Université de Toulouse

Formalisation sémantique et vérification de structures composites

Iulia-Elena Dragomir

Encadrant: Iulian Ober
Equipe: MACAO

Objectif du stage

- Introduire les structures composite d'UML dans le profil OMEGA UML
- Définir un ensemble de principes pour clarifier leur sémantique
- Formalisation en OCL avec Topcased
- Formalisation en Isabelle/HOL

Sommaire

- Vue d’ensemble sur OMEGA v1
- Structures composites
- Formalisation en OCL
- Formalisation en Isabelle/HOL

Contexte: le langage OMEGA

- Profil UML pour la spécification et vérification des systèmes temps-réel

- Consiste en :

Un grand sous-ensemble d'UML

+

Des contraintes sur la cohérence d'un modèle

+

Une sémantique opérationnelle formelle

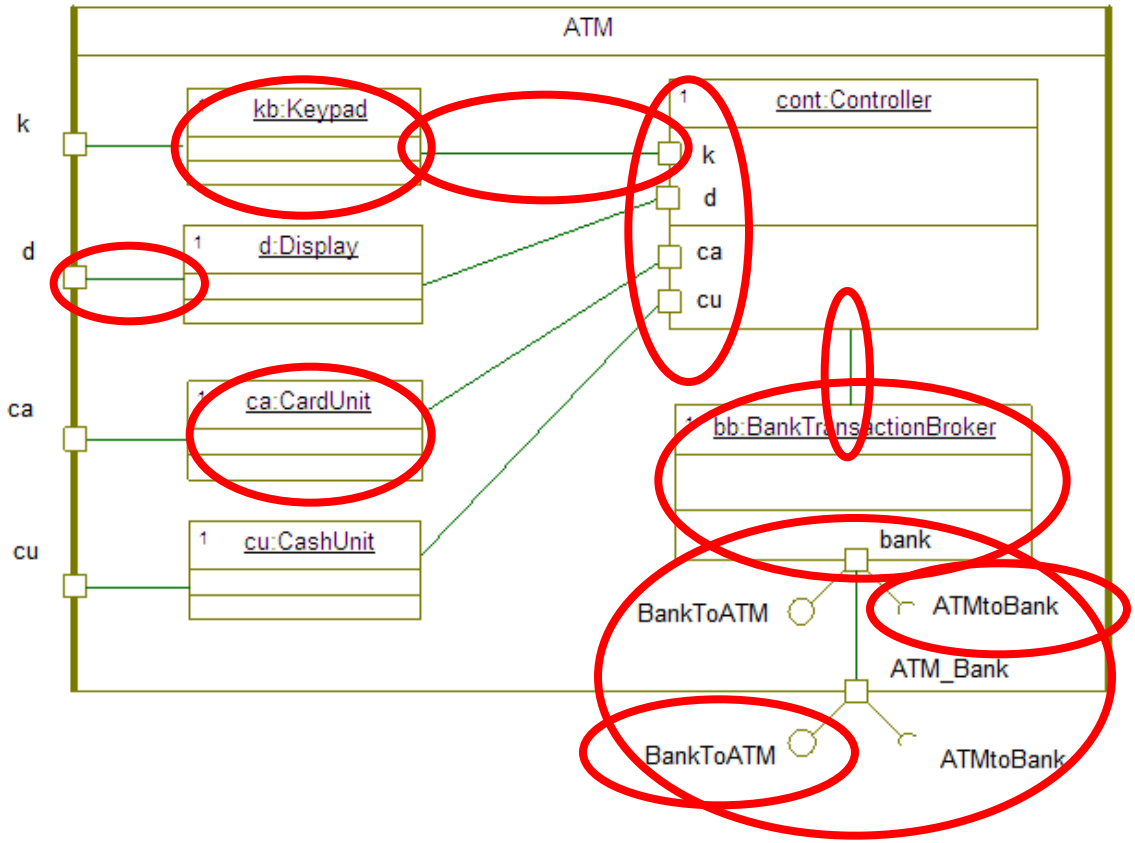
+

Des extensions pour le temps-réel et la vérification

Sommaire

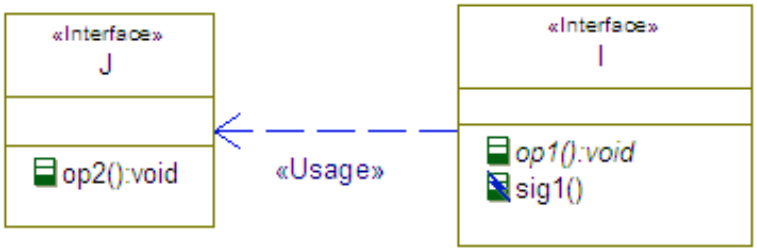
- Vue d'ensemble sur OMEGA v1
- Structures composites
- Formalisation en OCL
- Formalisation en Isabelle/HOL

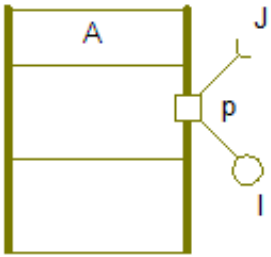
Structures composites



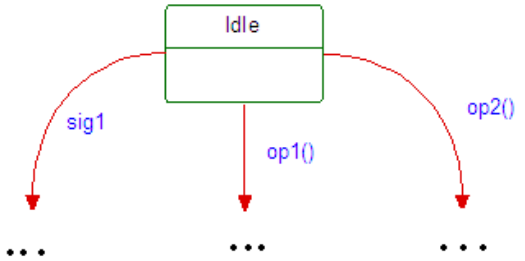
1. Sous-composants
2. Ports
3. Delegation connecteur (port-instance)
4. Delegation connecteur (port-port)
5. Assembly connecteur (instance-port)
6. Assembly connecteur (instance-instance)
7. Interface fournie
8. Interface requise

Directionalité de ports

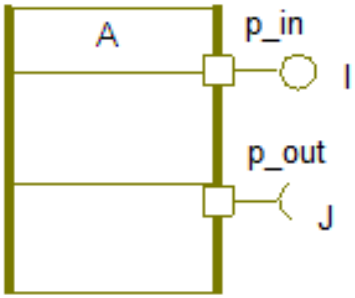




- Action exécutée par A:
  p.op2() //p conforme à J
- Pour chaque requête reçu par p, le système de types doit vérifier si elle corresponds au type du port

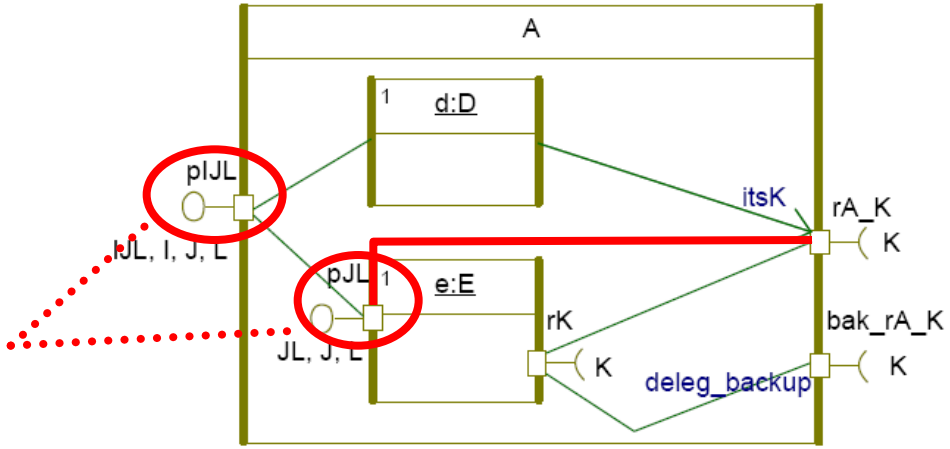


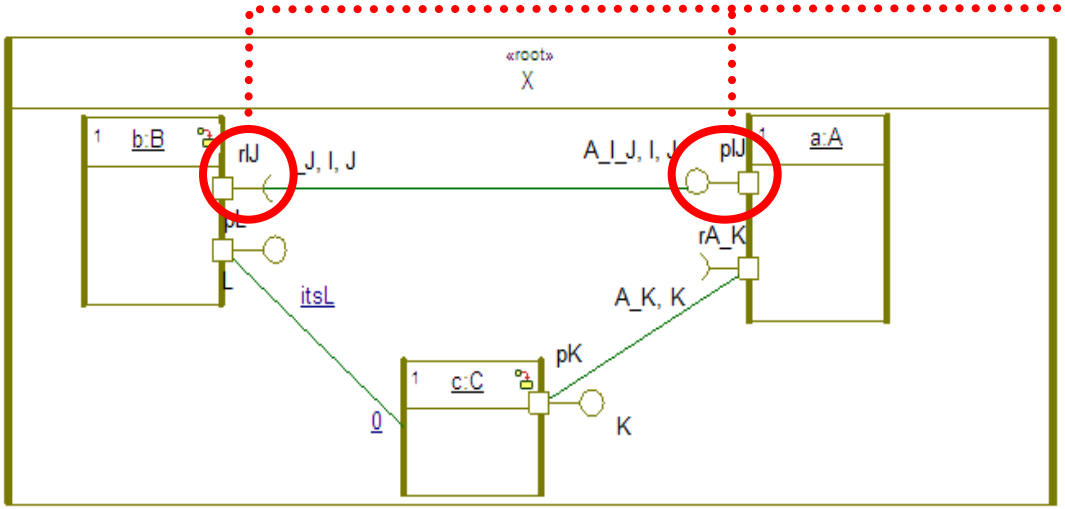
⇒ les ports bidirectionnels sont interdits!



Directionalité de connecteurs

Pour un «delegation connector», les deux ports doivent avoir la même direction

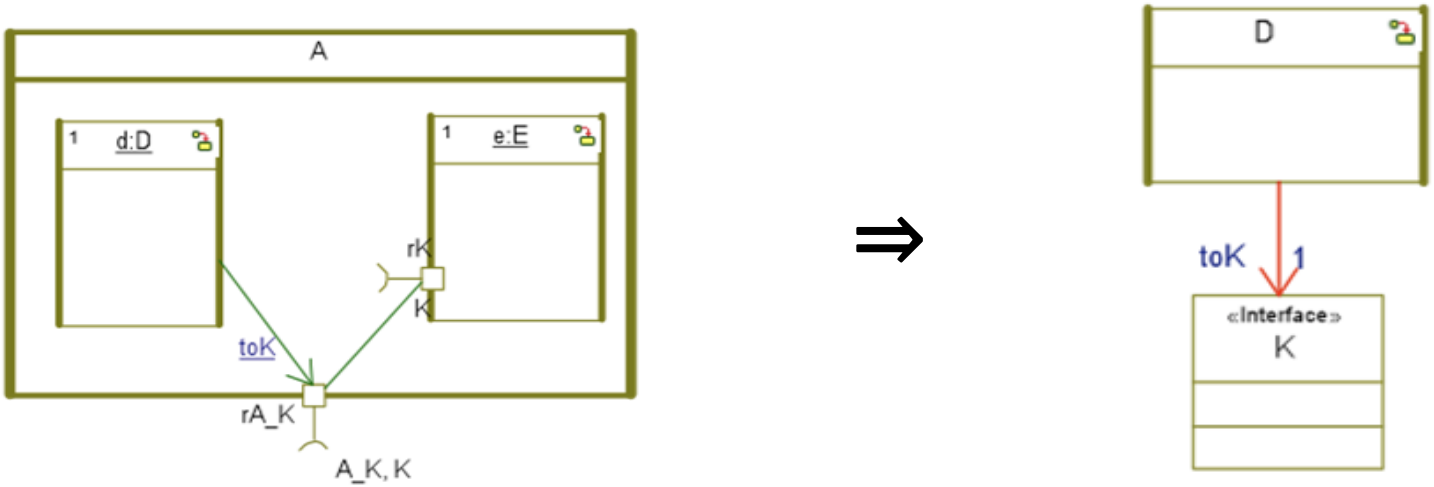




Pour un «assembly connector», un port doit être requis et l'autre fourni

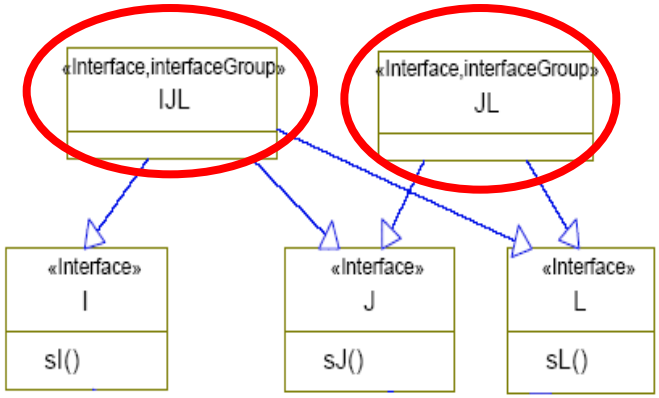
Typage statique de connecteurs

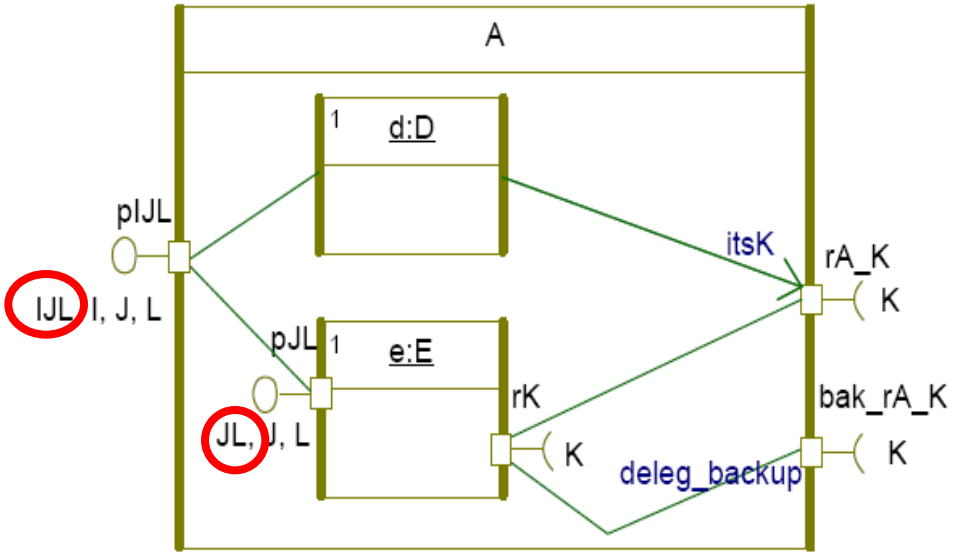
- En UML : le typage statique d'un connecteur avec une association est optionnel
- En OMEGA : il y a des cas où le typage est obligatoire



Support pour plusieurs interfaces sur le même port

- En UML : héritage des interfaces
- En OMEGA : interfaces « InterfaceGroup » qui ne sont pas prises en compte par le système de types

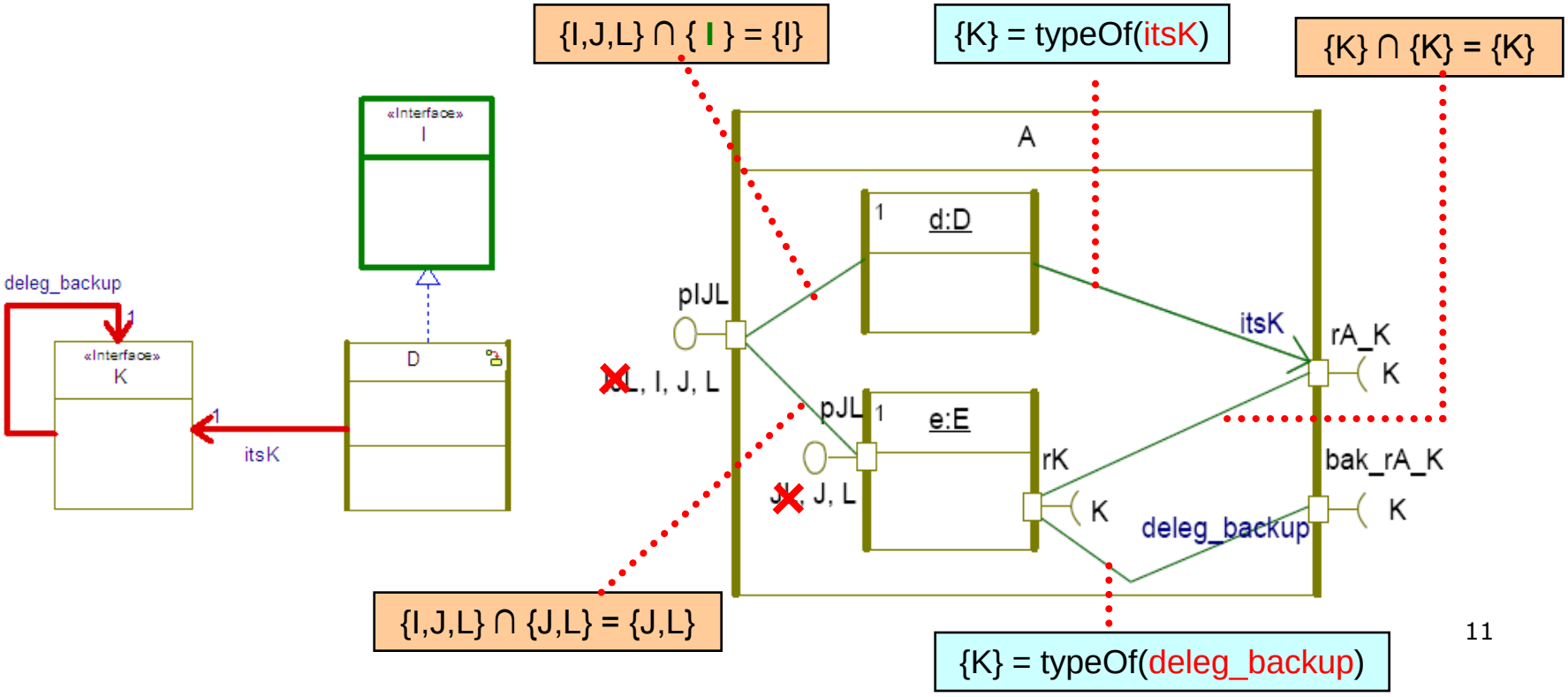




Ensemble des interfaces transportées

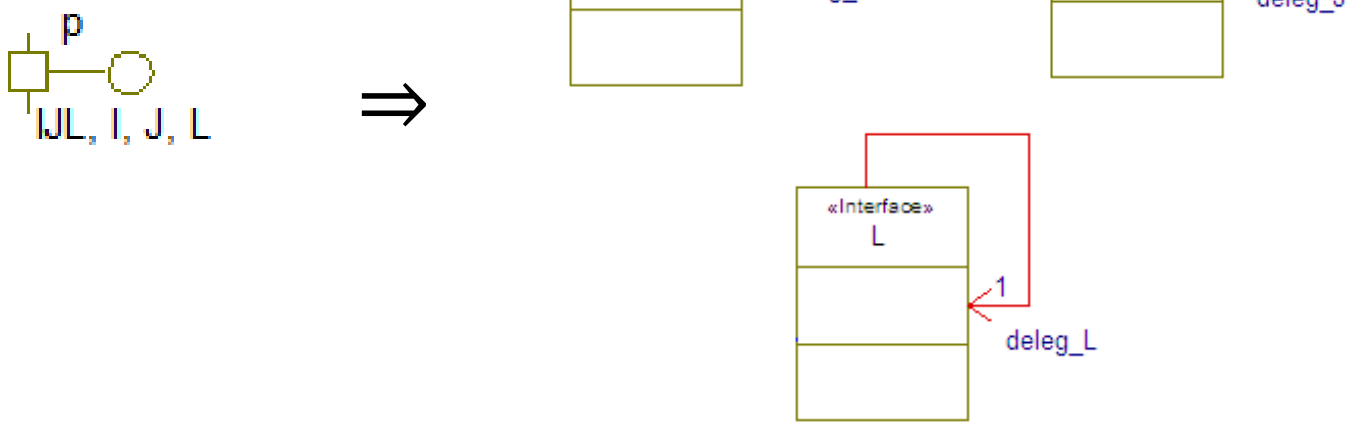
- Peut être calculé dans le cas de connecteurs partant des ports et non-typés avec association

= Intersection entre les ensembles des interfaces fournies/requises aux deux extrémités du connecteur



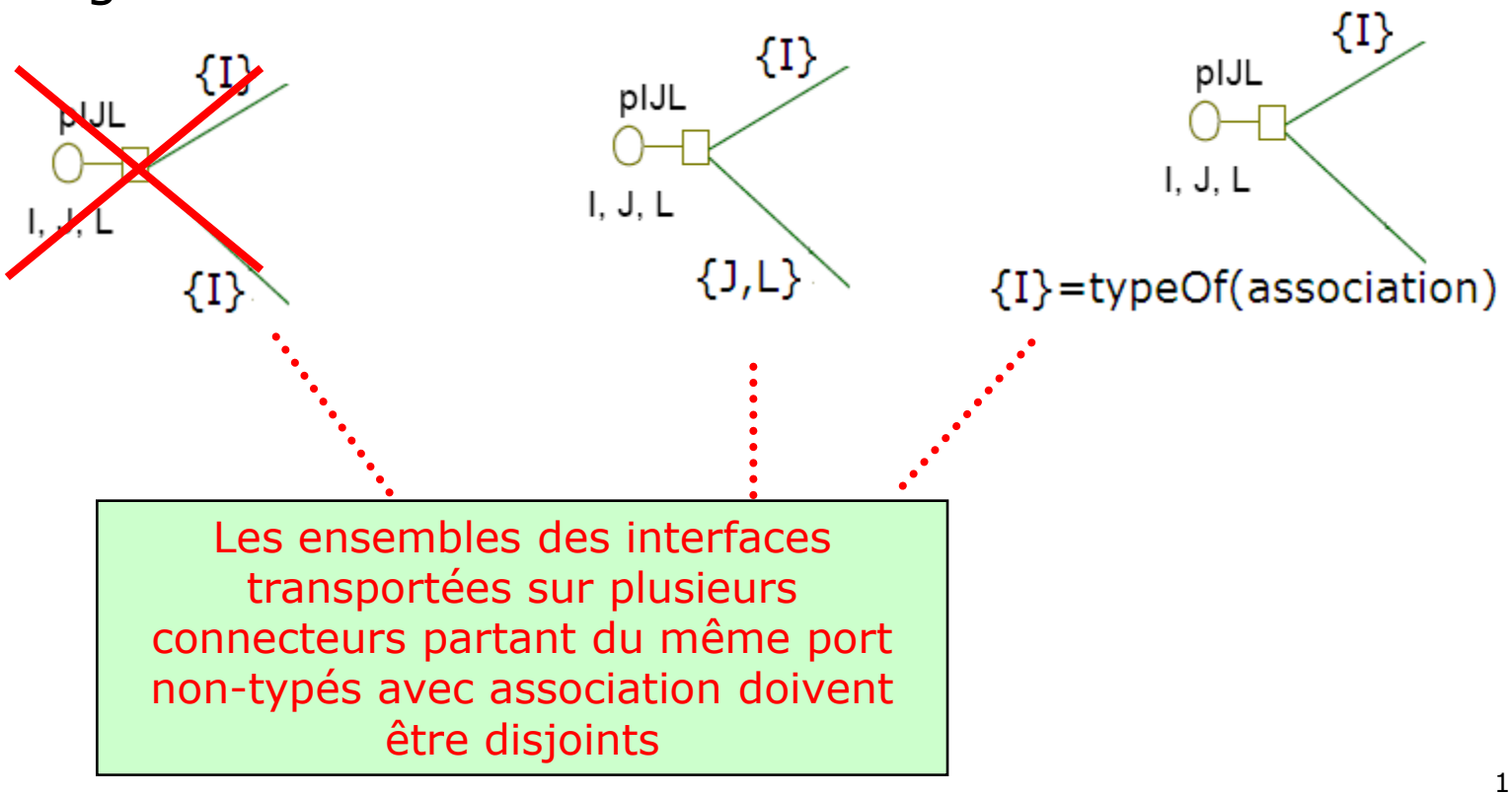
Association implicite et destination de routage

- En OMEGA : chaque interface a une association vers soi même
- L'association est initialisée avec la destination des requêtes conformes à l'interface propriétaire



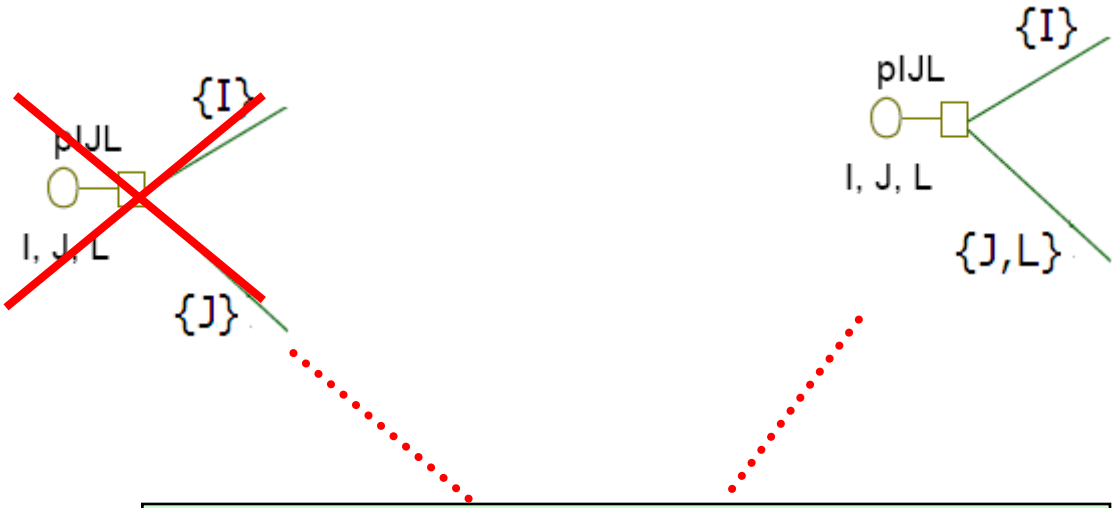
Comportement du port

- Par défaut : transférer les requêtes reçues (conformément à sa direction)
- Routage déterministe



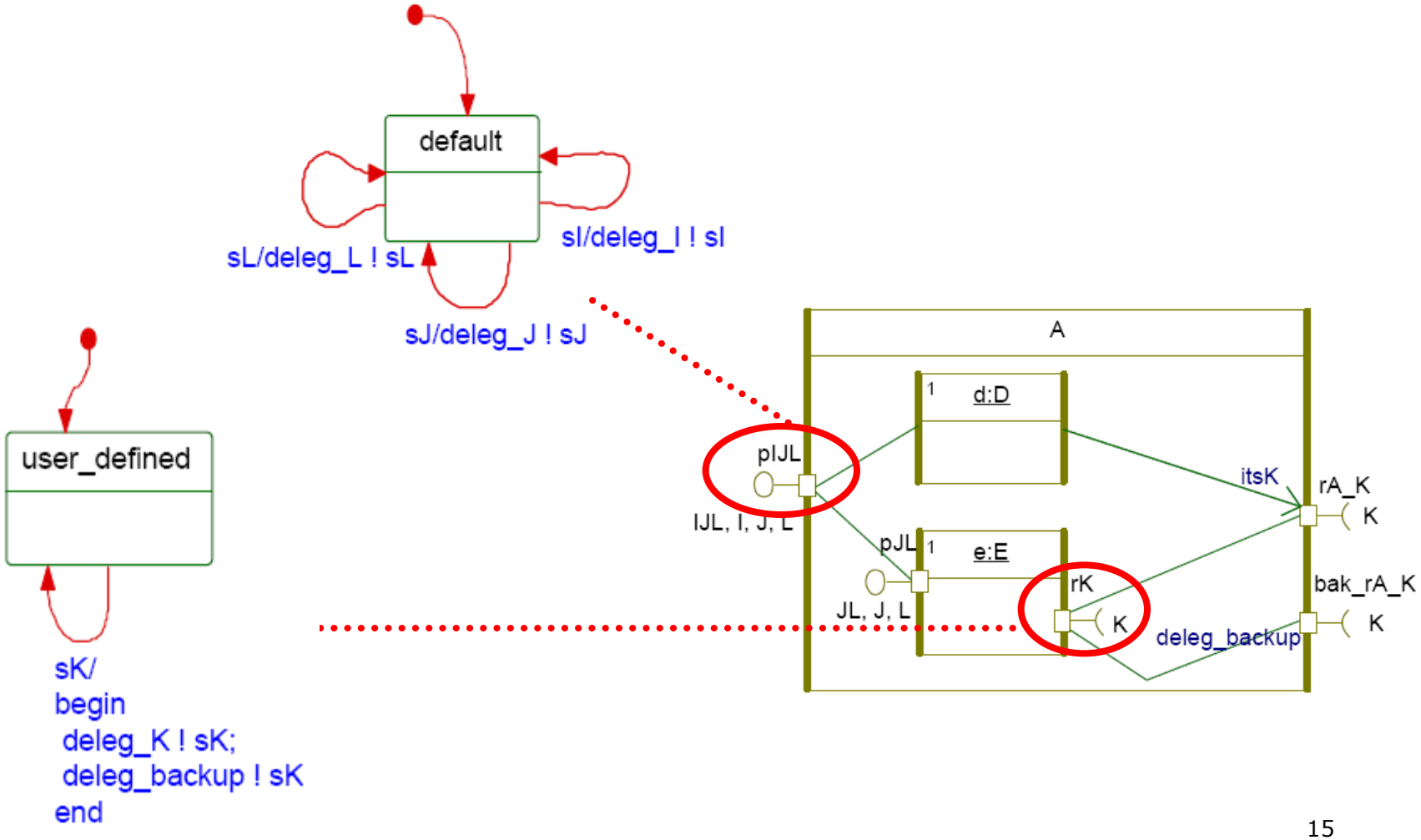
Comportement du port

- Complétude



Tous les interfaces fournies/requises d'un port doivent appartenir à un ensemble des interfaces transportées d'un connecteur

Comportement des ports



Sommaire

- Vue d'ensemble sur OMEGA v1
- Structures composites
- Formalisation en OCL
- Formalisation en Isabelle/HOL

Formalisation OCL – Ensemble des interfaces transportées

Calcule les interfaces fournies d'un sous-composant, directement ou indirectement réalisées

```
context Class
def: interfaces : Set(Classifier) =
  iRealizations->union(iRealizations->iterate(i:Interface;
     res:Set(Classifier)=Set{}| res->union(i.getParentsRec)))
  ->union(self.getParentsRec->iterate(c:Class; res:Set(Classifier)=
     Set{}| res->union(c.interfaces)))
  ->reject(isInterfaceGroup)
```

Calcule les interfaces fournies/requises d'un port

```
context Port
def: interfaces : Set(Classifier) =
  self.provided->reject(isInterfaceGroup)
```

...

Vérifie que chaque connecteur (typé ou non-typé avec une association) peut transférer des requêtes

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

Etude de cas : ATV Solar Wing Management

- Modèle réel développé en SysML par Astrium Space Transportation :
  - Hiérarchie sur 3 niveau
  - 37 blocs, dont 7 blocs composites
  - Après l'initialisation du système: 93 objets actifs, ~380 ports et ~200 connecteurs pour la communication

- Résultats :
  - Fautes de modélisation trouvées dans le modèle
  - Deadlock dans le comportement du modèle

Sommaire

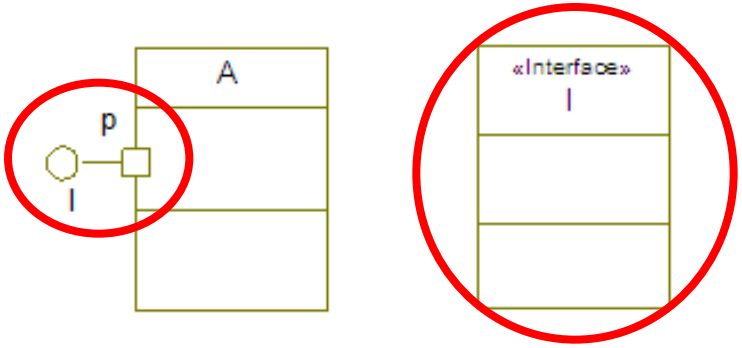
- Vue d'ensemble sur OMEGA v1
- Structures composites
- Formalisation en OCL
- Formalisation en Isabelle/HOL

Typage sûr des structures composites

Pour nos structures composites, typage sûr signifie:

1. Chaque requête sera transférée par des ports et connecteurs jusqu'à une destination finale
2. Chaque sous-composant recevra uniquement des requêtes compatibles avec les interfaces directement ou indirectement réalisées

Syntaxe abstraite



portDecl ::= *tname* × *portTb*
portTb ::= *portDirection* × *tname* × *tname*
portDirection ::= **provided** | **required**
…
intfDecl ::= *tname* × *intfTb*
intfTb ::= *intfType* × *(tname)list* × *(tname)list*
intfType ::= **interfaceGroup** | **none**
…
model ::= *(clssDecl)list* × *(intfDecl)list* × *(portDecl)list*

Modèle bien-formé

- Règles supplémentaires pour la syntaxe abstraite

wf_ports(M) = ∀ port dans le modèle M, son contrat est une interface ∧ son propriétaire est une classe

- Modèle OMEGA bien-formé

wf_model(M) = wf_structuresComposites(M) ∧ wf_reglesSupplementaires(M) ∧ ws_model(M)

Ensemble des interfaces transportées – Fonctions récursives en Isabelle/HOL

Calcule récursivement les interfaces réalisées par une interface

```
function recIntf :: "model ⇒ tname ⇒ (tname) list"
where
"recIntf M Intf = ( case (iface_ M Intf) of  None ⇒ [] |
                                             Some (st,si,ri) ⇒
                        ( if ws_model M  then
                             if si=[]  then []
                             else  si @ (concat (map (λ x. recIntf M x) si))
                          else [])
                  )
"
 by pat_completeness auto
 termination apply ( relation "ws_wfrel subint1")
```

Relation pour prouver la terminaison de la fonction récursive

```
 constdefs
   ws_wfrel :: "(model ⇒ (tname × tname)set) ⇒ ((model × tname) ×
(model × tname))set"
   "ws_wfrel R ≡ ((M, Intf), (M', Intf')). M'=M ∧ ws_model M ∧
(Intf',Intf)∈ R M"
```

Travaux futurs

- Prouver que les relations de terminaison sont bien-formé
- Finir la formalisation de la sémantique statique du profil OMEGA
- Définir la sémantique opérationnelle pour le profil
- Prouver le typage sûr

Sommaire

- Vue d'ensemble sur OMEGA v1
- Structures composites
- Formalisation en OCL
- Formalisation en Isabelle/HOL

Conclusions

- Structures composites = modèles plus cohérents et expressifs
- Ensemble de notions et règles pour clarifier leur sémantique
- Formalisation en OCL pour les fautes de modélisation
- Formalisation en Isabelle/HOL pour le typage sûr
- Implantation des règles dans le compilateur du profil OMEGA
- Validation avec des modèles réels

Bilan

- Formalisation en OCL et utilisation de Topcased
- Définition d'une syntaxe abstraite pour le profil OMEGA en Isabelle/HOL et formalisation de la sémantique statique
- Modélisation de systèmes en SysML avec Rhapsody7.5
- Utilisation d'Eclipse EMF et Eclipse UML2 pour le compilateur d'OMEGA

- Publications

1. Iulian Ober, Iulia Dragomir. *OMEGA2 : A new version of the profile and the tools*. *UML&AADL'2010 – 15th IEEE ICECCS, Oxford, Royaume Uni, 24/03/10-25/03/10,* IEEE, p. 373-378, 2010.
2. Iulian Ober, Iulia Dragomir. *Unambiguous UML composite structures : the OMEGA2 experience.* Soumise à *ACM/IEEE MoDELS'2010*, en attente de la décision.