UML Static Models in Formal Approach

Marcin Szlenk

Warsaw University of Technology
Institute of Control & Computation Engineering
Nowowiejska 15/19, 00-665 Warsaw, Poland
M.Szlenk@ia.pw.edu.pl

Abstract. The semantics of models written in UML is not precisely defined. Thus, it is hard to determine, how a given change in a model influences its meaning and, for example, to verify whether a given model transformation preserves the semantics of the model or not. In the paper a formal (mathematical) semantics of key elements of the UML static models is presented. The aim is to define the basic semantic relations between models: a consequence (implication) and equivalence. The goal of the definitions and examples presented in the article is to form a very basic, concise, theoretical foundation for the formal comparison of the UML static models, based on their meanings.

Key words: Software modeling, UML, Formal reasoning.

1 Introduction

Unified Modeling Language (UML) [9, 12] is a visual modeling language that is used to specify, construct and document software systems. The UML has been adopted and standardized by the *Object Management Group* (OMG). The UML specification [12], published by OMG, is based on a metamodeling approach. The metamodel (a model of UML) gives information about the abstract syntax of UML, but does not deal with semantics, which is expressed in a natural language. Furthermore, because UML is method-independent, its specification tends to set a range of potential interpretations rather than providing an exact meaning.

As far as software modeling is concerned, we can distinguish two types of models: dynamic and static. The dynamic model is used to express and model the behaviour of a problem domain or system over time, whereas the static model shows those aspects that do not change over time. UML static models are mainly expressed using a class diagram that shows a collection of classes and their interrelationships, for example generalization/specialization and association.

After the first UML specification was published, various propositions of UML formalization have appeared. The semantics of class diagrams was expressed using such formal languages as Z [4, 5], PVS-SL [1], description logic [2] and RAISE-SL [6]. Some of the works were restricted to the semantics of models, while the others were concerned with the issues of reasoning about models and model transformation. It seems that the subject of reasoning about UML static

PREPRINT OF THE PROCEEDINGS OF CEE-SET 2007, www.cee-set.org, pp. 111 - 124, 2007.

models still lacks a formal approach to the problem of the semantic equivalence of two models. There are some informal approaches but they result in unverifiable model transformation rules (see e.g. [3, 8]).

In the paper the syntax and semantics of a UML static model restricted to key elements of a class diagram are formally defined. The definitions which are presented here adhere to the UML 2. Using the proposed formalization, we show how one can reason about UML static models in a fully formal way, especially about their equivalence.

2 Metalanguage

As a language for defining the semantics of UML static models, we use basic mathematical notation. In this section we briefly outline only the list and function notation, as they may vary in different publications.

For a set A, $\mathcal{P}(A)$ denotes the set of all the subsets of A, and A^* denotes the set of all the finite lists of elements of A. The function $\mathbf{len}(l)$ returns the length of a list l. For simplicity, we add the expression $A^{*(2)}$, which denotes the set of all finite lists with a length of at least 2. The function $\pi_i(l)$ projects the i-th element of a list l, whereas the function $\overline{\pi}_i(l)$ projects all but the i-th element. The list $[a_1, \ldots, a_n]$ is formally equal to the tuple $(a_1, \ldots, a_n)$. For a finite set A, $|A|$ denotes the number of elements of A.

The partial function from A to B is denoted by $f \colon A \rightharpoonup B$, where the function $\mathbf{dom}(f)$ returns the domain of f. The expression $f \colon A \to B$ denotes the total function from A to B (in this case it holds $\mathbf{dom}(f) = A$).

3 Syntax

The key concepts used in UML static models: class, association and association class are considered here. All of them are types of classifier.[1] Taxonomical relationships among them, which are defined in the UML metamodel, are shown in Fig. 1. It is worth emphasizing that an association class is a single model element which is both an association and a class.

Definition 1 (Classifiers). *With* Classifiers *we denote a set of all the classifiers (the names of classes, associations and association classes) which may appear in a static model.*

Below we formally define abstract syntax of simple UML static models. The syntax is defined in a way which reflects the relationships from Fig. 1. It makes both the definition of the syntax and the semantics (discussed later) more concise.

Definition 2 (Model). *By a* (static) model *we understand a tuple*

$$\mathrm{M} = (\mathrm{classes}, \mathrm{assocs}, \mathrm{ends}, \mathrm{mults}, \mathrm{specs}) \text{ , \textit{where:}} \tag{1}$$

[1] Association is included as a type of classifier since the introduction of UML 2.0.

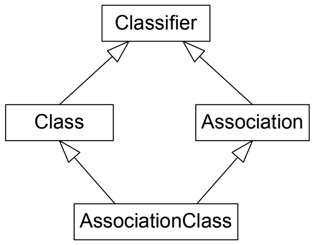

Fig. 1. The part of the hierarchy of classifiers in the UML metamodel [12]

1. M.classes *is a set of classes:*

$$\mathrm{M.classes} \subseteq \mathrm{Classifiers}\ . \tag{2}$$

2. M.assocs *is a set of associations:*

$$\mathrm{M.assocs} \subseteq \mathrm{Classifiers}\ . \tag{3}$$

For the model M*, a set of association classes and a set of all classifiers are thus respectively defined as:*

$$\mathrm{M.asclasses} =_{def} \mathrm{M.classes} \cap \mathrm{M.assocs}\ , \tag{4}$$

$$\mathrm{M.classifiers} =_{def} \mathrm{M.classes} \cup \mathrm{M.assocs}\ . \tag{5}$$

3. M.ends *is a function of association ends. The function maps each association to a finite list of at least two, not necessarily different, classes participating in the association:*

$$\mathrm{M.ends}\colon \mathrm{M.assocs} \to \mathrm{M.classes}^{*(2)}\ . \tag{6}$$

The position on the list M.ends(as) *uniquely identifies the association end. An association class cannot be defined between itself and something else [12, p. 47]:*

$$\forall ac \in \mathrm{M.asclasses} \cdot \forall i \in \{1, \ldots, \mathbf{len}(\mathrm{M.ends}(ac))\} \cdot \\ \pi_i(\mathrm{M.ends}(ac)) \neq ac\ . \tag{7}$$

4. M.mults *is a function of multiplicity of association ends. Multiplicity is a non-empty set of non-negative integers with at least one value greater than zero. The default multiplicity is the set of all non-negative integers (*$\mathbb{N}$*). The function assigns to each association a list of multiplicity on its ends:*

$$\mathrm{M.mults}\colon \mathrm{M.assocs} \to (\mathcal{P}(\mathbb{N}) \setminus \{\emptyset, \{0\}\})^{*(2)}\ . \tag{8}$$

As before, the position on the list M.mults(as) *identifies the association end. The multiplicity must be defined for each association end:*

$$\forall as \in \mathrm{M.assocs} \cdot \mathbf{len}(\mathrm{M.mults}(as)) = \mathbf{len}(\mathrm{M.ends}(as))\ . \tag{9}$$

5. M.specs *is a function of specializations. The function assigns to each classifier a set of all (direct or indirect) its specializations:*

$$\text{M.specs}\colon \text{M.classifiers} \to \mathcal{P}(\text{M.classifiers}) \ . \tag{10}$$

The specialization hierarchy must be acyclical [12, p. 53], what means that a classifier cannot be its own specialization:

$$\forall cf \in \text{M.classifiers} \cdot cf \notin \text{M.specs}(cf) \ . \tag{11}$$

By default a classifier may specialize classifiers of the same or a more general type [12, p. 54], i.e. class may be a specialization of class; association may be a specialization of association; association class may be a specialization of association class, class or association. Formally:

$$\forall cl \in \text{M.classes} \cdot \text{M.specs}(cl) \subseteq \text{M.classes}, \tag{12}$$

$$\forall as \in \text{M.assocs} \cdot \text{M.specs}(as) \subseteq \text{M.assocs}, \tag{13}$$

$$\forall cf \in \text{M.classifiers} \cdot \text{M.specs}(cf) \nsubseteq \text{M.asclasses} \Rightarrow cf \notin \text{M.asclasses} \ . \tag{14}$$

An association specializing another association has the same number of ends:

$$\forall as_1, as_2 \in \text{M.assocs} \cdot as_2 \in \text{M.specs}(as_1) \Rightarrow \mathbf{len}(\text{M.ends}(as_1)) = \mathbf{len}(\text{M.ends}(as_2)) \ , \tag{15}$$

which are connected to the same classifiers as in a specialized association or to their specializations [12, p. 39]:

$$\forall as_1, as_2 \in \text{M.assocs} \cdot as_2 \in \text{M.specs}(as_1) \Rightarrow \forall i \in \{1, \ldots, \mathbf{len}(\text{M.ends}(as_1))\} \cdot \pi_i(\text{M.ends}(as_2)) \in \{\pi_i(\text{M.ends}(as_1))\} \cup \text{M.specs}(\pi_i(\text{M.ends}(as_1))) \ . \tag{16}$$

The above definition does not include directly attributes of classes. However, an attribute has the same semantics as an association. An example of attributes and corresponding associations are shown in Fig. 2.[2]

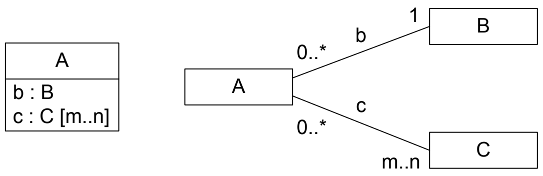

Fig. 2. Attributes and associations

Definition 3 (Models). Models *denotes a set of all the models, as in definition 2.*

[2] This unification of attributes and associations is new to UML 2.0.

4 Semantics

A classifier describes a set of instances that have something in common [9]. An instance of a class is called an object, whereas an instance of an association is called a link. A link is a connection between two or more objects of the classes at corresponding positions in the association. An instance of a class association is both an object and a link, so it can both be connected by links and can connect objects.

Definition 4 (Instances). Instances *denotes a set of all the potential instances of the classifiers from the set* Classifiers*.*

The existing instances of a classifier are called its extent. If two classifiers are linked by a specialization relationship, then each instance of a specializing (specific) classifier is also an instance of a specialized (general) classifier [12]. In other words, the extent of the specific classifier is a subset of the extent of the general one.

The classifier extent usually varies over time as objects and links may be created and destroyed. Thus, the classifiers' extents form a snapshot of the state of a modelled problem domain or system at a particular point in time.

Definition 5 (State). State *is a pair*

$$\mathrm{S} = (\mathrm{instances}, \mathrm{ends}) \ , \textit{where:} \tag{17}$$

1. S.instances *is a partial function of extents. The function maps each classifier to a set of its instances (extent):*

$$\mathrm{S.instances}\colon \mathrm{Classifiers} \rightharpoonup \mathcal{P}(\mathrm{Instances}) \ . \tag{18}$$

2. S.ends *is a partial function of link ends. The function assigns to each instance of an association, i.e. link, a list of instances of classes (objects) which are connected by the link:*

$$\mathrm{S.ends}\colon \mathrm{Instances} \rightharpoonup \mathrm{Instances}^{*(2)} \ . \tag{19}$$

The position on the list uniquely identifies the link end, which on the other hand, corresponds to an appropriate association end.

Definition 6 (States). *With* States *we denote a set of all the states as in the definition 5.*

The static model shows the structure of states (of a given domain or system) or, from a different point of view, defines some constraints on states. Thus, the model can be interpreted as the set of all such states in which the mentioned constraints are satisfied. Below we define the relationship between models and states as a relation of satisfaction: $\mathrm{Sat} \subseteq \mathrm{Models} \times \mathrm{States}$. If $\mathrm{Sat}(\mathrm{M}, \mathrm{S})$ holds then the constraints expressed as model M are satisfied in the state S. Next, we formally define the meaning of a model as the set of all states in which the model is satisfied.

Definition 7 (Satisfaction). *Let* S ∈ States *and* M ∈ Models*. The model* M *is* satisfied in the state S *and we write*

$$\mathrm{Sat}(\mathrm{M},\mathrm{S})\ , \textit{if and only if:} \tag{20}$$

1. S *specifies the extents of all classifiers in* M *(and maybe others, not depicted in the model* M*)*[3]*:*

$$\mathrm{M.classifiers} \subseteq \mathbf{dom}(\mathrm{S.instances})\ . \tag{21}$$

2. *An instance of a given association only connects instances of classes participating in this association (on the appropriate ends):*

$$\begin{aligned} &\forall as \in \mathrm{M.assocs} \cdot \forall ln \in \mathrm{S.instances}(as) \cdot \\ &\quad \mathbf{len}(\mathrm{M.ends}(as)) = \mathbf{len}(\mathrm{S.ends}(ln)) \wedge \\ &\quad \forall i \in \{1, \ldots, \mathbf{len}(\mathrm{M.ends}(as))\} \cdot \\ &\qquad \pi_i(\mathrm{S.ends}(ln)) \in \mathrm{S.instances}(\pi_i(\mathrm{M.ends}(as)))\ . \end{aligned} \tag{22}$$

3. *Instances of an association satisfy the specification of multiplicity on all association ends.*[4] *For any $n-1$ ends of n-ary association ($n \geq 2$) and $n-1$ instances of classes on those ends, the number of links they form with instances of the class on the remaining end belong to the multiplicity of this end [12, p. 40]:*

$$\begin{aligned} &\forall as \in \mathrm{M.assocs} \cdot \\ &\quad \forall i \in \{1, \ldots, \mathbf{len}(\mathrm{M.ends}(as))\} \cdot \forall p \in \mathrm{product}(as, i) \cdot \\ &\qquad |\{\, ln \in \mathrm{S.instances}(as) : \overline{\pi}_i(\mathrm{S.ends}(ln)) = p \,\}| \in \\ &\qquad \pi_i(\mathrm{M.mults}(as))\ , \end{aligned} \tag{23}$$

where:

$$\mathrm{product}(as, i) =_{def} \bigtimes_{j=1,\ j \neq i}^{\mathbf{len}(\mathrm{M.ends}(as))} \mathrm{S.instances}(\pi_j(\mathrm{M.ends}(as)))\ . \tag{24}$$

4. *An extent of an association includes, at most, one link connecting a given set of class instances (on given link ends):*[5]

$$\begin{aligned} &\forall as \in \mathrm{M.assocs} \cdot \forall ln_1, ln_2 \in \mathrm{S.instances}(as) \cdot \\ &\quad ln_1 \neq ln_2 \Rightarrow \exists i \in \{1, \ldots, \mathbf{len}(\mathrm{M.ends}(as))\} \cdot \\ &\qquad \pi_i(\mathrm{S.ends}(ln_1)) \neq \pi_i(\mathrm{S.ends}(ln_2))\ . \end{aligned} \tag{25}$$

[3] This issue is discussed in terms of "complete" and "incomplete" class diagrams in [5].

[4] The meaning of multiplicity for an association with more than two ends lacked precision in terms of its definition in UML prior to version 2.0. Possible interpretations are discussed in detail in [7].

[5] This condition does not have to be true for an association with a {bag} adornment. However, for the sake of simplicity, such associations are not considered here.

5. *An instance of a specializing classifier is also an instance of the specialized classifier:*

$$\begin{aligned}\forall cf_1, cf_2 \in \text{M.classifiers} \cdot cf_2 \in \text{M.specs}(cf_1) \Rightarrow \\ \text{S.instances}(cf_2) \subseteq \text{S.instances}(cf_1)\ .\end{aligned} \tag{26}$$

Definition 8 (Meaning). *Let* $\text{M} \in \text{Models}$ *and* $\mathcal{M}\colon \text{Models} \rightarrow \mathcal{P}(\text{States})$ *be the function which is defined as:*

$$\mathcal{M}(\text{M}) =_{def} \{\, \text{S} \in \text{States} : \text{Sat}(\text{M}, \text{S}) \,\}\ . \tag{27}$$

The value $\mathcal{M}(\text{M})$ *refers to the* meaning of M.

5 Consequence

The mathematically defined semantics of a UML model allows for the reasoning about the properties presented in a model. The properties which are implied from the semantics of a given model, and are expressed in this model somehow implicitly, may be shown directly in the form of a different model. The relationship between two such models is defined below as a relation of semantic consequence: $\Rightarrow\ \subseteq \text{Models} \times \text{Models}$. If for a given problem domain or system the properties expressed in the model M_1 are true and it holds $\text{M}_1 \Rightarrow \text{M}_2$, then for the forementioned problem domain or system the properties expressed in the model M_2 are also true.

Definition 9 (Consequence). *Let* $\text{M}_1, \text{M}_2 \in \text{Models}$. *The model* M_2 *is a* (semantic) consequence of M_1 *and can be expressed as*

$$\text{M}_1 \Rightarrow \text{M}_2\ , \textit{if and only if}\ \mathcal{M}(\text{M}_1) \subseteq \mathcal{M}(\text{M}_2)\ . \tag{28}$$

The relation of consequence $\Rightarrow$ is transitive in the set Models $((\text{M}_1 \Rightarrow \text{M}_2 \wedge \text{M}_2 \Rightarrow \text{M}_3) \Rightarrow (\text{M}_1 \Rightarrow \text{M}_3))$, so to show that one diagram ia a consequence of another, it can be proved in several simpler steps. Below one of the basic reasoning rules is formally presented. Some other examples are shown in Fig. 3. Many reasoning rules, including proof of their correctness, are presented in detail in [10].

Theorem 1 (Extending multiplicity). *Let* $\text{M}_1, \text{M}_2 \in \text{Models}$ *be such that:*

$$\begin{aligned} &\text{M}_1.\text{classes} = \text{M}_2.\text{classes}\ , \qquad && \text{M}_1.\text{mults} \neq \text{M}_2.\text{mults}\ , \\ &\text{M}_1.\text{assocs} = \text{M}_2.\text{assocs}\ , && \text{M}_1.\text{specs} = \text{M}_2.\text{specs}\ , \\ &\text{M}_1.\text{ends} = \text{M}_2.\text{ends}\ , \end{aligned} \tag{29}$$

and the models include the association $AS \in \text{M}_1.\text{assocs}$*, such that the multiplicity on its end* k *in the model* M_1 *is a proper subset of the multiplicity on this end in the model* M_2*:*

$$\pi_k(\text{M}_1.\text{mults}(AS)) \subset \pi_k(\text{M}_2.\text{mults}(AS))\ , \tag{30}$$

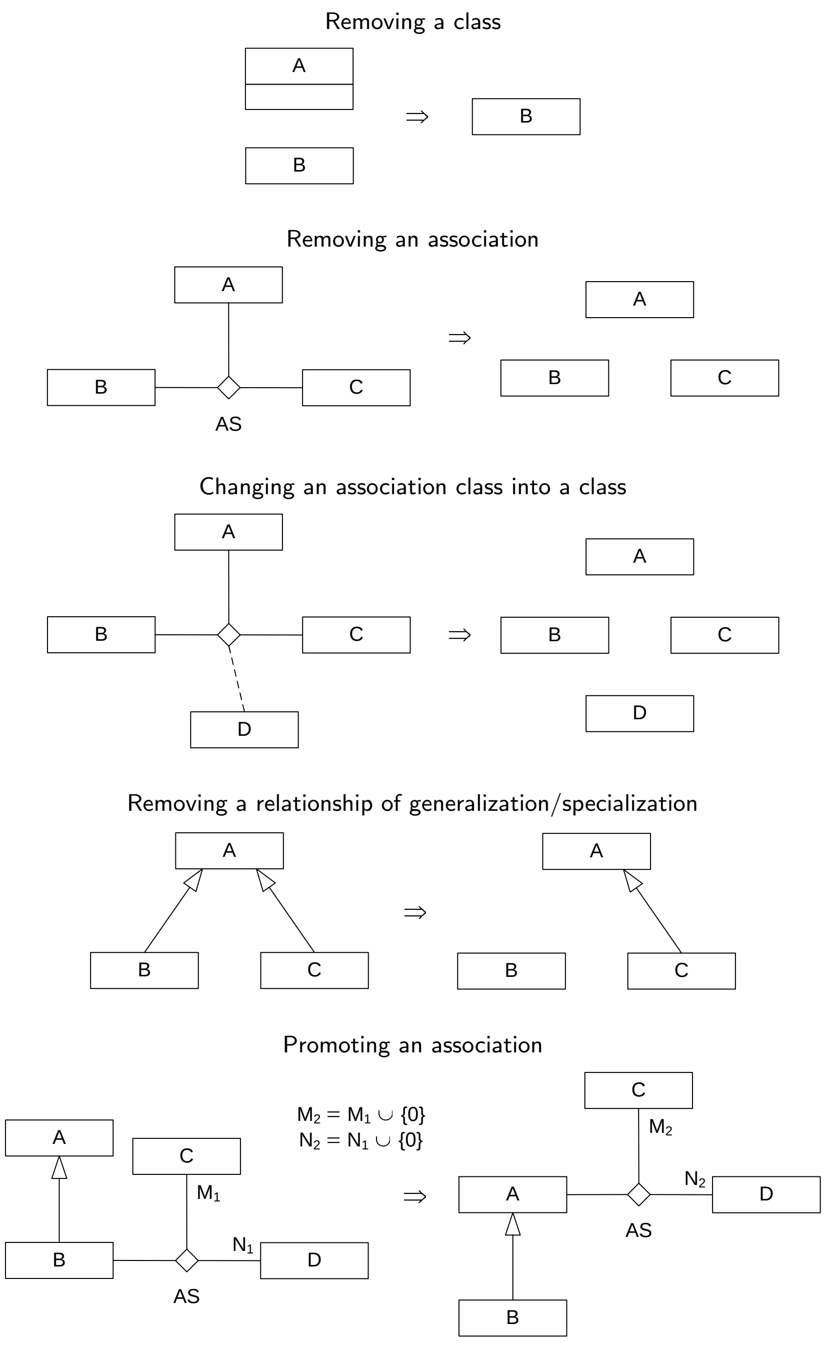

Fig. 3. Examples of a consequence relationship

whereas the multiplicity specifications on the other ends of the association AS and on the ends of the other associations are the same in both models:

$$\overline{\pi}_k(\mathrm{M}_1.\mathrm{mults}(AS)) = \overline{\pi}_k(\mathrm{M}_2.\mathrm{mults}(AS)) \ , \tag{31}$$

$$\forall as \in \mathrm{M}_1.\mathrm{assocs} \setminus \{AS\} \cdot \mathrm{M}_1.\mathrm{mults}(as) = \mathrm{M}_2.\mathrm{mults}(as) \ . \tag{32}$$

Then the model M_2 *is a consequence of* M_1 *(see Fig. 4).*

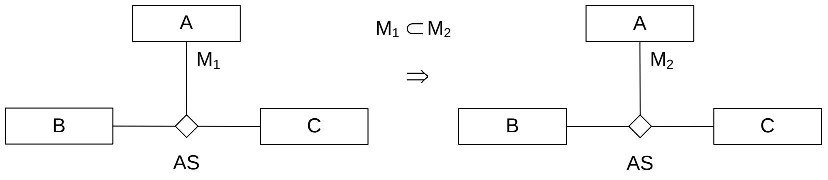

Fig. 4. Extending multiplicity

Proof. Let $\mathrm{S} \in \mathcal{M}(\mathrm{M}_1)$ (i.e. $\mathrm{Sat}(\mathrm{M}_1, \mathrm{S})$ holds). Within the framework of the proof that $\mathrm{Sat}(\mathrm{M}_2, \mathrm{S})$ holds we will show that point 3 of the definition 7 is satisfied. The satisfaction of the remaining points of this definition is implied directly from $\mathrm{Sat}(\mathrm{M}_1, \mathrm{S})$ and the condition (29).

Let $as \in \mathrm{M}_2.\mathrm{assocs}$, $i \in \{1, \ldots, \mathbf{len}(\mathrm{M}_2.\mathrm{ends}(as))\}$ and $p \in \mathrm{product}(as, i)$ (from the condition (29) the function 'product' has the same form for both models M_1 and M_2). If $as \neq AS$ or $i \neq k$, then the forementioned point is satisfied from $\mathrm{Sat}(\mathrm{M}_1, \mathrm{S})$ and the conditions (31) and (32). Otherwise, from $\mathrm{Sat}(\mathrm{M}_1, \mathrm{S})$ it holds:

$$|\{\, ln \in \mathrm{S}.\mathrm{instances}(AS) : \overline{\pi}_k(\mathrm{S}.\mathrm{ends}(ln)) = p \,\}| \in \pi_k(\mathrm{M}_1.\mathrm{mults}(AS)) \tag{33}$$

and from the condition (30):

$$|\{\, ln \in \mathrm{S}.\mathrm{instances}(AS) : \overline{\pi}_k(\mathrm{S}.\mathrm{ends}(ln)) = p \,\}| \in \pi_k(\mathrm{M}_2.\mathrm{mults}(AS)) \ . \tag{34}$$

5.1 Refinement

A refinement is a relationship that represents a fuller specification of something that has already been specified at a certain level of detail or at a different semantic level [9]. Fig. 5 shows seven simple models of the same problem domain but at different levels of detail (at different stages of development). Each consecutive model includes some more details about the modelled domain and thus can be treated as a refinement of any of the previous models. At the same time, if a given model is a refinement of another then they both are related by a consequence relationship, as it is shown in the figure. Generally, if $\mathrm{M}_1 \Rightarrow \mathrm{M}_2$ holds then the model M_1 is at least as detailed (complete or precise) description of a given problem domain or system as the model M_2.

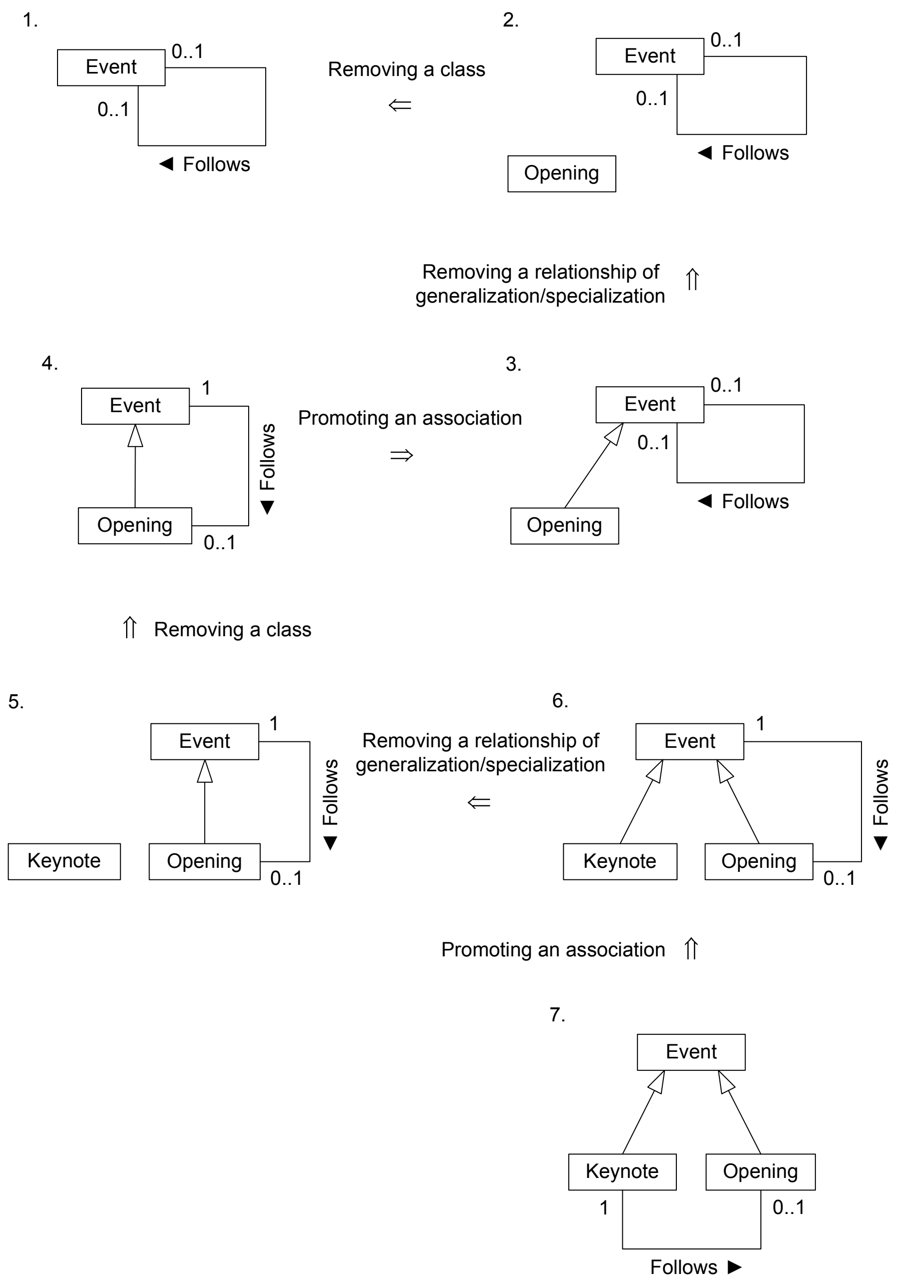

Fig. 5. Refinement vs. consequence relationship

6 Equivalence

Two models with exactly the same meaning are a specific case of the relation of consequence. Such cases are defined below as a relation of semantical equivalence: $\Leftrightarrow\ \subseteq \mathrm{Models} \times \mathrm{Models}$. If $\mathrm{M}_1 \Leftrightarrow \mathrm{M}_2$ holds, then the models M_1 and M_2 are completely interchangeable descriptions of a given problem domain or system (or their parts).

Definition 10 (Equivalence). *Let* $\mathrm{M}_1, \mathrm{M}_2 \in \mathrm{Models}$. *The model* M_1 *is* (semantically) equivalent to M_2 *and can be expressed as*

$$\mathrm{M}_1 \Leftrightarrow \mathrm{M}_2\ , \textit{if and only if}\ \mathrm{M}_1 \Rightarrow \mathrm{M}_2 \wedge \mathrm{M}_2 \Rightarrow \mathrm{M}_1\ . \tag{35}$$

6.1 An example of equivalence

For the specialization of an association, the UML metamodel [12] defines only two syntactical constraints (see the definition 2): an association specializing another association has the same number of ends and its ends are connected to the same classifiers as in a specialized association or to their specializations. In fact, these constraints partially reflect the semantics of a specializing association, which instances are the specific cases of instances of a specialized association. Between two such associations, however, other dependencies which have not been taken into account in the UML metamodel and which can be shown using our formalization also exist. Below we present one example of such dependencies.

Theorem 2 (Association specialization vs. multiplicity). *Let* $\mathrm{M}_1, \mathrm{M}_2 \in \mathrm{Models}$ *be such that:*

$$\begin{aligned} \mathrm{M}_1.\mathrm{classes} &= \mathrm{M}_2.\mathrm{classes}\ , & \mathrm{M}_1.\mathrm{mults} &\neq \mathrm{M}_2.\mathrm{mults}\ , \\ \mathrm{M}_1.\mathrm{assocs} &= \mathrm{M}_2.\mathrm{assocs}\ , & \mathrm{M}_1.\mathrm{specs} &= \mathrm{M}_2.\mathrm{specs}\ , \\ \mathrm{M}_1.\mathrm{ends} &= \mathrm{M}_2.\mathrm{ends}\ , \end{aligned} \tag{36}$$

and the models include the associations $AA, AB \in \mathrm{M}_1.\mathrm{assocs}$*, such that* AB *is a specialization of* AA*:*

$$AB \in \mathrm{M}_1.\mathrm{specs}(AA)\ , \tag{37}$$

and for the multiplicity on the end k *of the association* AB *the below condition holds:*[6]

$$\begin{aligned} &\pi_k(\mathrm{M}_2.\mathrm{mults}(AB)) = \\ &\qquad \pi_k(\mathrm{M}_1.\mathrm{mults}(AB)) \cap \{0, \ldots, \max(\pi_k(\mathrm{M}_1.\mathrm{mults}(AA)))\}\ , \end{aligned} \tag{38}$$

whereas the multiplicity specifications on the other ends of the association AB *and on the ends of the other associations are the same in both models:*

$$\overline{\pi}_k(\mathrm{M}_1.\mathrm{mults}(AB)) = \overline{\pi}_k(\mathrm{M}_2.\mathrm{mults}(AB))\ , \tag{39}$$

$$\forall as \in \mathrm{M}_1.\mathrm{assocs} \setminus \{AB\} \cdot \mathrm{M}_1.\mathrm{mults}(as) = \mathrm{M}_2.\mathrm{mults}(as)\ . \tag{40}$$

Then the model M_1 *is equivalent to* M_2 *(see Fig. 6).*

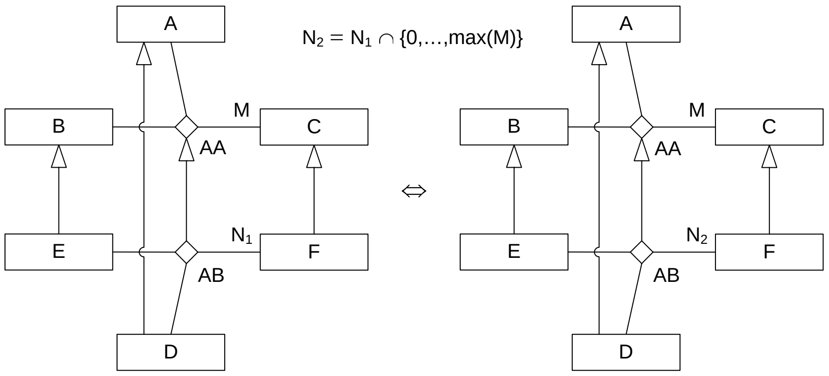

Fig. 6. Association specialization vs. multiplicity

Proof. Firstly, we will prove that $\mathrm{M}_1 \Rightarrow \mathrm{M}_2$ and then $\mathrm{M}_2 \Rightarrow \mathrm{M}_1$ hold.

$\mathbf{M_1 \Rightarrow M_2}$**.** Let $\mathrm{S} \in \mathcal{M}(\mathrm{M}_1)$. Within the framework of the proof that $\mathrm{Sat}(\mathrm{M}_2, \mathrm{S})$ holds we will show that point 3 of the definition 7 is satisfied. The satisfaction of the remaining points of this definition is implied directly from $\mathrm{Sat}(\mathrm{M}_1, \mathrm{S})$ and the condition (36).

Let $as \in \mathrm{M}_2.\mathrm{assocs}$, $i \in \{1, \ldots, \mathbf{len}(\mathrm{M}_2.\mathrm{ends}(as))\}$ and $p \in \mathrm{product}(as, i)$ (from the condition (36) the function 'product' has the same form for both models M_1 and M_2). If $as \neq AB$ or $i \neq k$, then the mentioned point is satisfied from $\mathrm{Sat}(\mathrm{M}_1, \mathrm{S})$ and the conditions (39) and (40). Otherwise, if we use the symbols:

$$\alpha_{AA}(i,p) =_{def} |\{\, ln \in \mathrm{S.instances}(AA) : \overline{\pi}_i(\mathrm{S.ends}(ln)) = p \,\}| \text{ and} \tag{41}$$
$$\alpha_{AB}(i,p) =_{def} |\{\, ln \in \mathrm{S.instances}(AB) : \overline{\pi}_i(\mathrm{S.ends}(ln)) = p \,\}| \ , \tag{42}$$

it remains to be shown that the below holds:

$$\alpha_{AB}(k,p) \in \pi_k(\mathrm{M}_2.\mathrm{mults}(AB)) \ . \tag{*}$$

Because $\mathrm{Sat}(\mathrm{M}_1, \mathrm{S})$ holds, therefore:

$$\alpha_{AA}(k,p) \in \pi_k(\mathrm{M}_1.\mathrm{mults}(AA)) \text{ and} \tag{43}$$
$$\alpha_{AB}(k,p) \in \pi_k(\mathrm{M}_1.\mathrm{mults}(AB)) \ , \tag{44}$$

and from condition (37) and from point 5 of the definition 7:

$$\mathrm{S.instances}(AB) \subseteq \mathrm{S.instances}(AA) \ . \tag{45}$$

Then the following inequality holds:

$$\alpha_{AB}(k,p) \leq \alpha_{AA}(k,p) \tag{46}$$

[6] If X is an infinite subset of $\mathbb{N}$, then we assume $\{0, \ldots, \max(X)\} =_{def} \mathbb{N}$.

and from the equation (43):

$$\alpha_{AB}(k,p) \in \{0,\dots,\max(\pi_k(\mathrm{M}_1.\mathrm{mults}(AA)))\} \ . \tag{47}$$

From the equation (44):

$$\alpha_{AB}(k,p) \in \pi_k(\mathrm{M}_1.\mathrm{mults}(AB)) \cap \{0,\dots,\max(\pi_k(\mathrm{M}_1.\mathrm{mults}(AA)))\} \tag{48}$$

and the property (*) holds on the assumption (38).

$\mathbf{M_2 \Rightarrow M_1}$. From the assumptions of our theorem:

$$\pi_k(\mathrm{M}_2.\mathrm{mults}(AB)) \subseteq \pi_k(\mathrm{M}_1.\mathrm{mults}(AB)) \ . \tag{49}$$

If the above sets are equal, then $\mathrm{M}_1 = \mathrm{M}_2$. Otherwise, the assumptions of theorem 1 are satisfied.

Example 1 (Imprecise multiplicity specification). Fig. 7 illustrates a situation at a hypothetical scientific conference, where participants can be the authors (or co-authors) of no more than two papers submitted to the conference. If the submitted paper is accepted for the presentation during the conference, it is presented by one of its authors. By virtue of theorem 2, one author cannot have more than two presentations.

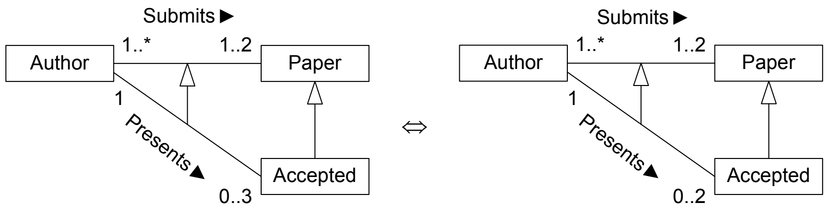

Fig. 7. Imprecise multiplicity specification

7 Conclusion

Theoretical research work in the area of UML formalization, although rather difficult to be applied directly in software engineering practice, can be useful in facilitating a better understanding of UML modeling concepts and can contribute to improving the UML specification itself. In the paper we have proposed a concise formalization of basic UML static models and have shown they can be helpful in formal reasoning. The presented formalization can easily include other elements of UML static models, which have not been addressed here, for example an aggregation, a composition or abstract classifiers [10]. It was also used to define the problem of the semantic consistency of individual models [11].

References

1. Aredo D., Traoré I., Stølen K.: Towards a Formalization of UML Class Structure in PVS. Research Report 272, Department of Informatics, University of Oslo (1999)
2. Berardi D., Calì A., Calvanese D., De Giacomo G.: Reasoning on UML Class Diagrams. Technical Report, Dipartimento di Informatica e Sistemistica, Università di Roma (2003)
3. Egyed A.: Automated Abstraction of Class Diagrams. ACM Transactions on Software Engineering and Methodology **11(4)** (2002) 449–491
4. Evans A.: Reasoning with UML Class Diagrams. Second IEEE Workshop on Industrial Strength Formal Specification Techniques (WIFT98) (1998)
5. France R.: A Problem-Oriented Analysis of Basic UML Static Requirements Modeling Concepts. Proceedings of OOPSLA'99 (1999) 57–69
6. Funes A., George C.: Formalizing UML Class Diagrams. In Liliana Favre (Ed.), UML and the Unified Process, Idea Group Publishing, (2003) 129–198
7. Génova G., Llorens J., Martínez P.: The Meaning of Multiplicity of N-ary Association in UML. Software and Systems Modeling **2(2)** (2002) 86–97
8. Gogolla M., Richters M.: Equivalence Rules for UML Class Diagrams. UML'98 - Beyond the Notation, First International Workshop (1998) 87–96
9. Rumbaugh J., Jacobson I., Booch G.: The Unified Modeling Language Reference Manual, Second Edition. Addison-Wesley (2004)
10. Szlenk M.: Formal Semantics and Reasoning about UML Conceptual Class Diagram (in Polish). PhD Thesis, Warsaw University of Technology (2005)
11. Szlenk M.: Formal Semantics and Reasoning about UML Class Diagram. Proceedings of DepCoS-RELCOMEX'2006 (2006) 51–59
12. UML 2.1.1 Superstructure Specification (formal/2007-02-05). Object Management Group (2007)