A Comprehensive Formal Theory for Multi-Level Conceptual Modeling

João Paulo A. Almeida[1], Claudenir M. Fonseca[1], Victorio A. Carvalho[2]

[1] Ontology & Conceptual Modeling Research Group (NEMO),
Federal University of Espírito Santo (UFES), Vitória, ES, Brazil
[2] Research Group in Applied Informatics, Informatics Department,
Federal Institute of Espírito Santo (IFES), Colatina, ES, Brazil
jpalmeida@ieee.org; claudenirmf@gmail.com; victorio@ifes.edu.br;

Abstract. Multi-level modeling extends the conventional two-level classification scheme to deal with subject domains in which classes are also considered instances of other classes. In the past, we have explored theoretical foundations for multi-level conceptual modeling and proposed an axiomatic theory for multi-level modeling dubbed MLT. MLT provides concepts for multi-level modeling along with a number of rules to guide the construction of sound multi-level conceptual models. Despite the benefits of MLT, it is still unable to deal with a number of general notions underlying conceptual models (including the notions used in its own definition). In this paper, we present an extension of MLT to deal with these limitations. The resulting theory (called MLT*) is novel in that it combines a strictly stratified theory of levels with the flexibility required to model abstract notions that defy stratification into levels such as a universal "Type" or, even more abstract notions such as "Entity" and "Thing".

Keywords: Conceptual Modeling, Multi-level Modeling, Metamodeling.

1 Introduction

The vast majority of conceptual modeling techniques are based on notions such as "class" and "type", capturing what subject matter experts refer to as "kinds", "categories" and "sorts" in their accounts of a subject domain. In several subject domains, the categorization scheme itself is part of the subject matter, and thus experts make use of categories of categories in their accounts. For instance, considering the software development domain [15], project managers often need to plan according to the *types of tasks* to be executed during the software development project (e.g. "requirements specification", "coding"). They may also need to classify those *types of tasks* giving rise to *types of types of tasks*. In this case, "requirements specification" and "coding" could be considered as examples of "technical task types", as opposed to "management task types". Finally, during project development, they need to track the execution of individual tasks (e.g. specifying the requirements of the system X). Thus, to describe the conceptualization underlying the software development domain, one needs to represent entities of different (but nonetheless related) classification levels,

such as *tasks (specific individual occurrences)*, *types of tasks*, and *types of types of tasks*. Other examples of multiple classification levels come from domains such as that of organizational roles (or professional positions) [8], biological taxonomy [26] and artifact types (e.g., product types) [29].

These subject domains require us to break the two-level divide between classes and instances, admitting classes that are also instances of other classes, and suggesting that there could be a multitude of classification 'levels' or meta-levels. The need to support the representation of subject domains dealing with multiple classification levels has given rise to what has been referred to as multi-level modeling [21]. Techniques for multi-level conceptual modeling must provide modeling concepts to deal with types in various levels and the relations that may occur among them. In the last decades, several approaches for the representation of multi-level models have been worked out, including those mostly focused on multi-level modeling from a model-driven engineering perspective (e.g. [14, 23]) and those that propose modeling languages for models with multiple levels of classification (e.g. [3, 22]). These approaches embody conceptual notions that are key to the representation of multi-level models, such as the existence of entities that are simultaneously types and instances, the iterated application of instantiation across an arbitrary number of (meta)levels, the possibility of defining attributes and values at the various type levels, etc.

Despite the recent advances in multi-level modeling approaches and tools, the literature on multi-level modeling still lacks a language-independent formal theory that captures the foundational concepts underlying multi-level modeling. We believe that such a theory could facilitate the identification of the characterizing features a multi-level approach should possess, being useful to support the proposal of well-founded multi-level modeling approaches. Further, it could be used as a foundation to clarify the semantics of existing approaches as well as to relate and harmonize different approaches to multi-level modeling.

In the past, the search for such a theory has led us to propose the MLT multi-level modeling theory [10]. MLT is founded on a basic instantiation relation and characterizes the concepts of individuals and types, with types organized in 'orders' and related by instantiation. MLT has been used successfully to analyze and improve the UML support for modeling the powertype pattern [11], to uncover problems in multi-level taxonomies on the Web [6], to found an OWL vocabulary that supports the representation of multi-level vocabularies in the Semantic Web [5], and to provide conceptual foundations for dealing with types at different levels of classification both in core [8] and in foundational ontologies [9].

While the theory has been fruitful for these applications, it is unable to account for types that defy a strict stratification scheme. This rules out abstract and general types such as "Entity" and "Type" (which are instances of themselves). We have observed that these types correspond to general notions that are ubiquitous in comprehensive conceptualizations (see e.g., the core of the Semantic Web with the notion of "Resource" or "Thing" ([31, 32]), (Foundation) Ontologies such as UFO ([16]), Cyc ([13]), DOLCE and BFO ([25]) with their notions of "Entity" or "Thing", Telos ([28]) with the notions of "Property"). Failure to account for such types restricts the generality of the theory, which motivates us to extend it.

This paper presents MLT*, which extends MLT with a focus on improving its generality. The theory is formally defined through axiomatization in first-order logic, building up on a primitive instantiation relation. In order to account for types that defy a strict classification scheme, we introduce the notions of orderless and ordered types. We precisely define the relations that may occur between orderless and ordered types, and define rules that apply to these relations. We show that the two-level scheme and the strictly stratified schemes are special cases allowed by the theory. We further show how MLT* is general enough to account for the types that are used in its own definition. We aim to provide a theory that is *comprehensive* enough as a semantic foundation for various multi-level modeling techniques.

All definitions of MLT* have been specified and tested with the support of Alloy [19]. Alloy allows the specification of first-order logic based models and supports model simulation (model finding) and verification (model checking) through exhaustive search in finite models. The rules that arise from the definitions and axioms in MLT* have been defined as assertions and verified in Alloy[1].

This paper is further structured as follows: section 2 presents requirements for a comprehensive theory for multi-level modeling; section 3 presents MLT*, addressing the set of requirements defined in section 2; section 4 discusses the implications of the theory to the practice of multi-level conceptual modeling and finally section 5 presents concluding remarks and topics for further investigation.

2 Requirements for a Comprehensive Multi-Level Theory

We establish here key requirements for a theory on multi-level modeling, substantiating these requirements with sources from the literature on multi-level modeling and justifying them based on the nature of multi-level phenomena.

First of all, an essential requirement for a multi-level modeling theory is to account for *entities of multiple classification levels*, which are related through chains of instantiation between the involved entities (requirement **R1**). This means that the theory must admit entities that represent both types (class) and instances (object) simultaneously [1], diverging thereby from the traditional two-level scheme, in which classification (instantiation) relations are only admitted between classes and individuals.

The size of chains of instantiation may vary according to the nature of the phenomena being captured and according to the model purposes. Because of this, a general-purpose theory should *admit an arbitrary number of classification levels* (**R2**) (including the two-level scheme as a special case). The ability to deal with an arbitrary number of levels is identified as a key a requirement by many authors (e.g., see [14] and [3]). Several examples of three and four level models are available in the literature as well as in structured data repositories such as Wikidata (in which there are more than 17,000 classes involved in multi-level taxonomies [6]).

Further, in previous work, some of us have found empirical evidence to support the claim that representations capturing chains of instantiation can benefit greatly from

[1] See https://github.com/nemo-ufes/mlt-ontology.

rules for organizing entities into levels [6]. We have found that over 87% of the classes in multi-level taxonomies in Wikidata were involved in errors that could have been prevented with some support to detect the inadequate use of instantiation (and its combination with subtyping) [6]. Based on this evidence, we consider that a multi-level modeling theory *should define principles (rules) for the organization of entities into levels* (**R3**). An example of this sort of principle, which is adopted in some prominent multi-level modeling approaches, is the so-called strict metamodeling principle [1], which prescribes the arrangement of elements into levels mandating that elements of a level only instantiate elements of the level immediately above.

While these principles are intended to guide the modeler in producing sound models, they should not obstruct the representation of genuine multi-level phenomena. The strict metamodeling principle, for example, excludes from the domain of enquiry abstract notions such as a universal "Type" or, an even more abstract notion such as "Thing". This is because their instances may be related in chains of instantiation, conflicting with the stratification imposed by the principle. Given that these general notions are ubiquitous in comprehensive conceptualizations (see e.g., the core of the Semantic Web with the notion of "Resource" or "Thing" ([31, 32]), (Foundation) Ontologies such as UFO ([16]), Cyc ([13]), DOLCE and BFO ([25]) with their notions of "Entity" or "Thing", Telos ([28]) with the notions of "Property"), we conclude that a comprehensive multi-level modeling theory should *admit types that defy a strictly stratified classification scheme* (**R4**) (with the general notion of "type" or "class" and the universal notion of "entity" or "thing" as paradigmatic special cases).

Finally, an important characteristic of domains spanning multiple levels of classification is that there are domain rules that apply to the instantiation of types of different levels. For example, in a conceptual model encompassing the notions of "Dog Breed" and "Dog", all instances of "Dog Breed" (e.g. "Collie" and "Beagle") are types whose instances are instances of "Dog". Hence, in this setting, instances of "Dog Breed" specialize "Dog". Given the recurrence of this kind of scenario [24], which in the past motivated the powertype pattern [30], a comprehensive multi-level modeling theory should be able *to account for the rules that govern the instantiation of related types at different levels* (**R5**).

3 MLT*: A Theory for Multi-Level Modeling

This section presents MLT* showing how it satisfies the requirements defined in section 2. Section 3.1 describes basic notions of the theory (in order to satisfy requirements R1 and R2); section 3.2 discusses how types can be organized into strictly stratified levels (addressing R3); section 3.3 accounts for types that defy the rigid stratification scheme (addressing R4); finally, section 3.4 discusses the various structural relations that can be established between types (addressing R5). Throughout the sections we discuss the rules that arise from the formalization of the theory.

3.1 Basic Notions

The notions of *type* and *individual* are central for our multi-level modeling theory. *Types* are predicative entities that can possibly be applied to a multitude of entities (including types themselves). Particular entities, which are not types, are considered *individuals*. Each type is characterized by an *intension, which* is used to judge whether the type applies to an entity (e.g., whether something is a Person, a Dog, a Chair) (it is also called *principle of application* in [16]). If the intension of a type *t* applies to an entity *e* then it is said that *e is an instance of t.* Thus, the *instance of* relation (or *instantiation* relation) maps a type to the entities that fall under the type. The set of instances of a type is called the *extension* of the type [17]. We assume that the theory is only concerned with types with non-trivially false intensions, i.e., with types that have possible instances in the scope of the conceptualization being considered.

MLT* is formalized in first-order logic, quantifying over all possible individuals and types in a subject domain. The theory is built up from the instantiation relation, which is formally represented by a binary predicate iof(*e*,*t*) that holds if an entity *e* is instance of an entity *t* (denoting a type). For instance, the proposition iof(John,Person) denotes the fact that "John" is an instance of the type "Person". Note that here we do not account for modal or temporal aspects of instantiation; see [10] for a treatment of modal aspects where instantiation is 'world-indexed' and represented with a ternary predicate.

Using the iof predicate, we can define the ground notion of *individual* (D1). An entity is an individual iff it does not possibly play the role of type in instantiation relations. Conversely, an entity is a type iff it plays the role of type in instantiation relations, i.e., if there is some (possible) entity which instantiates it (D2). Definitions D1 and D2 create a dichotomy with all elements in the domain of quantification being considered either types or individuals.

$$\forall x(\text{individual}(x) \leftrightarrow \neg\exists y(\text{iof}(y,x))) \quad \text{(D1)}$$

$$\forall x(\text{type}(x) \leftrightarrow \exists y(\text{iof}(y,x))) \quad \text{(D2)}$$

We assume that all types are ultimately grounded on individuals (A1). Thus the transitive closure of the instantiation relation (iof'), always leads us from a type to one or more individuals:

$$\forall t(\text{type}(t) \rightarrow \exists x(\text{individual}(x) \wedge \text{iof}'(x,t))) \quad \text{(A1)}$$

Note that the definitions so far allow us to satisfy R1, as we place no restrictions on the kinds of entities that may instantiate a type. Thus, the theory would admit a model such as the one illustrated in Fig. 1. The figure depicts a chain of instantiation, with "Man" and "Woman" instantiating "PersonTypeByGender", and "John" and "Bob" instantiating "Man", while "Ana" instantiates "Woman". We use a notation inspired in the class and object notations of UML, and we use dashed arrows to represent relations that hold between the elements, with labels to denote the relation that applies (in this case *instance of*). This notation is used in all further diagrams in this paper. It is important to highlight here that our focus is not on the syntax of a multi-level model-

ing language and we use these diagrams to illustrate the concepts intuitively. Further, no constraint is placed on the size of instantiation chains, and thus, the theory would admit a model such as the one illustrated in Fig. 2 (satisfying R2).

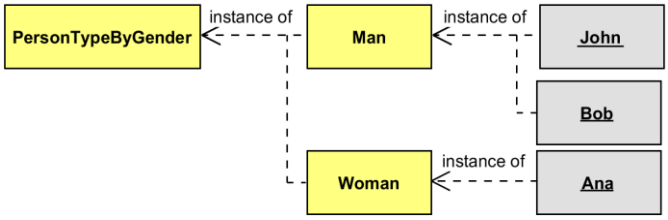

Fig. 1. An instantiation chain, where "Man" and "Woman" are both instances and classes.

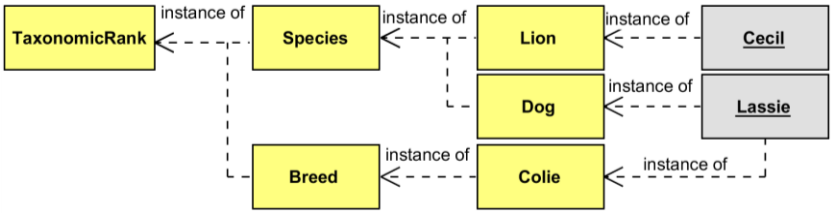

Fig. 2. A four-level instantiation chain with representing a biological domain.

We define some basic structural relations, starting with the ordinary specialization between types. A type *t specializes* another type *t'* iff in all possible instances of *t* are also instances of *t'*. According to this definition every type *specializes* itself. Since this may be undesired in some contexts, we define the *proper specialization* relation in which *t proper specializes t'* iff *t specializes t'* and *t* is different from *t'*.

$$\forall t_1, t_2(\text{specializes}(t_1, t_2) \leftrightarrow (\text{type}(t_1) \wedge \forall e(\text{iof}(e, t_1) \rightarrow \text{iof}(e, t_2)))) \quad \text{(D3)}$$

$$\forall t_1, t_2(\text{properSpecializes}(t_1, t_2) \leftrightarrow (\text{specializes}(t_1, t_2) \wedge \neg(t_1 = t_2))) \quad \text{(D4)}$$

We consider two types equal iff the sets of all their possible instances are the same [10][2]. This definition of equality only applies to elements which are not *individuals*, hence the 'guard' conditions on the left-hand side of the implication:

$$\forall t_1, t_2((\text{type}(t_1) \wedge \text{type}(t_2)) \rightarrow (t_1 = t_2) \leftrightarrow \forall x(\text{iof}(x, t_1) \leftrightarrow \text{iof}(x, t_2))) \quad \text{(D5)}$$

Building up on the specialization definition, we can now address the notion of powertype. Here we employ the seminal notion proposed by Cardelli [7]. According to [7], the same way specializations are intuitively analogous to subsets, *power types* can be intuitively understood as powersets. The powerset of a set A, is the set whose elements are **all** possible subsets of A including the empty set and A itself. Thus, "if A is a type, then Power(A) is the type whose elements are **all** the subtypes of A" (including A itself) [7]. Following Cardelli's definition, we define that a type *t1 is power*

[2] See [10] for a refinement of identity and specialization concerning modal distinctions.

type of a type *t2* iff all *instances of t1* are *specializations of t2* and all possible *specializations of t2* are *instances of t1*. In this case, *t2* is said the base type of *t1*:

$$\forall t_1, t_2(\text{isPowertypeOf}(t_1, t_2) \leftrightarrow (\text{type}(t_1) \wedge \forall t_3(\text{iof}(t_3, t_1) \leftrightarrow \text{specializes}(t_3, t_2)))) \quad \text{(D6)}$$

Given the definition of power type, it is possible to conclude that each type has at most one *power type* (theorem T1) and that each *type* is *power type of*, at most, one other type (theorem T2). (These theorems are proved in [10], which suggests a concrete syntactic constraint for a multi-level model: only one higher-order type can be linked to a base type through the *is power type of* relation.).

$$\forall p, t(\text{isPowertypeOf}(p, t) \rightarrow \neg\exists p'((p \neq p') \wedge \text{isPowertypeOf}(p', t))) \quad \text{(T1)}$$

$$\forall p, t(\text{isPowertypeOf}(p, t) \rightarrow \neg\exists t'((t \neq t') \wedge \text{isPowertypeOf}(p, t'))) \quad \text{(T2)}$$

3.2 Accounting for Stratification into Orders

Note that, thus far, the theory does not impose a principle of organization for the entities into (strictly stratified) 'levels'. In order to account for such kinds of principles, we use the notion of type order. Types whose instances are individuals are called *first-order types*. Types whose instances are *first-order types* are called *second-order types*. Those types whose extensions are composed of *second-order types* are called *third-order types*, and so on.

Types that follow this strictly ordered scheme are called *ordered* types. To define such a scheme formally, we define a notion of 'basic type'. A basic type is the most abstract type in its type order. For example, "Individual" is a basic type since it is the most abstract of all first-order types, classifying all instances of first-order types, i.e., all possible individuals. We define the constant "Individual" as follows:

$$\forall t((t = \text{Individual}) \leftrightarrow \forall x(\text{individual}(x) \leftrightarrow \text{iof}(x, t))) \quad \text{(A2)}$$

Like "Individual", there are basic types for each subsequently higher order, i.e., every instance of the basic type of an order i (i>1) specialize the basic type of the order immediately below (i-1). This is formalized by D7. (Note that i is only used to improve the intuition in the definition, and is not formally a variable.)

$$\forall b_i(\text{basictype}(b_i) \leftrightarrow (\text{type}(b_i) \wedge ((\forall x(\text{individual}(x) \leftrightarrow \text{iof}(x, b_i)) \vee \exists b_{i-1}(\text{basictype}(b_{i-1}) \wedge \forall t_{i-1}(\text{specialize}(t_{i-1}, b_{i-1}) \leftrightarrow \text{iof}(t_{i-1}, b_i))))))) \quad \text{(D7)}$$

A consequence of this definition of basic type is that the basic type of an order i (i>1) is the *powertype* of the basic type at the order immediately below (i-1), showing that the basic types are formed by the cascaded application of the powertype pattern. This is reflected in the following theorem (T3), which is the result of applying D6 to D7:

$$\forall b_i(\text{basictype}(b_i) \leftrightarrow (\forall x(\text{individual}(x) \leftrightarrow \text{iof}(x, b_i)) \vee$$

$$\exists b_{i-1}(\text{basictype}(b_{i-1}) \wedge \text{isPowertypeOf}(b_i, b_{i-1})))) \quad \text{(T3)}$$

Every ordered type that is not a basic type (e.g., a domain type) is an instance of one of the basic higher-order types (e.g., "1stOT", "2ndOT"), and, at the same time proper specializes the basic type at the immediately lower level (respectively, "Individual" and "1stOT"). Fig. 3 illustrates this pattern. Since "Person" applies to individuals, it is instance of "1stOT" and proper specializes "Individual". The instances of "PersonTypeByGender" are specializations of "Person" (e.g. "Man" and "Woman"). Thus, "PersonTypeByGender" is instance of "2ndOT" and proper specializes "1stOT".

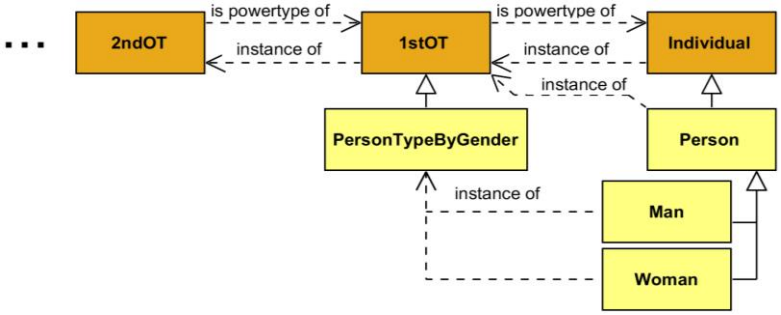

Fig. 3. Illustrating an important basic pattern of MLT and its intra-level structural relations.

Note that, the ellipsis in the left-hand side of the figure indicates that the theory admits an unbound number of higher-order basic types. Nevertheless, we have been careful not to necessitate the existence of such types in the theory. This means that the theory has finite models, and thus can be subject to analysis using a finite model checker/finder such as Alloy, which we have employed for verification of all theorems discussed here.

Having defined the structure of basic types we can define ordered type as a type that specializes one of the basic types (D8). Conversely, we can define *orderless types* as in D9.

$$\forall x(\text{orderedtype}(x) \leftrightarrow \exists b(\text{basictype}(b) \wedge \text{specializes}(x, b))) \quad \text{(D8)}$$

$$\forall x(\text{orderlesstype}(x) \leftrightarrow \text{type}(x) \wedge \neg\text{orderedtype}(x)) \quad \text{(D9)}$$

We can account now for a strictly stratified scheme. In this case, it would suffice to add an axiom stating that all types are ordered types, which would rule out types whose instances belong to different orders. The stratified scheme is thus a restriction of the more general theory we have, which admits *orderless types*.

Moreover, we can see that the theory can be further constrained to account for the two-level scheme as a particular case. For a two-level theory it would suffice to add to the strictly stratified scheme an axiom stating that there is a unique basic type (which would be "Individual").

3.3 Beyond Strictly Stratified Types

While a strictly stratified approach imposes a useful principle of organization for entities in multi-level models, it rules out types whose instances transcend this strict structure, i.e., types that have instances belonging to different levels or strata. For example, consider the type whose instances are all types admitted ("Type"). This type itself defies stratification into orders, since its instances are types at various different orders (e.g., "Lion", "Species", "Taxonomic Rank", etc.).

In order to capture the strictly stratified scheme while still guaranteeing the generality of the theory, we distinguish types into "OrderedType" (A3) and "OrderlessType" (A4). Instances of "OrderedType" are those types that fall neatly into a particular order. Instances of "OrderlessType" are those types whose instances belong to different orders. This constitutes a dichotomy, and together, "OrderedType" and "OrderlessType" form the notion of "Type" (A5), which classify all possible types. In their turn "Type" and "Individual" (A2) together form the universal notion of "Entity" (A6), which classify all possible entities (types and individuals).

$$\forall t(t = \text{OrderedType} \leftrightarrow \forall x(\text{orderedtype}(x) \leftrightarrow \text{iof}(x,t))) \quad \text{(A3)}$$

$$\forall t(t = \text{OrderlessType} \leftrightarrow \forall x(\text{orderlesstype}(x) \leftrightarrow \text{iof}(x,t))) \quad \text{(A4)}$$

$$\forall t((t = \text{Type}) \leftrightarrow \forall x(\text{type}(x) \leftrightarrow \text{iof}(x,t))) \quad \text{(A5)}$$

$$\forall t(t = \text{Entity} \leftrightarrow \forall x(\text{iof}(x,t))) \quad \text{(A6)}$$

The classification scheme formed by MLT* is presented in Fig. 4. A number of interesting observations can be made about the top-layer of MLT*. First of all, MLT*, differently from MLT, is able to account for the types used in its definition. All entities admitted are instances of "Entity", including all possible types and all possible individuals. All possible types are instances of "Type" and ultimately specializations of "Entity" (since their instances are entities). "Type" is thus the *powertype of* "Entity". All elements added in MLT* are instances of "OrderlessType", including (curiously) "OrderedType" (since its instances are types at different orders).

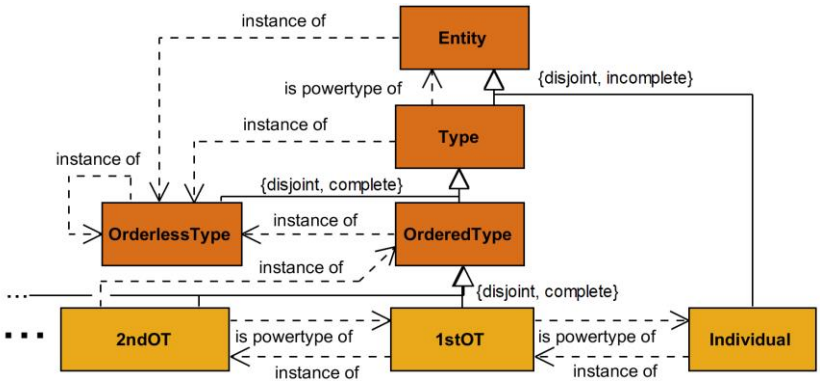

Fig. 4. MLT* classification scheme.

The instantiation relation has the following logical properties as a consequence of the definitions and axioms of the theory: whenever instantiation involves solely ordered types, it is *irreflexive*, *antissymetric* and *antitransitive*, leading to a strict stratification of types. When instantiation involves any orderless types, none of these properties can be asserted, as there are situations in which it is *reflexive* (e.g., "Type" is instance of itself), *symmetric* (e.g., "Entity" is instance of "Type" and vice-versa) as well as *transitive* (e.g., "OrderedType" is instance of "Type" which is instance of "Entity" and "OrderedType" is also instance of "Entity"). Further, an orderless type is never an instance of an ordered type. These characteristics of instantiation can be used to rule out models that violate the theory.

Table 1 summarizes the rules that concern which types of entities may be related through structural relations along with the logical properties of these relations.

Table 1. Summary of constraints on MLT* relations.

Relation (t → t')	Domain	Range	Constraint	Properties
specializes(t,t')	Orderless	Orderless	if *t* and *t'* are ordered types, they must be at the same type order	Reflexive, antissymetric, transitive
	Ordered	Orderless		
	Ordered	Ordered		
properSpecializes(t,t')	Orderless	Orderless		Irreflexive, antissymetric, transitive
	Ordered	Orderless		
	Ordered	Ordered		
isPowertypeOf(t,t')	Orderless	Orderless	*t* cannot be a first-order type if *t* and *t'* are ordered types, *t* must be at a type order immediately above the order of *t'*	Irreflexive, antissymetric, antitransitive
	Ordered	Ordered		

The notion of "Orderless Type" is useful not only for the domain-independent entities forming MLT*, but also for general notions in specific subject domains. Consider, for example, the domain of social entities in which a "Social Entity" is defined as an entity that is created by a social normative act. Instances of "Social Entity" include specific states of Brazil (individuals) such as "Rio de Janeiro" and "Espírito Santo", but also the first-order type "State" of which "Rio de Janeiro" and "Espírito Santo" are instances. As "SocialEntity" has instances at different orders (types and individuals), it is an instance of "OrderlessType", as shown in Fig. 5. The example also highlights that MLT* allows entities to have multiple instantiation relations. "RioDeJaneiro" and "EspíritoSanto", are both instances of "SocialEntity" and "State". Moreover, multiple specializations are also allowed in MLT*. In this sense, MLT* differs from a number of approaches in literature which limit these structural relations to a single class (see [24]).

The same mechanism that allows us to model *bona fide* self-instantiating types such as "Entity" and "Type" would permit a modeler to introduce paradoxical types, such as the type of all types that are not self-instantiated (the so-called Russellian property, due to Russell [18]). This type is paradoxical since it is both an instance and not an instance of itself. Note that this possibility does not threaten the overall consistency of the theory. This is because we do not assume in MLT* that there are types corresponding to any expressible unifying condition (i.e., we do not assume that given an arbitrary logical condition *F*, we can define the type with extension $[x \mid F(x)]$).

Types here, instead, are explicitly recognized entities describing intentionally identified properties shared by their instances. Lacking the ability to prove or introduce the existence of types in this sense, we are under no threat of such paradoxes [27].

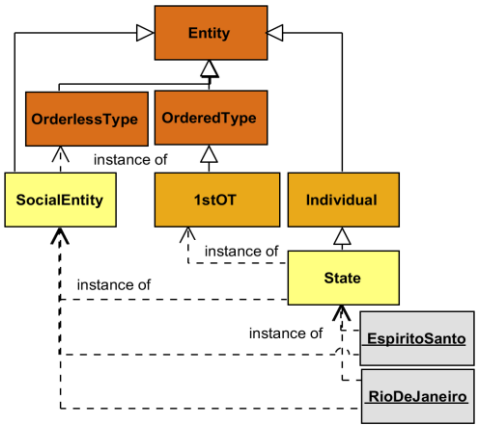

Fig. 5. Example of orderless type in domain model.

3.4 Cross-level Structural Relations

So far, the only *cross-level structural relations* we have considered is Cardelli's power type relation. Another definition of *power type* that has had great influence in the literature was proposed by Odell [30]. In order to satisfy R5, and account for the variations of the power type pattern in the literature, MLT* defines the categorization cross-level relation based on Odell's notion power type.

A type *t categorizes* a type *t'* iff all instances of *t* are proper specializations of *t'*. Note that, differently from the *is powertype of* relation (due to Cardelli), *t'* is not an instance of *t*, and further not all possible specializations of *t'* are instances of *t*. For instance, "EmployeeType" (with instances "Manager" and "Researcher") *categorizes* "Person", but is not the *powertype* of "Person", since there are specializations of "Person" that are not instances of "EmployeeType" ("Child" and "Adult" for example).

MLT* also defines some variations of the categorization relation. A type *t completely categorizes* a type *t'* iff every instance of *t'* is instance of at least one instance of *t*. Moreover, a type *t disjointly categorizes* a type *t'* iff every instance of *t'* is instance of at most one instance of *t*. Further, *t partitions t'* iff every instance of *t'* is instance of exactly one instance of *t*. For example, "PersonTypeByGender" *partitions* "Person" into "Man" and "Woman", and thus each instance of "Person" is either a "Man" or a "Woman" and not both. "EmployeeType" incompletely *categorizes* "Person", and thus there are persons that are not instances of "Manager", "Researcher" (or any other possible instance of "EmployeeType"). This kind of constraint is usually represented in UML through a generalization set, see [10] for a detailed comparison.

Rules concerning the types of entities that may be related through the variations of categorization and the logical properties of these relations are summarized in Table 2.

Table 2. Summary of constraints on MLT* categorization relations.

Relation (t → t')	Domain	Range	Constraint	Properties
categorizes(t,t')	Orderless	Orderless	*t* cannot be a first-order type if *t* and *t'* are ordered types, *t* must be at a type order immediately above the order of *t'*	Irreflexive, antissymetric, nontransitive
	Ordered	Orderless		
disjointlyCategorizes(t,t')	Ordered	Ordered		
completelyCategorizes(t,t')	Orderless	Orderless		Irreflexive, antissymetric, antitransitive
partitions(t,t')	Ordered	Ordered		

4 Implications for Multi-Level Modeling Approaches

We have observed in the literature that multi-level approaches often opt for one of two extremes: (i) to define relations that support the representation of instantiation chains, without necessarily binding a type to some level (what is referred to as a level-blind approach in [4], e.g., Kernel [12]), or (ii) to consider all classes to be strictly stratified. Some approaches that opt for (i) are able to account for all types which can be admitted by MLT*, however, they fail on providing rules to guide the use of the various structural relations (including instantiation). As shown in [6], this lack of guidance has serious consequences for the quality of the resulting representation. Approaches that opt for the other end of the spectrum (ii), lack support to a number of important abstract notions, including those very general notions that are used to articulate multi-level domains (such as "types", "clabjects", "entities"). This is the case of Melanee [3] and MetaDepth [22]. The combination of both approaches in our theory places it in a unique position in multi-level modeling approaches.

A few other knowledge representation approaches (such as Telos [20] and Cyc [13]) have, like MLT*, drawn distinctions between orderless and ordered types. Differently from MLT*, however, Telos does not provide rules for the various structural relations, including instantiation and specialization. (Mechanisms to address R5 in Telos were added with the notion of MGI in Deeptelos [21].) In its turn, Cyc, which employs a conceptual architecture for types that is most similar to MLT*'s top layer, includes rules for instantiation and specialization [13]. However, it does not address the cross-level relations (and associated rules) we discuss here.

MLT* shows that there is no dilemma between requirements R3 (to define principles for the organization of entities into levels) and R4 (to admit types that defy a strictly stratified scheme). It suggests the possibility of extending existing multi-level approaches that currently meet R3 but fail to meet R4 in order to meet both. For example, extensions of Melanee and MetaDepth could be worked out to allow some kind of selective stratification, beyond what is currently supported with the so-called star potency, in order to fully enable the representation of orderless types.

Further, since MLT* reveals that there is no inconsistency between powertype and clabject-based approaches, we consider it possible to extend clabject-based approaches such as Melanee and MetaDepth to support the representation of MLT* cross-level relations (in order to satisfy R5). Finally, we consider it possible to extend Deeptelos by including variations of the so-called MGI mechanism to capture the MLT* cross-level relations (and thereby fully address R5 in Deeptelos).

5 Conclusions and Future Work

In this paper, we have proposed a multi-level modeling theory that can account for the classification scheme underlying current multi-level modeling approaches. We have aimed for a simple but comprehensive approach in that it encompasses stratified and non-stratified schemes, and is able to accommodate the variations for the powertype pattern in the literature. We should stress that it is not our intention in this paper to propose a multi-level language, and that our use of a notation inspired in UML has been solely illustrative. As discussed in [16], a reference theory can be used to inform the revision and redesign of a modeling language, not only through the identification of semantic overload, construct deficit, construct excess and construct redundancy, but also through the definition of modeling patterns and semantically-motivated syntactic constraints. Thus, a natural application for MLT* is to inform the design of a well-founded multi-level conceptual modeling language or to promote the redesign of a language such as UML into a multi-level modeling language. This is the subject of ongoing research which will be reported soon.

Due to space limitations, we have not been able to address here the use of features (attributes and associations). In a multi-level context, since types are also instances, feature assignment in types becomes relevant, along with relations between features across different levels. Some of us have already addressed this issue previously [10] using the notion of 'regularity feature' in MLT, however, revisiting the notion in light of MLT* is still the subject of further investigation. This is particularly important to account for the deep characterization mechanisms in potency-based approaches [2].

Acknowledgements. This research is funded by CNPq (grants number 311313/2014-0, 461777/2014-2 and 407235/2017-5), CAPES (23038.028816/2016-41) and FAPES (69382549). Claudenir M. Fonseca is funded by CAPES. We thank Giancarlo Guizzardi for fruitful discussions in topics related to this paper.

References

1. Atkinson, C., Kühne, T.: Meta-level Independent Modelling. In: Intl .Workshop on Model Eng. at 14th European Conf. on Object-Oriented Programming. pp. 1–4 (2000)
2. Atkinson, C., Kühne, T.: Reducing accidental complexity in domain models. In: Software & Systems Modeling, vol. 7, pp. 345–359 (2008)
3. Atkinson, C., Gerbig, R.: Melanie: multi-level modeling and ontology engineering environment. In: Proc. 2nd Int. Master Class on MDE Modeling Wizards, ACM (2012)
4. Atkinson, C., Gerbig, R., Kühne, T.: Comparing Multi-Level Modeling Approaches. In: Proceedings of the 1st International Workshop on Multi-Level Modelling (2014)
5. Brasileiro, F., Almeida, J.P.A., Carvalho, V.A., Guizzardi, G.: Expressive Multi-level Modeling for the Semantic Web. ISWC 2016. LNCS, vol 9981. Springer, pp. 53-69 (2016)
6. Brasileiro, F., Almeida, J.P.A., Carvalho, V.A., Guizzardi, G.: Applying a Multi-Level Modeling Theory to Assess Taxonomic Hierarchies in Wikidata. Proc. Wiki Workshop 2016 at 25th Int. Conf. Companion on World Wide Web, pp. 975-980 (2016)
7. Cardelli, L.: Structural Subtyping and the Notion of Power Type. In Proc. Of the 15th ACM Symposium of Principles of Programming Languages, pp. 70-79 (1988)

8. Carvalho, V.A., Almeida, J.P.A.: A Semantic Foundation for Organizational Structures: A Multi-level Approach. IEEE EDOC 2015, pp. 50–59 (2015)
9. Carvalho, V.A., Almeida, J.P.A., Fonseca, C.M., Guizzardi, G.: Extending the Foundations of Ontology-based Conceptual Modeling with a Multi-Level Theory. In: ER (2015)
10. Carvalho, V. A., Almeida, J. P. A.: Towards a well-founded theory for multi-level conceptual modeling. In: Software & Systems Modeling, Springer, pp. 1-27 (2016)
11. Carvalho, V.A., Almeida, J.P.A., Guizzardi, G.: Using a Well-Founded Multi-Level Theory to Support the Analysis and Representation of the Powertype Pattern in Conceptual Modeling. In: 28th Intl. Conf. on Advanced Information Systems Engineering (2016)
12. Clark, T., Gonzalez-Perez, C., Henderson-Sellers, B.: A Foundation for Multi-Level Modelling. In: Proc. Workshop on Multi-Level Modelling, MODELS, pp. 43-52 (2014)
13. Foxvog, D.: Instances of Instances Modeled via Higher-Order Classes. In: Foundational Aspects of Ontologies (FOnt 2005), 28th German Conf. on AI, pp. 46-54 (2005)
14. Frank, U.: Multilevel Modeling. In: Business & Information Systems Engineering, vol. 6, pp. 319–337 (2014)
15. Gonzalez-Perez, C., Henderson-Sellers, B.: A powertype-based metamodelling framework. In: Software & Systems Modeling, vol. 5, pp. 72–90 (2006)
16. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. University of Twente, Enschede, The Netherlands (2005)
17. Henderson-Sellers, B.: On the Mathematics of Modeling, Metamodelling, Ontologies and Modelling Languages. Springer (2012)
18. Irvine, A. D., Deutsch, H.: Russell's Paradox. In: The Stanford Encyclopedia of Philosophy, https://plato.stanford.edu/archives/win2016/entries/russell-paradox/ (2016)
19. Jackson, D.: Software Abstractions: Logic, Language and Analysis. The MIT Press (2006)
20. Jarke, M., et al.: ConceptBase - A deductive object base for meta data management. In: Journal of Intelligent Information Systems, vol. 4, pp. 167–192 (1995)
21. Jeusfeld, M. A., Neumayr, B.: DeepTelos: Multi-level Modeling with Most General Instances. In: 35th International Conference, ER 2016. Springer, pp. 198–211 (2016)
22. Lara, J. de, Guerra, E.: Deep Meta-modelling with MetaDepth. In: Proceedings of the 48th International Conference, TOOLS 2010, Málaga, Spain (2010)
23. Lara, J. de, et al.: Extending Deep Meta-Modelling for Practical Model-Driven Engineering, The Computer Journal (2013)
24. Lara, J. de, Guerra, E., Cuadrado, J.S.: When and How to Use Multilevel Modelling. In: ACM Transactions on Software Engineering and Methodology, vol. 24, pp. 1–46 (2014)
25. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A.: Ontology Library. In: WonderWeb Deliverable D18 (2003)
26. Mayr, E.: The Growth of Biological Thought: Diversity, Evolution, and Inheritance. In: The Belknap Press (1982)
27. Menzel, C.: Knowledge representation, the World Wide Web, and the evolution of logic. In: Synthese vol. 182, pp. 269-295 (2011)
28. Mylopoulos, J., et al.: Telos: Representing Knowledge About Information Systems. In: ACM Transactions on Information Systems (TOIS), vol. 8, pp. 325–362 (1990)
29. Neumayr, B., Grün, K., Schrefl, M.: Multi-level domain modeling with m-objects and m-relationships. In: Proc. 6th Asia-Pacific Conf. Conceptual Modeling, New Zealand (2009)
30. Odell, J.: Power types. In: Journal of Object-Oriented Programing, 7(2), pp. 8-12.(1994)
31. W3C: RDF Schema 1.1. https://www.w3.org/TR/2014/REC-rdf-schema-20140225/ (2014)
32. W3C: OWL 2 Web Ontology Language -Document Overview (Second Edition). https://www.w3.org/TR/2012/REC-owl2-syntax-20121211 (2012)