

## E.2. Signatures

**Class/data type hierarchies.** A *class/data type hierarchy*  $(C, \leq_C)$  is given by a partial order where the set  $C$  contains the *class/data type names*, which are closed w.r.t. the *built-in data types* Boolean, UnlimitedNatural, Integer, Real, and String, i.e.,  $\{\text{Boolean, UnlimitedNatural, Integer, Real, String}\} \subseteq C$ ; and the partial ordering relation  $\leq_C$  represents a *generalization relation* on  $C$ , where  $c_1$  is a *sub-class/data type* of  $c_2$  if  $c_1 \leq_C c_2$ .

A *class/data type hierarchy map*  $\gamma : (C, \leq_C) \rightarrow (D, \leq_D)$  is given by a monotone map from  $(C, \leq_C)$  to  $(D, \leq_D)$ , i.e.,  $\gamma(c) \leq_D \gamma(c')$  if  $c \leq_C c'$ , such that  $\gamma(c) = c$  for all  $c \in \{\text{Boolean, UnlimitedNatural, Integer, Real, String}\}$ .

The *collection type constructors* OrderedSet, Set, Sequence, and Bag are used for representing the meta-attributes “ordered” and “unique” of MultiplicityElement according to the following table:<sup>1</sup>

	ordered	not ordered
unique	OrderedSet	Set
not unique	Sequence	Bag


The default is “not ordered” and “unique”.<sup>2</sup>


For a class/data type  $c \in C$  of a class/data type-hierarchy  $(C, \leq_C)$  and a collection type constructor

$$\tau \in \{\text{OrderedSet, Set, Sequence, Bag}\},$$

the expression  $\tau[c]$  denotes the induced *collection type*.

Let  $(C, \leq_C)$  be a class/data type hierarchy.

- An *attribute declaration* over  $(C, \leq_C)$  is of the form  $c.p : \tau[c']$  with  $c, c' \in C$ ,  $\tau$  a collection type constructor, and  $p$  an *attribute name*. (Attributes and association member ends are distinguished due to their different uses. In UML, both are of class Property.)
- A *query operation declaration* over  $(C, \leq_C)$  is of the form  $c.q(x_1 : \tau_1[c_1], \dots, x_r : \tau_r[c_r]) : \tau[c']$  with  $c, c_1, \dots, c_r, c' \in C$ ,  $\tau$  a collection type constructor,  $o$  an *operation name*, and  $x_1, \dots, x_r$  *parameter names*.
- An *association declaration* over  $(C, \leq_C)$  is of the form  $a(p_1 : \tau_1[c_1], \dots, p_r : \tau_r[c_r])$  with  $r \geq 2$ ,  $c_1, \dots, c_r \in C$ ,  $\tau_1, \dots, \tau_r$  classifier annotations,  $a$  an *association name*, and  $p_1, \dots, p_r$  *member end names*.<sup>3</sup> An association declaration  $\mathbf{a} = a(p_1 : \tau_1[c_1], \dots, p_r : \tau_r[c_r])$  yields the *property declarations*  $\mathbf{a}.p_i : \tau_i[c_i]$  for  $1 \leq i \leq r$ . An association declaration is *binary* if  $r = 2$ .<sup>4</sup>
- A *composition declaration* over  $(C, \leq_C)$  is of the form  $m(p_1 : \text{Set}[c_1], \blacklozenge p_2 : \tau_2[c_2])$  with  $c_1, c_2 \in C$ ,  $\tau_2$  a collection type constructor,  $m$  a *composition name*, and  $p_1, p_2$  *member end names*. A composition declaration  $\mathbf{m} = m(p_1 : \text{Set}[c_1], \blacklozenge p_2 : \tau_2[c_2])$  yields the *property declarations*  $\mathbf{m}.p_1 : \text{Set}[c_1]$  and  $\mathbf{m}.p_2 : \tau_2[c_2]$ . 

In UML, each Property may have AggregationKind composite. However, such an aggregation kind has no semantic meaning when the property is not a member end of an association: the UML Superstructure Specification 2.4.1 does not mention the aggregation kind in the description of the semantics of Property, and UML 2.5 explains the use of aggregations for Property as “to model circumstances in which *one instance* is used to group together a set of instances” (p. 112, our emphasis). Moreover, composite properties, i.e., properties with aggregation kind composite can only be member ends of binary associations (UML Superstructure Specification 2.4.1, p. 37; UML 2.5, p. 228) and their multiplicity must not exceed one (UML Superstructure Specification 2.4.1, p. 126; UML 2.5, p. 155). Thus, composition declarations are distinguished from general association declarations. 

**Class/data type nets (Signatures).** A *class/data type net*  $\Sigma = ((C, \leq_C), P, O, A, M)$  comprises a class/data type hierarchy  $(C, \leq_C)$  and a set  $P$  of attribute declarations, a set  $O$  of operation declarations, a set  $A$  of association declarations over  $(C, \leq_C)$ , and a set  $M$  of composition declarations over  $(C, \leq_C)$ , such that the following properties are satisfied:

- attribute names are unique along the generalization relation: if  $c_1.p_1 : \tau_1[c'_1]$  and  $c_2.p_2 : \tau_2[c'_2]$  are different property declarations in  $P$  and  $c_1 \leq_C c_2$ , then  $p_1 \neq p_2$ ;
- association and composition names are unique: if  $d_1$  and  $d_2$  are the names of two different association or composition declarations in  $M \cup A$ , then  $d_1 \neq d_2$ ;
- member end names are unique: if  $p_1, \dots, p_r$  are the member end names of an association declaration in  $A$  or a composition declaration in  $M$ , then  $p_i \neq p_j$  for  $1 \leq i \neq j \leq r$ ;<sup>5</sup>

<sup>1</sup>Cf. UML Superstructure Specification 2.4.1, p. 128; UML 2.5, p. 27.

<sup>2</sup>UML Superstructure Specification 2.4.1, p. 96; there does not seem to be default in UML 2.5.

<sup>3</sup>The member ends are ordered according to the UML Superstructure Specification 2.4.1, p. 29; UML 2.5, p. 206; hence they are represented in a tuple-like notation.

<sup>4</sup>Only binary association may show member ends that are properties not owned by the association (UML Superstructure Specification 2.4.1, p. 37; UML 2.5, p. 228). The property declarations induced by a more than binary association result in a query operation.

<sup>5</sup>In UML, member end names need not be unique. However, for (1) a simpler handling of selecting a particular member end in the sentences and avoiding the use of number selectors, and (2) making the notion of member ends “owned” by a class/data type, this constraint is added. An association declaration violating this uniqueness constraints can easily be transformed into an association declaration satisfying it by decorating member end names with the numbers  $1, \dots, r$ .