Aide Mémoire OCL 2.2

Eric Cariou – Université de Pau et des Pays de l'Adour

1 Constructions de base

Invariant :
context Type inv nomInv: *expr-bool*

Spécification d'une méthode :
context Type::*methode(var : Type, ...) : Type*
pre nomPre: *expr-bool*
post nomPost: *expr-bool (... result ... val@pre)*

Conditionnelles :
if *expr-bool* **then** *expr-bool* **else** *expr-bool* **endif**
expr-bool **implies** *expr-bool*

Définition d'une variable locale (invariant, pre/post) :
let *maVar : Type = expr* **in**

Définition d'une variable globale :
context Type def: *maVar : Type = expr*

Définition d'une opération :
context Type def: *monOp(param : Type, ...) : Type = expr*

Fermeture transitive :
elt -> **closure**(*assoc*)

Élément d'une énumération :
#valeur ou **Enum::valeur**

Accès aux propriétés :
- Collections : *col –> prop*
- Objet / élément : *elt.prop* (le « . » s'utilise aussi pour les navigations sur le modèle même quand on récupère des collections)

2 Collections

Sauf précision explicite d'un autre type de retour, les opérations suivantes pour les collections et les objets retournent une valeur booléenne.

2.1 Opérations communes

2.1.1 Inclusion

includes(elt) la collection contient l'élément *elt*

excludes(elt) la collection ne contient pas l'élément *elt*

includesAll(col) la collection contient tous les éléments de la collection *col*

excludesAll(col) la collection ne contient aucun des éléments de la collection *col*

2.1.2 Filtre / sélection

Syntaxe, au choix :

col –> operation (prop ...)

col –> operation (elt | elt.prop ...)

col –> operation (elt : Type | elt.prop ...)

select(expr-bool) retourne le sous-ensemble de la collection dont les éléments respectent la contrainte spécifiée

reject(expr-bool) idem mais ne garde que les éléments ne respectant pas la contrainte

exists(expr-bool) au moins un élément respecte la contrainte spécifiée

forAll(expr-bool) tous les éléments respectent la contrainte spécifiée

one(expr-bool) un et un seul des éléments respecte la contrainte spécifiée

any(expr-bool) retourne un des éléments de la collection respectant la contrainte spécifiée

collect(expr) retourne une collection (mise à plat) construite à partir d'une expression donnée

collectNested(expr) idem que collect mais sans la mise à plat si retourne une imbrication de collections

isUnique(expr) calcule une nouvelle collection comme pour le collect et retourne vrai s'il n'y existe pas deux éléments identiques

sortedBy(expr) retourne un ordered set contenant tous les éléments de la collection triés selon *expr*

2.1.3 Divers

isEmpty() la collection est vide

notEmpty() la collection n'est pas vide

size() nombre d'éléments de la collection

count(elt) nombre d'occurences de l'élément *elt* dans la collection

max() / min() / sum() retourne le max / le min / la somme des éléments de la collection (si d'un type d'éléments compatible)

Type.allInstances() la collection de tous les éléments de type *Type* présents dans le modèle

2.1.4 Conversions

asSet(), asBag(), asSequence(), asOrderedSet() transforme la collection courante en une collection équivalente d'un autre type

flatten() mise à plat de collections imbriquées

2.2 Opérations spécifiques à certaines collections

Voir le tableau ci-contre pour savoir quelle opération spécifique est accessible selon le type de la collection sur laquelle elle s'applique.

2.2.1 Relations entre collections

union(col) l'union de la collection avec *col*

intersection(col) l'intersection de la collection avec *col*

including(elt) la collection augmentée de l'ajout de l'élément *elt*

excluding(elt) la collection dont on a enlevé l'élément *elt*

- col la collection dont on a retiré les éléments se trouvant dans *col*

symmetricDifference(col) les éléments se trouvant dans la collection ou dans *col* mais pas dans les deux à la fois

2.2.2 Accès ordonné au contenu

first() le premier élément de la collection

last() le dernier élément de la collection

at(index : Integer) l'élément de la collection se trouvant en position *index*

indexOf(elt) la position de l'élément *elt* dans la collection

append(elt) la collection augmentée de l'élément *elt* placé à la fin

prepend(elt) la collection augmentée de l'élément *elt* placé au début

insertAt(index : Integer, elt) la collection augmentée de l'élément *elt* placé à la position *index*

subOrderedSet(lower : Integer, upper : Integer) l'ordered set contenant les éléments de la position *lower* à *upper* à partir d'un ordered set

subSequence(lower : Integer, upper : Integer) la séquence contenant les éléments de la position *lower* à *upper* à partir d'une séquence

reverse() la même collection mais avec les éléments inversés en position

	Set	Bag	OrderedSet	Sequence
union	×	×		×
intersection	×	×		
including	×	×		×
excluding	×	×		×
-	×			
symmetricDifference	×			
append			×	×
prepend			×	×
insertAt			×	×
at			×	×
indexOf			×	×
first			×	×
last			×	×
reverse			×	×
subSequence				×
subOrderedSet			×	

Types de collection :

- **Set** : pas d'ordre et pas de doublon
- **Bag** : pas d'ordre et doublons possibles
- **OrderedSet** : ordre et pas de doublon
- **Sequence** : ordre et doublons possibles

3 Objets

3.1 Type, spécialisation

oclIsTypeOf(type) l'élément est exactement du type *type*

oclIsKindOf(type) l'élément est du type *type* ou d'un de ses sous-types

oclAsTypeOf(type) l'élément est manipulé comme étant du type *type* (cast)

3.2 Divers

oclInState(state) l'élément est dans l'état *state*

oclIsNew() l'élément a été créé pendant l'exécution d'une opération (utilisable dans une post-condition)

oclIsUndefined() l'élément (attribut ou bout d'association en cardinalité 0..1) n'est pas positionné

oclIsInvalid() l'évaluation d'une expression renvoie un résultat invalide

4 Norme OCL

Accès à la norme OCL 2.2 sur le site de l'OMG :
`http://www.omg.org/spec/OCL/2.2/`