Expressive Multi-Level Modeling for the Semantic Web

Freddy Brasileiro[1], João Paulo A. Almeida[1],
Victorio A. Carvalho[1,2] and Giancarlo Guizzardi[1]

[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
freddybrasileiro@gmail.com, jpalmeida@ieee.org,
victorio@ifes.edu.br, gguizzardi@inf.ufes.br

Abstract. In several subject domains, classes themselves may be subject to categorization, resulting in classes of classes (or "metaclasses"). When representing these domains, one needs to capture not only entities of different classification levels, but also their (intricate) relations. We observe that this is challenging in current Semantic Web languages, as there is little support to guide the modeler in producing correct multi-level ontologies, especially because of the nuances in the constraints that apply to entities of different classification levels and their relations. In order to address these representation challenges, we propose a vocabulary that can be used as a basis for multi-level ontologies in OWL along with a number of integrity constraints to prevent the construction of inconsistent models. In this process we employ an axiomatic theory called MLT (a Multi-Level Modeling Theory).

Keywords: Multi-level modeling, metamodeling, semantic web, OWL

1 Introduction

The Semantic Web, or Web of Data, provides a common framework that allows data to be shared across application, enterprise, and community boundaries [1]. This is achieved by linking and publishing structured data using RDF languages, which provide a basis for producing reusable vocabularies for various domains of interest [2].

A Semantic Web vocabulary is built using the basic notion of *class*, which is present in both RDF Schema (RDFS) [3] and in the Web Ontology Language (OWL) [4]. A class (or type) is a ubiquitous notion in modern conceptual modeling approaches and is used to establish invariant features of the entities in a domain. Often, subject domains are conceptualized with entities in two levels: a level of classes, and a level of individuals which instantiate these classes. In many subject domains, however, classes themselves may also be subject to categorization, resulting in classes of classes (or metaclasses). For instance, consider the domain of biological taxonomies [5]. In this domain, a given *organism* is classified into *taxa* (such as, e.g., *Animal*, *Mammal*, *Carnivoran*, *Lion*), each of which is classified by a *biological taxonomic rank* (e.g., *Kingdom*, *Class*, *Order*, *Species*). Thus, to represent the knowledge underlying this domain, one needs to represent entities at different (but nonetheless related) classification levels. For example, *Cecil* (the lion killed in the Hwange National Park in Zimbabwe in 2015) is an instance of *Lion*, which is an instance of *Species*. *Species*, in its turn, is an instance of *Taxonomic Rank.* Other examples of multiple classification levels come from domains such as software development [6] and product types [7].

The need to support the representation of knowledge domains dealing with multiple classification levels has given rise to an area of investigation called *multi-level modeling* [7, 8]. A number of research initiatives have also been conducted to support multi-level modeling in the Semantic Web (e.g., [9–12]). These approaches exploit the fact that a class is itself an RDF resource and may thus be the subject or object of triples. OWL 2 explicitly adopts this strategy under the term "metamodeling", enabling the representation of facts that are stated about classes [13].

Despite these developments, the current support for the representation of *domains dealing with multiple levels of classification* in the web still lacks a number of important features. In some cases, there are no criteria or principles for the organization of vocabularies into levels, leading to problematic classification and taxonomic statements (see, e.g. [14]). Further, there has been no attention to the representation of the relations between types at different levels. For example, in the biological domain, it is key to represent that all instances of *Species* are subtypes of *Organism* (even when particular species are not represented explicitly), and that all instances of *Organism* belong to one and only one *Kingdom.*

In this paper, we address the challenges in the representation of domains with multiple levels of classification in the Semantic Web by proposing an OWL vocabulary that can be used as a basis for multi-level ontologies. By defining a taxonomy of reusable relations between types, the vocabulary enables the expression of domain rules that would otherwise not be captured. The vocabulary is based on a reference axiomatic theory called MLT [15]. The axioms and theorems of MLT are used to derive integrity constraints for multi-level vocabularies, offering guidance to prevent the construction of inconsistent vocabularies. Further, MLT rules are used to infer knowledge about the relations between types that is not explicitly stated. We focus on the support for domain metaclasses as opposed to language metaclasses, i.e., we focus on ontological instantiation instead of linguistic instantiation [16].

This paper is further structured as follows: Section 2 presents basic requirements for the representation of knowledge in domains with multiple classification levels; Section 3 reviews the current support for multi-level modeling in OWL as well as in related work in the literature; Section 4 presents briefly the MLT multi-level theory; Section 5 presents our approach to represent multi-level models in OWL reflecting the rules of MLT; and Section 5.3 presents concluding remarks.

2 Requirements for a Multi-Level Approach

An essential requirement for a multi-level modeling approach is *the ability to represent entities of multiple (related) classification levels*, capturing chains of instantiation between the involved entities (requirement **R1**). To comply with this requirement, the approach must admit entities that are, simultaneously, type (class) and instance (object) [17]. This means that a multi-level approach differs from the traditional two-level scheme, in which classification (instantiation) relations can only be established between classes and individuals. As a consequence, a multi-level modeling approach *should define principles for the organization of entities into levels* (**R2**). These principles should guide the modeler on the adequate use of classification (instantiation) relations. An example of this sort of principle, which is adopted in some prominent multi-level modeling approaches, is the so-called strict metamodeling principle [17]. It assumes that each element of a level must be an instance of an element of the level above. The lack of principles to guide organization of entities into levels may lead to the construction of unsound multi-level models. For example, in [14] we assessed Wikidata and found over 22,000 violations of the strict metamodeling principle. The identified problems seem to arise from inadequate use of instantiation and subclassing and could have been prevented with guidance from the editing/modeling environment.

Another important characteristic of domains with multiple levels of classification is that there are rules that apply to the instantiation of types of different levels. This kind of rule is present in an early and important approach for multi-level modeling, named the *powertype pattern* [18, 19], which establishes a relationship between two types such that the instances of a type (the so-called "powertype" or "higher-order" type) are specializations of a lower-level type (the so-called "base type"). For example, all instances of *Dog Breed* (e.g. *Collie* and *Beagle)* specialize the base type *Dog*. In order to represent *Dog Breed*, it is thus key to establish its relation with the *Dog* type (we call this sort of relation a *structural relation,* as it governs the instantiation of types at different levels). Further, one may need to represent whether an instance of *Dog* may instantiate: (i) only one, or (ii) more than one *Dog Breed*. In biological taxonomy, another rule concerning instantiation of types at different levels is that the instances of *Biological Taxonomic Rank* obey a sort of subordination chain such that every instance of *Phylum* specializes one instance of *Kingdom*, every instance of *Class* specializes one instance of *Phylum*, and so on. Thus, an expressive multi-level approach should be able *to capture rules for the instantiation of types at different levels* (**R3**).

Finally, in various domains, there are relations which may occur between entities of different classification levels. For example, consider the following domain rules: (i) each *Car* has an *owner* (a *Person*), (ii) each *Car* is classified as instance of a *Car Model*, and (iii) each *Car Model* is designed by a *Person.* In this domain, instances of *Person* (individuals) must be related simultaneously with instances of *Car Model* (which are classes) and also with instances of *Car*, i.e., instances of instances of *Car Model.* Thus, a *multi-level modeling approach should allow the representation of domain relations between entities in different classification levels* (**R4**).

3 Related Work

In this section, we review existing approaches to support the representation of multiple levels of instantiation, with a focus on multi-level modeling in RDF languages.

3.1 RDFS(FA)

In an early effort to organize the metamodeling architecture for RDF Schema (RDFS) 1.0 [20], Pan and Horrocks proposed RDFS(FA) [9]. They observed that "RDFS uses a single primitive *rdfs:Class* to implicitly represent possibly infinite layers of classes" (as it is an instance of itself) and that this creates barriers for understanding. They show examples on how this lack of a principle of organization for levels creates a so-called "layer mistake". Inspired by the fixed UML metamodeling architecture [21], they proposed the use of four layers: Metalanguage (M), Language (L), Ontology (O) and Instance (I). The *M Layer* is responsible for defining the language, where modelling primitives of this topmost layer have no types. The *L Layer* defines a language for specifying vocabularies and each entity in this layer is an instance of an entity in the *M Layer*. Vocabularies are defined in the *O Layer* ("Person" and "Animal" are examples of classes in this layer) and each element in this layer is an instance of an element in the *L Layer*. Lastly, the *I Layer* is populated with concrete individuals, which are instances of the vocabulary defined in *O Layer*.

Fig. 1 shows the result of applying this architecture to RDFS. RDFS classes are replicated in the *M* and *L Layers* with the respective prefix (*M* and *L*). In *O* layer, *Animal* and *Person* are represented as instances of *rdfsfa:LClass* (instead of *rdfs:Class*); and *John* and *Mary* in the *Instance Layer*, as an instance of *Person*.

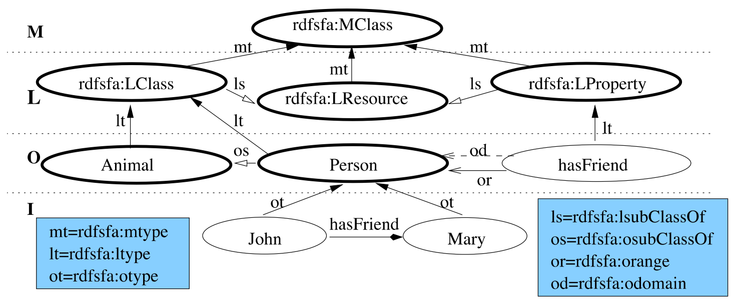

Fig. 1. Example of Directed Labeled Graph of RDFS(FA) (from [9])

This architecture organizes the language engineering effort, but it does not aim to address the representation of domains with multiple levels of classification. In fact, it is based on the two-level scheme for the representation of domains in the *O* and *I* layers, with classes at the *O* layer, and individuals at the *I* layer, related through *rdfsfa:otype* (which represents what is known as ontological instantiation [16]). Metaclasses are only used in the *domain-independent L* layer; classes at the *O* layer are related to classes at the *L* layer through *rdfsfa:ltype* (which represents what is known as linguistic instantiation [16]). In order to represent a domain type such as *Species* one would be forced to include it in the *L* layer, specializing *rdfsfa:LClass*, which would be inadequate according to [9], as language and ontology issues would be confused. In this case, one would have to instantiate *Species* using *rdfsfa:ltype*, clearly misusing linguistic instantiation [16]. In conclusion, RDFS(FA) satisfies requirements R1 and R2 only for linguistic instantiation, but not for ontological instantiation.

3.2 OWL 2

OWL 2 [4] explicitly introduced support for *metamodeling*, enabling the representation of classes of classes. For example, in Fig. 2, two subclasses of *Eagle*, namely *Golden Eagle* and *Steppe Eagle* are defined as instances of *Species*, which means that they are member of the set of all species. In Fig. 2 (as well as in the remainder of the paper) we use a notation that is largely inspired in UML. We use UML specialization to represent the *rdfs:subClassOf* properties, and dashed arrows to represent statements, with labels to denote the names of the predicates that apply. For instance, a dashed arrow labeled *rdf:type* between *Golden Eagle* and *Species* represents that the former is an instance of the latter. Finally, we use the instance specification notation (i.e., underlining an element's name) to represent an individual (e.g. Harry).

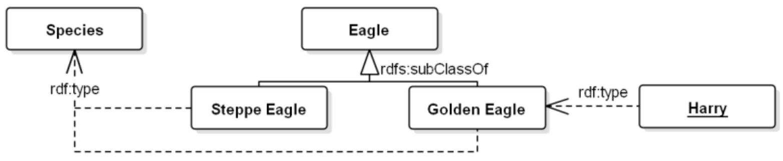

Fig. 2. OWL representation for biological taxonomic domain

OWL's multi-level modeling support is based on the notion of contextual semantics [10], often referred to as "punning", which means that a class is seen as an individual when it is an instance of another class, and that its interpretation as a class and as an individual are completely independent of each other. This "independent" interpretation means that a constraint stated to a class will not be considered when it is seen as an individual, which leads to non-intuitive interpretations [11]. For instance, consider the following statements: (i) *Harry* is an instance of *Golden Eagle*, and; (ii) *Golden Eagle* is the same as *Aquila chrysaetos*. Statement (i) treats *Golden Eagle* as a class, while statement (ii) treats *Golden Eagle* as an individual. These two aspects of *Golden Eagle* are never considered at the same time for reasoning. Thus, in this approach, it is impossible to infer that Harry is an instance of *Aquila chrysaetos*, which violates our intuition with respect to the multi-level model. We can say that while OWL 2 seems to satisfy R1 (admitting classes that are also instances), it does so only partially, given the notion of contextual semantics employed. The same can be said for the representation of relations between entities of different levels (partially satisfying R4).

OWL offers no principle of organization into levels (failing to satisfy R2). Further, punning also prevents us from correctly expressing the relation between a higher-order class and a base class in the powertype pattern, which inevitable involves considering the specializations of the base class as types and instances simultaneously (failing thus to satisfy R3). Finally, considering the open world assumption, it is also impossible to formally identify in this fragment above that Harry is an individual, as there could be unstated *rdf:type* declarations involving Harry as a class. Further, given the same assumption, it would be impossible to identify that *Species* (in isolation) is a metaclass; in other words, we cannot express when modeling *Species* (and omitting its instances) that all its instances are classes (in particular subclasses of *Organism*).

3.3 OWL FA

Later, Pan and Horrocks also proposed OWL FA [11], a metamodeling extension of OWL 1 DL, with an architecture based on RDFS(FA). They argue that OWL 1 Full supports some metamodeling by allowing users to use the built-in vocabulary without restrictions, but that leads to undecidability (as Motik pointed out [10]). They then propose a decidable extension of OWL 1 DL that can reuse existing reasoners.

While RDFS(FA) uses prefixes (M, L, O and I) to indicate the layer in which a class or axiom belongs, OWL FA intuitively introduces a layer number in its constructors and axioms, through annotations. The semantics of OWL FA [22, 23] takes into account elements that share the same URIs and interpret them dependently (in contrast to OWL 2). For instance, if *Golden Eagle* and *Aquila chrysaetos* are stated as the same and *Harry* is an instance of *Golden Eagle*, OWL FA assumes that *Harry* must be an instance of *Aquila chrysaetos*. However, it does not allow property assertions between layers except for instantiation. For example, subclassing and domain relations must be between classes at the same layer (failing thus to satisfy R4).

While RDFS(FA) allows instantiations only from *Instance Layer to Ontology Layer*, OWL FA allows the representation of multiple levels of instantiation. Thus, we understand here that identifying layers by numbers addresses the limitation of RDFS(FA) (see 3.1) thus satisfying R1 fully. Moreover, as advantages when compared to the current *multi-level modeling* support of OWL 2 (see 3.2), OWL FA: (i) interprets dependently elements that share the same URI, and; (ii) it introduces restrictions for instantiation and subclassing, providing some criteria for the organization into levels (R2). Finally, OWL FA offers no special support for the representation of constraints for the instantiation of types at different levels (not satisfying R3).

3.4 PURO

Svatek et al. [12] proposed the PURO approach which includes an OWL vocabulary that can be used as a basis for multi-level domain vocabularies. In PURO, each entity of a domain vocabulary can be annotated with a PURO term in order to clarify the entity's ontological status. The term *B-object* is used to refer to concrete individuals in the world (such as *Harry*). In contrast, the term *B-type* is used to refer to classes (such as *Eagle*). A *B-type* is analogous to an OWL class, however, *B-types* are organized into strata: instances of *1st order B-types* are *B-objects*, instances of *nth-order B-types* are *(n−1)th-order B-types* (for n > 1). The OWL vocabulary supporting the PURO approach only deals with *B-objects* and first-, second- and third-order *B-types*. *B-relationship* is analogous to an object property assertion and there are variations: (i) *B-instantiation* is an assertion to indicate that an entity instantiates a *B*-type; (ii) *B-axiom* express a relationship between the extensions of two *B-types* (e.g., subclassing); and (iii) *B-fact* express information about an entity, e.g., who discovered certain species. Finally, *B-relation* is analogous to OWL Object Property.

Similarly to OWL 2 and OWL FA, PURO has the required expressivity for representing multiple levels of instantiation (R1) through the notions of *B-object* and the *B-types*. Moreover, PURO defines rules for the organization of entities along levels (R2). Finally, PURO allows modelers to express domain relations between entities of different levels (R4); an example is provided in [12] in which a musician is considered an expert in a *type* of instrument (e.g., the musician *Yo-Yo Ma* is an expert in *Violin*). However, similarly to OWL 2 and OWL FA, PURO offers no special support for the representation of constraints for the instantiation of types at different levels (not satisfying R3).

3.5 Intermediate Conclusions

Table 1 summarizes the current support provided by each of the efforts discussed here according to the requirements defined in Section 2. We classified this support in three categories: fully covered (+), partially covered (+/-) and not covered (-). Despite providing support and guidance for representing multiple levels of classification, RDFS(FA) focuses on linguistic instantiation instead of ontological instantiation, hence the partial support for R1 and R2. OWL 2 fails in the representation of relations and constraints crossing levels, due to its contextual semantics, and hence offers partial support for R1 and R4. OWL FA and PURO offer full support for R1 and R2 through annotations, and PURO also supports domain relations crossing levels (R4). Despite the efforts in all these approaches, none of them support the representation of constraints involving instantiation relations across levels (thus, not satisfying R3).

Table 1. Support for multi-level modeling in RDFS languages

Requirement	**RDFS(FA)**	**OWL 2**	**OWL FA**	**PURO**
R1 – represents entities of multiple levels of classification	+/-	+/-	+	+
R2 – offers guidance for the organization of entities into levels	+/-	-	+	+
R3 – represents rules for the instantiation of types at different levels	-	-	-	-
R4 – supports domain relations between entities of different levels	-	+/-	-	+

4 MLT: A Theory for Multi-Level Modeling

Motivated by the lack of theoretical foundations for multi-level modeling, some of us have proposed a formal axiomatic theory called MLT [15] founded on the notion of (ontological) instantiation. MLT has been used successfully to analyze and improve the UML support for modeling the powertype pattern [24], to uncover problems in multi-level taxonomies on the web [14] and to provide conceptual foundations for dealing with types at different levels of classification both in core [25] and in foundational ontologies [26].

The theory is defined using first-order logic, quantifying over all possible entities (individuals and types). The *instance of* relation is represented in this formal theory by a binary predicate *iof(e,t)* that holds if an entity *e is instance of* an entity *t* (denoting a type). In order to accommodate the varieties of types in the multi-level setting, the notion of *type order* is used. Types having individuals as instances are *first-order types*, types whose instances are first-order types are *second-order types* and so on.

The logic constant "Individual" is used to define the conditions for entities to be considered individuals: *an entity is an instance of "Individual" iff it does not have any possible instance* (Axiom A1 in Table 2). The constant "First-Order Type" (or shortly "1stOT") *characterizes the type that applies to all entities whose instances are instances of "Individual"* (A2 in Table 2). Analogously, each *entity whose possible extension contains exclusively instances of "1stOT" is an instance of "Second-Order Type"* (or shortly "2ndOT") (A3 in Table 2). It follows from axioms A1, A2 and A3 that "Individual" is instance of "1stOT" which, in turn, is instance of "2ndOT". We call "Individual", "1stOT" and "2ndOT" the basic types of MLT. According to MLT, every possible entity must be instance of exactly one of its basic types (except the topmost type) (A4 in Table 2). We consider here only first- and second-order types. However, this scheme can be extended to consider as many orders as necessary [15].

Table 2. MLT Axioms

A1	$\forall x\ \text{iof}(x, \text{Individual}) \leftrightarrow \nexists y\ \text{iof}(y, x)$
A2	$\forall t\ \text{iof}(t, \text{1stOT}) \leftrightarrow \big(\exists y\ \text{iof}(y, t) \land \big(\forall x\ \text{iof}(x, t) \rightarrow \text{iof}(x, \text{Individual})\big)\big)$
A3	$\forall t\ \text{iof}(t, \text{2ndOT}) \leftrightarrow (\exists y\ \text{iof}(y, \text{t}) \land (\forall t' \text{iof}(t', t) \rightarrow \text{iof}(t', \text{1stOT})))$
A4	$\forall x\ \big(\text{iof}(x, \text{Individual}) \lor \text{iof}(x, \text{1stOT}) \lor \text{iof}(x, \text{2ndOT})\big) \lor (x = \text{2ndOT})$
D1	$\forall \text{t1}, \text{t2}\ \text{specializes}(\text{t1}, \text{t2}) \leftrightarrow (\exists \text{y}\ \text{iof}(\text{y}, \text{t1}) \land \big(\forall \text{e}\ \text{iof}(\text{e}, \text{t1}) \rightarrow \text{iof}(\text{e}, \text{t2})\big))$
D2	$\forall\ \text{t1}, \text{t2}\ \ \text{properSpecializes}(\text{t1}, \text{t2}) \leftrightarrow (\text{specializes}(\text{t1}, \text{t2}) \land \text{t1} \neq \text{t2})$
D3	$\forall \text{t1}, \text{t2}\ \text{isSubordinateTo}\ (\text{t1}, \text{t2}) \leftrightarrow$ $(\exists \text{x}\ \text{iof}(\text{x}, \text{t1}) \land (\forall \text{t3}\ \text{iof}(\text{t3}, \text{t1}) \rightarrow (\exists \text{t4}\ \text{iof}(\text{t4}, \text{t2}) \land \text{properSpecializes}(\text{t3}, \text{t4}))))$
D4	$\forall \text{t1}, \text{t2}\ \text{isPowertypeOf}(\text{t1}, \text{t2}) \leftrightarrow (\exists \text{x}\ \text{iof}(\text{x}, \text{t1}) \land (\forall \text{t3}\ \text{iof}(\text{t3}, \text{t1}) \leftrightarrow \text{specializes}(\text{t3}, \text{t2})))$
D5	$\forall \text{t1}, \text{t2}\ \text{characterizes}(\text{t1}, \text{t2}) \leftrightarrow (\exists \text{x}\ \text{iof}(\text{x}, \text{t1}) \land (\forall \text{t3}\ \text{iof}(\text{t3}, \text{t1}) \rightarrow \text{properSpecializes}(\text{t3}, \text{t2})))$
D6	$\forall \text{t1}, \text{t2}\ \text{completelyCharacterizes}(\text{t1}, \text{t2}) \leftrightarrow$ $(\text{characterizes}(\text{t1}, \text{t2}) \land (\forall \text{e}\ \text{iof}(\text{e}, \text{t2}) \rightarrow \exists \text{t3}\ (\text{iof}(\text{e}, \text{t3}) \land \text{iof}(\text{t3}, \text{t1}))))$
D7	$\forall \text{t1}, \text{t2}\ \text{disjointlyCharacterizes}\ (\text{t1}, \text{t2}) \leftrightarrow$ $(\text{characterizes}(\text{t1}, \text{t2}) \land \forall \text{e}, \text{t3}, \text{t4}\ ((\text{iof}(\text{t3}, \text{t1}) \land \text{iof}(\text{t4}, \text{t1}) \land \text{iof}(\text{e}, \text{t3}) \land \text{iof}(\text{e}, \text{t4})) \rightarrow \text{t3} = \text{t4})))$
D8	$\forall \text{t1}, \text{t2}\ \text{partitions}(\text{t1}, \text{t2}) \leftrightarrow (\text{completelyCategorizes}(\text{t1}, \text{t2}) \land \text{disjointlyCategorizes}(\text{t1}, \text{t2}))$

Some structural relations to support conceptual modeling are defined in MLT, starting with the ordinary specialization between types. A type *t specializes* another type *t'* iff *all instances of t are also instances of t'* (see definition D1 in Table 2). Since the reflexivity of the *specialization* relation may be undesired in some contexts, we define in MLT the *proper specialization* relation as follows*: t proper specializes t' iff t specializes t' and t is different from t'* (see D2 in Table 2). Additionally, MLT defines a *subordination* relation. *Subordination* between two higher-order types implies *specializations* between their instances, i.e., *t is subordinate to t'* iff every *instance of t proper specializes* an *instance of t'* (see D3 in Table 2). The definitions presented thus far guarantee that both *specializations*, *proper specializations* and *subordinations* may hold exclusively between types of the same order. We term these *intra-level relations*.

MLT also defines relations that occur between types of adjacent orders, the so-called *cross-level structural relations*. These relations are inspired on different notions of powertype in the literature. Based on the notion of *powertype* proposed by Cardelli [19] (which is founded on the notion of powerset), MLT defines a *powertype* relation between a higher-order type and a base type at a lower order: a type *t is powertype of* a base type *t'* iff all instances of *t specialize t'* and all possible *specializations of t' are instances of t* (see D4). Note that it follows from the axioms and definitions presented so far that "1stOT" *is powertype of* "Individual", i.e. all possible instances of "1stOT" specialize "Individual" and all possible specializations of "Individual" are instances of "1stOT". Analogously, "2ndOT" *is powertype of* "1stOT", and so on. Thus, every instance of a basic higher-order type ("1stOT" and "2ndOT") must specialize the basic type at the immediately lower level (respectively, "Individual" and "1stOT"). In other words, the notion of orders or levels in MLT can be seen as a result of the iterated application of Cardelli's notion of powertype to the basic types.

Odell [18], in turn, defined *powertype* simply as a type whose instances are subtypes of another type (the *base type*), excluding the *base type* from the set of instances of the *powertype*. Inspired on Odell's definition for powertypes, MLT defines the *characterization* relation between types at adjacent levels: *a type t characterizes a type t' iff all instances of t are proper specializations of t'* (definition D5). The *characterization relation* occurs between a higher-order type t and a base type t' when *the instances of t specialize* t' according to a specific *classification criteria.* Thus, differently from the cases involving (Cardelli's) *is powertype of* relation, there may be specializations of the base type t' that are not instances of t. For example, we may define a type named "Organism by Habitat" (with instances "Terrestrial Organism" and "Aquatic Organism") that *characterizes* "Organism", but is not a *powertype of* "Organism" since there are specializations of "Organism" that are not instances of "Organism by Habitat" (e.g. "Plant" and "Golden Eagle").

MLT defines some refinements of the cross-level relation of characterization, which are useful to capture further constraints in multi-level models. We consider that *a type t completely characterizes t' iff t characterizes t' and every instance of t' is instance of, at least, an instance of t* (D6). Moreover, *iff t characterizes t' and every instance of t' is instance of, at most, one instance of t* it is said that *t disjointly characterizes t'* (D7). Finally, a common use for the notion of powertype in the literature considers a higher-order type that, simultaneously, completely and disjointly characterizes a lower-order type. To capture this notion MLT defines the *partitions* relation. Thus, *t partitions t' iff each instance of the base type t' is an instance of exactly one instance of t* (D8). For example, considering the biological taxonomy for living beings we have that "Species" (and all other biological ranks) *partitions* "Organism".

A complete formalization of MLT in first-order logic can be found in [15], which presents proofs for all MLT theorems. Further, a formal specification in Alloy is provided in [27] and was used to verify the theorems and to simulate admissible models of the theory using the Alloy analyzer.

5 Applying MLT for Multi-Level Modeling Support in OWL

Aiming to improve the OWL support for multi-level modeling, we propose (i) a vocabulary based on distinctions put forth by MLT, and (ii) a number of derivation and integrity rules reflecting axioms and theorems of MLT. The proposed vocabulary aims at providing modelers with an expressive set of constructs to support the production of multi-level ontologies in OWL. The integrity rules, in their turn, are used to verify if ontologies built using the proposed vocabulary are well-formed according to MLT rules. Finally, the derivation rules make use of MLT rules to infer information not represented explicitly by the modeler.

5.1 OWL Vocabulary based on MLT Distinctions

The proposed vocabulary encompasses the representation of the basic types of MLT and the relations defined in the theory. The basic types of MLT are represented as instances (*rdf:type*) of *owl:Class*. The class representing the MLT *Individual* basic type is named *mlt:TokenIndividual*[1], the class representing the *First-Order Type* is named *mlt:1stOrderClass*, and the classes *mlt:2ndOrderClass* and *mlt:3rdOrderClass* represent, respectively, the *Second-order* and *Third-order basic types.* Considering that, according to MLT, instances of *Individual* are not instantiable (i.e. are not types), *mlt:TokenIndividual* does not specialize *owl:Class*. In contrast, the classes representing all other basic types have a *rdf:subClassOf* relation with *owl:Class* capturing the fact that their instances are classes (i.e. their instances are instantiable) (see Fig. 3).

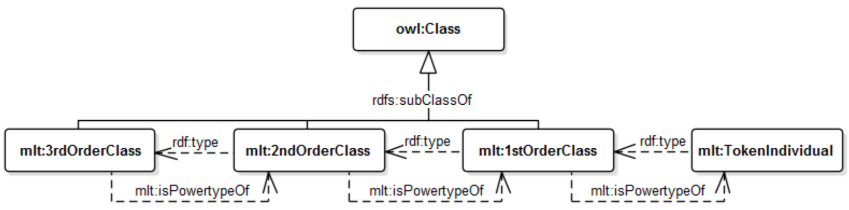

Fig. 3. Fragment of MLT Vocabulary for Order Classes and Individual.

[1] The term "TokenIndividual" was adopted here to avoid confusion with the term "Individual" in the OWL specification. "TokenIndividual" corresponds to what we call "Individual" in [15].

Concerning the MLT relations, *instance of* relations are represented as *rdf:type* properties and *specialization* relations are represented as *rdfs:subClassOf* properties. All other intra- and cross-level relations of MLT are represented in this vocabulary in a hierarchy of instances of *owl:ObjectProperty*, including at the top: *mlt:intraLevelProperty*, which is as a super-property for all MLT intra-level relations; and *mlt:crossLevelProperty*, which is a super-property for all MLT cross-level relations. The *subordination* relation of MLT is then represented by the property *mlt:isSubordinateTo* as a sub-property of *mlt:intraLevelProperty,* while the *characterization* (*mlt:characterizes)* and the *is power type of (mlt:isPowertypeOf)* relations are represented as sub-properties of *mlt:crossLevelProperty*. Finally, each variation of *characterization* (e.g. *complete characterization, disjoint characterization* and so on) is represented as a sub-property of *mlt:characterizes.*

These properties are also used in the vocabulary definition to represent relations that occur between the basic types of MLT. To capture the fact that the basic type in one *order* is instance of the basic type in an immediately higher order, statements with *rdf:type* are defined between the classes representing the basic types (e.g., *mlt:TokenIndividual rdf:type mlt:1stOrderClass, mlt:1stOrderClass rdf:type mlt:2ndOrderClass*). Further, *mlt:isPowertypeOf* is used to represent that a basic type in an order *is the powertype of* the basic type in the immediately-lower order (Fig. 3).

The MLT vocabulary allows the representation of domain rules concerning the instantiation of types in different levels. For example, Fig. 4 illustrates a fragment of an ontology in the biological taxonomy domain applying this vocabulary. In such an ontology, *Genus* and *Species* are represented as instances of *mlt:2ndOrderClass* (and, thus, as subclasses of *mlt:1stOrderClass*) meaning that their instances (e.g. *Panthera, Panthera Onca, and so on*) must specialize *mlt:TokenIndividual*, i.e. instances of their instances are non-instantiable elements (e.g. *Cecil*, the lion, which does not possibly have instances). The domain rule that every instance of *Species* must be a subclass of an instance of *Genus* is captured by the *mlt:isSubordinateTo* property between *Species* and *Genus.* Further, the *mlt:partitions* property between *Species* and *Panthera* captures the rule that every instance of *Panthera* must be instance of exactly one instance of *Species.* Finally, *Genus mlt:partitions Organism* and *Species mlt:partitions Organism*, to capture that every organism must be instance of exactly one *Genus* and instance of exactly one instance of *Species*. Note that domain modelers only need to declare their domain classes as instances and/or specializations of the MLT basic types. (As we shall discuss later in section 5.2, some of these relations can be inferred automatically, using derivation rules reflecting MLT axioms and theorems.)

Fig. 5 shows an example of an ontology representing employees and their roles in a company to illustrate the use of variations of *characterization* relations to capture domain rules. To capture the rule that each *Employee* must play one or more *Business Roles* in the company, *Business Role mlt:completelyCharacterizes Employee* meaning that every instance of *Employee* must be instance of at least one instance of *Business Role*. Further, to represent that an *Employee* may play at most one *Management Role, Management Role mlt:disjointlyCharacterizes Employee*.

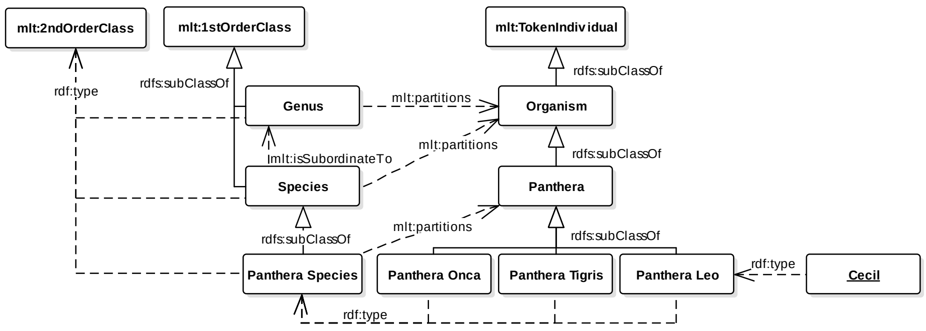

Fig. 4. Illustrating the use of *mlt:isSubordinateTo* and *mlt:partitions* properties.

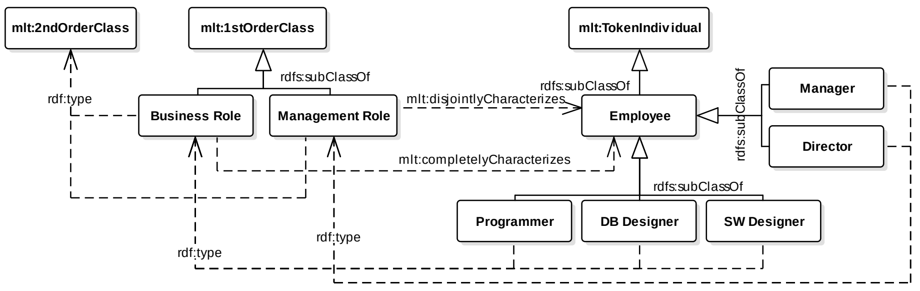

Fig. 5. Illustrating the use of *mlt:completelyCharacterizes* and *mlt:overlappinglyCharacterizes*.

5.2 Integrity constraints and derivation rules based on MLT

An important aspect of the proposed vocabulary is that it allows us to leverage rules of the MLT formalization in order to guide modelers in producing sound models. The rules discussed in this section ensure that the domain classes respect the stratification into orders.

Some of these rules are expressible in pure OWL and thus were directly included in the vocabulary. For example, a disjointness constraint (*owl:AllDisjointClasses*) is introduced to reflect the fact that the basic types of MLT are all mutually disjoint.

The majority of the MLT rules, though, are not expressible directly in OWL, and are represented here in SPARQL. This is the case of constraints concerning the domain and range of MLT structural relations. For example, *mlt:isPowertypeOf*, *mlt:characterizes* and all its variations must occur between classes of adjacent levels, i.e., if the domain is a *2ndOrderClass*, then the range must be a *1stOrderClass*, if the domain is a *3rdOrderClass*, then the range must be a *2ndOrderClass*, and so on. Table 3 shows the domain/range restrictions for MLT relations.

Table 3. Domain and range restrictions for multi-level relations.

Relation name	**Domain and Range**
rdfs:subClassOf	Classes of the same order (instances of 1st, 2nd or 3rd OrderClasses)
isSubordinateTo	Higher-order classes of the same order (2ndOrderClass or 3rdOrderClass)
rdf:type	Elements of adjacent levels.
isPowertypeOf	Classes of adjacent levels (2ndOrderClass →1stOrderClass or 3rdOrder-Class→2ndOrderClass)
characterizes	
completelyCharacterizes	
incompletelyCharacterizes	
disjointlyCharacterizes	
overlappinglyCharacterizes	

SPARQL queries are also provided to allow the verification of rules concerning the nature of the basic types of MLT. For example, considering that instances of Individual must have no instances, we provide an integrity constraint to verify if there are instances of instances of *mlt:TokenIndividual* (see Q1 in Fig. 6, which would detect violations of this constraint).

Integrity constraints are also provided to verify MLT theorems concerning characteristics of structural relations. For instance, given the definition of the *is powertype of* relation, a base class can have, at most, one higher-order class as powertype and a higher-order class may be the powertype of at most one base class. This suggests two clear integrity constraints: (i) a class can be the subject of at most one triple having *mlt:isPowertypeOf* as predicate (violations detected by Q2 in Fig. 6), and (ii) a class can be the object of at most one triple having *mlt:isPowertypeOf* as predicate. Another example is a constraint provided to allow the verification of the MLT theorem that states that if two classes *t1* and *t2* both partition the same class *t* then it is not possible for *t1* to be subclass of *t2* (Q3 in Fig. 6).

Q1
```
select distinct * where{
        ?x rdf:type mlt:TokenIndividual .
        ?y rdf:type ?x .
}
```

Q2
```
select distinct * where{
        ?p mlt:isPowertypeOf ?t .
        ?p mlt:isPowertypeOf ?t1 .
        FILTER (?t != ?t1) .
}
```

Q3
```
select distinct * where{
        ?t1 mlt:partitions ?t .
        ?t2 mlt:partitions ?t .
        ?t1 rdfs:subClassOf ?t2 .
}
```

Q4
```
select distinct * where{
        ?t2 rdfs:subClassOf+ ?t1 .
        ?t4 mlt:isPowerTypeOf ?t2 .
        ?t3 mlt:isPowerTypeOf ?t1 .
        minus{ ?p rdfs:subClassOf ?p1 . }
}
```

Q5
```
select distinct * where{
        ?t2 mlt:isPowerTypeOf ?t1 .
        ?t3 mlt:characterizes ?t1 .
        minus{ ?t3 rdfs:subClassOf ?t2 . }
}
```

Q6
```
select distinct * where{
 ?t rdf:type mlt:1stOrderClass .
 minus{
  ?t rdfs:subClassOf mlt:TokenIndividual.
} }
```

Fig. 6. SPARQL queries representing MLT rules

Considering that models built using our MLT vocabulary may exhibit incomplete information, we leverage MLT axioms and theorems to allow the inference of information not represented explicitly. For example, it follows from the axioms of MLT that, if *t* is subclass of *t1* then the *powertype of t* is subclass of the *powertype of t1*. This is reflected in a query to identify cases in which the *subclass of* relation is not represented between the power types (Q4). Since, according to MLT, if *t2* is powertype of *t1* and *t3* characterizes *t1*, then *t3* is subclass of *t2*, we provide a SPARQL query to identify cases in which the *powertypeOf* and the *characterization* relations are represented but the *subclass* relations are not (Q5 in Fig. 6). Further, since every instance of a basic higher-order type must specialize the basic type at the immediately lower level, we can identify some missing relations. For example, query Q6 in Fig. 6 allows the identification of cases in which types are represented as instances of *mlt:1stOrderClass* but their subclass relations with *mlt:TokenIndividual* are not represented.

Since MLT is formalized quantifying over *all possible entities*, some MLT definitions are not expressible considering the Open World Assumption (OWA). For instance, according to MLT if *t1* has instances such that all of them are also instances of *t2*, then we can conclude that *t1* is a subclass of *t2* (D1 in Table 2). This rule could not be captured in our approach since, considering the OWA, we cannot assume that all instances of an entity are represented in the knowledge base. Thus, these rules cannot be reflected in the implementation.

Finally, it is worth mentioning that, due to space limitations, we only expose here some rules to illustrate the approach. The vocabulary and the complete set of SPARQL queries is available at [27], including information on the traceability between MLT axioms and theorems and the implemented queries.

5.3 Final Considerations

Multi-level modeling addresses phenomena dealing with a number of complex notions and subtle relations that cross multiple levels of instantiation. These phenomena are ubiquitous in application domains, ranging from biology, to software engineering, from enterprise modeling to product classification [15]. Aside from the recurrence of these phenomena in practical cases, what also makes it of great importance is the fact that multi-level modeling seems to pose a significant challenge to modelers. As previously mentioned, in [14], we have empirically analyzed the presence of three anti-patterns related to multi-level modeling in Wikidata, finding over 22,000 occurrences of these anti-patterns. In fact, for one these anti-patterns, we found its manifestation in 85% of the cases of taxonomic hierarchies spanning more than one level in Wikidata! That study clearly indicates that for complex modeling phenomena such as these, an expressive engineering support must be offered for vocabulary engineers as well as semantic web application developers. In [27], we provide a technical report showing how each of these anti-patterns found in Wikidata could be avoided by using the artifact proposed in this paper, demonstrating the relevance of MLT-OWL using real-world data.

The recognition of the importance of offering support for multi-level modeling led many researchers in the Semantic Web community to propose solutions addressing this issue. Some prominent results in that respect are reviewed in this paper, namely, RDFS(FA), metamodeling (punning) in OWL 2, OWL FA and PURO. We have shown in our analysis of these related works that all of them fail to fully support the identified modeling desiderata.

We adopted as a basis for our work a theoretically sound and well-tested formal theory (MLT) that was shown to be able to address all these multi-level modeling requirements. We then decided to offer a set of engineering tools that together would implement the modeling distinctions and axiomatization of this theory. These tools include: (i) an OWL vocabulary (capturing the formal relations put forth by this theory); (ii) a set of OWL axioms that would capture derivation and integrity rules over this vocabulary put forth by the theory; and (iii) a set of SPARQL queries that would capture those derivation and integrity rules put forth by this theory but that could not be represented in OWL directly. We strongly believe that these tools amount to an important methodological and computational contribution for guiding modelers to produce sound multi-level models in the Semantic Web.

The reason why these phenomena are recurrent in a large variety of practical application domains is because they are genuine ontological phenomena (from a philosophical point of view) [26]. As such, we advocate that truly ontological considerations cannot be eschewed from a fuller analysis of multi-level modeling. Additionally, some initiatives have demonstrated that the systematic evaluation of the ontological consistency of Semantic Web ontologies and vocabularies can greatly benefit from the use of foundational distinctions and axioms ([28], [29]). In order to leverage the benefits of both a foundational ontology and a multi-level modeling theory, in [30] some of us have already combined MLT and the foundational ontology UFO [31]. A natural extension of this work is to enrich the set of engineering tools proposed here with support for the ontological distinctions and axiomatization of UFO (e.g., dealing with temporal aspects of anti-rigid concepts).

Acknowledgements. This research is funded by W3C Brasil and CNPq (grants 311313/2014-0, 485368/2013-7, 312158/2015-7 and 461777/2014-2). Freddy Brasileiro is funded by FAPES and Victorio A. Carvalho is funded by CAPES.

References

1. W3C: W3C Semantic Web Activity, https://www.w3.org/2001/sw/.
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. Int. J. Semant. Web Inf. Syst. 5, 1–22 (2009).
3. W3C: RDF Schema 1.1, (2014).
4. W3C: OWL 2 Web Ontology Language - Document Overview (Second Edition), (2012).
5. Mayr, E.: The Growth of Biological Thought: Diversity, Evolution, and Inheritance. Harvard University Press (1982).
6. Gonzalez-Perez, C., Henderson-Sellers, B.: A powertype-based metamodelling framework. Softw. Syst. Model. 5, 72–90 (2006).
7. Neumayr, B., Grun, K., Schrefl, M.: Multi-level domain modeling with m-objects and m-relationships. In: 6th Asia-Pacific Conference on Conceptual Modelling (2009).
8. Atkinson, C., Kühne, T.: The Essence of Multilevel Metamodeling. In: 4th International Conf. on the Unified Modeling Language (2001).
9. Pan, J.Z., Horrocks, I.: Metamodeling Architecture of Web Ontology Languages. In: Proc. of the 2001 Int. Semantic Web Working Symposium. pp. 131–149 (2001).
10. Motik, B.: On the Properties of Metamodeling in OWL. J. Log. Comput. 17, 617–637 (2007).

11. Pan, J.Z., Horrocks, I., Schreiber, G.: OWL FA: A metamodeling extension of OWL DL. In: Proc. of the 15th Intl Conf. on World Wide Web. pp. 1065–1066. ACM (2006).
12. Svatek, V., Homola, M., Kluka, J., Vacura, M.: Metamodeling-Based Coherence Checking of OWL Vocabulary Background Models. In: Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED 2013) (2013).
13. W3C: OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (Second Edition), (2012).
14. Brasileiro, F., Almeida, J.P.A., Carvalho, V.A., Guizzardi, G.: Applying a Multi-Level Modeling Theory to Assess Taxonomic Hierarchies in Wikidata. In: Wiki Workshop 2016 at 25th Int. Conference Companion on World Wide Web. pp. 975–980 (2016).
15. Carvalho, V.A., Almeida, J.P.A.: Toward a well-founded theory for multi-level conceptual modeling. Softw. Syst. Model. (2016).
16. Atkinson, C., Kühne, T.: Model-Driven Development: A Metamodeling Foundation. IEEE Softw. 20, 36 – 41 (2003).
17. 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).
18. Odell, J.: Power Types. J. Object-Oriented Programing. 7, 8–12 (1994).
19. Cardelli, L.: Structural subtyping and the notion of power type. In: Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages - POPL '88. pp. 70–79. ACM Press, New York, New York, USA (1988).
20. W3C: Resource Description Framework (RDF) Schema Specification 1.0, (2000).
21. OMG: Unified Modeling Language Specification v1.3, (1999).
22. Jekjantuk, N., Gröner, G., Pan, J.Z.: Modelling and Reasoning in Metamodelling Enabled Ontologies. Int. J. Softw. Informatics. 4, 277–290 (2010).
23. Gröner, G., Jekjantuk, N., Walter, T., Parreiras, F.S., Pan, J.Z.: Metamodelling and Ontologies. In: Ontology-Driven Software Development. pp. 151–174 (2013).
24. 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 Advandced Information Systems Engineering (2016).
25. Carvalho, V.A., Almeida, J.P.A.: A Semantic Foundation for Organizational Structures: A Multi-level Approach. In: 19th IEEE Intl Enterprise Distributed Object Computing Conference (EDOC 2015). pp. 50–59 (2015).
26. 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: 34rd International Conference on Conceptual Modeling (ER2015) (2015).
27. MLT, http://nemo.inf.ufes.br/mlt.
28. Fernandéz-López, M., Gómez-Pérez, A.: The integration of OntoClean in WebODE. In: EKAW'02 Workshop on Evaluation of Ontology-based Tools (EON2002). pp. 38–52 (2002).
29. Paulheim, H., Gangemi, A.: Serving DBpedia with DOLCE – More than Just Adding a Cherry on Top. In: The Semantic Web - ISWC 2015. pp. 180–196 (2015).
30. Guizzardi, G., Almeida, J.P.A., Guarino, N., Carvalho, V.A.: Towards an Ontological Analysis of Powertypes. In: Intl. Workshop on Formal Ontologies for Artificial Intelligence (FOFAI 2015). , Buenos Aires (2015).
31. Guizzardi, G.: Ontological foundations for structural conceptual models. Telematica Instituut Fundamental Research Series, Enschede (2005).