operations and signals
- Observers = objects monitoring the system (state and events) and giving verdicts about a safety property

The IFx Toolset

- Goal: Early model validation and debugging
- Principle: Transformation to communicating extended timed automata (IF Language)

Functionalities:

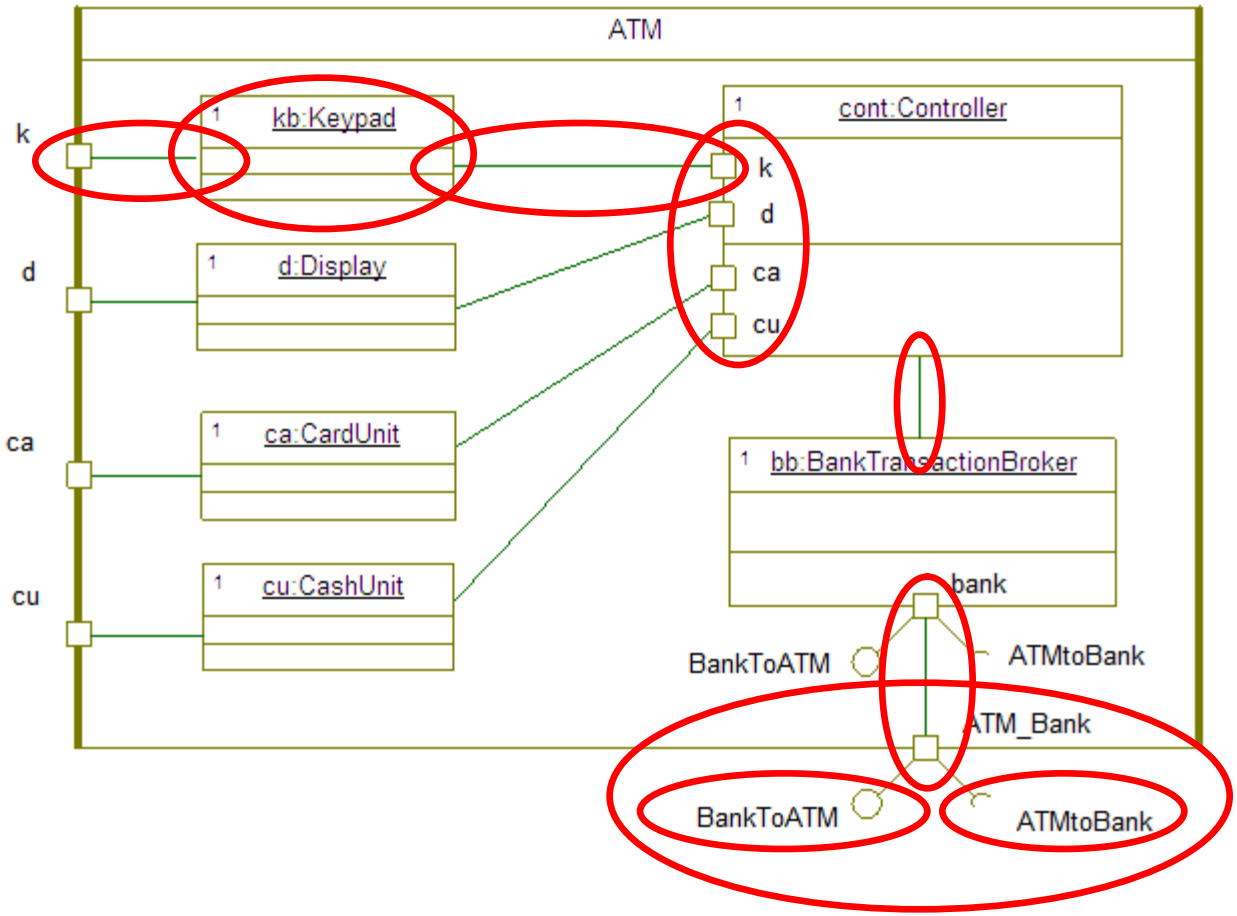
- Simulation
- Static analysis: dead code / variable elimination, slicing, …
- Model-checking: observers, state graph minimization, μ-calculus, …



Outline

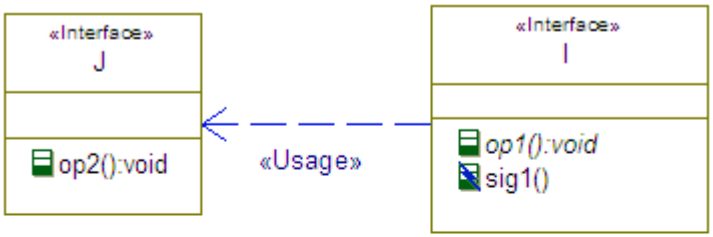
- Overview of the OMEGA Profile
- Composite Structures
- Implementation and Evaluation
- Conclusions and Future Work

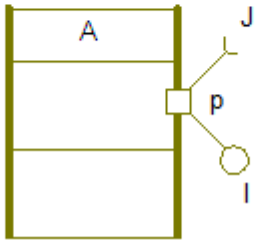
Composite Structures



1. Inner components (parts)
2. Ports
3. Delegation connector (port-instance)
4. Delegation connector (port-port)
5. Assembly connector (instance-port)
6. Assembly connector (instance-instance)
7. Provided interface
8. Required interface

Ports Directionality

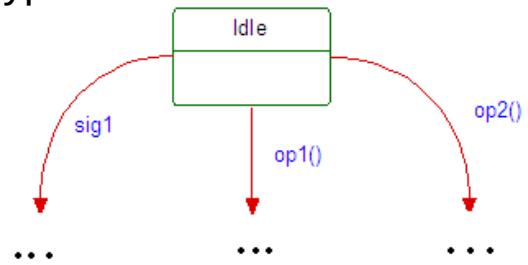




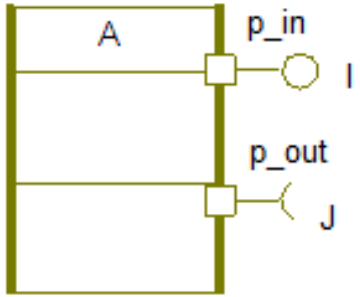
- Action executed by A:

  p.op2() //p conforms to J

- For each request received by p, the type system has to verify if it conforms with port type

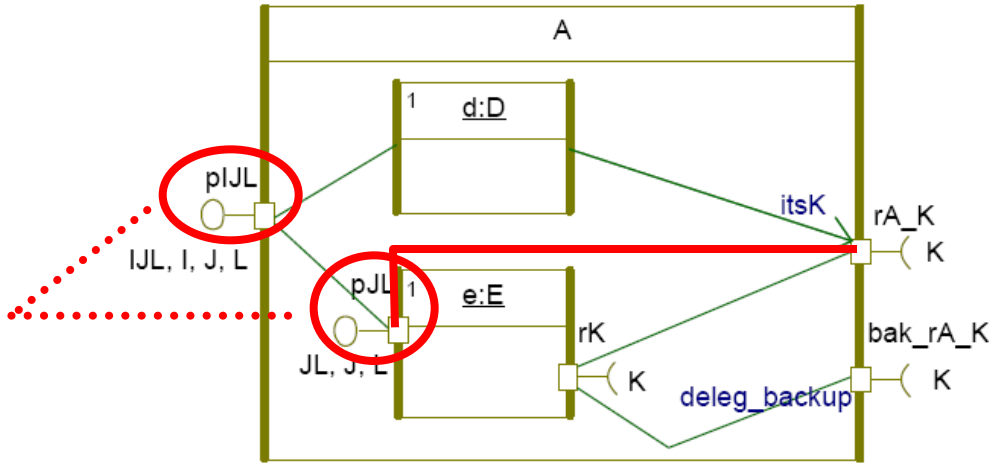


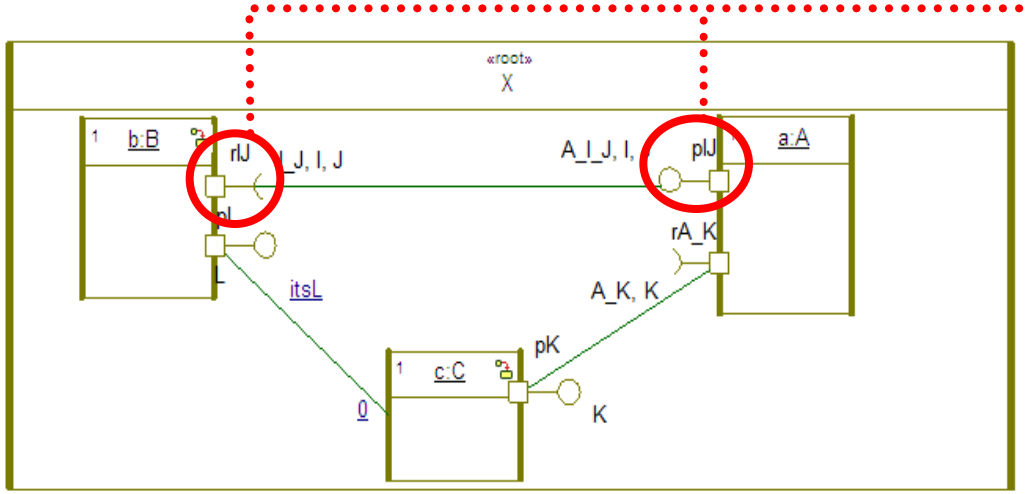
⇒ Bidirectional ports are forbidden!



Connectors directionality

For a delegation connector, both ports must have the same direction.

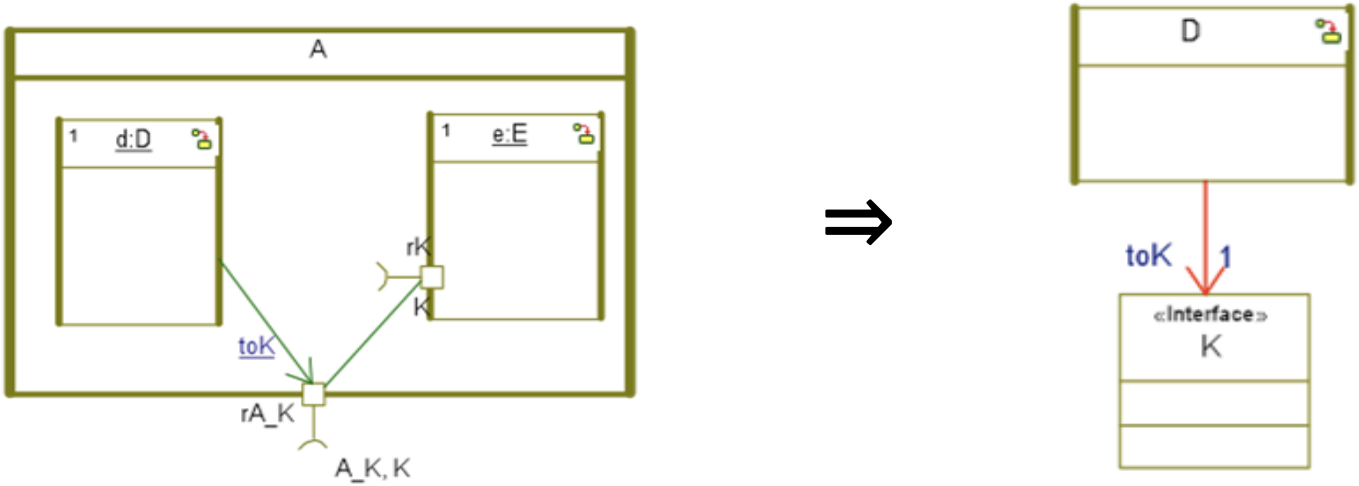




For an assembly connector, one port must be required and the other provided.

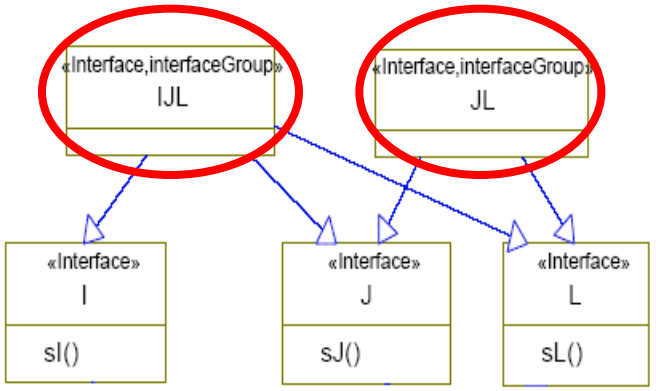
Static typing of connectors

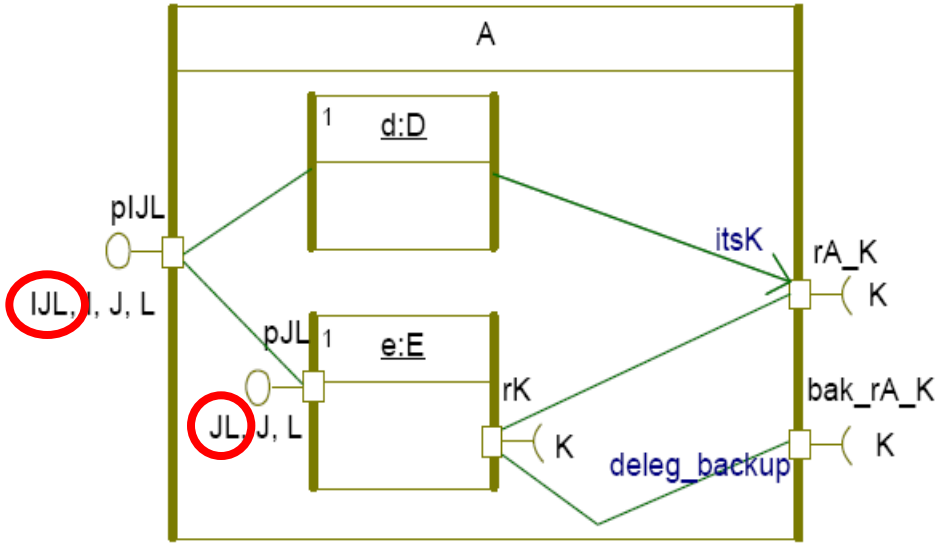
- UML: typing a connector with an association is optional
- OMEGA2: there are cases where typing a connector with an association is necessary



Providing / requiring multiple interfaces

- UML: interfaces inheritance
- OMEGA2: interfaces stereotyped «interfaceGroup» which are not taken into account by the type system

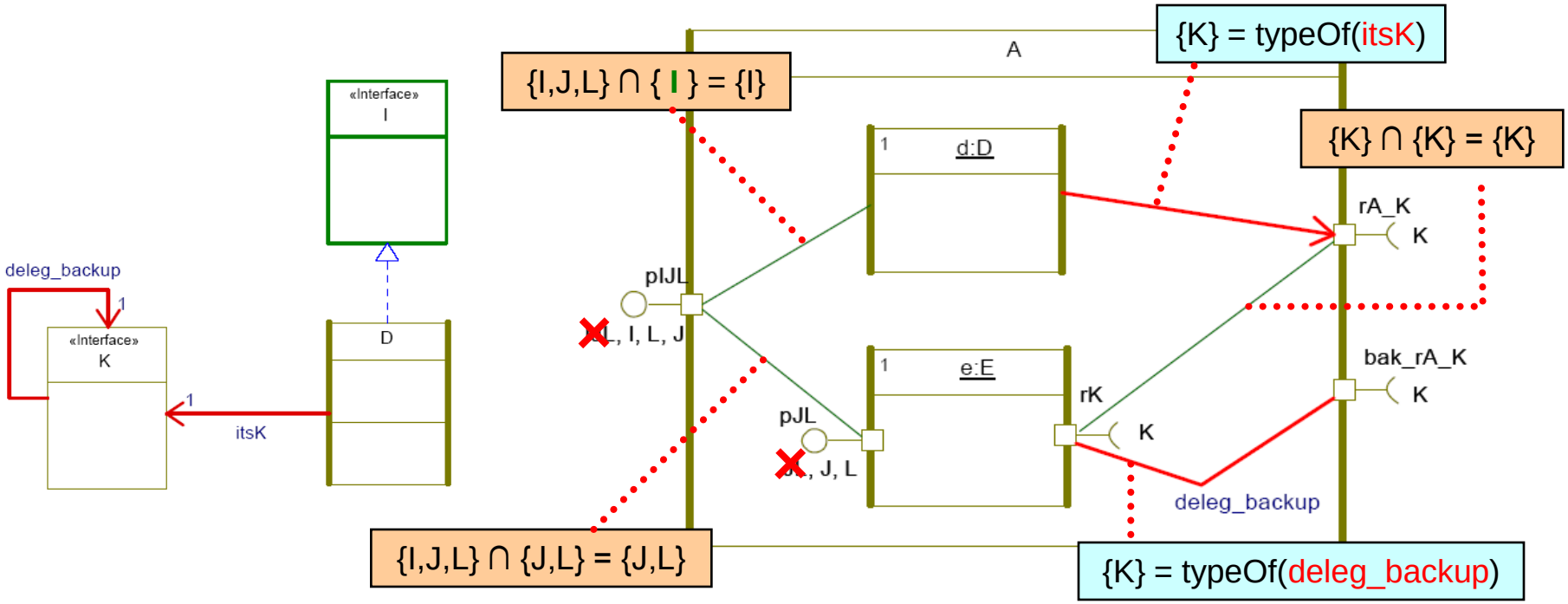




Set of transported interfaces

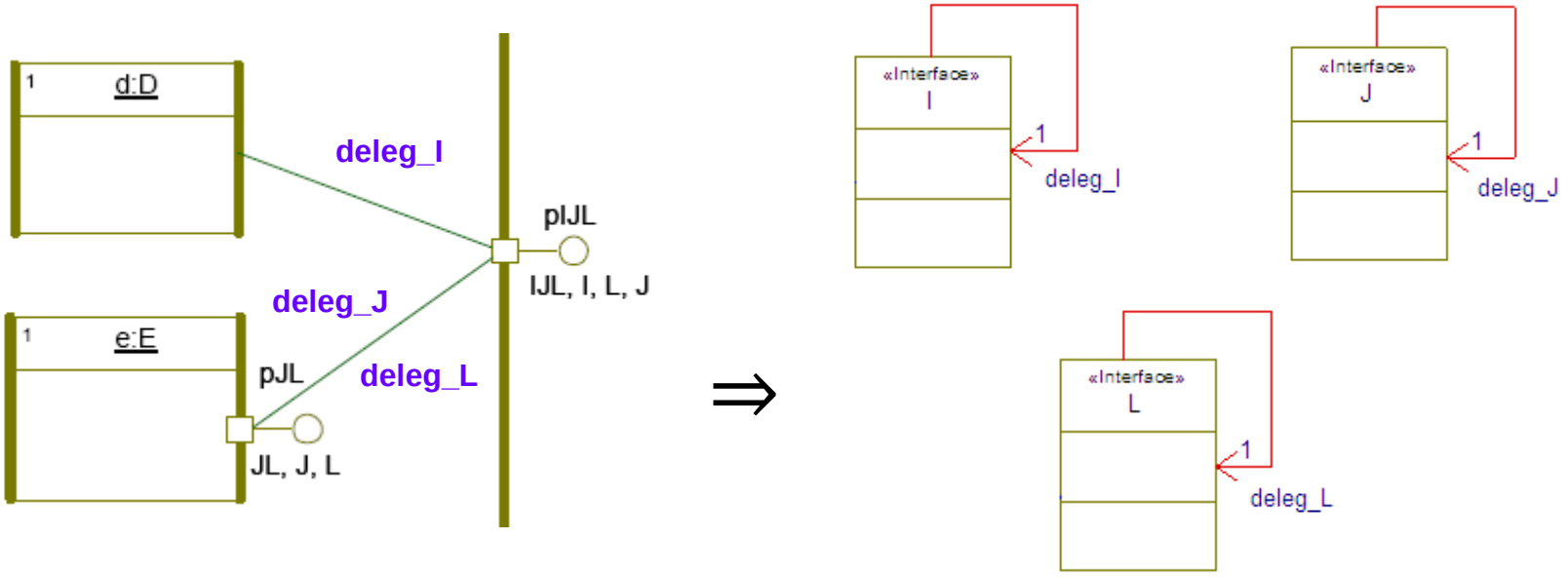
- Can be computed for connectors originating in ports and not-typed with associations

= Intersection between the two sets of provided / required interfaces at the two ends of the connector



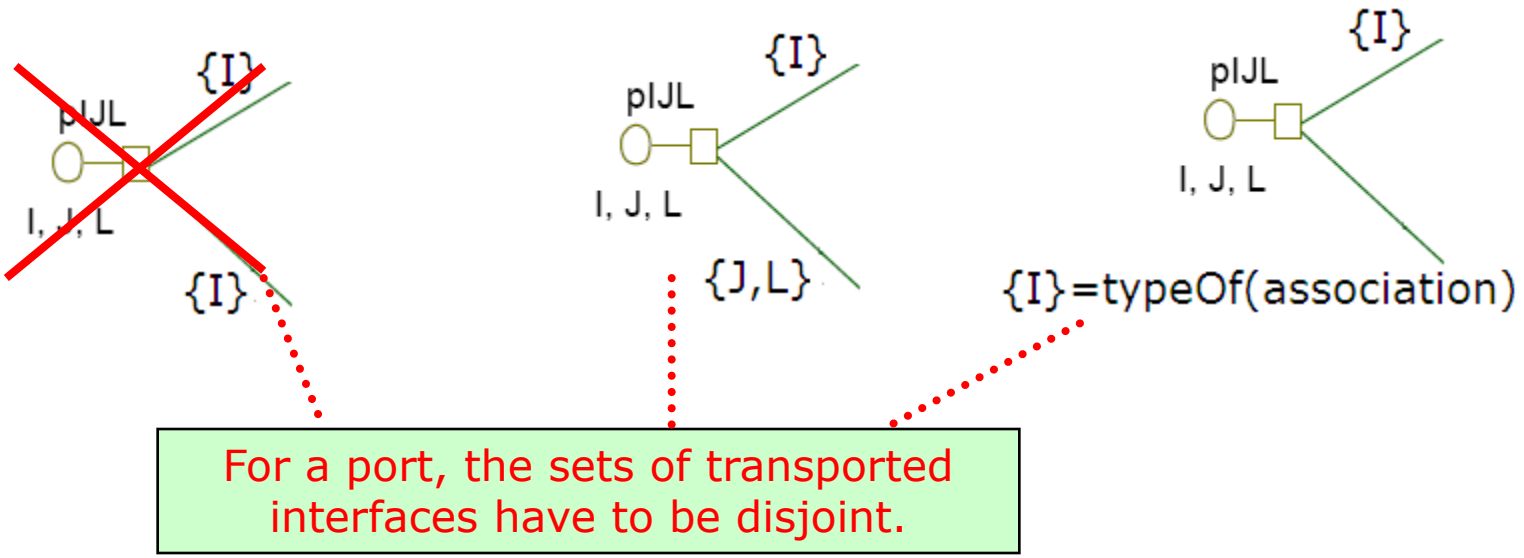
Implicit association and routing destination

- In OMEGA2, each interface has an association pointing to itself
- The association is initialized with the destination of requests conforming to the proprietary interface



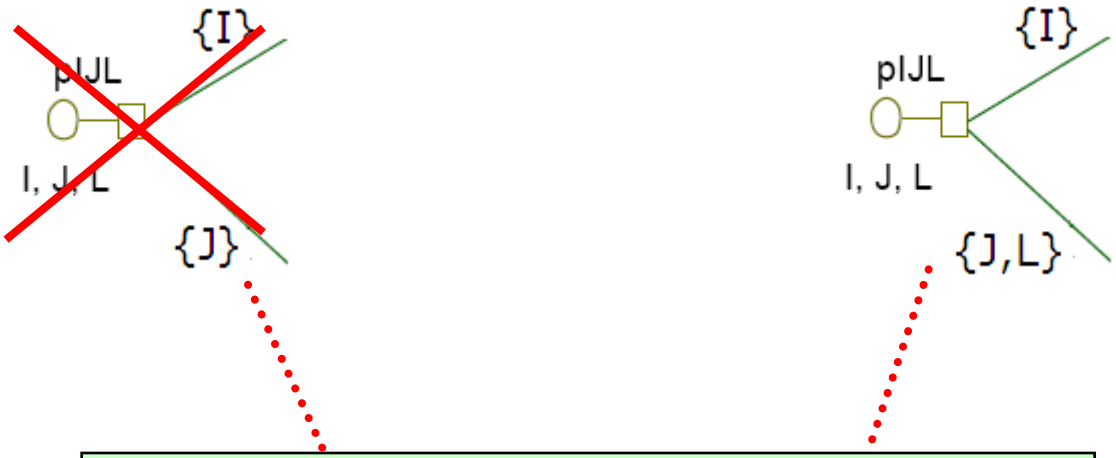
Port behavior

- By default: forwarding received requests (conform to its direction)
- Deterministic routing

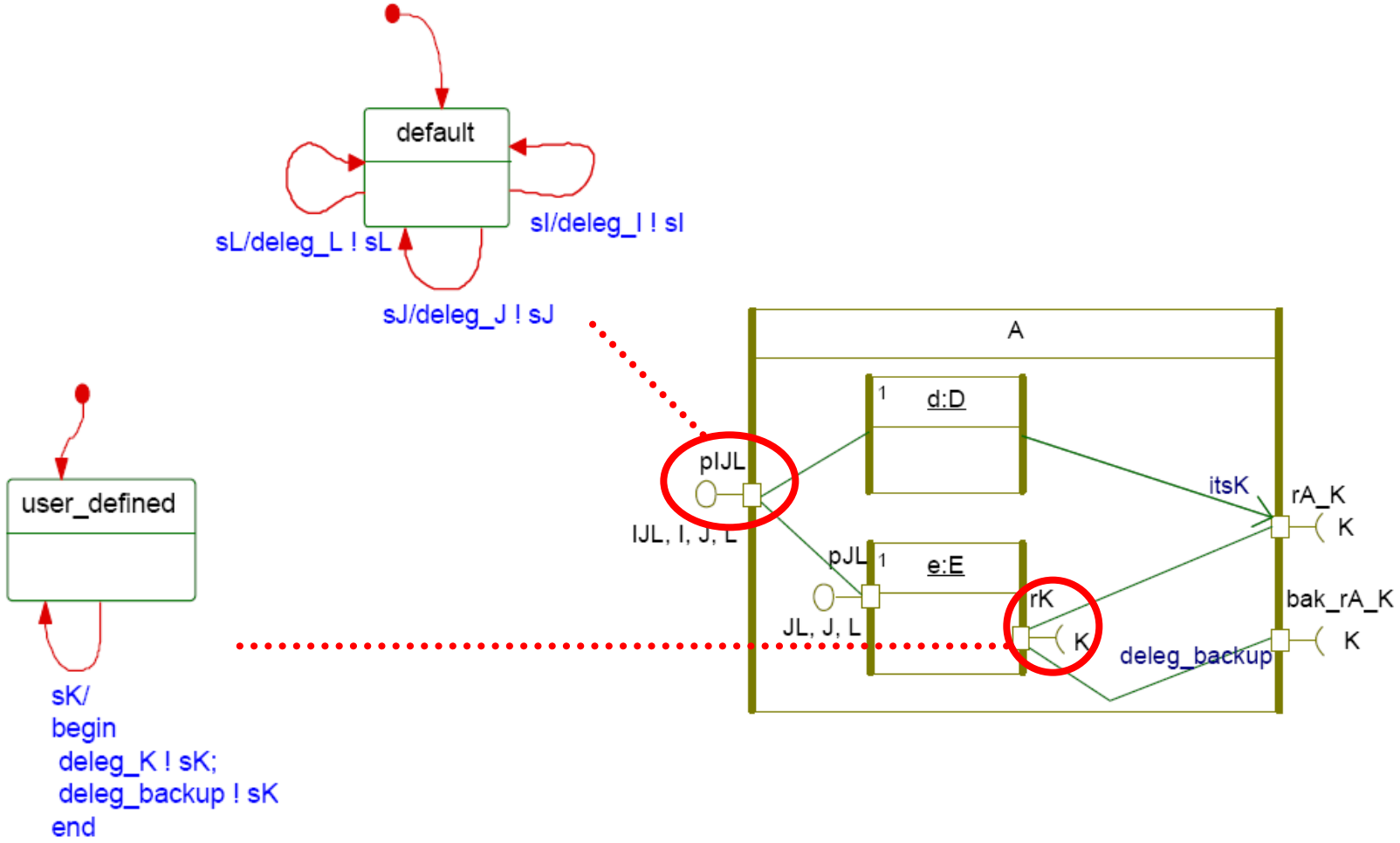


Port behavior

- Completeness



Ports behavior



Outline

- Overview of the OMEGA Profile
- Composite Structures
- Implementation and Evaluation
- Conclusions and Future Work

Composite Structures in IFx2

- Same overall architecture
  - Translation of OMEGA2 models to IF language
- Principles
  - Ports and connectors are handled as first class elements
  - Priority rules for partial order reduction of the state space in order to avoid combinatorial explosion

OCL Formalization

- Developed and evaluated over UML models in XMI format

Computes de set of transported interfaces

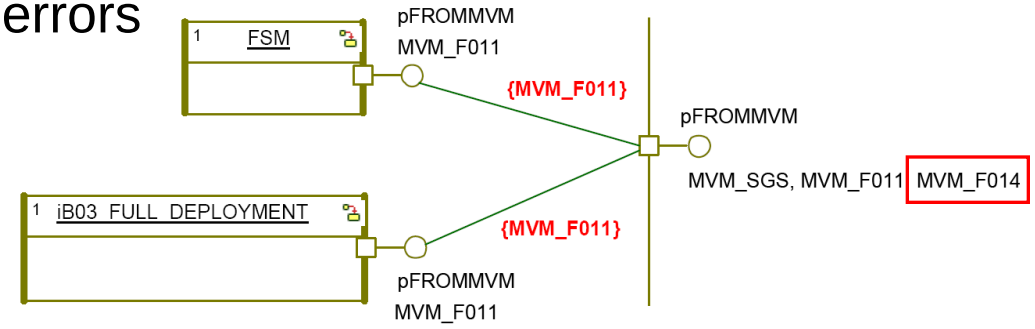
```
context Connector
def: setTransportedInterfaces : Set(Classifier) =
  if has2Parts
    then Set{OclInvalid}
  else if has2Ports
         then ((port1.interfaces)->intersection(port2.interfaces))
       else
         if isTyped
           then (port1.interfaces) -> intersection(associationEndPointType.interfaces)
         else (port1.interfaces) -> intersection(part1.type.interfaces)
         endif
       endif
  endif

-- Rule 6
inv SetOfTransportedInterfacesNonEmpty: self.setTransportedInterfaces->size()<>0
```

Verifies that a connector (typed or not-typed with an association) forwards requests

Reference: Iulia Dragomir and Iulian Ober. Well-formedness and typing rules for UML Composite Structures. arXiv/CORR submission no. 0136130, November 2010

Case study : ATV Solar Wing Management

- Complex model provided by Astrium Space Transportation
  - 3-level hierarchical architecture
  - 37 classes (from which 7 composite structures)
  - After system initialization: 93 active objects, ~380 ports and 200 connectors for communication
- Results
  - Untyped ports and connectors
  - Incomplete and non-unique ports
  - After simulation, modeling errors in system's behavior



Outline

- Overview of the OMEGA Profile
- Composite Structures
- Implementation and Evaluation
- Conclusions and Future Work

Conclusions and Future Work

- Composite structures = coherent and expressive models
- Approach based on a set of principles and notions for a clear operational semantics of OMEGA2 models
- Implementation in the IFx2 Toolset and evaluation on realistic models

- Current and future work
  - Formalization of Composite Structures type system and type safety proofs
  - Adaptation of the profile and tools to SysML