

On the Semantics of Associations and Association Ends in UML

Dragan Milicev

University of Belgrade

School of Electrical Engineering, Department of Computer Science

P.O. Box 35-54, 11120 Belgrade, Serbia, Yugoslavia

dmilicev@etf.bg.ac.yu

Abstract

Association is one of the key concepts in UML that is intensively used in conceptual modeling. Unfortunately, in spite of the fact that this concept is very old and is inherited from other successful modeling techniques, a fully unambiguous understanding of it, especially in correlation with other newer concepts connected with association ends, such as uniqueness, still does not exist. This paper describes a problem with one widely assumed interpretation of the uniqueness of association ends – the restrictive interpretation, and proposes an alternative – the intentional interpretation. Instead of restricting the association from having duplicate links, uniqueness of an association end in the intentional interpretation modifies the way in which the association end maps an object of the opposite class to a collection of objects of the class at that association end. If the association end is unique, the collection is a set obtained by projecting the collection of all linked objects. In that sense, the uniqueness of an association end modifies the view to the objects at that end, but does not constrain the underlying object structure. The paper demonstrates how intentional interpretation improves expressiveness of the modeling language and has some other interesting advantages. Finally, the paper gives a completely formal definition of the concepts of association and association ends, along with the related notions of uniqueness, ordering, and multiplicity. The semantics of the UML actions on associations are also defined formally.

Keywords

Object-oriented modeling, Unified Modeling Language (UML), association, association end, formal semantics, conceptual modeling, model-driven development

1 INTRODUCTION

For about a decade, the Unified Modeling Language (UML) has been widely recognized as a standardized, general-purpose, object-oriented language for specifying, visualizing, constructing, and documenting software systems [2]. Its applicability to different domains of software applications, as well as its expressiveness and richness of classical and novel object-oriented concepts, have significantly contributed to its wide adoption in industry and in academy.

UML, in particular, its parts aimed at structural modeling, have been especially intensively used for conceptual modeling, most notably in business and database applications. This fact is not surprising when it is noticed that the central language constructs used for conceptual modeling in UML – class, attribute, association, and generalization/specialization – have been basically inherited from much older modeling techniques, such as Entity-Relationship modeling [4] or Object Modeling Technique [13], which have been also extensively and successfully used for conceptual modeling for a long time.

Although standardized in 1997 and widely used in practice ever since, UML's main drawback is still the lack of a complete semantic formalization. Many concepts of UML, especially in its early

versions, have not had definitions precise enough to be interpreted unambiguously. In other words, many parts of a syntactically correct UML model can still be interpreted in different ways by different readers. This has been recognized as the main obstacle for UML to become a machine-interpretable, i.e., executable language. Instead, UML served predominantly for recording ideas, sketches, and design decisions in the early phases of software analysis and design.

The recent model-driven development (MDD) trends in software engineering [16, 15] have dramatically increased the importance of formalizing the semantics of UML, as the key prerequisite for its switch from a solely descriptive to an unambiguously interpretable and executable language. As a result, the major revision of UML has emerged in its version 2.0 [10], making a significant step towards precise semantics of concepts [14]. Much effort have been made to clarify the meaning of the widely adopted concepts in the very UML 2.0 specification [10, 14], as well as in other attempts before and around it [1, 3, 5, 6, 7, 8, 11, 12, 18].

Being one of the central structural concepts in UML, the concept of association is not an exception to this rule. Since it has been derived from the very old and intuitive notion of a relationship between entities in Entity-Relationship (ER) modeling [4], its basic meaning appears to be fairly clear, at least in its intention. Essentially, an association is a structural relationship between classes, which represents a set of links; links, as instances of association, are structural connections among objects, as instances of classes. Put another way, while a class (or entity in ER) represents a set of objects as its instances, an association (or relationship in ER) is a relationship among these sets; therefore, the object structure in a running system can be represented with a graph, whereby nodes are objects (as instances of classes) and edges are links (as instances of associations). The graph is typed, because its nodes (objects) are typed by their classes, and its edges (links) are typed by their associations.

In UML, this essentially simple concept has been significantly enhanced to increase expressiveness of the language. Unfortunately, these enhancements have introduced many new ambiguities in the interpretation of the entire concept. This is why many attempts have aimed at clarifying [5, 7, 18] and formalizing the semantics of associations [3, 6, 11, 12] and some of the related concepts, like aggregation [1] and multiplicity [8]. However, many issues regarding the specification of UML 2.0 [10] remain open, and many parts in the specification can still be interpreted in different ways.

One of such issues is related with the notion of *uniqueness* of association ends [10]. This paper demonstrates how it can be interpreted in a straightforward and apparently widely assumed way [9], which, however, imposes unnecessarily strong restrictions in modeling and thus reduces expressiveness of the language. The paper then proposes an alternative interpretation of this concept with relaxed modeling rules. The proposed interpretation appears to be more expressive and better aligned to the current definition of the action semantics in UML. The paper argues for these and some other advantages of the proposed interpretation. As its second goal, the paper presents a completely formal definition of the semantics of the concept of association, along with the accompanying concepts of multiplicity, uniqueness, and ordering of the association ends, as well as the definition of the semantics of UML actions in correlation with these concepts. The paper does this for a general case of N-ary associations.

The rest of the paper is organized as follows. Section 2 describes the considered problem of semantic interpretation of uniqueness of association ends and describes the straightforward interpretation and its drawbacks, as the motivation for the alternative interpretation proposed in this paper. Section 3 defines the proposed interpretation of associations (including association classes) and their ends, along with multiplicity, uniqueness, and ordering of association ends. Section 3 presents the arguments for accepting the proposed approach. Section 5 briefly summarizes the related work. The paper ends with conclusions.

2 MOTIVATION

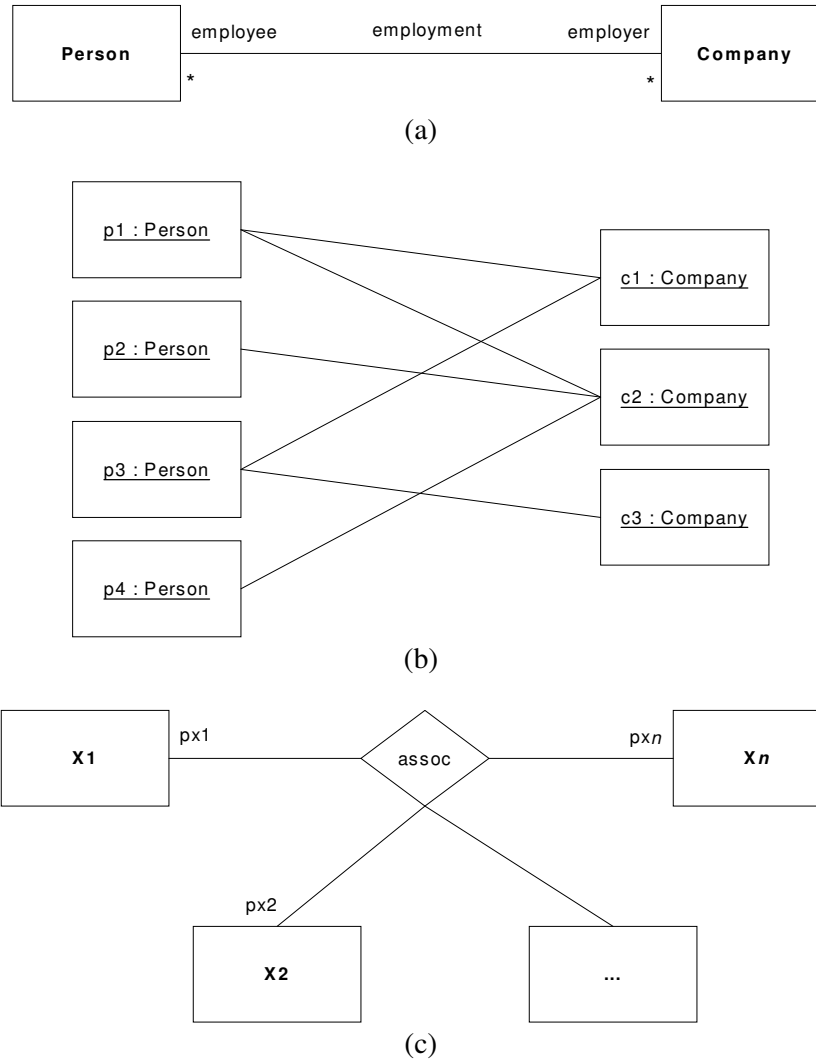


Figure 1: A basic example of associations in UML: (a) a binary association *employment*, relating two classes; (b) an object diagram depicting a sample object structure for the class model in figure (a); (c) a general case of an N-ary association relating classes X_1 , X_2 , ..., X_n .

According to the UML 2.0 specification [10], "an association specifies a semantic relationship that can occur between typed instances; it has at least two ends represented by properties, each of which is connected to the type of the end; more than one end of the association may have the same type... An association declares that there can be links between instances of the associated types. A link is a tuple with one value for each end of the association, where each value is an instance of the type of the end" ([10], pp. 36-37). For example, the binary association (i.e., association with two ends) named *employment* in Figure 1a defines a semantic relationship between the classes *Person* and *Company*, describing a set of links that can connect instances of these classes, representing the conceptual fact that a certain person (as an instance of *Person*) is an *employee* of a certain company (as an instance of *Company*), which plays the role of the *employer* in this relationship. Figure 1b shows an object diagram that represents a sample object structure with objects (depicted as rectangles) and links (depicted as lines) connected according to this class model. In that structure, the object *p1* of the class *Person* is linked to the objects *c1* and *c2* of the class *Company*, meaning that *p1* is employed by *c1* and *c2*, etc. In

total, there are six links of the association *employment*, which are pairs of objects they relate: $(p1, c1)$, $(p1, c2)$, $(p2, c2)$, $(p3, c1)$, $(p3, c3)$, and $(p4, c2)$. Deriving the set of companies in which a person p is employed from these pairs is straightforward: select the objects of *Company*, i.e., the second coordinates, of all those and only those pairs (links) that have p as the first coordinate. For example, $p3$ is employed by $c1$ and $c3$, $p4$ by $c2$, etc. The similar holds for the opposite direction.

Figure 1c shows a general case of an N-ary association having any finite number $N > 1$ of ends. In that case, every link has also N ends (it can be imagined as a "star" rather than a line), each of which is connected to an object of the appropriate class.

The UML 2.0 specification then continues with the description of the modifiers of each end of an association: "When one or more ends of the association have *isUnique=false* [are specified as *unique*], it is possible to have several links associating the same set of instances. In such a case, links carry an additional identifier apart from their end values. When one or more ends of the association are ordered, links carry ordering information in addition to their end values. For an association with N ends, choose any $N-1$ ends and associate specific instances with those ends. Then the collection¹ of links of the association that refer to these specific instances will identify a collection of instances at the other end. The multiplicity of the association end constrains the size of this collection. If the end is marked as ordered, this collection will be ordered. If the end is marked as unique, this collection is a set; otherwise it allows duplicate elements" ([10], pg. 37).

A straightforward (and seemingly widely assumed) interpretation of this specification is the following. If an association end, for example, the end *employee* in Figure 1a, is specified as *unique*, then the collection of objects of the class *Person* to which an object of the class *Company* is linked by links of the association *employment* must not contain duplicates, i.e., is a set. Since this collection is obtained as the collection of the first coordinates of the existing links (p, c) , where c is the considered object of *Company*, there must not be two pairs $(p1, c)$ and $(p2, c)$ such that $p1=p2$; this must hold for every c . Consequently, there must not be two equal links in the association, i.e., two pairs (or tuples in a general case) having all the same values. In other words, if one end of an association is unique, the association is a set, i.e., must not contain duplicate links. Then it simply follows that the other end is also unique [9].

To sum up, this straightforward interpretation implies that the uniqueness of an association end imposes a *constraint* to the association itself, because it prohibits having (creating) duplicate links of the association. For that reason, we will refer to this semantic interpretation as the *restrictive interpretation*. Its consequence is that an association should have all its ends of the same kind (unique or non-unique), because there is senseless to have one unique and one non-unique end of the same association [9]. We name this rule as the *uniqueness symmetry (meta)constraint*.²

The problems with the restrictive interpretation and its consequence – the uniqueness symmetry meta-constraint – are manifold:

- The uniqueness symmetry meta-constraint does not exist in the UML 2.0 specification. Consequently, a UML model with an association having both unique and non-unique ends is a

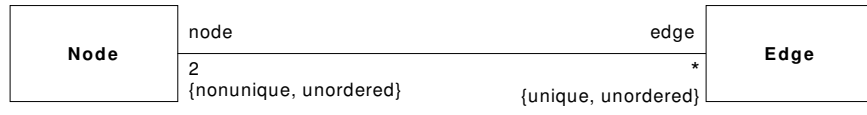
¹ In UML, the term "collection" denotes a generalized concept of several specialized kinds of value containers. When a collection is *non-unique* and *unordered* (default), it represents a multiset (a set that allows multiple occurrences of the same value, with irrelevant ordering of its elements, i.e., a bag); when it is *unique* and *unordered*, it represents a set (having no duplicates, with irrelevant ordering of elements); when it is *non-unique* and *ordered*, it represents a sequence (an ordered multiset, i.e., a multiset with some ordering of elements); finally, when it is *unique* and *ordered*, it represents an ordered set (a set without duplicates and with some ordering of elements).

² It is a meta-constraint, because it is a constraint imposed on the model, not on the object space. In other words, it is a constraint within the UML meta-model.

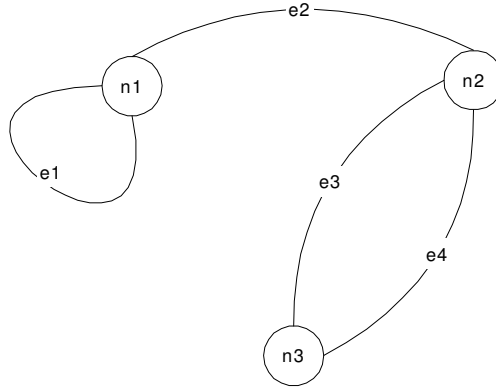
legal model, at least according to that specification, although it is not meaningful according to the restrictive interpretation.

- If the uniqueness symmetry meta-constraint were added to the UML 2.0 specification, it would impose additional restriction to the modeling: asymmetric associations with regards of uniqueness would become illegal.
- If the uniqueness symmetry meta-constraint were not added to the UML 2.0 specification, the specification itself would remain contradictory in a very subtle detail. Namely, the specification of the abstract meta-class *StructuralFeatureAction* ([10], pp. 305-306), which represents a generalization of all actions on structural features of a class, says: "If the structural feature is an association end, then actions on the feature have the same semantics as actions on the links that have the feature as an end." For example, for the Add Structural Feature Value Action ([10], pp. 254-255), it is explained that "if the feature is an association end, the semantics are the same as creating a link, the participants of which are the object owning the structural feature and the new value." Therefore, adding an already existing value to a non-unique association end over the Add Structural Feature Value Action should behave as if the feature were not an association end; that is, the value should be added to the feature as a duplicate. However, if the opposite association end is unique, such action would add an already existing link and thus violate the uniqueness of that end.
- Since at least one unique association end implies uniqueness of the entire association, the uniqueness meta-property is conceptually more appropriate for the association, not for every of its ends.
- As it will be demonstrated soon, the uniqueness symmetry meta-constraint reduces expressiveness of the modeling language.

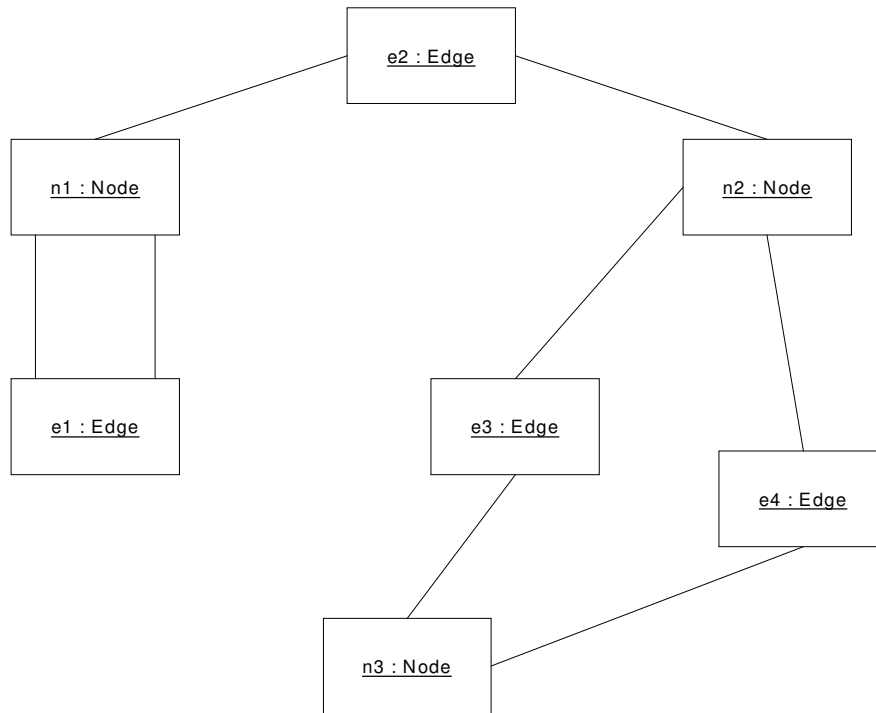
Fortunately, there is a solution to all these problems, because another semantic interpretation exists and is proposed in this paper. It fits completely in the current UML 2.0 specification, yet overcomes the listed deficiencies (unfortunately, this implies that the specification is ambiguous). The alternative interpretation does not make any difference in specifying *what kind of collection* is obtained for a considered association end by choosing some objects for the other ends and finding all links that have these objects as the corresponding values: if the considered end is unique, the collection is a set, otherwise it is a bag. Instead, the proposed interpretation treats differently *how* that collection is obtained from the links of the association. In that interpretation, the uniqueness of an association end is not a constraint on the association, but is construed as an *intention* of the modeler that designates how the collection at that end should be derived from the links. This is why we refer to this interpretation as the *intentional interpretation*.



(a)



(b)



(c)

Figure 2: An example of asymmetric association with regard of uniqueness: modeling a multigraph of nodes and edges. (a) The UML class model. (b) An example of a multigraph with three nodes ($n1$ to $n3$) and four edges ($e1$ to $e4$). (c) The UML object diagram rendering the object structure that corresponds to the graph example in (b) and the class model in (a); all links are of the same association.

The idea of the intentional interpretation is illustrated in Figure 2, using an example of modeling graphs with multiple edges allowed between nodes (i.e., multigraphs). Figure 2a depicts the UML class model that conceptualizes the notion of a multigraph. Two classes, *Node* and *Edge* are related with an association the links of which anchor edges of a graph to their nodes. Since an edge is anchored to nodes at exactly two its ends, but can be anchored to the same node at both ends, as shown

in Figure 2b, the association end *node* in Figure 1a is specified as non-unique (allows duplicates) and has the multiplicity of exactly two.

It is obvious then that the UML model in this case should allow multiple links between the same two objects. Figure 2b shows an example of a graph with three nodes (*n1* to *n3*) and four edges (*e1* to *e4*), while Figure 2c depicts the UML object diagram representing the object structure that corresponds to the graph example in Figure 2b and the class model in Figure 2a. The edge *e1* is anchored to the node *n1* at both its ends. This is why there are two links between *n1* and *e1*, i.e., two occurrences of the pair (*n1*, *e1*). Consequently, the association is a bag (multi-set) of links, not a set. Using the Z notation [17] for bags (elements enclosed with $\llbracket \rrbracket$), the association is the bag:

$\llbracket (n1, e1), (n1, e1), (n1, e2), (n2, e2), (n2, e3), (n2, e4), (n3, e3), (n3, e4) \rrbracket$

On the other side, the association end *edge* is specified as unique. Opposed to the restrictive interpretation, this does not restrict the existence of multiple occurrences of the same pair in the association, but determines *how* the set of objects of *Edge* to which the association end *edge* results for a certain object *n* of *Node* is obtained: it is obtained by projecting (or collapsing) the collection of all second coordinates *e* of the links (*n*, *e*) of the association to a set by removing duplicates. Using the dot notation for this collection, where e.g. *n1.edge* is the set of objects of *Edge* to which the end *edge* for *n1* results, it follows:

n1.edge = {*e1*, *e2*} – the set of *es* from all pairs (*n*, *e*) where *n* = *n1* (duplicates removed);

n2.edge = {*e2*, *e3*, *e4*} – the set of *es* from all pairs (*n*, *e*) where *n* = *n2* (duplicates removed);

n3.edge = {*e3*, *e4*} – the set of *es* from all pairs (*n*, *e*) where *n* = *n3* (duplicates removed).

The opposite association end results in non-unique collections, i.e., bags (multi-sets), and are:

e1.node = $\llbracket n1, n1 \rrbracket$ – the collection (bag) of all occurrences of *n* in pairs (*n*, *e*) where *e* = *e1*;

e2.node = $\llbracket n1, n2 \rrbracket$ – the collection (bag) of all occurrences of *n* in pairs (*n*, *e*) where *e* = *e2*;

e3.node = $\llbracket n2, n3 \rrbracket$ – the collection (bag) of all occurrences of *n* in pairs (*n*, *e*) where *e* = *e3*;

e4.node = $\llbracket n2, n3 \rrbracket$ – the collection (bag) of all occurrences of *n* in pairs (*n*, *e*) where *e* = *e4*.

It should be noted that the mentioned resulting collections (denoted with *n.edge* and *e.node*) can be obtained only by the appropriate actions that read links (Read Link Action or Read Structural Feature Action [10]) – any access to the underlying object structure is performed through UML actions. In other words, the only manifestation of the considered semantics of association and association ends are the effects of actions upon associations. Therefore, the proposed intentional interpretation affects actually the semantics of actions upon associations, not the very concept of association. Conceptually, the uniqueness of the association end *edge* affects the semantics of the Read Link Action or Read Structural Feature Action (denoted here with *n.edge*) so that these actions return "the set of all (distinct) edges coinciding with the given node *n*." For the opposite non-unique end *node*, these actions return "the collection of (all occurrences of) nodes to which the given edge *e* is anchored;" the multiplicity constraint 2 at that end constrains the size of that collection to exactly two.

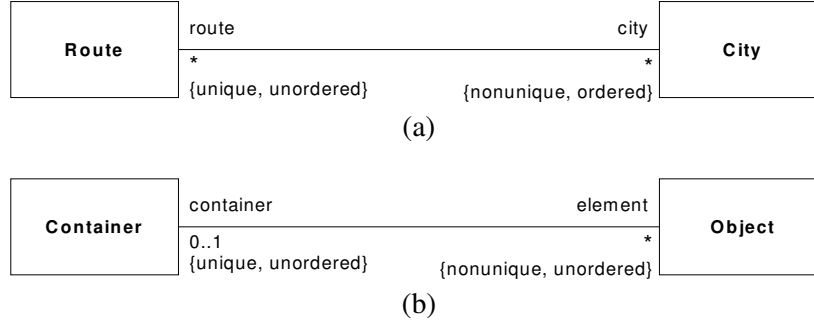


Figure 3: More examples of asymmetric uniqueness of association ends. (a) A *Route* for tourists is an ordered sequence of visited *Cities*, possibly visiting the same city several times. (b) A generic case of a *Container* that can contain a collection of *Objects*, possibly containing the same *Object* more than once, while an *Object* can be contained by at most one *Container*.

Another example is shown in Figure 3a. A *Route* for tourists is defined by an ordered sequence of the *Cities* it visits, possibly visiting the same city more than once. This is why the association end *city* is non-unique and ordered. The opposite association end *route* is marked as unique, conceptualizing the fact that `c.route` for a city *c* results in the set of (different) routes that visit the given city *c*. For example, if the route "Balkans Tour" visits the cities: Belgrade, Sarajevo, Athens, Rhodes, Istanbul, Sofia, Bucharest, and Belgrade in that order, while the route "Eastern Europe Tour" visits the cities: Prague, Budapest, Belgrade, Bucharest, Warsaw, and Moscow in that order, than the set of routes that visit Belgrade has two elements (these two routes) and is obtained by `b.route`, where *b* denotes Belgrade as an object of *City*.

Our final example in Figure 3b demonstrates a more serious expressiveness deficiency of the restrictive interpretation, while the intentional interpretation provides a simple solution. As the example is a generic one, having many concrete manifestations in specific domains, it demonstrates how the intentional interpretation ensures stronger expressiveness of the modeling language. In this generic case, a *Container* can contain a collection of *Objects* as its elements, possibly containing the same *Object* more than once, while an *Object* can be contained by at most one *Container*. Assuming the intentional interpretation, since the association end *container* is designated as unique, the upper multiplicity bound equal to one constrains the number of elements in the set of (distinct) Containers that contain a certain *Object*. The opposite end *element* is designated as non-unique and thus allows multiple occurrences of the same *Object* as elements of the *Container*. Note how the same example would behave if the restrictive interpretation were assumed: as the *element* end must be non-unique, the opposite end *container* should also be non-unique; however, the number of elements in the collection `obj.container` for a certain *Object* *obj* would then depend on the number of occurrences of that very *Object* in the *Container*, and would not allow limiting that number to one. Letting the upper multiplicity bound of that end be unlimited, however, would allow erroneous structures in which the same *Object* is an element of more than one different *Container*. A separate user-defined constraint must be specified to resolve this problem then.

The previous example illustrates one general deficiency of the restrictive interpretation. In any case when an association end *a* must be non-unique due to some domain-specific reasons, but the multiplicity constraint at the opposite end *b* should limit the number of different objects to which an object of the class at the side *a* can be related, the restrictive interpretation fails. For the example in Figure 3b, if the number of different *Routes* by which a *City* can be visited should be limited by any reason whatsoever, e.g., to at most one or to a certain number, but the same *City* can still be re-visited by the same *Route* a finite but unlimited number of times, the restrictive interpretation cannot provide a direct solution. Moreover, all three given examples are rather generic and can be specialized in many different domain-specific and isomorphic cases. For example, any specific conceptual domain that can

be modeled with a multigraph shown in Figure 2 or with the Container-Object pattern shown in Figure 3b, may raise the same issues discussed here. This concludes our motivation for adopting and formalizing the proposed intentional interpretation.

3 SEMANTICS



Figure 4: Sample model for describing the semantics of binary associations and association ends.

This section presents an informal, yet precise and unambiguous explanation of the semantics of association and association ends, along with the related concepts of multiplicity, ordering, and association classes. Fully formal definitions in the Z language [17] are given in the Appendix. For the sake of simplicity of explanations, only binary associations will be considered in this section. The generalization for N-ary associations is straightforward and is given in the Appendix.

The explanations will be based on a generic association ass_{xy} between classes x and y shown in Figure 4. It is assumed that the association end px is the first, and the association end py is the second in the order of ends of the association ass_{xy} (association ends are ordered within an association and the small filled triangle in diagrams points to the last end in the order). In these discussions, it is irrelevant whether an association end belongs to the opposite class or to the association [10] – the only difference is in whether the opposite class does or does not have the corresponding feature. The basic semantics of association and its ends is independent of this aspect.

Associations and Association Ends

An association describes a collection of links, whereby a link is an ordered pair the values of which refer to objects. For the considered sample model in Figure 4, the association ass_{xy} describes a collection of links in the form (x, y) , whereby x is an object of x and y is an object of y . This collection is not a set, i.e., it allows duplicates of pairs. Besides, it is never ordered; consequently, an association is a bag. For example, if the subscripts of x and y indicate different objects of x and y , respectively, at a certain point t_1 in time, the association ass_{xy} may be the following collection (the ordering of the pairs in the collection is not relevant):

$$ass_{xy_{t_1}} = \llbracket (x_1, y_1), (x_1, y_2), (x_1, y_3), (x_2, y_1), (x_2, y_3), (x_2, y_1), (x_3, y_2) \rrbracket$$

Just for the purpose of referencing, the collections will be denoted with $ass_{xy_{t_i}}$, with the meaning "the collection denoted with ass_{xy} at the time t_i ."

The Create Link action executed for this association simply adds a new pair to this collection. For example, if a Create Link action with the parameters (x_1, y_2) is applied twice to this association $ass_{xy_{t_1}}$, the collection becomes:

$$ass_{xy_{t_2}} = \llbracket (x_1, y_1), (x_1, y_2), (x_1, y_3), (x_2, y_1), (x_2, y_3), (x_2, y_1), (x_3, y_2), (x_1, y_2), (x_1, y_2) \rrbracket$$

An association end defines a mapping of each object of the class at the opposite end to a collection of objects of the class at that end. If that association end is non-unique, that collection represents the corresponding values from all those and only those pairs from the association that have

the given object as the corresponding coordinate.³ For the given example, supposing that p_Y is non-unique, for an object x of X , the mapping $p_Y(x)$ is derived from the association to represent the collection of the second coordinates y of all those and only those pairs from ass_{xy} that have x as the first coordinate. The similar holds for non-unique p_X . Therefore, at the time t_1 (the ordering of collection elements is irrelevant):

$$p_Y(x_1)_{t_1} = \llbracket y_1, y_2, y_3 \rrbracket$$

$$p_Y(x_2)_{t_1} = \llbracket y_1, y_3, y_1 \rrbracket$$

$$p_Y(x_3)_{t_1} = \llbracket y_2 \rrbracket$$

$$p_X(y_1)_{t_1} = \llbracket x_1, x_2, x_2 \rrbracket$$

$$p_X(y_2)_{t_1} = \llbracket x_1, x_3 \rrbracket$$

$$p_X(y_3)_{t_1} = \llbracket x_1, x_2 \rrbracket$$

Similarly, at the time t_2 (the ordering of collection elements is irrelevant):

$$p_Y(x_1)_{t_2} = \llbracket y_1, y_2, y_3, y_2, y_2 \rrbracket$$

$$p_Y(x_2)_{t_2} = \llbracket y_1, y_3, y_1 \rrbracket$$

$$p_Y(x_3)_{t_2} = \llbracket y_2 \rrbracket$$

$$p_X(y_1)_{t_2} = \llbracket x_1, x_2, x_2 \rrbracket$$

$$p_X(y_2)_{t_2} = \llbracket x_1, x_3, x_1, x_1 \rrbracket$$

$$p_X(y_3)_{t_2} = \llbracket x_1, x_2 \rrbracket$$

Note that if an association end belongs to the opposite class as its feature, it still represents the same collection defined by the association end and thus has the same semantics. For example, if p_Y belongs to X , then $x.p_Y$ at a certain time represents the same collection defined by the mapping $p_Y(x)$ at that time:⁴

$$x_1.p_{Yt_1} = p_Y(x_1)_{t_1} = \llbracket y_1, y_2, y_3 \rrbracket$$

$$x_2.p_{Yt_1} = p_Y(x_2)_{t_1} = \llbracket y_1, y_3, y_1 \rrbracket$$

$$x_3.p_{Yt_1} = p_Y(x_3)_{t_1} = \llbracket y_2 \rrbracket$$

The Destroy Link action specifies a link, i.e., a pair of values to be removed from the association. For each of the non-unique association ends of the association, it can be specified whether all values from the collection defined by that end should be removed (according to the value of the *isRemoveDuplicates* meta-attribute of the Remove Structural Feature Value Action [10], pp. 297-298). Let us use the symbol $\forall$ as a prefix of a value in a pair to indicate that all pairs having that value at that coordinate should be removed; if this symbol is missing, only one such pair should be removed. For example:

- $\text{DestroyLink}(ass_{xy}, (x_1, y_2))$ means "remove one pair (x_1, y_2) from ass_{xy} ."

³ Strictly speaking, this mapping is the definition of the semantics of the Read Link Action and Read Structural Feature Action applied to the association end as a property of the class, as already discussed, because the effect of the actions is the only manifestation of an association end's semantics.

⁴ This observation actually means that the semantics of the Read Link Action and Read Structural Feature Action applied to the association end as a property of the class are the same.

- $\text{DestroyLink}(\text{assxy}, (\forall x_1, y_2))$ means "remove those and only those pairs (x_1, y_2) from assxy , such that $\text{px}(y_2)$ does not contain x_1 any more;" it is easy to see that this means "remove all pairs (x_1, y_2) from assxy ."
- $\text{DestroyLink}(\text{assxy}, (x_1, \forall y_2))$ means "remove those and only those pairs (x_1, y_2) from assxy , such that $\text{py}(x_1)$ does not contain y_2 any more;" it is easy to see that this means "remove all pairs (x_1, y_2) from assxy ." Consequently, $\text{DestroyLink}(\text{assxy}, (\forall x_1, y_2))$, $\text{DestroyLink}(\text{assxy}, (x_1, \forall y_2))$, and $\text{DestroyLink}(\text{assxy}, (\forall x_1, \forall y_2))$ are all equivalent.

If an association end is designated as unique, then for each object of the opposite class, it defines a sub-collection of the entire collection of coordinates, such that the duplicate values are reduced to a single one; in other words, it is a projection of the entire collection to a set. For the considered example, if py is unique, then:

```

py(x1)t1 = {y1, y2, y3}
py(x2)t1 = {y1, y3}
py(x3)t1 = {y2}
py(x1)t2 = {y1, y2, y3}
py(x2)t2 = {y1, y3}
py(x3)t2 = {y2}

```

Similarly, if px is unique, then:

```

px(y1)t1 = {x1, x2}
px(y2)t1 = {x1, x3}
px(y3)t1 = {x1, x2}
px(y1)t2 = {x1, x2}
px(y2)t2 = {x1, x3}
px(y3)t2 = {x1, x2}

```

As it can be noticed, uniqueness is not a characteristic of the association, but of its end(s). Uniqueness defines the way the collection of ordered pairs that constitute the association is projected on its end for each object of the opposite class. If the end is non-unique, the projection is one-to-one correspondent to the original; otherwise, the projection defines a sub-collection.

If an association end is unique, the Destroy Link action always removes the given value from the collection of that end totally. Precisely, if py is unique, then $\text{DestroyLink}(\text{assxy}, (x, y))$ means "remove those and only those pairs (x, y) from assxy , such that $\text{py}(x)$ does not contain y any more;" as already shown, this simply means "remove all pairs (x, y) from assxy ," denoted as $\text{DestroyLink}(\text{assxy}, (\forall x, \forall y))$.

It is interesting to note that association ends (or more precisely, the actions upon them), as projections of their association, are the only possibility to view the association, i.e., the only manifestation of links, because there are actions in UML that read the collections defined by the association ends, but not by the association per se. The collection of the association can be just inferred from its projections, not explicitly retrieved. There is no action in UML that would return the entire collection assxy , but only the action(s) that would return the collections $\text{py}(x)$ or $\text{px}(y)$ – these are Read Link Action and Read Structural Feature Action.

As an interesting implication of such semantics, an association with both its ends specified as unique actually behaves like a set. Really, since both ends are unique, adding a new pair (x, y) that already exists in the association does not affect the collections of any of its ends, i.e., does not affect the association's projections. The Destroy Link action, as described, has always the semantics of removing all pairs with the given values (x, y) from the association. Since the collections of the ends, as projections of the association, are the only available views to the association, exactly the same projections would be obtained by treating the association as a set of pairs (without duplicates) instead of as a bag (allowing duplicates) with the described semantics of actions. For example, if the association assxy is at a certain moment in time:

$\llbracket (x_1, y_1), (x_1, y_2), (x_1, y_3), (x_2, y_1), (x_2, y_3), (x_3, y_2) \rrbracket$

and $\text{CreateLink}(x_1, y_2)$ is executed, then the association becomes, if it is treated as a bag:

$\llbracket (x_1, y_1), (x_1, y_2), (x_1, y_3), (x_2, y_1), (x_2, y_3), (x_3, y_2), (x_1, y_2) \rrbracket$

or remains the same, if it is treated as a set:

$\llbracket (x_1, y_1), (x_1, y_2), (x_1, y_3), (x_2, y_1), (x_2, y_3), (x_3, y_2) \rrbracket$

The $\text{DestroyLink}(x_1, y_2)$ action has the same effect on both of these and produces the following:

$\llbracket (x_1, y_1), (x_1, y_3), (x_2, y_1), (x_2, y_3), (x_3, y_2) \rrbracket$

As it can be concluded, if (and only if) all ends of an association are unique, then the association itself is a set, meaning that it does not have duplicate tuples. It is important to emphasize that this conclusion does not have any implications on the outwardly visible behavior and interpretation of associations and their ends, but is just important for an implementation. Namely, an implementation may rely on this conclusion to make the storage and manipulation of associations with all unique ends, which are most frequent in practice, more efficient than of other associations.

Just as another illustration, let us suppose that the association end p_y is unique, while the end p_x is non-unique. If the association ass_{xy} at a certain time is:

$\text{ass}_{xy_{t_2}} = \llbracket (x_1, y_1), (x_1, y_2), (x_1, y_3), (x_2, y_1), (x_2, y_3), (x_2, y_1), (x_3, y_2), (x_1, y_2), (x_1, y_2) \rrbracket$

then the association ends define the following projections:

$p_y(x_1)_{t_2} = \{y_1, y_2, y_3\}$

$p_y(x_2)_{t_2} = \{y_1, y_3\}$

$p_y(x_3)_{t_2} = \{y_2\}$

$p_x(y_1)_{t_2} = \llbracket x_1, x_2, x_2 \rrbracket$

$p_x(y_2)_{t_2} = \llbracket x_1, x_3, x_1, x_1 \rrbracket$

$p_x(y_3)_{t_2} = \llbracket x_1, x_2 \rrbracket$

Multiplicity

The multiplicity of an association end is an implicit constraint attached to that end, which constrains the cardinality of (i.e., the number of elements in) every collection specified by that end. Therefore, the multiplicity is not a constraint on the very association, but on the cardinality of its projections only. For example, let ass_{xy} at the time t_2 be:

$\text{ass}_{xy_{t_2}} = \llbracket (x_1, y_1), (x_1, y_2), (x_1, y_3), (x_2, y_1), (x_2, y_3), (x_2, y_1), (x_3, y_2), (x_1, y_2), (x_1, y_2) \rrbracket$

If the association end p_y is non-unique, then the cardinalities of collections (denoted here with #) are:

$\#(p_y(x_1))_{t_2} = 5$

$\#(p_y(x_2))_{t_2} = 3$

$\#(p_y(x_3))_{t_2} = 1$

If the association end p_y is unique, then the cardinalities are:

$\#(p_y(x_1))_{t_2} = 3$

$\#(p_y(x_2))_{t_2} = 2$

$\#(p_y(x_3))_{t_2} = 1$

Ordering

Ordering is yet another characteristic of the association end, and not of the association itself. Ordering of the association end p_Y means that for each x of X , $p_Y(x)$ is an ordered collection, i.e., a sequence. More formally, there is a one-to-one mapping from $p_Y(x)$ to a range of successive positive integer numbers $1..n$, where $n = \#(p_Y(x))$; the integer assigned to a collection element will be referred to as its *index*. From now on, if an association end is ordered, we will consider that the elements of its collection are written in that order and enclosed in symbols $\langle \rangle$, and thus the indices are implied from the position of the element in the sequence. The question now is how ordering works with Create Link and Destroy Link actions that affect the entire association.

Let us first consider a non-unique ordered association end p_Y . Since $p_Y(x)$ has all those and only those values y for which there is a pair (x, y) in ass_{xy} , an index can be assigned to the y coordinate of every pair in ass_{xy} . The index will be shown in square brackets aside the value in a pair. For example, let ass_{xy} at the time t_2 be:

$$ass_{xy_{t_2}} = \llbracket (x_1, y_1[4]), (x_1, y_2[1]), (x_1, y_3[2]), (x_2, y_1[2]), (x_2, y_3[3]), (x_2, y_1[1]), (x_3, y_2[1]), (x_1, y_2[5]), (x_1, y_2[3]) \rrbracket$$

The non-unique ordered collections of p_Y would be (in order):

$$\begin{aligned} p_Y(x_1)_{t_2} &= \langle y_2, y_3, y_2, y_1, y_2 \rangle \\ p_Y(x_2)_{t_2} &= \langle y_1, y_1, y_3 \rangle \\ p_Y(x_3)_{t_2} &= \langle y_2 \rangle \end{aligned}$$

If the other end p_X is also non-unique and ordered, then the first coordinates x of the pairs from the association also have their indices, completely independent of the indices of the values y . For example, let ass_{xy} at the time t_2 be:

$$ass_{xy_{t_2}} = \llbracket (x_1[3], y_1[4]), (x_1[2], y_2[1]), (x_1[1], y_3[2]), (x_2[2], y_1[2]), (x_2[2], y_3[3]), (x_2[1], y_1[1]), (x_3[1], y_2[1]), (x_1[3], y_2[5]), (x_1[4], y_2[3]) \rrbracket$$

The non-unique ordered collections of p_Y would be the same as above, while the non-unique ordered collections of p_X would be (in order):

$$\begin{aligned} p_X(y_1)_{t_2} &= \langle x_2, x_2, x_1 \rangle \\ p_X(y_2)_{t_2} &= \langle x_3, x_1, x_1, x_1 \rangle \\ p_X(y_3)_{t_2} &= \langle x_1, x_2 \rangle \end{aligned}$$

It is clear that ordering is not a characteristic of the association by any means: the indices of the two values in a pair are independent, as well as the indices of the values having different opposite coordinates (e.g., indices of x s for two different y s). Therefore, an association cannot be treated as ordered, even when both its ends are ordered.

If an association end is ordered, the Create Link action must specify the index that defines the position at which the value will be inserted in the corresponding projection. When the pair is added to the association, the indices of the values from the same projection are adjusted accordingly, so that the new value is inserted at the given position. For the last example, when both ends p_X and p_Y are ordered, a Create Link action must specify indices for both coordinates. For the last given $ass_{xy_{t_2}}$ at the time t_2 , after the execution of the action $CreateLink(ass_{xy}, (x_3[1], y_3[2]))$, the association will look like follows:

$$ass_{xy_{t_3}} = \llbracket (x_1[3], y_1[4]), (x_1[2], y_2[1]), (x_1[2], y_3[2]), (x_2[2], y_1[2]), (x_2[3], y_3[3]), (x_2[1], y_1[1]), (x_3[1], y_2[1]), (x_1[3], y_2[5]), (x_1[4], y_2[3]), (x_3[1], y_3[2]) \rrbracket$$

while the ordered projections to the ends become:

$$\begin{aligned}
p_Y(x_1)_{t_3} &= \langle y_2, y_3, y_2, y_1, y_2 \rangle \\
p_Y(x_2)_{t_3} &= \langle y_1, y_1, y_3 \rangle \\
p_Y(x_3)_{t_3} &= \langle y_2, y_3 \rangle
\end{aligned}$$

$$\begin{aligned}
p_X(y_1)_{t_3} &= \langle x_2, x_2, x_1 \rangle \\
p_X(y_2)_{t_3} &= \langle x_3, x_1, x_1, x_1 \rangle \\
p_X(y_3)_{t_3} &= \langle x_3, x_1, x_2 \rangle
\end{aligned}$$

Therefore, the command `CreateLink(assxy, (x3[1], y3[2]))` simply means "add the pair (x₃[1], y₃[2]) to the association assxy," and has the effect that y₃ is added to p_Y(x₃) at the position 2, and x₃ is added to p_X(y₃) at the position 1, while x₁ and x₂ are moved one place forward in p_X(y₃) therefore.

When an association end is non-unique and ordered, the Destroy Link action must also specify the index for that association end in order to unambiguously determine which pair will be removed from the association. For the last given assxy_{t₃} at the time t₃, after the execution of the action `DestroyLink(assxy, (x3[1], y3[2]))`, the association will look exactly as it was before the same link was created:

$$\begin{aligned}
assxy_{t_4} = & \llbracket (x_1[3], y_1[4]), (x_1[2], y_2[1]), (x_1[1], y_3[2]), (x_2[2], y_1[2]), (x_2[2], y_3[3]), \\
& (x_2[1], y_1[1]), (x_3[1], y_2[1]), (x_1[3], y_2[5]), (x_1[4], y_2[3]) \rrbracket
\end{aligned}$$

Ordered unique association ends have a bit more complex interpretation. Let p_Y be unique and ordered. Since p_Y(x) in general represents a part of the collection of all the values y in the pairs (x, y) of assxy, such that duplicates are removed, only those values y that belong to p_Y(x) do have indices; other values y in the pairs (x, y) of assxy do not. For example, the association assxy at a certain moment t₅ may look like (p_X is still considered as non-unique and ordered):

$$\begin{aligned}
assxy_{t_5} = & \llbracket (x_1[3], y_1[2]), (x_1[2], y_2), (x_1[1], y_3[1]), (x_2[1], y_1), (x_2[2], y_3[2]), \\
& (x_2[2], y_1[1]), (x_3[1], y_2[1]), (x_1[3], y_2[3]) \rrbracket
\end{aligned}$$

The projections p_Y include only those values y that have indices (in order):

$$\begin{aligned}
p_Y(x_1)_{t_5} &= \langle y_3, y_1, y_2 \rangle \\
p_Y(x_2)_{t_5} &= \langle y_1, y_3 \rangle \\
p_Y(x_3)_{t_5} &= \langle y_2 \rangle
\end{aligned}$$

$$\begin{aligned}
p_X(y_1)_{t_5} &= \langle x_2, x_2, x_1 \rangle \\
p_X(y_2)_{t_5} &= \langle x_3, x_1, x_1 \rangle \\
p_X(y_3)_{t_5} &= \langle x_1, x_2 \rangle
\end{aligned}$$

It is now the question who and how determines which values y in the pairs of the entire association for a certain x have, and which ones do not have an index? The answer is – the actions Create Link and Destroy Link that affect the association (and the corresponding Structural Feature Actions that have the same semantics). Basically, these actions work on the association in a manner that has the same effect on the ordered unique association end as the actions on unique ordered attributes and variables have on attributes and variables.

The Create Link action must specify the index of the value that will be inserted in the ordered association end. Besides, if that value already exists in the unique collection, the value is moved to the given index. Such effect is achieved by the following behavior of actions. Let p_Y be unique and ordered. `CreateLink(assxy, (x, y[i]))` adds the pair (x, y[i]) to the association assxy; if assxy already contains a pair (x, y[?]), where ? denotes any index, the index of y in that pair is removed. Consequently, the y from that pair will not exist any more in p_Y(x) at its old position, but will be "moved" to the new position i as directed by the new pair (x, y[i]). Other indices in p_Y(x) are

adjusted to constitute a continuous sequence of integers with the new index and preserve the proper ordering. For the last example of $assxy_{t5}$, after the execution of the action $CreateLink(assxy, (x_1[1], y_2[1]))$, the association becomes:

$$assxy_{t6} = \llbracket (x_1[3], y_1[3]), (x_1[3], y_2), (x_1[1], y_3[2]), (x_2[1], y_1), (x_2[2], y_3[2]), \\ (x_2[2], y_1[1]), (x_3[2], y_2[1]), (x_1[4], y_2), (x_1[1], y_2[1]) \rrbracket$$

The projections py include only those values y that have indices (in order):

$$py(x_1)_{t6} = \langle y_2, y_3, y_1 \rangle$$

$$py(x_2)_{t6} = \langle y_1, y_3 \rangle$$

$$py(x_3)_{t6} = \langle y_2 \rangle$$

$$px(y_1)_{t6} = \langle x_2, x_2, x_1 \rangle$$

$$px(y_2)_{t6} = \langle x_1, x_3, x_1, x_1 \rangle$$

$$px(y_3)_{t6} = \langle x_1, x_2 \rangle$$

The Destroy Link action for an association with an ordered unique association end must have an effect on the projections of that end as any action that removes a value from an ordered unique structural feature, and vice versa. Therefore, it can either specify the index of the value or the value itself – either of these uniquely identifies the value to be removed. Let us keep assuming that py is unique and ordered and px is non-unique and ordered. Then:

- $DestroyLink(assxy, (x[i], y[j]))$ removes from the association the unique pair $(x[i], y[j])$, and adjusts the rest of the indices in $py(x)$ and $px(y)$ accordingly; it can be thus interpreted as "remove $x[i]$ from $px(y)$ and remove $y[j]$ from $py(x)$."
- $DestroyLink(assxy, (x[i], y))$ removes from the association the pairs $(x[i], y[?])$, where y has or does not have an index, and adjusts the rest of the indices in $py(x)$ and $px(y)$ accordingly; it can be thus interpreted as "remove $x[i]$ from $px(y)$." Note that $DestroyLink(assxy, (?[i], y))$ has the same semantics as $DestroyLink(assxy, (x[i], y))$, because x is uniquely identified as the value at the position i in $px(y)$.
- $DestroyLink(assxy, (x, y[j]))$ has the same semantics as $DestroyLink(assxy, (x[i], y[j]))$, because the pair $(x[i], y[j])$ is unique in the association; it can be interpreted as "remove $y[j]$ from $py(x)$." Note that $DestroyLink(assxy, (x, ?[j]))$ has the same semantics as $DestroyLink(assxy, (x, y[j]))$, because y can be uniquely identified as the value at the position j in $py(x)$; it can be thus interpreted as "remove the value from the position j from $px(y)$."

It is interesting to note that in the case when both association ends are unique and ordered, as already said, the association behaves like a set. In such a set, there will be no values without indices.

Association Classes

Association class in UML is a model element that has both association and class properties. An association class can be seen as an association that also has class properties, or as a class that also has association properties. It not only connects a set of classifiers but also defines a set of features that belong to the relationship itself and not to any of the classifiers [10].

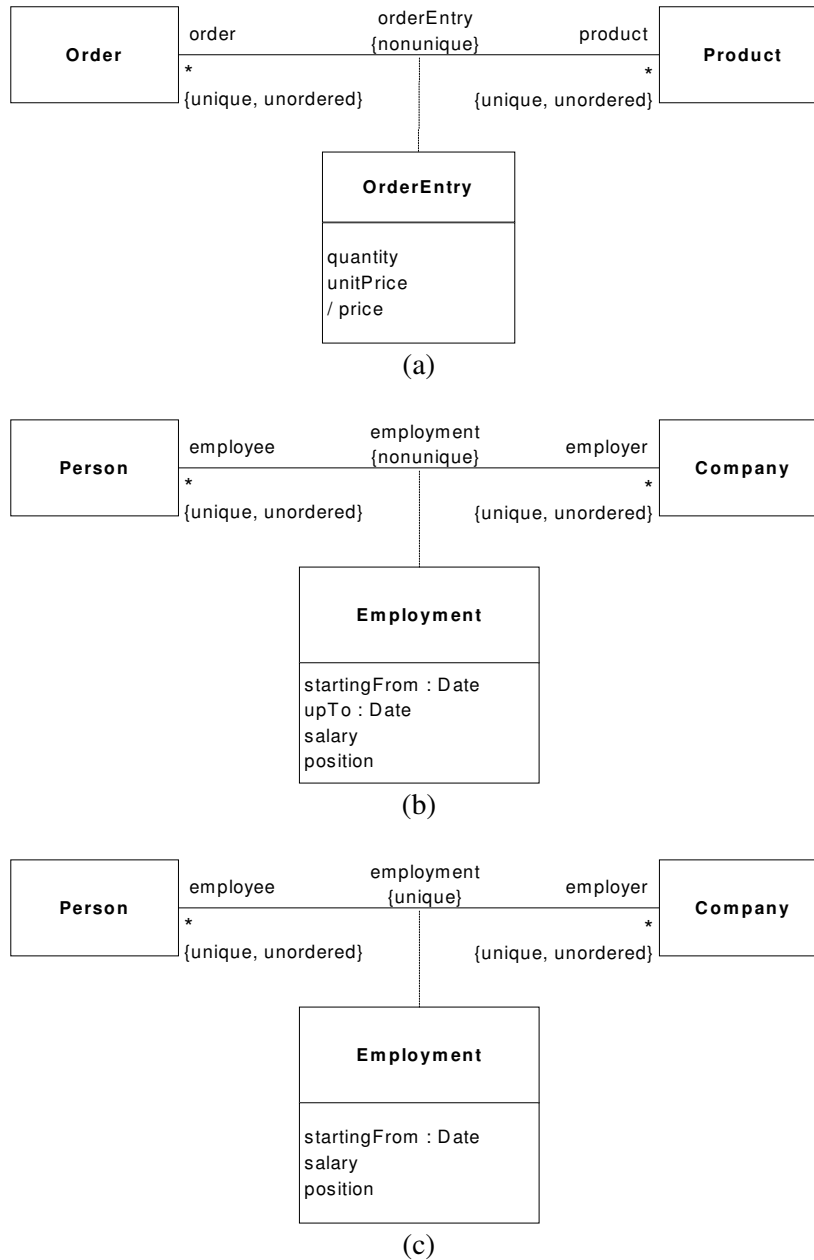


Figure 5: Examples of association classes with different and independent uniqueness of the association and its ends. (a) An excerpt from the conceptual model of an order-processing system. (b) The conceptual model for employment history tracking. (c) The conceptual model for recording the current employment of persons in companies.

Being both a kind of association and a kind of a class, all observations given so far for associations hold for association classes, too. However, a crucial difference between an instance of a pure association and an instance of an association class is that the former is a pure link without identity, while the latter is a link and an object, having its identity as an inherent characteristic.⁵ This fact opens

⁵ It could be noted that pure links do not need to have identity, even when there are multiple links relating the same set of objects. The described semantics of the association as a bag of links and the actions on it successfully handle multiple occurrences of links without requiring their identity. The actions simply add or remove (i.e., count) occurrences of links to or from the bag. The completely formal specification in the Appendix states this mathematically.

an issue specific for association classes only, the result of which is the need for the uniqueness modifier of the very association class, independent of the uniqueness of its ends. Unfortunately, such modifier does not exist in the current UML 2.0 specification. This sub-section describes the rationale for its introduction using the examples shown in Figure 5.

Figure 5a shows an excerpt from the conceptual model of an order-processing system. In that model, an *Order* is a list of *Products*, while each Product can occur several times in the list; these occurrences are referred to as *Order Entries*. For each Order Entry, the quantity, unit price, and total price of the Product referenced in that entry are provided. The association end *product* is designated as unique because the intention of the modeler is that it results in the set of (distinct) Products appearing in a certain Order. Similarly, the unique association end *order* gives the set of (distinct) Orders in which the given Product has been ordered. If any of these sets should be limited in size, the multiplicity constraint can specify that. However, the association class itself must certainly allow duplicate links (as pairs of linked objects), each link being also a separate object with its identity. Every such object carries important information, such as quantity and price. For the given object *ord* of *Order*, the collection `ord.orderEntry` gives all link-objects attached to *ord*, and thus provides the access to that information. Note that these objects must have their identity and internal state, because the system should modify that state of a particular object of interest (e.g., when setting the quantity).

A similar case is with the model for employment history tracking in Figure 5b. Each instance of the *Employment* association class represents the fact that the given *Person* used to be or is still employed in the given Company, providing other necessary information about that employment, such as period, salary, and position in the Company. Since the same Person can be re-employed in the same Company several times during different periods, this association class should allow multiple links (as pairs of linked objects). Of course, every of these links would be a separate object with its identity and its own properties that carry the information about the given period of employment. The unique association end *employer* results in the set of distinct Companies in which the given Person has been ever employed. The similar holds for the opposite end.

An interesting opposite circumstance arises when almost the same model conceptualizes a different problem domain. The association class *Employment* in the model shown in Figure 5c conceptualizes the current (instead of past) employment of a Person in a Company, carrying again the additional information about that employment. However, unlike in the previous case, the association class should not allow duplicate links, because there should not exist two different link-objects of *Employment* recording the employment of the same Person in the same Company at the present time, because it would cause clashing of information about that inherently unique employment. This is why the very association class is designated as unique, unlike in the previous cases.

As a conclusion of this brief analysis, the uniqueness modifier should be provided for association classes in UML, specifying the uniqueness of the association itself, independently of the uniqueness of its ends. This requirement can be easily met by adding a meta-attribute *isUnique* to the *AssociationClass* meta-class. If this meta-attribute is set to *true*, the association class should not allow duplicate links. Otherwise, its semantics is as described for pure associations. Note that this meta-attribute is not needed for pure associations, because their instances do not have identity and state.

4 CONSEQUENCES

The proposed intentional interpretation of association ends has several important advantages:

- It eliminates the need for adding the uniqueness symmetry meta-constraint for associations to the UML 2.0 specification. Note that the addition of that meta-constraint would cause disruption in compatibility and model compliance, because existing models with asymmetric uniqueness of association ends would become illegal.

- As already demonstrated, it improves the expressiveness of UML and makes conceptual modeling easier in many cases.
- It provides better flexibility of modeling, because models can evolve by changing the uniqueness of association ends independently, while preserving the existing object structure. Note that promotion of an association end from non-unique (less "restrictive") to unique (more "restrictive") does not affect the underlying collection of links, which does not become illegal due to existing duplicates; this promotion just affects the behavior of the system (i.e., the results of actions).
- In case when all association ends of a pure association (not association class) are unique, the association behaves (and can be implemented) exactly the same as in the restrictive interpretation.
- It fits better to the notion of association ends as properties, whereby the Read Links Action returns a collection of the specified uniqueness and ordering. Besides, the Structural Feature Actions that modify an association end as a property of a class behave as if the property were not an association end, even in case of asymmetric uniqueness.
- It is comprehensible for the people working in conceptual modeling of database applications. Namely, the difference of the semantics between a non-unique and unique association end, i.e., the Read Links Action for these kinds of ends, is completely analogous to the difference of the effects of the `SELECT` statement in SQL without or with the `DISTINCT` clause. This kind of applications is expected to benefit most from the addressed concepts of association ends and their uniqueness.
- The described concepts and actions are straightforward to implement in relational databases. Principally, an association corresponds directly to a table, each row of which represents a link. Foreign keys can implement the values identifying objects within a link, while an additional field must exist for the index of each ordered end. The Read Links Action can be implemented by the appropriate `SELECT` SQL query, having the `DISTINCT` clause if the association end is unique. The Create Link and Destroy Link actions can be implemented according to the described semantics.

However, the proposed approach has also some drawbacks in comparison with the restrictive interpretation:

- It has a bit more complicated semantics requiring somewhat more difficult implementation. However, the increase in complexity of the semantics and its implementation is not so big to disqualify the advantages.
- The notion of ordering is not completely orthogonal to the concept of uniqueness, as it appears to be.
- There is a need for an additional *isUnique* meta-attribute for association classes as the support for the new semantics of unique association classes, which is not needed for the restrictive interpretation. However, its addition does not affect the compliance of existing models to the standard, because the meta-attribute may have a default value. Its addition may only affect the semantics of an existing model if the default value of this meta-attribute is *true* (to ensure restrictive interpretation for association classes with unique ends), but all ends of an association are non-unique.

As a conclusion, the mentioned drawbacks do not seem to be so convincing to invalidate the listed advantages of the proposed approach.

5 OVERVIEW OF THE RELATED WORK

Many authors have addressed the formal semantics of associations, either in UML or in the preceding modeling languages having analogous concepts.

Stevens [18] argues for two kinds of associations, according to their conceptual nature and usage – "static," representing the structural links between objects that principally live longer than during an interaction between objects, and "dynamic," which are transient and exist only during interactions. Génova et al. [7] contribute to this understanding and propose different terminology with the "structural" and "contextual" view of associations. Barbier et al. [1] study another modifier of an association end – aggregation – and provide an in-depth analysis and explanations of its semantics. Génova et al. [8] discuss the deficiencies of the interpretation of multiplicity for N-ary associations in UML and propose solutions. The subjects of all these studies are almost independent of the subject of this paper and their results are completely orthogonal to the conclusions of this paper.

Many other works [3, 5, 6, 11, 12] propose different approaches to formalizing the semantics of associations, among other concepts. Although they use different formal methods for that, they basically agree in the way they interpret associations. However, they do not address uniqueness and ordering of association ends, and especially do not propose the intentional interpretation of uniqueness. This absence of deeper analysis and formalization of the more advanced concepts related with associations and their ends is understandable, since these concepts, most notably uniqueness, have been added recently to the newer versions of UML. In particular, the uniqueness asymmetry issue was raised very recently [9], for the last version of UML 2.0.

To the best of our knowledge, this paper is the first that proposes the alternative to the traditional restrictive interpretation – the intentional interpretation – which solves the uniqueness asymmetry issue, and also provides a complete formalization of the semantics of association ends, their uniqueness, ordering, and actions upon them.

6 CONCLUSION

This paper has studied the semantics of association as one the key concepts in UML that is intensively used in conceptual modeling. It has described the uniqueness symmetry problem with the widely assumed interpretation of the uniqueness of association ends – the restrictive interpretation, and proposed an alternative – the intentional interpretation. Instead of restricting the association from having duplicate links, uniqueness of an association end in the intentional interpretation modifies the way in which the association end maps an object of the opposite class to a collection of objects of the class at that association end. If the association end is unique, the collection is a set obtained by projecting the collection of all linked objects. In that sense, the uniqueness of an association end modifies the view to the objects at that end, but does not constrain the underlying object structure. It turns out that intentional interpretation improves expressiveness of the modeling language and has other interesting advantages discussed in the paper. Finally, the paper has given a completely formal definition of the concepts of association and association ends, along with the related notions of uniqueness, ordering, and multiplicity. The semantics of the UML actions on associations has been also defined formally.

Although this paper has covered several important issues related with associations in UML 2.0, many others, such as generalization/specialization and redefinition of associations, as well as derivation, redefinition, and subsets/unions of association ends, still remain open. The precise semantics of some of these concepts, such as specialization of associations and redefinition of ends, and especially their interference, are still very vague or explicitly left as semantic variation points in the UML 2.0 specification ([10], pg. 38). The others, such as subsets and unions, seem to be closer to formalization and the results of this paper may contribute to that. Anyway, all these issues require future research.

APPENDIX

The Appendix gives formal specifications of the proposed semantics in Z [17], for the general case of N-ary associations.

Sample Model

The sample model shown in Figure 6 will be used in this formalization. In that model, there is one association with four ends, having all four combinations of (orthogonal) unique/non-unique and ordered/unordered modifiers. The model shows how the formalization can be generalized for arbitrary N-ary associations:

- for every end of an N-ary association, the parts of each Z schema that correspond to that kind of end (regarding uniqueness and ordering) exist (or are omitted if such kind of association end does not exist); these parts may be either entire declarations/predicates or their parts; in general, the existence of the parts that correspond to one association end is independent of the other association ends;
- in some cases, some parts of specifications are not independent of other association ends; such parts are denoted with the following symbols:
 - ⌚ This part exists only when the association has no unique ends
 - ⌚ This part exists only when the association has a unique end.

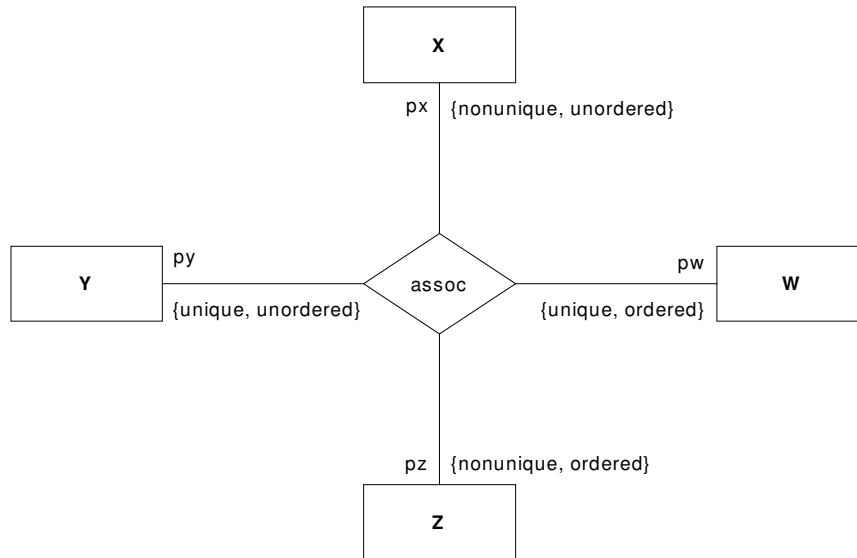


Figure 6: Sample generic model used for formalization.

Conventions and Definitions

Apart from the standard Z notation [17], the following conventions and definitions will be used in the formalizations that follow.

Informal comments start with // and last to the end of line. They do not carry semantics.

// Size of a bag - the number of elements in the bag:

[X]

$size_ : bag\ X \rightarrow \mathbb{N}$

$\forall b : bag\ X \bullet size\ b = (\mu n : \mathbb{N} \bullet (\forall s : seq\ X \mid b = items\ s \bullet n = \# s))$

// Insertion of an element x into a sequence s at the given position i , denoted with $s \downarrow \{x \mapsto i\}$:

[X]

$_ \downarrow _ : seq\ X \times \{r : X \leftrightarrow \mathbb{N}_1 \mid \# r = 1\} \rightarrow seq\ X$

$\forall s : seq\ X ; x : X ; i : \mathbb{N}_1 \bullet s \downarrow \{x \mapsto i\} = (1..(i-1) \upharpoonright s) \frown \langle x \rangle \frown (i.. \# s \upharpoonright s)$

Formal Semantics

// Classes as sets (object spaces):

[X, Y, Z, W]

// The extents of the classes – the (finite) sets of all live objects of the classes:

Classes

$extX : \mathbb{F}\ X ; extY : \mathbb{F}\ Y$

$extZ : \mathbb{F}\ X ; extW : \mathbb{F}\ W$

// Association – a collection (bag) of tuples of the form $(x, y, iz \mapsto z, iw \mapsto w)$,
 // where $iz > 0, iw \geq 0$ are indices (0 denotes absence of index), satisfying the given invariant.
 // Note: if the association has at least one unique ordered end, the collection has no duplicates;
 // formally, the bag maps all tuples to 1; it can be thus represented as a set in Z.
 // Otherwise, it is a "true" bag. For the sake of generality, it is always treated as a bag here.

$Assoc == bag\ (extX \times extY \times (\mathbb{N}_1 \times extZ) \times (\mathbb{N} \times extW))$

Association

Classes

$assoc : Assoc$

$\forall x : extX ; y : extY ; w : extW \bullet$

$\{ z : extZ ; iz : \mathbb{N}_1 ; iw : \mathbb{N} \mid (x, y, iz \mapsto z, iw \mapsto w) \in dom\ assoc \bullet iz \mapsto z \} \in seq\ extZ$

$\forall x : \text{ext}X ; y : \text{ext}Y ; z : \text{ext}Z \bullet$
 $\{ w : \text{ext}W ; iz : \mathbb{N}_1 ; iw : \mathbb{N}_1 \mid (x, y, iz \mapsto z, iw \mapsto w) \in \text{dom } \text{assoc} \bullet iw \mapsto w \} \in \text{iseq } \text{ext}W$

// Association ends – mappings of tuples of objects to collections of objects:

AssociationEnds

Classes

$px : \text{ext}Y \times \text{ext}Z \times \text{ext}W \rightarrow \text{bag } \text{ext}X$

$py : \text{ext}X \times \text{ext}Z \times \text{ext}W \rightarrow \mathbb{P} \text{ ext}Y$

$pz : \text{ext}X \times \text{ext}Y \times \text{ext}W \rightarrow \text{seq } \text{ext}Z$

$pw : \text{ext}X \times \text{ext}Y \times \text{ext}Z \rightarrow \text{iseq } \text{ext}W$

$\forall x : \text{ext}X ; y : \text{ext}Y ; z : \text{ext}Z ; w : \text{ext}W \bullet$
 $x \in \text{dom } px (y, z, w) \Leftrightarrow y \in py (x, z, w) \Leftrightarrow z \in \text{ran } pz (x, y, w) \Leftrightarrow w \in \text{ran } pw (x, y, z)$

// Multiplicity constraints on association ends ($pxmin, pxmax, \dots, pwmin, pwmax$

// are upper and lower multiplicity bounds of the corresponding association ends, respectively):

Multiplicity

AssociationEnds

$\forall x : \text{ext}X ; y : \text{ext}Y ; z : \text{ext}Z ; w : \text{ext}W \bullet$
 $pxmin \leq \text{size } px (y, z, w) \leq pxmax \wedge$
 $pymin \leq \# py (x, z, w) \leq pymax \wedge$
 $pzmin \leq \# pz (x, y, w) \leq pzmax \wedge$
 $pwmin \leq \# pw (x, y, z) \leq pwmax$

// Read Links Action – defines how px, py, pz , and pw are obtained from assoc :

ReadLinksAction

Association

AssociationEnds

$\forall x : \text{ext}X ; y : \text{ext}Y ; z : \text{ext}Z ; w : \text{ext}W \bullet$
 $px (y, z, w) \# x = \text{size } \{ iz : \mathbb{N}_1 ; iw : \mathbb{N} ; n : \mathbb{N} \mid (x, y, iz \mapsto z, iw \mapsto w) \mapsto n \in \text{assoc} \bullet$
 $(x, y, iz \mapsto z, iw \mapsto w) \mapsto n \}$
 $\forall x : \text{ext}X ; z : \text{ext}Z ; w : \text{ext}W \bullet$
 $py (x, z, w) = \{ y : \text{ext}Y ; iz : \mathbb{N}_1 ; iw : \mathbb{N} \mid (x, y, iz \mapsto z, iw \mapsto w) \in \text{dom } \text{assoc} \bullet y \}$
 $\forall x : \text{ext}X ; y : \text{ext}Y ; w : \text{ext}W \bullet$
 $pz (x, y, w) = \{ z : \text{ext}Z ; iz : \mathbb{N}_1 ; iw : \mathbb{N} \mid (x, y, iz \mapsto z, iw \mapsto w) \in \text{dom } \text{assoc} \bullet iz \mapsto z \}$
 $\forall x : \text{ext}X ; y : \text{ext}Y ; z : \text{ext}Z \bullet$
 $pw (x, y, z) = \{ w : \text{ext}W ; iz : \mathbb{N}_1 ; iw : \mathbb{N}_1 \mid (x, y, iz \mapsto z, iw \mapsto w) \in \text{dom } \text{assoc} \bullet iw \mapsto w \}$

// The association's initial state:

AssociationInit

Association

$assoc = \emptyset$

// Create Link Action – defines how a link (as a tuple $(x?, y?, iz? \mapsto z?, iw? \mapsto w?)$)
// is added to the association (as a bag of tuples):

CreateLinkAction

$\Delta Association$

$x? : extX$

$y? : extY$

$z? : extZ ; iz? : \mathbb{N}_1$

$w? : extW ; iw? : \mathbb{N}_1$

// Auxiliary variables for improving readability:

$assoc1a, assoc1b : Assoc$

$assoc2a, assoc2b : Assoc$

$assoc3a, assoc3b : Assoc$

// Insertion indices for z and w in case $iz?$ and $iw?$ are out of range:

$izins, iwins : \mathbb{N}_1$

// Replace every $(x?, y?, iz \mapsto z, iw \mapsto w?)$ such that $iz \geq iz?$ with $(x?, y?, iz + 1 \mapsto z, iw \mapsto w?)$:

$assoc1a = assoc \uplus$

$\{ x : extX ; y : extY ; z : extZ ; iz : \mathbb{N}_1 ; w : extW ; iw : \mathbb{N} \mid x = x? \wedge y = y? \wedge w = w? \wedge iz \geq iz?$
 $\wedge (x, y, iz \mapsto z, iw \mapsto w) \in \text{dom } assoc \bullet$
 $(x, y, iz \mapsto z, iw \mapsto w) \mapsto (assoc \# (x, y, iz \mapsto z, iw \mapsto w)) \}$

$assoc1b = assoc1a \uplus$

$\{ x : extX ; y : extY ; z : extZ ; iz : \mathbb{N}_1 ; w : extW ; iw : \mathbb{N} \mid x = x? \wedge y = y? \wedge w = w? \wedge iz \geq iz?$
 $\wedge (x, y, iz \mapsto z, iw \mapsto w) \in \text{dom } assoc \bullet$
 $(x, y, iz + 1 \mapsto z, iw \mapsto w) \mapsto (assoc \# (x, y, iz \mapsto z, iw \mapsto w)) \}$

// Replace $(x?, y?, iz \mapsto z?, iw_0 \mapsto w?)$ with $(x?, y?, iz \mapsto z?, 0 \mapsto w?)$:

$assoc2a = assoc1b \uplus$

$\{ x : extX ; y : extY ; z : extZ ; iz : \mathbb{N}_1 ; w : extW ; iw : \mathbb{N}_1 \mid x = x? \wedge y = y? \wedge z = z? \wedge w = w?$
 $\wedge (x, y, iz \mapsto z, iw \mapsto w) \in \text{dom } assoc1b \bullet$
 $(x, y, iz \mapsto z, iw \mapsto w) \mapsto (assoc1b \# (x, y, iz \mapsto z, iw \mapsto w)) \}$

$assoc2b = assoc2a \uplus$

$\{ x : extX ; y : extY ; z : extZ ; iz : \mathbb{N}_1 ; w : extW ; iw : \mathbb{N}_1 \mid x = x? \wedge y = y? \wedge z = z? \wedge w = w?$
 $\wedge (x, y, iz \mapsto z, iw \mapsto w) \in \text{dom } assoc1b \bullet$
 $(x, y, iz \mapsto z, 0 \mapsto w) \mapsto (assoc1b \# (x, y, iz \mapsto z, iw \mapsto w)) \}$

// Replace every $(x?, y?, iz \mapsto z?, iw \mapsto w)$ such that $iw \geq iw?$ and $iw < iw_0$ with

// $(x?, y?, iz \mapsto z?, iw + 1 \mapsto w)$:

if $(\exists x : extX ; y : extY ; z : extZ ; iz : \mathbb{N}_1 ; w : extW ; iw_0 : \mathbb{N}_1 \mid x = x? \wedge y = y? \wedge z = z? \wedge w = w? \bullet$
 $(x, y, iz \mapsto z, iw_0 \mapsto w) \in \text{dom } assoc1b)$ then

$assoc3a = assoc2b \uplus$

```

{ x : extX ; y : extY ; z : extZ ; iz :  $\mathbb{N}_1$  ; iz0 :  $\mathbb{N}_1$  ; w : extW ; iw :  $\mathbb{N}_1$  ; w0 : extW ; iw0 :  $\mathbb{N}_1$  |
x = x?  $\wedge$  y = y?  $\wedge$  z = z?  $\wedge$  w0 = w?  $\wedge$  iw  $\geq$  iw?  $\wedge$  iw < iw0  $\wedge$ 
(x, y, iz0  $\mapsto$  z, iw0  $\mapsto$  w0)  $\in$  dom assoc1b  $\wedge$  (x, y, iz  $\mapsto$  z, iw  $\mapsto$  w)  $\in$  dom assoc2b •
(x, y, iz  $\mapsto$  z, iw  $\mapsto$  w)  $\mapsto$  (assoc2b # (x, y, iz  $\mapsto$  z, iw  $\mapsto$  w)) }  $\wedge$ 
assoc3b = assoc3a  $\uplus$ 
{ x : extX ; y : extY ; z : extZ ; iz :  $\mathbb{N}_1$  ; iz0 :  $\mathbb{N}_1$  ; w : extW ; iw :  $\mathbb{N}_1$  ; w0 : extW ; iw0 :  $\mathbb{N}_1$  |
x = x?  $\wedge$  y = y?  $\wedge$  z = z?  $\wedge$  w0 = w?  $\wedge$  iw  $\geq$  iw?  $\wedge$  iw < iw0  $\wedge$ 
(x, y, iz0  $\mapsto$  z, iw0  $\mapsto$  w0)  $\in$  dom assoc1b  $\wedge$  (x, y, iz  $\mapsto$  z, iw  $\mapsto$  w)  $\in$  dom assoc2b •
(x, y, iz  $\mapsto$  z, iw + 1  $\mapsto$  w)  $\mapsto$  (assoc2b # (x, y, iz  $\mapsto$  z, iw  $\mapsto$  w)) }
else
assoc3a = assoc2b  $\uplus$ 
{ x : extX ; y : extY ; z : extZ ; iz :  $\mathbb{N}_1$  ; w : extW ; iw :  $\mathbb{N}_1$  |
x = x?  $\wedge$  y = y?  $\wedge$  z = z?  $\wedge$  iw  $\geq$  iw?  $\wedge$  (x, y, iz  $\mapsto$  z, iw  $\mapsto$  w)  $\in$  dom assoc2b •
(x, y, iz  $\mapsto$  z, iw  $\mapsto$  w)  $\mapsto$  (assoc2b # (x, y, iz  $\mapsto$  z, iw  $\mapsto$  w)) }  $\wedge$ 
assoc3b = assoc3a  $\uplus$ 
{ x : extX ; y : extY ; z : extZ ; iz :  $\mathbb{N}_1$  ; w : extW ; iw :  $\mathbb{N}_1$  |
x = x?  $\wedge$  y = y?  $\wedge$  z = z?  $\wedge$  iw  $\geq$  iw?  $\wedge$  (x, y, iz  $\mapsto$  z, iw  $\mapsto$  w)  $\in$  dom assoc2b •
(x, y, iz  $\mapsto$  z, iw + 1  $\mapsto$  w)  $\mapsto$  (assoc2b # (x, y, iz  $\mapsto$  z, iw  $\mapsto$  w)) }

// Add (x?, y?, iz?  $\mapsto$  z?, iw?  $\mapsto$  w?) to assoc:
izins = min { iz?, max ({0}  $\cup$  { x : extX, y : extY, z : extZ, iz :  $\mathbb{N}_1$ , w : extW, iw :  $\mathbb{N}$  |
x = x?  $\wedge$  y = y?  $\wedge$  w = w?  $\wedge$  (x, y, iz  $\mapsto$  z, iw  $\mapsto$  w)  $\in$  dom assoc • iz }) + 1 }
iwins = min { iw?, max ({0}  $\cup$  { x : extX, y : extY, z : extZ, iz :  $\mathbb{N}_1$ , w : extW, iw :  $\mathbb{N}_1$  |
x = x?  $\wedge$  y = y?  $\wedge$  z = z?  $\wedge$  (x, y, iz  $\mapsto$  z, iw  $\mapsto$  w)  $\in$  dom assoc • iw }) + 1 }
assoc' = assoc3b  $\uplus$  [(x?, y?, izins  $\mapsto$  z?, iwins  $\mapsto$  w?)]

```

// The effect that the Create Link Action should have on the association ends:

CreateLinkAssociationEnds

Δ AssociationEnds

x? : extX

y? : extY

z? : extZ ; iz? : $\mathbb{N}_1$

w? : extW ; iw? : $\mathbb{N}_1$

px' = px $\oplus$ {(y?, z?, w?) $\mapsto$ (px (y?, z?, w?) $\uplus$ [x?])}

py' = py $\oplus$ {(x?, z?, w?) $\mapsto$ (py (x?, z?, w?) $\cup$ {y?})}

pz' = pz $\oplus$ {(x?, y?, w?) $\mapsto$ (pz (x?, y?, w?) $\downarrow$ {z? $\mapsto$ iz?})}

pw' = pw $\oplus$ {(x?, y?, z?) $\mapsto$ (pw (x?, y?, z?) $\uparrow$ (extW $\setminus$ {w?}) $\downarrow$ {w? $\mapsto$ iw?})}

// The complete definition of the Destroy Link Action on association is omitted for the sake of brevity

// The effect that the Destroy Link Action should have on the association ends:

$\Delta AssociationEnds$

$x? : extX$

$y? : extY$

$z? : extZ ; iz? : \mathbb{N}_1$

$w? : extW$

if

$x? \notin \text{dom } px(y?, z?, w?) \vee y? \notin \text{py}(x?, z?, w?) \vee z? \notin \text{ran } pz(x?, y?, w?) \vee w? \notin \text{ran } pw(x?, y?, z?)$
 $\vee pz(x?, y?, w?) (iz?) \neq z?$

then

$px' = px \wedge py' = py \wedge pz' = pz \wedge pw' = pw$

else

$\forall x : extX ; y : extY ; z : extZ ; w : extW \bullet$

if

$x \neq x? \vee y \neq y? \vee z \neq z? \vee w \neq w?$

then

$px'(y, z, w) = px(y, z, w) \wedge py'(x, z, w) = py(x, z, w) \wedge$

$pz'(x, y, w) = pz(x, y, w) \wedge pw'(x, y, z) = pw(x, y, z)$

else

$\odot px'(y, z, w) \# x = px(y, z, w) \# x - 1 \wedge$ // Only when the association has no unique ends

$\odot px'(y, z, w) \# x = 0 \wedge$ // Only when the association has a unique end

$py'(x, z, w) = py(x, z, w) \setminus \{y\} \wedge$

$\odot pz'(x, y, w) = \mathbb{N}_1 \setminus \{iz?\} \upharpoonright pz(x, y, w) \wedge$ // Only when the association has no unique ends

$\odot pz'(x, y, w) = pz(x, y, w) \upharpoonright (extZ \setminus \{z\}) \wedge$ // Only when the association has a unique end

$pw'(x, y, z) = pw(x, y, z) \upharpoonright (extW \setminus \{w\})$

REFERENCES

- [1] Barbier, F., Henderson-Sellers, B., Le Parc-Lacayrelle, A., Bruel, J.-M., "Formalization of the Whole-Part Relationship in the Unified Modeling Language," *IEEE Trans. Software Engineering*, Vol. 29, No. 5, May 2003, pp. 459-470
- [2] Booch, G., Rumbaugh, J., Jacobson, I., *The Unified Modeling Language User Guide*, Addison-Wesley Longman, 1999
- [3] Bourdeau, R. H., Cheng, B. H. C., "A Formal Semantics for Object Model Diagrams," *IEEE Trans. Software Engineering*, Vol. 21, No. 10, October 1995, pp. 799-821
- [4] Chen, P. P., "The Entity-Relationship Model," *ACM Transactions on Database Systems*, Vol. 1, No. 1, 1976, pp. 9-36
- [5] Diskin, Z., Dingel, J., "Towards Formal Semantics for Associations in UML 2.0," manuscript being prepared for submission (available through private communication), 2005
- [6] France, R. B., "A Problem-Oriented Analysis of Basic UML Static Requirements Modeling

Concepts," *Proc. 1999 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '99)*, ACM SIGPLAN Notices, Vol. 34, No. 10, November 1999, pp. 57-69

- [7] Génova, G., Llorens, J., Fuentes, J. M., "UML Associations: A Structural and Contextual View," *Journal of Object Technology*, Vol. 3, No. 7, July-August 2004, pp. 83-100
- [8] Génova, G., Llorens, J., Martínez, P., "The Meaning of Multiplicity of N-ary Associations in UML," *Software and Systems Modeling*, Vol. 1, No. 2, February 2002, pp. 86-97
- [9] Object Management Group, *Issues for Mailing list of the UML 2.0 Superstructure and Infrastructure Revision Task Force*, Issue 5977, June 2003, <http://www.omg.org/issues/uml2-rtf.open.html#Issue5977>
- [10] Object Management Group, *UML 2.0 Superstructure Specification*, revised final adopted specification (ptc/04-10-02), October 8, 2004, <http://www.omg.org>
- [11] Övergaard, G., "A Formal Approach to Relationships in the Unified Modeling Language," *Proc. PSMT'98 Workshop on Precise Semantics for Modeling Techniques*, Technische Universität München, TUM-19803, 1998
- [12] Övergaard, G., *Formal Specification of Object-Oriented Modelling Concepts*, PhD Thesis, Department of Teleinformatics, Royal Institute of Technology, Stockholm, Sweden, November 2000
- [13] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., *Object-Oriented Modeling and Design*, Prentice-Hall International, 1991
- [14] Selic, B., "On the Semantic Foundations of Standard UML 2.0," *Formal Methods for the Design of Real-Time Systems: International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM-RT 2004)*, Bertinora, Italy, September 13-18, 2004, Springer-Verlag Lecture Notes in Computer Science, Vol. 3185/2004, pp. 181-199
- [15] Selic, B., "The Pragmatics of Model-Driven Development," *IEEE Software*, Vol. 20, No. 5, September/October 2003, pp. 19-25
- [16] Selic, B., Ramackers, G., Kobryn, C., "Evolution, Not Revolution," *Communications of the ACM*, Vol. 45, No. 11, November 2002, pp. 70-72
- [17] Spivey, J. M., *The Z Notation: A Reference Manual*, Prentice-Hall International, 1992
- [18] Stevens, P., "On the Interpretation of Binary Associations in the Unified Modeling Language," *Software and Systems Modeling*, Vol. 1, No. 1, January 2002, pp. 68-79