

On the Interaction of Inter-Relationship Constraints

Azzam Maraee
Computer Science Department
Ben-Gurion University, ISRAEL
mari@cs.bgu.ac.il

Mira Balaban
Computer Science Department
Ben-Gurion University, ISRAEL
mira@cs.bgu.ac.il

ABSTRACT

MDE and software evolution call for model-level design support, that includes reasoning capabilities such as query answering, verification and validation, static analysis and model transformation. Automation of all activities requires well-defined semantics for models. This is particularly important for the class diagram model, which is central in UML. However, since UML specification is verbal and imprecise, the exact meaning of many class diagram constructs and their interaction is still obscure. There are major problems with the inter-association constraints *subsets*, *union*, *redefinition*, *association specialization*, *association-class specialization* and *XOR*. Although their standard semantics is ambiguous and their interaction unclear, the UML meta-model intensively uses these constraints. Moreover, some of these interactions have been declared in the UML meta-model as *variation points*.

The paper discusses possible semantics of the inter-association constraints *subsets*, *union*, *redefinition* and *association-class specialization*, analyzes their interaction, and suggests coherent semantics that minimizes contradictions with the meta-model (*association specialization* and *XOR* are left out, due to space limitations). The paper also introduces rules that enforce model correctness. This paper is the first to provide an inclusive analysis of all inter-association constraints.

Categories and Subject Descriptors

D.2.2 [Software Engineering]: Design Tools and Techniques—*Computer-aided software engineering (CASE) Object-oriented design methods*.

General Terms

Design, Languages, Reliability, Verification.

Keywords

Inter-association constraints, semantics, subsetting, union, redefinition, association-class hierarchy, design, consistency,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...\$10.00.

finite-satisfiability, incomplete-design, correctness rule.

1. INTRODUCTION

With the emergence of UML 2 [18], several new concepts regarding the relationships between associations have been added: Property subsetting, property union and property redefinition. These concepts, together with previous ones (association specialization, association-class hierarchies and xor) provide an important mechanism for defining new modeling languages [1]. They have been widely used in defining the UML 2 meta-model. Nevertheless, their semantics is ambiguous, their inter-relationships obscure, and some constraints have contradictory interpretations.

The problems of inter-association constraints have been analyzed in several works. Subsetting, redefinition, association specialization and union have been studied in [1, 11, 12, 13, 2, 16, 7]. These works try to settle various semantic issues. Yet, an overall consideration of all constraints, including association-class hierarchy, is still missing.

This paper analyzes the inter-association constraints and suggests coherent semantics. Our aim is to minimize contradictions with the meta-model. The paper studies interactions with other class diagram constraints and suggests rules that enforce model correctness. To the best of our knowledge, the full picture of interaction between the constraints has not been studied before. Due to space limitation, the paper includes only the analysis of *subsets*, *union*, *redefinition*, and *association-class specialization*. This work is part of our overall work on correctness of class diagrams [15, 4, 3], which requires a well-defined semantics for all constructs.

Section 2 formally defines the UML class diagram. Section 3 studies subsetting, Section 4 studies union; Section 5 studies redefinition; Section 6 studies association-class hierarchy and Section 7 concludes the paper.

2. BACKGROUND

A class diagram is a structural abstraction of a real world phenomenon. The model consists of basic elements, descriptors and constraints. This section defines the abstract syntax and the semantics of the class diagram (subset). Our definitions follow the works of [20, 19, 17, 1]. We use a *property oriented* abstract syntax, since it enables a better formulation of the semantics of inter-association constraints.

2.1 Abstract syntax

The subset of UML2.0 class diagrams considered in this paper includes classes, associations, and four kinds of constraints: Multiplicity constraints on binary associations, class

hierarchy constraints, inter-association constraints, and Generalization set (GS) constraints. Attributes are omitted.

A *class diagram* is a tuple $(\mathcal{C}, \mathcal{A}, \mathcal{P}, \text{props}, \text{source}, \text{target}, \mathcal{Q}, \mathcal{QD}, \mathcal{Qdom}, \text{Constraint})$ where

- \mathcal{C} is a set of *class* symbols.
- \mathcal{A} is a set of *association* symbols.
- \mathcal{P} is a set of *property* symbols (sometimes called *association end*). Property symbols denote mappings derived from their associations.
- $\text{props}: \mathcal{A} \rightarrow \mathcal{P} \times \mathcal{P}$ is a 1:1 and onto assignment of (unique) properties to association symbol. For a property p , there is a unique $a \in \mathcal{A}$, such that $\text{props}(a) = (p, *)$ or $\text{props}(a) = (*, p)$, where $*$ is a wild card. We write $\text{assoc}(p)$ or $\text{assoc}(p_1, p_2)$ for the association of p or of (p_1, p_2) , and $\text{props}_1(a), \text{props}_2(a)$ for the two properties of a .
- $\text{target}: \mathcal{P} \rightarrow \mathcal{C}$ and $\text{source}: \mathcal{P} \rightarrow \mathcal{C}$ are 1:1 mappings of properties to classes such that for an association a with $\text{props}(a) = (p_1, p_2)$, $\text{target}(p_1) = \text{source}(p_2)$ and $\text{target}(p_2) = \text{source}(p_1)$. In Figure 1a, $\text{target}(p_1) = \text{source}(p_2) = C_1$ and $\text{source}(p_1) = \text{target}(p_2) = C_2$.

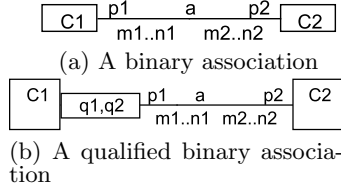


Figure 1: Regular and qualified binary associations

- \mathcal{Q} : A set of qualifier attribute symbols (shortly *q-attributes*).
- \mathcal{QD} : A set of domains (sets), termed *q-attribute domains*. Each domain has a known cardinality.
- $\mathcal{Qdom}: \mathcal{Q} \rightarrow \mathcal{QD}$: A mapping of q-attributes to q-attribute domains.
- *Constraint* is a set of *constraints* as follows:
 1. *Multiplicity constraints on binary associations*: $\text{mul}: \mathcal{P} \rightarrow (\mathbb{N} \cup \{0\}) \times (\mathbb{N} \cup \{*\})$ assigns multiplicity constraints to property symbols. For simplicity we use a compact symbolic representation, where association a in Figure 1a is denoted $a(p_1 : C_1[m_1, n_1], p_2 : C_2[m_2, n_2])$. The functions $\text{minMul}: \mathcal{P} \rightarrow \{\mathbb{N} \cup \{0\}\}$ and $\text{maxMul}: \mathcal{P} \rightarrow \{\mathbb{N} \cup \{*\}\}$ give the minimum and maximum multiplicities assigned to a property, respectively.
 2. *Qualifier constraints on binary associations*:
 1. $\text{q-att}: \mathcal{P} \rightarrow \mathcal{Q}$ is a multi-valued function that assigns a (possibly empty) set of q-attribute symbols to a property. The assigned property is called a *qualified property* and its association is a *qualified association*.
 2. $\text{q-mul}: \mathcal{P} \times 2^{\mathcal{Q}} \rightarrow (\mathbb{N} \cup \{0\}) \times (\mathbb{N} \cup \{*\})$ is a partial function that assigns multiplicity constraints to pairs of qualified property symbols and

their q-attributes. $\text{q-mul}(p, \{q_1, \dots, q_n\})$ is defined only for $p \in \mathcal{P}$, and $q_i \in \mathcal{Q}, i = 1..n$, such that $\text{q-att}(p) = \{q_1, \dots, q_n\}$. The functions $\text{q-min}: \mathcal{P} \times 2^{\mathcal{Q}} \rightarrow (\mathbb{N} \cup \{0\})$ and $\text{q-max}: \mathcal{P} \times 2^{\mathcal{Q}} \rightarrow (\mathbb{N} \cup \{*\})$ give the minimum and maximum multiplicities assigned by q-mul , respectively. In Figure 1b, $\text{q-att}(p_2) = \{q_1, q_2\}$ and $\text{q-mul}(p_2, \{q_1, q_2\}) = (m_2, n_2)$. Note, that the visualization is on the opposite side of the qualified property. The symbolic representation of the qualified association is:

$a(p_1 : C_1[m_1, n_1], p_2 : C_2\{q_1, \dots, q_n\}[m_2, n_2])$, where $\text{q-att}(p_2) = \{q_1, \dots, q_n\}$.

3. *Class hierarchy*: A binary relationship \prec on the set of class symbols: $\prec \subseteq \mathcal{C} \times \mathcal{C}$. Henceforth we use the notation $C_2 \prec C_1$, where C_1 is the *superclass* and C_2 is the *subclass* (also called *direct descendant*). The transitive reflexive closure of \prec is called the *descendant* relation, and denoted \prec^* . Its irreflexive version is denoted \prec^+ .
4. *Generalization set (GS) constraints*: GS is an $(n+1)$ -ary relationship on \mathcal{C} , for $n \geq 2$. An element $\langle C, C_1, \dots, C_n \rangle$ in GS must satisfy: For $i, j = 1, \dots, n$ (1) $C \neq C_i$; (2) $C_i \neq C_j$; (3) $C_i \prec C$.

C is called the *superclass* and the C_i -s are called the *subclasses*. A GS tuple is associated with a *constraint* $\text{const} \in \{\langle \text{disjoint} \rangle, \langle \text{overlapping} \rangle, \langle \text{complete} \rangle, \langle \text{incomplete} \rangle, \langle \text{disjoint, complete} \rangle, \langle \text{disjoint, incomplete} \rangle, \langle \text{overlapping, complete} \rangle, \langle \text{overlapping, incomplete} \rangle\}$. We use the symbolic representation $GS(C, C_1, \dots, C_n; \text{const})$ for GS constraints.

5. *Aggregation and Composition*: Two unary relationships on the set of property symbols denoted by p^a and p^c respectively. Visually, composition is shown by a filled diamond adornment on the composite association end while aggregation is shown as an open diamond.

2.2 Semantics

The standard set theoretic semantics of class diagrams associates a class diagram with *instances* I , that have a semantic domain and an *extension mapping*, in which class extensions are sets of objects in the semantic domain, association extensions are relationships among class extensions, and q-attribute-domain extensions are data types. For a symbol x , x^I is its denotation in I . The elements of class extensions are called *objects*, the elements of association extensions are called *links* and the elements of data types are called *values*. The semantics of the constraints with respect to an instance I is defined as follows:

Property: For a property p , p^I is a multi-valued function from its source class to its target class: $p^I: \text{source}(p)^I \rightarrow \text{target}(p)^I$. p^I is restricted by the multiplicity constraints: For every $e \in \text{source}(p)^I$, $\text{minMul}(p) \leq |p^I(e)| \leq \text{maxMul}(p)$. The upper bound constraint is ignored if $\text{maxMul}(p) = *$.

Association: For an association a with $\text{props}(a) = (p_1, p_2)$, p_1^I and p_2^I are restricted to be inverse functions of each other. That is, $p_1^I = (p_2^I)^{-1}$. The association denotes all object pairs that are related by its properties. That is, $a^I = \{(e, e') \mid e \in \text{target}(p_1)^I, e' \in \text{target}(p_2)^I, p_2^I(e) = e'\}$.

Qualifier: A qualified property denotes a mapping that refines the denotation mapping of the property. If $\text{q-att}(p) =$

$\{q_1, \dots, q_n\}$, and $Qdom(q_i) = D_i$, $i = 1..n$, then the mappings of p^I are partitioned by a function $q_map : source(p)^I \times D_1 \times \dots \times D_n \rightarrow target(p)^I$ that satisfies:

1. For every $e \in source(p)^I$ and $d_i \in D_i$, $i = 1..n$: $q_min(p, q_1, \dots, q_n) \leq |q_map(e, d_1, \dots, d_n)| \leq q_max(p, q_1, \dots, q_n)$.
2. q_map is a partition of p^I : For every $e \in source(p)^I$ and $d_i \in D_i$, $i = 1..n$, $q_map(e, d_1, \dots, d_n) \subseteq p^I(e)$.
3. For every $e \in source(p)^I$ and $d_i, d'_i \in D_i$, $i = 1..n$: $q_map(e, d_1, \dots, d_n) \cap q_map(e, d'_1, \dots, d'_n) = \emptyset$.

Aggregation and composition: An **aggregation** denotes two constraints: 1) *Irreflexivity*: For $\mathbf{e} \in source(p^a)^I$, $p^a(\mathbf{e}) \neq \mathbf{e}$; 2) *Transitivity on the aggregation relation*. Together, these constraints imply antisymmetry:

For aggregation properties p_1^a, \dots, p_n^a , such that $target(p_i^a) = source(p_{i+1}^a)$, $i = 1, n-1$, if $\mathbf{e} \in source(p_1^a)^I$, then $p_n^a(p_{n-1}^a(\dots(p_1^a(\mathbf{e})))) \neq \mathbf{e}$.

A **composition** is an aggregation with two additional constraints: 1) A composition property p^c is not multi-valued; 2) *multi-composition* constraint: For composition properties p_1^c, p_2^c such that $source(p_1^c) = source(p_2^c)$, if $\mathbf{e} \in source(p_1^c)^I$, then $p_1^c(\mathbf{e}) = p_2^c(\mathbf{e})$.

Class hierarchy constraints: Class hierarchy constraints denote subset relations between the extensions of the involved classes. That is, for $C_1 \prec C_2$, $C_1^I \subseteq C_2^I$. *GS* constraints have the following meaning: *disjoint* : $C_i^I \cap C_j^I = \emptyset, \forall i, j$; *overlapping* : For some i, j , it might be $C_i^I \cap C_j^I \neq \emptyset$; *complete*: $C^I = \bigcup_{i=1}^n C_i^I$; *incomplete*: $\bigcup_{i=1}^n C_i^I \subseteq C^I$.

According to the official semantics of the *incomplete* and *overlapping* constraints [18] they are "non-constraints", in the sense that they do not impose any restriction: The subclasses might cover or not and might be overlapping or not, respectively. The semantics adopted by the this paper, requires that these constraints are properly satisfied in at least one legal instance.

2.3 Verification of Class Diagrams

Class diagrams are models written by people, and therefore, usually suffer from modeling problems like inconsistency, redundancy, and abstraction errors. The two major correctness requirements imposed on class diagrams are *consistency* and *finite satisfiability*. *Consistent* means that the diagram has a legal instance, i.e., its constraints are not contradictory [5]. A *finite satisfiability* means that it is a non-empty and finite legal instance [14, 15, 3].

In view of the wide spread usage of UML class diagrams and the difficulties of producing high quality models, it is essential to equip UML CASE tools with reasoning capabilities like verification of correctness problems, model transformation and redundancy checking [5, 4, 9]. The quality of models is especially important for the emerging Model Driven Engineering (MDE) approach, in which software is developed by repeated transformations of models. Current management tools for class diagrams provide editing, syntax and code generation (class skeleton) services, but do not support necessary meta-modeling and MDE activities. Development of verification of MDE activities requires well-defined semantics for class diagrams. The current set theoretic semantics provided by specification does not gives accurate definition for the inter-relationship constraints. The

next sections provide an accurate semantics for these constraints with discussion of their interactions with the later constraints.

3. SUBSETTING CONSTRAINT

Subsetting is a constraint defined between two properties (association ends). It constrains set of instances that populate an association end (the *subsetting* end) to be a subset of the set of instances that populates the other association end (the *subsetting* end). Consider the class diagram in Figure 2. The subsetting constraint on the association end *presentation* states that the papers presented by an author are included in the papers written by that author.

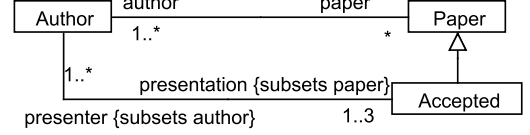


Figure 2: A class diagram with subsetting constraints

Syntax: Subsetting defines a binary relationship, denoted \prec , on the set of property symbols: $\prec \subseteq \mathcal{P} \times \mathcal{P}$. $p_2 \prec p_1$, stands for “ p_2 subsets p_1 ”.

Semantics: Let $p_1, p_2 \in \mathcal{P}$, $source(p_2) = C_1$ and $p_2 \prec p_1$. Then, for every entity $\mathbf{e} \in C_1^I$, $p_2^I(\mathbf{e}) \subseteq p_1^I(\mathbf{e})$.

UML 2 well-formedness rules: $p_2 \prec p_1$, imposes three restrictions: (1) $source(p_2) \prec^* source(p_1)$, i.e., $source(p_2)$ is either $source(p_1)$ or its descendant (direct or not); (2) $target(p_2) \prec^* target(p_1)$; (3) $maxMul(p_2) \leq maxMul(p_1)$.

[1, 10] show that subsetting is symmetric with respect to its association. That is, the existence of a subsetting constraint on one association end entails the existence of a subsetting constraint on the opposite association end. Formally:

Correctness rule (Subsetting incomplete design) Let $p_1, p_2 \in \mathcal{P}$. If $p_2 \prec p_1$ then **add** $p_2^{-1} \prec p_1^{-1}$.

3.1 Interaction of Subsetting with Other Constraints

The UML meta-model does not constrain the interaction between the subsetting constraint with other constraint, although such interaction can cause design problems like finite satisfiability, consistency and incomplete design. In this subsection, we analyze such interactions, and suggest correctness rules. Each correctness rule is related to a particular type design problem. These rules can enhance the *well formedness rules* of the meta-model.

Interaction of subsetting and qualifier constraints.

The restriction that meta-model imposes on the multiplicity constraint of the subsetting property, does not take qualified association ends into consideration. In Figure 3, every instance \mathbf{e} of TV-Network on a given day \mathbf{d} must be related to a broadcast schedule (that is, $|q_map(\mathbf{e}, \mathbf{d})| = 1$, see the qualifier’s semantics in Section 2). Therefore, \mathbf{e} must be related to seven broadcast schedules for all seven days: $|bcsch^I(\mathbf{e})| = 7$. On the other hand, the subsetting constraint requires $bcsch^I(\mathbf{e}) \subseteq contents^I(\mathbf{e})$. But, $|contents^I(\mathbf{e})| = 1$. This contradictory creates a consistency problem. It is caused by the implicit multiplicity constraint

of the qualifier, which is not taken care of by the UML well-formedness rules. The following rule prevents this problem.

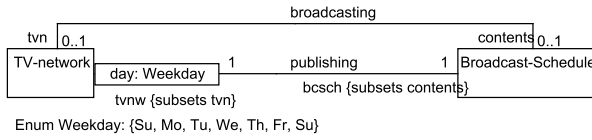


Figure 3: A subsetting with a qualifier constraint

Correctness rule (Subsetting-qualifier consistency) Let $a(p_1 : C_1[m_1, n_1], p_2 : C_2\{q_1, \dots, q_n\}[m_2, n_2])$ be a qualified association, where $Dom(q_i) = D_i$ and let $|D_i| = k_i$. Then,

- For each property $p \prec^+ p_2$, if $k_i \neq \infty$, $i = 1, n$, then $maxMul(p) \leq n_2 \cdot \prod_{i=1}^n k_i$.
- For each property p such that $p_2 \prec^+ p$, $maxMul(p) \geq n_2 \cdot \prod_{i=1}^n k_i$.

In Figure 3, $maxMul(contents) = 1 < 1 \cdot |\{Su, Mo, Tu, We, Th, Fr, Su\}| = 1 \cdot 7$ which violates the above rule (part).

Interaction of subsetting and aggregation/composition constraints.

Figure 4.a presents a problem of composition cycles (noticed by [1, 10]): In every legal instance I , for $e \in D^I$, $p_1^I(q_2^I(e)) = e$. The following rule prevents this inconsistency:

Correctness rule (Subsetting-composition consistency) Let $p_1, p_2 \in \mathcal{P}$, and $p_2 \prec^+ p_1$. Then, p_1 is an aggregation (composition) property, i.e. $p_1^a(p_1^c)$, iff p_2 is not an aggregation (composition) property.

Figure 4b presents a problem of incomplete design that enables the following scenario: For $e \in C^I$, $p_2^I(e) \neq p_1^I(e)$, which violates the multi-composition constraint. The multi-composition constraint can be enforced by adding the subsetting constraint $p_2 \prec q_2$.

Correctness rule (Subsetting-composition incomplete design) Let $p_1, p_2 \in \mathcal{P}$, and $target(p_2) \prec^* target(p_1)$. Then if $minMul(p_1) > 0$ then **add** $p_2 \prec p_1$.

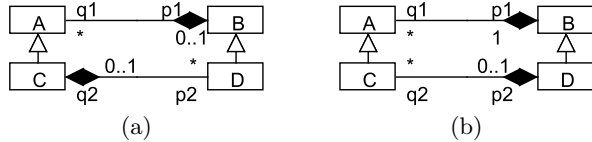


Figure 4: A composition with subsetting constraints

Interaction of subsetting and GS constraints.

Consider the class diagram in Figure 5a. Let $e \in A_1^I$ and $e \in A_2^I$. Then $|b_1^I(e)| = 1$ and $|b_2^I(e)| = 1$, and $B_1, b_1^I(e) \cap b_2^I(e) = \emptyset$, by the disjointness constraint on of B_1 and B_2 . Since $b_i \prec b$, $i = 1, 2$, $b_1^I(e) \cup b_2^I(e) \subseteq b^I(e)$. Therefore $|b^I(e)| = 2$, which violates the multiplicity constraint on b . We suggest to consider this situation as an incomplete design, missing a disjointness constraint on classes A_1 and A_2 , as in Figure 5b.

Correctness rule (Subsetting-GS-constraint incomplete design) Let $p, p_1, \dots, p_n \in \mathcal{P}$, $GS(target(p), target(p_1), \dots, target(p_n); disjoint)$, and $p_i \prec p$, $i = 1, n$. Then, if (1)

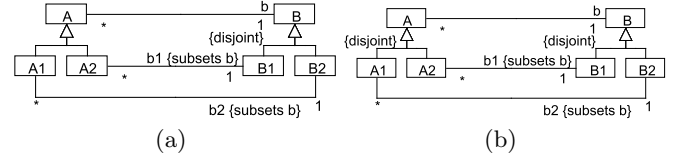


Figure 5: Interaction of *subsetting* with disjoint constraint

For each p_i , $minMul(p_i) = maxMul(p)$; or (2) For each $i \neq j$, $minMul(p_i) + minCard(p_j) \geq maxMul(p)$; then **add** $GS(source(p), source(p_1), \dots, source(p_n); disjoint)$.

Indeed, Figure 5a satisfies condition (2).

Interaction of subsetting and multiplicity constraints.

Figure 6 presents a finite satisfiability problem due to interaction of subsetting and multiplicity constraints. In every legal instance I , for $e \in C^I$, $|b_3^I(e)| = 2$, and also $b_3^I(e) \subseteq b_1^I(e)$. But, the upper association cycle r, q dictates $|b_1^I(e)| = 1$.

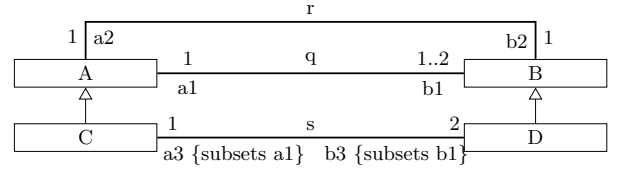


Figure 6: A finite satisfiability problem due to interaction between subsetting and multiplicity constraints

In Figure 6, the upper association cycle by itself does not present any problem, since the maximum multiplicity 2 on the b_1 association end is never realized (we say that the b_2 multiplicity constraint is not *tight*). The problem is caused just by the interaction of the subsetting and the non-tight multiplicity constraint. The following correctness rule formalizes the problematics caused by this interaction:

Correctness rule (Subsetting-multiplicity finite satisfiability) Let $p_1, p_2 \in \mathcal{P}$, $p_2 \prec^+ p_1$ such that $maxMul(p_1)$ is not tight, i.e., there is no legal finite instance I and an object $e \in source(p_1^I)$, such that $|p_1^I(e)| = maxMul(p_1)$. Then if $maxMul(p_2) = maxMul(p_1)$ then there is a finite satisfiability problem.

This correctness rule is not purely syntactic, since it relies on the identification of tight multiplicity constraints, which is still an open problem (that we intend to investigate).

4. DERIVED UNION CONSTRAINT

A property may be marked as being a *derived union*, meaning that its value set is the union of the value sets of the properties that subset it [18]. In Figure 7, the declaration of the association end *course* as derived union, implies that a student can only register to required or elective courses. Dropping the union constraint enables students to register to a course which is neither required nor elective.

Syntax: Union is a constraint on the set of property symbols, denoted p^u . p^u is termed a *union property*.

Semantics: Let $p^u \in \mathcal{P}$, $A \in \mathcal{C}$, and $A \preceq source(p)$. Then, in every legal instance I , for every object $e \in A^I$, $p^{uI}(e) = \bigcup_{p' \prec^+ p^u} p'^I(e)$.

Similarly to the subsetting constraint, union is also sym-

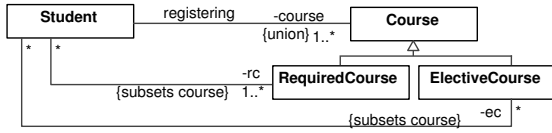


Figure 7: A class diagram with a union constraint

metric (rule is omitted), and we present only one constraint interaction due to space limitations.

Interaction of union and multiplicity constraints.

For a union property p^u , the sum of the minimum multiplicity constraints of its subsetting properties cannot be smaller than the minimum multiplicity constraint of p^u . This rule is a direct result of the union semantics.

Correctness rule (Union-multiplicity consistency) Let $p^u \in \mathcal{P}$. Then $\sum_{p' \prec^+ p^u} \minMul(p') \geq \minMul(p^u)$.

Figure 8 is inconsistent since it contradicts the Union-multiplicity constraint rule:

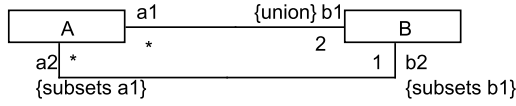


Figure 8: A consistency problem due to interaction of union and multiplicity constraints

5. PROPERTY REDEFINITION CONSTRAINT

Property redefinition is a constraint that enables a redefinition of property characteristics like *name*, *time*, *visibility*, *multiplicity* [21, 18]. In this work we focus on redefinition of type and multiplicity constraints.

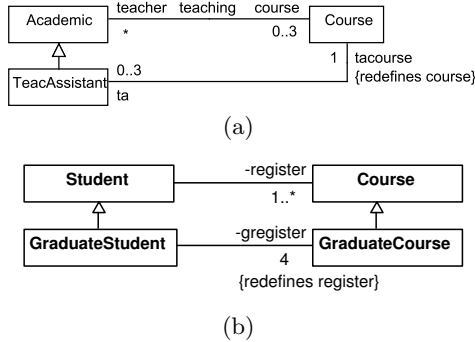


Figure 9: Class diagrams with redefinition constraints

In Figure 9a, the association end *tacourse* redefines the multiplicity constraint of the *course* association end. It restricts a teaching assistant to teach exactly one course, instead of up to three courses. In Figure 9b, the association end *gregister* redefines the type (and the multiplicity) constraint of the *register* association end. It restricts a graduate student to register to exactly four graduate courses, instead of to any number of general courses.

Syntax: A redefinition is a binary constraint, denoted by \triangleright , on the set of property symbols. Henceforth, we use the

notation $p_2 \triangleright p_1$, where p_2 is the *redefining* property and p_1 the *redefined* property.

Semantics:

The semantics of redefinition, as defined in the meta-model, is vague and ambiguous. Three possible semantics have been discussed [12, 1, 13, 2, 7, 16]. We briefly analyze each semantics and choose the one that minimizes contradictions with the meta-model.

Consider the class diagram in Figure 10, where the star on the class hierarchy between A_2 and B_2 denotes $B_2 \prec^* A_2$ ¹.

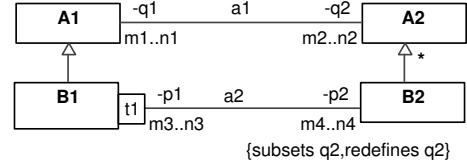


Figure 10: Class diagrams with redefinition constraints

- One association semantics** [10, 16]: Redefinition is understood as a constraint on the redefined property, while the association of the redefining property is syntactically ignored. The constraint for Figure 10: For a legal instance I ,

- For each $e \in A_1^I$ and $e \notin B_1^I$: $q_2^I(e) \in A_2^I$ and $m_2 \leq |q_2^I(e)| \leq n_2$.
- For each $e \in B_1^I$ (and of course $e \in A_1^I$): $q_2^I(e) \in B_2^I$ and $m_4 \leq |q_2^I(e)| \leq n_4$.

This semantics ignores the opposite property of the redefining property (i.e. p_1 in Figure 10), including its possible associated constraints, like multiplicity, qualifier and composition. In addition, it ignores a possible subsetting constraint on the redefining property (as in Figure 10). Note that a combination of redefinition and subsetting on the same property appears in the meta-model ([18], p. 307).

- Covariant semantics** [12, 13]: Redefinition excludes the source of the redefining property from the source of the redefined property. The constraint for Figure 10: For a legal instance I , q_2^I is defined only on $A_1^I - B_1^I$:

$$q_2 : A_1^I - B_1^I \rightarrow A_2^I$$

Under this semantics, subsetting and redefinition contradict each other. That is, a redefining property cannot subset the same property, as in Figure 10. As noted above, the meta-model includes such combinations. In addition, [8, 12] note a type safety problem that results from covariant subtyping. This semantics also raises a conceptual problem, as it neglects the inter-relationship between the associations of the redefined and redefining properties.

- Subsetting semantics** [2, 7]: Redefinition strengthens the subsetting constraint with additional constraints. The constraint for Figure 10:

- $q_2^I / B_1^I = p_2^I$. That is, for $e \in B_1^I$, $q_2^I(e) = p_2^I(e)$.

¹This is an extended class diagram notation used in our class diagram pattern catalog www.cs.bgu.ac.il/~umlc.

b. For each $e \in B_1^I : q_2^I(e) \subseteq B_2^I$ and $m_4 \leq |q_2^I(e)| \leq n_4$.

Under this semantics, redefinition entails subsetting (which require only $q_2^I/B_1^I \subseteq p_2^I$).

The interaction between subsetting and redefinition constraints has been declared a *semantic variation point* in the meta-model ([18], p. 41), although the meta-model itself contains this kind of interaction. The subsetting semantics solves this *variation point*, and hence, minimizes necessary modifications in the meta-model.

The correctness rules of redefinition are similar to those of subsetting. We present only few new rules.

Interaction of redefinition, subsetting, union and disjoint GS constraints.

Let \triangleright^+ denote the transitive closure of \triangleright .

Correctness rule (Redefinition-subsetting-union incomplete design) Let $p_1, p_2, p_3 \in \mathcal{P}$, then:

1. If $(p_2 \triangleright^+ p_1)$ then **add** $p_2 \prec p_1$.
2. $(p_3 \triangleright^+ p_1) \wedge (p_3 \prec^+ p_2) \wedge (p_2 \triangleright^+ p_1)$ then **add** $p_3 \triangleright p_2$.
3. $(p_3 \triangleright^+ p_1) \wedge (p_3 \prec^+ p_2) \wedge (p_2 \prec^+ p_1)$ then **add** $p_3 \triangleright p_2$.

Correctness rule (Redefinition-subsetting-union-disjoint incomplete design) Let $p_1 \prec^+ p^u$, where p^u is a union property. Then, if for each property $p' \prec^+ p$, $source(p_1) \prec^* C_1$ and $source(p') \prec^* C'$, for some “disjoint” C_1, C' (i.e., there is a GS($C, \dots, C_1, \dots, C' \dots$; disjoint) constraint), then **add** $p_1 \triangleright p$.

6. ASSOCIATION-CLASS HIERARCHY

An association-class is a class associated with single association in a way that identifies the class objects with the association links in a 1:1 manner [21]. Figure 11a illustrates a association-class *Authorization* for the *authorization* association.

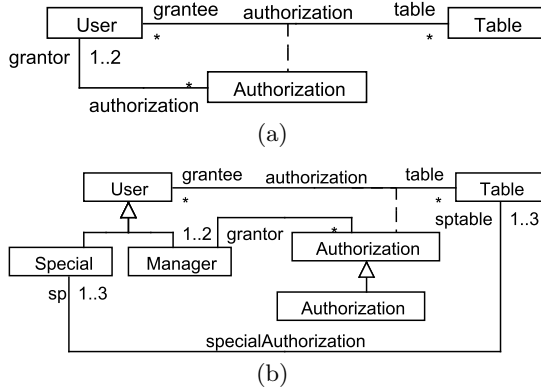


Figure 11: Association Class

Syntax of association-classes: The class diagram syntax is extended with \mathcal{AC} , a set of association-class symbols, $\mathcal{C} \cap \mathcal{AC} = \emptyset$, and a 1:1 mapping $ac: \mathcal{AC} \rightarrow \mathcal{A}$.

Semantics of association classes: Association-classes are classes whose objects are identified by object-pairs of their associations. For an association-class C and a legal instance I , there exists a 1:1 and onto function: $ac_{C,I}: C^I \rightarrow ac(C)^I$.

6.1 Semantics of Association-Class Hierarchy

Association-class hierarchy is frequently used in ontologies and in the translation of description logics to class diagrams. However, the semantics of association-class hierarchy is not given much attention in the literature, and there is no specific definition in the meta-model besides a single small reference: “the specialization and refinement rules defined for class and association are also applicable to association-class”.

This definition gives rise to multiple interpretations, which further complicates the analysis of the interaction with the other inter-association constraints, subsetting, redefinition and association-specialization. Following are three possible interpretations:

1. **Subsetting semantics:** Association-class hierarchy is a class hierarchy that induces a subsetting constraint between the involved associations. This semantics allows a single object pair of the super association-class to identify two different objects in the super association. In Figure 12a, the pair (sp_1, t_1) identifies the objects aut_2 and $saut_1$ in contradiction to the intended semantics.
2. **Regular-Class-Hierarchy semantics:** This semantics ignores the involved associations. Figure 12b shows an object of association-class *SpecialAuthorization* that is identified by two different object-pairs, in contradiction to the intended semantics. The *USE* system adapts this semantics².
3. **Unified-Mapping semantics:** This semantics constrains the subsetting semantics to have identical mappings. For association-classes $A \prec B$ and a legal instance I , $ac_{A,I} = ac_{B,I/A}$: For each object $e \in A^I$, $ac_{A,I}(e) = ac_{B,I}(e)$.

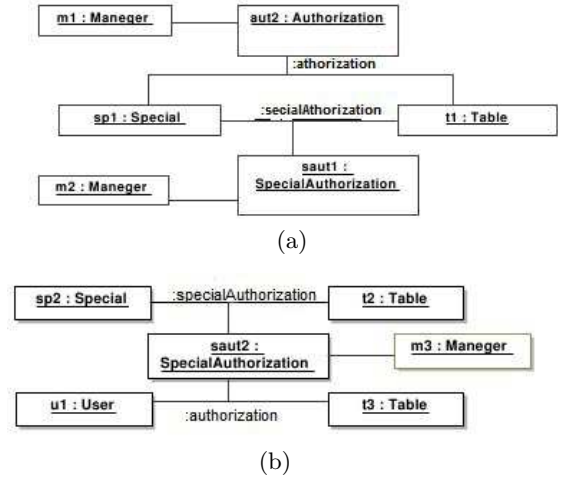


Figure 12: Possible instances of the association-class hierarchy in Figure 11b

Henceforth, we adopt the Unified-Mapping semantics for hierarchy of association-classes.

²<http://www.db.informatik.uni-bremen.de/projects/USE/>

Property Correspondence.

Association-class hierarchy raises a unique problem of *property correspondence*, between the properties of the involved associations. Figures 13b 13c present two legal instance diagrams of Figure 13a. In both instances, m_1 is a *manager* and sp_1 is a *special*. However, in Figures 13b, m_1 is a grantor and sp_1 is a grantee, while in Figure 13c m_1 is a grantee and sp_1 is a grantor. The problem is that the association-class hierarchy semantics does not specify correspondences between the properties of the sub-association and the super-association. Therefore, the syntax of association-class hierarchy cannot be a plain class hierarchy syntax, but must include specification of property correspondence as a sub-setting constraint. For example:

SpecialAuthorization \prec *Authorization*
with *manager* \prec *grantor*
sp \prec *grantee*

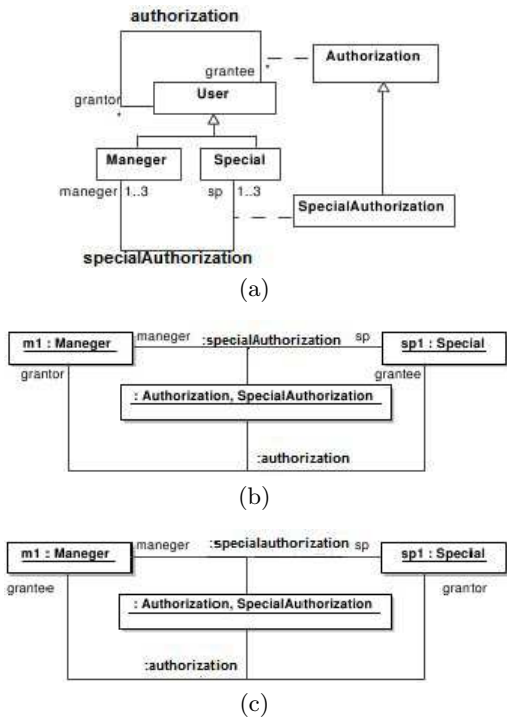


Figure 13: A reflexive association-class

In sum, the suggested syntax for association-class hierarchy involves subsetting constraints between the corresponding properties:

$$A \prec B \text{ with } p_A \prec p_B \\ q_A \prec q_B$$

6.2 Interaction of Association-Class Hierarchy with Other Constraints

This section presents two rules of incomplete design, that result from the interaction of association-class hierarchy with generalization-set and redefinition constraints.

Interaction of association-class hierarchy with GS constraints.

Consider Figure 14. It presents a complex situation of a GS-constraint on association-classes.

1. If $cons1 = disjoint$, then in every legal instance I , Q^I and W^I are disjoint since $ac_{W,I} = ac_{R,I/W}$ and $ac_{Q,I} = ac_{R,I/Q}$. Therefore, the GS constraint $GS(R, W, Q; disjoint)$ is entailed (i.e. $cons2 = disjoint$).
2. If $cons2 = complete$ and $m_2 > 0$ then in every legal instance I , every object $e \in A^I$ participates in r^I . Since $cons2 = complete$, the mappings $ac_{W,I}$ and $ac_{Q,I}$ cover the $ac_{R,I}$ mapping of R . Therefore, the relations w^I, q^R include all pairs of r^I , implying $A^I \subseteq A_1^I \cup A_2^I$.

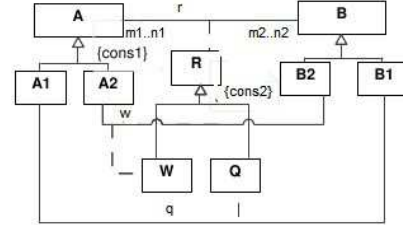


Figure 14: An association-class hierarchy

This situation is generalized in the following correctness rule:

Correctness rule (Association-class-hierarchy-GS incomplete design) Let R, R_1, \dots, R_n be association-classes such that $R_i \prec^+ R, i = 1, n$, and $ac(R) = r$ is an association between classes A and B , and $ac(R_i) = r_i$ is an association between A_i and B_i , where $A_i \prec^+ A$ and $B_i \prec^* B$. Then

1. If the diagram includes $GS(A, A_1, \dots, A_n; disjoint)$, then **add** $GS(R, R_1, \dots, R_n; disjoint)$.
2. If the diagram includes $GS(R, R_1, \dots, R_n; complete)$ and the minimum multiplicity constraint on the B end of r is greater than zero, then **add** $GS(A, A_1, \dots, A_n; complete)$.

Interaction of association-class hierarchy with redefinition constraints.

Consider the class diagram in Figure 15a. In every legal instance I , the restriction of property p_2^I to C^I equals property q_2^I , since q_2 redefines p_2 . Therefore, the instance in Figure 15b is a legal instance. Yet, it includes two different objects of the association-classes $R = ac(assoc(p_2))$, and $Q = ac(assoc(q_2))$. That is, although the associated object pairs are identical on their restriction to the subclasses, they are identified by different objects of the association-classes. It seems that a hierarchy relation between the association-classes is missing. We suggest to consider this situation as a case of incomplete design.

Correctness rule (Association-class-hierarchy-redefinition incomplete design) Let R, Q be association-classes where $ac(R) = r, ac(Q) = q$, $props(r) = (p_1, p_2)$, and $props(q) = (q_1, q_2)$. Then if $q_2 \triangleright p_2$, then **add** $Q \prec R$ with $q_2 \prec p_2, q_1 \prec p_1$.

7. CONCLUSION AND FUTURE WORK

We presented a coherent semantics for the inter-association constraints with a maximum compatibility with the Meta-Model. The paper also addressed correctness problems that

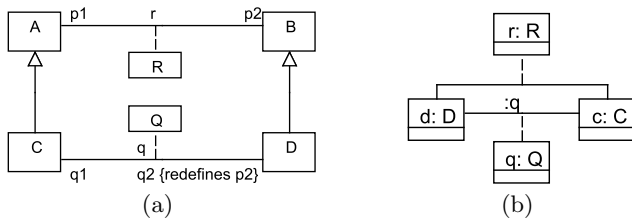


Figure 15: Redefinition with association class hierarchies

result from these constraints, and suggested correctness rules than can be added to the meta-model, as well-formedness rules.

We plan to add these rules to our catalog of class diagram correctness patterns [6]. We intend to extend the *Finite-Sat* algorithm to apply to inter association constraints and develop additional identification methods.

The complex interaction among the inter-association constraints and among the other class diagram constraints raises interesting questions concerning the overall set of constraints associated with class diagrams: Their uniform semantics and in particular their pragmatics. Undoubtedly, the current set of constraints is complex, both on the formal and the intuitive understanding levels. Modelers cannot be expected to master the complex interactions of the constraints. Therefore, the well-formedness rules should be automated and embedded in model-level IDEs. An interesting research direction involves the study of what can be termed *model complication*, and suggest appropriate model metrics.

References

- [1] M. Alanen and I. Porres. A metamodeling language supporting subset and union properties. *Software and Systems Modeling*, 7(1):103–124, 2008.
- [2] C. Amelunxen and A. Schürr. Formalising model transformation rules for UML/MOF 2. *IET Software*, 2(3):204–222, 2008.
- [3] M. Balaban and A. Maraee. Finite satisfiability of uml class diagrams with constrained class hierarchy. *Submitted*.
- [4] M. Balaban, A. Maraee, and A. Sturm. Management of correctness problems in uml class diagrams – towards a pattern-based approach. *International Journal of Information System Modeling and Design*, 1(1):24–47, 2010.
- [5] D. Berardi, D. Calvanese, and D. Giacomo. Reasoning on uml class diagrams. *Artificial Intelligence*, 168:70–118, 2005.
- [6] BGU Modeling Group. UML Class Diagram Design Pattern. <http://www.cs.bgu.ac.il/umc/>, 2010.
- [7] D. Bildhauer. On the relationships between subsetting, redefinition and association specialization. In *Ninth Conference on Databases and Information Systems*, 2010.
- [8] F. Buttner and M. Gogolla. On generalization and overriding in uml 2.0. In *UML Modeling Languages and Applications*. Springer, 2004.
- [9] J. Cabot, R. Clariso, and D. Riera. Verification of uml ocl class diagrams using constraint programming. In *IEEE International Conference on Software Testing Verification and Validation Workshop (ICSTW’08)*, 2008.
- [10] C. Costal, C. GÓMEZ, and P. Nieto. On the semantics of redefinition, specialization and subsetting of associations in uml (extended version). Technical report, Universitat Politècnica de Catalunya., 2010.
- [11] D. Costal and C. Gómez. On the use of association redefinition in uml class diagrams. In *Conceptual Modeling-ER 2006*, pages 513–527. Springer, 2006.
- [12] D. Costal and C. Gomez. On the use of association redefinition in uml class diagrams. *Lecture Notes in Computer Science*, 4215:513, 2006.
- [13] A. Kleppe and A. Rensink. On a graph-based semantics for uml class and object diagrams. In *GT-VMT 2008*. EASST, 2008.
- [14] M. Lenzerini and P. Nobili. on the satisfiability of dependency constraints in entity-relationship schemata. *Information Systems*, 15(4):453–461, 1990.
- [15] A. Maraee, V. Makarenkov, and B. Balaban. Efficient recognition and detection of finite satisfiability problems in uml class diagrams: Handling constrained generalization sets, qualifiers and association class constraints. In *MCCM08*, 2008.
- [16] P. Nieto, D. Costal, and C. Gomez. Enhancing the semantics of uml association redefinition. *Data & Knowledge Engineering*, 70 (2):182–207, 2011.
- [17] Object Management Group. *UML 2.0 Object Constraint Language Specification*, 2006.
- [18] OMG. The uml 2.0 superstructure specification. Specification Version 2, Object Management Group, 2009.
- [19] M. Richters. *A precise approach to validating UML models and OCL constraints*. PhD thesis, Universität Bremen, 2002.
- [20] M. Richters and M. Gogolla. On formalizing the uml object constraint language ocl. In *Conceptual Modeling - ER 98*, 1998.
- [21] J. Rumbaugh., G. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual Second Edition*. Addison Wesley, 2004.