

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

FREDDY BRASILEIRO SILVA

REPRESENTATION OF MULTI-LEVEL DOMAINS ON THE WEB

VITÓRIA

2016

FREDDY BRASILEIRO SILVA

REPRESENTATION OF MULTI-LEVEL DOMAINS ON THE WEB

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Orientador: Prof. Dr. João Paulo A. Almeida

VITÓRIA

2016

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Setorial Tecnológica,
Universidade Federal do Espírito Santo, ES, Brasil)

B823r Brasileiro, Freddy, 1987-
Representation of multi-level domains on the web / Freddy
Brasileiro. – 2016.
90 f. : il.

Orientador: João Paulo Andrade Almeida.
Dissertação (Mestrado em Informática) – Universidade
Federal do Espírito Santo, Centro Tecnológico.

1. Web semântica. 2. Metamodelagem. 3. OWL (Web
Ontology Language). 4. Modelagem multi-nível. I. Almeida, João
Paulo Andrade. II. Universidade Federal do Espírito Santo.
Centro Tecnológico. III. Título.

CDU: 004

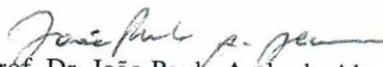


“Representation of Multi-Level Domains on the Web”

Freddy Brasileiro Silva

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Informática.

Aprovada em 28 de setembro de 2016 por:


Prof. Dr. João Paulo Andrade Almeida - PPGI/UFES


Prof. Dr. Vitor Estevão Silva Souza - PPGI/UFES


Prof. Dr. Fernando Silva Parreiras - FUMEC

O julgamento dessa dissertação foi realizado com a participação por meio de videoconferência do **membro externo** o Prof. Dr. Fernando Silva Parreiras seguindo as normas prescritas na portaria normativa no. 1/2016. Desse modo, a assinatura do membro externo é representada neste documento pela respectiva assinatura do presidente da comissão julgadora, o Prof. Dr. João Paulo Andrade Almeida.

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
Vitória-ES, 28 de setembro de 2016.

Confere com original

Aline Oliveira Amaral

Aline Oliveira Amaral
Secretária
Programa de Pós-Graduação em Engenharia Elétrica
Mestrado/Doutorado - CT/UFES
Sisape 1163007

AGRADECIMENTOS

Primeiramente, gostaria de agradecer a minha família por todo suporte dado até hoje. Sem eles, eu não teria chegado até aqui. Minha mãe Lana, por todo carinho e confiança. Minha vó Ana, a quem sou muito grato. Meus queridos irmãos, Thiago e Fábia. Meu tio Lindberg Júnior e minha prima Fernanda, por participação importante no meu desenvolvimento pessoal e intelectual. Aos meus queridos, tios Normindo e Dalva.

A minha amada Larissa, minha namorada, por estar sempre por perto, me dando carinho e suporte em todos os momentos. Muito obrigado por ter me motivado e me ajudado a ter força para concluir esse trabalho, principalmente na reta final. Eu te amo.

Ao meu orientador João Paulo, por compartilhar seu conhecimento e experiência, contribuindo muito para a minha evolução. Um agradecimento especial ao Victorio A. Carvalho, que apesar de não poder ser oficialmente membro da banca, teve participação especial, dando suporte e motivação durante todo o desenvolvimento deste trabalho. Ao professor Giancarlo Guizzardi pelas excelentes contribuições dadas a este trabalho.

Aos meus companheiros de Menthor: Bernardo, John, Luiz Olavo e Tiago. Aos meus companheiros de Nemo: Archimedes, Cássio, César, Fabiano, Filipe, Gabriel, Jordana, Julio, Laylla, Lucas, Pedro Paulo, Victor Amorim, Victorio, Vinicius Sobral, etc. Aos professores do Nemo: Giancarlo, João Paulo, Monalessa, Renata, Ricardo e Vítor Souza. Aos colegas de projetos Padtec: Rodrigo Stange e professores Anilton e Maxwell.

Esse trabalho foi financiado pela FAPES (Fundação de Amparo à Pesquisa e Inovação do Espírito Santo) (283/2014), e foi desenvolvido no escopo de projetos de fomento da própria FAPES, da CAPES, do W3C Brasil e do CNPq.

RESUMO

Estratégias de modelagem conceitual e representação de conhecimento frequentemente tratam entidades em dois níveis: um nível de classes e um nível de indivíduos que instanciam essas classes. Em vários domínios, porém, as próprias classes podem estar sujeitas a categorização, resultando em classes de classes (ou metaclasses). Ao representar estes domínios, é preciso capturar não apenas as entidades de diferentes níveis de classificação, mas também as suas relações (possivelmente complexas). No domínio de taxonomias biológicas, por exemplo, um dado *organismo* (por exemplo, o leão *Cecil* morto em 2015 no Parque Nacional Hwange no Zimbábue) é classificado em diversos táxons (como, por exemplo, *Animal*, *Mamífero*, *Carnívoro*, *Leão*), e cada um desses táxons é classificado por um ranking taxonômico (por exemplo, *Reino*, *Classe*, *Ordem*, *Espécie*). Assim, para representar o conhecimento referente a esse domínio, é necessário representar entidades em níveis diferentes de classificação. Por exemplo, *Cecil* é uma instância de *Leão*, que é uma instância de *Espécie*. *Espécie*, por sua vez, é uma instância de *Ranking Taxonômico*. Além disso, quando representamos esses domínios, é preciso capturar não somente as entidades diferentes níveis de classificação, mas também suas (possivelmente complicadas) relações. Por exemplo, nós gostaríamos de afirmar que instâncias do gênero *Panthera* também devem ser instâncias de exatamente uma instância de *Espécie* (por exemplo, *Leão*). A necessidade de suporte à representação de domínios que lidam com múltiplos níveis de classificação deu origem a uma área de investigação chamada *modelagem multi-nível*. Observa-se que a representação de modelos com múltiplos níveis é um desafio em linguagens atuais da Web Semântica, como há pouco apoio para orientar o modelador na produção correta de ontologias multi-nível, especialmente por causa das nuances de restrições que se aplicam a entidades de diferentes níveis de classificação e suas relações. A fim de lidar com esses desafios de representação, definimos um vocabulário que pode ser usado como base para a definição de ontologias multi-nível em OWL, juntamente com restrições de integridade e regras de derivação. É oferecida uma ferramenta que recebe como entrada um modelo de domínio, verifica conformidade com as restrições de integridade propostas e produz como saída um modelo enriquecido com informações derivadas. Neste processo, é empregada uma teoria

axiomática chamada MLT (uma Teoria de Modelagem Multi-Nível). O conteúdo da plataforma Wikidata foi utilizado para demonstrar que o vocabulário poderia evitar inconsistências na representação multi-nível em um cenário real.

ABSTRACT

Often, subject domains are conceptualized with entities in two levels: a level of classes, and a level of individuals which instantiate these classes. In several subject domains, however, classes themselves may be subject to categorization, resulting in classes of classes (or metaclasses). To represent these domains, one needs to capture not only entities of different classification levels, but also their (possibly intricate) relations. In the domain of biological taxonomies, for instance, a given *organism* (e.g. *Cecil*, the lion killed in the Hwange National Park in Zimbabwe in 2015) 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* is an instance of *Lion*, since he exhibits those common features. For example, *Cecil* is an instance of *Lion*, which is an instance of *Species*. *Species*, in its turn, is an instance of *Taxonomic Rank*. Moreover, when representing these domains, one needs to capture not only entities of different classification levels, but also their (possibly intricate) relations. For example, we would like to state that instances of the genus *Panthera* must also be instances of exactly one instance of *Species* (e.g. *Lion*). 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*. We observe that the representation of multi-level domains 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 define a vocabulary that can be used as basis for the definition of multi-level ontologies in OWL. This vocabulary is accompanied by integrity constraints to prevent the construction of inconsistent models as well as derivation rules to derive knowledge that is not explicit in the model. We offer a tool that receives as input a domain model, checks its conformance with the proposed integrity constraints and produces an output model containing the original domain model plus derived information. In this process, we employ an axiomatic theory called MLT (a Multi-Level

Modeling Theory). We use Wikidata content to demonstrate that the approach can prevent the construction of inconsistent multi-level representations in a realistic setting.

LIST OF FIGURES

Figure 2-1. The UML notation for the powertype pattern	21
Figure 3-1. Labeled graph representing a triple about Tim Berners-Lee	31
Figure 3-2. Fragment of RDFS vocabulary	31
Figure 3-3. Labeled graph representing triples about Tim Berners-Lee, London, Person and City	32
Figure 3-4. Example of Directed Labeled Graph of RDFS(FA) (from [9])	34
Figure 3-5. Fragment of OWL vocabulary	36
Figure 3-6. OWL representation for biological taxonomic domain	36
Figure 3-7. Short representation for Taxonomic Biological Domain in Wikidata.....	41
Figure 4-1. Fragment of MLT Vocabulary for Order Classes and Individual.....	46
Figure 4-2. Illustrating the use of <code>mlt:isSubordinateTo</code> and <code>mlt:partitions</code> properties.	47
Figure 4-3. Illustrating the use of <code>mlt:completelyCategorizes</code> and <code>mlt:overlappinglyCategorizes</code>	48
Figure 4-4. Tool's flowchart	58
Figure 5-1. Wikidata information about Tim Berners-Lee and his professional occupation	62
Figure 5-2. Illustration of Anti-Pattern 1	64
Figure 5-3. Scenario about earthquake found in Wikidata for AP1	64
Figure 5-4. Scenario about egg waffle found in Wikidata for AP1	65
Figure 5-5. Illustration of Anti-Pattern 2.....	65
Figure 5-6. Scenario about excavator found in Wikidata for AP2	66
Figure 5-7. Illustration of Anti-Pattern 3.....	66
Figure 5-8. Scenario about urban park found in Wikidata for AP3.....	67
Figure 5-9. Derivation messages for the example of Tim Berners-Lee profession from Wikidata.....	70
Figure 5-10. Inconsistency messages for the example of Tim Berners-Lee profession from Wikidata.....	71

Figure 5-11. Derivation messages for the example of earthquakes from Wikidata	72
Figure 5-12. Derivation messages for the example of excavators from Wikidata	73
Figure 5-13. Derivation messages for the example of urban parks from Wikidata	73
Figure 5-14. Inconsistency messages for the example of urban parks from Wikidata	74
Figure 5-15. Derivation messages for the example of Biological Taxonomy with MLT relations	75

LIST OF TABLES

Table 2-1. MLT Rules	24
Table 3-1. Support for multi-level modeling in RDFS languages	42
Table 4-1. Domain and range restrictions for multi-level relations.	49
Table 4-2. Integrity Constraints Corresponding to MLT Rules	50
Table 4-3. Derivation Rules Corresponding to MLT Rules.....	53
Table 5-1. Results for AP1 and AP2	67
Table 5-2. Results for AP3	68
Table II-1. SPARQL query for Wikidata Anti-Patterns	87

ABBREVIATIONS AND ACRONYMS

API	Application Programming Interface
IRI	Internationalized Resource Identifier
MLT	Multi-Level Modeling Theory
OCA	Orthogonal Classification Architecture
OWL	Web Ontology Language
OWL FA	OWL with Fixed metamodeling Architecture
RDF	Resource Description Framework
RDFS	RDF Schema
RDFS(FA)	RDFS with Fixed metamodeling Architecture
UFO	Unified Foundation Ontology
UML	Unified Modeling Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium

SUMMARY

1	INTRODUCTION	15
1.1	MOTIVATION	15
1.2	OBJECTIVES	17
1.3	APPROACH	17
1.4	STRUCTURE	19
2	MULTI-LEVEL MODELING	20
2.1	POWERTYPE PATTERN	20
2.2	CLABJECTS AND DEEP INSTANTIATION.....	21
2.3	MLT: A THEORY FOR MULTI-LEVEL MODELING.....	22
2.4	REQUIREMENTS FOR A MULTI-LEVEL APPROACH.....	27
3	RELATED WORK: CURRENT APPROACHES TO MULTI-LEVEL MODELING IN THE SEMANTIC WEB.....	30
3.1	RDFS	30
3.2	RDFS(FA)	33
3.3	OWL 2	35
3.4	OWL FA.....	37
3.5	PURO	38
3.6	THE WIKIDATA APPROACH	39
3.7	CONCLUDING REMARKS	41
4	EXPRESSIVE MULTI-LEVEL MODELING FOR THE SEMANTIC WEB	44
4.1	PRELIMINARY CONSIDERATIONS.....	44
4.2	THE MLT VOCABULARY FOR THE SEMANTIC WEB.....	45
4.3	INTEGRITY CONSTRAINTS BASED ON MLT.....	48
4.4	MODEL COMPLETION BASED ON MLT.....	51
4.5	IMPLEMENTATION.....	57

4.6	CONCLUDING REMARKS.....	58
5	VALIDATING THE APPROACH WITH TAXONOMIC HIERARCHIES FROM WIKIDATA	61
5.1	A DIAGNOSIS OF MULTI-LEVEL TAXONOMIC HIERARCHIES IN WIKIDATA.....	61
5.2	APPROACH: DETECTION OF ANTI-PATTERNS	63
5.3	APPLYING MLT RULES TO WIKIDATA CONTENT.....	68
5.3.1	Examples on Avoiding Anti-Patterns	69
5.3.2	Example on Deriving Data from Valid Models.....	74
5.4	CONCLUDING REMARKS.....	75
6	CONCLUSIONS.....	77
6.1	CONTRIBUTIONS	77
6.2	LIMITATIONS	78
6.3	FUTURE WORK.....	79
	BIBLIOGRAPHY	81
	APPENDIX I - THE MLT VOCABULARY FOR THE SEMANTIC WEB.....	85
	APPENDIX II - SPARQL QUERIES FOR WIKIDATA ANTI-PATTERNS	87
	APPENDIX III - OWL FILES FOR EMPIRICAL EVALUATION	88
	TIM BERNERS-LEE PROFESSION.....	88
	EARTHQUAKES.....	88
	EXCAVATOR	89
	URBAN PARK.....	89
	BIOLOGICAL TAXONOMY.....	90

1 INTRODUCTION

1.1 MOTIVATION

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 (Resource Description Framework) languages, which provide a basis for producing reusable vocabularies for various domains of interest [2].

The notion of *class* is a key concept for representing vocabularies in Semantic Web approaches, such as RDF Schema (RDFS) [3] and Web Ontology Language (OWL) [4]. 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. For example, a class *Lion* can capture common features of a specific set of animals that are felines and that have a specific set of morphological characteristics (such as weight, height, length, tail size, longevity). Then, *Cecil* (the lion killed in the Hwange National Park in Zimbabwe in 2015) is an instance of *Lion*, since it exhibits those common features.

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* 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]–[13]). These approaches exploit the fact that a class is itself an RDF resource and may thus be the subject or object of RDF triples. OWL 2 explicitly adopts this strategy under the term “metamodeling”, enabling the representation of facts that are stated about classes [4].

Despite these developments, the current support for the representation of *domains dealing with multiple levels of classification* in the Semantic 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 *Species*. Representing these relations between entities at different levels of classification is not possible in current approaches.

Finally, knowledge bases, such as Wikidata [15] and DBPedia [16], have a lot of taxonomic hierarchies with entities at different classification levels (particular individuals, types of individuals, types of types of individuals, etc.). Aside from the recurrence of domains that deal with multiple levels of classification, what also makes it of great importance is the fact that multi-level modeling seems to pose a significant challenge to modelers. For instance, analyzing Wikidata, we identified a significant number of problematic classification and taxonomic statements especially when involving multiple levels of classification and instantiation [14]. We hypothesize this can be explained by the lack of adequate support for the representation of multi-level domains in the Semantic Web. Thus, in order to allow the proper representation of multi-level domains in the Semantic Web, this thesis puts forward a vocabulary, guidelines and tools for representing multi-level domains in the Semantic Web.

1.2 OBJECTIVES

The overall aim of this research is to provide mechanisms to support the representation of expressive multi-level models in the Semantic Web. As specific objectives of this research, we aim at:

1. Defining requirements for the representation of multi-level domains, emphasizing the expressiveness and quality of the resulting multi-level models;
2. Reviewing the current Semantic Web approaches concerning their adherence to these requirements;
3. Providing an approach to support the expressive representation of multi-level domains on the web;
4. Providing guidelines and tools for modelers, in order to ensure the soundness of multi-level models;
5. Evaluate items 3 and 4 in a real knowledge base.

1.3 APPROACH

In order to meet the overall aim of this research, we investigate both multi-level modeling approaches in general (such as the *powertype pattern* [17], [18] and *clabjects* [19]), and those specifically targeted for the Semantic Web. In this investigation, we identify a number of common features of multi-level modeling approaches, as well as a number of recurrent limitations. This forms the basis for us to formulate requirements for a desirable multi-level modeling approach (specific objective 1).

We evaluate a number of initiatives that have been conducted to support multi-level modeling in the Semantic Web, including the native support of RDFS [3] and OWL 2 [4] standards proposed by W3C (World Wide Web Consortium), and further approaches that aimed at improving them, namely: RDFS(FA) [9] and OWL FA [11]. We further evaluate independent approaches such as PURO [12] and the representation strategy underlying Wikidata [15]. In each case, we point out which requirements are supported by each approach, also discussing their limitations (specific objective 2). We are concerned mainly

with their support for the representation of domain metaclasses as opposed to language metaclasses.

We base our approach on a well-founded multi-level theory that captures the foundational concepts underlying multi-level modeling, a multi-level modeling theory called MLT [20]. The choice for MLT as semantic foundations for this work is due to the fact it aims to achieve the requirements we deem desirable for an expressible multi-level modeling approach. This theory formally characterizes the nature of classification levels, and precisely defines the relations that may occur between elements of different classification levels, which aims to fill the gap of existent approaches. The proposed representation maintains correspondence with MLT as a foundational theory, in line with the reasoning defended in [21], [22], that a modeling representation ought to provide primitives that reflect categories in a foundational theory.

We propose an OWL vocabulary reflecting MLT distinctions (MLT-OWL), that can be used as a basis for multi-level ontologies (specific objective 3). Axioms and theorems of MLT are incorporated into integrity constraints which are applied to multi-level vocabularies that employ MLT-OWL, offering thus guidance to prevent the construction of inconsistent vocabularies. MLT rules are further employed to perform the derivation of knowledge about the relations between elements that are not explicitly stated (specific objective 4).

In the proposed solution, we opt to conform to the existing metamodeling structure that is imposed by RDFS/OWL. Further, since the expressiveness of OWL does not allow us to represent all applicable multi-level modeling rules, we adopt a complementary strategy: we represent some of the MLT rules in native OWL when possible, and complement those with SPARQL [23] queries when necessary.

Finally, to illustrate the applicability of the approach in a realistic setting, we evaluate it using real-world data obtained from Wikidata content (specific objective 5). We analyze occurrences of violations of MLT integrity constraints and show how our approach could provide guidance to populate knowledge bases avoiding the identified problems. Our approach is capable of warning users, showing which statements are problematic in light of MLT.

1.4 STRUCTURE

This thesis is further structured as follows: Chapter 2 presents a brief introduction to multi-level modeling and discusses the multi-level modeling theory employed in this work (MLT) along with the basic requirements for the representation of multi-level domains; Chapter 3 reviews the current support for multi-level modeling in the Semantic Web as well as in related work in the literature; Chapter 4 presents our approach to represent multi-level domains in the Semantic Web reflecting the rules of MLT, including the OWL vocabulary and integrity and derivations rules implemented in SPARQL; Chapter 5 presents results of our analysis of current Wikidata content; Chapter 6 presents concluding remarks.

2 MULTI-LEVEL MODELING

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]. The interest in multi-level modeling has led to a number of research initiatives in this subject (e.g. [6]–[8], [17], [18]). We present here first two influential approaches for multi-level modeling research: the *powertype pattern* (discussed in Section 2.1) and the approaches based on the notion of *clabject* (discussed in Section 2.2). These approaches have influenced the MLT axiomatic theory we employ further in this work (described in Section 2.3). Since there is yet no consensus on the requirements that multi-level modeling approaches must satisfy [24], we define desirable requirements for our own multi-level modeling approach (Section 2.4) based on the common features and limitations of current approaches.

2.1 POWERTYPE PATTERN

A seminal approach to the representation of domains with multiple levels of classification relies on the notion of *powertype*. This approach raised from the identification of patterns to represent the relationship between a class of categories (the *powertype*) and a class of more concrete entities (the *base type*) [20]. Odell [17] defines a *powertype* as a type whose instances are subtypes of another type. For example, if we consider *Sugar Maple*, *Apricot*, *American Elm* and *Saguaro* as subtypes of *Tree* and as instances of *Tree Species*, then *Tree Species* is a powertype of *Tree* according to Odell’s definition.

Cardelli [18], in its turn, defines *powertype* in an analogy with the notion of powerset. Observing that the *powerset* of a set A , is the set whose elements are all possible subsets of A including A itself, Cardelli defines that the *powertype* of T is a type whose instances are all possible specializations of T , including T itself. For example, we can define a type *Tree Powertype* having as instances all possible specializations of *Tree*, including all instances of *Tree Species* and *Tree* itself as well as all other specializations of tree with any criteria such as *Big Tree*, *Small Tree*, *Tree Seedling*, *Old Tree*, *Blooming Tree*. Note that, differently from

Cardelli, Odell admits the existence of specializations of the *base type* that are not instances of the *powertype*.

Following Odell's notion of *powertype*, the UML (Unified Modeling Language) 2.4.1 specification [25] includes means to relate a *powertype* to a generalization set, meaning that all the subtypes involved in such generalization set are instances of the related *powertype*. In Figure 2-1, the generalization set is composed by the specializations of *Tree* (namely *Sugar Maple*, *Apricot*, *American Elm* and *Saguaro*), and the text “:Tree Species” means that these subtypes are instances of *Tree Species*. In this case, the *disjoint* constraint means that *Tree* specializations have no instances in common, and the *incomplete* constraint means that there are instances of *Tree* that are not instances of *Sugar Maple*, *Apricot*, *American Elm* and *Saguaro*.

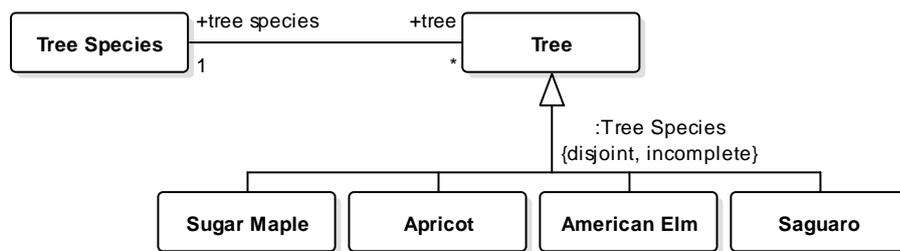


Figure 2-1. The UML notation for the powertype pattern

2.2 CLABJECTS AND DEEP INSTANTIATION

In [19], Atkinson and Kühne argue that to model instantiation between arbitrary adjacent levels, it is necessary to follow two fundamental properties of multi-level modeling: strict metamodeling [26] and adhering to the notion of *clabjects*. In the strict metamodeling principle, every element of an M_n level model is an instance of exactly one element of an M_{n+1} level model (except for the top level).

The authors coined the term *clabject* to refer to elements that have two facets: a type (or class) facet and an instance (or object) facet. For example, we may consider that *iPhone 5* has both an instance facet (it is instance of *Mobile Phone Model*) and a type facet (some mobile phones, such as *my iPhone*, are instances of *iPhone 5*). The type facet of *iPhone 5* captures some regularities that its instances have (e.g. every *iPhone 5* has a 4-inch screen

and each one has a particular *imei* number) while the instance facet defines values for some attributes captured by *Mobile Phone Model* (e.g. considering that each *Mobile Phone Model* has a *launch date*, *iPhone 5* is an instance of *Mobile Phone Model* and has *launch date* equals to “21 Sept, 2012”).

Atkinson and Kühne also proposed in [8], [27] the notion of *deep instantiation* as a means to provide multiple levels of classification whereby an element at some level can describe features of elements at each level beneath that level. They introduce the notion of *potency* that is assigned to every model element at every model level. Potency defines the length of the instantiation chain that is allowed below the element, in such way that an element of potency 0 corresponds to a concrete individual and cannot be instantiated (i.e., an element of potency 0 is not a *clabject*). When a *clabject* instantiates another *clabject*, the potency of the *clabject* being instantiated is given by the potency of the *clabject* being instantiated minus one. For example, considering the cited example of mobile phone model, *Mobile Phone Model*, *iPhone 5* and *myiPhone5* have potencies 2, 1 and 0, respectively. Finally, in contrast with the powertype pattern, deep instantiation does not obligate the modeler to represent the base type. This design decision has the general purpose of reducing the number of modeled concepts. Even when the base type is represented, the approach does not support the representation of the relation between the base type and the higher-order type. In the aforementioned example, we would be unable to express that each instance of *Mobile Phone* must also be an instance of exactly one instance of *Mobile Phone Model* (e.g. *iPhone 5*) (that specializes *Mobile Phone*).

2.3 MLT: A THEORY FOR MULTI-LEVEL MODELING

Motivated by the lack of theoretical foundations for multi-level modeling, Carvalho and Almeida have proposed a formal axiomatic theory called MLT [20]. MLT formally characterizes the nature of classification levels, and precisely defines the relations that may occur between elements of different classification levels. The authors conceived MLT to be a foundational theory useful to guide the development of well-founded languages for multi-level conceptual modeling and to provide the modeler with basic concepts and

patterns to conceptualize domains that require multiple levels of classification. Corroborating the authors original intentions, MLT has been successfully used to analyze and improve the UML support for modeling the powertype pattern [28], 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 [29] and in foundational ontologies [30].

The theory is founded on the notion of (ontological) instantiation and characterizes the concepts of *individuals* and *types*, with types organized in levels related by instantiation. It 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 2-1). Two types are considered equal iff the sets of all their possible instances are the same (Axiom A2). The constant “First-Order Type” (or shortly “1stOT”) *categorizes the type that applies to all entities whose instances are instances of “Individual”* (A3 in Table 2-1). Analogously, each *entity whose possible extension contains exclusively instances of “1stOT” is an instance of “Second-Order Type”* (or shortly “2ndOT”) (A4 in Table 2-1). Further, instances of “Third-Order Type” are defined analogously to 1stOT and 2ndOT (see Axiom A5). It follows from axioms A1, A3, A4 and A5 that “Individual” is instance of “1stOT” (Theorem T1 in Table 2-1), which is instance of “2ndOT” (Theorem T2 in Table 2-1). “2ndOT”, in its turn, is instance of “3rdOT” (T3). We call “Individual”, “1stOT”, “2ndOT” and “3rdOT” the basic types of MLT. From the combinations of A1 to A5, MLT states that the basic types have no instances in common, i.e., their extensions are disjoint (Theorem T4). According to MLT, every possible entity must be instance of exactly one of its basic types (except the topmost type) (A6 in Table 2-1). Finally, axioms A1 to A6 prescribe a strictly stratified organization of entities into orders, which results that the instance of relation is asymmetric (Theorem T5) and anti-transitive (T6).

Table 2-1. MLT Rules

A1	$\forall x \text{ iof}(x, \text{Individual}) \leftrightarrow \exists y \text{ iof}(y, x)$
A2	$\forall t1, t2 (\neg \text{iof}(t1, \text{Individual}) \wedge \neg \text{iof}(t2, \text{Individual})) \rightarrow ((t1 = t2) \leftrightarrow (\forall e \text{ iof}(e, t1) \leftrightarrow \text{iof}(e, t2)))$
A3	$\forall t \text{ iof}(t, \text{1stOT}) \leftrightarrow (\exists y \text{ iof}(y, t) \wedge (\forall x \text{ iof}(x, t) \rightarrow \text{iof}(x, \text{Individual})))$
A4	$\forall t \text{ iof}(t, \text{2ndOT}) \leftrightarrow (\exists y \text{ iof}(y, t) \wedge (\forall t' \text{ iof}(t', t) \rightarrow \text{iof}(t', \text{1stOT})))$
A5	$\forall t \text{ iof}(t, \text{3rdOT}) \leftrightarrow (\exists y \text{ iof}(y, t) \wedge (\forall t' \text{ iof}(t', t) \rightarrow \text{iof}(t', \text{2ndOT})))$
A6	$\forall x (\text{iof}(x, \text{Individual}) \vee \text{iof}(x, \text{1stOT}) \vee \text{iof}(x, \text{2ndOT}) \vee \text{iof}(x, \text{3rdOT}) \vee (x = \text{3rdOT}))$
D1	$\forall t1, t2 \text{ specializes}(t1, t2) \leftrightarrow (\exists y \text{ iof}(y, t1) \wedge (\forall e \text{ iof}(e, t1) \rightarrow \text{iof}(e, t2)))$
D2	$\forall t1, t2 \text{ properSpecializes}(t1, t2) \leftrightarrow (\text{specializes}(t1, t2) \wedge t1 \neq t2)$
D3	$\forall t1, t2 \text{ isSubordinateTo}(t1, t2) \leftrightarrow (\exists x \text{ iof}(x, t1) \wedge (\forall t3 \text{ iof}(t3, t1) \rightarrow (\exists t4 \text{ iof}(t4, t2) \wedge \text{properSpecializes}(t3, t4))))$
D4	$\forall t1, t2 \text{ isPowertypeOf}(t1, t2) \leftrightarrow (\exists x \text{ iof}(x, t1) \wedge (\forall t3 \text{ iof}(t3, t1) \leftrightarrow \text{specializes}(t3, t2)))$
D5	$\forall t1, t2 \text{ categorizes}(t1, t2) \leftrightarrow (\exists x \text{ iof}(x, t1) \wedge (\forall t3 \text{ iof}(t3, t1) \rightarrow \text{properSpecializes}(t3, t2)))$
D6	$\forall t1, t2 \text{ completelyCategorizes}(t1, t2) \leftrightarrow (\text{categorizes}(t1, t2) \wedge (\forall e \text{ iof}(e, t2) \rightarrow \exists t3 (\text{iof}(e, t3) \wedge \text{iof}(t3, t1))))$
D7	$\forall t1, t2 \text{ disjointlyCategorizes}(t1, t2) \leftrightarrow (\text{categorizes}(t1, t2) \wedge \forall e, t3, t4 ((\text{iof}(t3, t1) \wedge \text{iof}(t4, t1) \wedge \text{iof}(e, t3) \wedge \text{iof}(e, t4)) \rightarrow t3 = t4))$
D8	$\forall t1, t2 \text{ partitions}(t1, t2) \leftrightarrow (\text{completelyCategorizes}(t1, t2) \wedge \text{disjointlyCategorizes}(t1, t2))$
T1	$\text{iof}(\text{Individual}, \text{1stOT})$
T2	$\text{iof}(\text{1stOT}, \text{2ndOT})$
T3	$\text{iof}(\text{2ndOT}, \text{3rdOT})$
T4	$\exists x ((\text{iof}(x, \text{Individual}) \wedge \text{iof}(x, \text{1stOT})) \vee (\text{iof}(x, \text{Individual}) \wedge \text{iof}(x, \text{2ndOT})) \vee (\text{iof}(x, \text{Individual}) \wedge \text{iof}(x, \text{3rdOT})) \vee (\text{iof}(x, \text{1stOT}) \wedge \text{iof}(x, \text{2ndOT})) \vee (\text{iof}(x, \text{1stOT}) \wedge \text{iof}(x, \text{3rdOT})) \vee (\text{iof}(x, \text{2ndOT}) \wedge \text{iof}(x, \text{3rdOT})))$
T5	$\exists x, y (\text{iof}(x, y) \wedge \text{iof}(y, x))$
T6	$\exists x, y, z (\text{iof}(x, y) \wedge \text{iof}(y, z) \wedge \text{iof}(x, z))$
T7	$\forall t \text{ iof}(t, \text{1stOT}) \leftrightarrow \text{specializes}(t, \text{Individual})$
T8	$\forall t \text{ iof}(t, \text{2ndOT}) \leftrightarrow \text{specializes}(t, \text{1stOT})$
T9	$\forall t \text{ iof}(t, \text{3rdOT}) \leftrightarrow \text{specializes}(t, \text{2ndOT})$
T10	$\text{isPowertypeOf}(\text{1stOT}, \text{Individual})$
T11	$\text{isPowertypeOf}(\text{2ndOT}, \text{1stOT})$
T12	$\text{isPowertypeOf}(\text{3rdOT}, \text{2ndOT})$
T13	$\forall p, t \text{ isPowertypeOf}(p, t) \rightarrow \exists p' (p \neq p') \wedge \text{isPowertypeOf}(p', t)$
T14	$\forall p, t \text{ isPowertypeOf}(p, t) \rightarrow \exists t' (t \neq t') \wedge \text{isPowertypeOf}(p, t')$
T15	$\forall t1, t2, t3, t4 (\text{specializes}(t2, t1) \wedge \text{isPowertypeOf}(t4, t2) \wedge \text{isPowertypeOf}(t3, t1)) \rightarrow \text{specializes}(t4, t3)$
T16	$\forall t1, t2, t3 (\text{isSubordinateTo}(t1, t2) \wedge \text{categorizes}(t2, t3)) \rightarrow \text{categorizes}(t1, t3)$
T17	$\forall t1, t2, t3 (\text{isPowertypeOf}(t2, t1) \wedge \text{categorizes}(t3, t1)) \rightarrow \text{properSpecializes}(t3, t2)$
T18	$\forall t1, t2, t3 (\text{partitions}(t1, t3) \wedge \text{partitions}(t2, t3)) \rightarrow \neg \text{properSpecializes}(t1, t2)$

Some structural relations to support conceptual modeling are defined in MLT, starting with the ordinary specialization between types. A type $t1$ *specializes* another type $t2$ iff *all instances of $t1$ are also instances of $t2$* (see definition D1 in Table 2-1). Since the reflexivity of the *specialization* relation may be undesired in some contexts, we define in MLT the *proper specialization* relation as follows: *$t1$ proper specializes $t2$ iff $t1$ specializes $t2$ and $t1$ is different from $t2$* (see D2 in Table 2-1). The definition D1 and the Axioms A3, A4 and A5 lead to a basic pattern in MLT: every type that is not one of MLT's basic types (e.g., a domain type) is an instance of one of the basic higher-order types (e.g., "1stOT", "2ndOT" and "3rdOT"), and, at the same time proper specializes the basic type at the immediately lower level (Theorems T7, T8 and T9). Additionally, MLT defines a *subordination* relation. *Subordination* between two higher-order types implies *specializations* between their instances, i.e., *$t1$ is subordinate to $t2$ iff every instance of $t1$ proper specializes an instance of $t2$* (see D3 in Table 2-1). 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 [18], MLT defines a *powertype* relation between a higher-order type and a base type at a lower order: a type $t1$ *is powertype of* a base type $t2$ iff *all instances of $t1$ specialize $t2$ and all possible specializations of $t2$ are instances of $t1$* (see D4). Note that it follows from the axioms and definitions presented so far that "1stOT" *is powertype of* "Individual" (T10), 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" (T11), and so on (T12). Thus, every instance of a basic higher-order type ("1stOT", "2ndOT" and "3rdOT") must specialize the basic type at the immediately lower level (respectively, "Individual", "1stOT" and "2ndOT"). 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. According to MLT, *each type has at most one powertype* (Theorem T13) and *that each type is powertype of, at most, one other type* (Theorem T14), which is a concrete syntactic constraint for a multi-level model: each type could have exactly one powertype. Finally, from the

powertype definition of Cardelli [18], MLT states that *if a type $t2$ specializes a type $t1$ then the powertype of $t2$ specializes the powertype of $t1$* (T15).

Differently from Cardelli, Odell [17] 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 *categorization* relation between types at adjacent levels: *a type $t1$ categorizes a type $t2$ iff all instances of $t1$ are proper specializations of $t2$* (definition D5). The *categorizes* relation occurs between a higher-order type $t1$ and a base type $t2$ when *the instances of $t1$ specialize $t2$* 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 $t2$ that are not instances of $t1$. For example, we may define a type named “Organism by Habitat” (with instances “Terrestrial Organism” and “Aquatic Organism”) that *categorizes* “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”). From the definitions D3 and D5 and, further, D2, it is concluded that *if a type $t1$ is subordinate to $t2$ and $t2$ categorizes a type $t3$ then $t1$ categorizes $t3$* (Theorem T16). Now, considering D4, D5 and D2, if a type $t2$ is *powertype of a type $t1$* and a type $t3$ *categorizes* the same base type $t1$ then all instances of $t3$ are also *instances of the powertype $t2$* (T17).

MLT defines some refinements of the cross-level relation of categorization, which are useful to capture further constraints in multi-level models. We consider that *a type $t1$ completely categorizes $t2$ iff $t1$ categorizes $t2$ and every instance of $t2$ is instance of, at least, an instance of $t1$* (D6). Moreover, *iff $t1$ categorizes $t2$ and every instance of $t2$ is instance of, at most, one instance of $t1$* it is said that *$t1$ disjointly categorizes $t2$* (D7). Finally, a common use for the notion of powertype in the literature considers a higher-order type that, simultaneously, *completely and disjointly categorizes* a lower-order type. To capture this notion MLT defines the *partitions* relation. Thus, *$t1$ partitions $t2$ iff each instance of the base type $t2$ is an instance of exactly one instance of $t1$* (D8). For example, considering the biological taxonomy for living beings we have that “Species” (and all other biological ranks) *partitions* “Organism”. Finally, *a consequence of the partitions definition is that, if two types $t1$ and $t2$ both partitions the same type $t3$ then it is not possible for $t1$ to specialize $t2$* (T18). A complete formalization of MLT in first-order logic can be found in [20].

2.4 REQUIREMENTS FOR A MULTI-LEVEL APPROACH

We establish here four requirements we judge important for multi-level modeling approaches, and that will guide the solutions proposed in this work. These requirements focus on the expressiveness of the resulting models to capture knowledge in multi-level domains. We indicate sources in the literature that establish similar requirements to corroborate the relevance of the requirements identified here.

First of all, we consider an essential requirement for a multi-level modeling approach *the ability to represent entities of multiple (related) classification levels* ([6], [7]), capturing chains of instantiation between the involved entities (requirement **R1**). (This requirement is also suggested by Gonzalez-Perez and Henderson-Sellers [6] and Neumayr et al. [7].) To comply with it, the approach must admit entities that are, simultaneously, type (class) and instance (object), complying thus to the notion of *clabject* [19]. 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.

A second requirement we establish is that 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 [19]. It assumes that each element of a level must be an instance of an element of the level above. Our motivation for these requirements is has an empirical nature, since we have observed that the lack of principles to guide organization of entities into levels often leads to the construction of unsound multi-level models [14].

Another important characteristic of domains with multiple levels of classification is that there are domain rules that apply to the instantiation of types of different levels. This kind of rule is suggested by Gonzalez-Perez and Henderson-Sellers [6], inspired in the *powertype pattern* [17], [18]. For example, all instances of *Dog Breed* (e.g. *Collie* and *Beagle*) specialize the base type *Dog*. It is thus key that multi-level modeling approaches support the representation of what kind of relationship exists between *Dog Breed* and *Dog* (we call this

sort of relation a *structural relation*, as it governs the instantiation of types at different levels). For instance, one may need to represent whether an instance of *Dog* may instantiate: (i) only one, or (ii) more than one *Dog Breed*. Moreover, we would like to represent whether an instance of *Dog* must instantiate at least one *Dog Breed*. In this case, an instance of *Dog* must instantiate exactly one (i.e., at least one and only 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* (e.g., *Chordate phylum* specializes *Animal kingdom*), every instance of *Class* specializes one instance of *Phylum* (*Mammal class* specializes *Chordate 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). (A requirement also identified by Neumayr et al. [7].)

The characterization of MLT basic types in tandem with the definition that every entity must be instance of one basic type, provide support for the representation of multiple levels of classification (R1) as well as guidelines for the organization of entities into levels (R2). Further, the MLT structural relations provide expressive support to capture rules for the instantiation of types at different levels (R3). Finally, according to MLT, domain relations between entities in different levels are allowed (R4). Thus, since MLT is a well-founded theory which satisfies the four requirements, we choose it as semantic foundation for our approach.

In Chapter 3, we review the current Semantic Web approaches concerning their adherence to these requirements. They are later satisfied by the approach proposed in Chapter 4.

In order to show the limitations of the various approaches in the representation of a multi-level domain, we used as a paradigmatic example 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*). For example, *Cecil* is an instance of *Lion*, which is an instance of *Species*. As consequence of being an instance of *Lion*, *Cecil* is also instance of *Panthera* (instance of *Genus*), *Felidae* (instance of *Family*), *Carnivoran* (instance of *Order*), *Mammal* (instance of *Class*), *Chordata* (instance of *Phylum*), *Animal* (instance of *Kingdom*), and *Organism*. *Species*, *Genus*, *Family*, *Order*, *Class*, *Phylum* and *Kingdom*, in its turn, are instances of *Taxonomic Rank*. Moreover, in this domain, all instances of *Species* are subtypes of *Organism*, and all instances of *Organism* belong to one and only one *Species*. The same occurs from *Genus*, *Family*, *Order*, *Class*, *Phylum* and *Kingdom* to *Organism*. For example, all instances of *Genus* are subtypes of *Organism*, and all instances of *Organism* belong to one and only one *Genus*. Further, all instances of *Species* are subtypes of instances of *Genus*, all instances of *Genus* are subtypes of instances of *Family*, all instances of *Family* are subtypes of instances of *Order*, and so on. For example, *Lion* (instance of *Species*) is subtype of *Panthera* (instance of *Genus*), *Panthera* is subtype *Felidae* (instance of *Family*), *Felidae* is subtype of *Carnivoran* (instance of *Order*), and so on. Finally, an instance of *Species* is named by a *Person*. For instance, the researcher Lessner named the *Vivaron haydeni* species, an extinct reptile related to crocodiles [31].

3 RELATED WORK: CURRENT APPROACHES TO MULTI-LEVEL MODELING IN THE SEMANTIC WEB

In this chapter, we examine the state-of-the-art in approaches that support multi-level modeling in the web. Initially, we discuss one of the main recommendations of W3C for the Semantic Web, namely, RDFS (Section 3.1). We then discuss a related work regarding improvements in this recommendation: RDFS(FA) (Section 3.2), which proposes improvements in RDFS. Further, we discuss other main recommendation of W3C: OWL (Section 3.3); for then discuss OWL FA (Section 3.4), which proposes improvements in OWL. Moreover, we present PURO (Section 3.5), which is implemented in OWL, and the Wikidata approach (Section 3.6), which underlies the Wikidata knowledge base and defines its own vocabulary and vocabulary structuring mechanisms. Finally, in Section 3.7, we present a summary of the existing approaches and their evaluation according to the requirements for multi-level modeling approaches previously defined in Chapter 2.

3.1 RDFS

RDF (Resource Description Framework) and RDFS (RDF Schema) [3] are languages proposed by W3C for use in the Semantic Web. They are intended for creating vocabularies and for publishing and linking data on the web. RDFS extends the basic vocabulary of RDF aiming to provide terms for creating domain vocabularies.

First of all, RDF uses the notion of “triple” to represent information. In a triple, a resource (the subject) is connected to a literal or to other resource (the object) through a property (the predicate). For RDF, a resource is anything identified by an IRI (Internationalized Resource Identifier) [32] (e.g., the IRI <http://www.wikidata.org/entity/Q80> identifies *Tim Berners-Lee* in Wikidata [15]). Thus, to represent that *Tim Berners-Lee* was born in *London*, for example, we must create a triple which states that *Tim Berners-Lee* (subject) *was born in* (predicate) *London* (object). Figure 3-1 shows a labeled graph representing this statement.

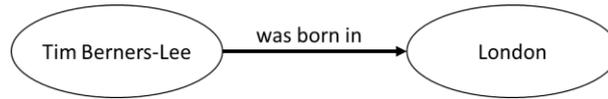


Figure 3-1. Labeled graph representing a triple about Tim Berners-Lee

The notion of class (*rdfs:Class*) is introduced in RDFS to represent specific sets of individuals that share the same characteristics. Thus, domain classes (such as *Person*) are represented as instances of *rdfs:Class* through the *rdf:type* property. *rdfs:Class* is the main primitive of RDFS, such that other important primitives are defined as its instances (including *rdfs:Class* itself): these include *rdfs:Resource*, which represents anything that has an IRI (e.g. *Tim Berners Lee*, *London*, *Person* or “*was born in*”), and *rdf:Property*. It is important to note that both *rdfs:Class* and *rdf:Property* are subclasses of *rdfs:Resource*. Thus, domain classes and properties are considered special kinds of resources, along with concrete individuals. Further, RDFS also introduces *rdf:Property* to represent predicates (such as “*was born in*”). Finally, *rdfs:subClassOf* and *rdf:type* properties are two important primitives of RDFS that are instances of *rdf:Property*. *rdfs:subClassOf* is used to represent that all instances of a class must be instances of other class. Figure 3-2 shows this fragment of RDFS. In Figure 3-2 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 instantiation statements, with labels to denote the names of the predicates that apply. For instance, a dashed arrow labeled *rdf:type* between *rdfs:subClassOf* and *rdf:Property* represents that the former is an instance of the latter. The notation used to elaborate Figure 3-2 is used in all further diagrams.

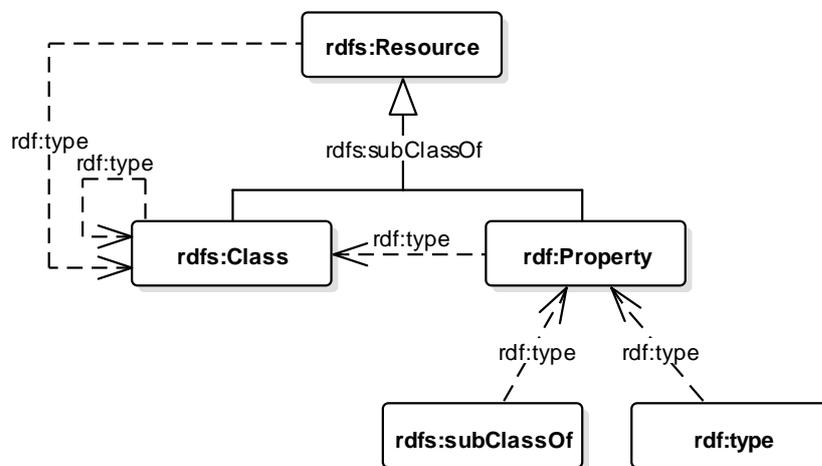


Figure 3-2. Fragment of RDFS vocabulary

We could extend the information about *Tim Berners-Lee* to say that he is an instance of *Person* and that *London* is an instance of *City*. For that, we must state that *Tim Berners-Lee* and *London* are instances of (through *rdf:type*) *Person* and *City*, respectively. With RDFS, we are able to extend Figure 3-1, and explicitly represent the classes in the domain. For example, we could represent *Person* and *City* as instances of the language primitive *rdfs:Class*, since these represent sets of individuals that share the same characteristics. Figure 3-3 shows this example.

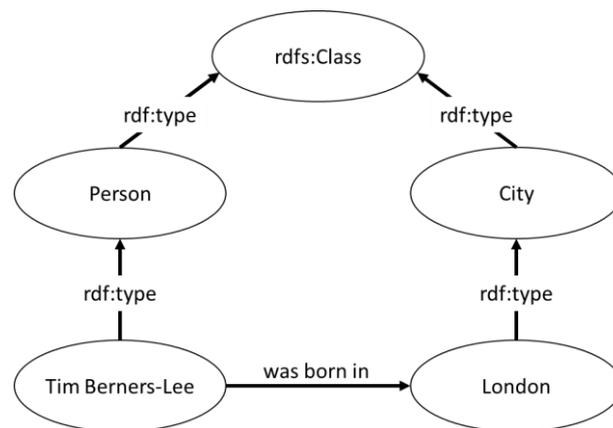


Figure 3-3. Labeled graph representing triples about Tim Berners-Lee, London, Person and City

Note that, the fact *Tim Berners-Lee* is an instance of *Person* concerns information about the domain, while *Person* being an instance of *rdfs:Class* is related to its representation in a language (RDFS). Observing these two different cases of instantiation, Atkinson and Kühne proposes the Orthogonal Classification Architecture (OCA) [33]. OCA is a modeling framework that distinguishes domain-oriented “ontological” classification relationships from language-infrastructure-oriented “linguistic” classification relationships and organizes them according to the tenets of strict metamodeling [33]. While “linguistic” metamodeling is concerned with language definition issues, “ontological” metamodeling is concerned with classification relations that may occur between domain types.

The *rdf:type* property is used indistinctively to represent both ontological and linguistic instantiations. Considering the usage of *rdf:type* to represent ontological instantiation, we can state that RDFS provides support for representing multiple levels of classification, satisfying the requirement R1. Moreover, as aforementioned, the only constraint for building triples in RDFS is that the subject must be a resource (i.e., something identified by an IRI) and that the object can be a resource or a literal. Thus, triples can link any pair

of resources, without restrictions, allowing the representation of domain relations between entities of different orders (requirement R4). Further, no rules are defined to guide the use of *rdf:type* property nor to support the organization of entities into classification levels. Therefore, we conclude that RDFS does not satisfy the requirement R2. A consequence of this to the biological taxonomy example described in Chapter 2 is that RDFS provides no guidelines concerning the possible relations between *Cecil*, *Lion*, *Species* and *Taxonomic Rank*. Note that, according to MLT we can identify that these are entities at different orders and that there are rules that apply to them (such as, e.g., that *Lion* cannot be an instance of *Taxonomic Rank*, that *Species* cannot subclass *Taxonomic Rank*). Finally, RDFS does not provide constructs to express, for example, that every instance of an instance of a powertype must also be instance of the base type (the power type pattern), not satisfying thus the requirement R3. In the biological taxonomy example, the lack of support for R3 makes it impossible to represent that all instances of *Species* must specialize exactly one instance of *Genus*.

3.2 RDFS(FA)

In an early effort to organize the metamodeling architecture for RDFS 1.0, Pan and Horrocks proposed RDFS(FA) [9]. They observed that “RDFS uses a single primitive *rdf: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”, where the modeler ends-up making inadequate *ad hoc* language extensions. The authors argue that these extensions are undesirable and that the modeler may confuse language extension with domain modeling, since the same mechanisms can be used for both. Inspired by the fixed UML metamodeling architecture [34], 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 modeling 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*.

Figure 3-4 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*.

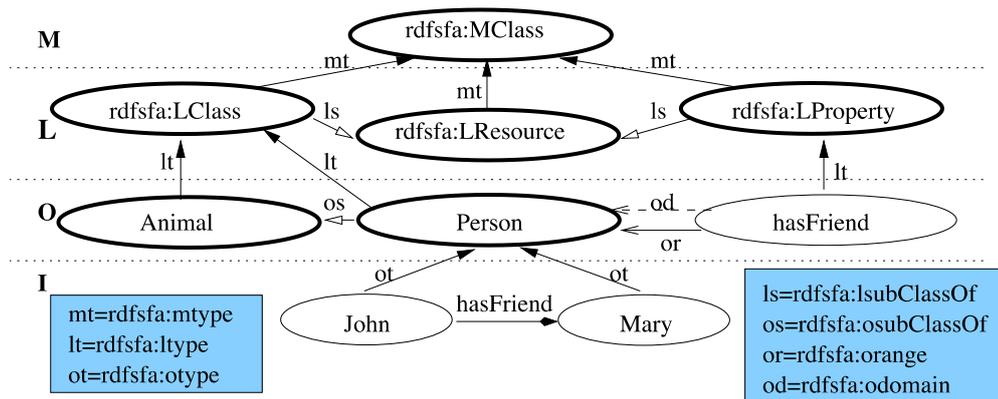


Figure 3-4. Example of Directed Labeled Graph of RDFS(FA) (from [9])

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 [33]). 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 [33]). 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 [33]. In conclusion, RDFS(FA) satisfies requirements R1 and R2 only for linguistic instantiation, but not for ontological instantiation. A consequence of this limitation is that in the biological taxonomy example described in Chapter 2, RDFS(FA) does not allow modelers to represent that *Lion* is an instance of *Species* (R1). Since RDFS(FA) does not support R1 for ontological instantiation, then it does not offer special support for expressing: (i) the relations

between a higher-order class and a base class in the powertype pattern (failing thus to satisfy R3); and domain relations crossing levels (not satisfying R4). As a consequence, important multi-level constraints cannot be represented in the biological taxonomy scenario. For example, it is not possible to capture the constraint that every instance of *Organism* must be instance of exactly one instance of *Species*.

3.3 OWL 2

OWL 2 [4] is also a language proposed by W3C, which is defined in terms of RDFS. OWL introduces new terms (besides the existing ones in RDFS), with the purpose of increasing the expressiveness of domain vocabularies, while maintaining key choices in the Semantic Web underlying RDF and RDFS. We focus here on the features of OWL that can be used to represent multi-level models.

OWL introduces a new term for representing classes: *owl:Class*, which is an instance of *rdfs:Class*. Moreover, these terms are declared as subclasses (specializations) of each other, which makes them equivalent. OWL introduces a superclass of *rdfs:Resource* which is used to classify everything: *owl:Thing*. Since RDFS allows properties to connect a resource to a literal or to a resource, OWL introduces terms which specializes *rdf:Property* to distinguish each of these cases: (i) instances of *owl:DatatypeProperty* connect a resource to a literal, and (ii) instances of *owl:ObjectProperty* connect a resource to a resource. Thus, domain relations (such as “*was born in*”) are represented as instances of *owl:ObjectProperty*. Further, instances of (i) *rdfs:Class/owl:Class* and (ii) *owl:ObjectProperty* are instances of *owl:Thing*, since all of them are (indirectly) subclasses of it. And, since concrete individuals are included as resources, they are also instances of *owl:Thing*. This fragment of OWL is presented in Figure 3-5. To increase the readability of the diagram, we omitted the representation of *rdf:type* from *owl:Thing* to *owl:Class* and from all other elements to *rdfs:Class*.

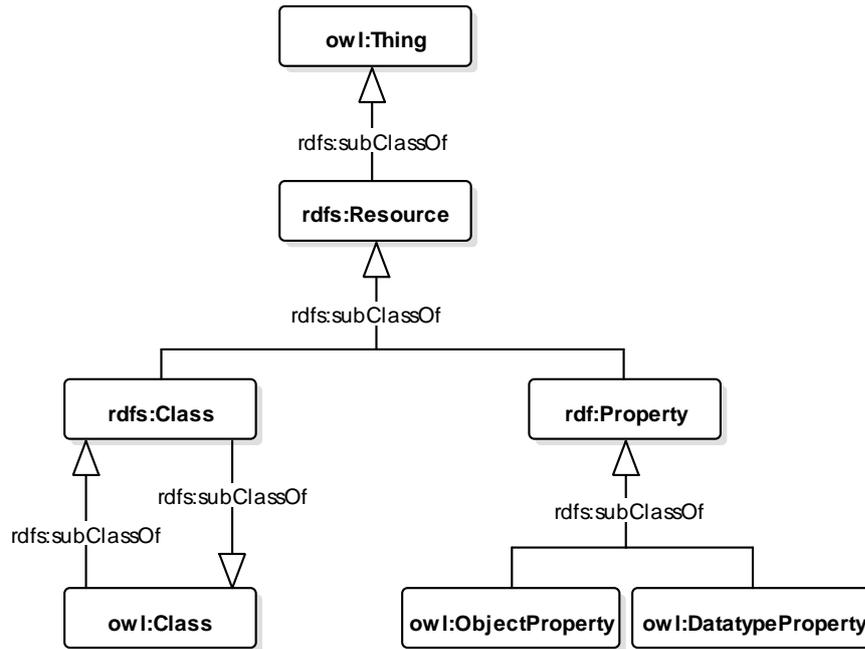


Figure 3-5. Fragment of OWL vocabulary

Similarly to RDFS, OWL supports the representation of classes of classes, but explicitly under the term *metamodeling*. For example, in Figure 3-6, two subclasses of *Eagle*, namely *Golden Eagle* and *Steppe Eagle* are defined as instances of *Species*, which means that they are members of the set of all species. Finally, we use the instance specification notation of UML (i.e., underlining an element's name) to represent an individual (e.g. Harry).

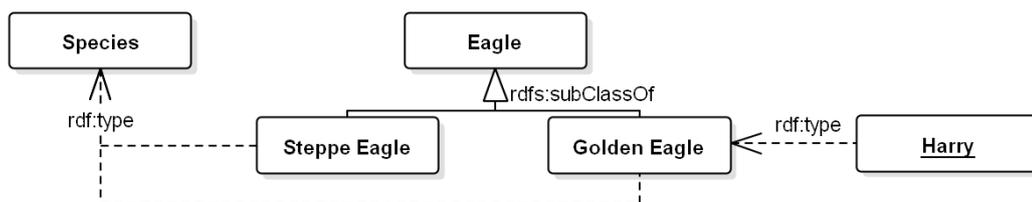


Figure 3-6. OWL representation for biological taxonomic domain

Despite introducing new elements, the OWL metamodeling architecture is similar to RDFS's. However, since OWL was designed with concerns for the tractability of reasoning and inference, the designers have opted to constrain the language's expressiveness. Because of this, OWL's multi-level modeling support is based on the notion of contextual semantics [10], often referred to as *punning*. According to *punning* principles, (i) a class is seen as an individual when it is an instance of another class, and (ii) the interpretation of an entity as a class and it is completely independent of its interpretation as an individual. This "independent" interpretation means that a constraint

stated to the interpretation of an entity as a class will not be considered when this entity 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 derive 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 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*).

3.4 OWL FA

Pan and Horrocks also proposed OWL FA [11], [13], 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 allowing the reuse of existing reasoners.

While RDFS(FA) uses prefixes (M, L, O and I) to indicate the layer to 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 [13], [35] take into account elements that share the same URIs (Uniform Resource Identifier) and interpret them dependently (in contrast to OWL). 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). As a consequence, in the paradigmatic example described in Chapter 2, it is not possible to represent who named the *Golden Eagle* species.

The numbered layers appear to merge *Ontology* and *Instance Layers* of RDFS(FA). Thus, we understand here that identifying layers by numbers addresses the limitation of RDFS(FA) (see 3.2) thus satisfying R1 fully. Moreover, as advantages when compared to the current *multi-level modeling* support of OWL (see 3.3), 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).

3.5 PURO

Svatek et al. [12] propose 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 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 and OWL FA, PURO offers no special support for the representation of constraints for the instantiation of types at different levels (not satisfying R3).

3.6 THE WIKIDATA APPROACH

The importance of structured data on the web has become clear in the recent years, and has fed developments to make it possible for data to be shared and reused across application, enterprise, and community boundaries [1]. Currently, many initiatives focus on structured data in an effort to facilitate the automated processing of data, as opposed to human consumption through natural language. One prominent initiative with this focus is Wikidata [15]: a project of the Wikimedia Foundation to capture the structured data underlying Wikipedia, the popular online encyclopedia, and other Wikimedia sister projects. The content of Wikidata is available under a free license, and can thus be consumed and linked to other data sets on the linked data web.

The Wikidata repository consists mostly of *items* and *statements* about these items. *Items* are used “to represent all the things in human knowledge, including topics, concepts, and objects”, and are given a unique identifier, a label and a description [36]. *Statements* are used “for recording data about an item”, and “consist of (at least) one property-value pair”; they serve to “connect items to each other, resulting in a linked data structure” [37].

In order to organize Wikidata’s content, some items (termed *classes*) may be used to classify other items through the *instance of* property (which has the unique identifier P31). For example, the item *London* (Q84) is related to the item *city* (Q515) through the *instance of* property, to represent the fact that London is a city. Further, classes can be related through the *subclass of* (P279) taxonomic property, defining thus hierarchies of classes, from more general to more specific ones [38]. For example, *city* and *country* (Q6256) are subclasses of *administrative territorial entity* (Q56061), which is a subclass of *human-geographic territorial entity* (Q15642541). The definition of *instance of* provided in Wikidata is informal and silent about its formal logic properties (symmetry, reflexivity and transitivity). Moreover, Wikidata declares that *instance of* and *rdf:type* are *equivalent properties* (P1628). Further, *subclass of* is characterized as transitive and asymmetric (i.e., antisymmetric and irreflexive) and as *equivalent property of rdfs:subClassOf*.

To illustrate this, we extracted from Wikidata a fragment of a biological taxonomy and the classification of *Cecil* (the lion) in such taxonomy. *Cecil is instance of Panthera Leo*, which is *instance of Species*. *Species*, in its turn, is *instance of Taxonomic Rank*. Considering the definition of *subclass of*, we can conclude that *Cecil* is also *instance of Panthera* and, consequently, of all its super classes. See Figure 3-7. Additionally, in order to increase the readability of the diagram, we use dashed rectangles to group elements that instantiate the same other element and draw only one arrow from the border of the rectangle to the other element.

While, in the model of Figure 3-7, modelers have been able to organize the model adequately into strata, there is no support to prevent a Wikidata contributor from violating this conformant structure. For example, a clearly incorrect modification introducing a new entity (e.g. “Simba”) which is both an instance of *Panthera Leo* and *Species* would go undetected, and would result in an inconsistent hierarchy¹.

¹ In fact, we have detected a large number of those problematic hierarchies (see Chapter 5).

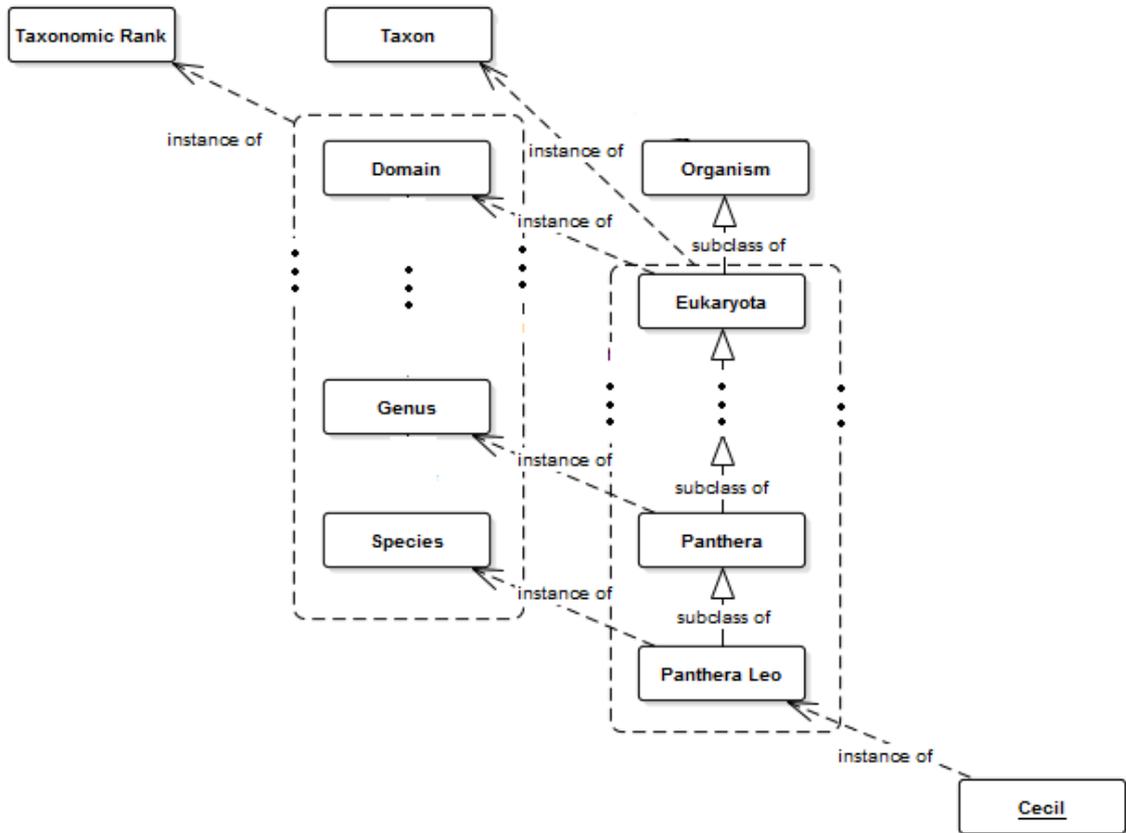


Figure 3-7. Short representation for Taxonomic Biological Domain in Wikidata

Similarly with the previously presented approaches, Wikidata provides support for representing many levels of instantiation (R1) through the possibility of chains of the *instance of* property. Like RDFS, Wikidata allows statements between any pair of items, thus it is possible to represent domain relations between entities in different levels (R4). However, since Wikidata is silent about formal logic properties of *instance of*, it offers no special support both for organization of entities along levels (not satisfying R2) and for the representation of constraints for the instantiation of types at different levels (not satisfying R3).

3.7 CONCLUDING REMARKS

Here, we summarize the review of the current Semantic Web approaches concerning their adherence to the four requirements we established for multi-level approaches in Chapter 2. Table 3-1 summarizes the analysis, classifying their support for each requirement considering three categories: fully covered, partially covered and not covered.

Table 3-1. Support for multi-level modeling in RDFS languages

Requirement	RDFS	RDFS(FA)	OWL 2	OWL FA	PURO	Wikidata
R1 – represents entities of multiple levels of classification	Yes.	Partially.	Partially.	Yes.	Yes.	Yes.
R2 – offers guidance for the organization of entities into levels	No.	Partially.	No.	Yes.	Yes.	No.
R3 – represents rules for the instantiation of types at different levels	No.	No.	No.	No.	No.	No.
R4 – supports domain relations between entities of different levels	Yes.	No.	Partially.	No.	Yes.	Yes.

We consider that RDFS provides support for representing multiple levels of classification (R1) and that it allows the representation of domain relations between entities of different orders (R4). However, no specific rules are defined to guide the use of the *rdf:type* property (not satisfying R2) and no mechanism is provided to constrain the instantiation of types at different classification levels (R3).

The Wikidata approach is quite similar to RDFS. Two of its main properties, *instance of* (P31) and *subclass of* (P279), are said to be equivalent to (P1628) RDFS properties *rdf:type* and *rdfs:subClassOf*, respectively. Similarly to RDFS, the Wikidata approach supports multiple levels of classification (R1) and domain relations crossing levels (R4). However, it offers no special support concerning the usage of instantiation and the relation between higher-order types and base types, failing to support both R2 and R3.

RDFS(FA) provides support and guidance for representing multiple levels of classification, however focusing on linguistic instantiation instead of ontological instantiation. Because of this focus, when compared with RDFS, RDFS(FA) sacrifices R1 (which we consider only partially addressed) as well as R4, while providing support for R2. Since RDFS(FA) does not support R1 for ontological instantiation, it does not offer special support for expressing the relations between a higher-order class and a base class in the powertype pattern (not satisfying R3).

Similarly to RDFS, OWL supports the representation of classes of classes, but under the term *metamodeling*. Because of the need to maintain tractable inferencing mechanisms, OWL metamodeling is based on the *contextual semantics* [10] (*punning*). This choice affects the representation of relations and constraints crossing levels and hence offers only partial support for R1 and R4 in contrast with RDFS, which we consider offers full support for R1 and R4. OWL fails to support R2 and R3.

Finally, OWL FA and PURO offer full support for R1 and R2. However, OWL FA restricts domain relations only for entities in the same level, while PURO supports domain relations crossing levels (R4). Thus, PURO appears as the approach that meets the largest number of requirements we have identified.

Despite the efforts in all these approaches, none of them support the representation of constraints involving instantiation relations across levels (i.e., none of them satisfy R3). The shortcomings of the existing approaches motivate our investigation into a novel approach for multi-level modeling for the Semantic Web.

4 EXPRESSIVE MULTI-LEVEL MODELING FOR THE SEMANTIC WEB

In this chapter, we present our approach to improve the expressivity of multi-level modeling in the Semantic Web. For that, we first present and justify some basic choices for the approach (Section 4.1). Then, we show the proposed vocabulary based on distinctions put forth by MLT (Section 4.2), and a number of integrity (Section 4.3) and derivation (Section 4.4) rules reflecting axioms and theorems of MLT. Further, we explain how the vocabulary and rules are combined in an application that helps modelers to produce sound models (Section 4.5). Finally, we present some conclusions regarding the usage of our approach (Section 4.6).

4.1 PRELIMINARY CONSIDERATIONS

In this thesis, we are interested in providing expressiveness for representing multi-level domains in the Semantic Web satisfying the four requirements for multi-level modeling approaches defined in Section 2.4 and maintaining the current standards of Semantic Web languages and metamodeling infrastructures.

We use MLT as foundation for this approach, since MLT achieves the four-requirements. Then, considering that RDFS has the ability to represent entities of multiple (related) classification levels (satisfying R1) we maintain the use of *rdf:type* to represent the instantiation relations. This design decision corroborates with the principle of Linked Data of reusing terms from widely deployed vocabularies whenever their semantics correspond to the intended ones [39].

To provide some organization for the multi-level models (in order to satisfy R2), we: (i) represent the basic types of MLT as instances of *owl:Class* (adhering to the infrastructure standard), and (ii) provide queries implementing MLT rules to ensure that the stratification into orders is respected by domain vocabularies that instantiate and specialize these basic types.

To provide support to capture rules for the instantiation of types at different levels (in order to satisfy R3) we: (i) incorporate the MLT relations representing them as instances of *owl:ObjectProperty* (adhering to the infrastructure standard), and (ii) provide queries to verify the soundness of vocabularies that use these relations.

Finally, considering that OWL supports the representation of domain relations between entities in different classification levels, the domain relations in our approach are represented as instances of *owl:ObjectProperty* adhering, thus, to the infrastructure standard and satisfying to R4.

4.2 THE MLT VOCABULARY FOR THE SEMANTIC WEB

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*², 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 an *rdf:subClassOf* relation with *owl:Class* capturing the fact that their instances are classes (i.e. their instances are instantiable) (see Figure 4-1).

² 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 [20]. The choice for the prefix “token” comes from its use in linguistics. In linguistics, the term refers to entities that do not have instances, contrasting with the notion of types [51].

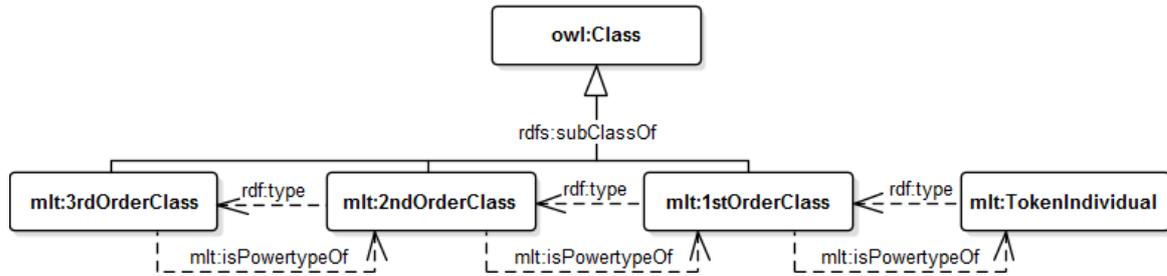


Figure 4-1. Fragment of MLT Vocabulary for Order Classes and Individual.

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 *categorization* (*mlt:categorizes*) and the *is powertype of* (*mlt:isPowertypeOf*) relations are represented as sub-properties of *mlt:crossLevelProperty*. Finally, each variation of *categorization* (e.g. *complete categorization*, *disjoint categorization* and so on) is represented as a sub-property of *mlt:categorizes*.

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*), capturing thus the MLT Theorems T1-T3. 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 (Figure 4-1), capturing thus the Theorems T10-T12. Finally, we use *owl:AllDisjointClasses* to capture in OWL the MLT Theorem T4, which states that MLT basic types are disjoint. The vocabulary is available at [40] and at Appendix I.

The MLT vocabulary allows the representation of domain rules concerning the instantiation of types in different levels. For example, Figure 4-2 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 4.4, some of these relations can be inferred automatically, using derivation rules reflecting MLT axioms and theorems.)

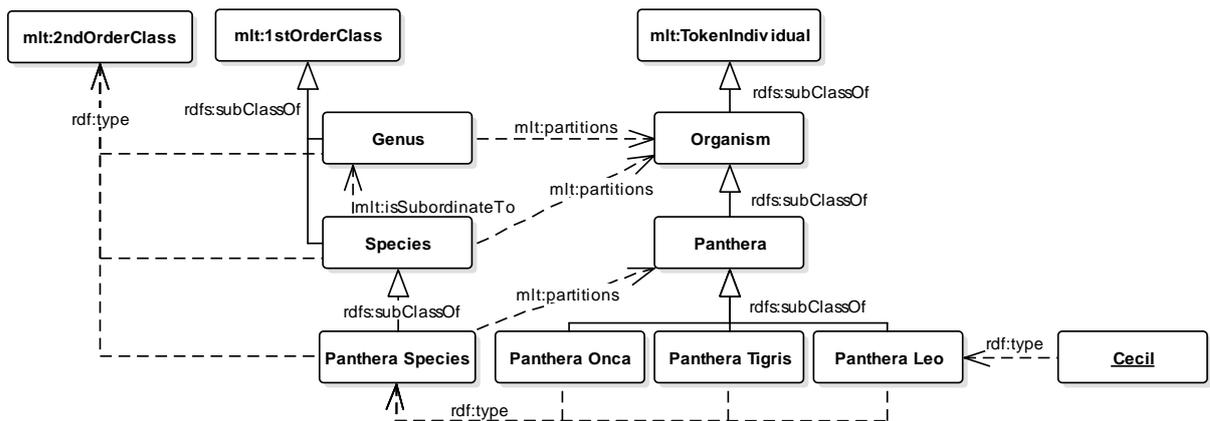


Figure 4-2. Illustrating the use of *mlt:isSubordinateTo* and *mlt:partitions* properties.

Figure 4-3 shows an example of an ontology representing employees and their roles in a company to illustrate the use of variations of *categorization* 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:completelyCategorizes 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:disjointlyCategorizes Employee*.

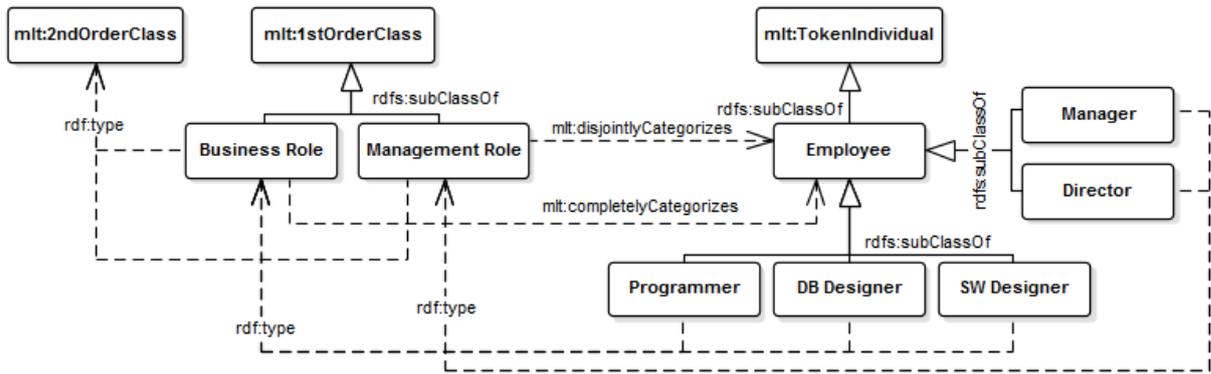


Figure 4-3. Illustrating the use of `mlt:completelyCategorizes` and `mlt:overlappinglyCategorizes`.

4.3 INTEGRITY CONSTRAINTS 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 (Theorem T4 - Table 2-1).

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:categorizes* 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 4-1 shows the domain/range restrictions for MLT relations.

Table 4-1. Domain and range restrictions for multi-level relations.

Relation name	Domain and Range
<i>rdfs:subClassOf</i>	Classes of the same order (instances of 1st, 2nd or 3rd Order Classes)
<i>isSubordinateTo</i>	Higher-order classes of the same order (2ndOrderClass or 3rdOrderClass)
<i>rdf:type</i>	Elements of adjacent levels.
<i>isPowertypeOf</i>	Classes of adjacent levels: - 2ndOrderClass → 1stOrderClass - 3rdOrderClass → 2ndOrderClass
<i>categorizes</i>	
<i>completelyCategorizes</i>	
<i>incompletelyCategorizes</i>	
<i>disjointlyCategorizes</i>	
<i>overlappinglyCategorizes</i>	

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 IC1 in Table 4-2, which would detect violations of this constraint).

According to the definition of *categorization* (see D5) the instances of the higher-order type are *proper specializations* of the base type, i.e. the base type cannot be an instance of the higher-order type that categorizes it. The integrity constraint IC2 captures this issue.

Furthermore, axioms A1 and A6 (see Table 2-1) prescribe a strictly stratified organization of entities into orders, which is called strict metamodeling principle. As a result, the strict metamodeling principle is also guarantee through two SPARQL queries checking properties of instantiation: (i) asymmetry (see IC3 in Table 4-2), and (ii) and anti-transitivity (see IC4 in Table 4-2).

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 IC6 in Table 4-2), and (ii) a class can be the object of at most one triple having *mlt:isPowertypeOf* as predicate (see IC5 in Table 4-2).

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* (IC7 in Table 4-2).

Table 4-2. Integrity Constraints Corresponding to MLT Rules

MLT Rule	#Query	Integrity Constraint SPARQL query
A1	IC1	<pre>#Integrity constraint IC1 based on MLT Axiom A1 #This query checks scenarios where individuals have instances SELECT DISTINCT * WHERE { ?x rdf:type mlt:TokenIndividual . ?y rdf:type ?x }</pre>
D5	IC2	<pre>#Integrity constraint IC2 based on MLT Definition D5 #This query checks scenarios where a type t1 categorizes t2 and #t2 instantiates t1 SELECT DISTINCT * WHERE { ?t1 mlt:categorizes ?t2 . { ?t2 rdf:type ?t1 . }UNION{ ?t3 rdfs:subClassOf ?t2 . ?t2 owl:sameAs ?t3 . ?t3 rdf:type ?t1 . } }</pre>
T5	IC3	<pre>#Integrity constraint IC3 based on MLT Theorem T5 #This query checks scenarios where asymmetry is violated SELECT DISTINCT * WHERE { ?x rdf:type ?y . ?y rdf:type ?x }</pre>
T6	IC4	<pre>#Integrity constraint IC4 based on MLT Theorem T6 #This query checks scenarios where stratification is violated SELECT DISTINCT * WHERE { ?x rdf:type ?y . ?y rdf:type ?z . ?x rdf:type ?z }</pre>
T13	IC5	<pre>#Integrity constraint IC5 based on MLT Theorem T13 #This query checks scenarios where a type has more than one powertype SELECT DISTINCT * WHERE { ?p mlt:isPowertypeOf ?t . ?p1 mlt:isPowertypeOf ?t . FILTER (?p NOT IN (?p1)) }</pre>

MLT Rule	#Query	Integrity Constraint SPARQL query
T14	IC6	<pre>#Integrity constraint IC6 based on MLT Theorem T14 #This query checks scenarios where a type is powertype of more than one #type SELECT DISTINCT * WHERE { ?p mlt:isPowertypeOf ?t . ?p mlt:isPowertypeOf ?t1 . FILTER (?t NOT IN (?t1)) }</pre>
T18	IC7	<pre>#Integrity constraint IC7 based on MLT Theorem T18 #This query checks scenarios where a type t3 is partitioned by two #other types t1 and t2, and t1 is subclass of t2 SELECT DISTINCT * WHERE { ?t1 mlt:partitions ?t3 . ?t2 mlt:partitions ?t3 . ?t1 rdfs:subClassOf ?t2 }</pre>

4.4 MODEL COMPLETION BASED ON MLT

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 instance, it follows from the axioms of MLT that, instances of a basic higher-order type are entities whose instances are instances of a type at the immediately lower level. And the inverse is also valid, instances of a basic type are entities whose types are instances of a type at the immediately higher level. For example, query DR1 (Table 4-3) allows the identification of instances of *mlt:TokenIndividual* whose types are not represented as instances of *mlt:1stOrderClass*. While the query DR2 (Table 4-3) allow the identification of instances of *mlt:1stOrderClass* whose instances are not represented as instances of *mlt:TokenIndividual*. The same pattern occurs for the queries DR3 and DR4, and DR5 and DR6 (Table 4-3).

MLT also defines some structural relations which occur between types of the same order (intra-level relations) and between types of adjacent levels (cross-level relations). The first intra-level relation is the ordinary specialization between types, which defines that a type specializes other type, then every instance of the former is also instance of the last. The query DR7 (Table 4-3) allows the identification of cases in which the subclassing of two types is represented but the instances of the subtypes are not represented as instances of

the super type. Note that this derivation could be made using existent reasoners, since this rule is intrinsic of the definition of *rdfs:subClassOf*. However, since we deal with multiple levels of classification, we are not dealing with OWL-DL. Thus, the current reasoners cannot guarantee evaluation in a viable time.

According to the definition of the first cross-level relation, when *t1* is the *powertype* of *t2* and exists a *t3* which is instance of *t1*, then *t3* specializes *t1*. The query DR8 (Table 4-3) identifies cases in which *t1* is represented as the *powertype* of *t2* and *t3* is represented as instance of *t1*, but *t3* is not represented as subclass of *t1*. Even in the definition of the *powertype* of relation, if *t1* is the *powertype* of *t2* and *t3* specializes *t1*, then *t3* is an instance of *t1*. The query DR9 (Table 4-3) allows the identification of cases where the instantiation is not represented.

Now, the definition of the *categorize* relation states that all instances of a class must specialize other class. For that, query DR10 (Table 4-3) identifies cases when a type *t1* categorizes a type *t2* and the instances of *t1* are not represented as subclasses of *t2*. Further, this relation has two specializations: *completelyCategorizes* and *disjointlyCategorizes*. The queries DR11 and DR12 (Table 4-3) identify cases when these variations are represented but the *categorize* relation is not. Lastly, the *partitions* relation is a specialization of both *completelyCategorizes* and *disjointlyCategorizes*. Thus, the query DR13 (Table 4-3) looks for cases when the super relations are represented but the sub relation is not, and the query DR14 (Table 4-3) looks for cases when the sub relation is represented but the super relations are not.

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 DR15 (Table 4-3) 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. Even more, DR16 (Table 4-3) works inversely, it allows the identification of subclasses of *mlt:TokenIndividual* which are not represented as instances of *mlt:1stOrderClass*. The pattern occurs between the queries DR17 and DR18, and between DR19 and DR20.

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 powertypes (DR21 in Table 4-3).

Even more, according to MLT, if $t1$ is subordinate to $t2$ and $t2$ *categorizes* $t3$, then $t1$ *categorizes* $t3$. The query DR22 (Table 4-3) identifies cases where the subordination between $t1$ and $t2$, and the *categorization* between $t2$ and $t3$ occurs, but the *categorization* between $t1$ and $t3$ is not represented.

Finally, according to MLT, if $t2$ is powertype of $t1$ and $t3$ *categorizes* $t1$, then $t3$ is subclass of $t2$, we provide a SPARQL query to identify cases in which the *isPowertypeOf* and the *categorization* relations are represented but the *subclass* relations are not (DR23 in Table 4-3).

Table 4-3. Derivation Rules Corresponding to MLT Rules

MLT Rule	Query	Derivation SPARQL query
A3	DR1	<pre>#Derivation rule DR1 based on MLT Axiom A3 #This query derives instances of instances of 1stOrderClass #as instances of TokenIndividual CONSTRUCT { ?x rdf:type mlt:TokenIndividual }WHERE { ?t rdf:type mlt:1stOrderClass . ?x rdf:type ?t }</pre>
	DR2	<pre>#Derivation rule DR2 based on MLT Axiom A3 #This query derives types of instances of TokenIndividual #as instances of 1stOrderClass CONSTRUCT { ?t rdf:type mlt:1stOrderClass }WHERE { ?x rdf:type ?t . ?x rdf:type mlt:TokenIndividual . filter(?t != mlt:TokenIndividual) }</pre>
A4	DR3	<pre>#Derivation rule DR3 based on MLT Axiom A4 #This query derives instances of instances of 2ndOrderClass #as instances of 1stOrderClass CONSTRUCT { ?t1 rdf:type mlt:1stOrderClass }WHERE { ?t rdf:type mlt:2ndOrderClass . ?t1 rdf:type ?t }</pre>

MLT Rule	Query	Derivation SPARQL query
	DR4	<pre>#Derivation rule DR4 based on MLT Axiom A4 #This query derives types of instances of 1stOrderClass #as instances of 2ndOrderClass CONSTRUCT { ?t rdf:type mlt:2ndOrderClass }WHERE { ?t1 rdf:type ?t . ?t1 rdf:type mlt:1stOrderClass . filter(?t != mlt:TokenIndividual) }</pre>
A5	DR5	<pre>#Derivation rule DR3 based on MLT Axiom A4 #This query derives instances of instances of 3rdOrderClass #as instances of 2ndOrderClass CONSTRUCT { ?t1 rdf:type mlt:2ndOrderClass }WHERE { ?t rdf:type mlt:3rdOrderClass . ?t1 rdf:type ?t }</pre>
	DR6	<pre>#Derivation rule DR4 based on MLT Axiom A4 #This query derives types of instances of 2ndOrderClass #as instances of 3rdOrderClass CONSTRUCT { ?t rdf:type mlt:3rdOrderClass }WHERE { ?t1 rdf:type ?t . ?t1 rdf:type mlt:2ndOrderClass . filter(?t != mlt:TokenIndividual) }</pre>
D1	DR7	<pre>#Derivation rule DR7 based on MLT Definition D1 #This query derives as instances of t2 #the instances of t1 which is subclass of t2 CONSTRUCT { ?e rdf:type ?t2 }WHERE { ?t1 rdfs:subClassOf+ ?t2 . ?e rdf:type ?t1 }</pre>
D4	DR8	<pre>#Derivation rule DR8 based on MLT Definition D4 #This query derives as subclass of t2 #the instances of t1 which is powertype of t2 CONSTRUCT { ?t3 rdfs:subClassOf ?t2 }WHERE { ?t1 mlt:isPowertypeOf ?t2 . ?t3 rdf:type ?t1 . ?t1 rdf:type ?t1Type . filter(?t1Type != mlt:TokenIndividual) }</pre>
	DR9	<pre>#Derivation rule DR9 based on MLT Definition D4 #This query derives as instances of t1 #the subclasses of t2, in which t1 is powertype of t2 CONSTRUCT { ?t3 rdf:type ?t1 }WHERE { ?t1 mlt:isPowertypeOf ?t2 . ?t3 rdfs:subClassOf* ?t2 . ?t1 rdf:type ?t1Type . filter(?t1Type != mlt:TokenIndividual) }</pre>

MLT Rule	Query	Derivation SPARQL query
D5	DR10	<pre>#Derivation rule DR10 based on MLT Definition D5 #This query derives as subclasses of t2 #the instances of t1 which categorizes t2 CONSTRUCT { ?t3 rdfs:subClassOf ?t2 }WHERE { ?t1 mlt:categorizes ?t2 . ?t3 rdf:type ?t1 . ?t1 rdf:type ?t1Type . filter(?t1Type != mlt:TokenIndividual) }</pre>
D6	DR11	<pre>#Derivation rule DR11 based on MLT Definition D6 #If t1 completelyCategorizes t2 #this query derives that t1 categorizes t2 CONSTRUCT { ?t1 mlt:categorizes ?t2 }WHERE { ?t1 mlt:completelyCategorizes ?t2 }</pre>
D7	DR12	<pre>#Derivation rule DR12 based on MLT Definition D7 #if t1 disjointlyCategorizes t2 #this query derives that t1 categorizes t2 CONSTRUCT { ?t1 mlt:categorizes ?t2 }WHERE { ?t1 mlt:disjointlyCategorizes ?t2 }</pre>
D8	DR13	<pre>#Derivation rule DR13 based on MLT Definition D8 #if t1 partitions t2 #this query derives that #t1 completelyCategorizes t2 and that t1 disjointlyCategorizes t2 CONSTRUCT { ?t1 mlt:completelyCategorizes ?t2 . ?t1 mlt:disjointlyCategorizes ?t2 }WHERE { ?t1 mlt:partitions ?t2 }</pre>
	DR14	<pre>#Derivation rule DR14 based on MLT Definition D8 #if t1 completelyCategorizes t2 and that t1 disjointlyCategorizes t2 #this query derives that t1 partitions t2 CONSTRUCT { ?t1 mlt:partitions ?t2 }WHERE { ?t1 mlt:completelyCategorizes ?t2 . ?t1 mlt:disjointlyCategorizes ?t2 }</pre>
T7	DR15	<pre>#Derivation rule DR15 based on MLT Theorem T7 #if t is an instance of 1stOrderClass #this query derives that t is subclass of TokenIndividual CONSTRUCT { ?t rdfs:subClassOf mlt:TokenIndividual }WHERE { ?t rdf:type mlt:1stOrderClass }</pre>
	DR16	<pre>#Derivation rule DR16 based on MLT Theorem T7 #if t is a subclass of TokenIndividual #this query derives that t is an instance of 1stOrderClass CONSTRUCT { ?t rdf:type mlt:1stOrderClass }WHERE { ?t rdfs:subClassOf+ mlt:TokenIndividual }</pre>

MLT Rule	Query	Derivation SPARQL query
T8	DR17	<pre>#Derivation rule DR17 based on MLT Theorem T8 #if t is an instance of 2ndOrderClass #this query derives that t is subclass of 1stOrderClasss CONSTRUCT { ?t rdfs:subClassOf mlt:1stOrderClasss }WHERE { ?t rdf:type mlt:2ndOrderClass }</pre>
	DR18	<pre>#Derivation rule DR18 based on MLT Theorem T9 #if t is a subclass of 1stOrderClass #this query derives that t is an instance of 2ndOrderClass CONSTRUCT { ?t rdf:type mlt:2ndOrderClass }WHERE { ?t rdfs:subClassOf+ mlt:1stOrderClass }</pre>
T9	DR19	<pre>#Derivation rule DR19 based on MLT Theorem T9 #if t is an instance of 3rdOrderClass #this query derives that t is subclass of 2ndOrderClass CONSTRUCT { ?t rdfs:subClassOf mlt:2ndOrderClasss }WHERE { ?t rdf:type mlt:3rdOrderClass }</pre>
	DR20	<pre>#Derivation rule DR20 based on MLT Theorem T9 #if t is a subclass of 2ndOrderClass #this query derives that t is an instance of 3rdOrderClass CONSTRUCT { ?t rdf:type mlt:3rdOrderClass }WHERE { ?t rdfs:subClassOf+ mlt:2ndOrderClass }</pre>
T15	DR21	<pre>#Derivation rule DR21 based on MLT Theorem T15 #if t3 is powertype of t1, t4 is powertype of t2, #and that t2 is subclass of t1 #this query derives that t4 is subclass of t3 CONSTRUCT { ?t4 rdfs:subClassOf ?t3 }WHERE { ?t2 rdfs:subClassOf+ ?t1 . ?t4 mlt:isPowertypeOf ?t2 . ?t3 mlt:isPowertypeOf ?t1 }</pre>
T16	DR22	<pre>#Derivation rule DR22 based on MLT Theorem T16 #if t1 is subordinate to t2 and t2 categorizes t3 #this query derives that t1 categorizes t3 CONSTRUCT { ?t1 mlt:categorizes ?t3 }WHERE { ?t1 mlt:isSubordinateTo ?t2 . ?t2 mlt:categorizes ?t3 }</pre>
T17	DR23	<pre>#Derivation rule DR23 based on MLT Theorem T17 #if t2 is powertype of t1 and t3 categorizes t1 #this query derives that t3 is subclass of t2 CONSTRUCT { ?t3 rdfs:subClassOf ?t2 }WHERE { ?t2 mlt:isPowertypeOf ?t1 . ?t3 mlt:categorizes ?t1 }</pre>

4.5 IMPLEMENTATION

In order to facilitate the combined usage of the MLT vocabulary in OWL and the MLT rules in SPARQL (described in Sections 4.2-4.4), we developed an application that receives as input an OWL file describing a domain ontology. It detects violations of MLT rules and, in the case of valid models, it produces an OWL output file containing the original domain ontology plus derived information following MLT rules.

Our application is developed using Jena [41], which is a widely used Java API (Application Programming Interface) that provides support to manipulate RDF models (which includes RDFS and OWL) and to execute SPARQL queries into these models. For that, we used two bundles: `jena-core-2.10.1` and `arq-2.8.7`. The former is the core RDF API, provides mechanisms to handle RDF models and its triples. For example, with the core RDF API we are able to handle OWL's classes, properties and individuals. The latter, ARQ is a SPARQL Processor for Jena, i.e., a query engine for Jena that supports the SPARQL. Moreover, we use the reasoner Hermit [42] (version 1.3.8.4) in order to check MLT rules expressed in OWL, such as the disjointness of the MLT basic types (Theorem T4 in Table 2-1).

The application loads a domain model in order to execute the MLT rules. First, MLT rules implemented in OWL, such as Theorem T4 (Table 2-1) which is implemented through *owl:AllDisjointClasses*, are checked. Then, MLT integrity constraints are checked running SPARQL queries. If any inconsistency is found, a diagnosis is shown and the application ends. Further, knowledge is derived from the execution of MLT derivation rules also implemented in SPARQL. Derivation rules are executed repeatedly, until no new statements are generated. This is because knowledge derived in an iteration may trigger new derivations. MLT integrity constraints and rules implemented in OWL and in SPARQL are checked again in order to verify whether any inconsistent information was introduced during the execution of derivation rules. Again, if any consistency is found, a diagnosis is shown and the application ends. Finally, the enriched domain model is serialized. These steps are shown in the tool's flowchart presented in Figure 4-4.

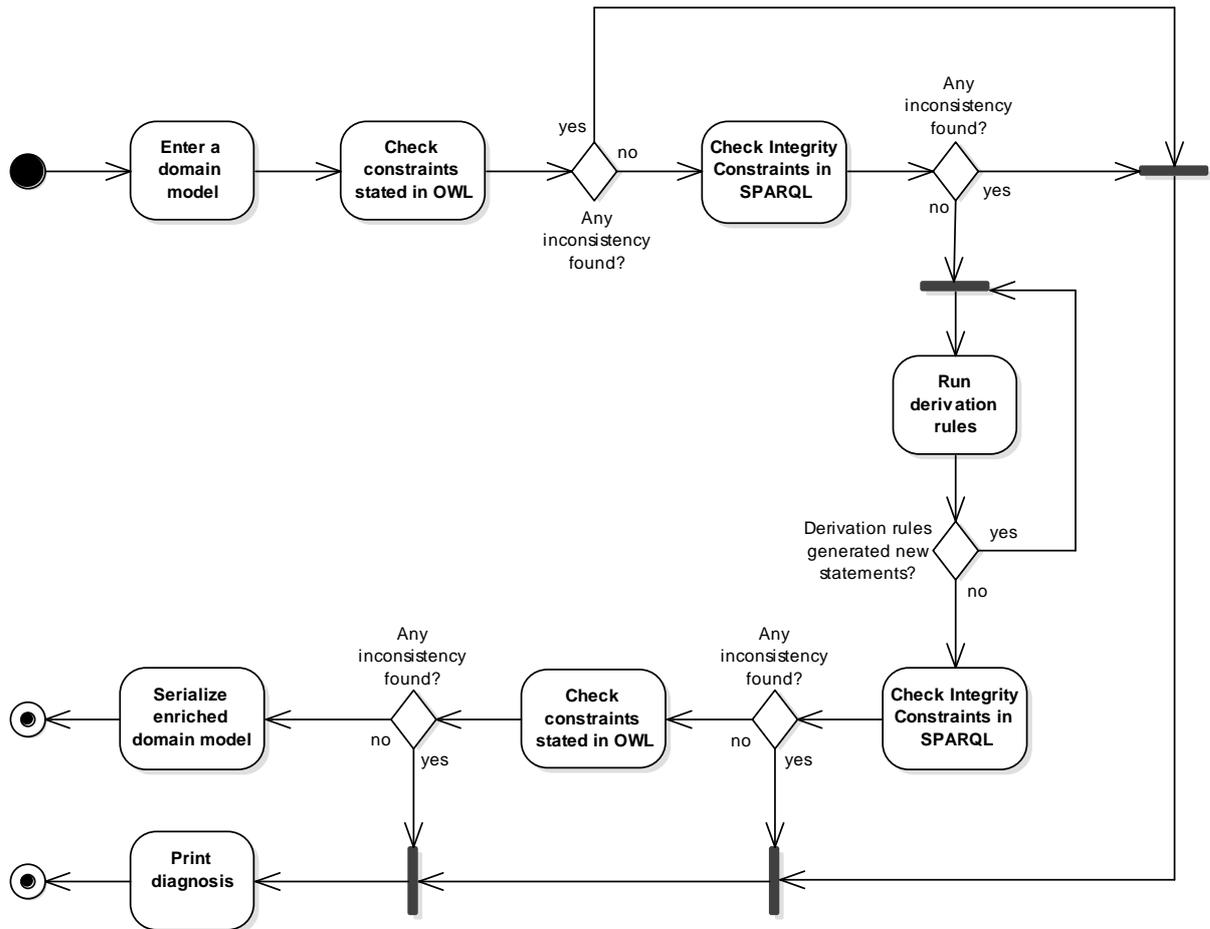


Figure 4-4. Tool's flowchart

It is key that these models have statements involving MLT, e.g. that a resource is an instance of or subclass of the MLT basic types. Otherwise, our application will check only Theorems T5 and T6 (Table 2-1) about strict metamodeling, which are independent of MLT's basic types and relations.

4.6 CONCLUDING REMARKS

Despite being the standard for representing vocabularies on the Web, OWL lacks expressivity for representing all MLT integrity constraints and derivation rules. Thus, only MLT basic types and relations, and a few set of theorems (T1-T4 and T10-T12) have been represented directly in OWL. Some of other MLT rules (A1, A3-A5, D1, D4-D8, T5-T9 and T13-T18) have been implemented using SPARQL, using SELECT and CONSTRUCT primitives.

A few rules could not be (fully- or partially-) implemented, neither in OWL or SPARQL, due to the Open World Assumption (OWA). This is the case for rules A2, A6 and D3, which could not be fully-implemented, and D1, D4-D7, which were only partially implemented. This is because since MLT is formalized quantifying over *all possible entities*, some MLT definitions are not expressible considering the OWA. For instance, according to MLT, two types are equal iff the sets of *all their possible instances* are the same (A2 in Table 2-1). Further, 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 2-1). Again, according to MLT, if *all possible instances* of a type $t1$ also specializes a type $t2$, then $t1$ is the powertype of $t2$ (D4). Analogously to Definitions D5, D6 and D7 from Table 2-1. These rules 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.

Since the notion of equality between types (defined in axiom A2, which cannot be implemented) is central to capture the difference between *specialization* and *proper specialization* (D2 in Table 2-1), it was not possible to reflect definition D2 in the implementation. Although, the notion of *proper specialization* is used in MLT definition of *categorization* (D5) to capture that the base type itself is not an instance of the higher-order type that *categorizes* it. This issue has given rise to integrity constraint IC2 presented in Section 4.3. Finally, the notion of *proper specialization* is also central to the definition of *subordination* (D3), which was not reflected in the implementation.

According to MLT, each entity in our domain of inquiry is necessarily an instance of exactly one of its basic types. However, due to OWA, we cannot assume that all instantiation relations are represented and, thus, it is not possible to capture this MLT rule as a constraint. When the information about the basic type instantiated by a domain entity is neither declared nor derived, a warning is given to the modeler.

Thus, to accomplish the requirements defined in Section 2.4 our approach defines an OWL vocabulary representing MLT and implements MLT rules in SPARQL. The choice for MLT is due to the fact MLT met these requirements. As discussed in Section 4.5, we developed this approach into an application, which receives an OWL model as input and

run MLT rules, in order to check consistency and to derive knowledge. With this application, we are now able to assess a real knowledge base for its conformance with MLT rules.

We have refrained from an annotation-based approach such as that of OWL FA [43] and PURO [12]). An annotation-based approach deviates from the standard use of *rdf:type* for representing instantiation, and a key aspect of the multi-level model ends up in annotations outside the model. Moreover, the OWL specification [4] recommends that metamodeling should be done through *punning*, while annotations should be used when the added information is not part of the domain (e.g. the date that John included an information). The use of *punning* makes a model incompatible with OWL-DL and unfortunately DL-reasoners do not support reasoning on cross-level constraints (*punning* included). Thus, it is not possible to run MLT rules through an existing DL-reasoner. Moreover, the contextual semantics [10] of OWL also makes impossible to represent MLT rules in SWRL [44] (Semantic Web Rule Language), since it is based on OWL DL. Therefore, considering our focus on supporting expressive multi-level modeling, since the DL-reasoners do not support the expressivity needed to represent the rules of MLT our approach relies on SPARQL queries to check constraints and to derive information.

5 VALIDATING THE APPROACH WITH TAXONOMIC HIERARCHIES FROM WIKIDATA

To provide empirical evidence of the applicability of MLT-OWL approach to assess real-world content dealing with multiple classification levels, in this chapter we apply this approach to assess the Wikidata content from the perspective of multi-level modeling. First, we present our considerations of semantic correspondence between Wikidata properties and MLT relations, and the problems found in the multi-level taxonomic hierarchies in Wikidata (Section 5.1). Then, we present some anti-patterns that occur in Wikidata and that violate the aforementioned stratification into classification levels (Section 5.2). Further, we show how the MLT approach can be used for avoiding these anti-patterns and how it can be used in valid ontologies (Section 5.3). Finally, we present some conclusions of the usage of MLT-OWL approach in Wikidata content (Section 5.4).

5.1 A DIAGNOSIS OF MULTI-LEVEL TAXONOMIC HIERARCHIES IN WIKIDATA

The definition of *instance of* provided in Wikidata is informal and silent about its formal logic properties (symmetry, reflexivity and transitivity). However, observing its use in Wikidata content, we have concluded that its purpose is similar to the *iof* relation of MLT: to denote that a type applies to an element. Therefore, in order to apply MLT to validate taxonomic hierarchies in Wikidata, we consider the semantics of its *instance of* property to correspond to that of the *iof* relation in MLT. Further, *subclass of* provided in Wikidata is characterized as transitive and asymmetric (antisymmetric and irreflexive). We consider the semantics of the *subclass of* property in Wikidata to correspond to that of the *proper specialization* relation in MLT. The establishment of the semantics of *instance of* and *subclass of* properties in terms of MLT allow us to use the MLT rules to validate Wikidata content.

Considering the chain of instantiations in Figure 3-7 (page 41) we can clearly detect a notion of levels: *Cecil*, *Organism*, *Taxon* and *Taxonomic Rank* are at different levels of classification. If we assume *Cecil* as an *instance of Individual* (of MLT), since we know that it

has no instances, we can apply the MLT basic pattern (Theorems T7, T8 and T9, Table 2-1) to deduce new information from the diagram in Figure 3-7. First, we can derive that *Panthera Leo* and all its super classes are both *instances of 1stOT* (from Axiom A3) and then *subclasses of Individual* (from Theorem T7). Consequently, the classifiers of *Organism* types (e.g., *Taxon*, *Domain*, *Species*) are both *instances of 2ndOT* (from Axiom A4) and then *subclasses of 1stOT* (from Theorem T8). Finally, *Taxonomic Rank* is derived as *instance of 3rdOT* (from Axiom A5) and then *subclass of 2ndOT* (from Theorem T9).

The example of biological taxonomic rank illustrated in Figure 3-7 conforms to the stratification underlying MLT rules, following its basic pattern. However, there is no automated support or guidelines to prevent a Wikidata contributor from violating this conformant structure. For example, a clearly incorrect modification introducing a new entity (e.g. “Simba”) which is both an instance of *Panthera Leo* and *Species* would go undetected, and would result in an inconsistent hierarchy. In fact, we have observed many occurrences of such problematic hierarchies in current Wikidata content.

For example, take Wikidata information about *Tim Berners-Lee* and his professional occupation (a fragment of which is depicted in Figure 5-1). *Tim* is considered *instance of Computer Scientist*. In its turn, *Computer Scientist* is indirectly *subclass of Profession*. Thus, we can conclude *Tim* is an *instance of Profession*(!), which clearly violates our sense of what a *Profession* is. Formally, these statements could be considered inconsistent in the light of MLT: since *instance of* is anti-transitive (Theorem T6, Table 2-1) and *Computer Scientist* is *instance of Profession*, *Tim* cannot be *instance of Profession*.

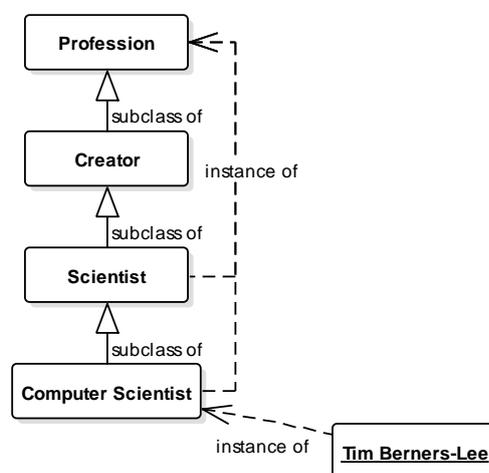


Figure 5-1. Wikidata information about Tim Berners-Lee and his professional occupation

Now, considering Tim Berners-Lee as an *Individual* (of MLT), since it has no instances, we can apply the MLT basic pattern to deduce information. First, we conclude that *Computer Scientist* and all its super classes are both *instances of 1stOT* (from Axiom A3) and *subclasses of Individual* (from Theorem T7). Consequently, since instances of *Profession* are *instances of 1stOT*, *Profession* is both *instance of 2ndOT* (from Axiom A4) and *subclass of 1stOT* (from Theorem T8). Here, we realize that *Profession* is *instance of both 1stOT and 2ndOT*, which is invalid by A6 (see Table 2-1).

We have observed similar problems concerning multiple levels of classification in other domains represented in Wikidata, such as transport, software and sports. In section 5.2, we present the results of some queries we have submitted to Wikidata in order to detect potential problematic scenarios. We highlight some issues identified and discuss them in the light of MLT.

5.2 APPROACH: DETECTION OF ANTI-PATTERNS

In order to obtain some indication of the use of multi-level hierarchies in Wikidata, we have queried for three simple cases of anti-patterns that violate the aforementioned stratification. Figure 5-2 illustrates Anti-Pattern 1 (AP1) that looks for pairs of items (A, Z) such that one (Z) is simultaneously a *subclass of* and an *instance of* the other (A). This anti-pattern can appear under many configurations, i.e., one (Z) can be a direct *subclass of* the other (A) or there may be a chain of *subclass of* properties between the involved items. The fragment illustrated in Figure 5-1 (concerning Tim Berners-Lee's professional occupation) includes two occurrences of this anti-pattern with chains of *subclass of* properties of length 2 and 3. Regardless of the size of this chain, the occurrence of this pattern prevents stratification into classification levels, and creates a formal contradiction: classes A and Z would be simultaneously at the same level (because they are related by specialization) and at adjacent levels (because they are related by instantiation). Table II-1 (Appendix II) shows the SPARQL query associated to AP1 that considers a transitive closure for *subclass of* statements. For this anti-pattern, we have found 14320 occurrences, covering many domains, such as software, sports, biology, food, profession.

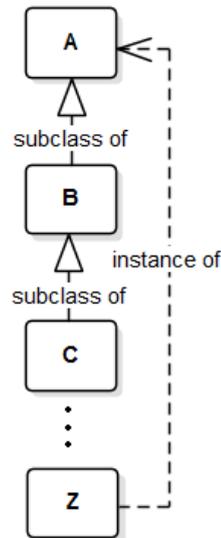


Figure 5-2. Illustration of Anti-Pattern 1

Figures Figure 5-3 and Figure 5-4 show examples of problematic fragments identified through Anti-Pattern 1. Figure 5-3 shows that *earthquake* (Q7944) is both *instance of* and *subclass of* *natural disaster* (Q8065). This fragment seems to have an unclear interpretation. Does the Wikidata contributor consider *earthquake* to be a *natural disaster* or a special type of *natural disaster*?

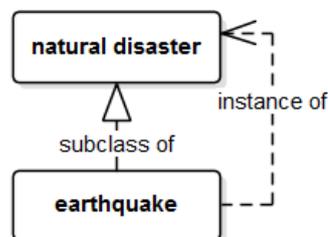


Figure 5-3. Scenario about earthquake found in Wikidata for AP1

This lack of clarity that results from the occurrence of AP1 has practical implications for the properties of the items involved. For instance, considering that instances of natural disaster are specific *events* (Q1190554), i.e., specific occurrences of *natural disasters*, then these instances may be represented as having a *point in time* feature (P585). For example, we can say that the *1985 Mexico City earthquake* took place on September 19th, 1985. However, since *earthquake* is also declared to be an *instance of* *natural disaster* and, thus, an *instance of* *event*, *earthquake* itself could also be associated to a *point in time*. Notice, however, that *earthquake* is more naturally thought of as a *subclass of* *natural disaster*, i.e., as a specific kind of *natural disaster*, and a specific kind of event. But, in this case, it would be

problematic to attribute a specific *point in time* to this particular class of events. So, in this example, it seems that the undesired relation is the *instance of* relation between *earthquake* and *natural disaster*.

Analogously, Figure 5-4 shows that *Egg waffle* (Q837620) is both *instance of* and indirectly a *subclass of food* (Q375). In this case, it is unclear whether an instance of *food*, *waffle* and *Egg waffle* would represent a particular portion of food (the egg waffle John had for breakfast), or a kind of food (such as *waffle* or *Egg waffle*).

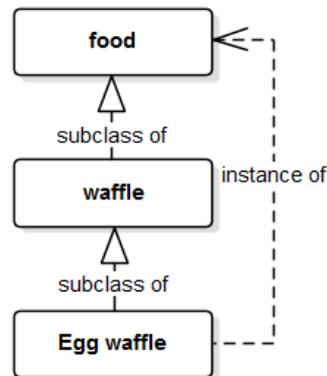


Figure 5-4. Scenario about egg waffle found in Wikidata for AP1

A second anti-pattern (AP2) is illustrated in Figure 5-5. In this case, we have that an item (C) has two direct super classes (A and B) such that one of the super classes is an instance of the other (B is instance of A). Similarly to AP1, the occurrences of AP2 present logical inconsistencies that rise from the violation of the stratification into classification levels. In this case, all *instances of C* are also *instances of A* and B. However, *instances of B* cannot be *instances of A*, since B is itself *instance of A*. Table II-1 (Appendix II) presents the SPARQL query that can be used to detect occurrences of AP2. By running this query, we have found 257 occurrences, covering domains, such as diseases, biology, food and colors.

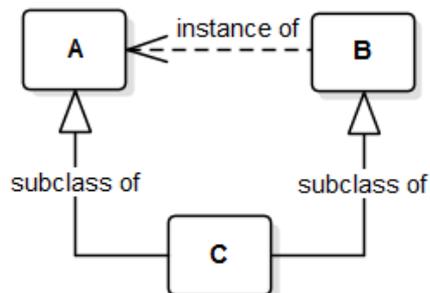


Figure 5-5. Illustration of Anti-Pattern 2

Figure 5-6 illustrates that *crawler excavator* (Q5182961) is declared to be a *subclass of* both *excavator* and *heavy equipment*, and *excavator* (Q182661) is an *instance of* *heavy equipment* (Q874311).

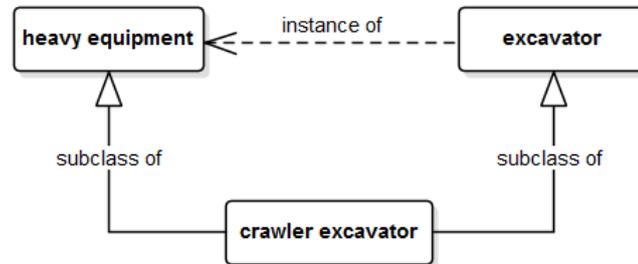


Figure 5-6. Scenario about excavator found in Wikidata for AP2

Finally, a third anti-pattern (AP3) is illustrated in Figure 5-7. This anti-pattern represents cases in which the anti-transitivity of the *instance of* relation is violated, making stratification unfeasible. In the case depicted in Figure 5-7, *C* would have to be simultaneously one and two classification levels below *A*. The query shown in Table II-1 (Appendix II) can be used to detect instances of AP3. By running it, we have found 6708 occurrences of AP3.

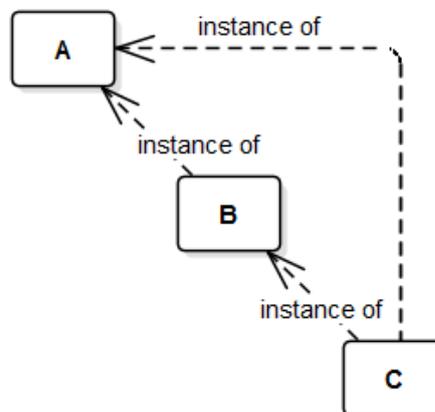


Figure 5-7. Illustration of Anti-Pattern 3

Figure 5-8 illustrates an example of AP3. *Central Park* (Q160409), the public park at the center of Manhattan in New York City, is considered an *instance of* both *urban park* (Q22746) and *park* (Q22698), while *urban park* is also an *instance of* *park*. As a subclass of *geographic location* (Q2221906), *park* defines the property *coordinate location* (P625). Thus, we could state that *Central Park* and *urban park* have values for this, which is plausible for *Central Park* that has coordinate location 40°46'57"N, 73°57'58"W. In this, it sounds

problematic to attribute a value of coordinate location to *urban park*. So, in this example, it seems that the undesired relation is the *instance of* relation between *urban park* and *park*, which should possibly be replaced by a *subclass of* relation. This anti-pattern often occurs in chains with terms such as: *award* (Q618779), *Chinese surname* (Q1093580), *family name* (Q101352), *Voivodeship road* (Q1259617), *Mikroregion* (Q11781066) and *natural region* (Q1970725).

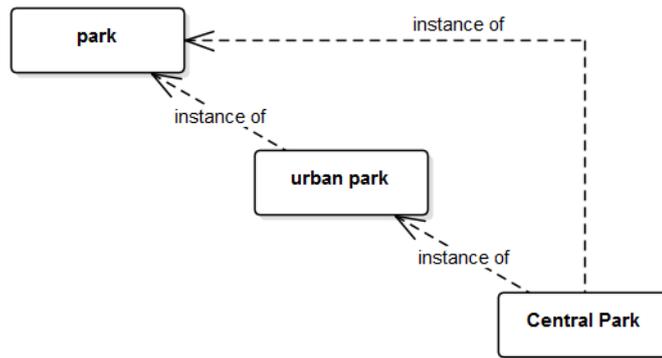


Figure 5-8. Scenario about urban park found in Wikidata for AP3

Table 5-1 summarizes the results we have obtained from our queries into the Wikidata simplified dump for AP1 and AP2. The total number of classes involved in taxonomic hierarchies is 337102. This number is obtained by counting the items that are either a subject or an object in “subclass of” statements. From this total number of items, 17819 classes are also the object of “instance of” statements, which means they are simultaneously classes and instances of other classes, and thus involved in hierarchies spanning more than one level of classification (our target classes for this investigation). From this number of classes, we have found 15177 classes involved in AP1 (85%) and 441 classes involved in AP2 (2.5%). Thus, a significant percentage of the classes involved in hierarchies spanning more than one level of classification violate the stratification of classification levels.

Table 5-1. Results for AP1 and AP2

Number of classes in any taxonomic hierarchy	337,102
Number of classes in taxonomic hierarchies spanning more than one level	17,819
Number of classes involved in AP1	15,177
Number of classes involved in AP2	441

Table 5-2 summarizes the results for AP3. Here, we contrast: (i) the total number of items in chains of instantiation with three levels (items A, B, and C, where C is an instance of B and B is an instance of A) with: (ii) the number of those items in occurrences of AP3, in which the third item in the chain (C) is also an instance of the first item in the chain (A), violating the stratification. Only 0.1% of the items that occur in these instantiation chains violate the stratification. The relatively low number of occurrences of this anti-pattern when contrasted with AP1 and AP2 corroborates our intuition that it is the combined use of sub classification and instantiation (a characteristic of AP1 and AP2) that is most challenging to Wikidata contributors.

Table 5-2. Results for AP3

Number of items in chains of instantiation with three items	6,963,059
Number of items in AP3 in these chains	7,082

5.3 APPLYING MLT RULES TO WIKIDATA CONTENT

Here, we aim to show how the proposed MLT-OWL approach is capable of preventing a number of issues found in Wikidata multi-level taxonomical hierarchies. As reported in Section 5.2, there are over 22,000 occurrences of three anti-patterns related to multi-level modeling in current Wikidata content; all these occurrences would have been prevented by using MLT-OWL. We show how each of three anti-patterns can be detected using MLT-OWL. In addition, we demonstrate how MLT-OWL approach could be used to complete valid models using fragments of Wikidata content.

The full set of integrity constraints and derivation rules, and the MLT vocabulary (described in Section 4.1) are combined in a tool, described in section 4.5. The tool receives as input an OWL file describing a domain ontology. It detects violations of MLT rules and, in the case of valid models, it produces an OWL output file containing the original domain ontology plus derived information following MLT rules.

The following subsections are organized as follows: subsection 5.3.1 shows how the problems detected in the example ontologies extracted from Wikidata content (presented

in Section 5.2) could be avoided through the analyze of MLT-OWL tool (which detects the multi-level modeling problems); and subsection 5.3.2 presents other examples in which the tool is used to derive additional information, based on the MLT rules, from valid domain ontologies.

5.3.1 Examples on Avoiding Anti-Patterns

Tim Berners-Lee's profession

In Figure 5-1 we present the example about *Tim Berners-Lee* profession, which is referent to AP1. As we said, to make MLT rules useful, at least one entity of the domain model must be *instance of* or *subclass of* the MLT basic types (otherwise, only IC3 and IC4 will be checked). Then, in addition to the information presented in the example, we added the information that *Tim Berners-Lee* is an instance of *mlt:TokenIndividual*, since we know that he is a concrete individual and he has no instances. This example is serialized in the file named *tim-berners-lee-profession.owl* and it is available at [40] and at Appendix III. Selecting the file containing this example, our implementation runs the MLT derivation rules and it shows the messages presented in Figure 5-9.

```

DERIVATION MESSAGES
Derived using DR2 (A3): [ComputerScientist, rdf:type, mlt:1stOrderClass]
Derived using DR2 (A3): [Scientist, rdf:type, mlt:1stOrderClass]
Derived using DR2 (A3): [Profession, rdf:type, mlt:1stOrderClass]
Derived using DR2 (A3): [Creator, rdf:type, mlt:1stOrderClass]
Derived using DR1 (A3): [Scientist, rdf:type, mlt:TokenIndividual]
Derived using DR1 (A3): [ComputerScientist, rdf:type, mlt:TokenIndividual]
Derived using DR4 (A5): [Profession, rdf:type, mlt:2ndOrderClass]
Derived using DR16 (T7): [Creator, rdfs:subClassOf, mlt:TokenIndividual]
Derived using DR16 (T7): [Profession, rdfs:subClassOf, mlt:TokenIndividual]
Derived using DR16 (T7): [Scientist, rdfs:subClassOf, mlt:TokenIndividual]
Derived using DR16 (T7): [ComputerScientist, rdfs:subClassOf, mlt:TokenIndividual]
Derived using DR3 (A4): [TimBernersLee, rdf:type, mlt:1stOrderClass]
Derived using DR18 (T9): [Profession, rdfs:subClassOf, mlt:1stOrderClass]
Derived using DR4 (A5): [ComputerScientist, rdf:type, mlt:2ndOrderClass]
Derived using DR4 (A5): [Creator, rdf:type, mlt:2ndOrderClass]
Derived using DR4 (A5): [Scientist, rdf:type, mlt:2ndOrderClass]
Derived using DR16 (T7): [TimBernersLee, rdfs:subClassOf, mlt:TokenIndividual]
Derived using DR6 (A5): [Profession, rdf:type, mlt:3rdOrderClass]
Derived using DR5 (A5): [TimBernersLee, rdf:type, mlt:2ndOrderClass]
Derived using DR20 (T10): [Profession, rdfs:subClassOf, mlt:2ndOrderClass]
Derived using DR6 (A5): [ComputerScientist, rdf:type, mlt:3rdOrderClass]
Derived using DR6 (A5): [Scientist, rdf:type, mlt:3rdOrderClass]
Derived using DR6 (A5): [Creator, rdf:type, mlt:3rdOrderClass]
Derived using DR18 (T9): [TimBernersLee, rdfs:subClassOf, mlt:1stOrderClass]

```

Figure 5-9. Derivation messages for the example of Tim Berners-Lee profession from Wikidata

Further, our implementation runs the MLT integrity constraints and it shows the messages presented in Figure 5-10. Note that there are occurrences of inconsistency from three integrity constraints. In Figure 5-9, we have from DR2 (A3) that *ComputerScientist* and *Profession* are derived as instances of *mlt:1stOrderClass*, since they have a *mlt:TokenIndividual* as its instance (*Tim Berners-Lee*). However, then, we have from DR1 (A3) that *ComputerScientist* is an instance of *mlt:TokenIndividual*, since it is an instance of *Profession* (which is an instance of *mlt:1stOrderClass*). This scenario is not allowed according to Integrity Constraint IC1 (Axiom A1), where instances of *mlt:TokenIndividual* cannot have instances.

In the second occurrence of inconsistency, the message shows that this model violates IC4 (T6), about stratification. Note that *Tim* is an instance of *Computer Scientist* and, transitively, he is also instance of *Profession*. However, *Computer Science* also is instance of *Profession*. Then, IC4 is useful to avoid AP3.

Finally, this model also violates Theorem T4. This is due to the fact that the basic types of MLT are disjoint (Theorem T4). Thus, since *ComputerScientist*, *Scientist*, *Profession* and *Creator*

are derived as instances of more than one MLT basic type, this model is inconsistent according to MLT.

```

Inconsistencies by IC1 (A1)
  [ComputerScientist, rdf:type, mlt:TokenIndividual]
  [TimBernersLee, rdf:type, ComputerScientist]

  [Scientist, rdf:type, mlt:TokenIndividual]
  [TimBernersLee, rdf:type, Scientist]

Inconsistencies by IC4 (T6)
  [TimBernersLee, rdf:type, ComputerScientist]
  [ComputerScientist, rdf:type, Profession]
  [TimBernersLee, rdf:type, Profession]

  [TimBernersLee, rdf:type, Scientist]
  [Scientist, rdf:type, Profession]
  [TimBernersLee, rdf:type, Profession]

Inconsistencies by Theorem T4
  [ComputerScientist, rdf:type, http://www.nemo.inf.ufes.br/mlt#3rdOrderClass]
  [ComputerScientist, rdf:type, http://www.nemo.inf.ufes.br/mlt#2ndOrderClass]
  [ComputerScientist, rdf:type, http://www.nemo.inf.ufes.br/mlt#TokenIndividual]
  [ComputerScientist, rdf:type, http://www.nemo.inf.ufes.br/mlt#1stOrderClass]

  [Scientist, rdf:type, http://www.nemo.inf.ufes.br/mlt#3rdOrderClass]
  [Scientist, rdf:type, http://www.nemo.inf.ufes.br/mlt#2ndOrderClass]
  [Scientist, rdf:type, http://www.nemo.inf.ufes.br/mlt#TokenIndividual]
  [Scientist, rdf:type, http://www.nemo.inf.ufes.br/mlt#1stOrderClass]

  [TimBernersLee, rdf:type, http://www.nemo.inf.ufes.br/mlt#2ndOrderClass]
  [TimBernersLee, rdf:type, http://www.nemo.inf.ufes.br/mlt#1stOrderClass]
  [TimBernersLee, rdf:type, http://www.nemo.inf.ufes.br/mlt#TokenIndividual]

  [Profession, rdf:type, http://www.nemo.inf.ufes.br/mlt#3rdOrderClass]
  [Profession, rdf:type, http://www.nemo.inf.ufes.br/mlt#2ndOrderClass]
  [Profession, rdf:type, http://www.nemo.inf.ufes.br/mlt#1stOrderClass]

  [Creator, rdf:type, http://www.nemo.inf.ufes.br/mlt#3rdOrderClass]
  [Creator, rdf:type, http://www.nemo.inf.ufes.br/mlt#2ndOrderClass]
  [Creator, rdf:type, http://www.nemo.inf.ufes.br/mlt#1stOrderClass]

```

Figure 5-10. Inconsistency messages for the example of Tim Berners-Lee profession from Wikidata

Earthquakes

In Figure 5-4 we present the example about *earthquakes*, which is referent to AP1. In addition to the information presented in the example, we added the information that *earthquake* is an instance of *mlt:1stOrderClass*. This example is serialized in the file named *earthquakes.owl* and it is available at [40] and at Appendix III. Selecting the file containing this example, our implementation shows the messages presented in Figure 5-11. Note that

there are occurrences of inconsistency from one integrity constraint. Since *earthquake* and *natural disaster* are derived as instances of more than one MLT basic type, this model violates Theorem T4 and it is inconsistent according to MLT.

```

DERIVATION MESSAGES
Derived using DR4 (A5): [natural_disaster, rdf:type, mlt:2ndOrderClass]
Derived using DR16 (T7): [earthquake, rdfs:subClassOf, mlt:TokenIndividual]
Derived using DR18 (T9): [natural_disaster, rdfs:subClassOf, mlt:1stOrderClass]
Derived using DR17 (T8): [earthquake, rdf:type, mlt:2ndOrderClass]
Derived using DR6 (A5): [natural_disaster, rdf:type, mlt:3rdOrderClass]
Derived using DR20 (T10): [natural_disaster, rdfs:subClassOf, mlt:2ndOrderClass]
Derived using DR19 (T9): [earthquake, rdf:type, mlt:3rdOrderClass]

Inconsistencies by Theorem T4
    [earthquake, rdf:type, http://www.nemo.inf.ufes.br/mlt#3rdOrderClass]
    [earthquake, rdf:type, http://www.nemo.inf.ufes.br/mlt#2ndOrderClass]
    [earthquake, rdf:type, http://www.nemo.inf.ufes.br/mlt#1stOrderClass]

    [natural_disaster, rdf:type, http://www.nemo.inf.ufes.br/mlt#3rdOrderClass]
    [natural_disaster, rdf:type, http://www.nemo.inf.ufes.br/mlt#2ndOrderClass]

```

Figure 5-11. Derivation messages for the example of earthquakes from Wikidata

Excavator

In Figure 5-6 we present the example about *excavator*, which is referent to AP2. In addition to the information presented in the example, we added the information *crawler excavator* is an instance of *mlt:1stOrderClass*. This example is serialized in the file *excavator.owl* and it is available at [40] and at Appendix III. Selecting the file containing this example, our implementation shows the messages presented in Figure 5-12. Note that there are occurrences of inconsistency from one integrity constraint. Since *crawler excavator* is derived as instance of more than one MLT basic type, this model violates Theorem T4 and it is inconsistent according to MLT.

```

DERIVATION MESSAGES
Derived using DR4 (A5): [heavy_equipment, rdf:type, mlt:2ndOrderClass]
Derived using DR16 (T7): [excavator, rdfs:subClassOf, mlt:TokenIndividual]
Derived using DR15 (T7): [crawler_excavator, rdf:type, mlt:1stOrderClass]
Derived using DR18 (T9): [heavy_equipment, rdfs:subClassOf, mlt:1stOrderClass]
Derived using DR17 (T8): [crawler_excavator, rdf:type, mlt:2ndOrderClass]

Inconsistencies by Theorem T4
    [crawler_excavator, rdf:type, http://www.nemo.inf.ufes.br/mlt#2ndOrderClass]
    [crawler_excavator, rdf:type, http://www.nemo.inf.ufes.br/mlt#1stOrderClass]

```

Figure 5-12. Derivation messages for the example of excavators from Wikidata

Urban Park

In Figure 5-8 we present the example about the *Central Park*, which is referent to AP3. In addition to the information presented in the example, we added the information that *Central Park* is an instance of *mlt:TokenIndividual*. This example is serialized in the file named *urban-park.owl* and it is available at [40] and at Appendix III. Selecting the file containing this example, our implementation runs the MLT derivation rules and it shows the messages presented in Figure 5-13.

```

DERIVATION MESSAGES
Derived using DR2 (A3): [urban_park, rdf:type, mlt:1stOrderClass]
Derived using DR2 (A3): [park, rdf:type, mlt:1stOrderClass]
Derived using DR1 (A3): [urban_park, rdf:type, mlt:TokenIndividual]
Derived using DR4 (A5): [park, rdf:type, mlt:2ndOrderClass]
Derived using DR16 (T7): [park, rdfs:subClassOf, mlt:TokenIndividual]
Derived using DR16 (T7): [urban_park, rdfs:subClassOf, mlt:TokenIndividual]
Derived using DR3 (A4): [CentralPark, rdf:type, mlt:1stOrderClass]
Derived using DR18 (T9): [park, rdfs:subClassOf, mlt:1stOrderClass]
Derived using DR4 (A5): [urban_park, rdf:type, mlt:2ndOrderClass]
Derived using DR16 (T7): [CentralPark, rdfs:subClassOf, mlt:TokenIndividual]
Derived using DR6 (A5): [park, rdf:type, mlt:3rdOrderClass]
Derived using DR18 (T9): [urban_park, rdfs:subClassOf, mlt:1stOrderClass]
Derived using DR5 (A5): [CentralPark, rdf:type, mlt:2ndOrderClass]
Derived using DR20 (T10): [park, rdfs:subClassOf, mlt:2ndOrderClass]
Derived using DR6 (A5): [urban_park, rdf:type, mlt:3rdOrderClass]
Derived using DR18 (T9): [CentralPark, rdfs:subClassOf, mlt:1stOrderClass]
Derived using DR20 (T10): [urban_park, rdfs:subClassOf, mlt:2ndOrderClass]

```

Figure 5-13. Derivation messages for the example of urban parks from Wikidata

Further, our implementation runs the MLT integrity constraints and it shows the messages presented in Figure 5-14. Note that there are occurrences of inconsistency from

three integrity constraints. We have, in Figure 5-13, that from DR2 (A3) that *urban park* and *park* are derived as instances of *mlt:1stOrderClass*, since they have a *mlt:TokenIndividual* as its instance (*Central Park*). However, from DR1 (A3) we have that *urban park* is an instance of *mlt:TokenIndividual*, since it is an instance of *park*. This scenario is not allowed according to IC (A1), where instances of *mlt:TokenIndividual* cannot have instances.

In the second occurrence of inconsistency, Figure 5-14 shows that this model violates IC4 (T6), about stratification. Note that *Central Park* is an instance of both *park* and *urban park*, and *urban park* is also an instance of *park*.

Finally, this model also violates Theorem T4, this is due to the fact that the basic types of MLT are disjoint. Thus, since *urban park*, *Central Park* and *park* are derived as instances of more than one MLT basic type, this model is inconsistent according to MLT

```

Inconsistencies by IC1 (A1)
  [urban_park, rdf:type, mlt:TokenIndividual]
  [CentralPark, rdf:type, urban_park]

Inconsistencies by IC4 (T6)
  [CentralPark, rdf:type, urban_park]
  [urban_park, rdf:type, park]
  [CentralPark, rdf:type, park]

Inconsistencies by Theorem T4
  [urban_park, rdf:type, http://www.nemo.inf.ufes.br/mlt#3rdOrderClass]
  [urban_park, rdf:type, http://www.nemo.inf.ufes.br/mlt#TokenIndividual]
  [urban_park, rdf:type, http://www.nemo.inf.ufes.br/mlt#1stOrderClass]
  [urban_park, rdf:type, http://www.nemo.inf.ufes.br/mlt#2ndOrderClass]

  [CentralPark, rdf:type, http://www.nemo.inf.ufes.br/mlt#2ndOrderClass]
  [CentralPark, rdf:type, http://www.nemo.inf.ufes.br/mlt#1stOrderClass]
  [CentralPark, rdf:type, http://www.nemo.inf.ufes.br/mlt#TokenIndividual]

  [park, rdf:type, http://www.nemo.inf.ufes.br/mlt#3rdOrderClass]
  [park, rdf:type, http://www.nemo.inf.ufes.br/mlt#2ndOrderClass]
  [park, rdf:type, http://www.nemo.inf.ufes.br/mlt#1stOrderClass]

```

Figure 5-14. Inconsistency messages for the example of urban parks from Wikidata

5.3.2 Example on Deriving Data from Valid Models

In Figure 3-7, we present the example about *Cecil* (the lion) which is available at the Wikidata content. For this example, we introduced the information that *Cecil* is an

instance of *mlt:TokenIndividual*, since we know that he is a concrete individual and he has no instances. Moreover, we also introduced in this example the MLT structural relations *isSubordinateTo* and *partitions*, as we exemplified in Figure 4-2. This example is serialized in the file named *biological-taxonomy.owl* and it is available at [40] and at Appendix III. Selecting the file containing this example, our implementation shows the messages presented in Figure 5-15.

Note that, as said in Section 4.1, this is an example of a valid model. From the information that *Cecil* is an instance of *mlt:TokenIndividual*, all other entities are derived as *subclasses of* or *instances of* MLT basic types. Further, from D8, which defines *mlt:partitions*, the tool derives that *PantheraSpecies* is related to *Panthera* through both *mlt:completelyCategorizes* and *mlt:disjointlyCategorizes*.

```

DERIVATION MESSAGES
Derived using DR2 (A3): [PantheraLeo, rdf:type, mlt:1stOrderClass]
Derived using DR2 (A3): [Organism, rdf:type, mlt:1stOrderClass]
Derived using DR2 (A3): [Panthera, rdf:type, mlt:1stOrderClass]
Derived using DR13 (D8): [PantheraSpecies, mlt:completelyCharacterizes, Panthera]
Derived using DR13 (D8): [PantheraSpecies, mlt:disjointlyCharacterizes, Panthera]
Derived using DR4 (A5): [Species, rdf:type, mlt:2ndOrderClass]
Derived using DR11 (D6): [PantheraSpecies, mlt:characterizes, Panthera]
Derived using DR16 (T7): [Panthera, rdfs:subClassOf, mlt:TokenIndividual]
Derived using DR16 (T7): [Organism, rdfs:subClassOf, mlt:TokenIndividual]
Derived using DR16 (T7): [PantheraLeo, rdfs:subClassOf, mlt:TokenIndividual]
Derived using DR6 (A5): [TaxonomicRank, rdf:type, mlt:3rdOrderClass]
Derived using DR15 (T7): [PantheraTigris, rdf:type, mlt:1stOrderClass]
Derived using DR15 (T7): [PantheraOnca, rdf:type, mlt:1stOrderClass]
Derived using DR18 (T9): [Species, rdfs:subClassOf, mlt:1stOrderClass]
Derived using DR5 (A5): [Genus, rdf:type, mlt:2ndOrderClass]
Derived using DR17 (T8): [PantheraSpecies, rdf:type, mlt:2ndOrderClass]
Derived using DR20 (T10): [TaxonomicRank, rdfs:subClassOf, mlt:2ndOrderClass]
Derived using DR18 (T9): [Genus, rdfs:subClassOf, mlt:1stOrderClass]

```

Figure 5-15. Derivation messages for the example of Biological Taxonomy with MLT relations

5.4 CONCLUDING REMARKS

We have analyzed Wikidata content from the perspective of multi-level modeling. We have observed a number of occurrences of violations of the stratification of levels in Wikidata, which indicate that some support for multi-level modeling could be beneficial in order to support contributors in the collaborative creation of multi-level taxonomies.

The queries we have used are the first step in automating this support. In addition to identifying possible problematic occurrences, we understand that more methodological guidance is required for contributors to understand the challenges in multi-level taxonomies and in particular to distinguish clearly between instantiation and specialization.

Future work is required in order to assess whether the items of “class” (Q16889133) and “metaclass” (Q19361238) could be used to provide more explicit support for multi-level modeling in Wikidata. In any case, we have found that these items are rarely employed, and that they seem limited by a three level system (instances, class and metaclass). In the biological taxonomy domain, we see that a fourth level is required (where “Taxonomic Rank” lies).

6 CONCLUSIONS

In this chapter, we present the conclusions of this thesis. In particular, we discuss our contributions (Section 6.1), the limitations of our approach (Section 6.2) and future work pointing directions for improvements of this research (Section 6.3).

6.1 CONTRIBUTIONS

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 [20]. 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] (discussed in Chapter 5), 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 Chapter 5, we also provide a technical report showing how each of these anti-patterns found in Wikidata could be avoided by using the artifact proposed in this thesis, 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 thesis, namely, RDFS, RDFS(FA), metamodeling (punning) in OWL, OWL FA, PURO and Wikidata. 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.

6.2 LIMITATIONS

Currently, our approach only checks MLT rules in batch style. Alternately, some of the rules could be checked in real-time if embedded properly in a modeling environment. A real-time verification strategy could bring some benefits to both the usability and performance of the approach. Concerning the usability, if the problems are identified at the moment they arise it is easier to the user to understand and fix them. Further, the strategy of facing the users with the errors in real-time could help them to learn the MLT rules, avoiding thus the recurrence of errors. Concerning performance issues, the execution time and the amount of memory required to execute queries drastically increases according to the number of triples in the dataset. This problem is highlighted in queries using transitive closure (such as DR9, DR18, DR20 and DR21 in Table 4-3). Using a real-time strategy, we could verify the impact of an inserted statement considering only the portion of data affected by it, avoiding the need of a complete verification. Applying this strategy, the execution time of the complete verification would be split and the required amount of memory would decrease. Thus, a natural extension to this work could be to implement a real-time strategy to check MLT rules. Such strategy could be implemented embedding the MLT rules in a Semantic Web modeling environment, such

as Protégé [45] or Wikidata [15], or implementing from scratch a new modeling environment which incorporates these rules.

6.3 FUTURE WORK

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) [30]. 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 ([46], [47]). In order to leverage the benefits of both a foundational ontology and a multi-level modeling theory, in [48] Guizzardi et al. have already combined MLT and the foundational ontology UFO [22]. 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). Incorporating UFO in our approach would also lead to the need of incorporating MLT aspects in the current transformations from OntoUML (a language for ontology-driven conceptual modeling based on UFO) to OWL (such as [49], [50]).

Moreover, this work could be extended to cover the MLT accounts for attributes and relationships reflecting a mechanism that has been called “deep instantiation” in the multi-level modeling literature. An attribute such as the height of a person or a relation of friendship in a social network, has effects only at the immediately lower level, complying to what has been called “shallow instantiation” [8]. MLT intends to cover attributes and relationships of higher-order types that aim at capturing regularities over *instances of its instances*. Following [48] these attributes are classified as *regularity attributes*. For example, consider we would like to state that all instances of *iPhone 5* must have *screen size* equals to 4 inches. Considering that *Mobile Phone Model* defines the attribute *model screen size* and its instance *iPhone 5* defines the attribute *screen size*, we could state that the attribute *model screen size* somehow constrains the attribute *screen size*. Thus, if *iPhone 5* has *model screen size*

equals to 4 inches, it means that all instances of *iPhone 5* must have *screen size* equals to 4 inches. The representation of regularity attributes, and the constraints that arise from the use of regularity attributes are not yet addressed in the present work.

Finally, since we rely minimally on inferential mechanisms for OWL, the approach can be easily applied to RDF(S). In that case, the MLT basic types can be represented as instances of *rdfs:Class* (instead of *owl:Class*) and MLT structural relations can be represented as instances of *rdf:Property* (instead of *owl:ObjectProperty*). We intend to report on a version of the vocabulary for RDF(S) in order to benefit a wider range of users.

BIBLIOGRAPHY

- [1] W3C, “W3C Semantic Web Activity.” [Online]. Available: <https://www.w3.org/2001/sw/>. [Accessed: 11-Jan-2016].
- [2] C. Bizer, T. Heath, and T. Berners-Lee, “Linked Data - The Story So Far,” *Int. J. Semant. Web Inf. Syst.*, vol. 5, no. 3, pp. 1–22, Jan. 2009.
- [3] W3C, “RDF Schema 1.1.” W3C Recommendation, 2014.
- [4] W3C, “OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (Second Edition).” W3C Recommendation, 2012.
- [5] E. Mayr, *The Growth of Biological Thought: Diversity, Evolution, and Inheritance*. Harvard University Press, 1982.
- [6] C. Gonzalez-Perez and B. Henderson-Sellers, “A powertype-based metamodelling framework,” *Softw. Syst. Model.*, vol. 5, no. 1, pp. 72–90, 2006.
- [7] B. Neumayr, K. Grun, and M. Schrefl, “Multi-level domain modeling with m-objects and m-relationships,” in *6th Asia-Pacific Conference on Conceptual Modelling*, 2009.
- [8] C. Atkinson and T. Kühne, “The Essence of Multilevel Metamodeling,” in *4th International Conf. on the Unified Modeling Language*, 2001.
- [9] J. Z. Pan and I. Horrocks, “Metamodeling Architecture of Web Ontology Languages,” in *Proc. of the 2001 Int. Semantic Web Working Symposium*, 2001, pp. 131–149.
- [10] B. Motik, “On the Properties of Metamodeling in OWL,” *J. Log. Comput.*, vol. 17, no. 4, pp. 617–637, 2007.
- [11] J. Z. Pan, I. Horrocks, and G. Schreiber, “OWL FA: A metamodeling extension of OWL DL,” in *Proc. of the 15th Intl Conf. on World Wide Web*, 2006, pp. 1065–1066.
- [12] V. Svatek, M. Homola, J. Kluka, and M. Vacura, “Metamodeling-Based Coherence Checking of OWL Vocabulary Background Models,” in *Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED 2013)*, 2013, vol. 1080.
- [13] G. Gröner, N. Jekjantuk, T. Walter, F. S. Parreiras, and J. Z. Pan, “Metamodelling and Ontologies,” in *Ontology-Driven Software Development*, 1st ed., 2013, pp. 151–174.
- [14] F. Brasileiro, J. P. A. Almeida, V. A. Carvalho, and G. Guizzardi, “Applying a Multi-Level Modeling Theory to Assess Taxonomic Hierarchies in Wikidata,” in *Proceedings of the 25th International Conference Companion on World Wide Web*, 2016, pp.

- 975–980.
- [15] D. Vrandečić and M. Krötzsch, “Wikidata: A Free Collaborative Knowledgebase,” in *Communications of the ACM*, 2014, vol. 57, no. 10, pp. 78–85.
 - [16] “DBpedia.” [Online]. Available: <http://dbpedia.org>.
 - [17] J. Odell, “Power Types,” *J. Object-Oriented Programming*, vol. 7, no. 2, pp. 8–12, 1994.
 - [18] L. Cardelli, “Structural subtyping and the notion of power type,” in *Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages - POPL ’88*, 1988, pp. 70–79.
 - [19] C. Atkinson and T. Kühne, “Meta-level Independent Modelling,” in *Intl. Workshop on Model Eng. at 14th European Conf. on Object-Oriented Programming*, 2000, pp. 1–4.
 - [20] V. A. Carvalho and J. P. A. Almeida, “Toward a well-founded theory for multi-level conceptual modeling,” *Softw. Syst. Model.*, pp. 1–27, Jun. 2016.
 - [21] R. Weber, *Ontological Foundations of Information Systems*. Coopers & Lybrand, 1997.
 - [22] G. Guizzardi, *Ontological foundations for structural conceptual models*. Enschede: Telematica Instituut Fundamental Research Series, 2005.
 - [23] W3C, “SPARQL 1.1 Query Language.” W3C Recommendation, 2013.
 - [24] C. Atkinson, R. Gerbig, and T. Kühne, “Comparing Multi-Level Modeling Approaches,” in *Proceedings of the 1st International Workshop on Multi-Level Modelling*, 2014.
 - [25] OMG, “UML Superstructure Specification – Version 2.4.1.” 2011.
 - [26] C. Atkinson, “Meta-Modeling for Distributed Object Environments,” in *Proceedings of the Enterprise Distributed Object Computing Workshop (EDOC ’97)*, 1997, pp. 90–101.
 - [27] C. Atkinson and T. Kühne, “Reducing accidental complexity in domain models,” *Softw. Syst. Model.*, vol. 7, no. 3, pp. 345–359, 2008.
 - [28] V. A. Carvalho, J. P. A. Almeida, and G. Guizzardi, “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.
 - [29] V. A. Carvalho and J. P. A. Almeida, “A Semantic Foundation for Organizational Structures: A Multi-level Approach,” in *IEEE 19th International Enterprise Distributed Object Computing Conference (EDOC 2015)*, 2015, pp. 50–59.
 - [30] V. A. Carvalho, J. P. A. Almeida, C. M. Fonseca, and G. Guizzardi, “Extending the Foundations of Ontology-based Conceptual Modeling with a Multi-Level Theory,” in *34rd International Conference on Conceptual Modeling (ER2015)*, 2015.

- [31] E. J. Lessner, M. R. Stocker, N. D. Smith, A. H. Turner, R. B. Irmis, and S. J. Nesbitt, “A new rauisuchid (Archosauria, Pseudosuchia) from the Upper Triassic (Norian) of New Mexico increases the diversity and temporal range of the clade,” *PeerJ*, vol. 4, p. e2336, 2016.
- [32] W3C, “Internationalized Resource Identifiers (IRIs) - RFC3987.” 2005.
- [33] C. Atkinson and T. Kühne, “Model-Driven Development: A Metamodeling Foundation,” *IEEE Softw.*, vol. 20, no. 5, pp. 36–41, 2003.
- [34] OMG, “Unified Modeling Language Specification v1.3.” 1999.
- [35] N. Jekjantuk, G. Gröner, and J. Z. Pan, “Modelling and Reasoning in Metamodelling Enabled Ontologies,” *Int. J. Softw. Informatics*, vol. 4, pp. 277–290, 2010.
- [36] Wikidata Project, “Help:Items.” [Online]. Available: <https://www.wikidata.org/wiki/Help:Items>. [Accessed: 27-Jan-2016].
- [37] Wikidata Project, “Help:Statements.” [Online]. Available: <https://www.wikidata.org/wiki/Help:Statements>. [Accessed: 27-Jan-2016].
- [38] Wikidata Project, “Help:Modelling.” [Online]. Available: <https://www.wikidata.org/wiki/Help:Modelling>. [Accessed: 21-Jan-2016].
- [39] T. Heath and C. Bizer, *Linked Data: Evolving the Web into a Global Data Space*, vol. 1, no. 1. 2011.
- [40] “MLT.” [Online]. Available: <http://nemo.inf.ufes.br/mlt>.
- [41] The Apache Software Foundation, “Jena.” [Online]. Available: <https://jena.apache.org/>.
- [42] “Hermit.” [Online]. Available: <http://www.hermit-reasoner.com/>.
- [43] J. Z. Pan, I. Horrocks, and G. Schreiber, “OWL FA: A metamodeling extension of OWL DL,” *Proc. Int. Work. OWL Exp. Dir.*, 2005.
- [44] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Groszof, and M. Dean, “SWRL: A Semantic Web Rule Language Combining OWL and RuleML.” W3C Member Submission, 2004.
- [45] Stanford Center for Biomedical Informatics Research, “Protégé.” [Online]. Available: <http://protege.stanford.edu/>.
- [46] M. Fernández-López and A. Gómez-Pérez, “The integration of OntoClean in WebODE,” in *EKA'02 Workshop on Evaluation of Ontology-based Tools (EON2002)*, 2002, pp. 38–52.
- [47] H. Paulheim and A. Gangemi, “Serving DBpedia with DOLCE – More than Just

- Adding a Cherry on Top,” in *The Semantic Web - ISWC 2015*, vol. 9366, 2015, pp. 180–196.
- [48] G. Guizzardi, J. P. A. Almeida, N. Guarino, and V. A. Carvalho, “Towards an Ontological Analysis of Powertypes,” in *Intl. Workshop on Formal Ontologies for Artificial Intelligence (FOFAI 2015)*, 2015.
- [49] P. P. F. Barcelos, V. S. Amorim, F. B. Silva, M. E. Monteiro, and A. S. Garcia, “An Automated Transformation from OntoUML to OWL and SWRL,” in *Proceedings of the 6th Seminar on Ontology Research in Brazil (Ontobras 2013)*, 2013, pp. 130–141.
- [50] V. C. Zamborlini, “Estudo de Alternativas de Mapeamento de Ontologias da Linguagem OntoUML Para OWL: Abordagens Para Representação de Informação Temporal,” Universidade Federal do Espírito Santo, 2011.
- [51] L. Wetzel, “Types and Tokens.” *The Stanford Encyclopedia of Philosophy* (Spring 2014 Edition), Edward N. Zalta (ed.), URL = <http://plato.stanford.edu/archives/spr2014/entries/types-tokens/>.

APPENDIX I - THE MLT VOCABULARY FOR THE SEMANTIC WEB

```

@prefix mlt: <http://www.nemo.inf.ufes.br/mlt#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

mlt:TokenIndividual rdf:type owl:Class ;
  owl:equivalentClass [
    rdf:type owl:Restriction ;
    owl:onProperty [
      owl:inverseOf mlt:isPowertypeOf
    ] ;
    owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;
    owl:onClass mlt:1stOrderClass
  ] ;
  rdf:type mlt:1stOrderClass .

mlt:1stOrderClass rdf:type owl:Class ;
  owl:equivalentClass [
    rdf:type owl:Restriction ;
    owl:onProperty [
      owl:inverseOf mlt:isPowertypeOf
    ] ;
    owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;
    owl:onClass mlt:2ndOrderClass
  ] ;
  rdfs:subClassOf rdfs:Class ,
  [ rdf:type owl:Restriction ;
    owl:onProperty mlt:isPowertypeOf ;
    owl:maxQualifiedCardinality "1"^^xsd:nonNegativeInteger ;
    owl:onClass mlt:TokenIndividual
  ] ;
  rdf:type mlt:2ndOrderClass ;
  mlt:isPowertypeOf mlt:TokenIndividual .

mlt:2ndOrderClass rdf:type owl:Class ;
  owl:equivalentClass [
    rdf:type owl:Restriction ;
    owl:onProperty [
      owl:inverseOf mlt:isPowertypeOf
    ] ;
    owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;
    owl:onClass mlt:3rdOrderClass
  ] ;
  rdfs:subClassOf rdfs:Class ,
  [ rdf:type owl:Restriction ;
    owl:onProperty mlt:isPowertypeOf ;
    owl:maxQualifiedCardinality "1"^^xsd:nonNegativeInteger ;
    owl:onClass mlt:1stOrderClass
  ] ;
  rdf:type mlt:3rdOrderClass ;
  mlt:isPowertypeOf mlt:1stOrderClass .

mlt:3rdOrderClass rdf:type owl:Class ;
  rdfs:subClassOf rdfs:Class ,
  [ rdf:type owl:Restriction ;
    owl:onProperty mlt:isPowertypeOf ;

```

```

        owl:maxQualifiedCardinality "1"^^xsd:nonNegativeInteger ;
        owl:onClass mlt:2ndOrderClass
    ] ;
    mlt:isPowertypeOf mlt:2ndOrderClass .
mlt:mltProperty rdf:type owl:ObjectProperty ;

mlt:intraLevelProperty rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf mlt:mltProperty .

mlt:crossLevelProperty rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf mlt:mltProperty .

mlt:isSubordinatedTo rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf mlt:intraLevelProperty ;

mlt:isPowertypeOf rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf mlt:crossLevelProperty ;

mlt:categorizes rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf mlt:crossLevelProperty ;

mlt:completelyCategorizes rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf mlt:categorizes ;
    owl:propertyDisjointWith mlt:incompletelyCategorizes .

mlt:disjointlyCategorizes rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf mlt:categorizes ;
    owl:propertyDisjointWith mlt:overlappinglyCategorizes .

mlt:incompletelyCategorizes rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf mlt:categorizes .

mlt:overlappinglyCategorizes rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf mlt:categorizes ;

mlt:partitions rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf mlt:completelyCategorizes,
        mlt:disjointlyCategorizes .

[
    rdf:type owl:AllDisjointClasses ;
    owl:members (
        mlt:TokenIndividual
        mlt:1stOrderClass
        mlt:2ndOrderClass
        mlt:3rdOrderClass
    )
] .

```

APPENDIX II - SPARQL QUERIES FOR WIKIDATA ANTI-PATTERNS

To access data from Wikidata, we used the *Simplified and derived RDF dumps* of Wikidata from January 4th, 2016, available at *RDF Exports from Wikidata*³. Moreover, we have queried these using SPARQL, where *instance of* and *subclass of* are represented as *rdf:type* and *rdfs:subClassOf*, respectively. Note that, in this dump, whenever an item is *subclass of* another item or when it has subclasses or instances, then it is declared to be an *instance of owl:Class* (through the *rdf:type* property).

Note that since we are concerned only with the *instance of* properties occurring between items in Wikidata, the SPARQL query ignores the triples that declare resources to be *instances of owl:Class*; these are artificial triples introduced in the dump as part of the RDF representation strategy and do not correspond to *instance of* statements in Wikidata.

Table II-1. SPARQL query for Wikidata Anti-Patterns

Anti-Pattern	SPARQL query
AP1	<pre>SELECT DISTINCT * WHERE { ?Z rdf:type ?A . ?Z rdfs:subClassOf+ ?A . }</pre>
AP2	<pre>SELECT DISTINCT * WHERE { ?B rdf:type ?A . ?C rdfs:subClassOf ?A . ?C rdfs:subClassOf ?B . }</pre>
AP3	<pre>SELECT DISTINCT * WHERE { ?C rdf:type ?B . ?B rdf:type ?A . ?C rdf:type ?A . filter(?A != owl:Class) . }</pre>

³ <http://tools.wmflabs.org/wikidata-exports/rdf/>

APPENDIX III - OWL FILES FOR EMPIRICAL EVALUATION

TIM BERNERS-LEE PROFESSION

```

@prefix wd: <http://www.wikidata.org/entity/> .
@prefix mlt: <http://www.nemo.inf.ufes.br/mlt#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

wd:Profession rdf:type owl:Class .

wd:Creator rdf:type owl:Class ;
    rdfs:subClassOf wd:Profession .

wd:Scientist rdf:type owl:Class,
    wd:Profession;
    rdfs:subClassOf wd:Creator .

wd:ComputerScientist rdf:type owl:Class,
    wd:Profession;
    rdfs:subClassOf wd:Scientist .

wd:TimBernersLee rdf:type mlt:TokenIndividual ,
    wd:ComputerScientist .

```

EARTHQUAKES

```

@prefix wd: <http://www.wikidata.org/entity/> .
@prefix mlt: <http://www.nemo.inf.ufes.br/mlt#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

wd:earthquake rdf:type owl:Class ,
    mlt:1stOrderClass ,
    wd:natural_disaster;
    rdfs:subClassOf wd:natural_disaster .

wd:natural_disaster rdf:type owl:Class .

```

EXCAVATOR

```

@prefix wd: <http://www.wikidata.org/entity/> .
@prefix mlt: <http://www.nemo.inf.ufes.br/mlt#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

wd:crawler_excavator rdf:type owl:Class ;
    rdfs:subClassOf wd:excavator ,
        wd:heavy_equipment .

wd:excavator rdf:type owl:Class,
    mlt:1stOrderClass ,
    wd:heavy_equipment .

wd:heavy_equipment rdf:type owl:Class .

```

URBAN PARK

```

@prefix wd: <http://www.wikidata.org/entity/> .
@prefix mlt: <http://www.nemo.inf.ufes.br/mlt#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

wd:park rdf:type owl:Class .

wd:urban_park rdf:type owl:Class,
    wd:park .

wd:CentralPark rdf:type mlt:TokenIndividual ,
    wd:park ,
    wd:urban_park .

```

BIOLOGICAL TAXONOMY

```

@prefix wd: <http://www.wikidata.org/entity/> .
@prefix mlt: <http://www.nemo.inf.ufes.br/mlt#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

wd:TaxonomicRank rdfs:type owl:Class .

wd:Genus rdfs:type owl:Class ,
          wd:TaxonomicRank .

wd:Species rdfs:type owl:Class,
            wd:TaxonomicRank ;
            mlt:isSubordinatedTo wd:Genus .

wd:Organism rdfs:type owl:Class .

wd:Panthera rdfs:type owl:Class,
            wd:Genus ;
            rdfs:subClassOf wd:Organism .

wd:PantheraLeo rdfs:type owl:Class,
               wd:PantheraSpecies;
               rdfs:subClassOf wd:Panthera .

wd:PantheraOnca rdfs:type owl:Class ;
                rdfs:subClassOf wd:Panthera .

wd:PantheraTigris rdfs:type owl:Class ;
                 rdfs:subClassOf wd:Panthera .

wd:PantheraSpecies rdfs:type owl:Class ;
                  rdfs:subClassOf wd:Species ;
                  mlt:partitions wd:Panthera .

wd:Cecil rdfs:type mlt:TokenIndividual ,
           wd:PantheraLeo .

```