Journal of Universal Computer Science, vol. 16, no. 20 (2010), 2904-2933
submitted: 7/1/10, accepted: 13/9/10, appeared: 1/11/10 © J.UCS

Validating Modal Aspects of OntoUML Conceptual Models Using Automatically Generated Visual World Structures

Alessander Botti Benevides
(Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department, Federal University of Espírito Santo (UFES), Brazil
abbenevides@inf.ufes.br)

Giancarlo Guizzardi
(Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department, Federal University of Espírito Santo (UFES), Brazil
gguizzardi@inf.ufes.br)

Bernardo Ferreira Bastos Braga
(Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department, Federal University of Espírito Santo (UFES), Brazil
bfbbraga@inf.ufes.br)

João Paulo Andrade Almeida
(Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department, Federal University of Espírito Santo (UFES), Brazil
jpalmeida@ieee.org)

Abstract: Assessing the quality of conceptual models is key to ensure that conceptual models can be used effectively as a basis for understanding, agreement and construction of information systems. This paper proposes an approach to assess conceptual models defined in OntoUML by transforming these models into specifications in the logic-based language Alloy. These Alloy specifications include the modal axioms of the theory underlying OntoUML, allowing us to validate the modal meta-properties representing ontological commitments of the OntoUML types and relations.
Key Words: Model Validation and Analysis, Knowledge Representation Formalisms and Methods, Formal Definitions and Theory
Category: I.6.4, I.2.4, D.3.1

1 Introduction

John Mylopoulos [Mylopoulos 1992] defines conceptual modeling as "the activity of formally describing some aspects of the physical and social world around us for purposes of understanding and communication". In this view, a conceptual model is a means to represent what modelers (or stakeholders represented by modelers) perceive in some portion of the physical and social world, *i.e.*, a means to express their conceptualization [Guizzardi 2005] of a certain universe of discourse.

If conceptual models are to be used effectively as a basis for understanding, agreement, and, perhaps, construction of an information system, conceptual models should express as accurately as possible a modeler's intended conceptualization. More specifically, the model should ideally describe all states of affairs that are deemed *admissible* and rule out those deemed *inadmissible* according to the conceptualization [Guizzardi 2005].

In pace with Degen *et al.* [Degen et al. 2001], we argue that "every domain-specific ontology must use as framework some upper-level ontology". This claim for an upper-level (or foundational) ontology underlying a domain-specific ontology is based on the need for fundamental ontological structures, such as theory of parts, theory of wholes, types and instantiation, identity, dependence, unity, *etc.*, in order to properly represent reality. From an ontology representation language perspective, this principle advocates that, in order for a modeling language to meet the requirements of expressiveness, clarity and truthfulness in representing the subject domain at hand, it must be an ontologically well-founded language in a strong ontological sense, *i.e.*, it must be a language whose modeling primitives are derived from a proper foundational ontology [Guarino and Guizzardi 2006, Guizzardi 2006].

An example of a general conceptual modeling and ontology representation language that has been designed following these principles is the version of UML proposed in [Guizzardi 2005]. This language (later termed OntoUML) has been constructed in a manner that its metamodel reflects the ontological distinctions prescribed by the Unified Foundation Ontology (UFO). UFO is a foundational ontology designed specially for conceptual modeling languages. The ontological categories comprising UFO are motivated by a number of theories in formal ontology, philosophical logics, cognitive science and linguistics. Moreover, formal constraints have been incorporated in OntoUML's metamodel in order to incorporate the formal axiomatization in UFO. Therefore a UML model that is ontologically misconceived taking UFO into account is syntactically invalid when written in OntoUML.

The OntoUML language has been able to provide mechanisms for addressing a number of classical conceptual modeling problems [Guizzardi et al. 2004], and the language has been successfully employed in application domains [Gonçalves et al. 2007, Oliveira et al. 2007]. However, one would certainly be naive to assume that modelers make no mistakes while constructing the models and that they fully understand the theory that supports the language. These cases could lead to ill-defined conceptual models, which may be: (i) syntactically incorrect; (ii) syntactically correct, but unsatisfiable; (iii) syntactically correct, satisfiable, but invalid according to the intended conceptualization.

Previous efforts in addressing the assessment of OntoUML models have focussed on syntactic correctness (the type (i) of ill-defined conceptual models) and led to the specification of OntoUML's syntactical constraints as OCL expressions on the language's metamodel and the building of a graphical editor [Benevides and Guizzardi 2009] that is capable of automatic syntax verification. In this paper, we go beyond syntax verification and aim at addressing the validity of OntoUML models by simulation (type (iii)).

We believe that, in general, performing validation of OntoUML models is not an easy task. Many of the ontological meta-properties incorporated into OntoUML are modal in nature and it may be difficult for human beings to reason upon the several possible changes in the instances in a set of worlds. In our previous workshop paper [Benevides et al. 2009], we have discussed an approach based on the generation and presentation of instances of OntoUML models in order to provide visualizations of the possible changes in the instances in distinct worlds. Indeed, we believe that by confronting the results of his/her specifications with the expected ones, the modeler can improve his/her confidence in the validity of the model. Here, we extend the previous version of the paper by clarifying the technical contributions of our research and by introducing a discussion on temporal interpretation, which were not accessible in [Benevides et al. 2009]. We also further elaborate on the presentation of the OntoUML language, namely, we have added a deeper discussion on relational dependence and on the treatment of part-whole relations. Moreover, we extend the approach presented there to include rigid mixin universals (categories). Here, we explain the constructs of Alloy that are used in the OntoUML to Alloy transformation. Finally, we also improve the coverage of related work.

More specifically, we discuss an approach based on formal specifications in the logic-based language Alloy [Jackson 2006] to generate instances of an OntoUML model. In our approach, the Alloy specification is fed into the Alloy Analyzer to generate an instance[1] composed of a set of objects (atoms) representing instances of the classifiers taken from the OntoUML model and a world structure that reveals the possible dynamics of object creation, classification, association and destruction. Each world in this structure represents a snapshot of the objects and relations that exist in that world. A world structure is necessary since the meta-properties characterizing most of the ontological distinctions in UFO are modal in nature. Therefore, we believe that the sequence of possible snapshots in this world structure will improve our confidence on claims of validity.

Although there are other works concerning type (iii) models (for example [Anastasakis et al. 2007, Gogolla et al. 2007]), none of them deals with ontologically well-founded conceptual modeling languages.

This article is further structured as follows. [Section 2] briefly comments on the system of modal logics employed in this article. [Section 3] presents a running

[1]In order to avoid the many overloadings of the term "model", the Alloy developers call them instances instead [Jackson 2002, p. 267].

example that is used to introduce concepts from OntoUML and Alloy languages, and also to define a transformation from OntoUML models to Alloy specifications. [Section 4] presents an illustration of a validation for the running example. [Section 4] also discusses the temporal world structure and the customization of visualization themes in the Alloy Analyzer to provide visualization mechanisms to the generated instances which we believe to be more amenable to human users. [Section 5] discusses related work. Finally, [Section 6] presents our final considerations.

2 A Note on the Quantified System of Modal Logics

Before we begin discussing the ontological distinctions behind the OntoUML system, a brief note on the modal logics employed in this article is needed. We make use here of a language L of quantified modal logics with identity. The alphabet of L contains the traditional operators of $\wedge$ (conjunction), $\neg$ (negation), $\rightarrow$ (conditional), $\leftrightarrow$ (biconditional), $\forall$ (universal quantification), $\exists$ (existential quantification), with the addition of the equality operator $=$ and the modal operators $\Box$ (necessity) and $\Diamond$ (possibility). The following holds for these two latter operators: (1) $\Diamond \mathrm{A} \overset{\text{def}}{=} \neg\Box\neg \mathrm{A}$; (2) $\Box \mathrm{A} \overset{\text{def}}{=} \neg\Diamond\neg \mathrm{A}$. Additionally, we add that the models assumed here are the so-called *normal models* [Fitting and Mendelsohn 1999], *i.e.*, the equality operator is defined between individuals in the domain of quantification in each world, and equality if it holds, it holds necessarily. In other words, the formula $\forall x, y((x = y) \rightarrow \Box(x = y))$ is valid.

A model-theoretic semantics for this language can be given by defining an interpretation function δ that assigns values to the non-logical constants of the language and a world structure S. In this language, S has a structure $\langle \mathcal{W}, \mathrm{R}, \mathcal{D} \rangle$ where $\mathcal{W}$ is a non-empty set of worlds, R represent an accessibility relation between worlds such that $\langle w, w' \rangle \in \mathrm{R}$ iff w' is accessible from w, and $\mathcal{D}$ is a function mapping worlds to non-empty domains of objects. Therefore, we are assuming here a varying domain of quantification of an *actualist* modal logics, hence, we have that in each world w, the domain of quantification $\mathcal{D}(w)$ contains only the individuals that are assumed to exist in that world. Here, unless explicitly mentioned, we take worlds to represent maximal states of affairs which can be factual or counterfactual. Informally, we can state that the truth of *formulæ* involving the modal operators can be defined such that the semantic value of formula $\Box$A is true in world w iff A is true in every world w' accessible from w. Likewise, the semantic value of formula $\Diamond$A is true in world w iff A is true in at least one world w' accessible from w.

Finally, in [Section 3], following the original formal characterization of the OntoUML language [Guizzardi 2005], we assume all worlds to be equally accessible and, as a result, we have the language of quantified modal logic QS5 with varying domain frames.

3 A Whirlwind Tour

In this section, we briefly introduce Alloy language and some concepts of UFO ontology. Then we introduce the running example and deepen the explanation of UFO and Alloy syntax by means of the running example, its corresponding OntoUML model and Alloy specification. For a complete presentation and formal characterization of OntoUML and Alloy, one should refer to [Guizzardi 2005] and [Jackson 2006], respectively.

3.1 The Logic-based Language Alloy

Alloy offers a set-based formula syntax by which one can express constraints that are amenable to a fully automatic semantic analysis [Jackson 2002, pp. 256,257]. Moreover, there is a tool, named Alloy Analyzer[2], that supports *simulation* of specifications, in which the consistency of an invariant or operation is demonstrated by generating an instance. If an Alloy specification has at least one instance, it is said to be consistent [Jackson 2002, pp. 260,267] [Jackson 2006, p. 3]. This approach is sometimes called "lightweight formal methods", because it tries to obtain the benefits of traditional formal methods, such as theorem proving techniques, at lower cost [Jackson 2006, p. XIII].

The search for instances is conducted in a space whose dimensions are specified by the user in a "scope", which assigns a bound to the number of objects of each type [Jackson 2006, p. 3]. An instance is within a scope of k if it assigns to each type a set consisting of no more than k atoms. If the analysis succeeds in finding an instance to an specification, consistency is demonstrated. Failure to find an instance within a given scope, however, does not prove that the specification is inconsistent, because, since the kernel in which Alloy is based is undecidable, it is impossible to determine automatically whether an Alloy specification is consistent [Jackson 2002, p. 267] [Jackson 2006, p. 259]. In other words, the inexistence of an instance that fits in a scope k does not imply that there is no scope larger than k in which an instance exists.

Furthermore, by constraining the search to a finite scope, the analysis of Alloy specifications is decidable, and as a SAT problem, it is NP-complete. From version four, the Alloy Analyzer translates constraints to be solved from Alloy into boolean constraints, which are fed to the SAT-based model finder Kodkod[3]. From [Jackson 2006, p. XII]:

> "As solvers get faster, so Alloy's analysis gets faster and scales to larger problems. Using the best solvers of today, the analyzer can examine spaces that are several hundred bits wide (that is, of 10^{60} cases or more)." [Jackson 2006, p. XII]

[2] `http://alloy.mit.edu/community`.
[3] `http://alloy.mit.edu/kodkod`.

Moreover, when translating Alloy specifications into boolean *formulæ*, Alloy Analyzer applies a variety of optimizations, where the most important is *symmetry breaking*. Every Alloy specification has an intrinsic symmetry given by the possibility to permute the atoms in any instance of a command, without ceasing to satisfy the Alloy specification. So, the space of assignments (possible solutions) can be divided into equivalence classes, and the solver has to search for only one assignment at each equivalence class [Jackson 2006, p. 151].

In pace with Daniel Jackson [Jackson 2002, p. 260], we believe that "simulation helps catch errors of overconstraint, by reporting, contrary to the user's intent, that no instance exists within the finite bounds of a given "scope"", or errors of underconstraint, "by showing instances that are acceptable to the specification but which violate an intended property.".

As the specification of the running example in Alloy must take into account some modal distinctions taken from UFO, then we will progressively present the Alloy syntax, by means of partial specifications of the running example, as we present some of the UFO's modal distinctions. Moreover, by showing how an OntoUML model can be specified in Alloy, we will progressively define the transformation patterns from OntoUML to Alloy.

3.2 The Ontologically Well-founded Modeling Language OntoUML

The OntoUML language is an ontologically well-founded version of the class diagram part of UML 2.0, proposed in [Guizzardi 2005], so that its metamodel reflects the ontological distinctions prescribed by UFO. Moreover, these distinctions are motivated by a number of formal meta-properties, some of which will be discussed in the sequel. Due to space limitations, we concentrate here on a fragment of the UFO ontology, with a specific focus on those distinctions that are spawned by variations in meta-properties of a modal nature. These categories are depicted in [Fig. 1] and are briefly discussed in the remainder of this section by using a running example, whose OntoUML rendering is depicted in [Fig. 2]. Since OntoUML is a modelling language whose metamodel is designed to be isomorphic to the UFO ontology, these leaf ontological distinctions for Universals in [Fig. 1] appear as modelling primitives in the language (see stereotyped classes and relationships in [Fig. 2]).

3.3 OntoUML, Alloy and the Transformation

Our example basically consists of a domain about persons, their phases in life, their biological organs, namely brains and hearts, organizations and relationships between organizations and persons. More specifically, a person must be born either a man or a woman, and must be either living or deceased. While living, a person can be said to be in a phase of child, teenager or adult. Furthermore

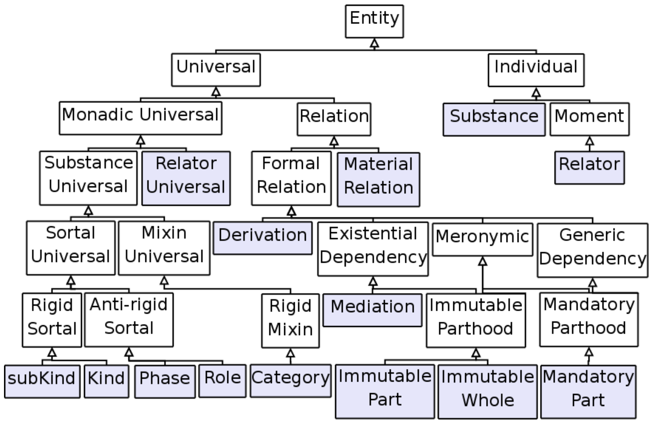


Figure 1: *UFO taxonomy excerpt [Guizzardi 2005]*

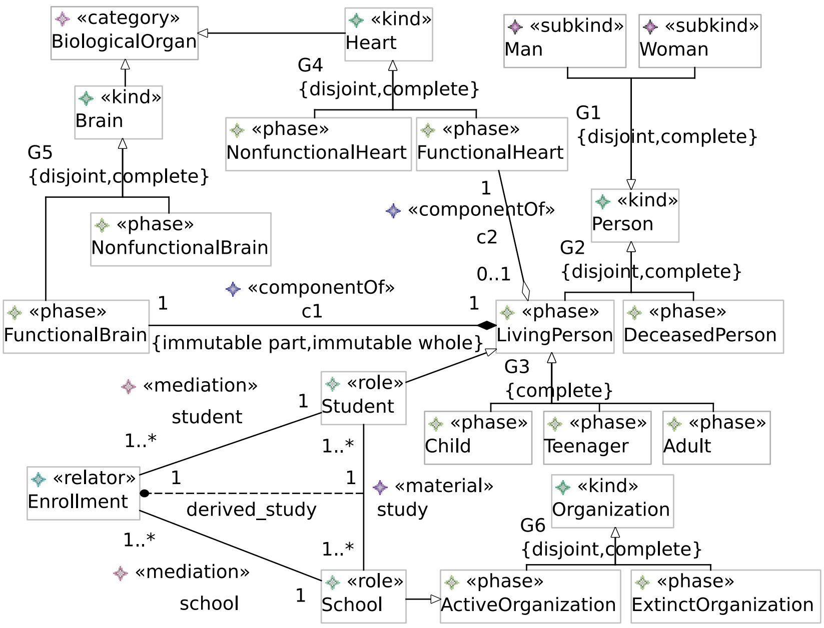


Figure 2: *Running example*

persons can play the role of students while enrolled to organizations, which in turn will play the role of a school. As we explain OntoUML concepts, we will further constrain this example in order to illustrate some ontological choices made. This choices are reflected in [Fig. 2] by names of stereotypes decorating classes and relationships, which represents the leaf concepts shown in [Fig. 1].

The UFO categorization starts with a general catch-all notion of Entity. Entity can be distinguished in Universal and Individual, where Individuals are entities that exist in reality possessing a unique identity, and Universals, conversely, are space-time independent pattern of features, which can be realized in a number of different Individuals. In our example, an individual person, such as *John*, would be an Individual, while the concept of Person would be an Universal. In Alloy, Individuals are represented as atoms and the instantiation of Universals by Individuals is represented as inclusion in a set that, in turn, represents the Universal's extension.

Universal can be distinguished in Monadic Universal and Relation (entities that glue together other entities). Within the category of Monadic Universal, in order to show the differences between Substance Universal and Relator Universal, we need to explicate what are Substances and Moments.

3.3.1 Substances and Moments

The distinction between Substances and Moments[4] is based on the formal notion of existential dependence, a modal notion that can be briefly defined as follows:

Definition 1 (existential dependence): Let the predicate ε denote existence[5]. We have that an Individual x is *existentially dependent* on another Individual y (symbolized as $ed(x, y)$) iff, as a matter of necessity, y must exist whenever x exists, or in other words, that in every world w in which x exists, then y must also exist in w. Since we assume that every Individual in the domains of quantification possibly exists and possibly do not exist (*i.e.*, there is no necessarily existing Individuals) [Guizzardi 2005], formally, we have that: $ed(x, y) \stackrel{\text{def}}{=} \Box(\varepsilon(x) \rightarrow \varepsilon(y))$. ■

Substances are existentially independent Individuals, *i.e.*, there is no Entity y *disjoint* from x that must exist whenever a Substance x exists. Let $\leq$ represent the (improper) *part-of* relation. This constraint can be formalized as follows: $disjoint(x, y) \stackrel{\text{def}}{=} \neg\exists z((z \leq x) \wedge (z \leq y))$ and $\forall x, y((\text{Substance}(x) \wedge \text{Substance}(y) \wedge disjoint(x, y)) \rightarrow (\neg ed(x, y) \wedge \neg ed(y, x)))$.

Examples of Substances include ordinary mesoscopic objects such as an individual person and an organization. Conversely, a Moment is an Individual that is existentially dependent on other Individuals. A moment can be existentially dependent on one single Individual (*e.g.*, a color) or on multiple Individuals

[4]The notion of moment comes originally from the writtings of E. Husserl to denote an existentially dependent entity (sometimes named Accident, Trope or Particularized Property). Thus, this notion as used here bears no relation to the common-sense use of the term as a temporal instant.

[5]Notice that in an actualist system, the existence operator ε can then be explicitly defined such that $\varepsilon(x) \stackrel{\text{def}}{=} \exists y(y = x)$.

(*e.g.*, an enrollment), in which case they are named Relational Moments or simply Relators. The particular sort of existential dependence Relation connecting a Relator to the Individuals on which it is dependent is the Formal Relation of *mediation* (m). This relation can be formally characterized as follows: (i) $\forall x, y(m(x, y) \rightarrow (\text{Relator}(x) \wedge \text{Substance}(y)))$, (ii) $\forall x, y(m(x, y) \rightarrow ed(x, y))$ and (iii) $\forall x(\text{Relator}(x) \rightarrow \exists y, z(\neg(y = z) \wedge m(x, y) \wedge m(x, z)))$.

So, a Substance Universal is a Universal whose instances are Substances (*e.g.*, the Universal Person), while a Relator Universal is a Universal whose instances are Individual Relational Moments (*e.g.*, the particular enrollment connecting *John* and a certain University).

3.3.2 Substance Universals

Substance Universals can be Sortal Universals or Mixin Universals. Sortal Universals are Universals that provide a principle of individuation and identity to its instances (*e.g.*, the Universal Brain), while Mixin Universals are abstractions of properties of instances of Sortal Universals (*e.g.*, BiologicalOrgan). We need to define some modal notions to be able to make further distinctions within these categories.

Definition 2 (Rigidity): A Universal U is rigid if for every instance x of U, x is necessarily (in the modal sense) an instance of U. In other words, if x instantiates U in a given world w, then x must instantiate U in every possible world w' in which x exists. This can be formally expressed by the following formula schema: $R(\text{U}) \stackrel{\text{def}}{=} \Box(\forall x(\text{U}(x) \rightarrow \Box(\varepsilon(x) \rightarrow \text{U}(x))))$. ■

Substance Universals that are rigid are named Kinds and subKinds. Due to the transitivity of instantiation over the subtyping relation, if x instantiates a subKind SK then x instantiates every Universal of which SK is a subtype of. At the root of this rigid specialization chain we have a Kind, *i.e.*, a Kind is the unique ultimate Rigid Sortal that the Individual x instantiates. For instance, in [Fig. 2], Person is a Kind that is partitioned in the subKinds Man and Woman.

In Alloy, we model Kinds and subKinds as *signatures*. A signature is a declaration of a set that can contain only atoms. Alloy allows the definition of subsignatures (subsets) by the keywords "`in`", which collapses the $\in$ and $\subseteq$ set-theoretic operators, and "`extends`", which is used to declare disjoint subsignatures of a signature ([Line 2] of [Listing 1]). The keyword "`abstract`" ([Line 1] of [Listing 1]) indicates that when an "`abstract`" signature "`S`" is extended by other subsignatures "`S1`", ..., "`Sn`", then all the atoms of "`S`" must be atoms of at least one of the "`S1`", ..., "`Sn`" signatures. Moreover, all top-level signatures (*i.e.*, signatures that are subsignatures of no signature) are pairwise disjoint.

By modeling Kinds as top-level signatures in Alloy ([Lines 1 and 3] of [Listing 1]), (i) the instances of these signatures are automatically pairwise disjoint, what is suitable because these instances are meant to be distinct objects carrying distinct identities; and (ii) an instance of a signature never ceases to instantiate this signature, what reifies the notion of rigidity [Definition 2].

Listing 1: *Alloy signatures*

```
1 abstract sig Person {}
2 sig Man, Woman extends Person {}
3 sig Heart, Brain, Organization {}
```

Returning to subKinds, a subKind must be a refinement of a Kind, and we represent it in Alloy by making subKinds as subsignatures, and using the Alloy keyword "in" followed by the signature representing its supertype. If there is a GeneralizationSet constraining some subKinds to be disjoint, we declare them with the keyword "extends" instead of "in" ([see Line 2] of [Listing 1]); and if there is a GeneralizationSet constraining them to be complete, we declare a *signature fact* within the signature of the supertype constraining the set of instances of the supertype to be equal to the union of the sets of its subtypes' instances. Facts are logical statements about signatures and relations that are always true for the whole specification. When created within signatures, facts are called signature facts and are implicitly universally quantified over all the atoms of the signature. Finally, if there is a GeneralizationSet constraining the subtypes to partition the supertype, then we can substitute the signature fact by the keyword "abstract" before the supertype signature (as shown in [Line 1] of [Listing 1]).

Besides rigidity, UFO defines the concept of anti-rigidity, which allows dynamic classification of Individuals. Object Migration has been an important issue in the literature of conceptual modeling at least since the late seventies [Bachman and Daya 1977] and its role in capturing subtle semantics aspects of software systems can be summarized by the following quote from [Papazoglou and Krämer 1997]: "To effectively model complex applications in which constantly changing situations can be represented, a systems must be able to support the evolution ... of individual objects. The strict uniformity of objects contained in a class is unreasonable ... An object that evolves by changing its type dynamically is able to represent changing situations as it can be an instance of different types from moment to moment.".

Definition 3 (Anti-rigidity): A Universal U is anti-rigid if for every instance x of U, x is possibly (in the modal sense) not an instance of U. In other words, if x instantiates U in a given world w, then there must be a possible world w' in which x exists and in which x does not instantiate U. Formally, we have that: $AR(\mathrm{U}) \stackrel{\text{def}}{=} \Box(\forall x(\mathrm{U}(x) \rightarrow \Diamond(\varepsilon(x) \wedge \neg \mathrm{U}(x))))$. ■

Within the category of anti-rigid Substance Universals, we have a further distinction between Phases and Roles. Both Phases and Roles are specializations of Sortal Universals. However, they are differentiated w.r.t. their specialization conditions. For the case of Phases, the specialization condition is always an intrinsic one. In our example, we could classify Persons regarding their age, creating phase partitions such as Child, Teenager and Adult. For Roles, in contrast, their specialization condition is a relational one: a student is a LivingPerson who is enrolled in (has a *study* relation to) a School. Formally speaking, this distinction is based on a meta-property named Relational Dependence:

Definition 4 (Relational Dependence): A type T is relationally dependent on another type P via relation R iff for every instance x of T there is an instance y of P such that x and y are related via R. In the following formula schema, we have that: $RD(\mathrm{T,P,R}) \overset{\text{def}}{=} \Box(\forall x(\mathrm{T}(x) \rightarrow \exists y(\mathrm{P}(y) \wedge \mathrm{R}(x,y))))$. ■

Finally, as discussed in [Guizzardi 2005], Phases (contrarily to Roles) are always defined in a partition set. For instance, in [Fig. 2], the universals Child, Teenager and Adult define a phase partition for the Kind Person. As a consequence, we have that in an each world w, every Person is either a Child, a Teenager or an Adult in w and never more than of the these. Additionally, if x is a Child (Teenager, Adult) in w, there is always a possible world w' in which x will not be a Child, in which case he will be either a Teenager or an Adult.

There is no built-in notion of state change in Alloy. In order to represent object dynamics, we must reify a notion of state change by means of a world structure that will be presented in detail in [Section 4]. Features that are time dependent, such as individual existence, dynamic classification and transitory relationships must be indexed by worlds in which they occur, *i.e.*, anti-rigid universal instantiation is dynamic, thus we represent it as a relationship between worlds and Individuals.

Therefore, in order to represent modality in Alloy, we create a signature named "World", shown in [Line 1] of [Listing 2]. As we are adopting an actualist domain of quantification, then for every world w there is a *relation* named " domain_of_quantification " ([see Line 2] of [Listing 2]) representing its domain of quantification ($\mathcal{D}(w)$), which contains some (w, ts) tuples in which ts is a top-level signature.

In Alloy, relations are sets of tuples, which may be of any finite arity, but containing only atoms. As shown in [Listing 2], they must be declared as fields within signatures. [Line 2] depicts a relation named " domain_of_quantification " between the signatures "World" and the union of the signatures "Person", "Heart ", "Brain", "Organization" and "Enrollment" (the signature "Enrollment" will be explained later). The keyword "+" is the set-theoretic union operator, which can also be used to form sets from scalars. Also, the keyword "some" is equivalent to the cardinality "1..*" and, in [Line 2] of [Listing 2], it is used to constrain the cardinalities on the extremity connected to the union of the signatures "Person", "Heart", "Brain", "Organization" and "Enrollment".

As Phases and Roles are anti-rigid, then its extensions may vary from world

Listing 2: *Alloy relations*

```
abstract sig World {
  domain_of_quantification: some (Person + Heart +
      Brain + Organization + Enrollment)}
```

to world. Therefore, these classes are modeled within the “World” signature as binary relations from worlds to Kinds or subKinds that are its supertypes and that are in the domain of quantification of that world. We model subtyping in two ways, regarding the nature of the supertype. If the superclass is a Rigid Sortal RS, then we use the Alloy Range Restriction operator (“:>”) to constrain the set of tuples of the relation representing the subtype to be a subset of the set of tuples of the relation representing the domain of quantification in which the second element is an instance of RS. In Alloy, the expression r :> s contains the tuples of r that end with an element in s. Examples can be seen in [Lines 3, 6, 7 and 8] of [Listing 3]. Otherwise, as Phases and Roles must be (directly or indirectly) subtypes of Kinds, then a Phase (or a Role) that is indirectly subtype of a Kind is transitively constrained by constraints created for its supertypes that are directly subtypes of Kinds. Therefore, we declare the range of the Alloy relation as the signature of the (non-rigid) supertype, as shown in [Lines 4, 5 and 9] of [Listing 3]. In these lines, the “disj” keyword states pairwise disjointness of relations, and the “set” keyword implies the inexistence of cardinality restrictions (“*”).

Listing 3: *Modeling Phases and Roles in Alloy (extends [Listing 2])*

```
1 abstract sig World {
2 ⋮
3   disj LivingPerson, DeceasedPerson: set Person:>
      domain_of_quantification,
4   disj Adult, Child, Teenager: set LivingPerson,
5   Student: set LivingPerson,
6   disj FunctionalHeart, NonfunctionalHeart: set Heart
      :>domain_of_quantification,
7   disj FunctionalBrain, NonfunctionalBrain: set Brain
      :>domain_of_quantification,
8   disj ActiveOrganization, ExtinctOrganization: set
      Organization:>domain_of_quantification,
9   School: set ActiveOrganization}
```

Furthermore, in order to model GeneralizationSets of Phases and Roles, if the subclasses are disjoint and are not part of another disjoint GeneralizationSet, we use the keyword “disj” before their declaration ([Lines 3, 4, 6, 7 and 8] of [Listing 3]), otherwise, for each disjoint GeneralizationSet we create “disj [...] ” facts containing the disjoint subtypes. Signature facts are created within a second pair of braces, as shown in [Line 1] of [Listing 4], just after the ellipsis between the first pair of braces. If the GeneralizationSets are complete and the superclass is not a Rigid Sortal (if it is declared as a relation), then we create a signature fact within the signature "World" stating that the set of tuples of the relation representing the superclass is equal to the union of the set of tuples of the relations representing the subclasses ([see Line 4] of [Listing 4]). If the subclasses are complete and the superclass is a Rigid Sortal, then we constrain the domain of quantification to only contain instances of the superclass that are also instances of at least one subclass ([Lines 2, 6, 8 and 10] of [Listing 4]).

Furthermore, Phases are always defined in a partition set $\langle P_1, \ldots, P_n \rangle$ constraining a Sortal Universal S [Guizzardi 2005, p. 103], and it is always possible (in the modal sense) for an instance x of S to become an instance of each P_i ($i \in 1, \ldots, n$) [Guizzardi 2005, p. 104]. Therefore, for any world w, if x is an instance of S in w, then x must be an instance of exactly one Phase of S in w and for each Phase P_i of S, there must exist a world in which x is an instance of P_i. In [Lines 3, 5, 7, 9 and 11] of [Listing 4] we show how we model the last constraint. In these lines, the Alloy keyword "@" is used to prevent a field name from being expanded. In Alloy, field names are automatically expanded when used within the signature in which they were specified. For example, just like a reference to a field of a receiver in an object-oriented program, *LivingPerson* now implicitly refers to *this.LivingPerson*, which is not a relation, but a set of atoms.

Observe that these two last constraints together imply anti-rigidity. Therefore, there is no need to model anti-rigidity constraints for Phases. However, we have to model anti-rigidity for Roles, but only for the ones that are not subtypes of another Roles or Phases. Because, from the [Definition 3], when anti-rigidity is guaranteed for a class, then it is automatically guaranteed for all its subtypes. In other words, we only have to model anti-rigidity for the top level Roles. As in our running example all the Roles are subtypes of Phases, then there is no need to explicitly model anti-rigidity for them. However, for the sake of completeness, we show how we would model anti-rigidity in the commented [Lines 12 and 13] of [Listing 4].

Listing 4: *Modeling Constraints of Phases and Roles in Alloy (extends [Listing 3])*

```
1 abstract sig World {...}{
2   Person:>domain_of_quantification = LivingPerson +
        DeceasedPerson
3   all x: Person | some w0,w1: World | (x in
        w0.@LivingPerson) and (x in w1.@DeceasedPerson)
4   LivingPerson = Adult + Child + Teenager
5   all x: LivingPerson | some w0,w1,w2: World | (x in
```

```
    w0.@Child) and (x in w1.@Teenager) and (x in
    w2.@Adult)
Heart:>domain_of_quantification = FunctionalHeart +
    NonfunctionalHeart
all x: Heart | some w0,w1: World | (x in
    w0.@FunctionalHeart) and (x in
    w1.@NonfunctionalHeart)
Brain:>domain_of_quantification = FunctionalBrain +
    NonfunctionalBrain
all x: Brain | some w0,w1: World | (x in
    w0.@FunctionalBrain) and (x in
    w1.@NonfunctionalBrain)
Organization:>domain_of_quantification =
    ActiveOrganization + ExtinctOrganization
all x: Organization | some w0,w1: World | (x in
    w0.@ActiveOrganization) and (x in
    w1.@ExtinctOrganization)}
--all x: Student | some w: World | (x in
  w.@domain_of_quantification) and (x not in
  w.@Student)
--all x: School | some w: World | (x in
  w.@domain_of_quantification) and (x not in
  w.@School)
```

Regarding the Mixin Universals [see Fig. 1], the rigid ones are named Categories. A Category classifies entities having distinct indentity criteria and sharing some essential characteristic. As shown in [Fig. 2], BiologicalOrgan is modeled as a Category. In order to model Categories in Alloy, we use Alloy *functions*, which are reusable Alloy expressions. For example, the Alloy code "`fun BiologicalOrgan (): (Heart + Brain) {Heart + Brain}`" shows a nullary function in which the expression "`Heart + Brain`" represents both the type of the result and the result itself.

As Categories are abstract, their instances are always instances of at least one of their subtypes, and as they are rigid, their instances never cease to be instances of them. Therefore, if a Category has subtypes, then it can be modeled as a set that is the union of the instances of its subtypes. So, we model a Category as a nullary function composed of a constant output that is the union of the signatures/functions of all of its subtypes. For example, if the Category C_1 is the supertype of $C_2, \ldots, C_n$, then the function "C1" will be the union of "C2", ..., "Cn", as shown in [Line 1] of [Listing 5]. If there is a GeneralizationSet stating that $C_2, \ldots, C_n$ are disjoint, then we create a new fact stating that "C2", ..., "Cn" are pairwise disjoint, as shown in [Line 2] of [Listing 5].

Listing 5: *Modeling Categories*

```
fun C1(): (C2 + ... + Cn) {C2 + ... + Cn}
fact disjoint_categories {disj [C2,...,Cn]}
```

3.3.3 Relator Universals and Relations

As one can observe in [Fig. 1], the *Relation* category in UFO is differentiated in *Formal Relation* and *Material Relation*. Formal Relations are Relations that hold between two or more entities directly, without any further intervening Individual. Material Relations, conversely, in order to hold between a number of Individuals, require that a particular Relator exists *mediating* them. For instance, we can say that a particular student x studies in a particular school y iff there is an Enrollment z that mediates x and y. This situation is illustrated in [Fig. 2]. In this case, we write that the relation *study* is derived from the existence of the Relator Universal Enrollment. This relation of Derivation between a Material Relation and a Relator Universal is represented in OntoUML by the symbol •– – – – – –, in which the black circle is connected to the Relator Universal. In general, a Relation R can be formally defined by the following schema:

Definition 5 (Formal and Material Relations): Let $\varphi(a_1, \ldots, a_n)$ denote a condition on the Individuals $a_1, \ldots, a_n$. A Relation R is defined for the Universals $\mathrm{U}_1, \ldots, \mathrm{U}_n$ iff $\forall a_1, \ldots, a_n(\mathrm{R}(a_1, \ldots, a_n) \leftrightarrow ((\bigwedge_{i \leq n} \mathrm{U}_i(a_i)) \wedge \varphi(a_1, \ldots, a_n)))$. A Relation is called material if there is a Relator Universal U_R such that the condition φ is obtained from U_R as follows: $\varphi(a_1, \ldots, a_n) \leftrightarrow \exists k(\mathrm{U}_R(k) \wedge \bigwedge_{i \leq n} m(k, a_i))$. Otherwise, if such a Relator Universal U_R does not exists, R is termed a Formal Relation. ■

We have then that an n-tuple $(a_1, \ldots, a_n)$ instantiates a Material Relation R iff there is one Relator r (instance of U_R) that mediates (and is existentially dependent on) every single a_i.

Just as Kinds, Relator Universals also provide a principle of identity for their instances, but this principle is dependent on the principles provided by the Universals that they mediate. However, OntoUML makes no distinction between the ultimate Relator Universals and the Relator Universals that are subtypes of the former, inheriting its unique principle of identity. Therefore, we will take top-level Relator Universals as ultimate and model them in Alloy as signatures ([Line 1] of [Listing 6]), as we did for Kinds, and the non-top-level ones will be modeled as subsignatures, just as subKinds. The Mediation relationships will be explained further on.

3.3.4 Part-Whole Relations

Parthood is a relation of significant importance in conceptual modeling, being present in practically all conceptual modeling languages (*e.g.*, OML, UML, EER). An important aspect to be addressed by any conceptual theory of parthood is to stipulate the different status that parts can have w.r.t. the whole they compose. As discussed by [Guizzardi 2007], many of the issues regarding this point cannot be clarified without considering *modality*. Here, we only mention three different types of part-whole relations that are distinguished based on the distinction between the notion of Specific Dependence and the one of Generic Dependence.

Definition 6 (specific dependence): For every Individuals x and y, and types T_1 and T_2: $SD(x, y, \mathrm{T}_1, \mathrm{T}_2) \stackrel{\text{def}}{=} \Diamond(\varepsilon(x) \wedge \mathrm{T}_1(x)) \wedge \Box((\varepsilon(x) \wedge \mathrm{T}_1(x)) \rightarrow (\varepsilon(y) \wedge \mathrm{T}_2(y)))$, *i.e.*, whenever x exists instantiating T_1, then y must exist instantiating T_2. ■

Definition 7 (generic dependence): For every Individual x and types T_1 and T_2: $GD(x, \mathrm{T}_1, \mathrm{T}_2) \stackrel{\text{def}}{=} \Diamond(\varepsilon(x) \wedge \mathrm{T}_1(x)) \wedge \Box((\varepsilon(x) \wedge \mathrm{T}_1(x)) \rightarrow \exists y\ \mathrm{T}_2(y))$, *i.e.*, whenever x exists instantiating T_1 it is necessary that an instance of T_2 exists. ■

A parthood Relation of Immutable Part is a Relation that implies specific dependence [Definition 6]. Contrariwise, a Mandatory Parthood relation is one that implies generic dependence [Definition 7]. We can elaborate on our running example to distinguish the parthood relationships between a person and his/her heart and brain. One possible conceptualization is to define the relationship between a LivingPerson and his/her Brain (when functioning) as immutable. This conveys that each person can only be living while having a functional brain and vice-versa. A case of generic dependence could be depicted as a generic parthood relationship between a person and a heart. These two types of Relations are exemplified in [Fig. 2] by the Relations LivingPerson-FunctionalBrain and LivingPerson-FunctionalHeart, respectively. More specifically, an Immutable Part relation between the Universals LivingPerson and FunctionalBrain implies that: for every x instance of LivingPerson there is a Individual y instance of FunctionalBrain such that in every world w in which x is an instance of LivingPerson, the part-whole Relation part_of(y,x) holds (so y must be an instance of FunctionalBrain in w). In other words, x cannot be a LivingPerson without having that specific FunctionalBrain as part. An equivalent argumentation holds for the Immutable Whole parthood Relation from FunctionalBrain to LivingPerson. The Mandatory Parthood between the Universals LivingPerson and FunctionalHeart instead implies that: for every x instance of LivingPerson and in every world w, there is an instance of FunctionalHeart in that world such that the part-whole Relation part_of(y,x) holds there. In other words, x cannot be an instance of LivingPerson without an instance of FunctionalHeart as part, this instance, however, can change from world to world. The three Relations of Immutable Part, Immutable Whole and Mandatory Part are defined in the sequel:

Definition 8 (Immutable Part): For every Individuals x and y, and types T_1 and T_2: $IP(x, y, \mathrm{T}_1, \mathrm{T}_2) \stackrel{\text{def}}{=} \Diamond(\varepsilon(y) \wedge \mathrm{T}_2(y)) \wedge \Box((\varepsilon(y) \wedge \mathrm{T}_2(y)) \rightarrow (\varepsilon(x) \wedge \mathrm{T}_1(x) \wedge$

$(x \leq y)))$, *i.e.*, it is necessary that whenever y exists instantiating T_2, then x must exist as a part of y and instantiating T_1. ■

Definition 9 (Immutable Whole): For every Individuals x and y, and types T_1 and T_2: $IW(x, y, \mathrm{T}_1, \mathrm{T}_2) \overset{\text{def}}{=} \Diamond(\varepsilon(x) \wedge \mathrm{T}_2(x)) \wedge \Box((\varepsilon(x) \wedge \mathrm{T}_2(x)) \rightarrow (\varepsilon(y) \wedge \mathrm{T}_1(y) \wedge (x \leq y)))$, *i.e.*, it is necessary that whenever x exists instantiating T_2, then y must exist as a whole for x and instantiating T_1. ■

Definition 10 (Mandatory Part): For every Individual x and types T_1 and T_2: $MP(\mathrm{T}_1, \mathrm{T}_2, x) \overset{\text{def}}{=} \Diamond(\varepsilon(x) \wedge \mathrm{T}_1(x)) \wedge \Box((\varepsilon(x) \wedge \mathrm{T}_1(x)) \rightarrow \exists y(\mathrm{T}_2(y) \wedge (y \leq x)))$, *i.e.*, it is necessary that whenever x exists instantiating T_1, then there must exist an Individual as a part of x and instantiating T_2. ■

In Alloy, we model OntoUML relationships differently regarding their modal implications. If an OntoUML relationship imply existential dependence, *e.g.*, a Mediation relationship, then it is modeled as an Alloy relation within the dependent signature ([see Lines 2 and 3] of [Listing 6]). Additionally, we have to create a signature fact within "World" to guarantee that if an atom a is existentially dependent on an atom b, then for each world w, if $a \in \mathcal{D}(w)$, then $b \in \mathcal{D}(w)$ ([Line 7] of [Listing 7]).

Listing 6: *Modeling Relators and existential dependence relationships*

```
1 sig Enrollment{
2   student: one Person,
3   school: one Organization,
4   derived_study: student one -> one school}
```

Otherwise, OntoUML relationships are modeled as ternary relations within the "World" signature, *e.g.*, [see Lines 3, 4 and 5] of [Listing 7]. Additionally, an OntoUML relationship R that imply specific dependence from an anti-rigid Sortal Universal ARS to another classifier C, *i.e.*, if it is a Meronymic relationship R in which the meta-attributes isImmutablePart or isImmutableWhole are true, then we have to pose an additional constraint to guarantee that if an instance $rs1 \in \mathcal{D}(w)$ is such that $rs1$::ARS[6] in w and $rs1$ is related to an instance $rs2 \in \mathcal{D}(w)$ (such that $rs2$::C in w) by an instance r of R in w, then in whatever world w' in which $rs1 \in \mathcal{D}(w')$ and $rs1$::ARS, then $rs2 \in \mathcal{D}(w')$ and $rs2$::C and $rs1$ must be related to $rs2$ by an instance r' of R in w' ([see Lines 8 and 9] of [Listing 7]).

Subtyping and GeneralizationSets between OntoUML relationships are modeled similarly as for anti-rigid sortals. Regarding cardinalities, the Alloy language has some keywords for the most usual cardinalities, like "set", "lone", "one" and "some" meaning "0..*", "0..1", "1..1" and "1..*", respectively. In order to model single-tuple cardinalities of relationships, we can use these keywords on the declaration of a relation, as shown in [Lines 3, 4 and 5] of [Listing 7]. In the case the

[6] The symbol a::A means that a is an instance of A.

relationship is declared within a signature that is its first domain (like the ones in [Lines 2 and 3] of [Listing 6]), then we may have to include a signature fact in order to constrain the cardinality of its first extremity, as show in [Lines 10 and 11] of [Listing 7]. In order to model general “m..n” cardinalities, we can use the dot operator “.” to navigate to a relation’s extremity and the cardinality operator “#” to constrain the size of the set of elements in that extremity by writing inequalities. For example, if an OntoUML Relation R between A and B has an “m..n” cardinality constraint on the extremity B, we would write a signature fact like “`all a:A:>domain_of_quantification | (#(a.R)>= m)and (#(a.R)<= n)`”.

In order to model an n-ary Material Association M with tuples in $C_1 \times \ldots \times C_n$, we model the Derivation and the Mediations within the signature of the respective Relator R ([Lines 2, 3 and 4] in [Listing 6]), and model M within the signature “World” ([Line 3] in [Listing 7]). As Alloy is a first order language, then it does not allow the creation of tuples containing tuples. Therefore the Derivation relation will not be modeled as a relation between instances of R and M. Instead, we model it as an $(n+1)$-ary Relation with tuples in $R \times C_1 \times \ldots \times C_n$. We also constrain M in a way that for any world w, M is a subset of the set of Derivations of all the instances of R that are in $\mathcal{D}(w)$ ([Line 12] of [Listing 7]).

Listing 7: *Modeling relationships that do not imply existential dependence (extends [Listing 4])*

```
1  abstract sig World {
2    ⋮
3    study: set Student some -> some School,
4    c1: set FunctionalBrain one -> one LivingPerson,
5    c2: set FunctionalHeart one -> lone LivingPerson}{
6    ⋮
7    all x: Enrollment:>domain_of_quantification |
         x.school in School and x.student in Student
8    all x: Person, w0, w1: (@c1.x).Brain | (w0.@c1).x =
         (w1.@c1).x -- immutablePart.
9    all x: Brain, w0, w1: (@c1.Person).x | x.(w0.@c1) =
         x.(w1.@c1) -- immutableWhole.
10   all x: Student | some (student.x):>
         domain_of_quantification
11   all x: School | some (school.x):>
         domain_of_quantification
12   study = (Enrollment:>domain_of_quantification).
         derived_study}
```

Finally, whenever an abstract classifier has non-abstract subtypes, we will model it as if its subtypes form a complete GeneralizationSet. Otherwise, there is no need to model it in Alloy, as it will not be able to have atoms.

4 Instantiation for Validation

In this section, we will use the Alloy Analyzer to generate an instance for the Alloy specification of the running example, which was being defined along the previous sections. From now on, we will refer to the instances of the Alloy specification of the running example as instances of the running example.

Having defined UFO's axioms in Alloy, we have captured a notion of modality explicitly in the generated Alloy specification. This means that this specification reifies the notion of an actualist world structure. This is necessary to specify UFO's modal axioms, given no notion of modality is built-in in Alloy. While UFO's formalization is based on a totally accessible world structure, we believe that the inspection of instances in which worlds can the interpreted temporally is more suitable for validation purposes. Thus, we constrain the world structures so that the Alloy Analyzer will only produce instances following this temporal perspective.

In our ordinary language, we are able to talk about the current moment, the past, the possible future, and the facts that could have happened, but accidentally did not (*i.e.*, the counterfactuals). Therefore, we want our worlds to be interpreted as past worlds, future ones, counterfactual ones or the current one. So, we will represent the actualist world structure for QS5 in Alloy and validate UFO's axioms in their QS5 form, but we will also categorize the worlds into those four disjoint categories ordering them by a partial order relation of immediate succession. By constraining the world structures in this way, we will avoid generating instances that cannot present this temporal interpretation. [Listing 8] shows the modeling of this world structure in Alloy. We model the different types of worlds and their respective constraints regarding what types and quantities of worlds can be accessed by their immediate succession relations (named "next" in Listing 8). Also, we impose that (i) there cannot be temporal cycles ([Line 4] of [Listing 8]); (ii) a world can be the immediate next moment of at most one world ([Line 5] of [Listing 8]); and (iii) every world, except the current one, must reach the current world ([Lines 10, 13 and 16] of [Listing 8]). The Alloy keyword "^" represents transitive closure.

Listing 8: *World structure in Alloy*

```
1 module world_structure[World]
2 some abstract sig TemporalWorld extends World{
3   next: set TemporalWorld}{
4   this not in this.^(@next)
5   lone ((@next).this)}
```

```
one sig CurrentWorld extends TemporalWorld {} {
  next in FutureWorld}
sig PastWorld extends TemporalWorld {} {
  next in (PastWorld + CounterfactualWorld +
      CurrentWorld)
  CurrentWorld in this.^@next}
sig FutureWorld extends TemporalWorld {} {
  next in FutureWorld
  this in CurrentWorld.^@next}
sig CounterfactualWorld extends TemporalWorld {} {
  next in CounterfactualWorld
  this in PastWorld.^@next}
```

Since the most fundamental criteria for individuation are spatiotemporal and constrain instances of Sortal Universals to move on spatiotemporally continuous paths [Xu and Carey 1996], we further constrain the generation of instances to produce only examples in which atoms have continuous existence. This is specified in Alloy as "fact continuous_existence {all w: World, x: (next.w). domain_of_quantification | (x not in w.domain_of_quantification) => (x not in ((w.^next). domain_of_quantification))}".

Moreover, as the accessibility relation is total in QS5, then the modal operators of possibility ($\Diamond$) and necessity ($\Box$) will take worlds in the set of all worlds ($\mathcal{W}$). Thus, in order to reduce the the computational complexity of analyzing the Alloy specification, we will modify the definitions of these modal operators to use $\mathcal{W}$ instead of the accessibility relation. For the same reasons, we will constrain every atom to be in the domain of quantification of some world, otherwise, Alloy Analyzer could generate atoms that would not be shown. This fact can be written as " fact UFO_fact { all x: (Person + Heart + Brain + Organization + Enrollment)| some w:World | x in w.domain_of_quantification }".

[Fig. 3] depicts an instance of the running example that is automatically generated by the Alloy Analyzer. Although valid from a logic point of view, these presentations are not suitable to be inspected and reasoned upon by the human modeler. Fortunately, the Alloy Analyzer allows the creation of visualization themes [Rayside et al. 2007]. Here, we take advantage of this feature by providing two visualization themes, one for visualizing the temporal ordering of worlds (*e.g.*, [Fig. 4a]) and the other to visualize the atoms by showing only the ones that are in the domain of quantification of a selected world (*e.g.*, [Figs. 4b and 4c]).

As one can see, despite being a valid instance, the instance shown in [Fig. 4] is of little interest, as it only shows an active organization becoming an extinct organization in the current world. Instead of visualizing every instance generated by the tool, in order to find representative ones for validation purposes, we can

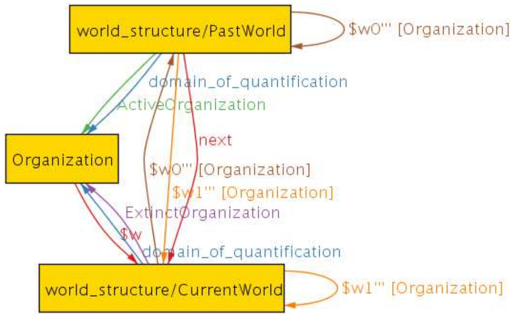


Figure 3: *An instance*

(a) *The temporal ordering of worlds*

(b) *Instance at the past moment*

(c) *Instance in the current moment*

Figure 4: *Atoms projected by worlds*

further qualify the type of instances we want the Alloy Analyzer to generate. For example, we will impose the generation of an instance having a person in at least one world, two disjoint hearts, at least one world in which there is a school, one of each type of worlds, and at maximum four atoms for each top-level signature. This constraint is shown in [Listing 9] and the generated instance is shown in [Figs. 5 and 6].

Listing 9: *Constraining the generation of instances*

```
run {(#Person = 1) and (#Heart = 2) and (#School >= 1)
     and (#CounterfactualWorld = 1) and (#PastWorld =
    1) and (#FutureWorld = 1)} for 4
```

As one can see, in the past world, the woman is deceased, but in the current world she is an adult, in a counterfactual one she could be a teenager and in a future one she may even become a child! As OntoUML does not contemplate the explicit modeling of the temporal ordering of Phases, we will introduce this ordering directly in the Alloy specification, as show in the [Listing 10].

Listing 10: *Modeling the ordering of the Phases*

```
fact an_ordering_for_the_phases{
```

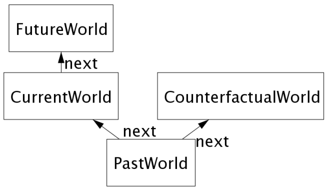
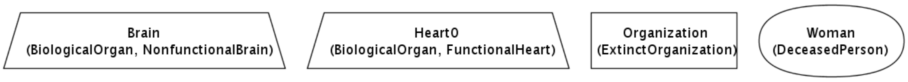


Figure 5: *The temporal ordering of worlds*

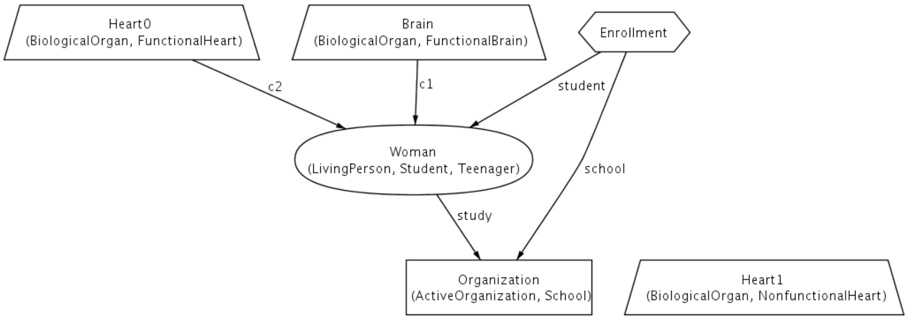
```
all x:Person, w0: World, w1:w0.next | (x in
    w0.DeceasedPerson) => (x not in w1.LivingPerson)
all x:Person, w0: World, w1:w0.next | ((x in
    w0.Child) => (x not in w1.Adult)) and ((x in
    w0.Teenager) => (x not in w1.Child)) and ((x in
    w0.Adult) => ((x not in w1.Child) and (x not in
    w1.Teenager)))
all x:Heart, w0: World, w1:w0.next | (x in
    w0.NonfunctionalHeart) => (x not in
    w1.FunctionalHeart)
all x:Brain, w0: World, w1:w0.next | (x in
    w0.NonfunctionalBrain) => (x not in
    w1.FunctionalBrain)
all x:Organization, w0: World, w1:w0.next | (x in
    w0.ExtinctOrganization) => (x not in
    w1.ActiveOrganization)}
```

Now, the generated atoms will instantiate the Phases in the correct ordering. An example is pictured in [Fig. 7], which has the same temporal ordering of worlds that is shown in [Fig. 5]. This instance exemplifies some important constraints like the rigidity [Definition 2] of the Kinds and Categories, *e.g.*, the woman never ceases to be an instance of the Kind Person while she is in a domain of quantification of a world); the anti-rigidity [Definition 3] of the Phases, *e.g.*, regarding the Phases Child, Teenager and Adult, for every world w in which the woman is in one of these Phases, there is a world w' in which she is not in that Phase; the anti-rigidity of the Roles, *e.g.*, the Role Student, in which for every world w in which the woman play this Role, there is a world w' in which she does not play this Role; the relational dependence [Definition 4] of the Roles is illustrated, *e.g.*, for the Role Student, in which the woman can only play this Role while related to an instance of School by an instance of Enrollment.

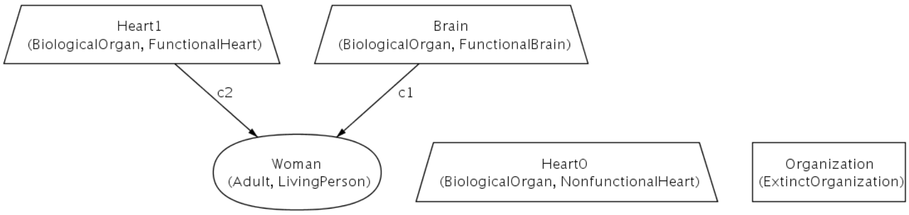
Also, this instance illustrates the immutability of both part and whole [Definitions 8 and 9] regarding the woman and her brain (she never changes her brain while alive and the brain never changes its whole when functional). One



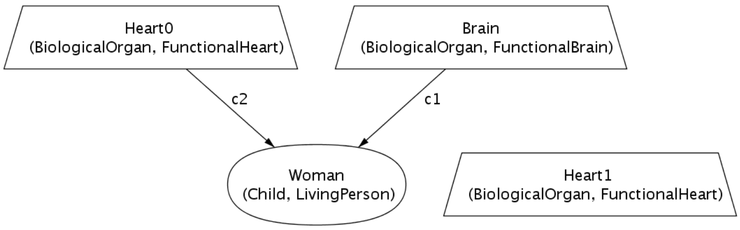
(a) *Instance at the past moment*



(b) *Instance in the counterfactual moment*



(c) *Instance in the current moment*

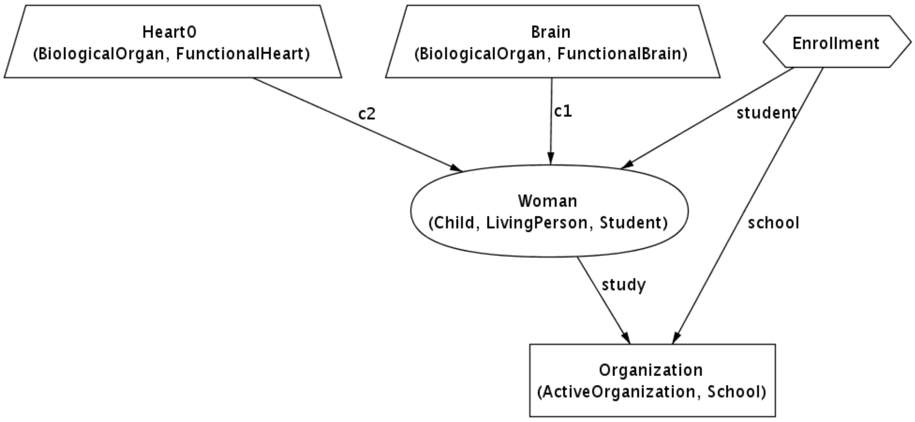


(d) *Instance in a future moment*

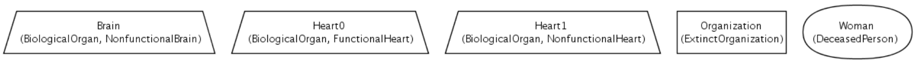
Figure 6: *Atoms projected by worlds*

can notice that in a future world, the woman will change her heart (maybe she will undergo a heart transplant), while her old heart will became nonfunctional. This behaviour is acceptable, as she is generically dependent on the Kind Heart.

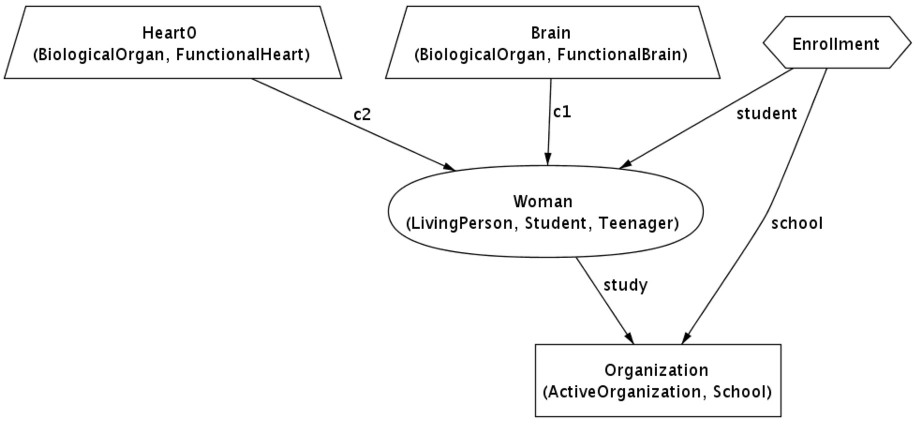
Finally, this instance also satisfies the constraint that imposes that if an atom instantiates a Phase in some world, then it must possibly instantiate every Phase in that Phase partition.



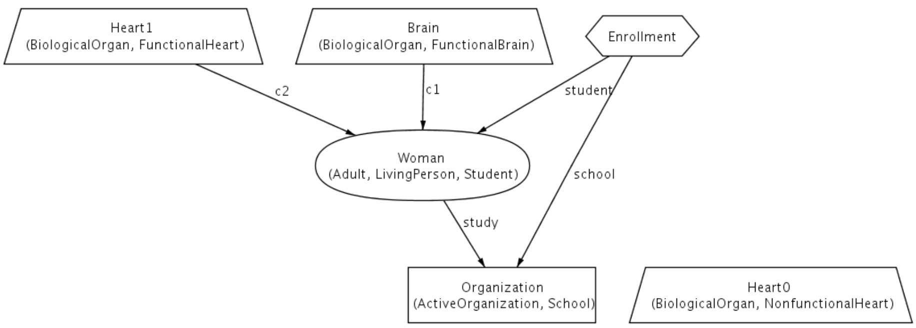
(a) *Instance at the past moment*



(b) *Instance in the counterfactual moment*



(c) *Instance in the current moment*



(d) *Instance in a future moment*

Figure 7: *Instance with ordered Phases*

5 Related Work

In [Braga et al. 2009, Braga et al. 2010], we have introduced an alternative approach to the one presented here, which considers a linear time structure instead of a branching time one. While the codification differences may be subtle, there are many consequences for instance visualization and conceptual model validation efforts that spawn from this variation in the time structure.

By using a branching time world structure, we are able to enforce and visualize key constraints of UFO categories that could not be considered in instances of a linear time world structure, like the anti-rigidity [Definition 3] and the (modal) possibility of an instance x of a sortal universal S to become an instance of each Phase that is a subtype of S [Guizzardi 2005, p. 104, formula 11] [see Page 13].

More specifically, the branching time approach allows the visualization of multiple possible realities at once, while the linear time one, in contrast, follows the typical approach of behavioral model validation and allows the visualization of sequences of states that reflect (part of) a particular possible execution of the model, a single possible sequence of events. Thus, in the linear time approach, we do not aim at showing an instance that "exercises modality" completely. Any instance of the visualization in this branching time approach will lead to examples of dynamic classification making anti-rigidity and phase changes explicit. In contrast, in the linear time approach, these examples can only appear by chance as opposed to begin systematically generated. The approach presented in this article allows the modeler to have visual information on how possibilities may unfold in time branches.

Consider the example in [Fig. 2]. In the linear time approach, the Alloy Analyzer may generate (by chance) an example of child mortality in which a person does not reach adulthood (technically, he/she never instantiates the "Adult" phase). In the branching time approach, the Alloy Analyzer would always reveal an alternative sequence of worlds in which the person reaches adulthood. This is important as it makes explicit the modal semantics of the OntoUML model, which may be hidden (by chance) in the linear time approach.

Moreover, while the linear time approach might be suitable for a mapping to particular codifications that follow a linear time structure (*e.g.*, temporal description logics such as DLR_{us} [Artale et al. 2008], or a reduction of the property change semantics to traditional OCL constraints over creation - change - destruction operators), the branching time one is fundamental for conceptual model validation, presenting the user with a fuller view of the model's underlying semantics. We emphasize that due to the discussed implications, changing from a linear time to a branching time world structure is not merely a minor codification issue, it affects the way humans perceive the simulation and how one may reason with the implications of the use of different OntoUML constructs.

Furthermore, these approaches have different complexity analysis. The branching time approach requires us to instantiate one atom for every world, whether factual (*viz.*, the pasts worlds and the current one) or not factual (*viz.*, the counterfactuals and futures worlds), and possibly extra atoms for every individual that only appear in some of the branches. By ignoring the branches and focusing on a single sequence of events, the linear time approach requires a smaller scope than the branching time one. Therefore, the linear time approach can save computational effort and possibly reach further into the state space, populating the examples with more individuals or simply improving the performance of analysis. Naturally, this means that, in the linear time approach, each execution of the Alloy Analyzer corresponds to different possible history lines, each of which corresponds to a "path" through the branching time structure. As a consequence, each "path" is considered in isolation and, therefore, no intersections between different history lines ("paths") are revealed. Finally, as the different history lines are considered in isolation, the linear time approach does not categorize history lines into factual or not factual.

Regarding different authors, the work that is most related to ours is the transformation shown in [Evermann 2009] of an ontologically well-founded language, proposed in [Evermann 2003], to OWL. This language is a UML profile that is based on the BWW ontology. However, to the best of our knowledge, despite allowing the verification of some formal aspects (possibly only the non-modal ones), this transformation proposes no support to validation in the sense we are dealing here.

Regarding languages that are not ontologically well-founded, several approaches in literature aim at assessing whether conceptual models comply with their intended conceptualizations. Although many approaches (*e.g.*, [Beato et al. 2004, Schinz et al. 2004]) focus on analysis of behavioural UML models, we are primarily concerned with structural models and thus refrain from further analysis of behavioural-focused work.

The USE (UML Specification Environment) tool proposed in [Gogolla et al. 2007] is able to indicate whether instances of a UML class diagram respect constraints specified in the model through OCL. Differently from our approach, which is based on the automatic creation of example world structures, in USE the modeler must specify sequences of snapshots in order to gain confidence on the quality of the model (either through the user interface or by specifying sequences of snapshots in a tool-specific language called ASSL, A Snapshot Sequence Language). Since no modal meta-property of classifiers is present in UML, this tool does not address modal aspects and validates constraints considering only a sole snapshot.

Finally, the approaches of [Massoni et al. 2004] and [Anastasakis et al. 2007][7] are similar to ours in that they translate UML class diagrams to Alloy. However,

[7] The UML2Alloy tool is available at `http://www.cs.bham.ac.uk/~bxb/UML2Alloy`.

both of them translate all classes into Alloy signatures, which suggests that no dynamic classification is possible in these approaches. Similarly to our approach, [Anastasakis et al. 2007] implements a model transformation using model-driven techniques to automatically generate Alloy specifications, while [Massoni et al. 2004] relies on manual translation to Alloy. Similar to USE, [Massoni et al. 2004] focuses on analysis and constraint validation on single snapshots. [Anastasakis et al. 2007] introduces a notion of state transition but still does not address the modal aspects of classes since these are not part of UML.

6 Final Considerations

A mature approach to conceptual modeling requires modelers to gain confidence on the quality of the models they produce, assessing whether these models express as accurately as possible an intended conceptualization. This paper contributes to that goal, by providing tools to support the modeler in the validation of a conceptual model in OntoUML.

Following a model-driven approach, we have defined and automated a transformation of OntoUML models into Alloy specifications. The generated Alloy specifications are fed into the Alloy Analyzer to create temporal world structures that show the possible dynamics of object creation, classification, association and destruction as implied by the model. The snapshots in this world structure confront a modeler with states-of-affairs that are deemed admissible by the model. This enables modelers to detect unintended states-of-affairs and take the proper measures to rectify the model. We believe that the example world structures support a modeler in the validation process, especially since it reveals how state-of-affairs evolve in time and how they may eventually evolve (revealing alternative scenarios implied by the model).

If the Alloy Analyzer fails to find an example world structure, this may indicate unsatisfiability, although no guarantee of unsatisfiability is given. This is a consequence of Alloy's choices to cope with tractability. For instance, Alloy searches for example structures within a restricted context, *i.e.*, a given finite maximum number of elements.

As future work, we aim at performing an empirical study about the effectiveness of this approach of validation based on simulations.

Moreover, we intend to incorporate support for domain constraints in our approach, *e.g.*, including OCL constraints in an OntoUML model. This will require transforming these constraints into Alloy in order to guarantee that the constraints are satisfied in all instances generated by the Alloy Analyzer.

Further, we intend to work on methodological support for the validation process, proposing guidelines for modelers to select relevant world structures. We will aim for an interactive approach in which a modeler can select which of the alternative scenarios to consider. We believe that this may help pruning the branches in the world structure keeping the size of this structure manageable.

Ideally, by exploring visualization techniques, we could use the instances generated by the Alloy Analyzer as example scenarios to be exposed to the stakeholders of the conceptual model (such as domain experts) in order to validate whether their conceptualization has been captured accurately by the modeler.

Acknowledgements

This research is funded by the Brazilian Research Funding Agencies FAPES (Process Number 45444080/09) and CNPq (Process Number 309382/2008-4, 309059/2008-9 and 481906/2009-6).

References

[Anastasakis et al. 2007] Anastasakis, K., Bordbar, B., Georg, G., Ray, I.: "UML2Alloy: A Challenging Model Transformation"; Proc. MoDELS 2007, Springer, Nashville (2007), 436-450, `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.65.9914&rep=rep1&type=pdf`.

[Artale et al. 2008] Artale, A., Guarino, N., Keet, C. M.: "Formalising Temporal Constraints on Part-Whole Relations"; Proc. KR 2008, AAAI Press, Sydney (2008), 673-683, `http://www.aaai.org/Papers/KR/2008/KR08-066.pdf`.

[Bachman and Daya 1977] Bachman, C. W., Daya, M.: "The Role Concept in Data Models"; Proc. VLDB '1977, VLDB Endowment, Tokyo (1977), 464-476.

[Beato et al. 2004] Beato, M. E., Barrio-Solórzano, M., Cuesta, C. E.: "UML Automatic Verification Tool (TABU)"; Proc. SAVCBS'04 at ACM SIGSOFT 2004/FSE-12, California (2004), 106-109, `http://www.eecs.ucf.edu/~leavens/SAVCBS/2004/savcbs04.pdf`.

[Benevides and Guizzardi 2009] Benevides, A. B., Guizzardi, G.: "A Model-based Tool for Conceptual Modeling and Domain Ontology Engineering in OntoUML"; Proc. ICEIS 2009, Springer, Milan (2009), 528-538, `http://www.inf.ufes.br/~gguizzardi/ICEIS_2009.pdf`.

[Benevides et al. 2009] Benevides, A. B., Guizzardi, G., Braga, B. F. B., Almeida, J. P. A.: "Assessing Modal Aspects of OntoUML Conceptual Models in Alloy"; Proc. ETheCoM 2009 at ER 2009, Springer, Gramado (2009), 55-64, `http://www.inf.ufes.br/~gguizzardi/benevides-et-al-2009.pdf`.

[Braga et al. 2009] Braga, B. F. B., Almeida, J. P. A., Guizzardi, G., Benevides, A. B.: "Transforming OntoUML into Alloy: Towards Conceptual Model Validation Using a Lightweight Formal Method"; Proc. UML&FM 2009 at ICFEM 2009, Rio de Janeiro (2009).

[Braga et al. 2010] Braga, B. F. B., Almeida, J. P. A., Guizzardi, G., Benevides, A. B.: "Transforming OntoUML into Alloy: Towards Conceptual Model Validation Using a Lightweight Formal Method"; ISSE (Innovations in Systems and Software Engineering), Springer-Verlag, London, 6, 1 (2010), 55-63, `http://www.springerlink.com/content/m1715n1220717158/fulltext.pdf`.

[Degen et al. 2001] Degen, W., Heller, B., Herre, H., Smith, B.: "GOL: Toward an Axiomatized Upper-Level Ontology"; Proc. FOIS 2001, ACM, Ogunquit, Maine (2001), 34-46, `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.6.8498&rep=rep1&type=pdf`.

[Evermann 2003] Evermann, J.: "Using Design Languages for Conceptual Modelling: The UML Case"; Faculty of Commerce and Business Administration, Management Information Systems, The University of British Columbia, Vancouver (2003), http://homepages.mcs.vuw.ac.nz/~jevermann/EvermannThesis03.pdf.

[Evermann 2009] Evermann, J.: "A UML and OWL Description of Bunge's Upper-Level Ontology Model"; SoSyM (Software and System Modeling), Springer-Verlag, 8, 2 (2009), 235-249.

[Fitting and Mendelsohn 1999] Fitting, M., Mendelsohn, R. L.: "First-Order Modal Logic"; Kluwer Academic Publishers, Norwell (1999).

[Gogolla et al. 2007] Gogolla, M., Büttner, F., Richters, M.: "USE: A UML-based Specification Environment for Validating UML and OCL"; Science of Computer Programming Special Issue on Experimental Software and Toolkits, 69, 1-3 (2007), 27-34.

[Gonçalves et al. 2007] Gonçalves, B., Guizzardi, G., Pereira.Filho, J. G.: "An Electrocardiogram (ECG) Domain Ontology"; Proc. WOMSDE 2007 at SBBD/SBES 2007, João Pessoa (2007), 68-81, http://www.lbd.dcc.ufmg.br:8080/colecoes/womsde/2007/006.pdf.

[Guarino and Guizzardi 2006] Guarino, N., Guizzardi, G.: "In the Defense of Ontological Foundations for Conceptual Modeling"; SJIS (Scandinavian Journal of Information Systems), 18, 1 (2006), 115-126, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.6281&rep=rep1&type=pdf.

[Guizzardi 2005] Guizzardi, G.: "Ontological Foundations for Structural Conceptual Models"; Centre for Telematics and Information Technology, University of Twente, Enschede (2005), http://doc.utwente.nl/50826.

[Guizzardi 2006] Guizzardi, G.: "On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models"; Proc. DB&IS'2006, IOS Press, Amsterdam (2006), 18-39.

[Guizzardi 2007] Guizzardi, G.: "Modal Aspects of Object Types and Part-Whole Relations and the de re/de dicto Distinction"; Proc. CAiSE'07, Springer-Verlag, Trondheim (2007), 5-20, http://www.loa-cnr.it/Guizzardi/CAISE-CR-2007.pdf.

[Guizzardi et al. 2004] Guizzardi, G., Wagner, G., Guarino, N., van Sinderen, M.: "An Ontologically Well-founded Profile for UML Conceptual Models"; Proc. CAiSE'04, Springer-Verlag, Latvia (2004), 112-126.

[Jackson 2002] Jackson, D.: "Alloy: a Lightweight Object Modelling Notation"; TOSEM (Transactions on Software Engineering and Methodology), ACM, New York, 11, 2 (2002), 256-290.

[Jackson 2006] Jackson, D.: "Software Abstractions : Logic, Language, and Analysis"; MIT Press, Cambridge, Massachusetts (2006).

[Massoni et al. 2004] Massoni, T., Gheyi, R., Borba, P.: "A UML Class Diagram Analyzer"; Proc. CSDUML04 at «UML» Conference 2004, Springer-Verlag, Lisbon (2004), 143-153.

[Mylopoulos 1992] Mylopoulos, J.: "Conceptual Modeling, Databases, and CASE: An Integrated View of Information Systems Development", chapter "Conceptual Modeling and Telos"; John Wiley & Sons, Chichester (1992), 49-68.

[Oliveira et al. 2007] Oliveira, F., Antunes, J., Guizzardi, R. S. S.: "Towards a Collaboration Ontology"; Proc. WOMSDE 2007 at SBBD/SBES 2007, João Pessoa (2007), 97-108, http://www.lbd.dcc.ufmg.br:8080/colecoes/womsde/2007/008.pdf.

[Papazoglou and Krämer 1997] Papazoglou, M. P., Krämer, B. J.: "A Database Model for Object Dynamics"; The VLDB Journal (The International Journal on Very Large Data Bases), Springer-Verlag, Secaucus, 6, 2 (1997), 73-96, http://ideas.repec.org/p/ner/tilbur/urnnbnnlui12-86860.html.

[Rayside et al. 2007] Rayside, D., Chang, F., Dennis, G., Seater, R., Jackson, D.: "Automatic Visualization of Relational Logic Models"; Proc. LED 2007, ECEASST, (2007), http://journal.ub.tu-berlin.de/index.php/eceasst/article/

view/94/89.

[Schinz et al. 2004] Schinz, I., Toben, T., Mrugalla, C., Westphal, B.: "The Rhapsody UML Verification Environment". Proc. SEFM'04, IEEE Computer Society, Beijing (2004), 174-183, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.71&rep=rep1&type=pdf.

[Xu and Carey 1996] Xu, F., Carey, S.: "Infants' Metaphysics: The Case of Numerical Identity"; Cognitive Psychology, 30 , 5 (1996), 111-153, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.1268&rep=rep1&type=pdf.