

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
DEPARTAMENTO DE INFORMÁTICA
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

VICTORIO ALBANI DE CARVALHO

**FOUNDATIONS FOR MULTI-LEVEL
ONTOLOGY-BASED CONCEPTUAL MODELING**

DOCTORAL THESIS

VITÓRIA-ES, BRAZIL
DECEMBER, 2016

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

C331f Carvalho, Victorio Albani de, 1980-
Foundations for multi-level ontology-based conceptual
modeling / Victorio Albani de Carvalho. – 2016.
167 f. : il.

Orientador: João Paulo Andrade Almeida.

Coorientador: Giancarlo Guizzardi.

Tese (Doutorado em Ciência da Computação) – Universidade
Federal do Espírito Santo, Centro Tecnológico.

1. Ontologia. 2. Modelos multiníveis (estatísticas).
3. Modelagem conceitual. 4. Modelagem multinível. I. Almeida,
João Paulo Andrade. II. Guizzardi, Giancarlo. III. Universidade
Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU:

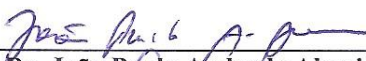


Foundations for Multi-level Ontology-based Conceptual Modeling

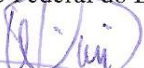
Victorio Albani de Carvalho

Tese submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Aprovada em 16 de dezembro de 2016 por:




Prof. Dr. João Paulo Andrade Almeida (Orientador)
Universidade Federal do Espírito Santo - Brasil



Prof. Dr. Giancarlo Guizzardi (Coorientador)
Universidade Federal do Espírito Santo - Brasil



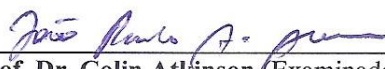
Prof. Dr. Ricardo de Almeida Falbo (Examinador Interno)
Universidade Federal do Espírito Santo - Brasil



Prof. Dr. Vítor Estêvão Silva Souza (Examinador Interno)
Universidade Federal do Espírito Santo - Brasil



Prof. Dr. Fernando Silva Parreiras (Examinador Externo)
Universidade FUMEC - Brasil

pl 

Prof. Dr. Colin Atkinson (Examinador Externo)
University of Mannheim - Alemanha

O julgamento dessa tese foi realizado com a participação por meio de videoconferência do membro externo o Prof. Dr. Colin Atkinson 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, 16 de dezembro de 2016.

“Everything should be made as simple as possible, but not simpler.”

-Albert Einstein

Acknowledgments

Firstly, I would like to express my gratitude to my advisor, Prof. João Paulo Almeida, for his guidance, patience, motivation and fellowship during this long journey. My sincere thanks also goes to my co-advisor, Prof. Giancarlo Guizzardi for the valuable contributions to this work, and to the other members of my thesis committee, namely, Prof. Ricardo Falbo, Prof. Vítor Souza, Prof. Fernando Parreiras, and Prof. Colin Atkinson for evaluating this thesis and contributing for its improvement.

I would like to thank all professors of the Nemo group for sharing their immense knowledge, and to my fellow labmates for the stimulating discussions and for the fun moments. I particularly thank my colleagues Claudenir and Freddy for the partnership in some interesting research topics, John for the tips in Alloy and Ecore, and Julio and Carlos for sharing with me the incertitude and angsts of being a doctoral candidate. I would also like to thank the secretary of the Graduate School, Glaydis, for all the support on the administrative tasks.

I thank my colleagues of the Federal Institute of Espírito Santo for granting me the license which allowed me to dedicate my time exclusively to the doctoral course during the last four years, and CAPES for the financial support.

My eternal gratitude to my family and friends for being part of my life. Their fellowship and support make everything easier. I would like to specially thank my parents, to whom I owe everything in my life.

Abstract

Considering that *conceptual models* are produced with the aim of representing certain aspects of the physical and social world *according to a specific conceptualization* and that *ontologies aim at describing conceptualizations*, there has been growing interest in the use of *ontologies* to provide a sound theoretical basis for the discipline of conceptual modeling. This has given rise to a research area called *ontology-based conceptual modeling*, with significant advances to conceptual modeling in the last decades. Despite these advances, *ontology-based conceptual modeling* still lacks proper support to address subject domains that require not only the representation of categories of individuals but also the representation of categories of categories (or types of types). The representation of entities of multiple (related) classification “levels” has been the focus of a separate research area under the banner of *multi-level modeling*, aiming to address the limitations of the conventional two-level modeling paradigm. Despite the relevant contributions of *multi-level modeling* and *ontology-based conceptual modeling*, their combination has not yet received due attention. This work explores this gap by proposing the use of formal theories for multi-level modeling in combination with foundational ontologies to support what we call *multi-level ontology-based conceptual modeling*. To provide a well-founded approach to multi-level conceptual modeling, we develop a theory called MLT that formally characterizes the nature of classification levels and precisely defines the relations that may occur between elements of different classification levels. In order to leverage the benefits of the use of a foundational ontology to domains dealing with multiple classification levels, we combine the proposed multi-level modeling theory with a foundational ontology. This combination results in a hierarchical modeling approach that supports the construction of multi-level conceptual models in a spectrum of levels of specificity, from foundational ontologies to domain models. To demonstrate the applicability of our multi-level ontology-based conceptual modeling approach, we employ it to develop a core ontology for organizational structure, a domain that spans multiple classification levels. Further, we show how MLT can be used as a reference theory to clarify the semantics and enhance the expressiveness of UML with respect to the representation of multi-level models. The resulting UML profile enables the practical application of MLT.

Keywords: conceptual modeling, models, ontology, foundational ontology, ontology-based conceptual modeling, classification levels, multi-level modeling, power type.

Resumo

Considerando que modelos conceituais são produzidos com o objetivo de representar certos aspectos do mundo físico e social de acordo com uma conceituação específica e que ontologias buscam descrever conceituações, tem havido crescente interesse no uso de ontologias para fornecer uma base teórica sólida para a disciplina de modelagem conceitual. Esse interesse deu origem a uma área de pesquisa denominada *modelagem conceitual baseada em ontologias*, com avanços significativos na modelagem conceitual nas últimas décadas. Apesar desses avanços, a modelagem baseada em ontologias não provê suporte adequado à modelagem de domínios que exigem a representação de categorias de indivíduos e de categorias de categorias (ou tipos de tipos). A representação de entidades de vários "níveis" de classificação tem sido o foco de uma área de pesquisa distinta denominada *modelagem multi-nível*. As iniciativas em modelagem multi-nível visam a contornar as limitações impostas pelo paradigma convencional de modelagem em dois níveis. Apesar das contribuições relevantes das áreas de modelagem multi-nível e de modelagem conceitual baseada em ontologias, a combinação dessas duas áreas ainda não recebeu a devida atenção. Este trabalho explora essa lacuna propondo o uso combinado de teorias formais para a modelagem multi-nível e de ontologias de fundamentação para apoiar o que chamamos de modelagem conceitual multi-nível baseada em ontologias. Para fornecer uma abordagem bem fundamentada à modelagem conceitual multi-nível, desenvolvemos uma teoria chamada MLT. MLT caracteriza formalmente a natureza dos níveis de classificação e define precisamente as relações que podem ocorrer entre elementos de diferentes níveis de classificação. A fim de aproveitar os benefícios do uso de ontologias de fundamentação na modelagem de domínios que abrangem vários níveis de classificação, combinamos MLT com uma ontologia de fundamentação. Essa combinação resulta em uma abordagem de modelagem que apoia a construção de modelos conceituais multi-níveis em um espectro de níveis de especificidade, desde ontologias de fundamentação até modelos conceituais de domínios específicos. Para demonstrar a aplicabilidade da nossa abordagem de modelagem conceitual multi-nível baseada em ontologias, a empregamos para desenvolver uma ontologia núcleo para estruturas organizacionais, um domínio que abrange vários níveis de classificação. Além disso, mostramos como MLT pode ser usada como uma teoria de referência para esclarecer a semântica e aumentar a expressividade de UML no que diz respeito à representação de modelos de multi-níveis. O perfil UML produzido viabiliza a aplicação prática de MLT pela comunidade de modelagem conceitual.

Palavras-chave: modelagem conceitual, modelo conceitual, ontologia, ontologia de fundamentação, modelagem conceitual baseada em ontologias, níveis de classificação, modelagem multi-nível, *power type*.

List of Figures

Figure 1 - An ontology is an artifact that explicitly represent a conceptualization.....	22
Figure 2 - Generality level of ontology as a continuum (FALBO et al., 2013).	23
Figure 3 - UFO <i>endurant individuals</i> and <i>universals</i> taxonomies	26
Figure 4 - An OntoUML diagram	27
Figure 5 - Illustrating the notation proposed by Odell to represent that each instance of the <i>base type</i> is instance of one instance of the <i>power type</i> (adapted from (ODELL, 1994)).....	30
Figure 6 - Illustrating the notation proposed by Odell to associate <i>power types</i> with subtypes partitions (adapted from (ODELL, 1994))	30
Figure 7 - Illustrating the deep instantiation approach using a deep-modeling tool called <i>Melanee</i> (ATKINSON; GERBIG, 2012).....	33
Figure 8 - The Orthogonal Classification Architecture (extracted from (ATKINSON; KÜHNE, 2003)).	34
Figure 9 - Basic foundations of MLT: <i>basic types</i> and <i>instance of</i> relations.	43
Figure 10 - Intra- level structural relations: specialization and proper specialization.....	44
Figure 11 - Using the theory to model a domain.....	45
Figure 12 - Instantiations and specializations between domain elements.	46
Figure 13 - Intra-level structural relations: subordination.....	47
Figure 14 - An example of subordination relation.	47
Figure 15 - Cross-level relations: <i>isPowertypeOf</i>	50
Figure 16 - An example of <i>isPowertypeOf</i> relation.	50
Figure 17 - Cross-Level relations: <i>categorization</i>	51
Figure 18 - An example of domain modeling applying <i>categorization</i> and <i>subordination</i> relations.	53
Figure 19 - Applying our theory basic pattern to the biological taxonomy for living beings.	56
Figure 20 - Using our theory to describe the structural relations that exist in biological taxonomy (relations between the notions of biological rank, biological taxon and living being).	57
Figure 21 - Illustrating the account for attributes.....	60
Figure 22 - Illustrating the notion of <i>regularity attributes</i>	62
Figure 23 - Illustrating a scenario in which relations in one order capture regularities over instances of types in one order lower.	63
Figure 24 - Illustrating a hierarchy of conceptual models founded on the MLT-UFO combination.	82
Figure 25 - Applying MLT to UFO fragment on endurants.....	83
Figure 26 - Applying the MLT notion of regularity attribute to interpret the UFO notion of <i>characterization</i>	84

Figure 27 - A domain second-order type specializing “Kind” and partitioning an instance of “Category”.....	86
Figure 28 - A domain second-order type specializing “Subkind” and partitioning an instance of “Kind”.	87
Figure 29 - Domain second-order types specializing “Subkind” and partitioning instances of “Category” and “Kind”.	87
Figure 30 - A domain second-order type specializing “Phase” and partitioning an instance of “Kind”.	88
Figure 31 - A domain second-order type specializing “Phase” and partitioning an instance of “Phase Mixin”.	89
Figure 32 - Domain second-order types specializing “Role” and <i>categorizing</i> instances of “Sortal Universal”.	90
Figure 33 - Domain second-order types specializing “Kind” and “Role” and partitioning instances of “Mixin Universal”	91
Figure 34 - A domain second-order type specializing “Category” and <i>partitioning</i> an instance of “Category”	92
Figure 35 - Using MLT-UFO combination to describe concepts involved in biological taxonomy.	92
Figure 36 - Second-order types specializing “Phase Mixin”	93
Figure 37 - Domain second-order types specializing “Role Mixin”.	94
Figure 38 - Second-order types specializing “Quality Universal”.	96
Figure 39 - Making explicit the relation between substantials and qualities taxonomies.	97
Figure 40 - Using the notion of regularity attribute to explain the relation between taxonomies of intrinsic moments and taxonomies of substantials.	98
Figure 41 - Making explicit the relation between substantials and modes taxonomies.	100
Figure 42 - Second-order types specializing “Relator Universal”.	101
Figure 43 - Explicitly representing the relation between taxonomies of substantials and taxonomies of relators.	102
Figure 44 - Illustrating a hierarchy of organizational structure conceptual models founded on MLT-UFO combination.	109
Figure 45 - Including concepts of the UFO social layer to the MLT-UFO combination model.	110
Figure 46 - Illustrating the definitions of the core organizational structure ontology and its use as the basis of a hierarchy of models.	111
Figure 47 - Illustrating the use of regularity attributes to constrain formal organizations and organizational units compositions.	113
Figure 48 - Fragment of the core organizational structure ontology that copes with agents that compose the organization and types of roles they may play.	114

Figure 49 - Illustrating the relation between business roles and formal organization member types (the <i>cover</i> relation).	115
Figure 50 - Representing the relation between taxonomies of admissions, taxonomies of formal organizations and taxonomies of formal organization members.....	117
Figure 51 - Representing the relation between taxonomies of assignments, taxonomies of organizational units and taxonomies of business roles.	118
Figure 52 - The UML notation for the power type pattern (adapted from (OMG, 2011))......	125
Figure 53 - Illustrating the use of «instantiation».	127
Figure 54 - Using «instantiation» to denote partitions relations.	128
Figure 55 - Combining partitions relations with “incomplete” generalization sets.	129
Figure 56 - Using «instantiation» to denote categorization relations that are disjoint but not complete.	130
Figure 57 - Using «instantiation» to denote categorization relations that are complete but not disjoint.....	131
Figure 58 - Using «instantiation» to denote categorization relations that are not complete nor disjoint.....	131
Figure 59 - Using the isEnumerated attribute to disambiguate a model.	132
Figure 60 - An example of inconsistent use of the isEnumerated attribute making the model syntactically invalid.	133
Figure 61 - Using «powerType» and «instantiation» to denote <i>is power type of</i> relations.	134
Figure 62 - Illustrating syntactic constraints concerning hierarchies of higher-order types.	135
Figure 63 - Illustrating syntactic constraints concerning types orders.	136

List of Tables

Table 1 - Intra-level structural relations characteristics.	48
Table 2 - Cross-level structural relations characteristics.....	55
Table 3 - Requirements addressed by power type-based approaches.	73
Table 4 - Requirements addressed by the analyzed multi-level modeling approaches.	78
Table 5 - Allowed base types for specializations of Substantial Universal.	103
Table 6 - The influence of the multiplicities in the semantics of «instantiation» associations.	128
Table 7 - Analyzing the combination of «instantiation» with generalization set constraints....	132
Table 8 - Syntact constraints for combined dependency stereotypes and generalization sets constraints (using <i>isEnumerated</i>).	133

List of Acronyms

ARIS – Architecture of Integrated Information Systems

DoDAF – Department of Defense Architecture Framework

EA – Enterprise Architecture

E-OPL – Enterprise Ontology Pattern Language

IEC – International Electrotechnical Commission

ISO – International Organization for Standardization

MLT – Multi-level Theory

OCA – Orthogonal Classification Architecture

OMG – Object Management Group

OWL – Ontology Web Language

UFO – Unified Foundational Ontology

UML – Unified Modeling Language

W3C – World Wide Web Consortium

Contents

Chapter 1.	Introduction	15
1.1	Context and Motivation.....	15
1.2	Objectives.....	17
1.3	Approach	17
1.4	Outline of this Thesis	19
Chapter 2.	Basic Notions	22
2.1	Ontology.....	22
2.2	Ontology-based Conceptual Modeling.....	23
2.2.1	Principles	23
2.2.2	Conceptual Modeling Founded on UFO	24
2.3	Multi-level Modeling	29
2.3.1	Power Types.....	29
2.3.2	Clajjects and Deep Instantiation.....	31
2.3.3	Ontological Instantiation and Linguistic Instantiation	33
2.4	Final Considerations.....	34
Chapter 3.	A Formal Theory for Multi-Level Conceptual Modeling	36
3.1	Requirements for a Multi-level Conceptual Modeling Theory	37
3.2	MLT Foundations: Basic Types and the Instantiation Relation.....	39
3.3	Intra-level Structural Relations	43
3.3.1	Specialization and Proper Specialization Relations	43
3.3.2	The Subordination Relation.....	46
3.4	Cross-level Structural Relations.....	48
3.4.1	The Power Type Of Relation.....	48
3.4.2	The Categorization Relation and its variations	50
3.4.3	Summary	54
3.5	A Paradigmatic Example.....	55
3.6	Accounting for Attributes and Relationships	57
3.7	Addressing Dynamic Classification	65
3.8	A Note on the Identity Conditions of Types	67
3.9	Addressed Requirements.....	68
3.10	Related Work	69
3.10.1	Power type-based Approaches	69
3.10.2	Deep Instantiation-based Approaches	73
3.11	Final Considerations.....	78
Chapter 4.	Multi-Level Ontology-based Conceptual Modeling with MLT and UFO	81

4.1	Overview	81
4.2	Combining MLT and UFO.....	82
4.3	Guidelines for the Application of MLT-UFO Combination	84
4.3.1	Second-Order Types Specializing “Sortal Universal”	86
4.3.2	Second-Order Types Specializing “Mixin Universal”	91
4.3.3	Second-Order Types Specializing “Moment Universal”	94
4.3.4	Summary	102
4.4	Related Work	103
4.5	Final Considerations.....	105
Chapter 5.	Applying MLT-UFO in the Development of a Core Ontology	107
5.1	A Core Organization Ontology Founded on MLT-UFO.....	108
5.1.1	Organizational Structure	110
5.1.2	Organization Roles and Allocations.....	114
5.2	Related Work	118
5.2.1	Organizational Structure Ontologies	119
5.2.2	Organizational Structure Approaches	121
5.3	Final Considerations.....	122
Chapter 6.	Using MLT to Revisit the Representation of Multi-Level Models in UML	124
6.1	UML’s Power Type Pattern Support in a Nutshell	125
6.2	Applying MLT to Revisit the Power type Support in UML	126
6.2.1	The «instantiation» Stereotype.....	127
6.2.2	An Additional Attribute for Generalization Sets.....	132
6.2.3	The «powerType» Stereotype	133
6.2.4	Syntactic Constraints Motivated by MLT Rules	134
6.3	Final Considerations.....	136
Chapter 7.	Final Considerations.....	138
7.1	Contributions.....	138
7.2	Future Perspectives	141
References.....		144
Appendix A.	Specification of MLT in Alloy – The basic theory.....	152
Appendix B.	Specification of MLT in Alloy - Addressing Dynamic Classification	160

Chapter 1. Introduction

This chapter presents an overview of this thesis and defines the basis for the subsequent chapters. It discusses the context in which the thesis is embedded, the motivation for conducting this work, the objectives, and the methodological aspects that have guided this research work. Finally, it presents the structure of this document.

1.1 Context and Motivation

Conceptual modeling is the activity of formally describing some aspects of the physical and social world around us for the purposes of understanding and communication (MYLOPOULOS, 1992). It is generally considered a fundamental activity in information systems engineering (OLIVÉ, 2007), in which a given subject domain is described independently of specific implementation choices (GUIZZARDI, 2005). The main artifact of this activity is a conceptual model, i.e., a specification aiming at representing a conceptualization of the subject domain of interest.

Given the scope and purpose of conceptual modeling, suitable techniques for this endeavor should be based on abstractions with consideration for human cognition and common sense (GUARINO, 1994; GUIZZARDI, 2005). A number of authors, such as Wand and Weber (1993), Guarino (1994) and Guizzardi (2005), have defended that ontologies can provide such abstractions and serve as a sound theoretical basis for the discipline of conceptual modeling, giving rise to a research area called *ontology-based conceptual modeling* (GUIZZARDI, 2005). As discussed by some of these authors, when conceptual modeling languages take into account formal distinctions, the potential misunderstandings and inconsistencies in conceptual models are reduced, i.e. when ontological concerns are addressed, the quality of conceptual models is improved, facilitating thus the understanding and communication about a domain of enquiry (CARRARETTO, 2012).

Two prominent examples of the use of ontologies to provide a sound basis for conceptual modeling are (GUARINO; WELTY, 2002a) and (GUIZZARDI, 2005). Guarino and Welty (2002) defined a methodology whose main goal is to detect formal and semantic inconsistencies in the properties defined in a conceptual model. They focus on defining a methodology to evaluate the soundness of taxonomic structures considering a set of metaproperties that characterize relevant aspects of categories of individuals and relations that make up the model. Guizzardi (2005), in his turn, defines a *foundational ontology* called UFO (*Unified Foundational Ontology*) and uses this ontology to base his conceptual modeling activities. As a *foundational ontology*, UFO captures domain-independent concepts which may be applied to any modeling effort,

including some metaproperties defined in (GUARINO; WELTY, 2002a). Further, aiming to provide an ontologically well-founded support to conceptual modeling, Guizzardi (2005) proposes OntoUML. OntoUML is a UML (OMG, 2011) profile that reflects the taxonomy of categories of UFO such that the distinctions of the foundational ontology can be used to provide beneficial constraints and modeling guidelines, ultimately leading to ontologically well-founded conceptual models. The resulting conceptual models represent *categories of individuals in the subject domain* (e.g., the “Person” kind, the “Child” phase, the “Student” role).

So far, the support to ontology-based conceptual modeling has been focused mainly on the analysis of properties of categories of individuals. As a consequence, these approaches are unable to describe subject domains in which the categorization scheme itself is part of the subject matter. In these subject domains, experts make use of categories of categories (i.e. types of types) in their accounts. For instance, considering the software development domain (GONZALEZ-PEREZ; HENDERSON-SELLERS, 2006), project managers often need to plan according to the *types of tasks* to be executed during the software development project (e.g. “requirements specification”, “coding”). They may also need to classify those *types of tasks* giving rise to *types of types of tasks*. In this case, “requirements specification” and “coding” could be considered as examples of “technical task types”, as opposed to “management task types”. Finally, during project development, they need to track the execution of individual tasks (e.g. specifying the requirements of the system X). Thus, to describe the conceptualization underlying the software development domain, one needs to represent entities of different (but nonetheless related) classification “levels”, such as *tasks (specific individual occurrences)*, *types of tasks*, and *types of types of tasks*. The need to support the representation of subject domains that deal with multiple classification levels has given rise to what has been referred to as *multi-level modeling* (ATKINSON; KÜHNE, 2001; NEUMAYR; GRÜN; SCHREFL, 2009).

Despite the recent advances in multi-level modeling approaches and tools, there is still no consensus on what kinds of constructs and concepts provide the best support for it. Actually, one of the fundamental challenges on multi-level modeling is the definition of what qualities an approach needs to possess in order to be characterized as a multi-level approach (ATKINSON; GERBIG; KÜHNE, 2014). Further, the literature on multi-level modeling lacks a formal theory capturing the foundational concepts underlying multi-level modeling. We believe that such a foundational theory could facilitate the identification of the characterizing features a multi-level approach should possess, being useful to support the proposal of well-founded multi-level modeling approaches. Further, such a theory could be used as foundations to clarify the semantics of existing approaches as well as to relate and harmonize different approaches to multi-level modeling.

Regardless of the growing interest on both *multi-level modeling* and *ontology-based conceptual modeling*, the combination of these two prominent areas has not yet received due

attention. The focus of the present work is on the use of formal theories for multi-level modeling in combination with foundational ontologies to support the design of ontologically well-founded conceptual models dealing with multiple classification levels, characterizing what we call *multi-level ontology-based conceptual modeling*.

1.2 Objectives

The general objective of this work is to extend the foundations of ontology-based conceptual modeling to incorporate support for dealing with multiple classification levels. We approach this objective with the combination of a formal theory for multi-level modeling and a foundational ontology, which allows us to leverage the benefits of ontology-based conceptual modeling to domains that includes categories of categories, providing, thus, support to *multi-level ontology-based conceptual modeling*.

In order to pursue this general objective, the following specific objectives are defined:

- SO1. To develop a formal theory that captures the conceptualization underlying multi-level modeling. Such a multi-level theory must formally characterize defining aspects of categories in the different classification levels as well as precisely define the relations that may occur between elements of different classification levels;
- SO2. To define an approach to support multi-level ontology-based conceptual modeling based on the combination of the proposed multi-level theory (SO1) with a foundational ontology. The proposed approach must guide the construction of models that incorporate the guidelines of the foundational ontology as well as those of the multi-level theory;
- SO3. To demonstrate the applicability of the proposed multi-level ontology-based conceptual modeling approach by applying it to construct a core ontology on a domain that spans multiple classification levels; and,
- SO4. To show how the proposed multi-level theory can be used as a reference theory to clarify the semantics and enhance the expressiveness of a widely employed modeling language with respect to the representation of multi-level models.

1.3 Approach

We follow a systematic approach based on a strict separation of concerns: first, the conceptualization underlying certain phenomena is captured by a reference theory; second,

representation strategies that reflect the conceptualization captured by the reference theory are devised, while addressing technological and pragmatic concerns¹.

As discussed by Guizzardi (2005), a theory that is to be suitable as a reference for a representation strategy should be constructed with the sole objective of capturing the best possible description of a certain vision of the world. Therefore, technological issues and language engineering concerns should play a minor role, if any.

In the case of multi-level modeling, a reference theory is not readily available in the literature. However, in the last decades, several approaches for the representation of multi-level models have been worked out, including those mostly focused on multi-level modeling from a model-driven engineering perspective (ATKINSON; KÜHNE, 2003; FRANK, 2014; LARA et al., 2013) and those that propose modeling languages (approaches) for models with multiple levels of classification (ATKINSON; GERBIG, 2012; DE LARA; GUERRA, 2010; NEUMAYR; GRÜN; SCHREFL, 2009). These approaches embody conceptual notions that are key to the representation of multi-level models, such as the existence of entities that are simultaneously types and instances (classes and objects), the iterated application of instantiation, etc. Therefore, they can be used to set requirements for a reference theory capturing the conceptualization underlying multi-level modeling, independently of technological and language engineering concerns.

We propose a reference theory formalized into an axiomatic theory named MLT. MLT characterizes the nature of classification levels, and precisely defines the relations that may occur between elements of different classification levels, addressing thereby SO1. The axioms and theorems of MLT provide us with rules and patterns to guide the design of sound multi-level models. Further, they allow us to contrast and relate conceptual notions that underlie different multi-level modeling approaches, enabling the harmonization of such approaches. We employ first-order logic in the specification of MLT and verify the theory's consistency as well as the validity of the proposed theorems by applying a lightweight formal method based on the use of Alloy (JACKSON, 2006). Alloy also allows us to simulate the specification to validate that it indeed matches the multi-level modeling phenomena we aim to characterize.

We subscribe to the approach put forward by Guarino (1994) and Guizzardi (2005) who have defended that conceptual modeling should be rooted in a set of domain-independent concepts from a formal *foundational ontology*. A foundational ontology captures a philosophically and cognitively well-founded general (meta) conceptualization dealing with formal aspects of entities irrespective of their particular nature, e.g., identity and unity, types and instantiation, rigidity, mereology, dependence (GUIZZARDI, 2005). In order to benefit from the use of a foundational ontology while addressing multi-level concerns, we combine MLT with the Unified Foundational

¹ This approach is worked out in (CARVALHO; ALMEIDA; GUIZZARDI, 2014) for the case of domain ontologies as reference theories supporting the design of domain-specific languages.

Ontology (UFO) (GUIZZARDI, 2005), addressing thereby SO2. We adopt UFO, as it has been successfully employed to analyze a number of classical conceptual modeling constructs, including Object Types and Taxonomic Structures, Attributes and Datatypes, Intrinsic and Relational Properties, Weak Entities, and Part-Whole Relations (GUIZZARDI, 2005).

The patterns and rules of MLT are applied to establish the relation between MLT and UFO, and later to establish the relation between a conceptual domain model and the MLT-UFO combination. This combination results in a hierarchical modeling approach that supports the construction of multi-level conceptual models in a spectrum of levels of specificity, from the foundational ontology to domain models. The models constructed following the proposed approach respect all rules and patterns of both MLT and UFO.

To illustrate the applicability of this approach to address multi-level ontology-based conceptual modeling, we use it to construct a core organizational structure ontology (a task corresponding to SO3). We show how the application of the approach leads to the proposal of a core ontology that can be extended with models to specify enterprise-specific structures, allowing modelers to cope with the large diversity in organizational structures and structuring approaches. The proposed core ontology contributes to the enterprise modeling area defining a semantic foundation for the organizational structure domain that reflects the domain's multi-level nature, differently from a number of existing organizational structure ontologies and modeling approaches.

To demonstrate the applicability of MLT as a reference theory in the redesign of modeling language mechanisms (SO4) we use the theory to found an analysis of the UML support for representing the power type pattern. The power type pattern is an early approach to represent certain multi-level phenomena and is used extensively in many important modeling initiatives, such as the ISO/IEC 24744 standard (ISO/IEC, 2007). The analysis allows us to demonstrate that UML's current support for the power type pattern lacks expressivity, clarity, and parsimony. By employing the result of this analysis, we propose a UML profile to address the exposed limitations. We use the formal rules of MLT to systematically incorporate syntactic constraints in the language, guiding thus the modeler to produce sound multi-level models. A tool that implements the proposed profile and performs syntactic verification of MLT rules is provided. Considering that UML is a *de facto* standard for conceptual modeling, the proposed UML profile and the provided tool support enable the practical application of MLT by the conceptual modeling community.

1.4 Outline of this Thesis

The remainder of this thesis is organized as follows:

- Chapter 2. Basic Notions: This chapter describes some notions that are required to substantiate this work, including a brief discussion on the different meanings attributed to “ontology”, an introduction to ontology-based conceptual modeling with a focus on UFO and OntoUML, and a brief introduction to multi-level conceptual modeling, highlighting the main concepts underlying some prominent multi-level modeling approaches that influence our multi-level theory.
- Chapter 3. A Formal Theory for Multi-Level Conceptual Modeling: This chapter presents a formal theory for multi-level modeling. First, it establishes the requirements to be satisfied and then discusses how the theory formally characterizes the classification levels and the entities that populate each level. Further, it presents the formal definitions of structural relations that occur between entities captured in a multi-level model. An extensive review of related work is provided in this chapter in order to position the proposed theory and the current literature on multi-level modeling.
- Chapter 4. Multi-Level Ontology-based Conceptual Modeling with MLT: This chapter discusses the MLT-UFO combination and presents guidelines for the introduction of second-order types (i.e. types of types) in ontologically well-founded conceptual models, respecting the rules of both theories. The MLT-UFO combination is positioned as a top layer in a hierarchy of ontology-based multi-level models.
- Chapter 5. Applying MLT-UFO in the Development of a Core Ontology: This chapter presents a core organizational structure ontology founded in the MLT-UFO combination, in order to demonstrate the application of our multi-level ontology-based modeling approach. The suitability of the organizational structure domain for this task is justified considering the multi-level nature of this domain, involving a number of types of types that vary in different organizations (including, e.g., organization unit types, employee types, types of employment relations).
- Chapter 6. Using MLT to Revisit the Representation of Multi-Level Models in UML: This chapter demonstrates the use of MLT as a reference theory to support the redesign of the UML’s power type pattern support. It presents an analysis of power types in UML in the light of MLT, demonstrating that the current support lacks *expressivity*, *clarity*, and *parsimony*. Then a UML profile that reflects the distinctions and rules of MLT to address the exposed limitations is proposed.
- Chapter 7. Final Considerations: This chapter summarizes the research contributions, lists the published papers, discusses some limitations of this work, and describes future perspectives.

- **Appendix A: Specification of MLT in Alloy – The basic theory:** This appendix presents a specification of MLT in Alloy that assumes, for simplification, that entities instantiate types necessarily, effectively dealing with rigid types in a static classification setting.
- **Appendix B: Specification of MLT in Alloy – Addressing Dynamic Classification:** This appendix presents a specification of MLT in Alloy that addresses dynamic classification.

Chapter 2. Basic Notions

This chapter describes some basic notions, which are required to substantiate this work. It starts with a brief discussion on the different meanings attributed to “ontology” in order to establish the sense that will be adopted here (Section 2.1). Using this notion of ontology, a brief characterization of ontology-based conceptual modeling is presented in Section 2.2. This section focuses on the use of the Unified Foundational Ontology (UFO) (GUIZZARDI, 2005) as a basis for ontology-based conceptual modeling. In order to establish the notion of multi-level conceptual modeling adopted in this work, a discussion on what characterizes multi-level conceptual modeling is conducted in Section 2.3. This section also discusses the main concepts underlying some of the most prominent multi-level modeling approaches in software engineering. Finally, some concluding remarks are presented (Section 2.4).

2.1 Ontology

Many meanings have been attributed to the term “ontology”. “Ontology” (with uppercase initial) refers to a philosophical discipline that studies the most general features of reality, dealing with relations between entities that belong to distinct domains of science (e.g., Physics, Chemistry, Biology), and also by entities recognized by common sense (GUARINO, 1998; GUIZZARDI, 2007).

With lowercase initial, “ontology” is used considering two perspectives, namely a philosophical and an engineering perspective. In the philosophical perspective, ontology is a *particular system of categories* accounting for a certain vision of the world independent of the language used to describe it (GUIZZARDI, 2007). This is what we call here *conceptualization*. In contrast, in the engineering perspective, this term is used to refer to *an artifact* represented in a specific language (GUARINO, 1998; GUIZZARDI, 2007). We adopt here this engineering perspective and following Gruber (1993) we define an *ontology* as “an explicit representation of a conceptualization”. Figure 1 illustrates the relation between an *ontology* and a *conceptualization* as adopted in this work. The *conceptualization* is illustrated with a cloud alluding to the fact that, in contrast to *ontologies* that are symbolic artifacts, *conceptualizations* are abstract entities that only exist in the thought realm.

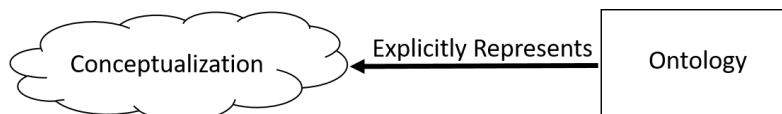


Figure 1 - An ontology is an artifact that explicitly represents a conceptualization.

Different ontologies are developed to capture conceptualizations with different levels of generality. Some authors have proposed taxonomies of types of ontologies that classify ontologies

according to their level of generality (FALBO et al., 2013; GUARINO, 1998; SCHERP et al., 2011): (i) a *foundational ontology* describes very general concepts independently of a particular problem or domain, such as object, property, relation, event, action, etc.; (ii) a *core ontology* provides a precise definition of structural knowledge in a specific field that spans across different domains (e.g., an ontology that describes different aspects of events); (iii) a *domain ontology* describes a conceptualization related to a generic domain (such as biology, software processes, genealogy); (iv) a *task ontology* describes a conceptualization related to a generic task (such as diagnosis, planning, meeting), and (v) an *application ontology* describes concepts dependent on a particular domain and task (e.g., a medical ontology that is defined by the specialization of a disease domain ontology and a diagnosis task ontology). As illustrated by Figure 2, these types of ontologies can be seen as regions in a spectrum or continuum with fuzzy boundaries between them (FALBO et al., 2013).

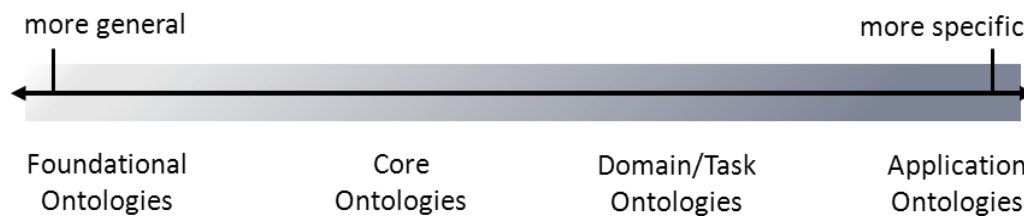


Figure 2 - Generality level of ontology as a continuum (FALBO et al., 2013).

Regarding the purpose of usage, ontologies can be classified as *reference ontologies* or as *operational ontologies* (FALBO et al., 2013; GUIZZARDI, 2007). *Reference ontologies* are designed to be used in an off-line manner to assist humans in tasks of meaning negotiation and consensus establishment. These ontologies should be constructed with the sole objective of making the best possible description of the domain in reality (GUIZZARDI, 2007). Once users have already agreed on a common conceptualization, specialized versions of a reference ontology can be created for run-time use. These versions are classified as *operational ontologies*, and sacrifice representation adequacy and theoretical foundation to guarantee desirable computational properties (e.g., expressiveness, and tractability) (FALBO et al., 2013; GUIZZARDI, 2007). The notion of *reference ontology* is especially important to this work, and will find use in the revision and definition of modeling approaches.

2.2 Ontology-based Conceptual Modeling

2.2.1 Principles

Authors such as Guizzardi (2007), and Wand and Weber (1993) have defended that an appropriate *conceptual modeling language* should provide modeling primitives that reflect the conceptual categories defined in a *reference ontology* capturing the conceptualization underlying the phenomena at hand. For example, a modeling language to represent genealogy trees should

provide modeling primitives reflecting concepts formally defined by a reference domain ontology in genealogy (e.g. “Man”, “Woman”, “biological father of”, etc.) (CARVALHO; ALMEIDA; GUIZZARDI, 2014; GUIZZARDI, 2005). Further, the real-world rules formalized by the *reference* ontology should guide the definition of syntactic constraints to govern how the modeling primitives can be combined (e.g. only instances of “Man” can be the source of “biological father of” relations). These syntactic constraints that have the purpose of reflecting real-world rules in the language, characterizes what we call *semantically-motivated syntactic constraints* (CARVALHO; ALMEIDA; GUIZZARDI, 2014).

Based on similar principles, Guarino (1994) and Guizzardi (2005) defend that an appropriate *general-purpose conceptual modeling language* should provide modeling primitives that reflect the conceptual categories defined in a foundational ontology. Thus, these languages should reflect a (meta) conceptualization, dealing with formal aspects of entities irrespective of their particular nature, e.g., identity and unity, types and instantiation, rigidity, mereology, dependence (GUIZZARDI, 2005). These initiatives concerning the use of ontologies to found conceptual modeling characterizes the research area under the banner of *ontology-based conceptual modeling*.

Founded on these ideas and aiming to provide foundations for conceptual modeling activities, Guizzardi (2005) proposes the Unified Foundation Ontology (UFO). UFO was used as a reference ontology to evaluate a fragment of UML and, based on this analysis, a UML extension for the purposes of conceptual modeling (dubbed OntoUML) has been proposed. Next section presents UFO and OntoUML focusing on relevant aspects to this work.

2.2.2 Conceptual Modeling Founded on UFO

The Unified Foundational Ontology (UFO) is a domain-independent system of categories aggregating results from disciplines such as Analytical Philosophy, Cognitive Science, Philosophical Logics and Linguistics. Over the years, UFO has been successfully employed to analyze the classical conceptual modeling constructs including Object Types and Taxonomic Structures, Part-Whole Relations, Intrinsic and Relational Properties, Weak Entities, Attributes and Datatypes, etc. (GUIZZARDI, 2005). Here we present a fragment of UFO that is relevant for this work. For an in-depth discussion, formal characterization and discussion regarding empirical support for UFO’s categories, see (GUIZZARDI, 2005).

UFO begins with a distinction between *universals* and *individuals*. Universals are patterns of features that can be realized in a number of individuals. For example, John and Mary are individuals that instantiate the universals “Man” and “Woman” respectively. UFO includes a taxonomy of individuals and a taxonomy of universals.

The topmost distinction in the taxonomy of individuals is that between endurants and events. Endurants (as opposed to events) are the individuals said to be wholly present whenever

they are present, i.e., they can endure in time, suffering a number of qualitative changes while maintaining their identity (e.g., a house, a person). Since in this work we are especially interested in a portion of UFO that accounts for structural (as opposed to dynamic) aspects of conceptual modeling, we focus on endurants. Endurants are further classified into *Substantials* and *Moments*. Substantials are existentially-independent endurants (e.g. a person, a forest). A moment, in contrast, is an endurant that *inheres in*, and, therefore, is existentially dependent of, another endurant(s). Moments that are dependent of one single individual are *Intrinsic Moments* (e.g. a person's age) whereas moments that depend on a plurality of individuals are instances of *Relator* (e.g. a marriage, an employment, an enrollment).

Intrinsic moments in UFO are further classified into *Qualities* and *Modes*. Those that are measurable and directly related to some quality structure are termed Qualities (e.g. a car's weight has a measurable value in a one-dimensional structure of positive numbers). In contrast, intrinsic moments not directly related to measurable structures are termed Modes (e.g., a person's skills, intentions, beliefs or symptoms).

These distinctions among individuals are reflected in the taxonomy of universals. Instances of Quality Universals have qualities as instances (e.g., the quality universal "Age" is instantiated by "Mick Jagger's age", "John's age"), instances of Mode Universals have modes as instances (e.g., "Disease" is instantiated by "John's diabetes", and "Skill" is instantiated by "John's programming skills"), instances of Relator Universal have relators as instances (e.g., "Marriage" is instantiated by "John and Mary's Marriage"), and instances of Substantial Universals have substantials as instances (e.g., "Person" is instantiated by "John", "Mary", "Mick Jagger").

The ontological category of Substantial Universal is further specialized according to the ontological notions of identity and rigidity. Substantial universals that carry a uniform principle of identity for their individuals are instances of *Sortal Universal* (e.g., "Person", "Car", "Organization"). In contrast, instances of *Mixin Universal* (or *Non-Sortal Universal*) represent abstractions of properties that are common to instances of various sortals (e.g., the mixin "Insurable Item" describes properties that are common to entities of different sortals such as "House", "Car", "Work of Art"). Moreover, a universal is said to be *rigid* if it classifies its instances necessarily (in the modal sense). In other words, if a universal *T* is rigid, then an instance *x* of *T* cannot cease to be an instance of *T* without ceasing to exist (e.g., "Person", "Organization"). In contrast, a universal is *anti-rigid* if its instances can move in and out of the extension of that universal without ceasing to exist (e.g., "Student", "Insured Organization")².

A *Rigid Sortal* that supplies a uniform principle of identity to its instances is termed *Kind* (e.g. "Person", "Organization"). Instances of Kind may be specialized by other rigid sortals that inherit the principle of identity supplied by the Kind. These rigid sortals are termed *Subkinds* (e.g.

² For the sake of simplicity, here we do not consider semi-rigid types, i.e. types that classify some of its instances necessarily and some of them only contingently (GUARINO; WELTY, 2002b).

“Man”, “Woman”). Anti-rigid sortals are further classified into the categories *Role* or *Phase*. Instances of *Role* classify substantials through the relational properties they bear in the scope of a relational context (e.g. “Employee”, “Employer”, “Husband”, “Student”), and thus are considered externally dependent universals. In contrast, instances of *Phase* classify substantials depending on one or more of their intrinsic properties (e.g. “Child”, “Adult”).

Rigid Mixins that represent abstractions of properties that apply to instances of different kinds are called *Category* universals (e.g., “Legal Entity” abstracting properties of persons and organizations). Analogously to *anti-rigid sortals*, *anti-rigid mixins* are classified into *Role Mixin* (which are externally dependent) and *Phase Mixin*. *Role Mixins* classify substantials of different kinds through common relational properties (e.g. “Customer”, abstracting relational properties applicable to persons and organizations). In contrast, *Phase Mixins* classify substantials of different kinds through common intrinsic properties (e.g. “Living Animal” and “Deceased Animal” that classify in a contingent manner instances of entities of kinds such as “Dog”, “Person”, “Cow”, etc.).

Figure 3 summarizes the discussion so far by depicting a fragment of UFO’s taxonomy of universals in the left-hand side (“Endurant Universal” and its specializations) and the taxonomy of individuals in the right-hand side (“Endurant” and its specializations). This fragment of the UFO ontology is presented here as a UML class diagram for presentation-purposes only. The actual representation of the ontology is captured in (GUIZZARDI, 2005) in a particular type of Intensional Modal Logics with Sortal Quantification.

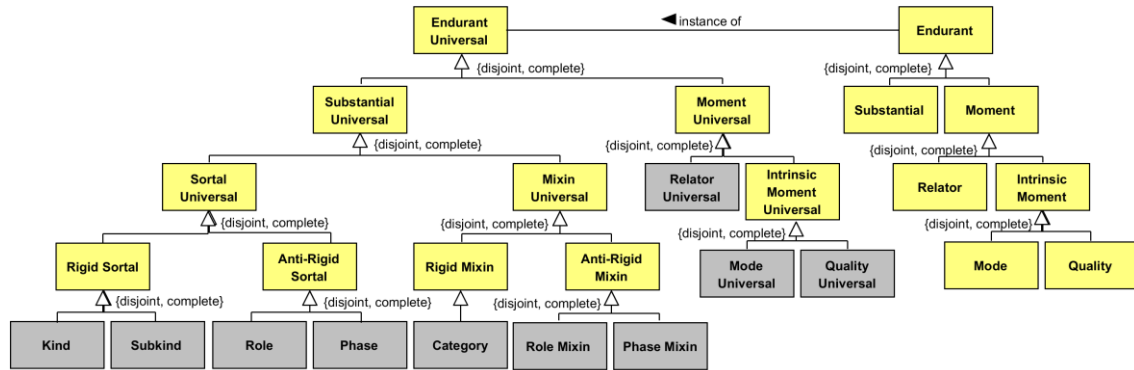


Figure 3 - UFO *endurant individuals* and *universals* taxonomies

In order to support the construction of ontology-driven conceptual models, a UML profile (dubbed OntoUML) was proposed in (GUIZZARDI, 2005). Over the years, it has been adopted by many research, industrial and government institutions worldwide, in areas ranging from Geology to Organ Donation, from Biodiversity Management to Logistics, from Software Engineering to Telecommunications (GUIZZARDI et al., 2015b). It has been also applied in the representation of some core ontologies founded on UFO, such as the organizational ontology O3 (PEREIRA; ALMEIDA, 2014) and the reference ontology for services UFO-S (NARDI et al., 2015). Besides the modeling language itself, the OntoUML approach also offers a model-based

environment for model construction, verbalization, code generation, formal verification and validation (BENEVIDES; GUIZZARDI, 2009). The environment provides a validation strategy based on visual model simulation (BENEVIDES et al., 2011; BRAGA et al., 2010) and offer support for detection, simulation and elimination of anti-patterns (GUIZZARDI; SALES, 2014). Further, automated transformation to OWL (GUIZZARDI; ZAMBORLINI, 2013) and to Alloy are available (BENEVIDES et al., 2011; BRAGA et al., 2010). OntoUML has also been considered as a candidate for addressing the OMG SIMF (Semantic Information Model Federation) Request for Proposal, after a report of its continuous successful use by a branch of the U.S. Department of Defense (U.S. DEPARTMENT OF DEFENSE, 2016).

OntoUML includes modeling primitives that reflect ontological distinctions put forth by UFO (these are represented as stereotypes for each of the leaf ontological categories of the UFO taxonomy of universals). Figure 4 depicts an OntoUML diagram including domain concepts to illustrate each aforementioned UFO notion.

OntoUML also defines stereotypes for associations. Associations stereotyped «characterization» are used to represent relations between substantial universals and intrinsic moment universals, meaning that instances of the former bear instances of the latter. For instance, in Figure 4, associations stereotyped «characterization» are used to capture that each person bears an age and a set of skills. Associations stereotyped «mediation», in their turn, are used to represent relations between relator universals and roles in a relational context. In Figure 4 associations stereotyped «mediation» are used to capture that an employment establishes the relationship between an employee and an employer.

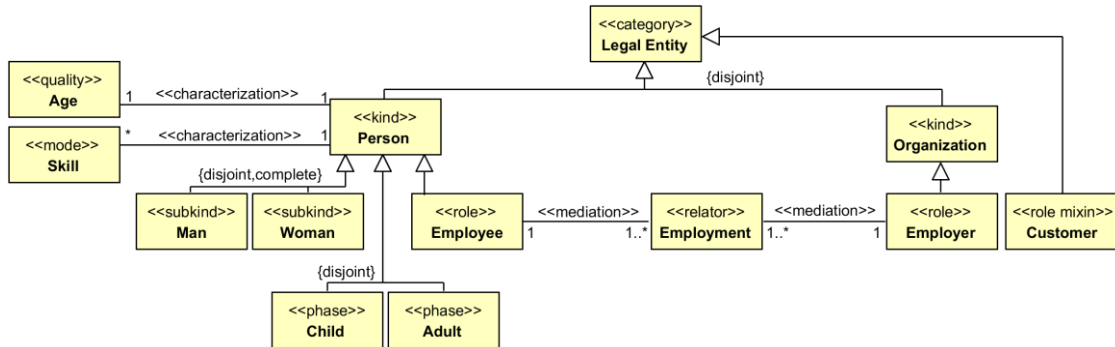


Figure 4 - An OntoUML diagram

In addition to prescribing specialized modeling primitives, OntoUML includes formal syntactic constraints that govern how these primitives can be combined, taking into account the ontological notions of *existential dependence*, *rigidity* and *principle of identity*, among others. Since these constraints are derived from the axiomatization of the foundational ontology, they are considered *semantically-motivated syntactic constraints*. These semantically-motivated syntactic constraints play a key role on guaranteeing the compliance of OntoUML diagrams to UFO rules, and allows the development of tool support to automate the verification of this compliance. We present here a subset of these semantically-motivated syntactic constraints, namely those that

govern how specialization between domain types may be applied to ensure that the resulting conceptual models are sound. The six presented constraints (or “rules”) are relevant for the analysis conducted in Chapter 4.

Considering that *Moment* and *Substantial* are disjoint categories, since no entity can be at the same time existentially-dependent and existentially-independent, instances of *Moment Universal* may not specialize instances of *Substantial Universal* and vice-versa (*rule U1*). The same applies to qualities, modes and relators; these are disjoint ontological categories, and thus instances of *Quality Universal*, *Mode Universal* and *Relator Universal* cannot specialize each other (*rule U2*).

In the case of substantial universals, a further rule arises from the distinction into sortals and mixins (i.e., non-sortals): instances of *Mixin Universal* cannot specialize instances of *Sortal Universal* (*rule U3*). This is because if a mixin universal were to specialize a sortal universal, it would inherit the principle of identity carried by that sortal universal. However, by definition, mixin universals do not carry a principle of identity.

A further constraint arises from the distinction into rigid and anti-rigid universals: rigid universals (instances of *Rigid Sortal* and *Rigid Mixin*) may not specialize instances of anti-rigid universals (*Anti-Rigid Sortal* and *Anti-Rigid Mixin*) (*rule U4*). If a rigid subtype were to specialize an anti-rigid supertype, its instances would instantiate the subtype necessarily (by definition) and would also instantiate the supertype necessarily (by virtue of specialization). However, as we considering an anti-rigid supertype – which applies contingently (i.e., non-necessarily) to its instances – this is not possible by definition. Note that the converse is admissible, and thus an anti-rigid type may specialize a rigid one as no contradiction arises.

Two rules emerge from the source of principle of identity in sortal specialization. First, an instance of *Kind* cannot specialize another sortal universal (*rule U5*). A kind K supplies a principle of identity to its instances, and, if it were to specialize another sortal universal S, it would inherit a principle of identity carried by S, i.e., supplied by S or by another kind that S specializes. As a consequence, K, in this case, would carry more than one principle of identity for its instances. As discussed in depth in (GUIZZARDI, 2005), these instances would become ontologically indeterminate individuals. Second, instances of *Subkind*, *Phase* and *Role* are sortals that carry – but do not supply – a principle of identity for their instances. Hence, they must specialize a unique instance of *Kind* (*rule U6*), which will supply the principle of identity for their instances.

Conceptual domain models constructed in OntoUML are able to express ontological properties of the types that apply to individuals in the subject domain. However, currently, no support is provided to represent domain-specific types of types, since the second-order types of OntoUML are predefined in the language profile (as stereotypes). Due to this limitation, some publications, such as (PEREIRA; ALMEIDA, 2014) and (FALBO et al., 2014) mention higher-order universals in OntoUML models using stereotypes whose semantics is not formally defined.

This motivates our investigation into the combination of UFO with a theory capable of accounting for multiple levels of classification.

2.3 Multi-level Modeling

Multi-level conceptual modeling addresses phenomena dealing with a number of complex notions and subtle relations that cross multiple levels of instantiation. These phenomena are ubiquitous in many application domains, such as, software development (GONZALEZ-PEREZ; HENDERSON-SELLERS, 2006), organizational roles (or professional positions) (PEREIRA; ALMEIDA, 2014), biological taxonomy (MAYR, 1982) and artifact types (e.g., product types) (NEUMAYR; GRÜN; SCHREFL, 2009). Many approaches for multi-level modeling have been proposed, with different purposes and founded on different concepts, as, for example, (ATKINSON; GERBIG, 2012; GONZALEZ-PEREZ; HENDERSON-SELLERS, 2006; LARA; GUERRA; CUADRADO, 2014; NEUMAYR; GRÜN; SCHREFL, 2009; ODELL, 1994; PARTRIDGE, 2005), although, to the best of our knowledge, none of them address ontological distinctions such as *existential dependence*, *rigidity* and *sortality*.

The following sections discuss the main concepts underlying some prominent multi-level modeling approaches in software engineering, which have influenced this work.

2.3.1 Power Types

The concept of *power type* is an early notion used to support the construction of models addressing more than one classification level. Although the notion of power type has had some influence over the majority of the multi-level approaches, different definitions for this term can be found in the literature. The work presented here was strongly influenced by two seminal definitions of power type, namely the proposals of (CARDELLI, 1988) and (ODELL, 1994).

The focus of Cardelli (1988) was on providing a logical theory to define the notion of power type. He considers that types are intentionally (in contrast with extensionally) defined and argues that types are not intended as arbitrary sets of elements, but as sets whose elements share a common structure. Further, he considers that subtypes should not be intended as arbitrary subsets, i.e., the subtypes of a type should be identified considering the structure they define to their instances. Based on these notions, Cardelli coined the notion of *power type* to characterize a type that captures the common structure of all types that specializes a specific type. According to (CARDELLI, 1988), the same way specializations are intuitively analogous to subsets, *power types* can be intuitively understood as powersets. The powerset of a set A is the set whose elements are **all** possible subsets of A including A itself. Thus, “if A is a type, then Power(A) is the type whose elements are **all** the subtypes of A” (including A).

Modeling languages issues were out of the scope of (CARDELLI, 1988). In contrast, Odell (1994) provided an informal definition of power type, and focused on how representing power types in object-oriented models using examples to illustrate his approach. The notion proposed by Odell in (ODELL, 1994) is the most referenced notion of power type in software engineering.

Odell (1994) stated simply that a *power type* is a type whose instances are subtypes of another type. To illustrate it, the author presented an example considering a type “Tree Species” having instances such as “Sugar Maple”, “Apricot”, “American Elm” and “Saguaro”. Since all instances of “Tree Species” specialize “Tree”, “Tree Species” is a *power type* of “Tree”. To facilitate the discussion here, we use the term *base type* to refer to the type that generalizes the *power type* instances, i.e., “Tree” is the *base type* of “Tree Species”.

Concerning the representation of *power types* in object-oriented conceptual models, Odell proposed the use of an association between a *base type* and a *power type* to express that each instance of the *power type* “classifies” zero or more instances of the *base type* and that each instance of the *base type* “is classified by” one instance of the *power type*. Figure 5 illustrates it representing the relation between “Tree” and “Tree Species”.

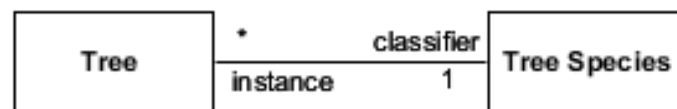


Figure 5 - Illustrating the notation proposed by Odell to represent that each instance of the *base type* is instance of one instance of the *power type* (adapted from (ODELL, 1994))

Odell has also proposed labeling the “subtype partitions” (generalization sets) of the base type as a way to identify which specializations of the *base type* are instances of each *power type*. For example, in order to identify that “Sugar Maple” and “Apricot” are instances of “Tree Species” the generalization set was labeled “:Tree Species” (see Figure 6). The support for power types of the current version of UML (OMG, 2011) is inspired in Odell’s original proposal.

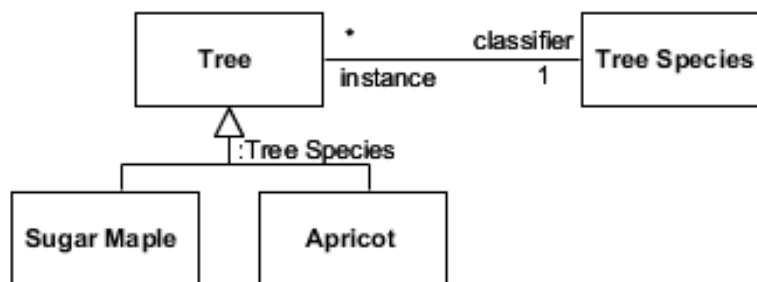


Figure 6 - Illustrating the notation proposed by Odell to associate *power types* with subtypes partitions (adapted from (ODELL, 1994))

Concerning the different notions of *power type*, it is important to notice that Odell’s definition is less strict than Cardelli’s definition (CARDELLI, 1988). Cardelli coins the *power set* concept stating that **all** the specializations of the *base type* are instances of the *power type*. Odell’s definition, in turn, does not comply with that restriction assuming that all instances of the *power type* specializes the *base type* but there may be specializations of the *base type* that are not

instances of the *power type*. For example, assuming that there may be specializations of “Tree” that are not instances of “Tree Species”, according to Cardelli’s *power type* definition “Tree Species” is not the power type of “Tree”, in contrast with Odell’s notion. Thus, as pointed out by (HENDERSON-SELLERS, 2012), the relation defined by Odell is misnamed *power type* since, in fact, it denotes a strict *subset* of the *power set*. An important consequence of this difference concerning the two definitions is that, according to Cardelli’s definition, each type has at most one power type, while, following Odell’s definition, a type may have more than one *power type*.

2.3.2 *Clabjects* and Deep Instantiation

The concept of power type is founded on the notion that “instances of types can also be types” (ODELL, 1994). While this notion can serve as a foundation for a multi-level approach, the power type pattern has often been used in two-level architectures, in which a model element is classified exclusively as an object (instance) or as a class (type). In this case, one needs to admit two entities: one capturing the class facet and another capturing the object facet of the power type. Further, some strategy to link the two facets must be adopted. For example, Gonzalez-Perez and Henderson-Sellers denote this informally in (GONZALEZ-PEREZ; HENDERSON-SELLERS, 2006) by drawing ellipses on top of diagrams to relate the two facets of a power type.

Atkinson and Kühne (ATKINSON; KÜHNE, 2000) propose a more radical revision of two-level architectures coining the notion of *clabject*. This notion is founded on the observation that every instantiable entity has both a type (or class) facet and an instance (or object) facet which are equally valid (ATKINSON; KÜHNE, 2000). For example, “Sugar Maple” could be considered a *clabject* since it has both an instance facet (it is instance of “Tree Species”) and a type facet (some instances of “Tree” are also instances of “Sugar Maple”). Thus, instances of power types can be considered *clabjects*. These notions are central to the multi-level theory we propose in this work

Atkinson and Kühne also recognize an important characteristic of multi-level domains: in such domains an element at some level can describe features of elements at each level beneath that level (ATKINSON; KÜHNE, 2001, 2008). Founded on this observation, they propose the notion of *deep instantiation* as a contrast to the tradition on object-oriented community of considering that a class can only define the properties of its direct instances (which they call “shallow instantiation” in (ATKINSON; KÜHNE, 2001)).

Besides adhering to the notion of *clabject*, the authors define the concept of *potency* (ATKINSON; KÜHNE, 2008). Deep instantiation is based on the idea of assigning a *potency* to every model element, i.e. to each *clabject*, attribute or association is assigned a potency. The potency of a model element is an integer that defines the depth to which a model element can be instantiated. When a *clabject* is instantiated from another *clabject* the potencies of the created *clabject* and of its *fields* and *association* are given by the original *clabject*, *fields* and *association*

potencies decremented by one. Objects have potency equal to zero indicating they cannot be instantiated, i.e. a clabject with potency zero corresponds to an object, if the potency of a *field* becomes zero then a value can be assigned to that *field*, and an association with potency zero corresponds to a link.

Further, every model element has a *level*. The level of a model element is an integer representing the model level in which the element resides (e.g. an M_0 element has *level* value 0). As well as the potency of an element, the element level value is also reduced by 1 when the element is instantiated. Thus, instantiation can only be applied to model elements whose potency and level are greater than 0. Moreover, elements whose potency is zero cannot be instantiated, regardless of their level value.

For example, suppose a scenario in which we want to model computer monitors and their models, such that each instance of “Monitor” is instance of one instance of “Monitor Model”, i.e. following (ODELL, 1994) “Monitor Model” is power type of “Monitor”. Further consider that each “Monitor” has a “serial number”, that each “Monitor Model” defines the “screen size” of its instances and that “Dell E1913” and “Dell E2216” are instances of “Monitor Model”.

Applying the *deep instantiation approach* (ATKINSON; KÜHNE, 2001, 2008) to capture this scenario we could define a *clabject* “Monitor Model” with potency 2. The attribute “screen size” could be defined as a field of “Monitor Model” with potency 1 denoting that each instance of “Monitor Model” has a specific value assigned to “screen size”. Finally, we could define “serial number” as a field of “Monitor Model” with potency 2, so that instances of “Monitor Model” would be *clabjects* in which “serial number” field would have potency of 1. Thus, instances of instances of “Monitor Model” would have a value assigned to “serial number”, since its potency would reach zero. For example, “Dell E1913” is an instance of “Monitor Model” with “screen size” 19” and “myMonitor” is an instance of “Dell E1913” with “1234” as its “serial number”. Figure 7 depicts this example, which is a simplified version of an example presented in (ATKINSON; KÜHNE, 2008).

Note that the *deep instantiation approach* allowed us to omit the representation of the “Monitor” type from our model since we were able to capture its attributes as fields of “Monitor Model” having potency 2. This illustrates what the authors consider to be one of the main benefits of the *deep instantiation-based approach*: the possibility of reducing “accidental complexity” in domain models since it supports multi-level modeling without the need of introducing types to the models only “because of the idiosyncrasies of a particular solution to deep characterization” (ATKINSON; KÜHNE, 2008).

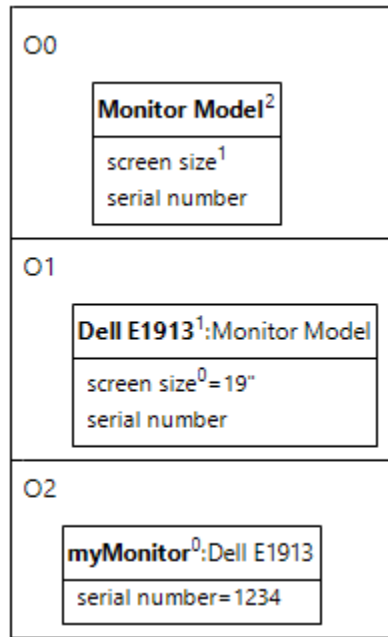


Figure 7 - Illustrating the deep instantiation approach using a deep-modeling tool called *Melanee* (ATKINSON; GERBIG, 2012)

Some formalizations of approaches based on deep instantiation can be found in the literature (e.g. (KENNEL, 2012; ROSSINI et al., 2014)), as well as proposals of modeling languages and tools to support deep modeling (ATKINSON; GERBIG, 2012; DE LARA; GUERRA, 2010).

2.3.3 Ontological Instantiation and Linguistic Instantiation

Atkinson and Kühne propose in (ATKINSON; KÜHNE, 2003) the Orthogonal Classification Architecture (OCA) to address the need of considering two different kinds of instantiation: the “linguistic instantiation” and the “ontological instantiation”. Whereas linguistic instantiation is used to define the relations between domain entities and linguistic constructs, ontological instantiations relate domain entities to other domain entities. Figure 8 illustrates the application of OCA defining two linguistic levels (L1 and L0) and three ontological levels (O2, O1 and O0).

In Figure 8, the level L1 accommodates the metamodel of a modeling language and models constructed in such language are considered to be at level L0. Thus, the elements in level L1 define the modeling language constructs and each element in level L0 is a “linguistic” instance of an element in L1. Besides the “linguistic” instantiation relations between elements in L0 and elements in L1, Figure 8 captures the “ontological” classifications that occur between elements in a common “linguistic” level. An “ontological” instantiation is used to represent the relation between a domain element and a domain category that classify such element. For example, Figure 8 captures the fact that “Lassie” (a real-world dog represented in a model) is an “ontological” instance of “Collie” (a domain category represented in the same model) which, in its turn, is an

“ontological” instance of “Breed”. Since our focus is on conceptual modeling (and not language engineering or language metamodeling), this work is focused on “ontological” instantiation.

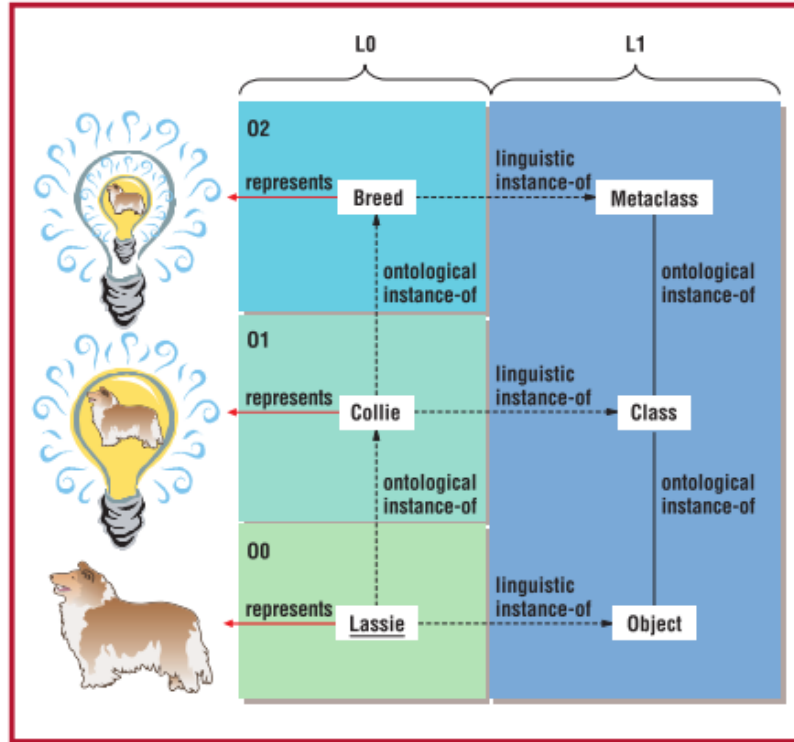


Figure 8 - The Orthogonal Classification Architecture (extracted from (ATKINSON; KÜHNE, 2003)).

2.4 Final Considerations

In this chapter, we have discussed some key developments of the last decades in *ontology-based conceptual modeling* and in *multi-level modeling*.

Following authors such as (GUARINO, 1998; GUIZZARDI, 2005; WAND; WEBER, 1993) we argued that an appropriate conceptual modeling language should provide modeling primitives that reflect the conceptual categories defined in an ontology, characterizing what we call *ontology-based conceptual modeling*. We have discussed the use of the Unified Foundational Ontology (UFO) (GUIZZARDI, 2005) as a basis for ontology-based conceptual modeling, concluding that, so far, its support to ontology-based conceptual modeling has been focused on the analysis of properties of categories of individuals. As a consequence, this approach is unable to account for subject domains in which the categorization scheme itself is part of the subject matter.

Multi-level modeling, in its turn, has not yet received the required conceptual treatment. As we have seen, the debate concerning multi-level modeling in software engineering is mainly focused on the use of models as engineering artifacts, especially in the context of model-driven engineering. Nevertheless, they provide important notions, such as power types and clabjects, which will be explored later in our multi-level conceptual modeling theory.

With these observations in mind, the general goal of this work can be seen under two complementary points of view, as we aim to: (i) extend the ontology-based conceptual modeling foundations to consider multiple classification levels; and (ii) contribute to the multi-level modeling field by proposing a multi-level approach that addresses ontological distinctions.

The first step towards our multi-level ontology-based conceptual modeling approach is the development of a formal theory that captures the conceptualization underlying multi-level modeling. Next chapter presents this theory.

Chapter 3. A Formal Theory for Multi-Level Conceptual Modeling

There is ample psychological evidence to support the hypothesis that humans conceive the physical and social world using some notion of “categories” and use categorization or classification strategies since a pre-language age of 3-4 months (see (GUIZZARDI, 2005), pp. 114-118). Thus, it is no surprise that a vast majority of conceptual modeling techniques are based on notions such as “class” and “type”, and that multi-level modeling approaches recur to “types” of “types” to support the representation of domains dealing with multiple classification levels. Despite this common ground of most multi-level modeling approaches, there is still no consensus on what kinds of constructs and concepts provide the best support for it.

In the last decades many approaches for multi-level modeling have been proposed, with different purposes and founded on different concepts (LARA; GUERRA; CUADRADO, 2014; NEUMAYR; SCHREFL; THALHEIM, 2011), most of them focused on proposing modeling languages and model-driven development approaches. We believe the multi-level modeling area can benefit from a theory capturing the nature of multi-level domains, independent of language and technological issues. This theory could be applied to provide semantic foundations in activities such as the design of new multi-level conceptual modeling languages and approaches, the redesign of existing languages/approaches and the harmonization of approaches founded on different concepts.

Aiming to provide a theory that achieves these desiderata, we develop a formal theory for multi-level modeling called MLT. MLT is built up from a basic *instantiation* relation and characterizes the concepts of *individuals* and *types*, with types organized in levels related by instantiation. The theory defines relations that occur between entities in the same level, as well as relations that may hold between elements of different levels. Further, MLT accounts for attributes and relationships (CHEN, 1976), providing support to discuss how an element at some level can influence features of elements at a lower level. The theory is formally defined through axiomatization in first-order logic, and verified and validated applying a lightweight formal method.

Although we do not propose a language for multi-level conceptual modeling, we explore patterns that emerge from the application of the theory, as well as modeling constraints to ensure that multi-level models respect the theory axioms, clearly suggesting semantically-motivated syntactic constraints to multi-level modeling languages founded on MLT. Since our focus is on conceptual modeling (and not language engineering or language metamodeling), we address

“ontological instantiation” and we are thus unconcerned with “linguistic instantiation” (ATKINSON; KUHNE, 2003).

This chapter is further organized as follows. Section 3.1 presents a set of requirements we believe that a theory should meet to be used as a reference theory for multi-level conceptual modeling. Section 3.2 presents the basic types of MLT, which play a key role in characterizing the classification levels. Section 3.3 presents the definitions of the intra-level structural relations, i.e. the relations that may occur between elements in the same classification level. Section 3.4, in its turn, discusses the cross-level relations, i.e., the relations that may occur between elements in different classification levels. Section 3.5 illustrates the application of the theory to the domain of biological taxonomy. Section 3.6 presents the MLT accounts for attributes and relationships. Section 3.7 discusses how MLT addresses dynamic classification (a required feature to enable the combination of MLT with UFO). Section 3.8 presents some remarks on the identity conditions of types. Section 3.9 discusses how MLT achieves the defined requirements. Section 3.10 positions MLT with respect to related work, and Section 3.11 presents some final considerations.

3.1 Requirements for a Multi-level Conceptual Modeling Theory

We establish here six requirements we judge important for a multi-level modeling theory to provide an adequate support to the task of describing multi-level domains. We indicate sources in the literature that have already established similar requirements to corroborate the relevance of the requirements identified here.

An essential requirement for a multi-level modeling theory is *to account for entities of multiple (related) classification levels*, capturing chains of instantiation between the involved entities (**R1**). To meet this requirement, the theory must admit entities that are, simultaneously, type (class) and instance (object) (ATKINSON; KÜHNE, 2000).

Another key requirement for a multi-level modeling theory is *to define principles for the organization of entities into levels, clearly characterizing the nature of entities populating each level* (**R2**). These principles should guide the adequate use of classification (instantiation) relations. The lack of principles to guide the organization of entities into levels may lead to the construction of unsound multi-level models. For example, in (BRASILEIRO et al., 2016a) we assessed Wikidata and found over 22,000 violations of the strict metamodeling principle. The identified problems seem to arise from inadequate use of instantiation and subtyping, and could have been prevented with guidance from the editing/modeling environment.

Considering that the number of levels specified in a conceptual model may vary according to the nature of the phenomena being captured and to the model purposes, a multi-level theory should *allow an arbitrary number of classification levels* (**R3**). The ability to deal with an

arbitrary number of levels is pointed out by authors such as (ATKINSON; GERBIG, 2012; FRANK, 2014), as a key requirement for multi-level modeling approaches in the context of language engineering.

An important characteristic of domains spanning multiple levels of classification is that there are rules that apply to the instantiation of types of different levels. This kind of rule is present in an early and important approach for multi-level modeling, named the *power type pattern* (CARDELLI, 1988; ODELL, 1994), which establishes a relationship between two types such that the instances of a type (the so-called “powertype” or “higher-order” type) are specializations of a lower-level type (the so-called “base type”). For example, all instances of *Mobile Phone Model* (e.g. *Iphone5* and *Zenfone2*) specialize the base type *Mobile Phone*. In order to represent *Mobile Phone Model*, we need to establish its relation with the *Mobile Phone* type (we call this sort of relation a *structural relation*, governing the instantiation of types at different levels). Further, one may need to represent whether an instance of *Mobile Phone* may instantiate: (i) only one, or (ii) more than one *Mobile Phone Model*. The biological taxonomy domain is rich in rules concerning instantiation of types at different levels. For example, it defines that the instances of *Biological Taxonomic Rank* obey a sort of subordination chain such that every instance of *Phylum* specializes one instance of *Kingdom*, every instance of *Class* specializes one instance of *Phylum*, and so on. In order to capture these nuances of multi-levels domains, a multi-level theory *should precisely define structural relations that account for rules for the instantiation of types at different levels* (R4).

In conceptual modeling, types can be seen as entities that capture common features of other entities which are considered their instances. These features are often captured using the notions of *attributes* and *relationships* (CHEN, 1976). A recurrent phenomena in domains dealing with multiple classification levels is that features of types in one classification level may constrain features in one level lower. For example, considering that every mobile phone has a screen we may define screen size as a feature that characterizes mobile phones. Further, consider that mobile phones models categorize the mobile phones with respect to some features, including the screen size. In this scenario, the screen size defined by an instance of “Mobile Phone Model” constrains the feature “screen size” of all its instances (e.g. defining that *Iphone5* have screen size of “4-inch” means that every instance of *Iphone5* must have a 4-inch screen). To be able to capture this phenomenon, an appropriate multi-level modeling theory *should provide an account for features (attributes and relationships) of entities, including support to rules relating features of entities in different levels* (R5). An example of multi-level modeling approach that provides a mechanism to capture a specific sort of relations between attributes of entities in different levels is the one proposed in (ATKINSON; GERBIG, 2012) (this mechanism is further discussed in Section 3.10.2.)

Finally, in various domains, there are relations that 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 with instances of *Car*, i.e., individuals that are instances of instances of *Car Model*. To capture scenarios like this, a *multi-level modeling theory should admit the existence of domain relations between entities in different classification levels (R6)*. Next sections present a formal multi-level theory we developed observing requirements R1 to R6.

3.2 MLT Foundations: Basic Types and the Instantiation Relation

The notions of *type* and *individual* are central for our multi-level modeling theory. *Types* are predicative entities that can possibly be applied to a multitude of entities (including types themselves). Particular entities, which are not types, are considered *individuals*.

Each type is characterized by an *intension*, which is used to judge whether the type applies to an entity (e.g., whether something is a *Person*, a *Dog*, a *Chair*) (it is also called *principle of application* in (GUIZZARDI, 2005)). If the intension of a type t applies to an entity e then it is said that e is an *instance of* t . Thus, the *instance of* relation (or *instantiation relation*³) maps a type to the entities that fall under the type. The set of instances of a type is called the *extension* of the type (HENDERSON-SELLERS, 2012). We assume that the theory is only concerned with types with non-trivially false intensions, i.e., with types that have possible instances in the scope of the conceptualization being considered.

MLT is formalized in first-order logic, quantifying over all possible individuals and types. The *instantiation relation* is formally represented by a binary predicate $iof(e, t)$ that holds if an entity e is instance of an entity t (denoting a type). For instance, the proposition $iof(Vitória, City)$ denotes the fact that “Vitória” is an instance of the type “City”.⁴

We build up the theory axiomatization defining the conditions for entities to be considered *individuals*, with the constant “Individual” in axiom A1. An entity is an instance of “Individual” iff it does not possibly play the role of type in instantiation relations.

$$\forall x \text{ iof}(x, \text{Individual}) \leftrightarrow \nexists y \text{ iof}(y, x) \quad (\text{A1})$$

³ We are aware that certain approaches such as RM-ODP distinguish the terms *instantiation* and *instance*, but this distinction is not required here, and hence we use the terms interchangeably.

⁴ For the sake of clarity in the presentation, we focus in this section on types that apply necessarily to their instances (the so-called rigid types (GUIZZARDI, 2005)). A treatment of dynamic classification (and non-rigidity) is deferred to Section 3.7.

We consider that two types are equal iff the sets of all their possible instances are the same (see Axiom A2). Note that this definition of equality only applies to elements which are not *individuals*, hence the ‘guard’ conditions on the left-hand side of the implication.

$$\begin{aligned} \forall t_1, t_2 \left(\neg \text{iof}(t_1, \text{Individual}) \wedge \neg \text{iof}(t_2, \text{Individual}) \right) \rightarrow \\ ((t_1 = t_2) \leftrightarrow (\forall e \text{ iof}(e, t_1) \leftrightarrow \text{iof}(e, t_2))) \quad (\text{A2}) \end{aligned}$$

As a multi-level modeling theory, we deal with *types* that have *individuals* as instances as well as with types whose extension is composed by other types. In order to accommodate these varieties of types, the notion of *type order* is used. Types whose instances are individuals are called *first-order types*. Types whose instances are *first-order types* are called *second-order types*. Those types whose extensions are composed by *second-order types* are called *third-order types*, and so on. We use the term *higher-order type* to refer to types with order higher than one.

Axiom A3 characterizes “First-Order-Type” (or shortly “1stOT”), defining a first-order type as an entity whose instances are instances of “Individual”. Analogously, A4 and A5 characterize “Second-Order Type” (or “2ndOT”) and “Third-Order Type” (“3rdOT”). A4 defines that an entity t is a second-order type iff all its instances are first-order types (i.e., instances of “1stOT”), and A5 defines that an entity t is a third-order type iff all its instances are second-order types (i.e., instances of “2ndOT”).

This scheme can be simply extended to consider as many orders as necessary. However, we present our theory here for the sake of brevity considering only *first-order*, *second-order* and *third-order types*.

$$\forall t \text{ iof}(t, \text{1stOT}) \leftrightarrow (\exists y \text{ iof}(y, t) \wedge (\forall x \text{ iof}(x, t) \rightarrow \text{iof}(x, \text{Individual}))) \quad (\text{A3})$$

$$\forall t \text{ iof}(t, \text{2ndOT}) \leftrightarrow (\exists y \text{ iof}(y, t) \wedge (\forall t' \text{ iof}(t', t) \rightarrow \text{iof}(t', \text{1stOT}))) \quad (\text{A4})$$

$$\forall t \text{ iof}(t, \text{3rdOT}) \leftrightarrow (\exists y \text{ iof}(y, t) \wedge (\forall t' \text{ iof}(t', t) \rightarrow \text{iof}(t', \text{2ndOT}))) \quad (\text{A5})$$

Substituting t by *Individual* in axiom A3, one can see that “Individual” is an instance of “1stOT” (theorem T1). Analogously, using further axioms A4 and A5 we can show that “1stOT” is instance of “2ndOT” and “2ndOT” is instance of “3rdOT” (see theorems T2 and T3).

$$\text{iof}(\text{Individual}, \text{1stOT}) \quad (\text{T1})$$

$$\text{iof}(\text{1stOT}, \text{2ndOT}) \quad (\text{T2})$$

$$\text{iof}(\text{2ndOT}, \text{3rdOT}) \quad (\text{T3})$$

Theorem T4 states that “Individual”, “1stOT”, “2ndOT” and “3rdOT” have no instances in common, i.e., their extensions are disjoint. To see why this theorem holds, we need to analyze all the possible combinations of the basic types in pairs, starting from evaluating the possibility for an entity to be instance of both “Individual” and “1stOT”. According to A1, instances of “Individual” do not have instances, while according to A3 instances of “1stOT” necessarily have

some instance. Thus, no entity can be an instance of “Individual” and “1stOT” simultaneously. Using A1 in tandem with A4 and A5 we can also conclude that “Individual” does not have instances in common with “2ndOT” nor with “3rdOT”. Now, suppose an entity e , which is instance of both “1stOT” and “2ndOT”. Using A3 and A4, all its instances should be simultaneously “Individual” and “1stOT”, which is impossible, as we have already concluded. Hence, there are no entities which simultaneously instantiate “1stOT” and “2ndOT”. Following analogous reasoning and using axioms A4 and A5 one can conclude that “2ndOT” and “3rdOT” do not have instances in common. Finally, applying the same strategy and using axioms A3 in tandem with A5 one can see that “1stOT” and “3rdOT” have no entities in common.

$$\begin{aligned} & \neg x \left(\text{iof}(x, \text{Individual}) \wedge \text{iof}(x, \text{1stOT}) \right) \vee \left(\text{iof}(x, \text{Individual}) \wedge \text{iof}(x, \text{2ndOT}) \right) \vee \\ & \left(\text{iof}(x, \text{Individual}) \wedge \text{iof}(x, \text{3rdOT}) \right) \vee \left(\text{iof}(x, \text{1stOT}) \wedge \text{iof}(x, \text{2ndOT}) \right) \vee \\ & \left(\text{iof}(x, \text{1stOT}) \wedge \text{iof}(x, \text{3rdOT}) \right) \vee \left(\text{iof}(x, \text{2ndOT}) \wedge \text{iof}(x, \text{3rdOT}) \right) \quad (\text{T4}) \end{aligned}$$

Axiom A6 states that each entity in our domain of enquiry is necessarily an instance of “Individual”, “1stOT”, “2ndOT” or “3rdOT” (except “3rdOT” whose type is outside the scope of the formalization). This makes the set of extensions of “Individual”, “1stOT”, “2ndOT” and “3rdOT” a partition of the set of entities considered in the theory (and their union the domain of quantification).

$$\forall x \left(\text{iof}(x, \text{Individual}) \vee \text{iof}(x, \text{1stOT}) \vee \text{iof}(x, \text{2ndOT}) \vee \text{iof}(x, \text{3rdOT}) \right) \vee (x = \text{3rdOT}) \quad (\text{A6})$$

Axioms A1 to A6 prescribe a strictly stratified organization of entities into orders. As a result, the *instance of* relation in MLT is asymmetric (i.e. irreflexive and antisymmetric) (Theorem T5) and anti-transitive (Theorem T6). These properties of instantiation relations are consistent with those widely accepted in the conceptual modeling community (HENDERSON-SELLERS, 2012; KÜHNE, 2009).

$$\neg \exists x, y \left(\text{iof}(x, y) \wedge \text{iof}(y, x) \right) \quad (\text{T5})$$

$$\neg \exists x, y, z \left(\text{iof}(x, y) \wedge \text{iof}(y, z) \wedge \text{iof}(x, z) \right) \quad (\text{T6})$$

To see that T5 and T6 hold, one needs to observe that the stratification prescribed by axioms A1 to A6 guarantees that instantiation relations hold between two elements such that the latter is one order higher than the former. Thus, the instances of an entity are in one order lower than it, while its types are in one order higher.

To demonstrate the validity of T5 we follow a case based strategy considering all possible cases for entities in the domain of quantification according to A6:

- First, suppose y is an instance of “Individual”. Since instances of “Individual” do not have any possible instance (A1), $\text{iof}(x, y)$ is never true. Thus, T5 holds for this case.
- Suppose y is an instance of “1stOT”. According to A3, x must be an instance of “Individual” to make $\text{iof}(x, y)$ true. Since instances of “Individual” do not have any possible instance (A1), $\text{iof}(y, x)$ is never true. Thus, T5 holds for this case.

- Suppose y is an instance of “2ndOT”. According to A4, x must be an instance of “1stOT” to make $iof(x, y)$ true. If x is instance of “1stOT”, all its instances must be instances of “Individual” (A3), requiring y to be an instance of “Individual” to make $iof(y, x)$ true. Since y cannot be simultaneously instance of “2ndOT” and “Individual” (T4), T5 holds. The case in which y is an instance of “3rdOT” is analogous to this one.
- Finally, suppose that y is “3rdOT”. To see why $iof(y, x)$ is never true, we can consider all cases for x according to A6. If x is an instance of “Individual”, $iof(y, x)$ is false (A1). If x is an instance of “1stOT”, y would have to be an instance of “Individual” to make $iof(y, x)$ true. However, this is not possible, as instances of “Individual” do not have any possible instance (A1), and “3rdOT” does (T3). If x is an instance of “2ndOT”, y would have to be an instance of “1stOT” (A4) to make $iof(y, x)$ true. Being y an instance of “1stOT”, every instance of it would be an instance of “Individual” (A3). However, since y is “3rdOT”, its instances should be instances of “2ndOT” (A5). This is not possible, given T4. The case in which x is an instance of “3rdOT” is analogous. If x is “3rdOT”, y would have to be instance of “3rdOT” to make $iof(y, x)$ true (A5). Being y an instance of “3rdOT”, every instance of it would be an instance of “2ndOT”, which is impossible, considering that “3rdOT” and “2ndOT” have no instances in common (T4).

To demonstrate the validity of T6, we follow a case-based analysis similar to one we used to analyze T5.

- First, suppose z is an instance of “Individual”. Since instances of “Individual” do not have any possible instance (A1), $iof(y, z)$ is never true. Thus, T6 holds for this case.
- Suppose z is an instance of “1stOT”. According to A3, y must be an instance of “Individual” to make $iof(y, z)$ true. Since instances of “Individual” do not have any possible instance (A1), $iof(x, y)$ is never true. Thus, T6 holds for this case.
- Suppose z is an instance of “2ndOT”. According to A4, y must be an instance of “1stOT” to make $iof(y, z)$ true. If y is instance of “1stOT”, x must be an instance of “Individual” to make $iof(x, y)$ true (A3). Being z an instance of “2ndOT” and x an instance of “Individual”, $iof(x, z)$ is never true. Thus, T6 holds for this case. The case in which z is an instance of “3rdOT” is analogous to this one.
- Finally, suppose that z is “3rdOT”. In this case, to make $iof(y, z)$ true, y must obviously be an instance of “3rdOT”. If y is instance of “3rdOT”, x must be an instance of “2ndOT” to make $iof(x, y)$ true (A5). Being x an instance of “2ndOT”, it cannot be instance of “3rdOT” (T4). Thus, $iof(x, z)$ is never true and T6 holds in this case.

Note that the notion of order we have used is inspired on the ramified hierarchy introduced by Russell in his type theory (COQUAND, 2014). However, Russell’s main goal with the notion of order was to prevent circularity in the hierarchy of types and hence sets of a given order could include sets of an arbitrary lower order. Differently from Russell, in our theory a type can only

have instances at the immediately lower order, resulting in levels of entities. This is a common feature of the *instance of* relation in various techniques which adopt the so-called *strict metamodeling* principle (ATKINSON; KÜHNE, 2000). Further, stratified levels arise from the cascaded application of the power type pattern starting from first-order types.

Figure 9 illustrates the elements that form the basis for our multi-level modeling theory, using a notation that is largely inspired in UML. We use the UML class notation to represent the *basic types of the theory* (“Individual”, “1stOT”, “2ndOT” and “3rdOT”). We use associations as usual to represent relations between instances of the related types. The multiplicities of the associations reflect the constraints in the formalization. For example, each instance of “Individual” is *instance of* at least one instance of “1stOT”, and, on the inverse direction, each instance of “1stOT” has at least one instance of “Individual” in its extension. We use dependencies (dashed arrows) to represent when relations hold between the types, with labels to denote the names of the predicates that apply. For instance, a dashed arrow labeled *iof* between “Individual” and “1stOT” represents that the former is an instance of the latter (i.e., that $iof(Individual, 1stOT)$ holds). In Figure 9 the dashed arrows are justified by theorems T1-T3. The notation used to elaborate Figure 9 is used in all further diagrams in this chapter.

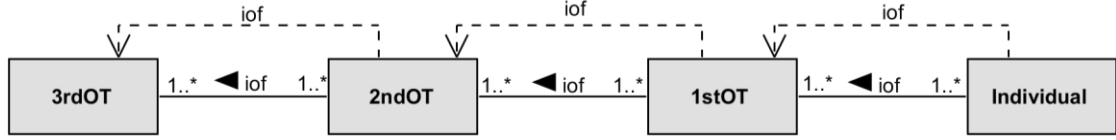


Figure 9 - Basic foundations of MLT: *basic types* and *instance of* relations.

3.3 Intra-level Structural Relations

MLT defines some relations that occur between types of the same order (the intra-level structural relations). All these definitions are based on the instantiation relation.

3.3.1 Specialization and Proper Specialization Relations

We start with the ordinary specialization between types. Definition D1 defines that $t1$ *specializes* $t2$ iff all instances of $t1$ are also instances of $t2$. Since instances of “Individual” do not have instances (A1), D1 states that specialization only applies to elements that are not *individuals* (i.e. elements that have some possible instances). As discussed in (HENDERSON-SELLERS, 2012; KÜHNE, 2009), *specialization* is a partial order relation (i.e., a reflexive, transitive and antisymmetric relation), which is guaranteed in this theory.

$$\forall t1, t2 \text{ specializes}(t1, t2) \leftrightarrow (\exists y \text{ iof}(y, t1) \wedge (\forall e \text{ iof}(e, t1) \rightarrow \text{iof}(e, t2))) \quad (D1)$$

According to D1, every type specializes itself. Since this may be undesired in some contexts, we define the *proper specialization* relation (we use the qualifier ‘proper’ as in ‘proper

subset' considering that the extension of the specialized type is a proper subset of the extension of the general type (HENDERSON-SELLERS, 2012)). Definition D2, thus, defines that $t1$ *proper specializes* $t2$ iff $t1$ *specializes* $t2$ and $t1$ is different from $t2$.

$$\forall t1, t2 \text{ properSpecializes}(t1, t2) \leftrightarrow (\text{specializes}(t1, t2) \wedge t1 \neq t2) \quad (D2)$$

Insofar as the instances of a type are defined by its *intension*, the *proper specialization* relation reflects the fact that the *intension* of the specializing type keeps the constraints stated by the *intension* of the specialized type and adds some other constraint(s) to it, some “additional classification criteria”. To put it more formally, consider two types, t and t' . If t' *proper specializes* t , this means that the intension of t' is given by the conjunction of the intension of t and a predicate that captures the additional classification constraints defined by t' with respect to t . Note that, since we consider there is no relevant type without possible instances, the resultant intension of t' cannot be a trivially false predicate. For example, consider that “Man” is a type that applies to every instance of “Person” of the male gender. Assuming this, the intension of “Man” is given by the conjunction between the intension of “Person” and being male. Thus, in this case, “Man” *proper specializes* “Person”, adding gender as a classification criterion.

Figure 10 augments Figure 9 by including the representation of specialization and proper specialization relations. Note that the axioms presented thus far guarantee that these relations may only hold between types of the same order, which is reflected in the diagram.

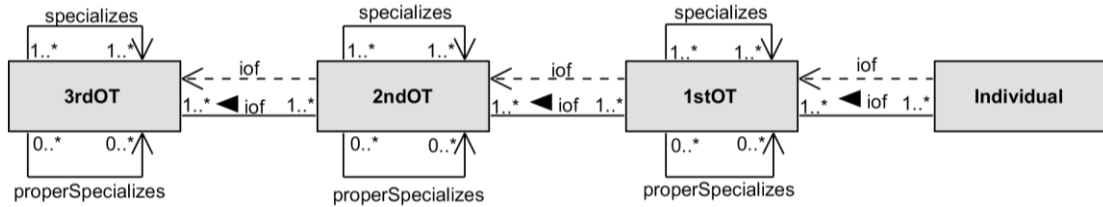


Figure 10 - Intra- level structural relations: specialization and proper specialization.

Substituting $t2$ for *Individual* in definition D1 and comparing the right-hand side of the resultant proposition with the right-hand side of axiom A3, we conclude that *an entity is instance of “1stOT” iff it specializes “Individual”* (theorem T7). Analogously, it follows from D1 and A4 that *an entity is instance of “2ndOT” iff it specializes “1stOT”* (theorem T8). Finally, from D1 and A5 one can see that *every instance of “3rdOT” specializes “2ndOT”* (theorem T9). Therefore, an important consequence of the theory presented so far is that any instance of a higher-order type (any instance of “1stOT”, “2ndOT”, and “3rdOT”) specializes the basic type at an immediately lower order.

$$\forall t \text{ iof}(t, 1stOT) \leftrightarrow \text{specializes}(t, \text{Individual}) \quad (T7)$$

$$\forall t \text{ iof}(t, 2ndOT) \leftrightarrow \text{specializes}(t, 1stOT) \quad (T8)$$

$$\forall t \text{ iof}(t, 3rdOT) \leftrightarrow \text{specializes}(t, 2ndOT) \quad (T9)$$

This leads to a basic pattern in the theory: Every type that is not a basic type (e.g., a domain type) is an instance of one of the basic higher-order types (“1stOT”, “2ndOT”, and “3rdOT”), and, at the same time, specializes the basic type at the immediately lower level (respectively, “Individual”, “1stOT”, “2ndOT”). For example, consider the enterprise domain, in which we may need a type to capture the concept of “Employee”. The type “Employee” classifies *individuals* (e.g. John or Mary), i.e., every *instance of* “Employee” is also *instance of* “Individual”. Thus, by axiom A3, we have that “Employee” *is instance of* “1stOT” and, considering T7, “Employee” *specializes* “Individual”. In fact, since “Employee” and “Individual” are different types, we can say that “Employee” *proper specializes* “Individual”. This basic pattern is illustrated in Figure 11. In order to preserve the intuition in the representation, we used the conventional UML notation to represent specializations (in this case to represent the fact that the proposition *properSpecializes*(Employee, Individual) holds). We have used the instance specification notation to represent an individual (John), while keeping the use of dashed arrows to show instantiation. The theory basic types are shaded to differentiate them from domain elements.

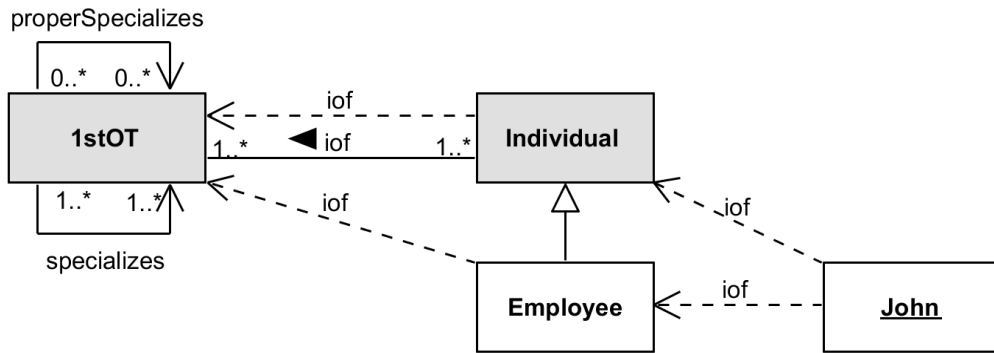


Figure 11 - Using the theory to model a domain.

MLT supports also *specializations* and *instantiations* occurring between domain elements. For instance, supposing we need to classify the employees according to their highest academic degrees, we can consider types such as “PhDEmployee” and “BachelorEmployee” to classify respectively employees having Ph.D. and bachelor degrees. These types are proper specializations of “Employee”, since their instances are also instances of “Employee”. Thus, by the transitivity of specialization, they also specialize “Individual” and, considering theorem T7, they are instances of “1stOT”.

Further, we may consider a *second-order* type called “EmployeeAcademicDegreeType” that has as instances the types that specialize “Employee” according to the academic degree (e.g. “PhDEmployee” and “BachelorEmployee”). More formally, “EmployeeAcademicDegreeType” is a type applied to types that have the intension given by the conjunction of the intension of “Employee” and a predicate that captures the property of having a specific highest academic degree. For example, the intension of “PhDEmployee” is given by the conjunction of the intension of “Employee” and a predicate that captures the property of having a Ph.D. academic degree, thus

“PhDEmployee” is instance of “EmployeeAcademicDegreeType”. Again applying the basic pattern, “EmployeeAcademicDegreeType” is an *instance of* “2ndOT” (since its instances are instances of “1stOT”) and *specializes* “1stOT” (see A4 and T8).

Figure 12 augments Figure 11 adding the discussed entities and relations. In order to increase the readability of the diagram, we use dashed rectangles to group elements that have a common link to other element and draw only one arrow between the border of the rectangle and the other element. For example, instead of representing two *iof* links between “EmployeeAcademicDegreeType” and its instances, we group its instances in a dashed rectangle and draw one *iof* link between such rectangle and “EmployeeAcademicDegreeType”. Moreover, we omitted the representation of some relations that are implied by the represented relations. For example, although we do not represent that “PhDEmployee” proper specializes “Individual”, it can be inferred by the fact that it proper specializes “Employee” which, in turn, proper specializes “Individual”. Finally, we represented that “John” is an instance of “PhDEmployee”, and thus, given the proper specialization relation semantics, it can be inferred that “John” is also instance of both “Employee” and “Individual”.

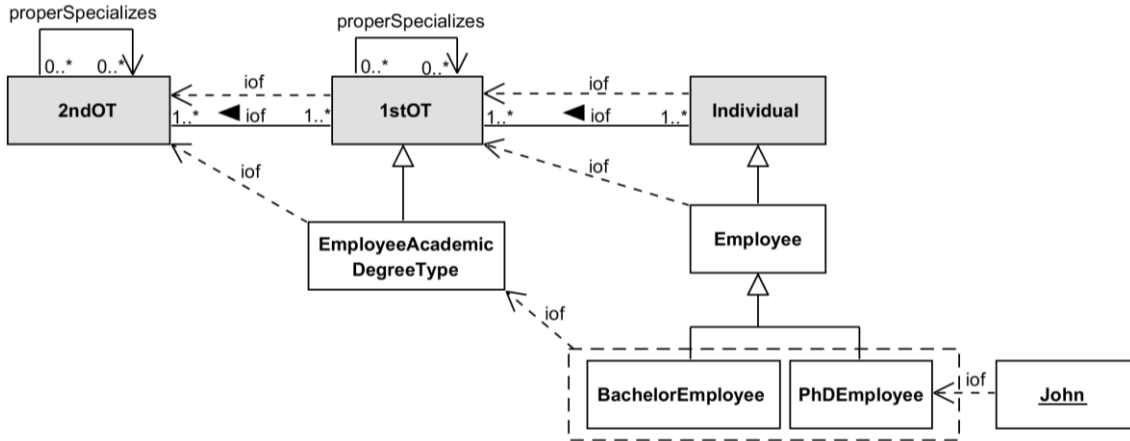


Figure 12 - Instantiations and specializations between domain elements.

3.3.2 The Subordination Relation

Consider now an extension of the example in Figure 12, in which we introduce a second *second-order type* called “EmployeeRoleType” beside “EmployeeAcademicDegreeType”. The instances of “EmployeeRoleType” are specializations of “Employee” according to the role they play (e.g. “Programmer” and “ResearchManager”). Consider further that, in order to reflect required qualifications in the domain, all instances of “EmployeeRoleType” must specialize instances of “EmployeeAcademicDegreeType”. In other words, the intension of each instance of “EmployeeRoleType” is given by the conjunction of the intension of an instance of “EmployeeAcademicDegreeType” and an additional constraint capturing the role their instances must play. For example, we may consider that “Programmer” *specializes* “BachelorEmployee” and “ResearchManager” *specializes* “PhDEmployee”. To allow modelers to capture this kind of

relations between higher-order types that implies *specializations* between their instances, MLT defines the notion of *subordination*.

We call *subordination* the relations that occur between two higher-order types $t1$ and $t2$ when $t1$ applies to types that have the intension given by the conjunction of the intension of an instance of $t2$ and a predicate that captures a constraint following some classification criteria. Therefore, D3 defines that $t1$ is subordinate to $t2$ iff every instance of $t1$ specializes an instance of $t2$. Subordination is a relation between types, and thus D3 excludes the possibility of *subordination* involving instances of “Individual” (i.e. entities with no possible instances).

$$\forall t1, t2 \text{ isSubordinateTo } (t1, t2) \leftrightarrow (\exists x \text{ iof}(x, t1) \wedge (\forall t3 \text{ iof}(t3, t1) \rightarrow (\exists t4 \text{ iof}(t4, t2) \wedge \text{properSpecializes}(t3, t4)))) \quad (D3)$$

Since *subordination* implies *specializations* between the instances of the involved types at one order lower, and *specializations* can only be established between types at the same order, *subordination* can only hold between *higher-order types* of equal order (see Figure 13).

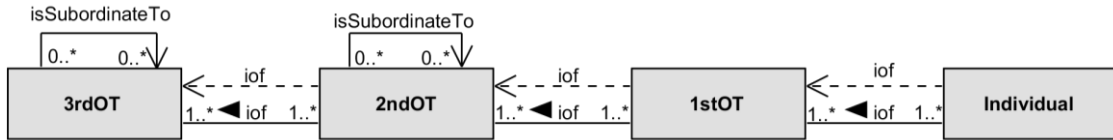


Figure 13 - Intra-level structural relations: subordination.

Figure 14 illustrates the augmented example, showing that “EmployeeRoleType” is *subordinate to* “EmployeeAcademicDegreeType”. Note that *subordination* between two higher-order types implies *specialization* between their instances but should be clearly distinguished from a specialization between the higher-order types (in the example, “EmployeeRoleType” does not specialize “EmployeeAcademicDegreeType”). Moreover, as we show later in Section 3.5, the use of *subordination relations* between higher-order types plays a fundamental role on the specification of taxonomies of types in one order lower.

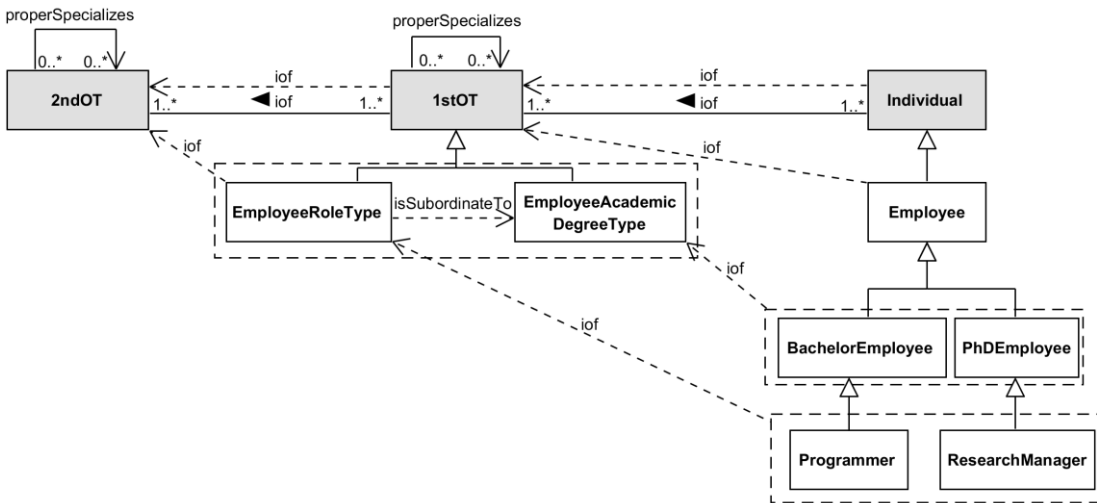


Figure 14 - An example of subordination relation.

Table 1 summarizes the characteristics of the defined intra-level structural relations.

Table 1 - Intra-level structural relations characteristics.

Name	Meaning	Domain and Range	Properties
Specialization <i>specializes(t1,t2)</i>	The intension of <i>t1</i> adds some additional constraints to the one of <i>t2</i> or both types have the same intension ($t2=t1$), i.e. every instance of <i>t1</i> is also an instance of <i>t2</i> .	Types of the same order (instances of 1stOT, 2ndOT or 3rdOT)	Reflexive, antisymmetric and transitive.
Proper Specialization <i>properSpecializes(t1,t2)</i>	The intension of <i>t1</i> adds some additional constraint to the one of <i>t2</i> i.e. every instance of <i>t1</i> is also an instance of <i>t2</i> and there is at least one instance of <i>t2</i> that is not instance of <i>t1</i> .		Irreflexive, antisymmetric and transitive
Subordination <i>isSubordinateTo(t1,t2)</i>	The intension of each instance of <i>t1</i> adds some classification criteria to the intension of some instance of <i>t2</i> i.e. every instance of <i>t1</i> proper specializes some instance of <i>t2</i> .	Higher-order types of the same order (instances of 2ndOT or 3rdOT)	

3.4 Cross-level Structural Relations

This section defines the relations that occur between types of adjacent levels (the so-called *cross-level structural relations*). These relations support our analysis of the notions of *power type* in the literature, as well as their full incorporation in the theory.

3.4.1 The *Power Type Of Relation*

The use of *power types* is one of the most common techniques for multi-level modeling. A seminal theory for the notion of *power type* was proposed by Cardelli (1988). According to (CARDELLI, 1988), “if *A* is a type, then *Power(A)* is the type whose elements are **all** the subtypes of *A*” (including *A*). Following Cardelli, definition D4 states that a type *t1* is *power type of* a type *t2* iff all instances of *t1* are specializations of *t2* and all possible specializations of *t2* are instances of *t1*. In this case, *t2* is said the *base type* of *t1*. Analyzing it in terms of the *intension* of the involved types, a type *t1* is *power type of* a type *t2* iff the *intension of t1* defines that its instances applies to instances of *t2* but does not define a classification criteria. Thus, the extension of *t1* is composed by all specializations of *t2*, including *t2* itself. Further, D4 guarantees that entities without instances (*individuals*) are not considered *power types of* other entities.

$$\forall t1, t2 \text{ isPowertypeOf}(t1, t2) \leftrightarrow (\exists x \text{ iof}(x, t1) \wedge (\forall t3 \text{ iof}(t3, t1) \leftrightarrow \text{specializes}(t3, t2))) \quad (\text{D4})$$

Recall that “Individual” is an *instance of* “1stOT” (theorem T1) and that all the types that specialize “Individual” are also *instances of* “1stOT” (theorem T7). Thus, it follows from the definition of *power type* (D4) that “1stOT” is *power type of* “Individual” (theorem T10). Analogously, “2ndOT” is *power type of* “1stOT” (theorem T11), and “3rdOT” is *power type of* “2ndOT” (theorem T12).

$$\text{isPowertypeOf}(\text{1stOT}, \text{Individual}) \quad (\text{T10})$$

$$\text{isPowertypeOf}(\text{2ndOT}, \text{1stOT}) \quad (\text{T11})$$

$$\text{isPowertypeOf}(\text{2ndOT}, \text{3rdOT}) \quad (\text{T12})$$

It is interesting to note that, to be a *power type*, a type must have an *intension* that defines that all its instances are specializations of the *base type* and, conversely, all specializations of the *base type* are instances of the *power type* (see D4). Thus, it is possible to conclude that each type has at most one *power type* (theorem T13) and that each *type* is *power type of*, at most, one other type (theorem T14). This suggests a concrete syntactic constraint for a multi-level model: only one power type can be linked to a base type through the *is power type of* relation.

$$\forall p, t \text{ isPowertypeOf}(p, t) \rightarrow \nexists p' (p \neq p') \wedge \text{isPowertypeOf}(p', t) \quad (\text{T13})$$

$$\forall p, t \text{ isPowertypeOf}(p, t) \rightarrow \nexists t' (t \neq t') \wedge \text{isPowertypeOf}(p, t') \quad (\text{T14})$$

Theorem T13 can be proved as follows: (i) supposing two higher order types, p and p' , are power type of t , according to D4, both p and p' should have as only instances all the specializations of t ; (ii) thus, applying axiom A2, we conclude that p is equal to p' ($p=p'$). Analogously, theorem T14 can be proved as follows: (i) supposing p is power type of t , according to D4, p should have as only instances all the specializations of t ; (ii) if we also consider a type t' such that p is power type of t' then p should have as only instances all the specializations of t' ; thus, $t = t'$.

In his accounts for the notion of *power type*, Cardelli (1988) proved that if a type $t2$ specializes a type $t1$ then the *power type of* $t2$ specializes the *power type of* $t1$. Since our definition for *isPowertypeOf* relation follows Cardelli's definition, we verified that this property is entailed by our theory. Theorem T15 formalizes this property. This may be used to check the syntax of power type hierarchies, and also to generate the power type hierarchy corresponding to the base type hierarchy.

$$\forall t1, t2, t3, t4 (\text{specializes}(t2, t1) \wedge \text{isPowertypeOf}(t4, t2) \wedge \text{isPowertypeOf}(t3, t1)) \rightarrow \text{specializes}(t4, t3) \quad (\text{T15})$$

T15 can be proved as follows: (i) considering that $t3$ is power type of $t1$ by definition D4 we conclude that $t1$ and all its specializations are *instance of* $t3$; (ii) considering the transitivity of specialization and that $t2$ specializes $t1$, we have that all specializations of $t2$ also specialize $t1$, and thus, all specializations of $t2$ are instance of $t3$; (iii) considering that $t4$ is power type of $t2$ by D4 we conclude that all instances of $t4$ are specializations of $t2$; (iv) thus, by (ii) and (iii) we conclude that all instances of $t4$ are also instances of $t3$, i.e., $t4$ specializes $t3$.

Given the power type definition (D4), if $p1$ is power type of $t1$ we conclude that $p1$ is one order higher than $t1$, i.e., if $p1$ is a second-order type ($\text{iof}(p1, 2\text{ndOT})$) then $t1$ is a first-order type ($\text{iof}(t1, 1\text{stOT})$), if $p1$ is a third-order type ($\text{iof}(p1, 3\text{rdOT})$) $t1$ is a second-order type ($\text{iof}(t1, 2\text{ndOT})$), and so on. Furthermore, since instances of "Individual" are not types, they cannot participate in *isPowertypeOf* relations as power type nor as base type. Figure 15 augments Figure 9 by including the representation of *isPowertypeOf* relations.

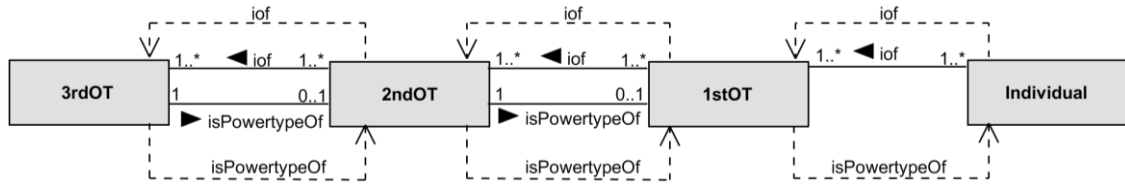


Figure 15 - Cross-level relations: *isPowertypeOf*.

Since the *power type* of a base type is a type whose intension defines that its instances classify instances of the base type, for each first-order type f it is always possible to define a second-order type s such that s is *power type of* f , and for each second-order type s it is possible to define a third-order type t such that t is *power type of* s . While the theory necessitates the existence of the power type of any type (except the power types of third-order types, which are outside the scope of the theory), the decision on whether to represent the *power type* of a particular type is a modeling decision. When the *power type* is not relevant for the domain being modeled it is often omitted from the model.

To illustrate the use of the *is power type of* relation, we augment the example of Figure 14 in Figure 16 introducing “EmployeeType”, which is *power type of* “Employee”. Consequently, all types that *specialize* “Employee” are *instances of* “EmployeeType”. Since the instances of “EmployeeType” are first-order types, “EmployeeType” is an *instance of* “2ndOT” and *specializes* “1stOT”. Further, since all instances of “EmployeeRoleType” are also *instances of* “EmployeeType”, it follows that “EmployeeRoleType” *specializes* “EmployeeType”. Analogously, “EmployeeAcademicDegreeType” *specializes* “EmployeeType”.

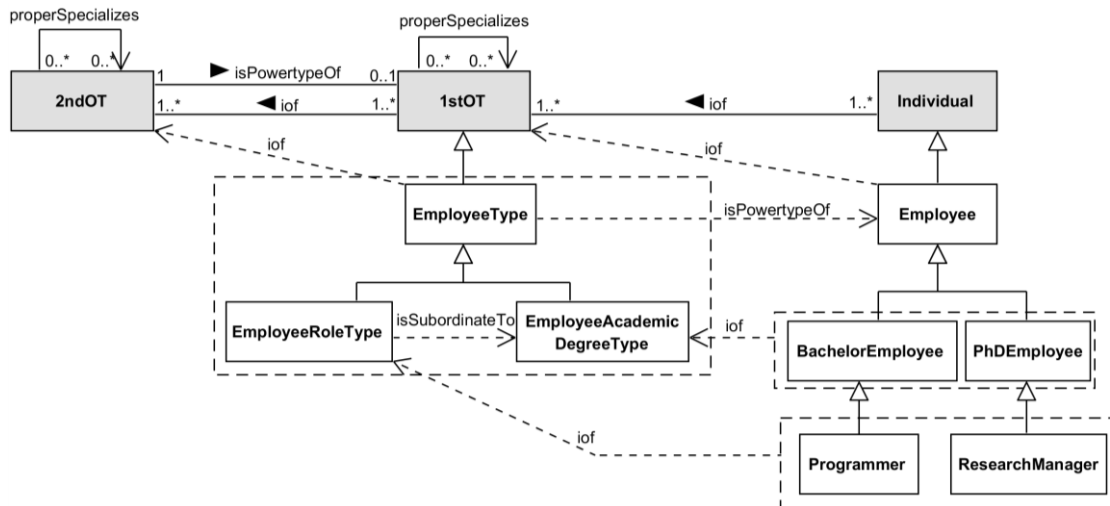


Figure 16 - An example of *isPowertypeOf* relation.

3.4.2 The Categorization Relation and its variations

Although the definition of *power type* we adopted here is compliant with the one proposed by Cardelli (1988), there are other definitions to this term in software engineering literature which have had great influence in practice, for example the definition in (ODELL, 1994).

In (ODELL, 1994), the author stated that a *power type* is a type whose instances are subtypes of another type. It is important to notice that Odell’s definition is less strict than Cardelli’s (CARDELLI, 1988). Cardelli follows the *power set* concept stating that **all** the specializations of the base type are instances of the *power type*. Odell’s definition, in turn, does not comply with that restriction. Thus, as pointed out by (HENDERSON-SELLERS, 2012), the relation defined by Odell is misnamed *power type* since, in fact, it denotes a *subset* of the *power set*.

Inspired on Odell’s definition (ODELL, 1994), we defined the categorization relation (Definition D5): a type t_1 categorizes a type t_2 iff all instances of t_1 are properSpecializations of t_2 . Further, D5 guarantees that categorization relations only apply to elements that are not individuals (i.e., elements that have instances).

$$\forall t_1, t_2 \text{ categorizes } (t_1, t_2) \leftrightarrow (\exists x \text{ iof}(x, t_1) \wedge (\forall t_3 \text{ iof}(t_3, t_1) \rightarrow \text{properSpecializes}(t_3, t_2))) \quad (D5)$$

The *categorization* relation occurs between a higher order type t_1 and a base type t_2 when the *intension* of t_1 defines that its instances *specialize* t_2 according to a specific *classification criteria* (i.e. *proper specializes* t_2). Thus, the instances of t_1 *specialize* t_2 but t_2 is not an *instance of* t_1 and there may be other types that *specializes* t_2 according to other *classification criteria* and, thus, are not *instances of* t_1 . *Categorization* relations only occur between types of adjacent levels (see Figure 17).

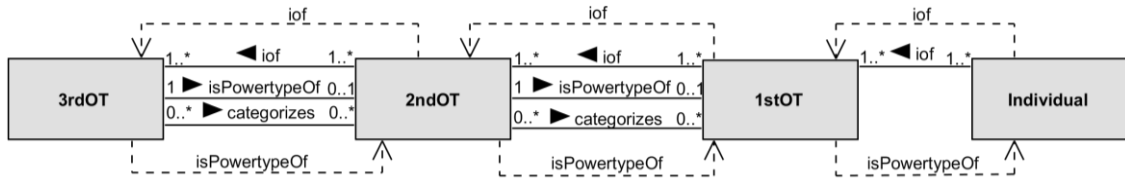


Figure 17 - Cross-Level relations: *categorization*.

Recall that, if a type t' *proper specializes* a type t , the *intension* of t' is given by the conjunction of the *intension* of t and a predicate that captures the additional constraints defined by t' with respect to t . Extending this reasoning, if a higher-order type h *categorizes* t , the *intension* of h establishes some criteria to define the additional constraint that is joined to the *intension* of t to compose the *intension* of its instances. Thus, every type t' whose *intension* extends the *intension* of t following the established criteria is considered an instance of h .

In our previous example, “EmployeeAcademicDegreeType” use the employees’ academic degree as a criterion to classify employees. Putting it more formally, the *intension* of “EmployeeAcademicDegreeType” defines that, to be considered an instance of it, a type must have its *intension* given by the conjunction of the *intension* of “Employee” and a constraint that captures the property of having a specific academic degree. Therefore, we conclude that “EmployeeAcademicDegreeType” *categorizes* “Employee”.

Still considering the previous example, the *intension* of “PhDEmployee” is given by the conjunction of the *intension* of “Employee” and a predicate that captures the property of having

a PhD degree. Thus, “PhDEmployee” is an instance of “EmployeeAcademicDegreeType”. Analogously, since the instances of “EmployeeRoleType” *specialize* “Employee” according to roles the employee was hired to play (*the classification criteria*), “EmployeeRoleType” also *categorizes* “Employee”, having instances such as “Programmer” and “Research Manager”.

Note that, if a type $t1$ is *subordinate to* $t2$ and $t2$ *categorizes* a type $t3$, considering the definitions of *subordination* (D3) and *categorization* (D5) we conclude that all instances of $t1$ *proper specialize* some instance of $t2$ and that all instances of $t2$ *proper specialize* $t3$. Applying the *proper specialization* definition (D2) it follows that all instances of $t1$ *proper specialize* $t3$ and, thus, $t1$ *categorizes* $t3$. This idea is formalized in theorem T16. This theorem can be used to check the completeness of models.

$$\forall t1, t2, t3 (\text{isSubordinateTo}(t1, t2) \wedge \text{categorizes}(t2, t3)) \rightarrow \text{categorizes}(t1, t3) \quad (\text{T16})$$

Further, considering the definitions of *power type* (D4), *categorization* (D5) and *proper specialization* (D2) we conclude that if a type $t2$ is *power type of* a type $t1$ and a type $t3$ *categorizes* the same base type $t1$, then all instances of $t3$ are also *instances of* the power type $t2$ and, thus, $t3$ *proper specializes* $t2$. This idea is formalized in theorem T17. Again, this theorem can be used to check the completeness of models: a model would be incomplete if it omits the specialization between a type that categorizes a base type and this base type’s power type.

$$\forall t1, t2, t3 (\text{isPowertypeOf}(t2, t1) \wedge \text{categorizes}(t3, t1)) \rightarrow \text{properSpecializes}(t3, t2) \quad (\text{T17})$$

Thus, considering our previous example, both “EmployeeAcademicDegree” and “EmployeeRoleType” *categorize* “Employee” and *proper specialize* “EmployeeType”.

Although intuitive, the notion of *categorization* does not capture some important and subtle aspects of the relation between the higher-order type and base type. For instance, if “EmployeeRoleType” *categorizes* “Employee” then instances of “Employee” can be *instances of* instances of “EmployeeRoleType”. However, it is not clear whether all instances of “Employee” must instantiate at least one instance of “EmployeeRoleType”. It is also silent on whether it is possible to have instances of “Employee” that are instances of more than one instance of “EmployeeRoleType”. To address these subtle aspects of the relation between higher-order types and base types MLT defines three variations of the categorization relation.

First of all, suppose that our company considers that each employee must play at least one role, i.e., in addition to the fact that “EmployeeRoleType” *categorizes* “Employee” the instances of “EmployeeRoleType” must completely classify the instances of “Employee”. In order to accommodate this expressiveness, we define a variation of *categorization* relation called *completeCategorization* (see definition D6). Thus, we are able to state that “EmployeeRoleType” *completely categorizes* “Employee”.

$$\begin{aligned} &\forall t1, t2 \text{ completelyCategorizes}(t1, t2) \leftrightarrow \\ &(\text{categorizes}(t1, t2) \wedge (\forall e \text{ iof}(e, t2) \rightarrow \exists t3 (\text{iof}(e, t3) \wedge \text{iof}(t3, t1)))) \end{aligned} \quad (\text{D6})$$

We also define a variation of *categorization* relation, called *disjointCategorization*, to accommodate the cases in which each *instance* of the base type is *instance of* at most one instance of the characterizing type. Thus, according to D7, a type *t1* *disjointlyCategorizes* *t2* iff *t1* *categorizes* *t2* and every instance of *t2* is *instance of*, at most, an *instance of* *t1*.

$$\forall t1, t2 \text{ disjointlyCategorizes}(t1, t2) \leftrightarrow (\text{categorizes}(t1, t2) \wedge \forall e, t3, t4 ((\text{iof}(t3, t1) \wedge \text{iof}(t4, t1) \wedge \text{iof}(e, t3) \wedge \text{iof}(e, t4)) \rightarrow t3 = t4))) \quad (D7)$$

In our example, we could consider that each employee falls under one classification according to his higher academic degree. Thus, “EmployeeAcademicDegreeType” simultaneously *disjointlyCategorizes* and *completelyCategorizes* “Employee”, i.e. each *instance of* “Employee” is *instance of* one and only one *instance of* “EmployeeExperienceType”. In this case, we say that “EmployeeAcademicDegreeType” *partitions* “Employee” (see Figure 18). D8 formally defines the *partition* relation.

$$\forall t1, t2 \text{ partitions}(t1, t2) \leftrightarrow (\text{completelyCategorizes}(t1, t2) \wedge \text{disjointlyCategorizes}(t1, t2)) \quad (D8)$$

The *intension* of a higher-order type which *partitions* a base type defines that its instances must apply to instances of the base type and also define a *classification criteria* such that each instance of the base type is classified by one and only one instance of the higher order type.

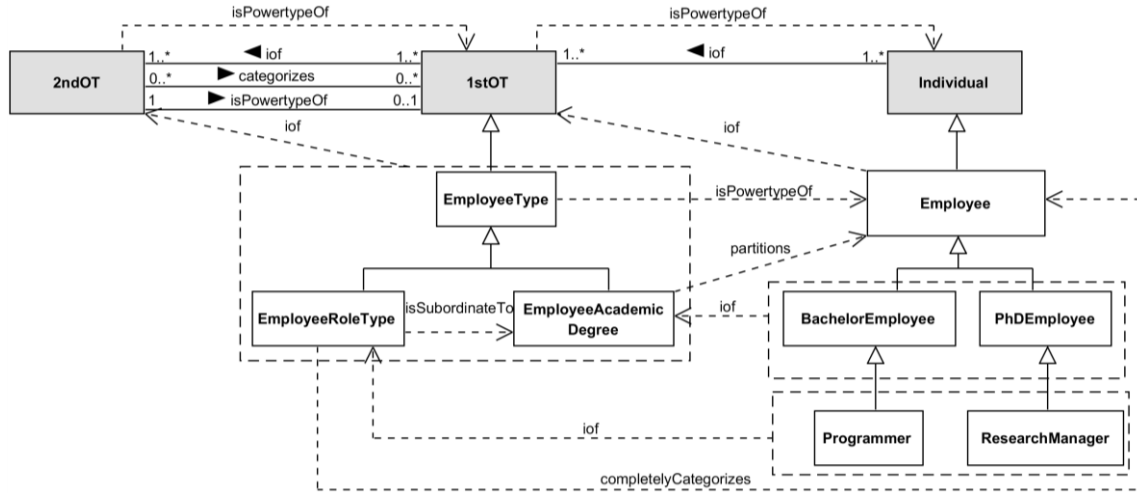


Figure 18 - An example of domain modeling applying *categorization* and *subordination* relations.

Although the definition that Odell gave to the notion of *power type* is aligned with the relation we call *categorizes*, all examples of use provided in (ODELL, 1994) exhibit relations that should be classified as *partitions* according to our theory. Henderson-Sellers (2012), following those examples of use, provided a set theoretic formalization for the notion we call here *partition*.

Since all power type-based relations (*power type of*, *categorization*, *complete categorization*, *disjoint categorization* and *partition*) define that the instances of their domains are *specializations/proper specializations* of their ranges, both their domains and their ranges are types. Further, their domains must be types in one order higher than their ranges. Thus, only higher-order types may play the role of domain of those power type-based relations. Since

completely categorizes, *disjointly categorizes* and *partition* relations all subsume *categorizes* relations, only the latter is represented in Figure 18 for simplicity.

A consequence of the *partitions* definition is that, if two types $t1$ and $t2$ both partition the same type $t3$ then it is not possible for $t1$ to specialize $t2$. This is captured in theorem T18. Again, this theorem suggests a clear syntactic constraint for a multi-level modeling language in the presence of more than one partition of the same base type.

$$\forall t1, t2, t3 \left(\text{partitions}(t1, t3) \wedge \text{partitions}(t2, t3) \right) \rightarrow \neg \text{properSpecializes}(t1, t2) \quad (\text{T18})$$

T18 can be proved as follows: (i) Using the definition of *partitions* (D8), we conclude that the instances of $t1$ form a disjoint and complete partition of $t3$. (ii) Supposing $t1$ *proper specializes* $t2$, using the definition of *proper specialization* (D2) we conclude that all instances of $t1$ must also be instances of $t2$ and $t2$ must have at least one additional instance that is not an instance of $t1$. (iii) Consider that $t4$ is the type that is instance of $t2$ and is not an instance of $t1$. Since $t2$ also partitions $t3$, then $t4$ must specialize $t3$. (iv) However, the instances of $t2$ that are also instances of $t1$ already completely and disjointly classifies the instances of $t3$. Thus, $t4$ does not have possible instances, and thus is not a valid type according to our theory. Therefore, there is no hypothesis in which $t1$ partitions $t3$, $t2$ partitions $t3$ and $t1$ specializes $t2$.

3.4.3 Summary

The definitions of MLT cross-level relations clarify and position conflicting notions of power type. Cardelli’s notion of power type, which is captured in MLT by the *isPowertypeOf* relation, is used to define higher-order types whose instances are all possible specializations of a lower-order type (including the base type itself, since specialization is reflexive). In its turn, Odell’s notion of power type, captured by *categorization* relations, is used to define higher-order types having all its instances as proper specializations of a lower order type.

Given their abstract nature, domain-specific examples of Cardelli’s power types are harder to be found. They may be used whenever the modeler wants to define a second-order type that may give rise to specializations of a base type following any kind of criteria. For example, considering the organizational domain, it may be useful to specify an “Employee Type” power type to generalize over all possible specializations of “Employee” that may be created. If the modeler wants to capture the notion that employees must be categorized according to a specific criteria, giving rise to less abstract second-order types, categorization relations are used. For example, a second-order type called “Employee Role” could be created to support the creation of specializations of “Employee” concerning the role employees play in the company. By virtue of MLT rules, the second-order type “Employee Role” would be a specialization of the (more abstract) second-order type “Employee Type” (as illustrated in Figure 18).

Table 2 summarizes some information about the cross-level relations. All these relations are irreflexive, antisymmetric and intransitive.

Table 2 - Cross-level structural relations characteristics.

Name	Meaning	Domain and Range
Instantiation $iof(e,t)$	The intension of t applies to e .	Elements of adjacent levels.
Power type $isPowertypeOf(t1,t2)$	The intension of $t1$ defines that its instances applies to instances of $t2$ but does not define a <i>classification criteria</i> . Thus, the extension of $t1$ is composed by all specializations of $t2$, including $t2$ itself.	Types of adjacent levels (2ndOT→1stOT or 3rdOT→2ndOT)
Categorization $categorizes(t1,t2)$	The intension of $t1$ defines that its instances applies to instances of $t2$ according to a specific <i>classification criteria</i> . Thus, the extension of $t1$ is composed by the proper specializations of $t2$ that follows the specified classification criteria.	
Complete Categorization $completelyCategorizes(t1,t2)$	A variation of <i>categorization</i> in which the classification criteria defined by the intension of $t1$ guarantees that each instance of $t2$ is instance of at least one instance of $t1$.	
Disjoint Categorization $disjointlyCategorizes(t1,t2)$	A variation of <i>categorization</i> in which the classification criteria defined by the intension of $t1$ guarantees that each instance of $t2$ is instance of at most one instance of $t1$.	
Partition $partitions(t1,t2)$	A variation of <i>categorization</i> in which the classification criteria defined by the intension of $t1$ guarantees that each instance of $t2$ is instance of exactly one instance of $t1$.	

3.5 A Paradigmatic Example

The previous section presented general implications of our theory for multi-level modeling. In this section we consider a representative application scenario to illustrate the theory expressiveness. We consider the biological taxonomy for living beings (MAYR, 1982), which is one of the most mature examples of taxonomical hierarchies. The biological taxonomy for living beings classifies *living beings* according to *biological taxa* in seven or more ranks, e.g., *kingdom*, *phylum*, *class*, *order*, *genus*, *species*, and *breed*.

According to MLT, every domain type is an instance of one of the basic higher-order types (“1stOT”, “2ndOT”, and “3rdOT”), and specializes the basic type at the immediately lower level (respectively, “Individual”, “1stOT”, “2ndOT”). Applying this pattern, we identify that (i) “LivingBeing” is an instance of “1stOT” and specializes “Individual” (since its instances are particular living beings), (ii) “BiologicalTaxon” and its specializations are instances of “2ndOT” and specializes “1stOT” (its instances are the first-order types which classify living beings, such as, e.g., the “Animalia” kingdom and the “Homo Sapiens” species⁵), and (iii) “BiologicalRank” specializes “2ndOT” and instantiates “3rdOT” (its instances are second-order types which classify taxa, such as, e.g., the “Species” taxon). Figure 19 illustrates this domain using the basic pattern.

⁵ Note that in biology there is a long and involved debate on the ontological status of taxa such as species (ERESHEFSKY, 2010). One of the interpretations is that each biological taxon (e.g., the “Homo Sapiens” species, the “Canis Lupus Familiaris” species) represents a group of animals rather than a kind or type of animal. We stay clear of this debate and represent species (and other taxa) as the type that is instantiated by all members of that group (and only by them) (e.g., “Human” and “Dog”).

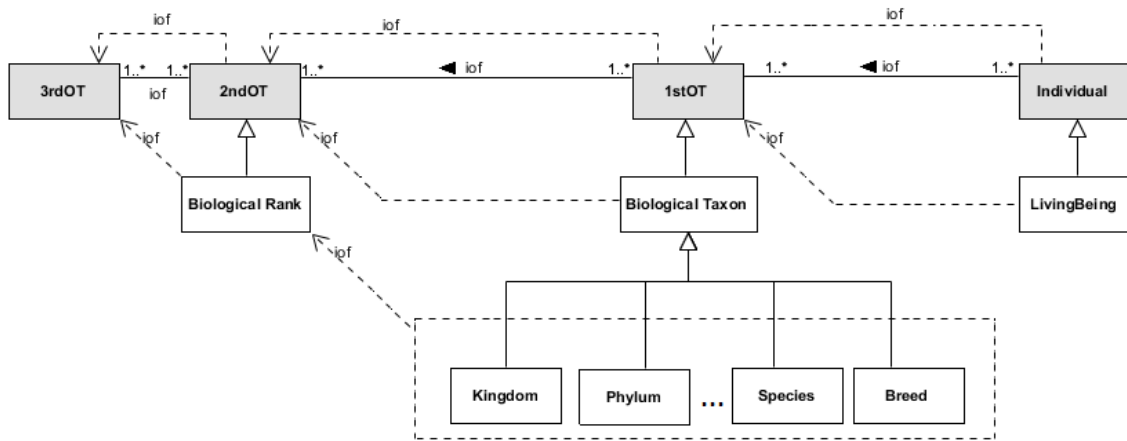


Figure 19 - Applying our theory basic pattern to the biological taxonomy for living beings.

Each “LivingBeing” is instance of one instance of each “Biological Rank”, i.e., each living being is instance of one kingdom, one phylum, and so on. Therefore, we conclude that each one of the seven instances of “BiologicalRank” *partitions* “LivingBeing”. Further, the instances of “Biological Rank” (*specializations* of “Biological Taxon”) obey a subordination chain such that every instance of “Phylum” proper specializes one instance of “Kingdom”, every instance of “Class” proper specializes one instance of “Phylum”, and so on. Thus, according to our theory, each instance of “Biological Rank” is *subordinate to* another instance of “Biological Rank”, forming a chain of subordination (except “Kingdom” which is the top of the chain). Since all instances of “Biological Rank” specialize “BiologicalTaxon” and each instance of “BiologicalTaxon” is instance of exactly one instance of “Biological Rank” (e.g., “Animal” is instance of “Kingdom”, Collie is instance of Breed, etc.) according to our theory, “Biological Rank” *partitions* “BiologicalTaxon”. Figure 20 illustrates how the notions in the theory can be employed; one instance of each represented biological rank is shown.

This example of application shows the expressiveness of our theory. We have explored the entities and relations to fully describe the structural arrangement of the biological taxonomy for living beings. The pattern to classify domain types as instantiations and specializations of the theory’s basic types permitted us to identify the order of each type involved. Using the partition relation we were able to (i) express how the instances of biological rank apply to living beings and (ii) to understand the relation between biological rank and biological taxon. The notion of *subordination* relation was central for understanding how the instances of biological rank are related to each other.

Finally, it allowed us to notice that the shape of tree that the biological taxonomy for living beings exhibits is explained by the combination of two characteristics, namely, (i) the *partitions* relations that all instances of “BiologicalRank” have with “LivingBeing”, and (ii) the chain of *subordination* that the instances of “BiologicalRank” form.

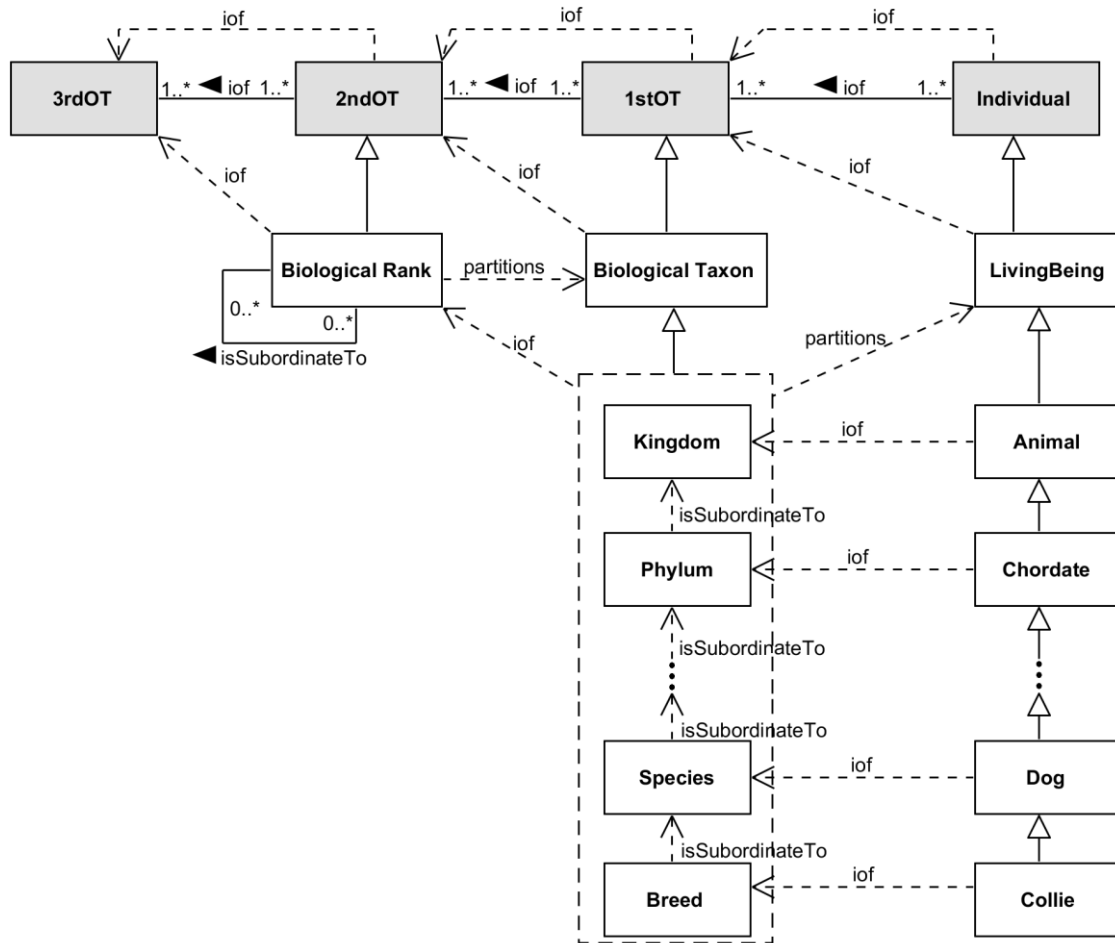


Figure 20 - Using our theory to describe the structural relations that exist in biological taxonomy (relations between the notions of biological rank, biological taxon and living being).

3.6 Accounting for Attributes and Relationships

As we have discussed so far, types capture common features of the entities that are considered their instances. If we say that “John” is an instance of the types “Person” and “Man”, this is because there are certain characteristics that he shares with other instances of “Person” (such as having a brain, being a biped mammal) and with other instances of “Man” (such as having a Y chromosome). These common features are referred to in the intension of the types and are often not explicitly represented in conceptual models. Differently from these common features, features that may vary across different instances of a type or even across different points in time, are often captured using the notions of *attributes* and *relationships* (both which trace back in the conceptual modeling literature to Chen’s work on the Entity Relationship model (CHEN, 1976)). Examples of attributes are a person’s height and weight, a mobile phone’s screen size and a computer’s storage capacity. Examples of relationships include a marriage between husband and wife, an employment between a person and an organization, the friendship between people in a social network, etc. This section extends our account to include these ubiquitous notions in conceptual modeling.

In order to account for attributes in MLT, we extend our domain of quantification (which thus far included only *types* and *individuals*) to cover also *attributes* and their possible *values* in different possible *worlds*. In order to keep our formalization simple despite this additional sorts of elements in the domain of quantification, the axioms defined in this section are formalized in many-sorted first-order logic, assuming four disjoint sets: a set ‘E’ of individuals and types, a set ‘A’ of attributes, a set ‘V’ of values that can be assigned to the attributes, and a set ‘W’ of possible worlds. The MLT axioms described in the previous sections can be understood in the light of this strategy as quantifying always over the set ‘E’ (composed by *individuals* and *types*).

To represent the relation between types and attributes, we define a ternary predicate *typeHasAttribute* (t, a, at) that holds if a type t has an attribute a of type at . For example, the proposition *typeHasAttribute* (*MobilePhone*, *serialNumber*, *String*) denotes that “serialNumber” is an attribute defined for the type “MobilePhone” having “String” as the type of its assignable values. Therefore, each instance of “MobilePhone” may assign instances of “String” to the attribute “serialNumber”⁶.

We consider that attributes are dependent on types. To capture this notion, Axiom A7 states that for each attribute a there must be some entity t which has a . Further, A7 states that each attribute has a unique type at for its values.

$$\forall a: A (\exists t: E, \exists! at: E (\text{typeHasAttribute}(t, a, at))) \quad (\text{A7})$$

To allow the representation of the values assigned to an attribute we define the predicate *hasValue*(e, a, v, w) that holds if an entity e assigns a value v to the attribute a in a world w . In order to cater for “multivalued” attributes, values assigned by entities to attributes are considered sets of entities. Therefore, the sort ‘V’ of possible values of attributes is, indeed, the powerset of the sort of entities ‘E’ ($V = \mathbb{P}(E)$), i.e. ‘V’ is the sort of all possible subsets of ‘E’, including the empty set and ‘E’ itself. For instance, the proposition *hasValue*(*MyPhone*, *SerialNumber*, $\{\text{“1234”}\}$, $w1$) states that a specific instance of “MobilePhone”, named “MyPhone”, has the unitary set $\{\text{“1234”}\}$ assigned to the attribute “SerialNumber” in a world “w1”.

We consider that a type t has an attribute a of type at , iff all instances of t have (at all possible worlds) a set of values v for a respecting attribute type at (i.e., all elements composing the set of values v must be instances of at). This definition is captured by D9. Axiom A8 defines that any entity that has a value for an attribute a must be an instance of a type that has the attribute a . Further, we consider that the scope of an attribute is limited to a specific type and its specializations. Thus, if two different types t and t' have a common attribute a it means that there is a type t'' such that t'' has the attribute a and both t and t' *specializes* t'' (see axiom A9).

⁶ Datatypes such as *String* and *Integer* can be considered first-order types whose instances (e.g. the integer value “1” and the string “xyz”) are “abstract entities” (see (GUIZZARDI, 2005), p. 327).

$$\begin{aligned} & \forall t, at: E, a: A \text{ (typeHasAttribute (t, a, at) } \leftrightarrow \\ & \quad (\neg \text{iof}(t, \text{Individual}) \wedge \neg \text{iof}(at, \text{Individual}) \wedge \forall e: E (\text{iof}(e, t) \rightarrow \\ & \quad \forall w: W, \exists! v: V (\text{hasValue}(e, a, v, w) \wedge \forall e': E (e' \in v \rightarrow \text{iof}(e', at)))))) \end{aligned} \quad (D9)$$

$$\forall e: E, a: A, v: V, w: W (\text{hasValue}(e, a, v, w) \rightarrow \exists t, at: E (\text{iof}(e, t) \wedge \text{typeHasAttribute (t, a, at)})) \quad (A8)$$

$$\begin{aligned} & \forall t, t', at: E, a: A ((\text{typeHasAttribute (t, a, at)} \wedge \text{typeHasAttribute (t', a, at)}) \rightarrow \\ & \quad \exists t'': E (\text{typeHasAttribute (t'', a, at)} \wedge \text{specializes}(t, t'') \wedge \text{specializes}(t', t''))) \end{aligned} \quad (A9)$$

As a consequence of the definitions and axioms present so far, if a type t' *specializes* t , then t' has all attributes of t , capturing the semantics of inheritance (see theorem T19). Theorem T19 can be proved as follows: (i) considering that t defines an attribute a , by D9 we infer that all instances of t must assign values to a ; (ii) Since t' specializes t , by the specialization definition we conclude that all instances of t' are also instances of t , and thus, all instances of t' must assign values to a ; (iii) Therefore, by D9 we conclude that t' also has the attribute a . Another consequence that follows from the definitions and axioms defined so far is that given an attribute a there exists one topmost type t that defines a , i.e. there is a type t that has a such that any other type t' that has the attribute a *specializes* t (see T20).

$$\begin{aligned} & \forall t, t', at: E, a: A ((\text{typeHasAttribute (t, a, at)} \wedge \text{specializes (t', t)}) \rightarrow \\ & \quad \text{typeHasAttribute (t', a, at)}) \end{aligned} \quad (T19)$$

$$\begin{aligned} & \forall a: A, \exists! t, at: E (\text{typeHasAttribute (t, a, at)} \wedge \\ & \quad \forall t': E (\text{typeHasAttribute (t', a, at)} \rightarrow \text{specializes (t', t)})) \end{aligned} \quad (T20)$$

The use of sets as values to the attributes allows the representation of multivalued attributes (by setting as the attribute value a set with more than one element) and the representation of optional attributes by allowing attributes to have an empty set as value. Definitions D10 and D11 capture the notions of mandatory and monovalued attributes in order to express constraints on the multiplicities of attributes. An attribute a is mandatory iff in every possible world, the values assigned to it are not empty sets. An attribute a is monovalued iff in every possible world, the values assigned to it by all entities are sets containing at most one value. Therefore, in order to express, for example, that each instance of “MobilePhone” has one and only one “serialNumber”, besides defining that *typeHasAttribute (MobilePhone, SerialNumber, String)* one should also state that *isMandatoryAttribute (SerialNumber)* and *isMonoValuedAttribute (SerialNumber)*.

$$\forall a: A (\text{isMandatoryAttribute (a)} \leftrightarrow \forall e: E, v: V, w: W (\text{hasValue}(e, a, v, w) \rightarrow \exists e': E (e' \in v))) \quad (D10)$$

$$\begin{aligned} & \forall a: A (\text{isMonoValuedAttribute (a)} \leftrightarrow \\ & \quad \forall e: E, v: V, w: W (\text{hasValue}(e, a, v, w) \rightarrow \forall e', e'': E ((e' \in v \wedge e'' \in v) \rightarrow e' = e''))) \end{aligned} \quad (D11)$$

In the case of multi-level modeling, attributes defined in higher-order types can be given a value for types. We assume that attributes defined in one order capture properties of elements of the immediately lower order and, thus, may have values assigned to them in one order lower. In

other words, attributes defined in first-order types have values assigned for individuals, attributes defined in second-order types have values assigned for first-order types, and so on.

Figure 21 illustrates the concepts presented so far. To capture that each instance of “MobilePhone” must have an IMEI number, a screen of a specific size and a specific storage capacity, the “MobilePhone” type defines three *mandatory* and *monovalued* attributes, namely “imei”, “screenSize” and “storageCapacity”. Therefore, assuming that all these attributes have values of type “String” we may state that $typeHasAttribute(MobilePhone, Imei, String)$, $typeHasAttribute(MobilePhone, ScreenSize, String)$, and $typeHasAttribute(MobilePhone, StorageCapacity, String)$ hold.

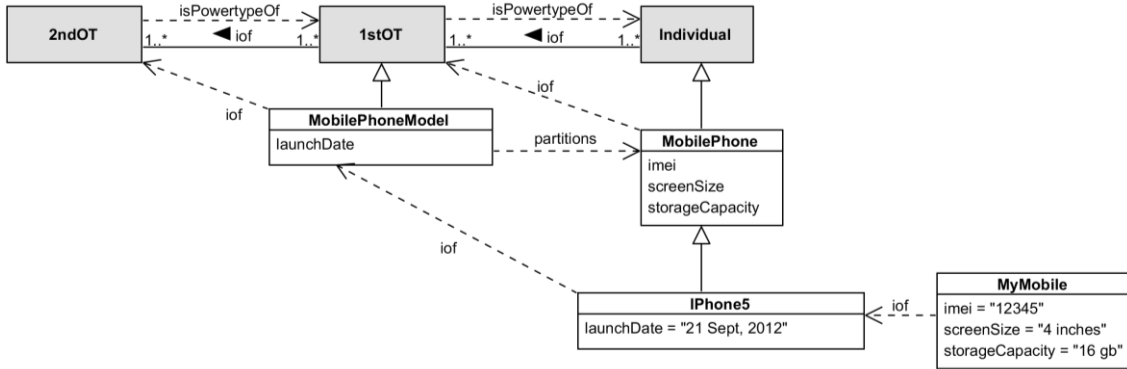


Figure 21 - Illustrating the account for attributes.

In Figure 21, “MyMobile” is an instance of “MobilePhone” (i.e. $iof(MyMobile, MobilePhone)$ holds) having “12345” as its IMEI number, a “4-inch” screen and “16 GB” of storage capacity (in Figure 21 we represented the assignment of values to attributes by adding, after the attribute name, an equality “=” followed by the valued assigned to the attribute; attributes are considered by default mandatory and monovalued). Assuming that Figure 21 illustrates the state-of-affairs of a world $w1$, we may state that $hasValue(MyMobile, Imei, \{“12345”\}, w1)$, $hasValue(MyMobile, ScreenSize, \{“4 inches”\}, w1)$ and $hasValue(MyMobile, StorageCapacity, \{“16 GB”\}, w1)$ hold.

Further, considering that each instance of “MobilePhone” must be classified by one instance of “MobilePhoneModel”, we define a type “MobilePhoneModel” that *partitions* “MobilePhone” (i.e., $partitions(MobilePhoneModel, MobilePhone)$ holds). To capture the official launch date of each mobile phone model, we define that “MobilePhoneModel” has an attribute named “launchDate” ($typeHasAttribute(MobilePhoneModel, LaunchDate, String)$). In Figure 21 “iPhone5” is an instance of “MobilePhoneModel” launched on “21 Sept., 2012” (i.e. $hasValue(IPhone5, LaunchDate, \{“21 Sept., 2012”\}, w1)$ holds).

All attributes introduced in the example so far only have effects at the immediately lower level, complying thus to what has been called “shallow instantiation” (ATKINSON; KÜHNE, 2001). However, a key characteristic of an account for attributes in a multi-level theory is that attributes defined in higher-order types (such as second- and third-order types) may affect the

intension of the instances of these higher-order types. In other words, some attributes of a higher-order type aim at capturing regularities over instances of its instances, constraining the set of possible instances of its instances. Following (GUIZZARDI et al., 2015a), we classify these attributes as *regularity attributes*.

Definition D12 formalizes the notion of *regularity attributes* as attributes that affect the intension of the instances of the types that have it, i.e. two instances having different values assigned to a regularity attribute must have different instances. Therefore, recalling that in MLT two types are the same if they have the exact same possible instances, D12 defines that, an attribute a is a regularity attribute iff every different value for a results in a different type⁷. Note that, since regularity attributes affect the intension of instances of a type, they can only be defined for *higher-order types* (thus not for individuals nor first-order types). This constraint is reflected in D12.

$$\begin{aligned} & \forall a: A \text{ (regularityAttribute } (a) \leftrightarrow \\ & (\forall t, at: E \text{ (typeHasAttribute } (t, a, at) \rightarrow (\neg \text{iof}(t, \text{Individual}) \wedge \neg \text{iof}(t, \text{1stOT})))) \wedge \\ & \forall e, e': E, v, v': V, w, w': W ((\text{hasValue}(e, a, v, w) \wedge \text{hasValue}(e', a, v', w') \wedge v \neq v') \rightarrow e \neq e')) \end{aligned} \quad (\text{D12})$$

Figure 22 extends Figure 21 adding to “MobilePhoneModel” the attributes “instancesScreenSize”, “instancesMinStorageCapacity” and “instancesMaxStorageCapacity”. All these attributes effectively serve as parameters in the intension of the instances of “MobilePhoneModel”, i.e. the values assigned to these attributes influence the selection of the possible instances of instances of “MobilePhoneModel”. Therefore, they are considered *regularity attributes*. For example, by assigning the value “4 inches” to the attribute “instancesScreenSize” of “iPhone5” we are representing that every instance of “iPhone5” must have 4-inch screens. Analogously, by assigning the values “16 GB” and “32 GB” respectively to the attributes “instancesMinStorageCapacity” and “instancesMaxStorageCapacity” of “iPhone5” we are representing that every instance of “iPhone5” must have storage capacity between 16 and 32 GB. Therefore, having a 4-inch screen and storage capacity between 16 and 32 GB are parts of the intension of “iPhone5”.

The influence of the regularity attributes of higher-order types over the intension of its instances may be reflected as constraints over the possible values for attributes of the base type. For example, the fact that “instancesScreenSize” is a *regularity attribute* of “MobilePhoneModel” is reflected by the fact that an instance of “MobilePhoneModel” must have as instances mobile phones having a specific “screenSize”. Therefore, the fact that “iPhone5” has the value “4 inches” assigned to the *regularity attribute* “instancesScreenSize” implies that every instance of “iPhone5” must have the value “4 inches” assigned to the attribute “screenSize”. Analogously,

⁷ A more comprehensive definition would acknowledge that differences in various regularity attributes simultaneously may cancel each other’s effects on the intension, thus we could add a *ceteris paribus* clause to definition D12, which would then state that an attribute a is a regularity attribute iff different values for a with *all other things equal* would result in a different type.

the values assigned to the *regularity attributes* “instancesMinStorageCapacity” and “instancesMaxStorageCapacity” of “MobilePhoneModel” constrain the possible values the instances of a specific (instance of) “MobilePhoneModel” may have assigned to the attribute “storageCapacity”. For example, the values “16GB” and “32 GB” respectively assigned to “instancesMinStorageCapacity” and “instancesMaxStorageCapacity” of “iPhone5” imply that its instances must have a value between 16 and 32 GB assigned to the “storageCapacity” attribute. To emphasize the relations between the *regularity attributes* of “MobilePhoneModel” and the attributes of “MobilePhone” and the constraints over their values, in Figure 22, we placed the related attributes in colored boxes that are linked to each other. Note that this is not meant as a modeling language construct, and our sole intention here is to draw attention to these relations involving regularity attributes.

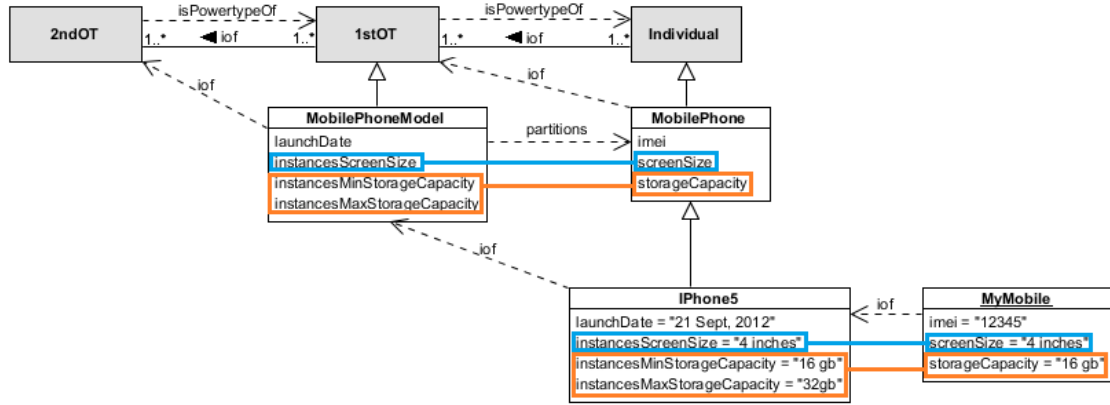


Figure 22 - Illustrating the notion of *regularity attributes*.

A mechanism to express the relations between attributes defined by types in one order and attributes of types in one order lower is a desirable feature of multi-level modeling languages. Indeed, some potency-based approaches to multi-level modeling include some support for the representation of regularity attributes. For example, in Melanie (ATKINSON; GERBIG, 2012) the notions of *durability* and *mutability* are used to capture situations in which the attribute of a higher-order type t directly influences the possible values that instances of instances of t may assign to an attribute (such as the relation between the attribute “instancesScreenSize” of “MobilePhoneModel” and the attribute “screenSize” of “MobilePhone” illustrated in Figure 22). For further discussion considering the relation between MLT and clabject and deep-instantiation based approaches see Section 3.10.2.

To account for basic relationships in our theory we follow a strategy similar to the one adopted by OWL (W3C, 2012), Telos (MYLOPOULOS, 1992) and Ecore (STEINBERG; BUDINSKY, 2008): we represent a binary relationship between two types $t1$ and $t2$ as an attribute defined in $t1$ with type $t2$ (actually representing an association end connected to $t2$). This treatment allows the reuse of the notions of mandatory, monovalued and regularity attributes. These can be applied on both association ends when necessary, giving rise to two opposing

attributes each one defined in each of the related types having the other type as the type of the attribute (again similarly to Ecore and OWL). For example, consider the scenario illustrated in Figure 23. According to Figure 23, each instance of “MobilePhone” “requires” an instance of “Processor” installed in it (“installedProcessor”) and each instance of “Processor” may be “installed in”, at most, one instance of “MobilePhone”. Thus: (i) “MobilePhone” has a mandatory and mono-valued attribute called “installedProcessor” having “Processor” as type (formally, $typeHasAttribute(MobilePhone, installedProcessor, Processor)$, $isMandatoryAttribute(installedProcessor)$ and $isMonoValuedAttribute(installedProcessor)$); and (ii) “Processor” has a mono-valued attribute called “installedIn” having “MobilePhone” as type (formally, $typeHasAttribute(Processor, installedIn, MobilePhone)$ and $isMonoValuedAttribute(installedIn)$).

Following the same approach, we could represent the relation between “MobilePhoneModel” and “ProcessorModel” depicted in Figure 23. To capture that each (instance of) “MobilePhoneModel” “is compatible with” one (instance of) “ProcessorModel” we could define in “MobilePhoneModel” a mandatory mono-valued attribute called “compatibleProcessorModel” having “ProcessorModel” as type (formally, $typeHasAttribute(MobilePhoneModel, compatibleProcessorModel, ProcessorModel)$). Moreover, we could capture the fact that each (instance of) “ProcessorModel” may be compatible with some (instance of) “MobilePhoneModel” by defining in “ProcessorModel” an attribute called “compatiblePhoneModels” having “MobilePhoneModel” as type (formally, $typeHasAttribute(ProcessorModel, compatiblePhoneModels, MobilePhoneModel)$).

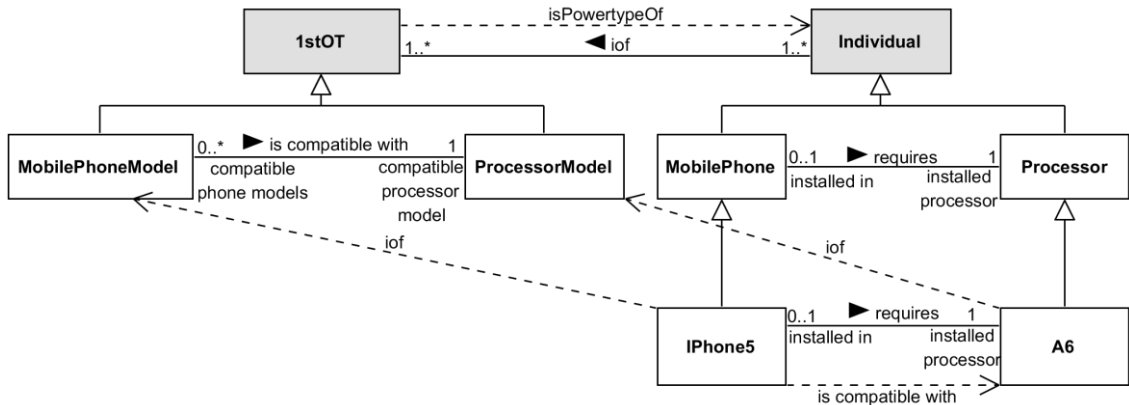


Figure 23 - Illustrating a scenario in which relations in one order capture regularities over instances of types in one order lower.

Whenever opposite attributes are defined, we need to relate the two attributes in order to constrain that whenever an instance of one type refers to an instance of the other type through an attribute, the reference in the opposite direction also holds. For example, we need to constrain that whenever an instance e of “MobilePhone” refers to an instance e' of “Processor” through the attribute “installedProcessor” then e' refers to e through the attribute “installedIn”. In this case,

we say that the attributes “installedProcessor” and “installedIn” are *opposite* attributes. The *refers to* predicate is formally defined in D13, while D14 formally defines the *is opposite* predicate. D13 states that an entity e refers to an entity e' through attribute a (in a world w), iff the value of e for a (in w) is a set that includes e' . D14, in its turn, states that two attributes a and a' are *opposite* iff whenever an entity e refers to an entity e' through the attribute a , e' refers to e through a' , and vice versa. In the example below, *isOpposite(installedIn, installedProcessor)* and *isOpposite(compatiblePhoneModels, compatibleProcessorModel)* hold.

$$\forall e, e': E, a: A, w: W (\text{refersTo}(e, e', a, w) \leftrightarrow \forall v: V (\text{hasValue}(e, a, v, w) \rightarrow (e' \in v))) \quad (D13)$$

$$\forall a, a': A (\text{isOpposite}(a, a') \leftrightarrow \forall e, e': E, w: W (\text{refersTo}(e, e', a, w) \leftrightarrow \text{refersTo}(e', e, a', w))) \quad (D14)$$

To see how the notion of *regularity attribute* is applicable to the attributes capturing relations between types, consider that instances of a (instance of) “MobilePhoneModel” may have installed on them only instances of the (instance of) “ProcessorModel” compatible with it. In this case, the intension of an instance of “MobilePhoneModel” is affected by the value assigned to its “compatibleProcessorModel” attribute. For example, in Figure 23 since the “iPhone5” “is compatible with” “A6” (i.e. “iPhone5” has “A6” as a “compatibleProcessorModel”), instances of “iPhone5” must have processors of type “A6” installed on them. Therefore, the attribute “compatibleProcessorModel” of the type “MobilePhoneModel” is a *regularity attribute* that constrains the possible values for the attribute “installedProcessor” of the type “MobilePhone”. Conversely, and following analogous reasoning, we can conclude that the attribute “compatiblePhoneModel” of the type “ProcessorModel” is a *regularity attribute* that constrains the possible values for the attribute “installedIn” of the type “Processor”.

This simple treatment of relations as attributes can be extended with a notion of relations as object-like entities (GUARINO; GUIZZARDI, 2015). This notion was already discussed by Chen in 1976 (CHEN, 1976), when he observes that “some people may view something (e.g. a marriage) as a relationship” (i.e. as a tuple that relates two entities), “while other people may view it as an entity” (i.e. as something that have its own life). Subscribing to Chen’s intuition and aiming to explain the very nature of relationships, Guarino and Guizzardi presented in (GUARINO; GUIZZARDI, 2015) an ontological theory of relationships as object-like entities. Following such theory, the relations can be reified giving rise to the so-called relator types.

Following (GUARINO; GUIZZARDI, 2015), MLT supports the representation of relator types as regular types. The formalization that we propose for attributes can also be used to establish the link between relata and relators. Further, the relator-based approach (GUARINO; GUIZZARDI, 2015) can be used to address n-ary relationships when necessary. An example of the use of relator types in multi-level modeling with MLT can be seen in Chapter 5. (including examples of types of relator types in the organizational structure domain).

3.7 Addressing Dynamic Classification

The MLT formalization in Sections 3.1 - 3.4 assumed for simplification that entities instantiate types necessarily, effectively dealing with rigid types in a static classification setting. In this section, we lift that restriction and discuss how dynamic classification can be addressed in MLT, accounting thus also for non-rigid types. Dynamic classification is key to conceptual modeling and in particular to ontology-based conceptual modeling (GUIZZARDI, 2005). Supporting dynamic classification allows the use of MLT as a basis for these kinds of conceptual models (e.g. see Chapter 4. and Chapter 5.).

By addressing dynamic classification, we want to support the notion that both individuals and types can change qualitatively keeping their identity. Consider for example, a hierarchy of second-order types in which the second-order type “Species” is specialized according to conservation status into “Not Threatened”, “Endangered” and “Extinct”. Making the types “Not Threatened”, “Endangered” and “Extinct” anti-rigid allows us to capture the fact that a particular species (say “Giant Panda”) can change types. Ontological implications of this approach to the nature of types are discussed in (GUIZZARDI et al., 2015a).

Our strategy to formalize this notion is based on the use of a world-indexed *instance of* relation, represented by a ternary predicate $iof(e, t, w)$ that holds if an entity e is instance of an entity t (denoting a type) in a world w . Consider for example that “John” is an instance of “Student” at world “w1” but not at “w2”, when he has graduated. In this case, we can state that $iof(John, Student, w1)$ and $\neg iof(John, Student, w2)$.

This modification to the instantiation predicate requires us to adjust some axioms of MLT accordingly. Axiom A1 was modified to express that, to be considered an instance of “Individual”, an entity must have no possible instance in any admissible world (see axiom A1’). Further, two types are considered the same iff they have the same instances in all possible worlds (see axiom A2’). Thus, two types whose extensions are contingently equal are not considered the same. For example, it allows us to capture that, although there is a possible world in which all instances of “Person” are also instances of “Student”, “Student” and “Person” are different types since an (instance of) “Person” is not necessarily an (instance of) “Student”.

$$\forall x, w (iof(x, Individual, w) \leftrightarrow \forall w' (world(w') \rightarrow \neg \exists y (iof(y, x, w')))) \quad (A1')$$

$$\begin{aligned} &\forall t, t', w ((\neg iof(t, Individual, w) \wedge \neg iof(t', Individual, w)) \rightarrow \\ &((t = t') \leftrightarrow \forall x, w' (iof(x, t, w') \leftrightarrow iof(x, t', w')))) \quad (A2') \end{aligned}$$

The characterizations of the other basic types must also be adjusted to consider possible worlds. Thus, axiom A3’ characterizes “First-Order-Type” (or shortly “1stOT”), defining a first-order type as an entity with at least one instance in a possible world and whose instances in all possible worlds are instances of “Individual”. Analogously, A4’ and A5’ characterize “Second-

Order Type” (or “2ndOT”) and “Third-Order Type” (“3rdOT”). Additionally, axiom A6’ adjusts A6 to state that, for *all possible worlds*, each entity in our domain of enquiry is either an instance of “Individual”, “1stOT”, “2ndOT” or “3rdOT” (except “3rdOT” whose type is outside the scope of the formalization).

$$\forall t, w \text{ (iof}(t, 1stOT, w) \leftrightarrow (\exists y, w' \text{ (iof}(y, t, w')) \wedge \forall x, w'' \text{ (iof}(x, t, w'') \rightarrow \text{iof}(x, Individual, w''))))) \quad (A3')$$

$$\forall t, w \text{ (iof}(t, 2ndOT, w) \leftrightarrow (\exists y, w' \text{ (iof}(y, t, w')) \wedge \forall t', w'' \text{ (iof}(t', t, w'') \rightarrow \text{iof}(t', 1stOT, w''))))) \quad (A4')$$

$$\forall t, w \text{ (iof}(t, 3rdOT, w) \leftrightarrow (\exists y, w' \text{ (iof}(y, t, w')) \wedge \forall t', w'' \text{ (iof}(t', t, w'') \rightarrow \text{iof}(t', 2ndOT, w''))))) \quad (A5')$$

$$\forall w, x \text{ ((world}(w) \wedge \neg \text{world}(x)) \rightarrow (\text{iof}(x, Individual, w) \vee \text{iof}(x, 1stOT, w) \vee \text{iof}(x, 2ndOT, w) \vee \text{iof}(x, 3rdOT, w) \vee (x = 3rdOT))) \quad (A6')$$

Since axiom A1’ defines that to be an instance of “Individual” in a world w an entity x must not have instances in any world, we can conclude that if an entity x is an instance of “Individual” in any world it is an instance of “Individual” in all possible worlds, i.e. instances of “Individual” are necessarily instances of it. Thus, “Individual” is a rigid type. Analogously, using axioms A3’ to A5’ we conclude that “1stOT”, “2ndOT” and “3rdOT” are also rigid types, i.e., the basic types of MLT are all rigid.

Concerning the intra- and the cross-level structural relations of MLT, they express properties that are not contingent to the involved types. Consider for example the specialization relation between $t1$ and $t2$: a type $t1$ specializes a type $t2$ iff in all possible worlds all instances of $t1$ are also instances of $t2$ (see definition D1’). We can observe that, by definition, it is not admissible for $t1$ to *specialize* $t2$ in a world w and not specialize it in another world w' . Thus, in contrast with the instantiation relation, the *specialization* relation is not world-indexed. The same reasoning applies to all other intra- and cross-level structural relations of MLT, namely, *proper specialization*, *subordination*, *power type of* and *categorization*. Therefore, the definitions of these relations in the formalization that accounts for dynamic classification are most similar to the ones presented in Sections 3.3 and 3.4, with minor adjustments concerning the quantification of possible worlds (see Definitions D1’, D2’, D3’, D4’, D5’, D6’, D7’ and D8’).

$$\forall t1, t2 \text{ (specializes}(t1, t2) \leftrightarrow (\exists y, w1 \text{ (iof}(y, t1, w1)) \wedge \forall e, w2 \text{ (iof}(e, t1, w2) \rightarrow \text{iof}(e, t2, w2)))) \quad (D1')$$

$$\forall t1, t2 \text{ (properSpecializes}(t1, t2) \leftrightarrow (\text{specializes}(t1, t2) \wedge \neg(t1 = t2))) \quad (D2')$$

$$\forall t1, t2 \text{ (isSubordinateTo}(t1, t2) \leftrightarrow (\exists x, w1 \text{ (iof}(x, t1, w1)) \wedge \forall t3, w2 \text{ (iof}(t3, t1, w2) \rightarrow \exists t4 \text{ (iof}(t4, t2, w2) \wedge \text{properSpecializes}(t3, t4)))))) \quad (D3')$$

$$\forall t1, t2 \text{ (isPowertypeOf}(t1, t2) \leftrightarrow (\exists x, w1 \text{ (iof}(x, t1, w1)) \wedge \forall t3, w2 \text{ (iof}(t3, t1, w2) \leftrightarrow \text{specializes}(t3, t2)))) \quad (D4')$$

$$\forall t1, t2 (\text{categorizes}(t1, t2) \leftrightarrow (\exists x, w1 (\text{iof}(x, t1, w1)) \forall t3, w2 (\text{iof}(t3, t1, w2) \rightarrow \text{properSpecializes}(t3, t2)))) \quad (D5')$$

$$\forall t1, t2 (\text{completelyCategorizes}(t1, t2) \leftrightarrow (\text{categorizes}(t1, t2) \wedge \forall w, e (\text{iof}(e, t2, w) \rightarrow \exists t3 (\text{iof}(e, t3, w) \wedge \text{iof}(t3, t1, w)))))) \quad (D6')$$

$$\forall t1, t2 (\text{disjointlyCategorizes}(t1, t2) \leftrightarrow (\text{categorizes}(t1, t2) \wedge \forall w, e, t3, t4 ((\text{iof}(t3, t1, w) \wedge \text{iof}(t4, t1, w) \wedge \text{iof}(e, t3, w) \wedge \text{iof}(e, t4, w)) \rightarrow t3 = t4))) \quad (D7')$$

$$\forall t1, t2 (\text{partitions}(t1, t2) \leftrightarrow (\text{completelyCategorizes}(t1, t2) \wedge \text{disjointlyCategorizes}(t1, t2))) \quad (D8')$$

Further, all the theorems presented in the previous formalization are also valid when considering dynamic classification. The theorems T1-T4 and T7-T9 presented in Sections 3.1 and 3.3 must be properly adapted considering the use of the world-indexed instantiation relation while the theorems T10 – T15 presented in Section 3.4 can be included in this formalization without modification.

3.8 A Note on the Identity Conditions of Types

The notion of equality of types is central to account for both Odell's and Cardelli's notions of power type. For example, according to Odell's notion, there is no instance of the "power type" that is equal to the base type. Further, considering Cardelli's sense, (in-)equality is key to establish the uniqueness of the power type for a given base type, as well as the uniqueness of a base type for a given power type. So, which notion of identity condition is adequate in the theory becomes an important issue.

As discussed in (SWOYER; ORILIA, 2014), there is a spectrum of options for the identity conditions of types, with respect to how finely they are individuated. In an "*infra-coarse*" account, types with the same extension are considered identical. This is what we would call an "*entirely extensional*" approach.

In contrast, in a "*medium-coarse*" account, types "are identical just in case they necessarily have the same extension" (SWOYER; ORILIA, 2014). "This seems to transpose the identity conditions for sets into an appropriately intensional key, and this is precisely how identity conditions for properties work in accounts that treat them as intensions" (SWOYER; ORILIA, 2014). In such accounts, intensions of types are functions from possible worlds to sets of objects therein (MONTAGUE, 1974). This is one of the approaches that Bealer (1982) uses for dealing with "intensional entities". This is what we would call an "*intensional*" approach.

Finally, in an "*ultra-fine*" approach (also referred to as "hyperintensional" approach (SWOYER; ORILIA, 2014)) types "are individuated almost as finely as the linguistic expressions that express them" (SWOYER; ORILIA, 2014). According to the "hyperintensional" approaches, two types can be considered distinct even in cases they necessarily have the same extension. For

example, if we consider two types t and t' such that the intension of t' is formed by a conjunction of the intension of t with a trivially “true” statement, adopting an “*ultra-fine*” approach t and t' must be considered two distinct types.

As discussed by Bealer (1982), the medium-coarse and the ultra-fine accounts have each their own value for the intensional conception of types, with different applications. He defends that the medium-coarse approach is ideally suited for treating the modalities (necessity, possibility, impossibility, contingency, etc.), and that the ultra-fine approach is valuable for dealing with intentional matters (belief, desire, perception, decision, etc.). He discusses that the ultra-fine approach, while ideally suitable for the treatment of intentional matters, “has only complicated the treatment of the modalities” (BEALER, 1982). Given the scope of the present work, we opt for the medium-coarse approach. It allows us to state the impossibility of individuals to have instances (A1) and later (in Section 3.7) to deal with types that apply contingently (or non-necessarily) to their instances (e.g., Student, Living Person).

In the formalization presented in Section 3.1, while modality is not formally treated, our choice for the medium-coarse account is reflected in how we stipulate the domain of quantification, which includes all possible types and all their possible instances. The fact that we quantify over all possible entities guides the interpretation of the axioms and definitions, in which quantifiers end up having some modal importance (even if informally). For example, Axiom A2 defines that two types are equal *iff all their possible instances coincide*. In other words, A2 states two types t_1 and t_2 are equal *iff it is inadmissible for an entity to be an instance of t_1 and not an instance of t_2* (there is no possible entity that is an instance of t_1 and not an instance of t_2). Our choice of language here (first-order logics instead of some sort of quantified modal logics) aims at making the theory more accessible.

In Section 3.7, we reify possible worlds, and thus, modality is addressed more explicitly in the formalization. Again, here we have opted not to use some sort of modal logic to retain the accessibility of the theory. The axiom that defines equality (A2') states that two types are equal *iff they have the same instances in all possible worlds*, clarifying that we take the approach that two types are the same *iff they are necessarily coextensional*. Since the extension of a type is world-dependent, it makes sense to talk about the distinction between extension (in a particular world) and intension (across worlds) (MONTAGUE, 1974).

3.9 Addressed Requirements

The MLT axiomatization is built up defining the conditions for entities to be considered instances of each one of the theory's basic types. For example, entities having no possible instances are instances of “Individual”, entities having individuals as instances are instances of “First-order Type” and so on. Further, MLT considers that each entity in the domain of enquiry

is instance of exactly one of its basic types. This stratified organization prescribed by MLT basic types accounts for chains of instantiation involving entities of multiple (related) classification levels (meeting requirement R1) and precisely characterizes the nature of entities in each level (meeting R2). Moreover, the basic types' schema can be extended to consider as many orders as necessary to capture a domain at hand, and, once defined a certain number of levels, MLT does not impose to domains the need of defining instances in all prescribed levels. Therefore, this schema allows an arbitrary number of classification levels (requirement R3).

Despite being key to meet three of the defined requirements, the strict stratification strategy imposes a limitation to the theory: MLT does not account for types having instances at different orders. Having the ability to account for this sort of types could be useful to capture the notion of a universal type (e.g. the type "Type" having as instances every entity that have instances). An extension of MLT to account for types having instances at different orders is an issue for future investigation.

The structural relations of MLT play a key role on capturing rules for the instantiation of types at different levels (requirement R4). For example, as discussed in Section 3.5, such relations allow the representation of rules concerning instantiation of types that characterizes the taxonomy of living beings: (i) using the *subordination relation* it is possible to capture that instances of each (instance of) "Biological Rank" proper specialize one instance of another (instance of) "Biological Rank", while (ii) by representing that each (instance of) "Biological Rank" *partition* "LivingBeing", we capture that that every instance of "LivingBeing" is instance of one instance of each (instance of) "Biological Rank".

The MLT account for attributes and relationships (discussed in Section 3.6) proposes the notion of regularity attributes as way to capture constraints involving features of entities in different classification levels (requirement R5). Further, the MLT account for relationships does not pose any restriction on domain relations between types in different classification levels (meeting R6). For instance, considering again the mobile phone domain used to illustrate section 3.6, we could define a relationship capturing that each (instance of) "MobilePhoneModel" (a second-order type) is "designed by" an instance of "Person" (a first-order type).

3.10 Related Work

3.10.1 Power type-based Approaches

Two early attempts to address multi-level modeling, namely *power types* (CARDELLI, 1988; ODELL, 1994) and *materialization* (PIROTTE et al., 1994), raised from the identification of patterns to represent the relationship between a class of categories and a class of more concrete entities. The notion of *power types* was adopted in the object-oriented model community (largely

influenced by (ODELL, 1994)) and *materialization* has been developed in the database community. Despite the different origins, *power type* and *materialization* are based on similar conceptualizations (ATKINSON; KÜHNE, 2008) and addressing the same concerns (GONZALEZ-PEREZ; HENDERSON-SELLERS, 2006). Both approaches establish a relationship between two types such that the instances of one are specializations (subtypes) of another.

Odell (1994) defined the concept of power type informally using regular associations between a class representing the power type and a base class. This differs from MLT's approach because cross-layer relations between types (*is power type of*, *categorizes* and *partitions*) have specialized semantics. This allows us to prescribe rules for the domain models that use these relations following the axioms in the theory.

Similarly to Odell (1994), Gonzalez-Perez and Henderson-Sellers (2006) use an association labeled “partitions” between a power type and a base type (called a “partitioned type” in their terminology). The authors illustrate their technique with a diagram in which “partitions” is modeled as a many-to-one association between “Task” and “TaskKind”, meaning that every instance of the partitioned type (“Task”) is linked to exactly one instance of the power type (“TaskKind”). In the sequel, they discuss that the “*partitions* association possesses instantiation semantics”, and that, because of this, “Task” is a special instance of “TaskKind” (the most generic kind of task). However, if “Task” itself is an instance of “TaskKind”, then the “partitions” association cannot be a many-to-one association between “Task” and “TaskKind”. This is because all instances of subtypes of “Task” are also instances of “Task”, and thus instances of at least *two* “TaskKinds” (one which is “Task” itself). The source of the difficulty seems to lie in that their “partitions” association is semantically overloaded, conflating two underlying notions, in terms of MLT: (i) the fact that “TaskKind” *partitions* “Task”, and (ii) the implied consequence that instances of “Task” are instances of instances of “TaskKind” (which in our theory is reflected in the *instance of* relation between “Task” as specialization of “Individual” and “TaskKind” as a specialization of “First-Order Type”). The modeler is free to determine whether “Task” itself is an instance of “TaskKind” (in which case he/she would replace (i) with the fact that “TaskKind” *is a power type of* “Task”). Note that the elements of our theory help us to identify the semantic overload, provide an explanation for the conceptual issue in this power type-based approach, and offer alternatives to express the modeler's intended conceptualization.

The UML 2.4.1 specification (OMG, 2011) attempts to cover the needs of multi-level modeling by including a power type association that relates a classifier (power type) to a generalization set composed by the generalizations that occur between the base classifier and the instances of the power type. Because of its dependence on the generalization set construct, the pattern can only be applied when specializations of the base type are explicitly modeled (otherwise there would be no generalization set). We consider this undesirable as it would rule

out simple models that are possible in our approach, e.g., one defining “Employee Type” as a power type of “Employee”, without forcing the modeler to define specific instances for “Employee Type”. While our theory necessitates the existence of entities for any type, and hence necessitates the existence of instances for “Employee Type”, it does not require these instances to be modeled explicitly, which is the case of the UML because of its choice to base the power type pattern in a structure that uses generalization sets (this issue is further discussed in Chapter 6. where we use MLT to conduct an analysis and a revision of the UML power type support).

Both the Gonzalez-Perez and Henderson-Sellers’ approach (2006) and the UML approach for power types (OMG, 2011) are founded on Odell’s notion and attempt to address the need of representing multi-level domains in a two level architecture. These approaches do not discuss what characterizes each classification level and do not provide a comprehensive set of principles to guide the organization of entities into levels. For example, the only rule in UML concerning the consistency of instantiation chains aims at avoiding a “power type” to be represented as an instance of itself. Therefore, we consider that these approaches partially meet requirement R1 (since they provide a workaround to account for multiple classification levels in a two level architecture), but do not meet R2 (since they lack rules to guide the correct use of the notion of power type). These approaches allow the definition of chains of power types with as many power types as necessary. Thus, although they do not discuss the notion of classification level, we consider that they meet the requirement R3. Further, we consider that the requirement R4 is partially met by such approaches, since the only modeling construct these approaches define to govern the instantiation of types at different levels is a relation that maps Odell’s notion of power type. Concerning the support for the representation of features (attributes and relationships) of entities, these approaches do not provide any construct to capture rules relating features of entities in different classification levels, not meeting requirement R5. Finally, since no restriction is set concerning domain relations involving entities in different classification levels, we consider that they meet requirement R6.

DeepTelos is a knowledge representation language which approaches multi-level modeling with the systematic application of the notion of “most general instance (MGI)” (JEUSFELD; NEUMAYR, 2016). The authors revisit the axiomatization of Telos (JARKE et al., 1995; MYLOPOULOS et al., 1990) and add the notion of MGI to Telo’s formal principles for instantiation, specialization, object naming and attribute definition. The notion of MGI can be seen as the opposite of the Odell’s power type relation. For example, to capture that “Tree Species” is a “power type” (in Odell’s sense) of “Tree”, in DeepTelos it would be stated that “Tree” is the “most general instance” (MGI) of “Tree Species”.

According to the authors, the modeling levels in DeepTelos are captured by hierarchies of “most general instances”. For example, we may design a multi-level model for hierarchies of products, by representing a type “Product”, a type “ProductModel” as the most general instance

of “Product”, and a type “ProductCategory” as the most general instance of “ProductModel”, with each application of the MGI construct characterizing a level. The authors do not clarify whether entities in different hierarchies could be considered to be at the same level. For example, it is not clearly defined whether instances of “ProductModel” and instances of “TreeSpecies” would be considering entities of the same classification level. In MLT, this is settled by defining the basic types as the top most type of each classification level. Given the aforementioned characteristics of the language, we consider that DeepTelos meets requirement R1, and partially meets R2. Further, since DeepTelos allows the definition of chains of MGI to represent as many levels as necessary, we consider it meets R3.

Considering that DeepTelos provides only the concept of MGI to constrain the instantiation of types in different levels, not elaborating on the nuances of the relations between higher-order types and base types, we consider that it partially meets requirement R4. Although, we consider it is possible to extend the DeepTelos built-in support by using its features of user-defined constraints and rules to formally define all the cross-level structural relations proposed in MLT.

Similarly to MLT, DeepTelos admits relations between types in different levels, meeting thus requirement R6, but, in contrast to MLT, its account for attributes does not include any support to explain the relationship between attributes of entities in different classification levels, not meeting requirement R5.

The notion of *power type* introduced by Odell (1994) in the object oriented community differs from the concept coined earlier by Cardelli (1988) since the latter is derived directly from the mathematical notion of *power set* while the former may be used more loosely as we discussed in Section 3.4. MLT is able to account for both definitions formally, revealing their differences. It covers the expressiveness of both approaches through formally-defined structural cross-level relations (*is power type of*, *categorizes* and *partitions*). Further, it allows us to show that a higher-order type that is related to a base type through the *categorizes* relation is necessarily a specialization of the power type of that base type. Thus, the power type of a base type is the most abstract higher-order type related to a base type.

An example of modeling approach that encompass both Odell’s and Cardelli’s notions of power type is defined in Boro (PARTRIDGE, 2005). Boro is a conceptual modeling approach whose semantics is founded on a four-dimensional (4D), extensional ontology. It provides constructs having purposes similar to the MLT relations of power type, partitions, specializations and subordination. However, while MLT considers an intensional semantics (as discussed in Section 3.8), the use of an extensional ontology as its foundations makes the mapping from Boro constructs to the set theory straightforward (PARTRIDGE et al., 2016). Because of this choice, Boro’s multi-level strategy is not suitable for use with a 3D ontology such as UFO, which requires dynamic classification. Further, differently from MLT, Boro does not discuss the relation between features (attributes and relationships) of entities in different orders.

Boro accounts for types in different classification levels and does not pose any restriction to the use of chains of power types, meeting, thus, R1 and R3. Boro does not explicitly discuss the notion of levels and the nature of entities in each level. Although, since it explicitly adopts an extensional semantics, we consider that the rules and principles derived straightforward from set theory may guide the adequate use of classification (e.g. by adopting the semantics of the set membership (\in) to the instantiation relation many rules to govern the use of instantiations are leveraged). Therefore, we consider that Boro partially meets requirement R2.

Differently from approaches based on Odell's notion and similarly to MLT, Boro defines a number of constructs to capture rules concerning the instantiation of types in different classification levels, meeting thus requirement R4. Concerning the support to the representation of features (attributes and relationships) of entities, it does not provide constructs to capture the relation between features of entities in different classification levels, not meeting R5. Finally, similarly to UML and Henderson-Sellers approaches, Boro does not pose any restriction to domain relations involving entities in different classification levels, meeting thus R6.

Table 3 summarizes our observations concerning the adequacy of the analyzed power type-based modeling approaches to the requirements we have established.

Table 3 - Requirements addressed by power type-based approaches.

Requirement	Approaches based on Odell's notion	Deep Telos	Boro
<i>R1 – To account for entities of multiple classification levels</i>	Partially.	Yes.	Yes.
<i>R2 – To define principles for the organization of entities into levels</i>	No.	Partially.	Partially.
<i>R3 – To allow an arbitrary number of classification levels</i>	Yes.	Yes.	Yes.
<i>R4 – To account for rules for the instantiation of types at different levels</i>	Partially.	Partially.	Yes.
<i>R5 – To account for rules relating features of entities in different levels</i>	No.	No.	No.
<i>R6 – To admit the existence of domain relations between entities in different levels</i>	Yes.	Yes.	Yes.

3.10.2 Deep Instantiation-based Approaches

The concept of power type is founded on the notion that “instances of types can also be types” (ODELL, 1994). Motivated by a similar observation, Atkinson and Kühne (ATKINSON; KÜHNE, 2000) coined the term *clabject*, emphasizing that every instantiable entity has both a type (or class) facet and an instance (or object) facet which are equally valid (ATKINSON; KÜHNE, 2000). This notion is valuable to our theory. The basic types of MLT, except the higher

order one, may be considered clabjects. For example, “Individual” is instance of “First-Order Type” (its instance facet) and a type for all entities that are not types (its type or class facet). (Note that, individuals in MLT (instances of “Individual”) have no class facet, and thus should not be referred to as clabjects.)

In (ATKINSON; KÜHNE, 2000), Atkinson and Kühne argue that a multi-level modeling framework should adhere to two fundamental principles: support for the *clabject* notion and *strict metamodeling*. *Strict metamodeling* (ATKINSON, 1997) assumes that each element of a level must be an instance of an element of the level above. We follow this principle with respect to the instantiation relation, and every entity in our domain of enquiry is instance of exactly one of the basic types and every entity can only be instance of entities at one order higher (all entities that have no instances are instances of “Individual”; “Individual” and all its specializations are instances of “First-Order Type”, and so on.) They also discuss that “some kind of ‘trick’ is needed at the top level”. The ‘trick’ we used in our theory is that the highest order foundational type is not instance of anything, since entities with higher order are not considered (see axiom A6 in Section 3.1). Alternatively, an infinite number of basic types may be considered at ever increasing orders, in which case a ‘trick’ at the top of the classification scheme would not be required. We have opted instead for a finite number of basic types to avoid necessitating the existence of an infinite number of levels, which would be an unnecessary ontological commitment for all conceptual modeling applications we have considered so far.

Atkinson and Kühne have also proposed a *deep instantiation-based* approach (ATKINSON; KÜHNE, 2001, 2008) as a means to provide for multiple levels of classification whereby an element at some level can describe features of elements at each level beneath that level. It is based on the idea of assigning to *clabjects* and *fields* (attributes and slots) a *potency* which defines how deep the instantiation chain produced by that *clabject* or *field* may become. When a *clabject* is instantiated from another *clabject* the potencies of the created *clabject* and of its *fields* are given by the original *clabject* and *fields* potencies decremented by one. Objects have potency equal to zero indicating they cannot be instantiated. If the potency of a *field* becomes zero then a value can be assigned to that *field*. For example, we could define a *clabject mobile phone model* with an attribute *IMEI* assigning a potency of 2 to both the type and the attribute. Therefore, instances of *mobile phone model* would be *clabjects* in which *IMEI* attribute would have potency of 1. Instances of instances of *mobile phone model* have a value assigned to *IMEI*, since its potency would reach zero.

The authors consider that the main benefit of *deep instantiation-based* approach is to reduce “accidental complexity” in domain models since it supports multi-level modeling without the need of introducing types to the models only “because of the idiosyncrasies of a particular solution to deep characterization” (ATKINSON; KÜHNE, 2008). They argue that *power type*-based solutions force the modelers to add unneeded types to the model. For instance, considering

the cited example of *mobile phone model*, using *power types* the modeler would be “forced” to represent the concept of *mobile phone*. Using deep instantiation, the modeler could define the *mobile phone* properties (e.g. *IMEI*) as properties of *mobile phone model* having potency of 2, being free to not represent the concept of *mobile phone*.

While the deep instantiation approach can reduce the number of entities represented in a model, this strategy should be used with parsimony. Important consequences of omitting base types in the current deep instantiation approach are that the modeler becomes unable to express whether the instances of a higher-order type (*mobile phone model* in previous example) are disjoint and/or covering types and we are also prevented from determining metaproperties (such as e.g., rigidity) of the base type (*mobile phone* in this case). Further, as discussed in (GUIZZARDI, 2005), conceptual models should always include kinds that define the principle of identity of individuals (in the example this type is *mobile phone*). If these types are omitted (and incorporated into higher-order types by using the notion of potency), the source of the principle of identity becomes hidden.

It is worth noticing that the deep instantiation approach allows the modeler to represent the base type if it is deemed desirable. However, if the modeler decides to represent the base type, the approach does not provide constructs to represent the relation between it and the higher-order type, not distinguishing thus between the different possible kinds of cross-level relations. As a consequence, the approach “as is” does not provide mechanisms to check if the rules concerning these relations are respected, e.g., to guarantee that all instances of the higher-order type (“Mobile Phone Model”) specialize the base type (“Mobile Phone”). We believe that the relations and rules we discuss here could be used to further evolve the deep instantiation approach.

In addition to having mechanisms aimed at simplifying the models by omitting base types, some recent deep instantiation approaches (KENNEL, 2012) also support the representation of a particular kind of regularity attribute by using a combination of the notions of attribute durability and mutability. The durability of an attribute indicates how far the attribute spans in an instantiation tree. The mutability of an attribute defines how often the attribute value can be changed over the instantiation tree. Consider for example a class such as “MobilePhoneModel” with potency 2. An attribute “screenSize” with durability 2 and mutability 1 will be given a value at the first instantiation (e.g., stating that the “Iphone5” has “screenSize” equals to 4 inches), and that value will determine the value of “screenSize” for the instances of instances of “MobilePhoneModel” (thus, all Iphone5s have a screen size of 4 inches). In our view, this representation captures the constraint relating an attribute of a first-order type (“screenSize”) and a regularity attribute of a second-order type (“instancesScreenSize”) as a single attribute with durability 2 and mutability 1 in a clabject with potency 2. This is a useful language mechanism for this particular kind of regularity attribute, which fully determines the value of the lower-level attribute. Unfortunately, this representation strategy is only capable of capturing the constraints

involving regularity attributes that determine the exact value that must be assigned to attributes in the lower order. It is not applicable, for example, to capture the constraint involving the regularity attributes “instancesMinStorageCapacity” and “instancesMaxStorageCapacity” of “MobilePhoneModel”: an instance *i* of “MobilePhone” must have assigned to the attribute “storageCapacity” a value equal or higher than the value assigned to “instancesMinStorageCapacity” and equal or lower than the value assigned to “instancesMaxStorageCapacity” of the “MobilePhoneType” instantiated by *i*. Thus, since the values of “instancesMinStorageCapacity” and “instancesMaxStorageCapacity” are not directly reflected in the value of a mobile phone attribute, they would be given durability and mutability of 1.

In an analysis of deep instantiation, Neumayr et al., (2014) observe that the approach is unable to capture certain domain scenarios in which a clabject is related to other clabjects at different instantiation levels. For example, consider a scenario in which every instance of “MobilePhoneModel” has a “designer” being an instance of “Person” and every instance of an instance of “MobilePhoneModel” (i.e., every instance of “MobilePhone”) has an “owner” which is also an instance of “Person”. In this scenario the type “Person” should have a relation with “MobilePhoneModel” (called “designer”) and another relation with “MobilePhone” (called “owner”). Considering that “Person” and “MobilePhone” are in the same level, the “owner” relation does not cross level boundaries. Nevertheless, “MobilePhoneModel” is placed in one level higher, and thus, the “designer” relation is crossing level boundaries, which is not allowed in that approach. Because of that, the authors introduce a Dual Deep Instantiation (DDI) approach distinguishing between source potency and target potency. An association thus becomes characterized by two potency numbers. Thus, in the aforementioned example, the “designer” relationship between “Person” and “MobilePhoneModel” would have both source and target potencies of 1 whereas the “owner” relationship would be defined having source potency of 1 and target potency of 2 (meaning that ownership relations hold between instances of “Person” and instance of instances of “MobilePhoneModel”). Another approach that allows for the representation of this kind of domain scenario is discussed in (ATKINSON; KUHNE, 2001). The approach is based on the definition of the so-called “metamodeling spaces”, each of which defines a separate set of instantiation levels. Levels in one metamodeling space are independent of levels in other spaces. As a consequence, an element in a particular metamodeling space *S* may be simultaneously related to elements in different levels as long as the target elements are not in *S*. For example, “Person” in a space *P* could be related to both “Mobile Phone” and “Mobile Phone Model” placed in different levels in another space *M*. Differently from (NEUMAYR et al., 2014) and (ATKINSON; KUHNE, 2001), our approach accommodates the domain scenario without a special mechanism, since relations between elements at different orders are allowed. In the example considered, we would represent the omitted base type “MobilePhone” defining that

“MobilePhoneModel” *partitions* “MobilePhone”, capturing thus, the fact that every instance of “MobilePhone” is instance of one instance of “MobilePhoneModel”. The “designer” relationship would be placed between “MobilePhoneModel” and “Person” whereas the “owner” relationship would be defined between “MobilePhone” and “Person”.

In (NEUMAYR; GRÜN; SCHREFL, 2009) another multi-level modeling approach that applies the notion of deep instantiation is proposed. The focus of this approach is also on reducing “unnecessary complexity”, improve readability and simplify maintenance and extension. The approach is founded in the concepts of *m-objects* and *m-relationships*. *M-objects* encapsulate different levels of abstraction that relate to a single domain concept. Analogously, *m-relationships* describe “relationships between *m-objects* at multiple level of abstraction”. An *m-object* can *concretize* another *m-object*. The *concretize* relationship comprises classification, generalization and aggregation relationships between the levels of an *m-object* (NEUMAYR; GRÜN; SCHREFL, 2009). We observe that this is a semantic overload between three relationships of quite different ontological nature, which could affect the understandability and usability of the approach.

Besides the so far mentioned initiatives, many other works focusing on deep instantiation-based approaches can be found in the literature, proposing alternative formalizations (ATKINSON; GERBIG, 2012; NEUMAYR; GRÜN; SCHREFL, 2009; ROSSINI et al., 2014), exploring its uses in different contexts (LARA et al., 2013; LARA; GUERRA; CUADRADO, 2014), and proposing tools for automated support (ATKINSON; GERBIG, 2012; DE LARA; GUERRA, 2010). These works focus on deep instantiation, which illustrates its wide acceptance as a basic mechanism for multi-level modeling approaches. However, none of these approaches aim at providing a semantic account that can be used to explain regularity attributes in deep instantiation and that supports the power type pattern and its variations.

With regard to the requirements we pursue in MLT, the deep instantiation approaches are, in general, characterized for admitting entities in multiple classification levels and for providing mechanisms to organize and capture the instantiation chains allowing an arbitrary number of classification levels. These approaches, therefore, meet requirements R1, R2 and R3. Such approaches usually assume that instantiations are the only relations that may cross level boundaries and do not provide constructs to capture rules concerning instantiation in different levels (such as the cross-level relations of MLT). Therefore, we consider that deep instantiation approaches in general do not satisfy requirements R4 nor R6 (although, as we have discussed, Neumayr et al., (NEUMAYR et al., 2014) propose a workaround to allow domain relations between elements in different levels). Finally, as we have discussed, some deep instantiation-approaches (such as (KENNEL, 2012)) provide mechanisms to capture particular kinds of rules relating features of entities in different levels. Thus, we consider they partially meet requirement

R5. Table 4 augments Table 3 including our analysis concerning the adequacy of deep instantiation-based approaches to the requirements we have established.

Table 4 - Requirements addressed by the analyzed multi-level modeling approaches.

Requirement	Approaches based on Odell's notion	Deep Telos	Boro	Deep-instantiation based Approaches
<i>R1 – To account for entities of multiple classification levels</i>	Partially.	Yes.	Yes.	Yes.
<i>R2 – To define principles for the organization of entities into levels</i>	No.	Partially.	Partially.	Yes.
<i>R3 – To allow a flexible number of classification levels</i>	Yes.	Yes.	Yes.	Yes.
<i>R4 – To account for rules for the instantiation of types at different levels</i>	Partially.	Partially.	Yes.	No.
<i>R5 - To account for rules relating features of entities in different levels</i>	No.	No.	No.	Partially.
<i>R6 – To admit the existence of domain relations between entities in different levels</i>	Yes.	Yes.	Yes.	No.

3.11 Final Considerations

This chapter presented a formal theory for multi-level conceptual modeling called MLT. The theory is formally defined using first-order logic and its consistency is verified using a lightweight formal method. Both the basic types and the structural relations defined in the theory are founded on the basic notion of (ontological) instantiation, which is applied regularly across levels, following the principle of strict (meta-)modeling. We have shown how the elements of the theory can be used as foundations for a domain theory: domain types instantiate and specialize the basic types of the theory.

To verify the consistency of MLT we have used Alloy (JACKSON, 2006). The axioms of MLT were represented as facts and the theorems were defined as assertions in an Alloy module. It allowed us to verify the satisfiability of our theory, to conduct some model simulations and to verify the theorems whose informal proofs have been discussed in this chapter. Appendix A presents a specification of MLT encompassing the axioms, definitions and theorems discussed in Sections 3.2 - 3.4. Such specification, thus, does not address dynamic classification. Appendix B,

in its turn, presents a specification of MLT that address dynamic classification, following the discussion presented in Section 3.7.

Using the structural cross-level relations defined in the theory (*is power type of*, *categorizes*, *partitions*), we are able to account for the different notions of power type in the literature, as well as to contrast and relate them. Since these relations are ultimately explained in terms of instantiation between entities of adjacent levels, the consequence of our account of power types is that we can formally harmonize power type and clabject-based approaches.

With respect to intra-level relations, we define the “ordinary” specialization relation and a *subordination* relation between higher-order types of the same order. Subordination allows for the creation of expressive multi-level models; subordination between higher-order types implies specialization between instances of the types related by subordination. An example of the usefulness of the subordination relation is shown in the biological taxonomy domain, in which taxonomic ranks (instances of “Second-Order Type”) are related by subordination in a sequence (with lower ranks subordinated to higher ranks). This ensures the taxonomy at the first-order level has an adequate structure (a taxonomic tree).

In order to facilitate the readers’ first contact with MLT, we have opted to suppress two direct extensions of the theory in the early part of this chapter: (i) the support for non-rigid types, and (ii) the generalization of the notion of order to support an infinite number of classification levels. With respect to (i) this extension is presented in Section 3.7. It allows instantiation to be contingent, thereby enabling dynamic classification, which is an important feature for conceptual modeling (GUIZZARDI, 2005). With respect to (ii), axioms A3, A4 and A5 would give way to an inductive definition for a basic type T_{i+1} based on the definition of the basic type at an immediately lower order T_i . The “disjointness” axiom (A6) would be modified accordingly.

The whole theory presented here is built up from an ‘opaque’ notion of instantiation, i.e., using instantiation as a primitive notion and not appealing to the ‘internals’ of intensions. The resulting theory is thus independent of any modeling choices or ontological commitments concerning the nature of ‘intensions’ of types. Naturally, this could be worked out in an extension of this work. To formally discuss the nature of ‘intensions’ of types, one could either opt for using a higher-order logics (which we avoid here intentionally since we aim at a more approachable formalization) or to reify and treat the intensions of types as structured elements with ‘parts’ or “constituents pretty much like the linguistic expressions that we use to speak about them” (SWOYER; ORILIA, 2014). The adequacy of these approaches is an issue for further investigation.

It is important to stress that it is not our intention in this chapter to propose a multi-level conceptual modeling language. Instead, we focus on the concepts that would constitute an adequate semantic domain for such a language. The theory we propose can be considered a

reference top-level ontology for types, with the main purpose of clarifying key concepts and relations for multi-level conceptualizations.

As discussed in (GUIZZARDI, 2005), a reference theory can be used to inform the revision and redesign of a modeling language, first through the identification of semantic overload, construct deficit, construct excess and construct redundancy, but also through the definition of modeling patterns and semantically-motivated syntactic constraints (CARVALHO; ALMEIDA; GUIZZARDI, 2014). This has been fruitful in the past in the revision of the UML, resulting in the OntoUML profile for conceptual modeling (GUIZZARDI, 2005). Thus, a natural application for MLT is to inform the (re-)design of a well-founded multi-level conceptual modeling language. Some earlier results to that extent are presented in Sections 3.1 - 3.4, showing: (i) how theorems of the theory reveal useful syntactic constraints for multi-level domain models; and (ii) how patterns of domain entities that are admissible by the theory can be reflected in modeling patterns. We have also been able to spot a deficiency in the UML given its reliance on the construct of generalization set to represent the power type pattern. Further, we have been able to identify cases of semantic overload in the power-type based technique presented in (GONZALEZ-PEREZ; HENDERSON-SELLERS, 2006), and in the m-objects approach (NEUMAYR; GRÜN; SCHREFL, 2009). Recently, Recker et al. (2011) reported results from a study with 528 modelers demonstrating that “users of conceptual modeling grammars perceive ontological deficiencies to exist and that these deficiency perceptions are negatively associated with usefulness and ease of use of these grammars”. This highlights the potential practical implications of our theory.

Next chapter presents a modeling approach based on the use of MLT as the top-most layer of UFO and incorporating the typology of universals of UFO as instances of “Second-order type” and specializations of “First-Order Type”. The resultant approach supports the design of conceptual models that can represent types as well as types of types while adhering to the rules of both MLT and UFO. Further, in Chapter 6, MLT is used as a reference to found the proposal of a UML extension that enhances its support to represent of power types.

Chapter 4. Multi-Level Ontology-based Conceptual Modeling with MLT and UFO

Both ontology-based conceptual modeling and multi-level modeling provide guidance to improve the quality of conceptual models: while the former addresses the use of ontological distinctions to reduce the potential misunderstandings and inconsistencies in conceptual models, the latter focuses on providing adequate support to the representation of multi-level phenomena. This chapter presents a modeling approach that benefits from both ontology-based conceptual modeling and multi-level modeling. It is founded on the combination of the formal multi-level theory we propose in Chapter 3. (MLT) with the Unified Foundational Ontology (UFO). This modeling approach aims to support the construction of ontologically well-founded conceptual multi-level models by leveraging the ontological distinctions put forth by UFO to domains dealing with types of types.

This chapter is further organized as follows. Section 4.1 presents an overall discussion about our modeling approach. Section 4.2 discusses the combination of MLT and UFO to provide foundations for ontology-based multi-level modeling. Section 4.3 identifies guidelines for multi-level conceptual modeling that arise from the MLT-UFO combination. Section 4.4 positions our modeling approach with respect to the existing work on multi-level conceptual modeling, and Section 4.5 presents some final considerations.

4.1 Overview

Our approach to multi-level ontology-based conceptual modeling proposes the construction of a hierarchy of ontologies founded on the combination of MLT and UFO. MLT is used as the topmost layer of the hierarchy. The basic pattern of MLT is applied to establish the relation between MLT and UFO, and later to establish the relation between a domain ontology and MLT-UFO. More specifically, the concepts of UFO instantiate and specialize elements of MLT, thereby respecting MLT's axioms and leveraging the use of structural relations and patterns of MLT in UFO. In turn, the concepts of domain conceptual models instantiate and specialize concepts of MLT and UFO, respecting all rules and patterns of both MLT and UFO.

The main goal of combining MLT and UFO to provide a foundational structure for conceptual modeling is to conjoin MLT notions of (i) (ontological) instantiation which is applied regularly across levels and (ii) cross-level relations that may occur between elements of different classification levels with the (iii) ontologically well-founded taxonomies of (first-order) universals and individuals of UFO. Consequently, conceptual models built with the MLT-UFO

combination benefit from the ontological distinctions of UFO as well as MLT concepts and patterns for multi-level modeling.

Here we use the term conceptual models as a general term to refer to a set of types of conceptual models concerning to different levels of generality. For example, following the classification of ontologies concerning their generality level proposed in (FALBO et al., 2013) (see Figure 2 in Chapter 2), we can imagine a hierarchy of models in which the MLT-UFO combination plays the role of a foundation ontology serving as basis to the construction of core ontologies. Core ontologies may be constructed applying the MLT-UFO combination, i.e. the core ontology concepts instantiate and specialize MLT-UFO concepts. Then, domain specific ontologies may extend core ontologies by instantiating and specializing the core ontology distinctions with the concepts that are required in a particular domain. The domain ontologies, in turn, may be extended by application ontologies that specialize and instantiate their general domain concepts to specific applications in such domains. This pattern can be repeatedly applied to create as much conceptual models specialization levels as necessary. This scenario is illustrated in Figure 24. Therefore, the first step to enable this modeling approach is to combine MLT and UFO.

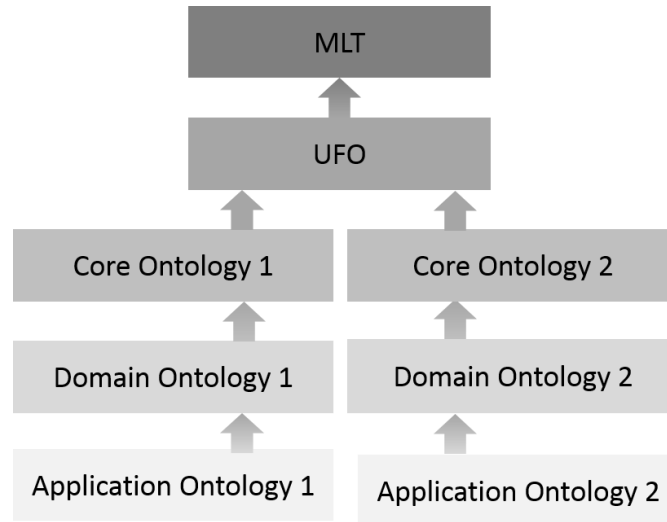


Figure 24 - Illustrating a hierarchy of conceptual models founded on the MLT-UFO combination.

4.2 Combining MLT and UFO

The relation between MLT and UFO is established through the application of the MLT basic pattern. In this setting the concepts of UFO *instantiate* and *specialize* elements of MLT. Since the UFO notion of individual is coincident with the MLT notion of Individual and the UFO notion of universal is encompassed by the MLT notion of first-order type (“1stOT”), we consider that the concepts in UFO taxonomy of individuals are instances of “1stOT” specializing “Individual” while the concepts in UFO taxonomy of universals are instances of “2ndOT” specializing “1stOT”. For each entity in the taxonomy of individuals (e.g., “Endurant”, “Substantial”,

(GUIZZARDI, 2005). Figure 26 uses associations to illustrate the UFO notions of *inherence* and *characterization*.

Assuming the naming conventions adopted in Figure 26 we can state that, in MLT terms, the attribute “characterized universals” of “Intrinsic Moment Universal” is a *regularity attribute* that constrains the possible values for the attribute “bearer individual” of the type “Intrinsic Moment”. For example, according to Figure 26, instances of “Color” may inhere in instances of “Physical Object” because “Color” characterizes “Physical Object”. In other words, since “Color” has “Physical Object” as a “characterized universal”, instances of “Color” have instances of “Physical Object” as “bearer individual”. Conversely, “characterizer universals” is a *regularity attribute* of “Substantial Universal” that constrains the possible values for the attribute “inherent moments” of “Substantial”. Therefore, instances of “Physical Object” have instances of “Color” as “inherent moments” since “Physical Object” has “Color” as a “characterizer universal”.

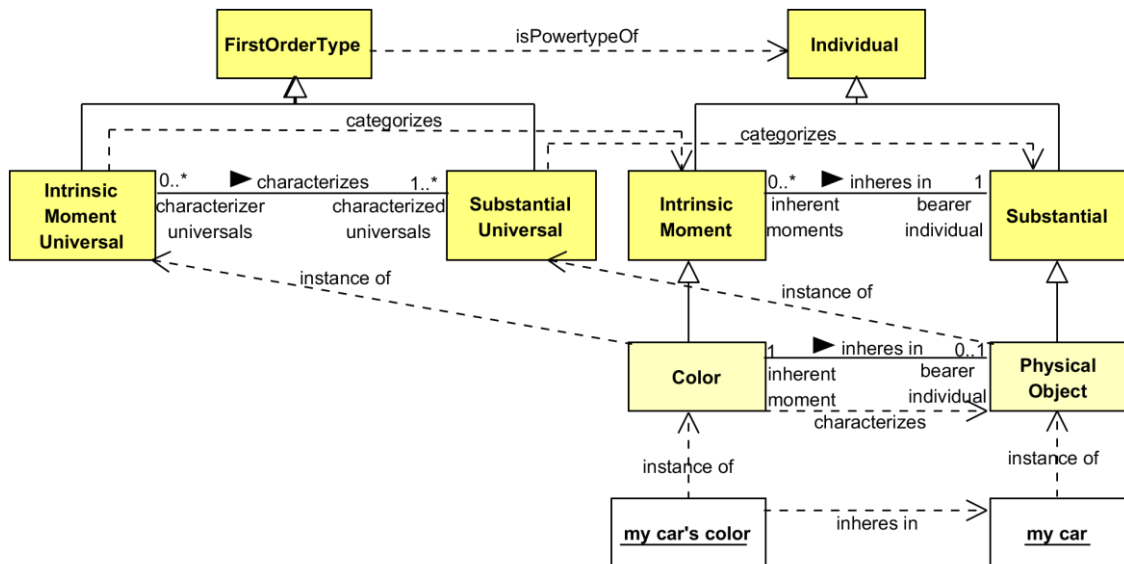


Figure 26 - Applying the MLT notion of regularity attribute to interpret the UFO notion of *characterization*.

This view of *characterization* as *regularity attribute* is used in Section 4.3.3 to propose a strategy to make explicit the relations between taxonomies of substantials and taxonomies of intrinsic moments. Similarly, the UFO notion of *mediation* may also be viewed as a *regularity attribute*. Therefore, a similar strategy can be applied to make explicit the relations between taxonomies of substantials and taxonomies of relators. This issue is also discussed in Section 4.3.3.

4.3 Guidelines for the Application of MLT-UFO Combination

In order to benefit from the ontological distinctions of UFO as well as the basic concepts and patterns for multi-level modeling of MLT, conceptual models built with the MLT-UFO

combination must adhere to the rules of both theories. Thus, every domain *first-order type* must: (i) instantiate one of the leaf ontological categories of UFO taxonomy of universals (and, consequently, instantiate MLT “1stOT”); and (ii) simultaneously, specialize one of the leaf ontological categories of UFO taxonomy of individuals (and thus, specialize “Individual”). For example, consider “Legal Entity” as instance of “Category”. Since “Category” specializes “1stOT” we conclude that “Legal Entity” is also a first-order type. Considering that “Category” categorizes “Substantial”, it follows that “Legal Entity” specializes “Substantial” (and, indirectly, specializes “Individual”). Analogously, “Person” and “Organization” are instances of “Kind” and specialize “Legal Entity” (indirectly specializing “Substantial” and “Individual”).

In our approach, we introduce *domain second-order types* as specializations of one of the leaf categories of UFO taxonomy of universals. In order to clarify which *first-order type* is ultimately specialized by instances of a *second-order type*, every domain second-order type has an MLT cross-level relation (i.e., *categorizes*, *disjointly categorizes*, *completely categorizes* or *partitions*) with a first-order type, which is termed its *base type*. Further, we consider that a *second-order type* also defines the *classification criteria* that its instances must apply to specialize the *base type*. As a result, we can use the MLT-UFO combination to provide rules and patterns for introducing second-order types in ontology-based domain models.

The rules for introducing second-order types in ontology-based domain conceptual models are defined considering the UFO leaf categories being specialized. For each specialization case, we identify the admissible ontological categories for a (first-order) base type as well as the MLT cross-level relations that may be established between the second-order type and the base type. This is possible by combining the MLT basic pattern, which determines that each instance of a second-order type specializes a (first-order) base type, with UFO rules that determine constraints for specialization relations according to the types’ ontological categories (discussed in Section 2.2.2).

As a general rule, we have that, according to UFO, substantial universals cannot specialize moment universals and vice-versa (*rule U1*), and thus a specialization of “Substantial Universal” cannot have an instance of “Moment Universal” as a base type. Conversely, a specialization of “Moment Universal” cannot have an instance of “Substantial Universal” as a base type. Specific rules that apply to second-order types specializing each leaf category of the UFO taxonomy of universals are considered in the sequence. First, in Section 4.3.1, we consider second-order types specializing “Sortal Universal” (viz., the leaf categories “Kind”, “Subkind”, “Phase” and “Role”). Second, in Section 4.3.2, we consider second-order types specializing “Mixin Universal” (viz., the leaf categories “Category”, “Phase Mixin”, “Role Mixin”). Finally, in Section 4.3.3, we consider second-order types specializing “Moment Universal” (viz., “Quality Universal”, “Mode Universal” and “Relator Universal”) and discuss the relation between taxonomies of moments and taxonomies of substantials.

4.3.1 Second-Order Types Specializing “Sortal Universal”

Specializations of “Kind”

According to UFO, kinds are sortal universals that supply identity criteria to their instances, and hence, they cannot specialize another sortal universal (*rule U5*). This rules out sortal universals as base types for second-order types *specializing* “Kind”. Further, considering that kinds are *rigid universals*, they cannot specialize *anti-rigid universals* (*rule U4*), which also rules out anti-rigid universals as base types for specializations of “Kind”. Therefore, by excluding sortal universals and anti-rigid universals as admissible base types, a second-order type *specializing* “Kind” must have a *rigid mixin sortal* as *base type* (i.e., an instance of “Category”).

Moreover, considering that each individual must be instance of exactly one kind, we can more specifically state that a specialization of “Kind” must *partition* an instance of “Category”, using as *classification criteria* the principle of identity supplied by the kind. For example, considering “Legal Entity” as a “Category” that generalizes properties of different kinds of legal entities, we may define a *second-order type* “Legal Entity Kind” that *specializes* “Kind” and *partitions* “Legal Entity” having as instances the kinds “Person” and “Organization”. Figure 27 illustrates this scenario.

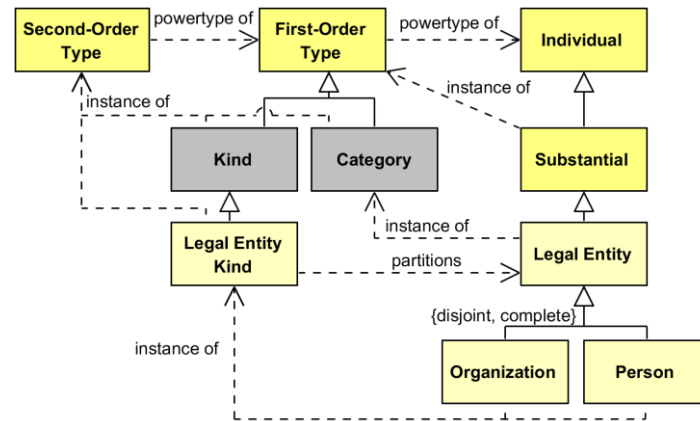


Figure 27 - A domain second-order type specializing “Kind” and partitioning an instance of “Category”.

Specializations of “Subkind”

Since subkinds are *rigid universals*, they cannot specialize instances of *anti-rigid universals* (*rule U4*). Thus, a *second-order type* specializing “Subkind” must have, as *base type*, an instance of “Rigid Sortal” (i.e., an instance of “Kind” or “Subkind”) or an instance of “Rigid Mixin” (i.e., an instance of “Category”).

Subkinds are common in taxonomies in which the more specific types form a partition of a more general type distinguishing instances according to immutable intrinsic properties (e.g., “Person” specialized into “Man” and “Woman” according to gender). In these cases, a second-order type that *specializes* “Subkind” *partitions* an instance of “Kind”, “Subkind” or “Category”

using some immutable intrinsic properties exemplified by their instances as classification criteria (see “Person Gender Subkind” with instances “Man” and “Woman” in Figure 28).

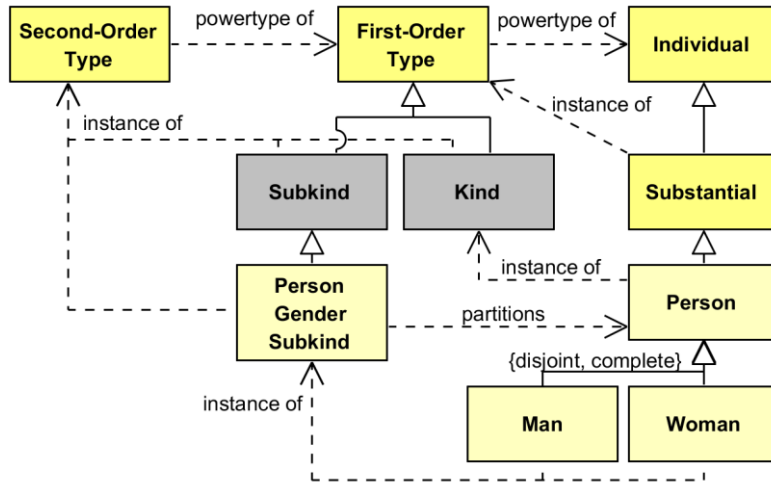


Figure 28 - A domain second-order type specializing “Subkind” and partitioning an instance of “Kind”.

Another example of *second-order type* that can be represented as a specialization of “Subkind” is “Horse Breed”, *partitioning* “Horse” having instances such as “Mustang Horse” and “Arabian Horse”. Note that MLT does not force the modeler to enumerate the instances of second-order types (such as “Horse Breed”), while still capturing the fact that its instances form a partition of the base type (“Horse”) (see Figure 29).

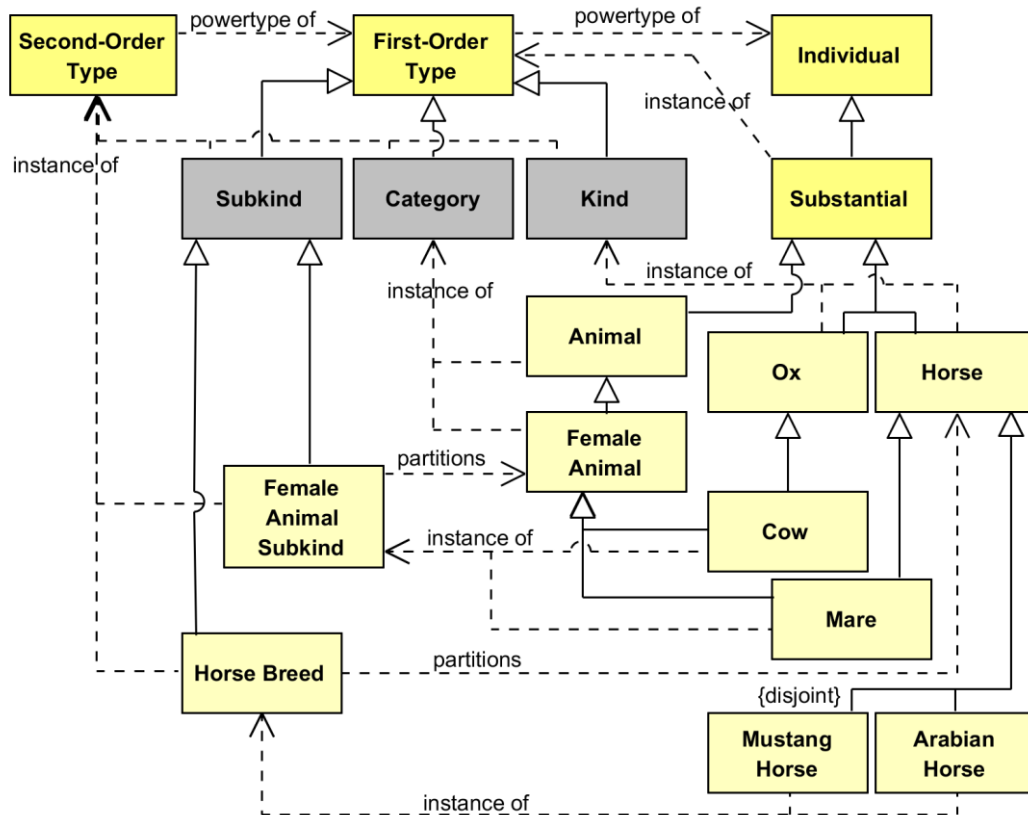


Figure 29 - Domain second-order types specializing “Subkind” and partitioning instances of “Category” and “Kind”.

Differently from the cases in which the base type is a kind or a subkind, when the base type is a category, in addition to specializing the base type, the instance of the higher-order type must also specialize some instance of “Kind”. This is a consequence of *rule U6*, which was captured through the *subordination* of “Subkind” to “Kind” in the MLT-UFO combination. In these cases, the principle of identity carried by the instances of the second-order type is not yet settled by the base type, and thus the principle of identity must be determined for each instance of the second-order type. In the example of Figure 29, the *second-order type* “Female Animal Subkind” *specializes* “Subkind” and *partitions* “Female Animal” having instances such as “Cow” and “Mare” (see Figure 29). “Cow” *specializes* “Ox”, inheriting its principle of identity, while “Mare” *specializes* “Horse”, inheriting thus a different principle of identity.

Specializations of “Phase”

According to UFO, instances of “Phase” are anti-rigid sortal universals. As such, they may specialize any substantial universal. A *second-order type* that *specializes* “Phase” may thus have as *base type* any instance of “Substantial Universal” regardless of rigidity and sortality (i.e., an instance of “Kind”, “Subkind”, “Phase”, “Role”, “Category”, “Phase Mixin” or “Role Mixin”).

As discussed in (GUIZZARDI, 2005), instances of “Phase” classify individuals of a specific kind using some mutable intrinsic property as classification criteria, forming partitions of a more general type. We can capture this notion with a *second-order type* that specializes “Phase” and *partitions* an instance of “Substantial Universal”. For example, in Figure 30, “Person Age Phase” *partitions* “Person” into “Child”, “Adult” and “Elder” according to the intrinsic property age.

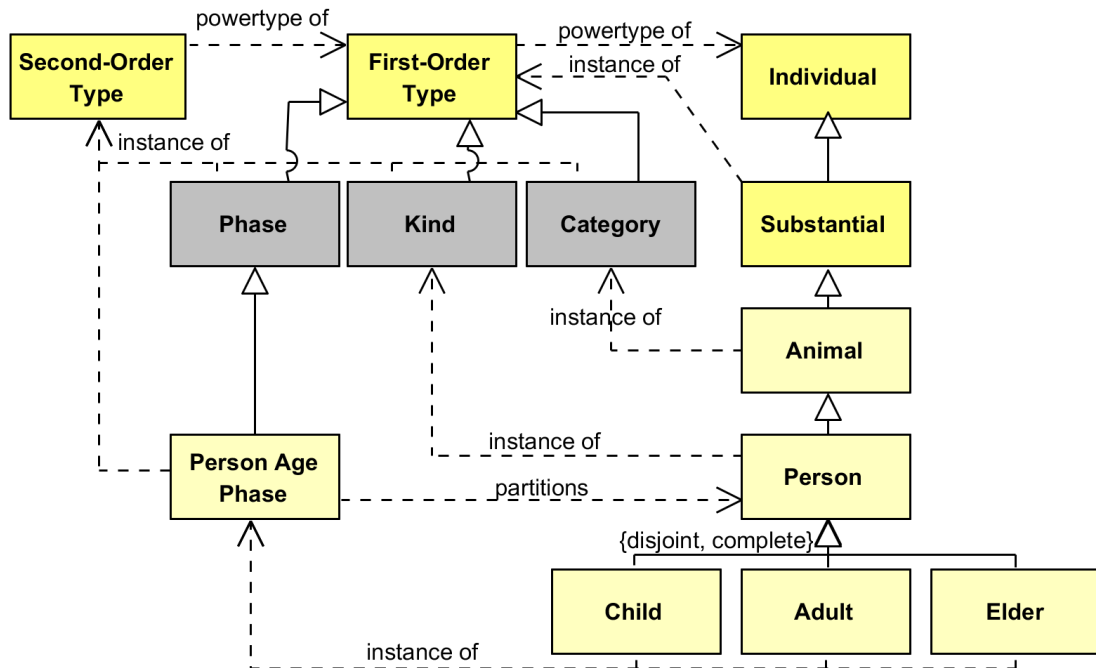


Figure 30 - A domain second-order type specializing “Phase” and partitioning an instance of “Kind”.

To capture domains in which some mutable intrinsic properties are applicable to individuals of different kinds, one may define *specializations* of “Phase” having as base types instances of “Mixin Universal” (i.e. “Category”, “Phase Mixin” and “Role Mixin”). In these cases, the principle of identity carried by the instances of the second-order type is determined for each instance, by establishing the kind that it specializes. In the example in Figure 31, the second-order type “Young Animal Phase” *partitions* “Young Animal” having “Child” and “Calf” as instances; “Child” *specializes* the kind “Person”, inheriting its principle of identity, while “Calf” *specializes* the kind “Ox”, inheriting thus a different principle of identity.

Figure 31 - A domain second-order type specializing “Phase” and partitioning an instance of “Phase Mixin”.

Similarly to instances of “Phase”, instances of “Role” are anti-rigid sortal universals and may specialize any substantial universal. Thus, a *second-order type* that *specializes* “Role” may have as *base type* any instance of “Substantial Universal” (i.e., an instance of “Kind”, “Subkind”, “Phase”, “Role”, “Category”, “Phase Mixin” or “Role Mixin”).

(ii) “Adult Role” whose instances specialize “Adult” (an instance of “Phase”) and include those roles that are played exclusively by adults, such as “Driver” and “Congressman”; and, (iii) “Employee Type” whose instances specialize “Employee” (an instance of “Role”) into “Temporary Employee” and “Permanent Employee”, considering the type of work contract they have. These examples of second-order types specializing “Role” are illustrated in Figure 32.

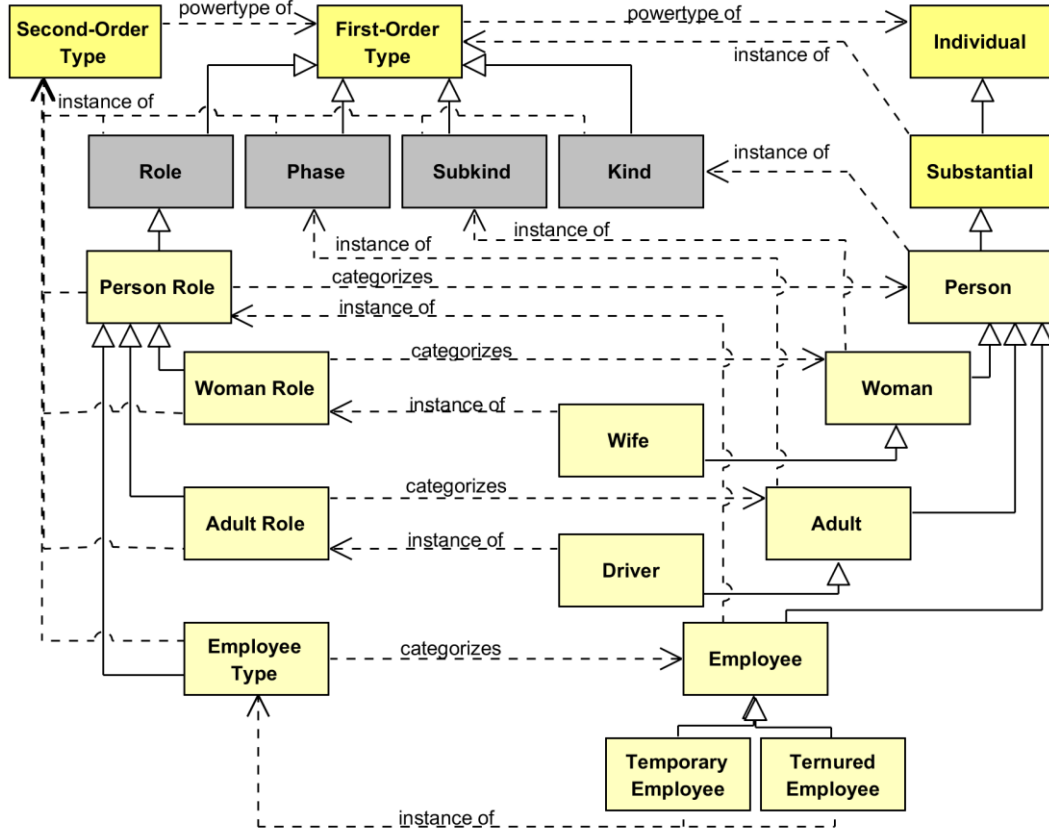


Figure 32 - Domain second-order types specializing “Role” and *categorizing* instances of “Sortal Universal”.

To capture domains in which roles can be played by individuals of different kinds, one may define second-order types that *specialize* “Role” having as base types instances of “Mixin Universal” (i.e., instances of “Role Mixin”, “Phase Mixin” or “Category”). In these cases, each instance of the second-order type must specialize a sortal universal to settle the principle of identity. For example, considering that “Customer” is a “Role Mixin” that generalizes a role that can be played by any “Legal Entity”, we may define a *second-order type* “Customer Type” that *specializes* “Role” and *partitions* “Customer” having as instances “Personal Customer” and “Corporate Customer”. “Personal Customer” *specializes* “Person”, inheriting its principle of identity, while “Corporate Customer” *specializes* “Organization”, inheriting thus a different principle of identity (see Figure 33).

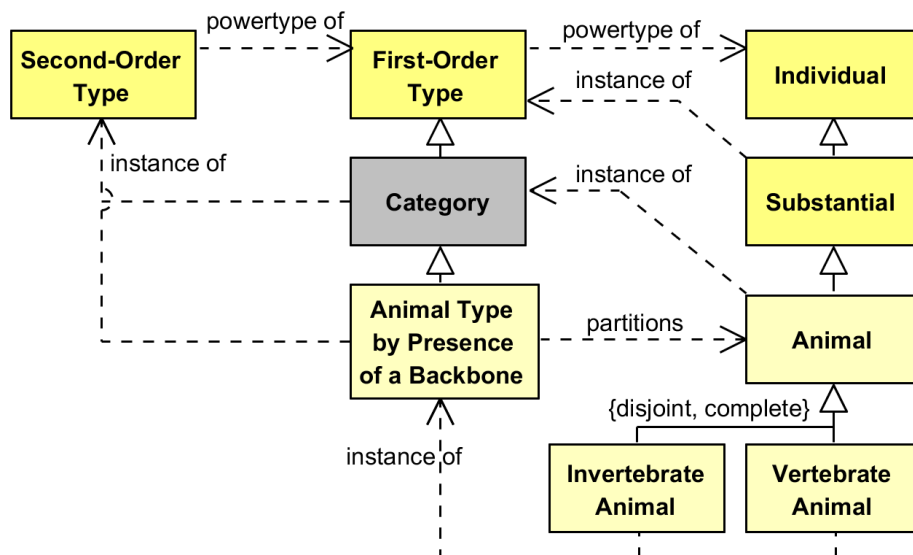


Figure 34 - A domain second-order type specializing "Category" and *partitioning* an instance of "Category".

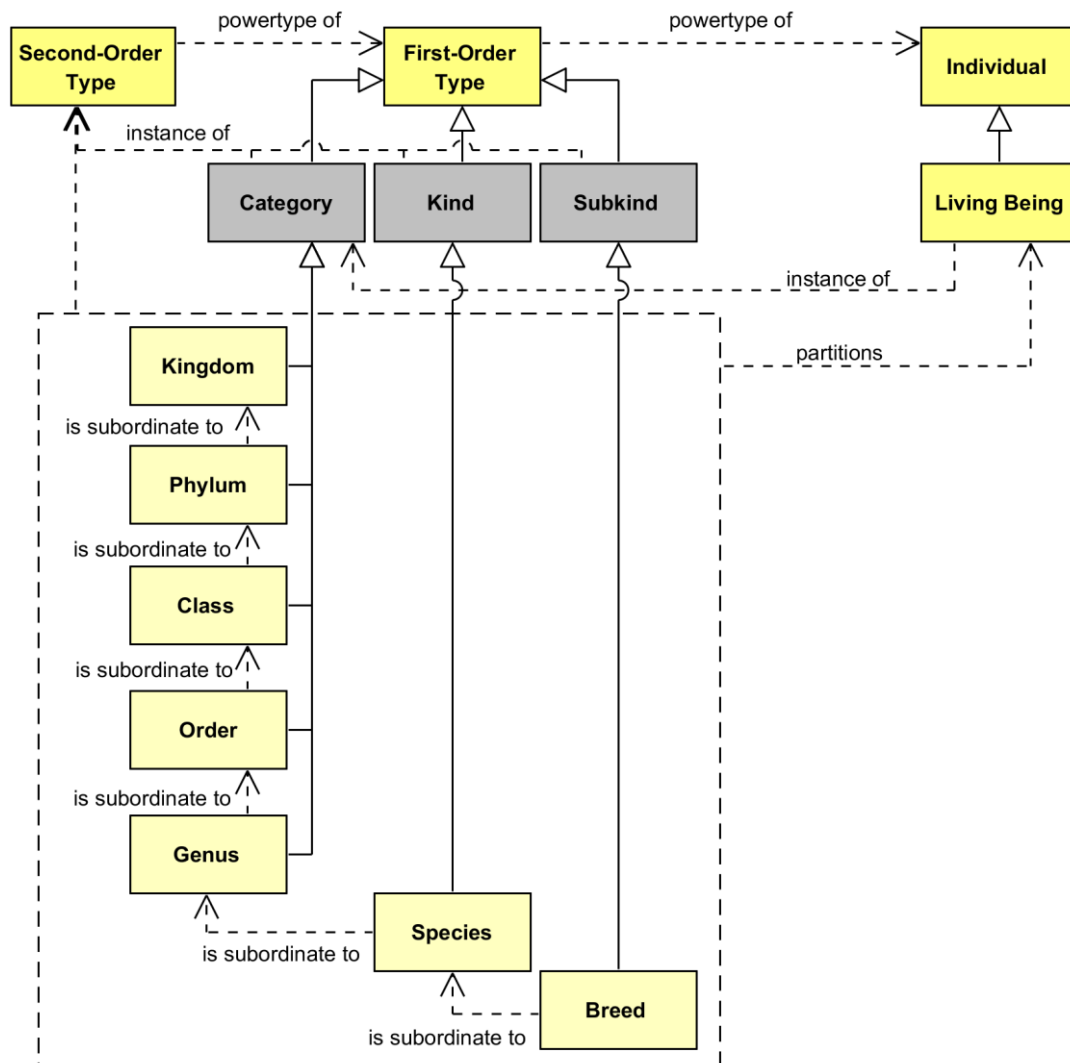


Figure 35 - Using MLT-UFO combination to describe concepts involved in biological taxonomy.

Other examples of second-order types specializing “Category” can be found in the biological taxonomy for living beings. Such taxonomy classifies living beings according to biological taxa in seven or more ranks, e.g., kingdom, phylum, class, order, genus, species, and breed. As discussed in Section 3.5, the seven biological ranks are *second-order types* that *partition* “Living Being” obeying a subordination chain such that “Phylum” *is subordinate to* “Kingdom”, “Class” *is subordinate to* “Phylum”, and so on. Assuming, that the identity criteria for living beings are provided by their species, we have that “Species” specializes “Kind” and *partitions* “Living Being”. In this case, since specializations of “Kind” may only be subordinate to specializations of “Category”, the five biological ranks to which “Species” *is subordinate*, namely “Kingdom”, “Phylum”, “Class”, “Order” and “Genus”, are *specializations* of “Category”. “Breed”, in its turn, is *subordinate to* “Species” and *specializes* of “Subkind”. This scenario is illustrated in Figure 35.

Specializations of “Phase Mixin”

Instances of “Phase Mixin” are anti-rigid mixin universals. As such, they may specialize any mixin universal. Thus, a *second-order type* that specializes “Phase Mixin” may have as *base type* any instance of “Mixin Universal” (i.e. an instance of “Category”, “Phase Mixin” or “Role Mixin”). Considering that *phase mixins* classify individuals of different kinds according to some mutable intrinsic property, the instances of “Phase Mixin” usually form partitions of a more general type. We can capture this notion with a *second-order type* that specializes “Phase Mixin” and *partitions* an instance of “Mixin Universal”. For example, in Figure 36, “Animal Phase” is a second-order type that *specializes* “Phase Mixin” and *partitions* “Animal” (an instance of “Category”) having as instances “Living Animal” and “Dead Animal”.

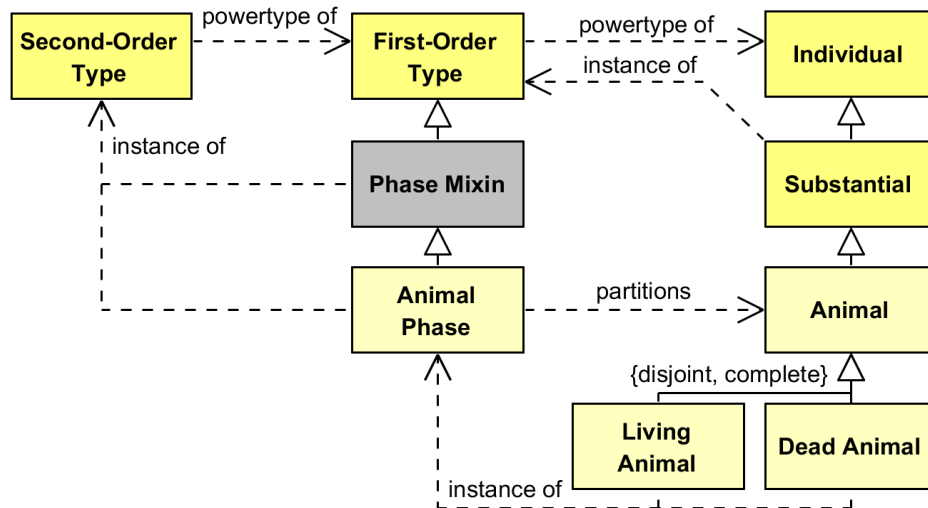


Figure 36 - Second-order types specializing “Phase Mixin”.

Specializations of “Role Mixin”

Similarly to instances of “Phase Mixin”, instances of “Role Mixin” are anti-rigid mixin universals and may only specialize other mixin universals. Thus, a *second-order type* that specializes “Role Mixin” may have as *base type* any instance of “Mixin Universal (i.e. an instance of “Category”, “Phase Mixin” or “Role Mixin”).

In contrast to specializations of “Phase Mixin”, second-order types specializing “Role Mixin” use as classification criteria some relational properties individuals bear in the scope of a relational context, allowing us to capture scenarios in which a role can be played by individuals of different kinds. For example, considering “Legal Entity” as a “Category” that generalizes properties of different kinds of legal entities, we may define a *second-order type* “Legal Entity Role” that *specializes* “Role Mixin” and *categorizes* “Legal Entity” having instances such as “Customer” and “Supplier”. The *second-order type* “Customer Role”, in its turn, specializes “Role Mixin” and *categorizes* “Customer” having as instances types that define roles that are played by customers such as “Account Holder” and “Insurance Beneficiary”. This scenario is illustrated in Figure 37.

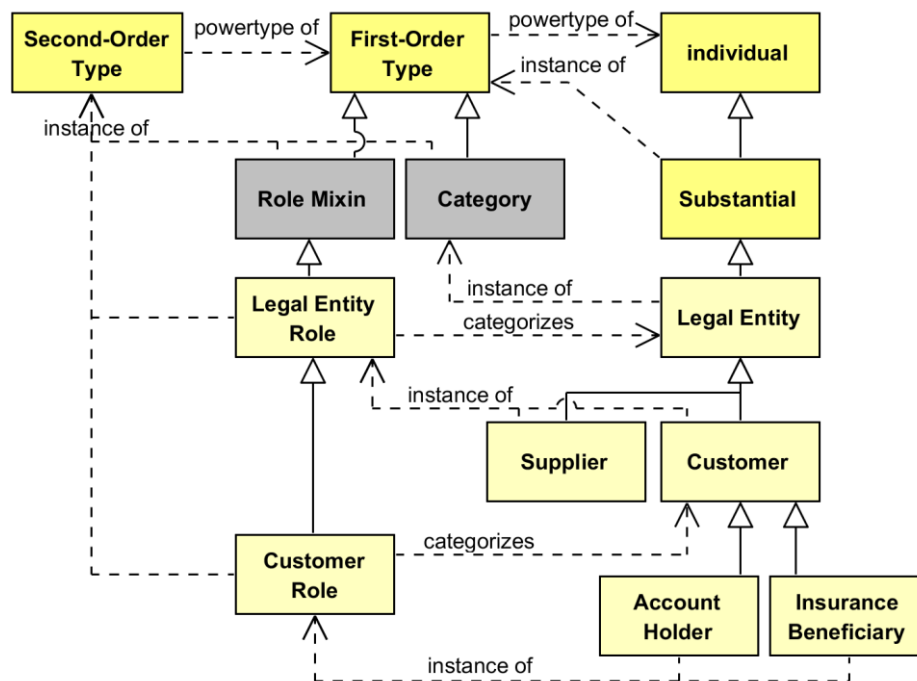


Figure 37 - Domain second-order types specializing “Role Mixin”.

4.3.3 Second-Order Types Specializing “Moment Universal”

Thus far we have focused on second-order types specializing “Substantial Universal”. As evidenced in the previous sections, these types define the criteria used by their instances to specialize a *base type* according to certain (intrinsic or relational) properties of substantials. These properties are formally accounted for in UFO using the notions of moment (particularized properties that are ultimately dependent on substantials) and moment universal (universals whose

instances are moments). As a consequence, in a conceptual domain model founded in UFO, moment universals are treated as types along with substantial universals. In this section, we address the second-order types specializing each leaf category of UFO’s taxonomy of moment universals, discussing rules for defining the admissible ontological category for their base types. We further explore the interrelations between moment universals and substantial universals in a domain conceptual model.

Differently from the taxonomy of substantial universals, in its original version, UFO does not elaborate on the rigidity and sortality of moment universals⁹. The UFO taxonomy of moment universals reflects the taxonomy of moment individuals, having thus three leaf concepts, namely “Quality Universal”, “Mode Universal” and “Relator Universal”. Considering that “Quality”, “Mode” and “Relator” are disjoint types *categorized* respectively by “Quality Universal”, “Mode Universal” and “Relator Universal”, it is not possible for an instance of one leaf concept of the moment universal taxonomy to specialize an instance of another leaf concept (*rule U2*). Adding to this the fact that instances of “Moment Universal” cannot specialize instances of “Substantial Universal” (*rule U1*), we conclude that specializations of each leaf concept of the moment universal taxonomy may only have instances of the same leaf concept as *base type*. Thus, specializations of “Quality Universal” must *categorize* instances of “Quality Universal”, specializations of “Mode Universal” must *categorize* instances of “Mode Universal” and specializations of “Relator Universal” must *categorize* instances of “Relator Universal”. We consider each of these cases in the sequence.

Specializations of “Quality Universal”

Following the theory of Conceptual Spaces (GÄRDENFORS, 2000), UFO assumes that for several perceivable or conceivable quality universals there is an associated *quality dimension* in human cognition. For example, “Age” and “Height” are associated with one-dimensional structures with a zero point isomorphic to the half-line of nonnegative numbers while “Color” can be associated to a structure (a quality domain) formed by three dimensions, named Hue, Saturation and Brightness. In Figure 38, following (GUIZZARDI, 2005), we use the UML construct of a structured datatype to model the color domain according to the Hue, Saturation and Brightness (HSB) scheme. In this representation, the datatype attributes “hue”, “saturation” and “brightness” are placeholders for the coordinates of each of the quality dimensions forming the color domain (GUIZZARDI, 2005). The separate identification of the “Color” quality universal allows us to use different quality structures if necessary, each of which with a separate

⁹There is a recent proposal for extending UFO in which the distinctions between sortals and mixins as well as between rigid and anti-rigid types are applied to moment universals (GUARINO; GUIZZARDI, 2015; GUIZZARDI et al., 2015b). This proposal, however, has neither been formally characterized nor incorporated into results derived from UFO (e.g., OntoUML). For this reason, this proposal is not considered here.

corresponding datatype. Further, we can account for the change in a quality of an entity by admitting that the value of the structured datatype may change. For example, we can account for the change of the color of a particular apple from green to red, and from red to brown by admitting changes on its values in the structured data type.

Second-order types specializing “Quality Universal” can be defined to *categorize* an instance of “Quality Universal” considering possible regions of the associated quality structure. For example, we can define a second-order type “Color Type” *specializing* “Quality Universal” and *categorizing* “Color” according to selected regions of a color domain having instances such as “Blue-Toned Color” and “Green-Toned Color” (see Figure 38). Since each instance of “Color Type” determines a region of the color domain, its instances (i.e., instances of “Color”) always have values for quality dimensions within the specified region. Determining whether the categorization is disjoint or not would allow us to represent whether there is overlap in the regions specified by the various instances of “Color Type”.

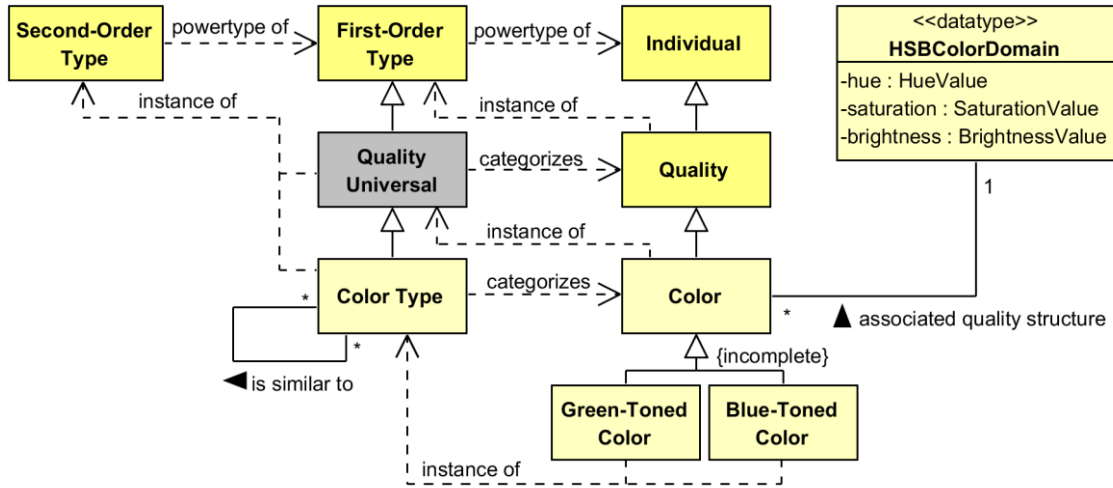


Figure 38 - Second-order types specializing “Quality Universal”.

The possibility of defining second-order types categorizing instances of Quality Universal allows us to define important relations between these instances, which appear very frequently in the characterization of quality universals. For instance, one might want to represent that a color type is similar to another one (a binary relation between instances of “Color Type”). For example, “Red-Toned Color” is similar to “Orange-Toned Color”, while “Red-Toned Color” is not similar to “Blue-Toned Color”.

As previously discussed, the relation between an intrinsic moment universal (a quality universal or mode universal) and a substantial universal is made explicit using the UFO notion of *characterization* (GUIZZARDI, 2005). For instance, recovering an example used in Section 4.2, consider an instance of “Category” called “Physical Object” which in a particular conceptualization encompasses all entities which are tangible and visible, such as a car, a fruit, and so on. To capture that a physical object has a color, we define that the quality universal

“Color” “characterizes” “Physical Object”, meaning that instances of “Color” are moments that may “inhere in” instances of “Physical Object”.

The same criteria used to specialize a quality universal can be used to specialize the characterized substantial universal, establishing a correspondence between the taxonomy of qualities and the taxonomy of substantials. For example, we could define substantial universals such as “Blue Object” and “Green Object” specializing “Physical Object” reflecting the specialization of “Color” into “Green-Toned Color” and “Blue-Toned Color”. In this setting, in addition to the second-order type specializing “Quality Universal”, we may define a second-order type specializing “Substantial Universal” whose instances are characterized by instances of the former. In the example at hand, we may define the domain second-order type “Physical Object Type by Color” with instances “Blue Object” and “Green Object”, each of which is characterized by an instance of “Color Type”. This is captured in the domain conceptual model of Figure 39, in two ways. First, the inherence relation between “Physical Object” and “Color” is specialized to link each instance of “Physical Object Type by Color” to the corresponding instance of “Color Type” (e.g., every instance of “Blue Object” bears an instance of “Blue-Toned Color”, every instance of “Green Object” bears an instance of “Green-Toned Color”, and so on). Second, a one-to-one association between “Physical Object Type by Color” and “Color Type” (characterizes) is used to represent that each instance of the former is defined considering an instance of the latter.

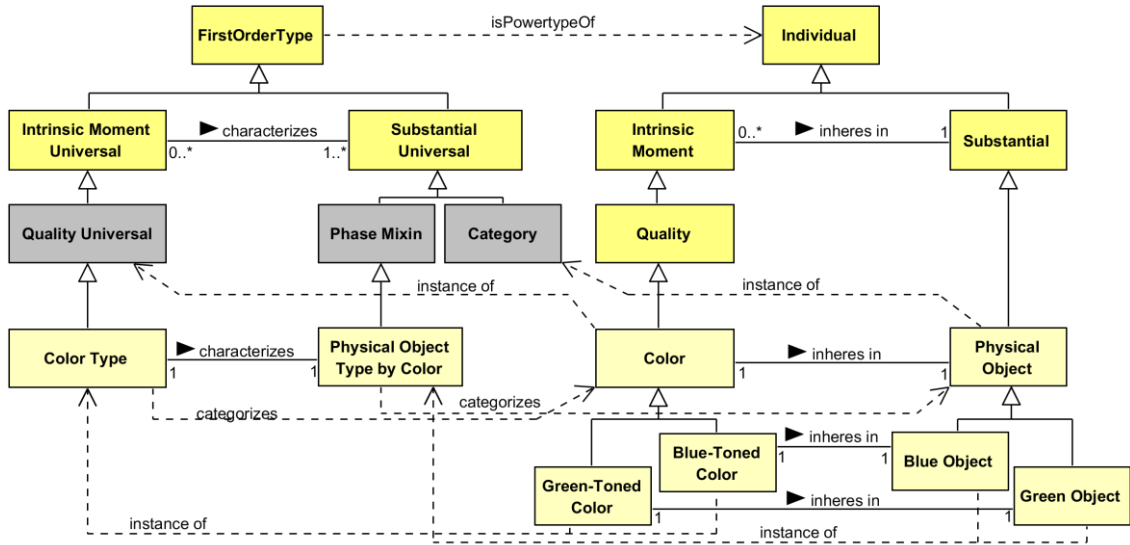


Figure 39 - Making explicit the relation between substantials and qualities taxonomies.

In Section 4.2 we apply the MLT notion of regularity attribute to interpret the UFO notion of characterization, concluding that “characterized universals” is a regularity attribute of “Intrinsic Moment Universal” that constrains the possible values for the attribute “bearer individual” of the type “Intrinsic Moment”, and that “characterizer universals” is a regularity attribute of “Substantial Universal” that constrains the possible values for the attribute “inherent moment” of the type “Substantial”. This scenario is illustrated in Figure 26. Figure 40 augments Figure 26 to show how the same view can be applied to explain the relation between a domain

second-order type specializing “Intrinsic Moment Universal” and a domain second-order type specializing “Substantial Universal”. According to Figure 40, the attribute “characterized universal” of “Color Type” can be seen as a regularity attribute that constrains the possible values for the attribute “bearer individual” of the type “Color”. For example, instances of “Blue-Toned Color” inheres in instances of “Blue Object” because “Blue-Toned Color” characterizes “Blue Object”. In other words, since “Blue-Toned Color” has “Blue Object” as a “characterized universal”, instances of “Blue-Toned Color” may have instances of “Blue Object” as “bearer individual”. Conversely, the attribute “characterizer universal” of “Physical Object Type by Color” is a regularity attribute that constrains the possible values for the attribute “inherent moment” of the type “Physical Object”. Therefore, for example, instances of “Blue Object” (such as my car) may have instances of “Blue-Toned Color” (such as my car’s color) as “inherent moment” because “Blue Object” has “Blue-Toned Color” as a “characterizer universal”.

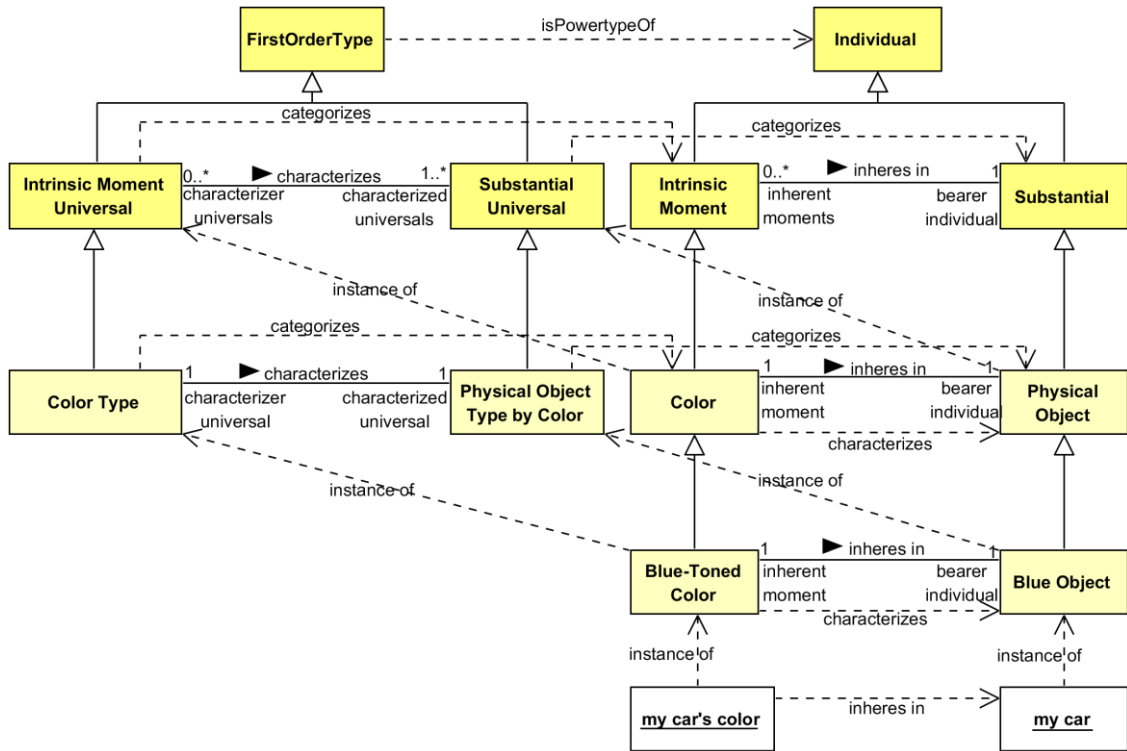


Figure 40 - Using the notion of regularity attribute to explain the relation between taxonomies of intrinsic moments and taxonomies of substantials.

To determine which of leaf ontological category is specialized by the introduced second-order type, we must consider: (i) the mutability of the quality involved in the adopted classification criteria (in this case, the mutability of the color of a physical object) and (ii) the ontological category of its base type (in this case, the ontological category of “Physical Object”).

For a mutable quality, the instances of the second-order type are in general anti-rigid types. For example, considering that the color of a physical object may change during its life cycle (migrating from the extension of a color type to another), we conclude that instances of “Physical Object Type by Color” are anti-rigid types (e.g. an apple may change from green to red, ceasing

to be an instance of “Green Object” and becoming an instance of “Red Object”). Considering “Physical Object” a category, two alternatives remain (“Phase” and “Phase Mixin”) for the leaf ontological category being specialized by the introduced second-order type, dependent on whether its instances carry a principle of identity. Considering that instances of “Physical Object Type by Color” have instances of different kinds, we define that it specializes “Phase Mixin” (see in Figure 39).

In the cases in which the qualities involved are immutable, we need to consider further the rigidity of the base type in order to determine whether the instances of the second-order type at hand are rigid or anti-rigid types. For an immutable quality and a rigid base type, the instances of the second-order type are rigid types, and thus, the second-order type must specialize either “Kind”, “Subkind” or “Category”. For example, consider a second-order type “Person Type by Gender” that partitions “Person” using a quality “Gender” as classification criteria. If we consider that the gender of a person is immutable and that “Person” is a rigid type, we can conclude that instances of “Person Type by Gender” (e.g. “Man” and “Woman”) are rigid types. Further, since the base type is a kind, “Person Type by Gender” must specialize “Subkind”. In contrast, for an immutable quality and anti-rigid base type, the instances of the second-order type are anti-rigid types and thus, the second-order type must specialize either “Phase”, “Role”, “Phase Mixin” or “Role Mixin”. For example, considering that “Child” is a phase, if we define a second-order type “Child Type by Gender” that uses gender as criteria to partition “Child” we have that “Child Type by Gender” specializes “Phase” with instances “Boy” and “Girl”.

Specializations of “Mode Universal”

Similarly to quality universals, mode universals may also be categorized according to some criteria. Thus, we may define a *second-order type* that *specializes* “Mode Universal” and *categorizes* an instance of “Mode Universal” using properties of the mode individuals as classification criteria. For example, we may define a *second-order type* “Disease Type” that *categorizes* “Disease” having instances such as “Diabetes” and “Hemophilia” (see Figure 41).

Considering that modes are reification of intrinsic properties inhering in substantials, similarly to quality universal, mode universals may *characterize* substantial universals. Further, the same criteria used by a second-order type specializing “Mode Universal” to categorize an instance M of “Mode Universal” can be used to categorize an instance S of “Substantial Universal” distinguishing instances according to the subtype of M they bear. For example, consider an instance of “Phase” called “Diseased Person” which encompasses all persons bearing some disease. To capture that diseases inhere in diseased persons (and, conversely, that diseased persons bear diseases), we may relate “Disease” to “Diseased Person” through a “inheres in” association. We may define a second-order type “Diseased Person Type” that categorizes “Diseased Person” according to the type of disease, having instances such as “Diabetic Person”

and “Hemophiliac Person”. In this scenario, each instance of “Disease Type” “characterizes” a specific instance of “Diseased Person Type”, meaning that the instances of each instance of “Disease Type” must “inheres in” one instance of a specific instance of “Diseased Person Type”. Therefore, the attribute “characterized universal” of “Disease Type” is a *regularity attribute* that constrains the possible values for the attribute “bearer individual” of the type “Disease”, and, conversely, the attribute “characterizer universal” of “Diseased Person Type” is a *regularity attribute* that constrains the possible values for the attribute “inherent individual” of the type “Diseased Person”

The same pattern we applied in the previous section to represent the relation between taxonomies of qualities and taxonomies of substantials is applied here to capture the correspondence between the taxonomy of modes and the taxonomy of substantials. As shown in Figure 41, it is captured in two ways. First, the “characterizes” association between “Intrinsic Moment Universal” and “Substantial Universal” is specialized giving rise to a one-to-one association between “Disease Type” and “Diseased Person Type”, representing that each instance of the latter is characterized by an instance of the former. Second, the “inheres in” relation between “Disease” and “Diseased Person” is specialized to link each instance of “Disease Type” to the corresponding instance of “Diseased Person Type” (e.g., every instance of “Diabetes” inheres in an instance of “Diabetic Person”).

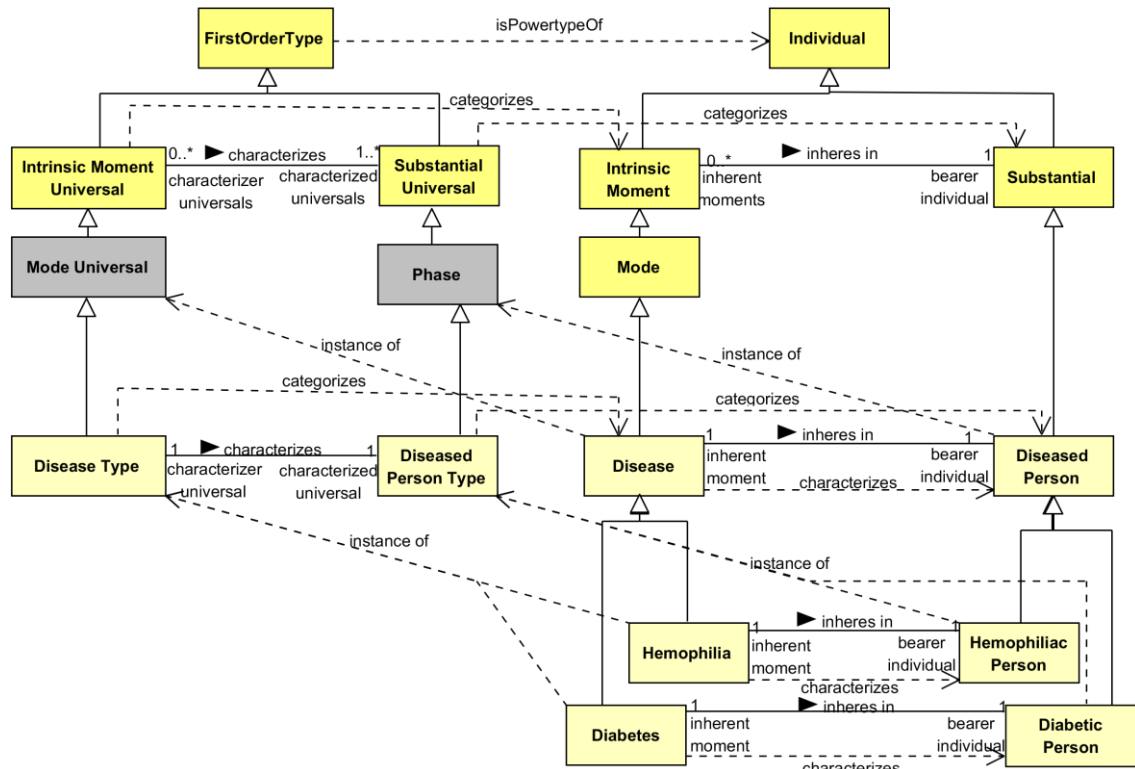


Figure 41 - Making explicit the relation between substantials and modes taxonomies.

Further, applying the same analysis we conducted in the previous section to determine which leaf ontological category is specialized by the introduced second-order type, we conclude

that “Diseased Person Type” specializes “Phase” since the base type “Diseased Person” is a “Phase”.

Specializations of “Relator Universal”

As discussed in (GUARINO; GUIZZARDI, 2015), besides having the power of connecting entities, relators also have their own intrinsic properties. Using these intrinsic properties as classification criteria, we may define specializations of “Relator Universal” that *categorize* instances of “Relator Universal”. For example, we may define a second-order type “Employment Type by Duration” that *partitions* “Employment” having “Temporary Employment” and “Tenured Employment” as instances (see Figure 42).

Relator universals may also be specialized considering properties of the connected entities (the relata). In this case, we may define a second-order type that *specializes* “Relator Universal” and *categorizes* an instance of “Relator Universal” using properties of the relata as criteria. For example, considering that important characteristics of employments (e.g., the types of claims and commitments entailed) vary according to the sort of the organization which is the employer, we may define a second-order type “Employment Type by Employer Nature” that specializes “Relator Universal” and *partitions* “Employment” having instances such as “Public Employment” and “Private Employment”. This scenario is illustrated in Figure 42.

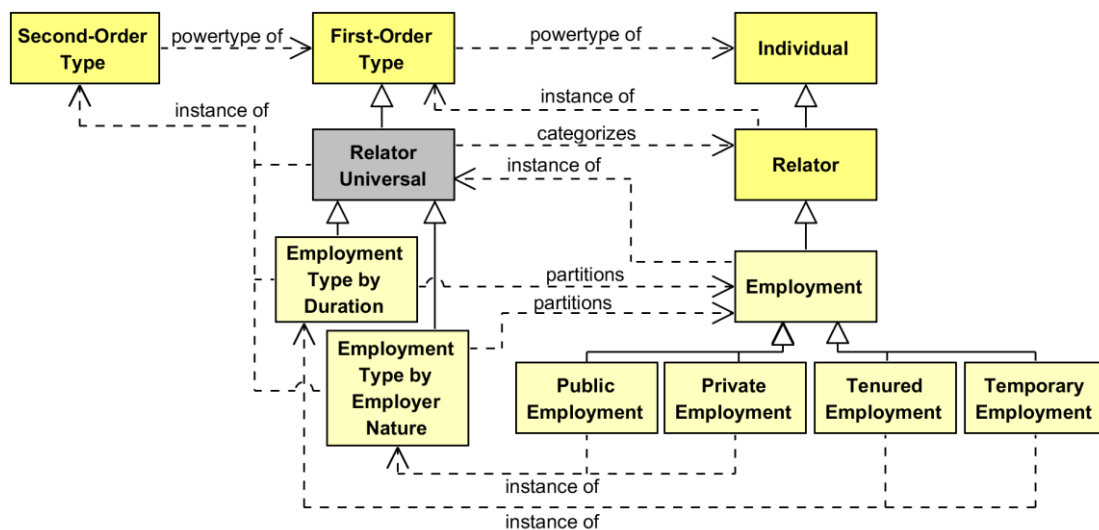


Figure 42 - Second-order types specializing “Relator Universal”.

In the previous sub-sections, we discussed that taxonomies of substantials may be created reflecting taxonomies of intrinsic moments. The same reasoning can be applied to taxonomies of relators: the criteria used by a second-order type specializing “Relator Universal” to categorize an instance of “Relator Universal” can be used to categorize an instance of “Role” mediated by the relator universal, distinguishing instances according to the type of relational properties they bear. For example, consider the role “Employee” encompassing all persons having an employment. To capture that each employee has at least one employment, we relate

“Employment” to “Employee” through a “relates” association, specializing the association that relates relators to substantials. Since every employee has employments, we may define a second-order type “Employee Type” that categorizes “Employee” according to the type of employment, having instances such as “Temporary Employee” and “Tenured Employee”. Every instance of “Employee Type” would have as instances objects having employments of a specific “Employment Type by Duration”. Therefore, the attribute “mediated universal” of “Employment Type by Duration” is a *regularity attribute* that constrains the possible values for the attribute “related individuals” of the type “Employment”. This scenario is captured in the domain conceptual model of Figure 43 in two ways. First, the “mediates” association between “Relator Universal” and “Role” is specialized giving rise to a one-to-one association between “Employment Type by Duration” and “Employee Type”. Second, the “involves” association between “Employment” and “Employee” is specialized to link each instance of “Employment Type by Duration” to the corresponding instance of “Employee Type”.

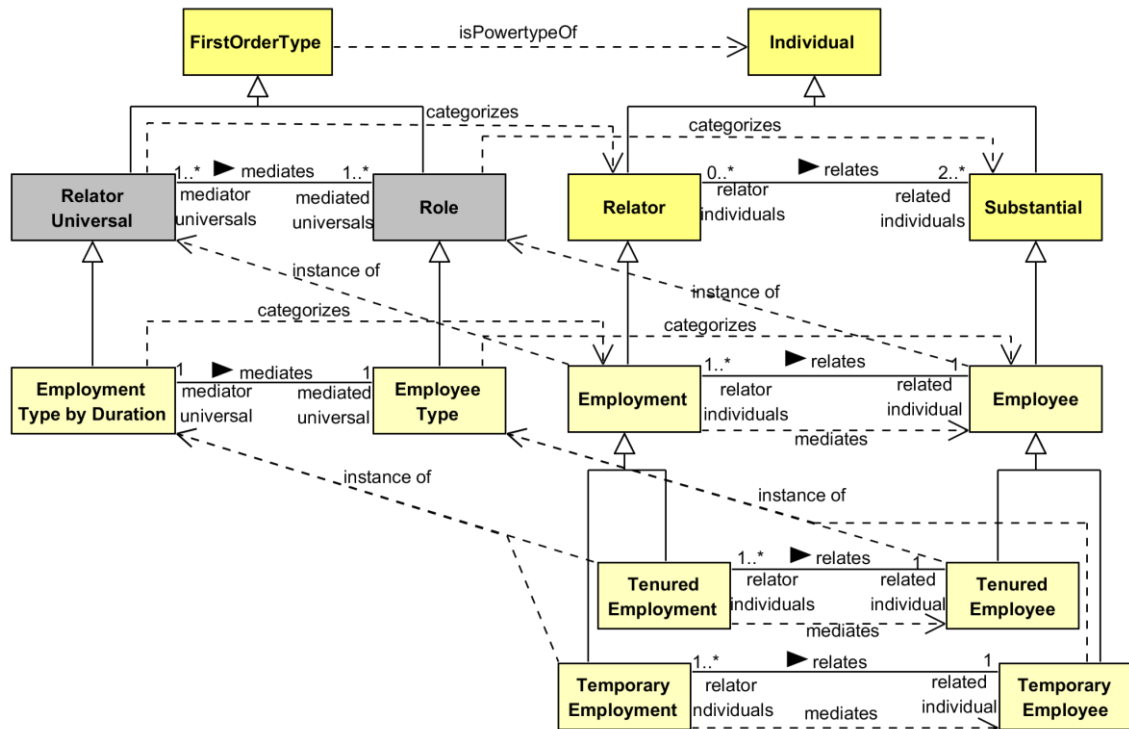


Figure 43 - Explicitly representing the relation between taxonomies of substantials and taxonomies of relators.

4.3.4 Summary

Table 5 summarizes the discussion concerning specializations of “Substantial Universal”. It presents the admissible ontological categories for base types of second-order types specializing each leaf category of “Substantial Universal”. The columns representing admissible base types are marked with a capital ‘X’, while inadmissible base types are marked with an hyphen ‘-’ (e.g., the table informs that it is admissible for a second-order type specializing “Subkind” to have an

instance of “Kind” as base type, but it is inadmissible for a specialization of “Subkind” to have an instance of “Subkind” as base type).

Table 5 - Allowed base types for specializations of Substantial Universal.							
	First-order Base Type						
Second-order Type	Instance of “Kind”	Instance of “Subkind”	Instance of “Role”	Instance of “Phase”	Instance of “Category”	Instance of “Role Mixin”	Instance of “Phase Mixin”
Specialization of “Kind”	-	-	-	-	X	-	-
Specialization of “Subkind”	X	-	-	-	X	-	-
Specialization of “Phase”	X	X	X	X	X	X	X
Specialization of “Role”	X	X	X	X	X	X	X
Specialization of “Category”	-	-	-	-	X	-	-
Specialization of “Phase Mixin”	-	-	-	-	X	X	X
Specialization of “Role Mixin”	-	-	-	-	X	X	X

As discussed in Section 4.3.3, the rules concerning the admissible ontological categories for base types of second-order types specializing “Moment Universal” are simpler, as sortality and rigidity are not addressed here for these types. In summary: (i) a specialization of “Quality Universal” must have an instance of “Quality Universal” as base type; (ii) a specialization of “Mode Universal” must have an instance of “Mode Universal” as base type; and (iii) a specialization of “Relator Universal” must have an instance of “Relator Universal” as base type.

4.4 Related Work

The strategy previously used in OntoUML (GUIZZARDI, 2005) was one in which the types represented in conceptual models could only instantiate the universals in UFO’s taxonomy of universals. These were represented by a fixed set of UML stereotypes, and thus a conceptual model could only have first-order types. In that approach, the axioms of the foundational ontology had to be incorporated into the syntax and semantics of the language profile (e.g., translated into corresponding syntactic rules as shown in (GUIZZARDI, 2005), or incorporated in a transformation of OntoUML into a logical formalism). This additional step is not necessary here as the structural relations and axioms of MLT-UFO are directly incorporated in the domain ontology. For example, concerning the combinations of the specialization patterns for second-order types, it is inadmissible for a domain second-order type that specializes “Kind” and “Subkind” to be subordinate to a domain second-order type which specializes “Role” or “Phase”. This is a consequence of the constraint in UFO that rules out the specialization of an anti-rigid universal by a rigid universal, together with the definition of subordination in MLT.

One recent initiative of applying ontological distinctions to discuss multi-level modeling issues can be found in (ERIKSSON; HENDERSON-SELLERS; ÅGERFALK, 2013). In (ERIKSSON; HENDERSON-SELLERS; ÅGERFALK, 2013), the authors propose a UFO-based approach that tries to avoid second-order types by employing a pattern based on the so-called *ontological square* comprising the categories of Substantial Universal/Substantial Individual and Moment Universal/Moment Individual, as well as their mutual relations. They provide an example in which “Horse” is considered a *substantial type*, a horse named “Prancer” is a *substantial object* (instance of “Horse”), “Breed” is a *moment type* and “Shetland Pony” is a *moment object* (instance of “Breed”). Since both *Prancer* and *Shetland Pony* are objects, there is no instance of relation between them. According to the authors, each instance of *Horse* is related to one instance of *Breed*, and one instance of *Breed* is related to many instances of *Horse*. Their assumption that the same *moment object* can be related to various *substantial objects* is a misinterpretation of a basic rule of the foundational ontology. In UFO, the relation between *moment objects (individuals)* and *substantial objects (individuals)* is that of *inherence*. In the ontology literature, in general, and in UFO, in particular, it is not possible for an intrinsic moment to inhere in two different individuals. What the authors seem to intend to represent is actually the relation between a *property* (in the ontological sense) “Shetland Pony” and a number of individuals in which this property is *exemplified* (also in the ontological sense (GUIZZARDI, 2005)). However, under this interpretation, “Shetland Pony” becomes a universal and “Breed” a second-order universal, defeating what they were trying to accomplish with their approach. Besides this ontological problem, the authors ignore the intuitive mechanisms of defining subtypes of a type according to properties of their instances and the benefits of such mechanisms. For example, using such approach, there is no support to represent properties that inhere only in instances of “Shetland Pony”.

Boro (PARTRIDGE, 2005) is a pioneer conceptual modeling approach in applying ontological concerns to address multiple-levels of classification. Our approach differs from Boro in its foundations: while BORO adopts a 4D ontology, our approach is founded on a 3D ontology; while BORO assumes an extensional criteria for types identity, we opt for an intensional criteria. The approach we propose here is, to the best of our knowledge, the first initiative on identifying patterns and constraints for higher-order types that is based on a 3D foundational ontology. Because of that, it enables us to proceed with a research program that directly incorporates these contributions to the practice of conceptual modeling via the OntoUML language and its associated tools.

4.5 Final Considerations

In this chapter, we have extended the Unified Foundational Ontology with the MLT multi-level theory in order to provide foundations for multi-level ontology-based conceptual modeling.

We have shown how the elements of MLT can be used to serve as the topmost layer of a hierarchy of conceptual models, from a foundational ontology to conceptual domain models. The concepts of the foundational ontology *instantiate* and *specialize* elements of MLT, respecting its axioms and using structural relations and patterns of MLT. In turn, the concepts of the conceptual domain model *instantiate* and *specialize* MLT-UFO, respecting MLT and UFO axioms and patterns. The result is an approach to define conceptual domain models that can represent types as well as types of types while adhering to the rules of a foundational ontology. UFO's original taxonomy of (first-order) universals is leveraged in order to provide patterns for types of types in the domain model. These patterns guide the modeler in the definition of higher-order types and their relations allowing the modeler to express modal properties of instances of higher-order types.

Another consequence of employing MLT concerns the engineering of UFO itself. UFO's taxonomies can now be explained in terms of instantiation of higher-order types. Further, as shown in Section 4.2, the relations of MLT (such as categorization) are used to explain how elements in the taxonomy of endurant universals relate to elements in the taxonomy of endurant individuals. Finally, the MLT notion of regularity attribute is applied to found a discussion concerning the UFO's notions of *characterization* and *mediation*. We discuss that the *characterization* relation holding between an intrinsic moment universal and a substantial Universal influences the intension of such universals by regulating the *inherence* relations that may hold between their instances. Analogously, the *mediation* relation holding between a relator universals and a role regulates the *involves* relations that may hold between their instances.

Given our focus on structural (as opposed to dynamic) aspects of conceptual modeling we do not discuss here the combination of MLT with the UFO portion that accounts for perdurants (events). Considering that the multi-level rules and patterns of MLT are independent of particular applications or domains, we believe that it is possible to extend the approach to encompass the UFO distinctions for events. This extension is an issue for future investigation.

While here we have focused the examples on the definition of domain models, the discussed approach forms the basis for further extension of UFO itself, as well as to include core ontologies in the hierarchy of models between the foundational ontology and domain models. In the future we will apply this approach to improve the formalization of UFO-based ontologies whose conceptualizations span multiple levels of classifications. To illustrate the applicability of this approach in the development of core ontologies spanning multiple levels of classification, we

have used it to construct a core organizational structure ontology. This subject is explored in next chapter.

Finally, we should highlight that our use of a notation inspired in UML to construct diagrams has been solely illustrative. It is not our intention to propose that notation as a modeling language. In Chapter 6. , we propose a UML profile that reflects MLT distinctions and rules to improve the expressivity, clarity, and parsimony of the UML support to model the power type pattern. Following a similar approach, a natural application for the MLT-UFO combination is to inform the design of a well-founded multi-level conceptual modeling language or to promote the redesign of a language, such as OntoUML, into a multi-level modeling language.

Chapter 5. Applying MLT-UFO in the Development of a Core Ontology

One of the key challenges in conceptualizing the organizational structure domain is that it can span multiple levels of classification, with *types* and *types of types* being part of the domain of enquiry. For instance, organizations may be staffed according to *role types* such as “Professor”, “Dean”, “Secretary”, “Project Leader”. They may also be structured according to different *types of organizational units*, such as “Division”, “Department”, “Section”, each of which may impose constraints on some required *role types* (e.g. each “Department” of the “Federal University of Espírito Santo” has a “Dean”). Thus, to describe the conceptualization underlying this domain, one needs to represent entities of different (but nonetheless related) classification levels, such as *individual persons* (“John”, “Mary”), *role types* (“Dean”, “Secretary”), *organizational units* (“Sales Division of Coca-Cola Inc.”, “Computer Science Department of the Federal University of Espírito Santo”) and *organizational unit types* (“Department”, “Division”). Furthermore, there is a large diversity of organizational structuring approaches in different enterprise settings, making the enumeration of a fixed set of role types and organizational unit types untenable. Considering the multi-level nature of the organizational structure domain and aiming to illustrate the applicability of our multi-level ontology-based conceptual modeling approach in the development of core ontologies, we have applied that approach to construct a core organizational structure ontology, which is reported in this chapter.

The task undertaken here is also relevant from the perspective of enterprise modeling. This is because some of the existing approaches for the representation of the organizational structure domain, such as ArchiMate (THE OPEN GROUP, 2012) and the W3C Org Ontology (W3C, 2014), do not offer modeling constructs to represent types of organizations and types of organizational units, focusing only on offering constructs for users to capture instance-level notions. Some other approaches such as ARIS do cover types of roles and organizational units (including elements such as “Position Type”, “Organization Unit Type” and “Person Type”), but present several semantic ambiguities (SANTOS; ALMEIDA; GUIZZARDI, 2013), lacking a clear semantic foundation for the concepts in the organizational structure domain (a problem which affects many other approaches concerning role-related concepts as discussed in (ALMEIDA; GUIZZARDI; SANTOS JR., 2009)).

Considering that the main goal of enterprise models is to represent organizational reality faithfully and thus serve for the purposes of documentation, analysis and communication, Enterprise Architecture (EA) modeling languages could benefit from the use of well-founded conceptual models as theoretical basis. The semantic shortcomings of some enterprise modeling

approaches have motivated several efforts in the past decade into suitable conceptual foundations to inform the design or redesign of enterprise modeling approaches. For example, the Unified Foundational Ontology (UFO) (GUIZZARDI, 2005) has been used to identify issues in the modeling of roles in some enterprise modeling approaches (ALMEIDA; GUIZZARDI; SANTOS JR., 2009) and other related organizational structure elements of ARIS (SANTOS; ALMEIDA; GUIZZARDI, 2013). That has led to a number of proposals of improvements to these EA approaches. More recently, more specialized semantic foundations have been explored, employing not only a foundational ontology but also core organizational ontologies in the evaluation and redesign of modeling languages. For example, UFO-S (NARDI et al., 2015) has been used to evaluate the service constructs of ArchiMate (NARDI; FALBO; ALMEIDA, 2014) and the O3 Organizational Ontology (aligned with the foundational ontology) to inform the extension of the ArchiMate metamodel for organizational structure modeling (PEREIRA; ALMEIDA, 2014). This allows for a more detailed analysis of certain domain-specific language aspects, incorporating relevant notions from the specialized literature for a particular domain of enquiry.

Despite these advances in enterprise modeling, a semantic foundation for the organizational structure domain that is capable of addressing the multi-level modeling issues is still lacking. Therefore, besides meeting the objective of illustrating the application of our modeling approach, the core ontology we propose here aims at addressing this gap, being thus, a contribution to the enterprise modeling area.

This chapter is further structured as follows: Section 5.1 presents the core organizational structure ontology built with the MLT-UFO combination; Section 5.2 positions the proposed ontology with respect to other organizational structure ontologies and with some organizational structure modeling approaches; finally, Section 5.3 presents concluding remarks.

5.1 A Core Organization Ontology Founded on MLT-UFO

In order to cope with the large diversity in organizational structures and structuring approaches, and following the modeling approach proposed in Chapter 4. , our core organizational structure ontology defines foundational concepts of organizational domains and may be extended by enterprise-specific ontologies that instantiate and specialize these basic distinctions with the concepts that are required in a particular organizational setting.

For instance, the core ontology defines generic concepts such as “Organization Type”, “Unit Type” and “Business Roles” and refrains from enumerating specific unit types such as “Division” and “Department”, as well as specific business roles such as “Vendor”, “Manager” and so on. These variations will be accommodated at more specific ontologies that extend the core organizational ontology to provide the elements that are used to account for organizational

reality in particular enterprise contexts. An example of a hierarchy of models is illustrated in Figure 44. The core organizational structure ontology extends MLT-UFO and is extended by two domain-specific organizational ontologies: one focused on universities and another for manufacturing companies. Finally, the domain-specific ontologies are further refined to provide concepts required in the context of specific universities (e.g., the Federal University of Esp rito Santo, UFES) and specific manufacturing companies (such as the Ford Motor Company).

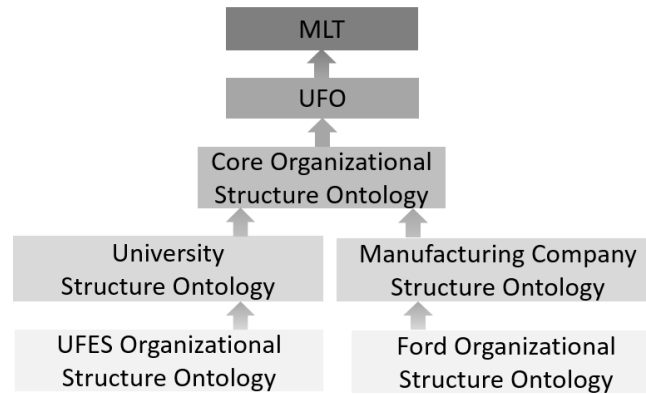


Figure 44 - Illustrating a hierarchy of organizational structure conceptual models founded on MLT-UFO combination.

Following the patterns proposed in Chapter 4. , every domain *first-order* type of our ontology (i) instantiates one of the leaf ontological categories of UFO taxonomy of universals (the types in dark grey in Figure 45) and, consequently, instantiates MLT “1stOT”; and (ii) simultaneously, specializes one of the leaf ontological categories of UFO taxonomy of individuals and, thus, specializes “Individual”. Further, every *second-order* type of our ontology specializes one of the leaf ontological categories of UFO taxonomy of universals and has an *MLT cross-level relation* with a *first-order* type.

Besides the UFO concepts discussed so far, we have also applied some concepts of the UFO social layer (GUIZZARDI; GUIZZARDI; FALBO, 2008) to support the construction of the core organizational structure ontology. The UFO taxonomy of individuals includes a social layer that specializes its core with distinctions to account for intentionality and social reality. It distinguishes between agentive and non-agentive objects. Agentive objects (instances of “Agent”) can perform actions and have intentional moments (intentions, desires and beliefs). Agents are differentiated in “Physical Agents” (e.g., a person) and “Social Agents” (e.g., an organization). The latter are created by speech acts and normative descriptions recognized by a numbers of agents. Figure 45 includes the required concepts of the UFO social layer (types in white) to the MLT-UFO combination model (augmenting thus Figure 25 of Chapter 4.).

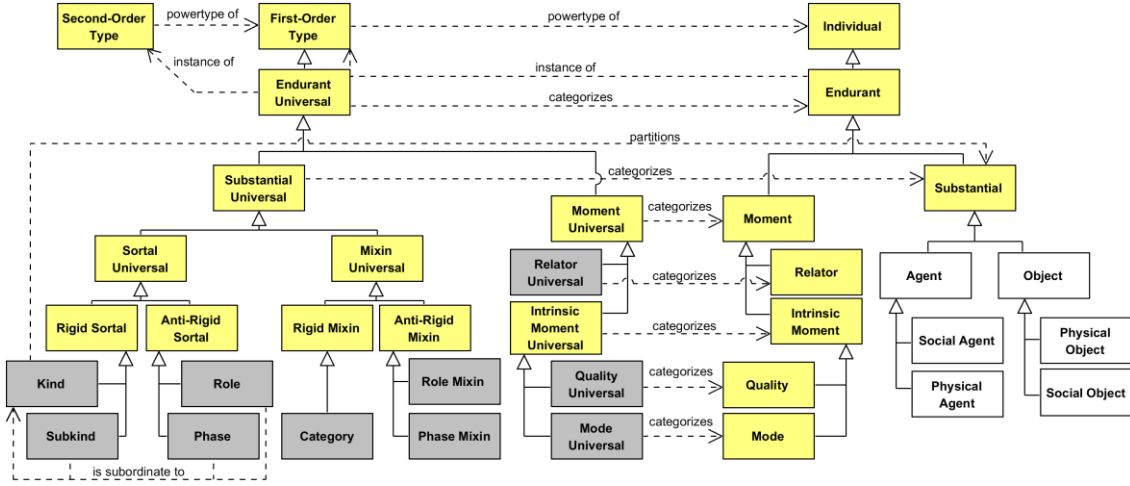


Figure 45 - Including concepts of the UFO social layer to the MLT-UFO combination model.

This ontology has been defined as a simplified version of O3 (PEREIRA; ALMEIDA, 2014; PEREIRA, 2015) focusing on the general concepts and highlighting the aspects that are specific to multi-level modeling. It adds to O3 *second-order types* to map the notions of “Unit Type”, “Organization Type” and “Assignment Type”. We discuss the ontology following two points of view: (i) organizational structure (in Section 5.1.1) and (ii) organizations roles and allocations (in Section 5.1.2).

5.1.1 Organizational Structure

Figure 46 illustrates the fragment of the ontology related with organizational structure concepts and its use as the basis of a hierarchy of models. Types that are part of the core ontology are shaded in dark gray. The types in light gray compose a fragment of a domain ontology about universities organizational structure. Finally, the types in white compose a model representing part of the organizational structure of the Federal University of Espírito Santo.

The topmost concept of this fragment is “Organization”, specializing the UFO notion of “Social Agent”. As defined in (ETZIONI, 1964), organizations are (artificial) social units built with the explicit intention of pursuing specific goals. Organizations include corporations, armies, hospitals and churches, but exclude tribes, ethnic groups, families and groups of friends. Members of an organization (which constitute the organization at a particular point in time) can be replaced or relocated to other functions while the organization persists in time.

We specialize “Organization” into “Formal Organization” and “Organizational Unit”. Formal organizations are formally recognized by the external environment. Their creation is determined by normative descriptions or speech acts that are recognized by the normative context in which they exist. Examples of formal organization include “Microsoft Inc.”, “The UK Government” and the “Federal University of Espírito Santo”. Organizational units are those organizations that are only recognized in the internal context of a formal organization and represent the working groups of a formal organization. Examples of organizational units include

the Marketing Department of Ford and the Sales Division of Coca-Cola. Considering that “Organization” is a general notion that does not provide a principle of identity to its instances we define it as an instance of “Category”. Further, assuming that “Formal Organization” and “Organizational Unit” provide a principle of identity to organizations and their units, both are considered instances of the UFO notion of “Kind”.

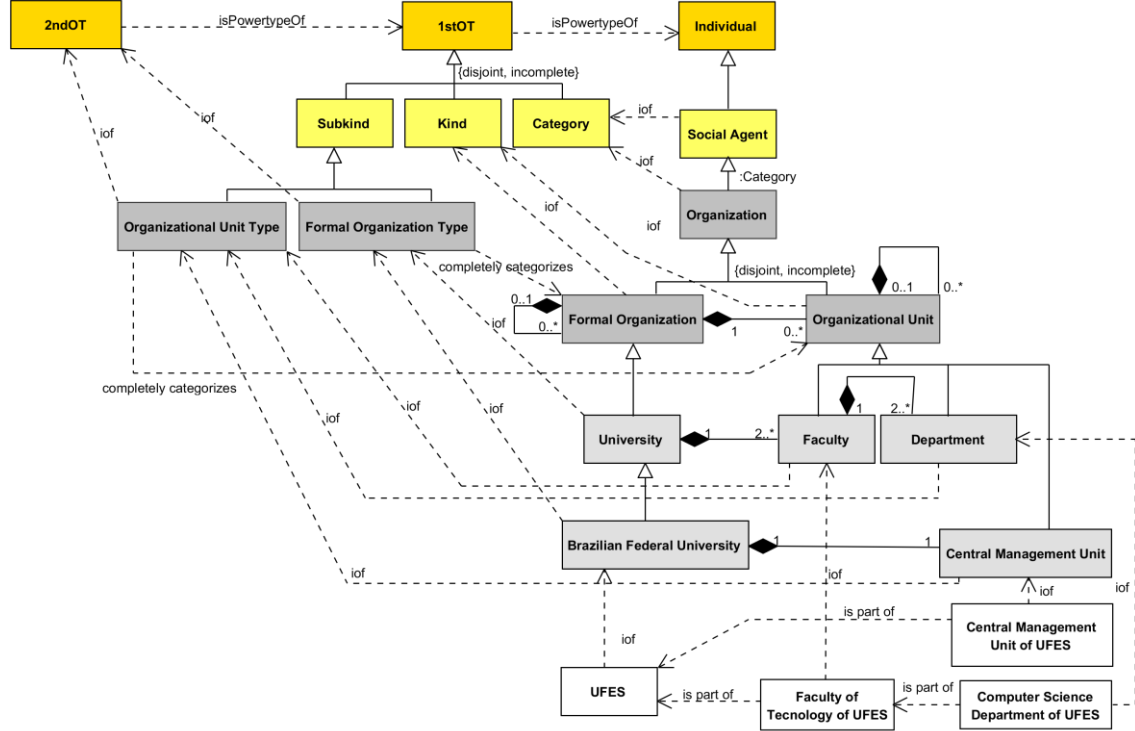


Figure 46 - Illustrating the definitions of the core organizational structure ontology and its use as the basis of a hierarchy of models.

We define a second-order type named “Formal Organization Type” whose instances are specializations of “Formal Organization”. Considering that every instance of “Formal Organization” is also instance of at least one instance of “Formal Organization Type” we state that “Formal Organization Type” completely categorizes (in MLT sense) “Formal Organization”. Examples of formal organization types specializing “Formal Organization” include “University”, “Corporation”, “Government”, and “Hospital”. Further, given that “Formal Organization” is an instance of “Kind”, and that the instances of “Formal Organization Type” are rigid types (i.e. the type of a formal organization do not change during the organization life cycle), it follows that instances of “Formal Organization Type” are subkinds in UFO sense. Therefore, following a pattern for second-order types discussed in Section 4.3.1, “Formal Organization Type” specializes the UFO notion of “Subkind”.

Analogously, we define “Organizational Unit Type” as a second-order type whose instances are rigid types that specialize “Organizational Unit” (such as “Department”, “Division” and “Project”). Thus, considering that (i) each instance of “Organizational Unit” is instance of, at least, one instance of “Organizational Unit Type”, and (ii) that “Organizational Unit Type” is a

(instance of) “Kind”, we conclude that (i) “Organizational Unit Type” completely categorizes “Organizational Unit”, and (ii) specializes “Subkind” (again, following a pattern discussed in Section 4.3.1).

Formal organizations may be composed of other formal organizations and of organizational units (see (ALMEIDA; GUIZZARDI, 2012) for a discussion on the whole-part relation of UFO applied at the organizational context). In this setting, a formal organization type may define the possible structures of its instances by constraining the types of organization (organizational units or other formal organizations) that may compose organizations of such type. Analogously, organizational units may be composed of other organizational units, and an organizational unit type may specify that an instance of it must be composed of other units of specific types. This is the basis for the definition of domain-specific types in an ontology that extends the core ontology. For instance, consider a domain ontology about university organizational structure. In this context, we can define “University” as an instance of “Formal Organization Type” (and thus, as a specialization of “Formal Organization”). Considering that universities are structured into faculties, which, in turn, are organized into departments we can define both “Faculty” and “Department” as instances of “Organizational Unit Type” and, thus, as specializations of “Organizational Unit”. Further, we may state that a “University” is composed of at least two faculties and each “Faculty” of a “University” is mandatorily composed of at least two “Departments”. Figure 46 illustrates it representing these domain-specific concepts in light gray.

Both formal organization types and organizational unit types may be specialized into domain-specific types. For example, supposing that all Brazilian federal universities must comply with the previous definition of university and, additionally, must have a “Central Management Unit”, we may capture it by creating: (i) an instance of “Formal Organization Type” called “Brazilian Federal University” as a specialization of “University”, and (ii) an instance of “Organizational Unit Type” called “Central Management Unit” having a mandatory composition relation with “Brazilian Federal University”. Considering these two new types are part of the university ontology, they are depicted in light gray in Figure 46.

This hierarchy of models can be further extended by creating models to express the structure of specific universities, such as the Federal University of Espírito Santo (UFES). For example, “UFES” can be represented as an instance of “Brazilian Federal University” being composed of some faculties (such as “Faculty of Technology of UFES” and the “Faculty of Law of UFES”) and a central management unit. The “Faculty of Technology of UFES” is composed of some departments such as the “Computer Science Department of UFES” and the “Electrical Engineering Department of UFES”. Some of the types that composes the structure organizational model of UFES are represented in Figure 46 with a white background.

The MLT notion of regularity attribute can be applied to capture the constraints imposed by the formal organization types to the possible composition of their instances. To illustrate it, in

Figure 47 “Formal Organization Type” is the source of two associations, each one capturing a regularity attribute. The attribute “compatible org types” of “Formal Organization Type” is a regularity attribute that regulates the attribute “component orgs” of “Formal Organization” constraining the compositions of formal organizations in other formal organizations according to their types. The regularity attribute “compatible unit types”, in its turn, regulates the attribute “component units”, constraining the composition of formal organizations in organizational units, again considering their types. Thus, considering the scenario in Figure 47, since “University” has “Research Center” and “Faculty” as “compatible unit types”, instances of “University” may be composed of research centers and faculties. For each value attributed to the “compatible unit type” regularity attribute one composition association is placed to capture the influence of the regularity attribute over the intension of the created type. In Figure 47 a composition association between “University” and “Faculty” captures that a university must have at least two faculties while the composition between “University” and “Research Center” captures that each university may have some research center (but it is not mandatory). Finally, since “University” has no value for the “compatible formal organization type” attribute, its instances may not be composed of other formal organizations.

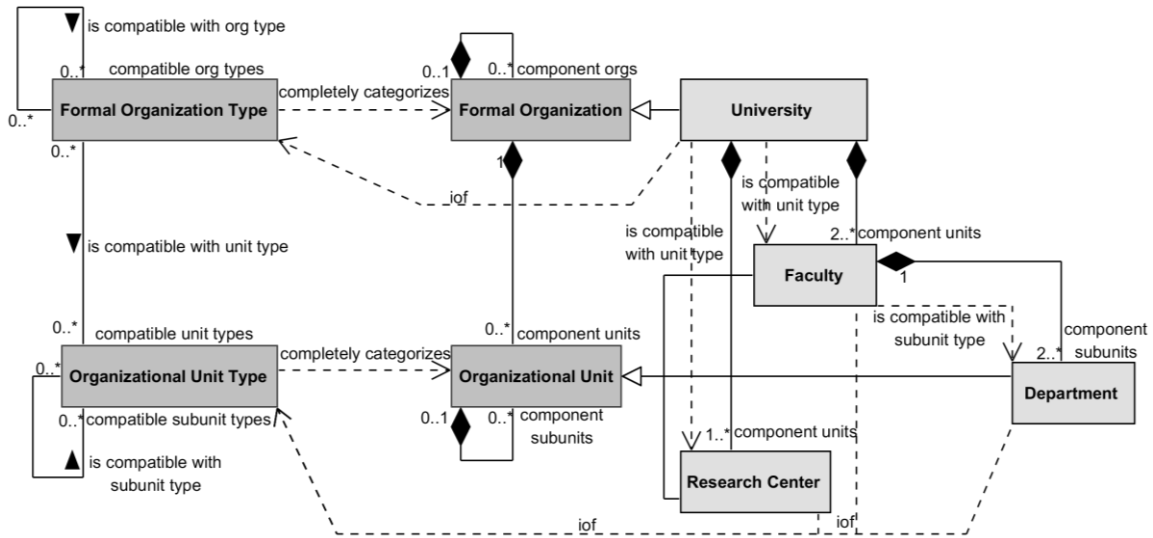


Figure 47 - Illustrating the use of regularity attributes to constrain formal organizations and organizational units compositions.

The same reasoning is applied to the composition of organizational units in other organizational units. The regularity attribute “compatible subunit types” of “Organizational Unit Type” regulates the attribute “component subunits” of “Organizational Unit”, such that organization units of a specific type t may only be composed of organizations units whose types are compatible with t . For example, in Figure 47, since “Faculty” has “Department” as its “compatible unit type”, instances of “Faculty” may be composed of instances of “Department”. The influence of the regularity attribute over the intension of “Faculty” is captured by a composition association which states that each faculty must be composed of at least two departments.

5.1.2 Organization Roles and Allocations

Figure 48 presents the concepts related with the agents that compose the organization and the types of roles they may play (types shaded in light grey). We are concerned in this fragment with the roles that persons play, first of all as a member of a formal organization, and then when they are given more specific places in organizational unit (organizational unit member).

We consider that only a specific kind of physical agents may compose organizations, namely the natural persons. To capture it we define that “Natural Person” specializes “Physical Agent” being instance of the UFO notion of “Kind”. To become a “Formal Organization Member”, a person (an instance of “Natural Person”) must be admitted by a formal organization, giving rise to an “Admission”. Analogously, the association between a person (playing the role of “Unit Member”) and an “Organizational Unit” (playing the role of “Allocation Unit”) is given by an “Assignment”. Thus, both “Admission” and “Assignment” specialize the UFO notion of “Relator” being instances of “Relator Universal”. “Employer” and “Allocation Unit” specialize, respectively, “Formal Organization” and “Organizational Unit”, being both instances of the UFO notion of “Role”, while “Formal Organization Member” and “Unit Member” are instances of “Role” and specialize “Natural Person”. Further, since “Natural Person” is a specialization of “Physical Agent” both, “Formal Organization Member” and “Unit Member” (indirectly) specialize the UFO notion of “Physical Agent”. This scenario is illustrated in Figure 48.

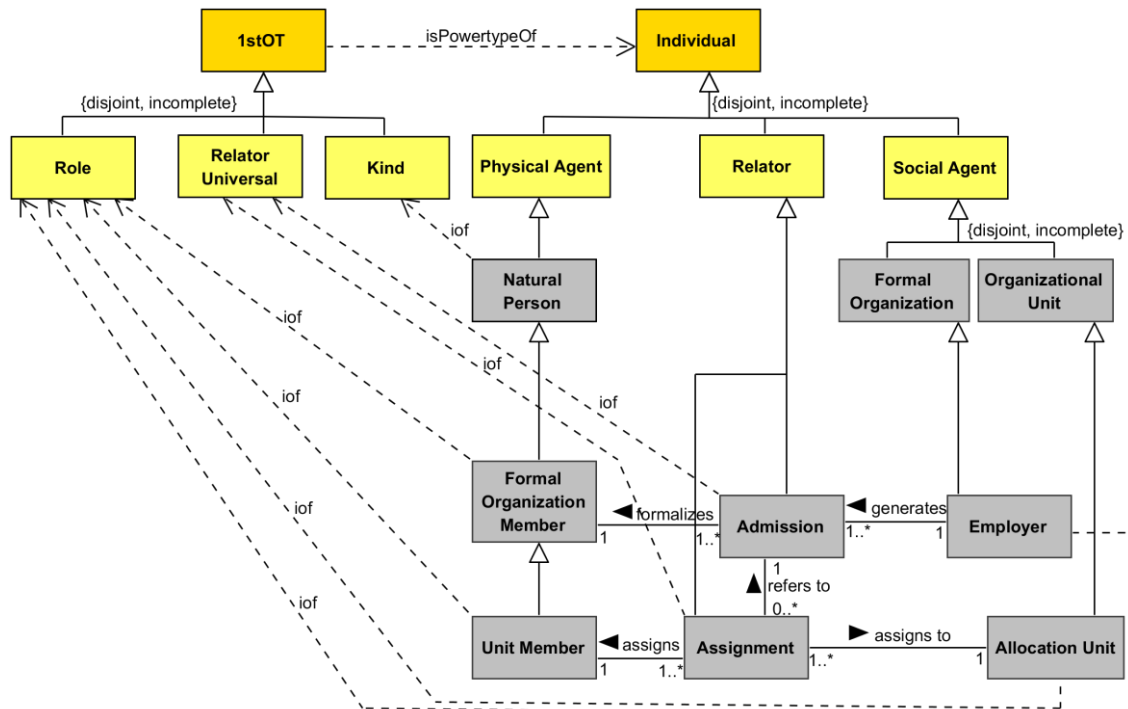


Figure 48 - Fragment of the core organizational structure ontology that copes with agents that compose the organization and types of roles they may play.

In order to play a particular role in an organizational unit, a person needs to be a formal organization member first. To capture this constraint, the assignments are tied with the admission

Specific formal organization member types define the set of roles (business roles) that a typified employee can occupy in the organization. Business roles define more specific capabilities, duties and prerogatives possibly in the scope of organizational units. Members who are instances of an (instance of) “Formal Organization Member Type” may play the business roles that *are covered by* such employee type. For example, we may state that, in the context of a university, both professors and management analysts may play the role of department dean while only professors may play the role of researchers, i.e., both “Professor” and “Management Analyst” *cover* “Department Dean” but only “Professor” *covers* “Researcher” (this scenario is illustrated in Figure 49).

Thus, we can define that if a role x *covers* a role y it means that: (i) instances of x are potential instances of y in the sense that individuals that plays the role x may also play the role y ; and (ii) for every instance i of y we have that i is instance of x or there is another role z such that z covers y and i is instance of z . Note that it is also possible to define cover relations between business roles (see Figure 49).

Each “Formal Organization Member Type” is admitted in the context of at least one “Formal Organization Type”, and for each “Formal Organization Type” there may be the necessity of certain formal organization member types. Thus, from the combination between a formal organization type and a formal organization member type arises the concept of “Admission Type”. The instances of “Admission Type” specializes “Admission” prescribing the employee types that are admissible in each formal organization type. Thus, “Admission Type” completely categorizes “Admission” (an instance of “Relator Universal”) and specializes the UFO notion of “Relator Universal” (following, thus, a pattern discussed in Section 4.3.3). The admission types allow the prescription of the possible allocations of an organization according to its type. For instance, we can define that a University must have, at least, twenty professors and two management analysts. Figure 50 illustrates this scenario (core ontology types are shaded in dark gray and the domain-specific ones are in light gray).

A variant of the pattern for the explicit representation of relations between taxonomies of substantials and taxonomies of relators (discussed in Section 4.3.3) is applied in Figure 50. In this setting the regularity attribute “possible org types” of “Admission Type” (represented in Figure 50 as an association end) constrains the attribute “organization” (again represented as an association end) of “Admission”, and, thus, the “is generated by” association between “Admission” and “Formal Organization” is redefined to link each instance of “Admission Type” to the corresponding instances of “Formal Organization Type”, accommodating specific multiplicities constraints. For example, since “University” is a “possible org type” of “University Professor Admission”, the “is generated by” association is redefined to link “University Professor Admission” to “University”, stating that every university must have, at least, twenty professors admissions. Similarly, the regularity attribute “possible member types” of “Admission Type”

constrains the attribute “member” of “Admission”, and, thus, the “formalizes” association between “Admission” and “Formal Organization Member” is redefined to link each instance of “Admission Type” to the corresponding instances of “Formal Organization Member Type”, accommodating specific multiplicities constraints. For example, since “Professor” is a “possible member type” of “University Professor Admission”, a specialization of the “formalizes” association is created to link “University Professor Admission” to “Professor”, stating that a professor may be admitted in many universities.

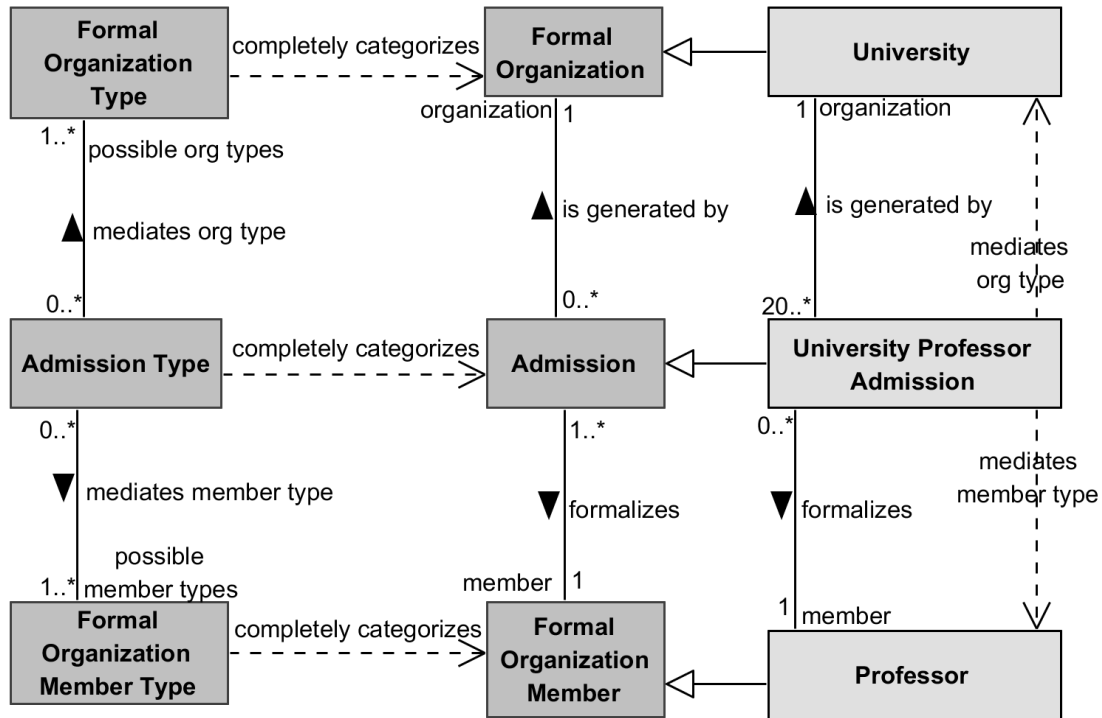


Figure 50 - Representing the relation between taxonomies of admissions, taxonomies of formal organizations and taxonomies of formal organization members.

Analogously, each “Business Role” is played in the context of at least one “Unit Type” and for each “Unit Type”, there may be the necessity of certain business roles. Thus, from the combination between a unit type and business role arises the concept of “Assignment Type”. The instances of “Assignment Type” specialize “Assignment”, prescribing the assignment types that are admissible in each unit type. Thus, “Assignment Type” completely categorizes “Assignment” (an instance of “Relator Universal”) and specializes the UFO notion of “Relator Universal” (following, thus, a pattern discussed in Section 4.3.3). The assignment types allow the prescription of the possible allocations of a unit according to its type. For instance, we can define that each department must have one member playing the role of department dean. Figure 51 illustrates this scenario (core ontology types are shaded in dark gray and the domain-specific ones are in light gray). Recall that, to play a business role in a unit, (i) the person must be first admitted in the formal organization to which such unit belongs, and (ii) the business role must be covered by the type instantiated by the person in the scope of its admission to the organization.

Again, a variant of the pattern for the explicit representation of relations between substantials and relators taxonomies is applied in Figure 51. In this setting the regularity attribute “possible unit types” of “Assignment Type” (represented in Figure 51 as an association end) constrains the attribute “unit” (again represented as an association end) of “Assignment”, and, thus, the “assigns to” association between “Assignment” and “Organizational Unit” is redefined to link each instance of “Assignment Type” to the corresponding instances of “Organizational Unit Type”, accommodating specific multiplicities constraints. For example, since “Department” is a “possible unit type” of “Department Dean Assignment”, the “assigns to” association is redefined to link “Department Dean Assignment” to “Department” stating that each department must have one department dean. Similarly, the regularity attribute “possible role types” of “Assignment Type” constrains the attribute “member” of “Assignment”, and, thus, the “assigns” association between “Assignment” and “Unit Member” is redefined to link each instance of “Assignment Type” to the corresponding instances of “Business Role”, accommodating specific multiplicities constraints. For example, since “Department Dean” is the “possible member type” of “Department Dean Assignment”, the “assigns” redefined to link “Department Dean Assignment” to “Department Dean” stating that each department dean is dean of exactly one department.

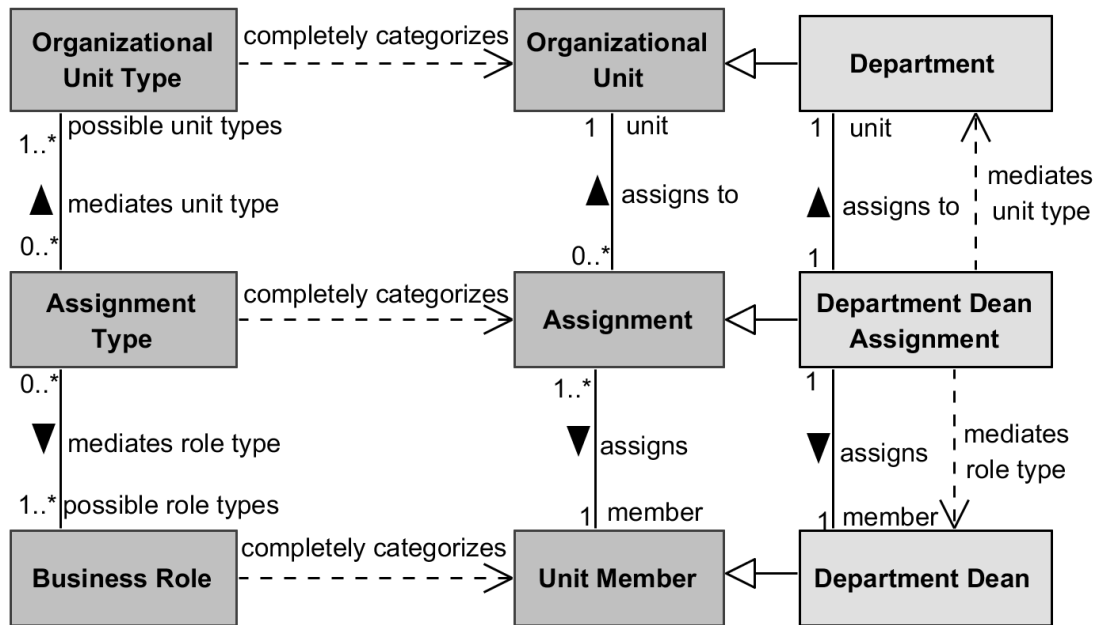


Figure 51 - Representing the relation between taxonomies of assignments, taxonomies of organizational units and taxonomies of business roles.

5.2 Related Work

As we have already argued, a suitable conceptualization for organizational structure spans multiple levels of classification, requiring a multi-level modeling approach. In this section, we

position our core ontology with respect to other organizational structure ontologies and modeling approaches.

5.2.1 Organizational Structure Ontologies

As discussed in (PEREIRA; ALMEIDA, 2014), the organizational structure domain has been the subject of a number of ontologies since the end of the 90s, including initiatives such as The AIAI Enterprise Ontology (EO) (USCHOLD; KING; MORALEE, 1998), the organization ontology for the TOVE enterprise model (FOX et al., 1998), the W3C Org Ontology (W3C, 2014), among others.

The AIAI Enterprise Ontology (EO) (USCHOLD; KING; MORALEE, 1998) is described in natural language and is based on formalized meta-ontology, with good coverage of concepts related to organization structure (PEREIRA; ALMEIDA, 2014). As discussed in (PEREIRA; ALMEIDA, 2014), EO includes a direct relationship between a “person” and an “organization unit” (“working for”), without the intermediary of roles or positions they play in the scope of an “organizational unit”. In case a person plays multiple roles, it is not possible to define which role is played in the context of each “organization unit”. Further, EO does not provide second-order notions such as “Organization Type”, “Unit Type” or “Business Role”. Instead, it defines fixed sets of types and business roles for specific domains (e.g., “Vendor”, “Customer”, “Reseller”). This makes it less general than the core ontology discussed here.

A similar remark can be made with respect to the organization ontology for the TOVE enterprise model (FOX et al., 1998), which chooses for a fixed hierarchical structure for the organizational with three levels: “organization”, “division” and “sub-division”. Choosing fixed roles and types restricts the applicability of these ontologies, making them unsuitable to any organizational contexts not structured according to these three levels. We take a different approach and aim here at a more general ontology while employing a hierarchical approach to cater for domain-specificity. Specific structures with instances of higher-order types can appear at a lower level of specificity, e.g., in an enterprise-specific ontology that extends the core organizational ontology for a particular organizational setting.

The W3C Org Ontology (W3C, 2014) is a W3C initiative aiming to provide support for publishing linked data of organizational information. Given its focus on the Semantic Web, the Org Ontology is an operational ontology and is defined in OWL. In terms of coverage of the domain, the Org Ontology does not provide concepts that would be comparable to our notions of “Formal Organization Type” and “Unit Type”, dealing only with instance-level notions for organizations and units. In the case of business roles for organizational members, it provides the concepts of “Role” and “Post”, although these are not considered types of employees. With respect to these concepts, first of all, we should note that there is not a clear conceptual difference between them as observed by Pereira (2015). A “Post” represents a position within an

organization and defines roles that are played by members holding such position (W3C, 2014). On the other hand, a “Role” denotes a role (in its general sense) that a member can play in the organization (the notion of organization here encompasses our notions of formal organization and organizational unit) (W3C, 2014). Despite the lack of a clearer definition on the conceptual differences between “Post” and “Role”, their functions in Org Ontology are similar to the ones we attribute to the notions of “Formal Organization Member Type” and “Business Role” in our ontology. Although, while we formally assume these concepts as “first-order” types, Org ontology faces them as individuals. Failing to represent their “type” nature, the semantics of the relation between members of the organization and the roles they play is not explicitly captured in the model. Differently, in our ontology, it is clear that roles are universals that are instantiated by members of the organization. This allows us to account for the fact that, while playing such roles, members of the organization bear a number of moments, such as commitments and claims, permissions, intentions, capabilities, skills, etc. Further, as types they can be included in specialization hierarchies which reveal relations between business roles. Finally, treating these concepts as types that specializes the UFO notion of “Role” allows us to formally capture their meta-properties (such as being anti-rigid and relationally dependent).

The concepts of the core organization structure ontology we propose here are based on the notions defined in the O3 ontology (PEREIRA; ALMEIDA, 2014; PEREIRA, 2015). Our ontology differs from O3 in its focus: we are concerned here solely with the most general concepts for organizational structure highlighting the aspects that are specific to multi-level modeling. For instance, we introduce the second-order concept of “Organizational Unit Type” and avoid fixing the distinctions used to specialize the notion of “Organizational Unit” such as the O3 distinctions between “structural” vs. “missionary units” and between “staff” vs. “line units”. Following our hierarchical modeling approach, these distinctions can be accounted for in a lower-level ontology. For example, applying the basic pattern of MLT the O3 notion of “Structural Unit” can be defined as an instance of “Organizational Unit Type” and, simultaneously, as a specialization of “Organizational Unit”. A topic for further investigation concerns the specification of other parts of O3 as an extension of this core ontology.

The Enterprise Ontology Pattern Language (E-OPL) (FALBO et al., 2014) is another example of ontology for organizations founded in UFO. It includes some notions for organizational structure, and aims to provide a basis for an enterprise pattern language whose fragments can be selected flexibly. E-OPL does not include concepts similar or corresponding to our notions of “Formal Organization Type” or “Organizational Unit Type”, but it provides a hierarchy of higher-order types having “Institutional Role” as its root concept. The E-OPL notion of “Institutional Role” encompasses our notions of “Business Role” and “Organization Member Type” and further distinguishes between positions and roles and discusses the existence of “informal roles”.

Since E-OPL is defined in the current version of OntoUML, which does not provide adequate support for higher-order types (as already discussed), a stereotype «hou» is used to mark the second-order types. Using our modeling approach E-OPL could benefit from the formal rules and concepts of MLT concerning multi-level modeling. Further, it would be possible to formally capture meta-properties of institutional roles by identifying that “Institutional Role” is a specialization of “Role” that *categorizes* “Human Resource”; this is currently not available to E-OPL due to the use of OntoUML. A revision of E-OPL at light of MLT-UFO combination is a topic for further investigation.

5.2.2 Organizational Structure Approaches

Many prominent enterprise architecture modeling approaches cater for the representation of organizational structures. For example, the Architecture of Integrated Information Systems (ARIS) (SCHEER, 1999) is an enterprise architecture framework that provides both a method for analysis and design of organizational aspects (including organizational structure) and a language for its representation. ARIS has a significant number of modeling constructs for organizational structure, including second-order notions such as organization unit type and unit member type. Nevertheless, as pointed out in (SANTOS; ALMEIDA; GUIZZARDI, 2013), ARIS lacks a sound semantic foundation and many of its modeling language constructs present problems such as redundancies and semantic overload (GUIZZARDI, 2005). As discussed in (SANTOS; ALMEIDA; GUIZZARDI, 2013) these problems also apply to the second-order notions. The ARIS metamodel seems to have evolved in an *ad hoc* manner and some of the second-order notions seem to have been introduced irregularly. For example, there is no support to represent the specialization of organizational unit types (while there is support to represent the specialization of “Person Types”); the notion of “Person Type” may be applied indiscriminately to persons and organization units alike; “positions” can be instances of “position types” but also “organizational unit types”. Please refer to (SANTOS; ALMEIDA; GUIZZARDI, 2013) for an in depth discussion on the semantics of ARIS organizational structure elements.

Another widely employed EA modeling language that includes organizational structure elements is ArchiMate (THE OPEN GROUP, 2012). A strength of the language is the broad coverage of a wide number of aspects of EA, and the possibility to describe relations between the various aspects. Nevertheless, the emphasis on providing an overview of relations seems to have led to a less sophisticated treatment of some aspects, and that includes the active structure domain (PEREIRA; ALMEIDA, 2014). In (PEREIRA; ALMEIDA, 2014) the authors conducted a semantic analysis of the fragment of the ArchiMate metamodel related with the representation of active structure revealing some problems caused by its lack of a sound semantics foundation. Concerning the ability to deal with second-order notions, such fragment of ArchiMate does not provide support to the specification of organizations types or organizational unit types. It provides

a construct called “Business Role” whose instances represent “the responsibility for performing specific behavior, to which an actor can be assigned”. We consider that the functions of “Business Role” in ArchiMate encompass the ones we attribute to the notions of “Formal Organization Member Type” and “Business Role” in our ontology. Although, differently from our approach, ArchiMate does not provide support to indicate that a business role is pertinent to a specific unit (PEREIRA; ALMEIDA, 2014). Further, while we assume business roles as types, ArchiMate faces them as individual. Therefore, all limitations induced by the decision of not representing the “type” nature of business roles that we have discussed in our analysis of the W3C Org Ontology are also present in ArchiMate.

Similarly to our approach, the Department of Defense Architecture Framework (DoDAF) (US DEPARTMENT OF DEFENSE, 2015) provides an account for the second-order notions of organizations types and role types (in its Operational Viewpoint OV-4) with a multi-level approach based on the IDEAS Foundational Ontology. Differently from our approach, they have chosen to consider the membership relations between organizations and their employees as “whole part” relations, which are specializations of the formal notion of “tuple” in IDEAS. A tuple is defined as an ordered pair of two things. In contrast, our approach considers the membership relations as specialization of the UFO notion of “relator”. In this view, the relationship between members and organizations (as well as the relationship between unit members and units) can: qualitatively change while maintaining their identity; be the subject of modal properties; be characterized by having both essential and accidental properties (GUARINO; GUIZZARDI, 2015). None of this is possible in the case in which relationships are reduced to tuples (see (GUARINO; GUIZZARDI, 2015) for a full discussion on the benefits of this foundational account to relationships).

5.3 Final Considerations

Aiming (i) to illustrate the application of the approach to support multi-level ontology-based conceptual modeling presented in Chapter 5. and (ii) to address the lack of a suitable foundation for organizational structure modeling, this chapter presents a core organizational structure ontology built with the combination of MLT and UFO. The foundational distinctions provided by UFO have allowed us to address some ontological issues concerning the core ontology concepts (e.g., anti-rigidity of role types, reification of relators), while the use of MLT has allowed us to address higher-order types and provides us with a sound basis to establish the relation between foundational ontology, core ontology and their specializations (e.g., an enterprise-specific ontology).

The result is a hierarchical modeling approach. The core organizational structure ontology may be extended by enterprise-specific ontologies that simultaneously instantiate and specialize

the core ontology distinctions with the concepts that are required in a particular organizational setting. Such a hierarchical approach is required to cope with the large diversity in organizational structures and structuring approaches.

The core ontology has been defined as a simplified version of O3 focusing on the general concepts and highlighting the aspects that are specific to multi-level modeling. It adds to O3 *second-order types* to map the notions of “Unit Type”, “Organization Type” and “Assignment Type”. It addresses second-order types that are not covered by ArchiMate, the W3C Org Ontology, the TOVE Enterprise Ontology and the AIAI Enterprise Ontology. These approaches propose a fixed set of organization and role types to their users, and thus cannot accommodate variations of enterprise settings.

Topics for further investigation concern the specification of other parts of O3 as an extension of this core ontology, and the specification of domain- or industry-specific models as an extension of O3 applying thus the proposed hierarchical modeling approach. Other topics of further investigation include revisiting the ontological analysis of ARIS conducted in (SANTOS; ALMEIDA; GUIZZARDI, 2013) and the extension of ArchiMate proposed in (PEREIRA, 2015) in light of MLT-UFO and the core ontology presented here.

Chapter 6. Using MLT to Revisit the Representation of Multi-Level Models in UML

Three fundamental quality attributes that must be reinforced in all conceptual modeling languages are *expressivity*, *clarity* and *parsimony* (HALPIN; MORGAN, 2008). The first refers to the ability of the language to capture all relevant aspects of the phenomena in reality it purports to represent; the second to how easy it is for the language users to unambiguously recognize which aspects of the underlying phenomena are represented; the third to how economic a language is in not forcing the modeler to represent more than it is necessary for a problem at hand. There is now a long tradition in conceptual modeling of using *Reference Theories* to evaluate and (re) design conceptual modeling languages according to these quality attributes (RECKER et al., 2011). Examples of fundamental conceptual modeling constructs that have been analyzed and re-designed following this strategy include types and taxonomic structures, part-whole relations, intrinsic and relational properties, roles, etc. (GUIZZARDI, 2005). Here, we use MLT as a reference theory to revisit the UML support to represent the power type pattern.

The power type pattern is extensively used in many important modeling initiatives. An example is the ISO/IEC 24744 standard (ISO/IEC, 2007). Moreover, this pattern can regularly be found in many catalogues of modeling best practices, in which it appears as an ingredient of other patterns (see, for instance, (FOWLER, 1997)). Finally, the relevance of this pattern has led to its adoption in the current version of the Unified Modeling Language (UML) (OMG, 2011), which allows modelers to specify a power type in the context of a “generalization set”.

UML is a *de facto* standard for conceptual modeling and information systems engineering. Moreover, it is the basis for ontology-driven conceptual modeling languages such as OntoUML (GUIZZARDI, 2005), which in the past years have gained increasing adoption in the conceptual modeling and ontology engineering communities (GUIZZARDI et al., 2015b). For this reason, we believe that providing precise and unambiguous semantics and advancing the UML support for modeling power types amounts to an important contribution for conceptual modeling, in general, and for ontology-driven conceptual modeling and ontology engineering, in particular. Aiming to achieve these goals, here we apply MLT to analyze the UML support for modeling the power type pattern.

By using MLT as a *reference theory*, we analyze and expose a number of limitations in the existing UML support for modeling the power type pattern. In particular, we demonstrate that this support: (i) lacks *expressivity*, for example for representing different definitions of power type that exist in the literature (e.g. CARDELLI, 1988; ODELL, 1994), each of which has relevant applications; (ii) that it lacks *clarity*, for example because it confounds constraints that apply to

power type instantiation with those that apply to corresponding generalization sets; (iii) that it lacks *parsimony*, for example because it forces the modeler to explicitly represent at least one instance of each power type. By employing the results of this analysis, we propose a UML profile for addressing the exposed limitations. We use the distinctions put forth by MLT to devise this profile, and we use the formal rules inherent to MLT to guide the development of the profile’s syntactic constraints.

The remainder of this chapter is structured as follows: Section 6.1 discusses UML’s current support for power types, revealing its limitations in light of MLT; Section 6.2 presents our proposal to extend a fragment of UML reflecting the rules of MLT and Section 6.3 presents concluding remarks.

6.1 UML’s Power Type Pattern Support in a Nutshell

The notion of *generalization set* is central to the UML’s power type pattern support. According to the UML 2.4.1 specification (OMG, 2011), each *generalization set* contains a particular set of *generalizations* that collectively describe the way a specific classifier (a class) is specialized into subclasses. To provide support to the power type pattern, UML includes in its “powertypes” package a meta-association that relates a classifier (the so-called “powertype”) to a generalization set that is composed by the generalizations that occur between the base classifier and the instances of the power type (OMG, 2011). The relation between the power type and the generalization set is represented in the UML notation by placing the name of the classifier next to the generalization set preceded by a colon. For example, in Figure 52 three specializations of “Tree” are defined, namely “Elm”, “Apricot” and “Saguaro”. The text “:Tree Species” denotes that the three subtypes enumerated in the generalization set are instances of “Tree Species” and that “Tree Species” is the “power type” of the generalization set. Note that the term “power type” as used in UML does not correspond to the notion of “power type” as proposed by Cardelli. (This issue is discussed in Section 6.2.) The “disjoint” constraint means that the subtypes have no instances in common while the “incomplete” constraint means that there are instances of “Tree” that are not instances of “Elm”, “Apricot” and “Saguaro”. The relation between the power type (e.g. “Tree Species”) and the base type (e.g. “Tree”) may be represented using a regular association with no special syntax and semantics.

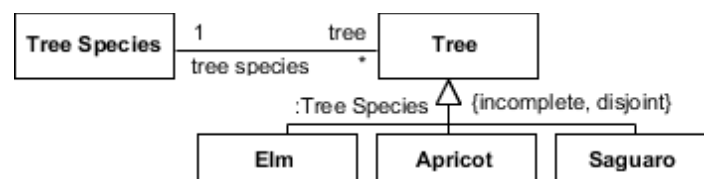


Figure 52 - The UML notation for the power type pattern (adapted from (OMG, 2011)).

A key observation is that for a classifier to be considered a “power type” in UML, it must be related to a generalization set. Thus, in UML, the power type pattern can only be applied when specializations of the base type are explicitly modeled (otherwise there would be no generalization set). We consider this undesirable as it rules out simple models such as one defining “Tree Species” as a “power type” of “Tree”, without forcing the modeler to define specific instances for “Tree Species”.

Furthermore, the only syntactic constraint defined in UML concerning power types is that “the classifier that maps to a generalization set may neither be a specific nor a general classifier in any of the generalization relationships defined for that generalization set” (OMG, 2011). While this rule prevents the power type from being involved in the generalization set defined to represent its own relation with the base type, this constraint is insufficient to rule out scenarios in which the power type is incorrectly related by generalization with types of any other levels.

6.2 Applying MLT to Revisit the Power type Support in UML

The application of MLT to revise the power type support in UML leads to the formulation of modeling recommendations to ensure: (i) a precise interpretation for the UML constructs used to express the power type pattern, (ii) a comprehensive support for the power type pattern including its variants in the literature, and; (iii) a number of syntactic rules to prevent the construction of inconsistent models.

First of all, we should observe that the UML specification is silent with respect to whether Cardelli’s notion of power type can be adopted. However, given that a *generalization set* can be said to define the *classification criteria* used to specialize the general type, the UML notion of power type seems to correspond to the *categorization relation* in MLT (not to the *is power type of* relation), in particular as other generalization sets may co-exist defining other classification criteria for the subtypes. This interpretation is corroborated by statements in the specification that explain that the subtypes of a base type are the instances of the “power type” (excluding the base type itself).

Further, we should observe that the semantics of the generalization sets constraints (e.g. “disjoint”, “incomplete”) is defined considering the types (classifiers) specified in the generalization set. UML does not provide any support to express constraints concerning the whole set of possible instances of the “power types”, and thus it is unable to differ between the variations of *categorization* defined in MLT. For example, the generalization set represented in Figure 52 is “incomplete” denoting that there are trees that are not instances of any represented species. Although, the generalization set does not capture the rule that every instance of “Tree” is an instance of one instance of “Tree Species”, i.e. that “Tree Species” *completely categorizes* “Tree”.

To address this expressiveness limitation, modelers usually use regular associations (with no special syntax and semantics) to represent the classification relations that hold between the base type and the higher-order type (as illustrated in Figure 52).

In the sequel, we propose a UML profile based on MLT that enriches the UML support to model the power type pattern.

6.2.1 The «instantiation» Stereotype

Our first recommendation is to mark the association between the base type and the higher order type with the «instantiation» stereotype, in order to distinguish it from other domain relations that do not have an instantiation semantics. An association stereotyped «instantiation» represents that instances of the target type are instantiated by instances of the source type and, thus, denote that there is a *categorization* relation between the involved types (regardless of possible generalization sets). For example, in Figure 53 an association stereotyped «instantiation» having “Tree” as source and “Tree Species” as target type is used to represent that instances of “Tree” are instances of instances of “Tree Species” and, conversely, that instances of “Tree Species” have instances of “Tree” as instances. Therefore, in MLT terms, it denotes that “Tree Species” *categorizes* “Tree”. Since this modeling structure does not rely on generalization sets, the modeler is not forced to represent instances of the power type, which would have been required in the case of plain UML.



Figure 53 - Illustrating the use of «instantiation».

The multiplicities of the “target” side of an «instantiation» association can be used to distinguish between the different variations of categorization. Whenever the lower bound multiplicity of the target association end is set to one, each instance of the base type is instance of, at least one instance of the power type. Thus, the higher order type completely categorizes the base type. In contrast, if the lower bound multiplicity of the target association end is set to zero, the inferred categorization relation is not a complete categorization. Analogously, if the upper bound multiplicity of the target association end is set to one, each instance of the base type is instance of, at most one instance of the higher order type. Thus, in this case, the higher order type disjointly categorizes the base type. In contrast, if the upper bound multiplicity of the target association end is set to many (*), the inferred categorization relation is not a disjoint categorization.

Table 6 summarizes the suggested interpretation in terms of MLT, considering different combinations of lower and upper bound multiplicities for the target association end. The combinations of multiplicities of the «instantiation» association with the values of the related

generalization set attributes create additional challenges for modelers using the power type pattern. These combinations are discussed in each of the following subsections, in which we expose some semantic issues.

Table 6 - The influence of the multiplicities in the semantics of «instantiation» associations.

UML Notation	Semantics in terms of MLT
	$\text{disjointlyCategorizes}(H, B) \wedge \text{completelyCategorizes}(H, B) \equiv \text{partitions}(H, B)$
	$\text{disjointlyCategorizes}(H, B) \wedge \neg \text{completelyCategorizes}(H, B)$
	$\text{completelyCategorizes}(H, B) \wedge \neg \text{disjointlyCategorizes}(H, B)$
	$\text{categorizes}(H, B) \wedge \neg \text{completelyCategorizes}(H, B) \wedge \neg \text{disjointlyCategorizes}(H, B)$

Lower and upper bound multiplicities set to one

When both the lower and the upper bound multiplicities of an «instantiation» association are set to one, we have that the power type simultaneously, *completely* and *disjointly categorizes* (i.e. *partitions*) the base type. For example, according to Figure 53 “Tree Species” *partitions* “Tree” (i.e. each instance of “Tree” is instance of exactly one instance of “Tree Species”). If it is used in tandem with a *complete* generalization set it means that all the instances of the higher-order type are enumerated in the diagram. For example, the model in Figure 54 represents that: (i) every instance of “Person” must be either an instance of “Man” or an instance of “Woman” and that (ii) “Man” and “Woman” are the only admissible instances of “Person Gender”.

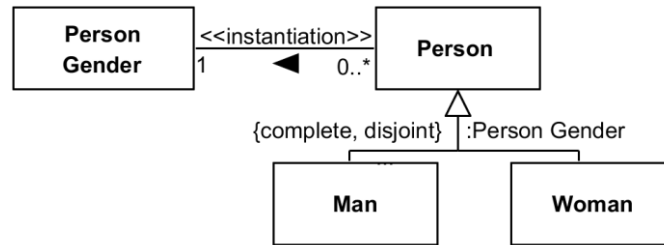


Figure 54 - Using «instantiation» to denote partitions relations.

At a first superficial inspection, one could consider that «instantiation» associations having the lower bound multiplicity (of the target association end) set to one could only be combined with a *complete* generalization set (as in Figure 54). However, this is not the case because the “*complete*” *constraint* represents whether all instances of the supertype are instances of one of the subtypes *in the generalization set*, and it is silent with respect to whether the higher-order type completely categorizes the base type. Thus, a combination of an «instantiation» association having both lower and upper multiplicities set to one in a pattern with an *incomplete* generalization set is admissible, and would mean that there are instances of the higher-order type that are not enumerated in the generalization set. For example, Figure 55 represents that: (i) each instance of “Tree” is instance of exactly one instance of “Tree Species” (represented by the «instantiation» association), (ii) “Elm”, “Apricot” and “Saguaro” are instances of “Tree Species”

(see the generalization set name), (iii) there are instances of “Tree” that are not instances of “Elm”, “Apricot” nor “Saguaro (represented by the *incomplete* constraint). Given the semantics of the «instantiation» stereotype in tandem with the semantics of the incomplete generalization set we can infer that (iv) there are instances of “Tree Species” that are not represented in the diagram.

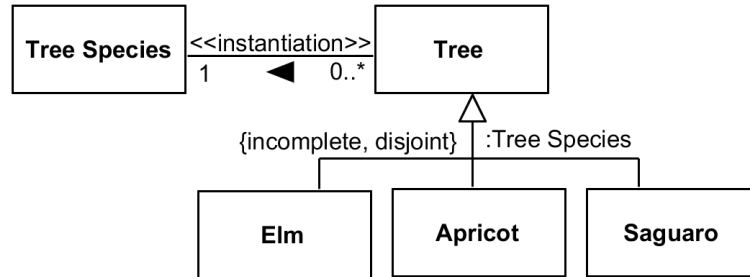


Figure 55 - Combining partitions relations with “incomplete” generalization sets.

Since the upper bound multiplicity of an «instantiation» association set to one means that each instance of the base type is instance of at most one instance of the higher-order type, a model combining it in a pattern with an overlapping generalization set is inconsistent, and thus, deemed syntactically invalid.

Lower bound multiplicity set to zero and upper bound set to one

An association stereotyped «instantiation» having the lower multiplicity set to zero and the upper bound multiplicity set to one denotes that the target type *disjointly categorizes* but *does not completely categorize* (in MLT sense) the source type. For example, suppose that an organization defines a type of roles called “Management Role” such that an employee cannot play more than one role of such type and it is not the case that all employees play some “Management Role”. This scenario is illustrated in Figure 56, showing “Organization President” and “Department Dean” as examples of instances of “Management Role”. The interpretation of the combination of an «instantiation» association having zero as the lower bound and one as the upper bound multiplicity with an incomplete generalization set is subtler than the cases we have discussed so far. In order to analyze this combination, we should first note that: (i) there are instances of “Employee” which are not instances of any instance of “Management Role” (as a consequence of the semantics of the «instantiation» association); and (ii) there are instances of “Employee” which are neither “Organization President” nor “Department Dean” (as a consequence of the semantics of *incomplete* generalization sets). The model is still silent with respect to whether all instances of “Management Role” are enumerated in this generalization set. It is possible that there are no other instances of “Management Role”, but an interpretation in which there are other management roles not mentioned in the model (e.g. “Division Head”) is also admissible.

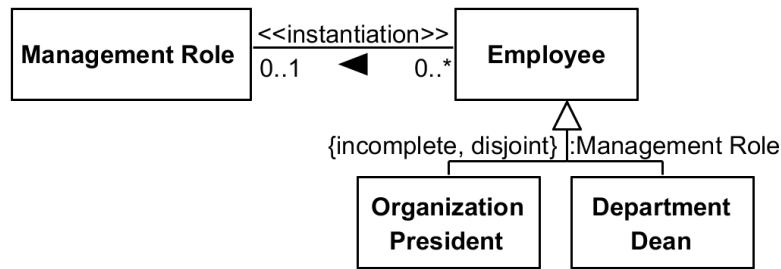


Figure 56 - Using «instantiation» to denote categorization relations that are disjoint but not complete.

Since an «instantiation» association having zero as the lower bound multiplicity implies that there are instances of the base type that are not instances of any instance of the higher-order type, a model combining it in a pattern with a complete generalization set is deemed syntactically invalid. Further, as previously discussed, the combination of an «instantiation» association with upper bound multiplicity set to one in a pattern with an overlapping generalization set is also deemed syntactically invalid.

Lower bound multiplicity set to one and upper bound set to many

An «instantiation» association having the lower multiplicity set to one and the upper bound multiplicity set to “many” (*) denotes that the target type *completely categorizes* but *does not disjointly categorize* (in MLT sense) the source type. For example, suppose that the rules of an organization define a type of roles called “Business Role” (having instances as “Programmer”, “DB Designer” and “Sw Designer”) such that every employee must play one or more roles of such type.

Associations stereotyped «instantiation» with “one” as lower bound multiplicity and “many” as upper bound multiplicity can be combined with any generalization sets despite they are *complete or incomplete, disjoint or overlapping*. However, the generalization sets constraints influence the semantics of the diagrams. For example, in Figure 57 the generalization set is *complete* and *disjoint* meaning each instance of “Employee” plays exactly one of the represented instances of “Business Role”. Therefore, since the multiplicities of the «instantiation» association between “Business Role” and “Employee” denotes that the instances of the former are overlapping, we conclude that there are non-represented instances of “Business Role” such that some of these instances are overlapping between them or some of them are overlapping with the represented ones. If the generalization set of Figure 57 were defined incomplete we could infer that there were non-represented instances of “Business Role” such that the whole set of instances of “Business Role” classifies all instances of “Employee” having some overlaps. Finally, considering the hypothesis in which the generalization set of Figure 57 were defined complete and overlapping we would have two possible interpretations: (i) all instances of “Business Role” are represented in the model or (ii) there are non-represented instances of “Business Role” but the represented ones already classify all instances of “Employee” having overlaps between them.

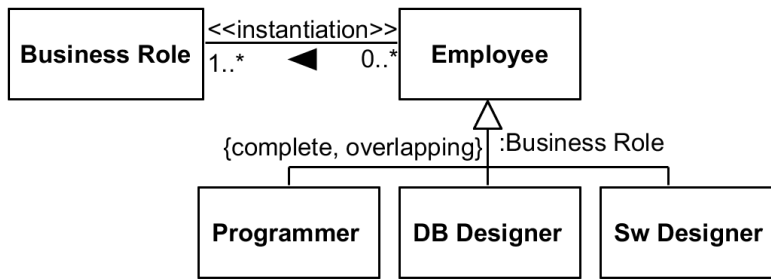


Figure 57 - Using «instantiation» to denote categorization relations that are complete but not disjoint.

Lower bound multiplicity set to zero and upper bound set to many

An «instantiation» association having the lower multiplicity set to zero and the upper bound multiplicity set to many (*) denotes that the target type *categorizes* (in MLT sense) the source type, however it is neither a *complete categorization* nor a *disjoint categorization*. Therefore, there may be instances of the base type that are instances of more than one instance of the higher-order type, and there may be instances of the base type that are not instances of any instance of the higher-order type. For example, Figure 58 consider a second-order type named “Social Role” whose instances represent roles that instances of “Person” may play in social relations, such as “Client”, “Employee” and “Husband”. Some instances of “Person” may play more than one “Social Role” and some other instances may play no social role.

Note that it is not possible to infer whether all instances of “Social Role” are represented or not in Figure 58: (i) they may all be enumerated, or (ii) there may be non-represented instances of “Social Role”. If the generalization set of Figure 58 were *disjoint*, the diagram would still be considered syntactically valid, and we could infer that there were non-represented instances of “Social Role” such that the whole set of instances of “Social Role” have some overlaps. Finally, if the generalization set of Figure 58 were *complete*, the diagram would be considered syntactically invalid since the whole set of instances of “Social Role” does not classify all instances of “Person”.

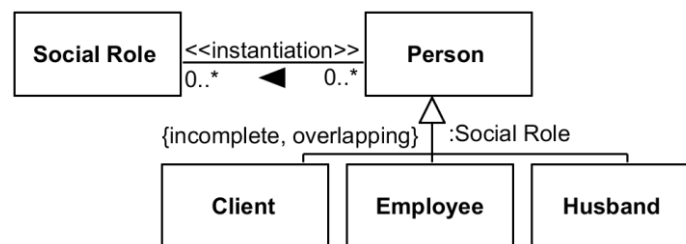


Figure 58 - Using «instantiation» to denote categorization relations that are not complete nor disjoint.

Table 7 summarizes the semantics of the combinations of the multiplicities of «instantiation» associations with the possible constraints of generalization sets, classifying each possible combination as: (i) *enumerated* if one can infer that all instances of the higher-order type are represented in the diagram; (ii) *non enumerated* if one can infer that there are instances of the higher-order type not represented in the diagram; (iii) *silent*: if it is not possible to infer whether

the instances of the higher-order type are enumerated or not; or (iv) *invalid* if the combination is syntactically invalid.

Table 7 - Analyzing the combination of «instantiation» with generalization set constraints.

Association Multiplicities		Generalization sets constraints			
Lower	Upper	{disjoint}		{overlapping}	
		{complete}	{incomplete}	{complete}	{incomplete}
1	1	<i>enumerated</i>	<i>non enumerated</i>	<i>invalid</i>	<i>invalid</i>
0	1	<i>invalid</i>	<i>silent</i>	<i>invalid</i>	<i>invalid</i>
1	*	<i>non enumerated</i>	<i>non enumerated</i>	<i>silent</i>	<i>non enumerated</i>
0	*	<i>invalid</i>	<i>non enumerated</i>	<i>invalid</i>	<i>silent</i>

6.2.2 An Additional Attribute for Generalization Sets

As we have previously discussed (and summarized in Table 7), three possible combinations of the multiplicities of «instantiation» associations with the possible constraints of generalization sets result in models that are syntactically valid but that are silent with respect to whether all instances of the higher-order type are enumerated in the generalization set. This is the case, for example, of the diagram presented in Figure 58, in which we combine the use of a «instantiation» association having the lower multiplicity set to zero and the upper bound multiplicity set to many (*) with an overlapping and incomplete generalization.

Our second recommendation is to address this kind of ambiguities by defining an additional attribute to generalization sets named *isEnumerated*. If *isEnumerated* = *true*, all instances of the higher-order type related to such generalization set are enumerated in the set. In contrast, if *isEnumerated* = *false*, there is at least one instance of the higher-order type related to the generalization set that is not represented in the set. This attribute is optional and only applicable to generalization sets that are associated with a power type. Note that this attribute is optional even in the cases in which a power type is present. In the absence of this attribute, the regular UML semantics applies, i.e. if it is not used, no information about the completeness of representation of the higher-order type instances is assumed. This is intended to keep compatibility with models build with plain UML.

Figure 59 revisits the example in Figure 58, removing the ambiguity that was present, setting *isEnumerated* to false to represent that there are other instances of “Social Role” beyond the ones represented in the model. This is represented with the “non-enumerated” generalization set constraint.

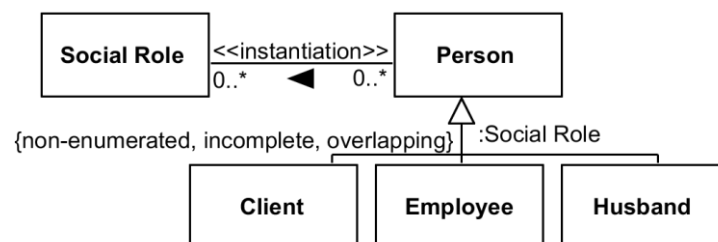


Figure 59 - Using the *isEnumerated* attribute to disambiguate a model.

The definition of *isEnumerated* attribute increases the language’s expressiveness and enables us to provide some additional syntactic constraints to guide the modeler. For example, in Figure 54 we could infer that “Man” and “Woman” are the only admissible instances of “Person Gender”, i.e., it is implicit that the generalization set *is enumerated*. Thus, the diagram in Figure 60 applies an inconsistent combination of generalization set constraints and dependency stereotype being syntactically invalid.

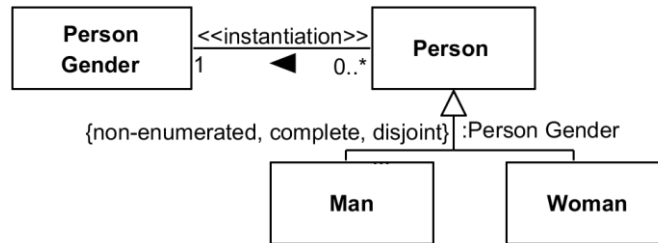


Figure 60 - An example of inconsistent use of the *isEnumerated* attribute making the model syntactically invalid.

Table 8 augments Table 7 summarizing the syntactical rules concerning the combination of the dependency stereotypes with the possible generalization sets constraints considering *the isCovering*, *isDisjoint* and *isEnumerated* attributes. This table replaces Table 7 if the modelers choose to use *isEnumerated*. The non-admissible combinations are identified with “✗” while “✓” identify the admissible ones.

Table 8 - Syntact constraints for combined dependency stereotypes and generalization sets constraints (using *isEnumerated*).

Association Multiplicities		Generalization sets constraints							
Lower	Upper	enumerated				non-enumerated			
		disjoint		overlapping		disjoint		overlapping	
		complete	incomplete	complete	incomplete	complete	incomplete	complete	incomplete
1	1	✓	✗	✗	✗	✗	✓	✗	✗
0	1	✗	✓	✗	✗	✗	✓	✗	✗
1	*	✗	✗	✓	✗	✓	✓	✓	✓
0	*	✗	✗	✗	✓	✗	✓	✗	✓

6.2.3 The «powerType» Stereotype

Our third recommendation is to use the «powerType» stereotype to represent Cardelli’s notion of power type (CARDELLI, 1988). If a class stereotyped «powerType» is the target of an «instantiation» association this means that this type *is power type of* the source type, i.e. the source type and all its specializations are instances of the target element. For example, in Figure 61, all types that (directly or indirectly) *specialize* “Person” are instances of “Person Type”.

According to Cardelli’s notion of power type the base type itself is instance of the higher-order type. Thus, in these cases, the lower bound multiplicity of the «instantiation» association must be set to one and the upper bound to many (*). Moreover, models in which the «powerType» stereotype is applied to types (classifiers) that are not target of any «instantiation» association are deemed syntactically invalid.

Another important syntactic constraint involving «powerType» is that, since a power type (in MLT) does not define a classification criteria to be applied to instances of the base type, there should be no generalization set anchored in types stereotyped «powerType» (i.e. *power type* relations do not give rise to generalization sets). For example, considering the scenario illustrated in Figure 61, a generalization set named “:Person Type” is not admissible. However, all subtypes of “Person”, despite the generalization sets in which they are involved, are instances of “Person Type”. Thus, all instances of “Person Gender” and “Social Role” are instances of “Person Type”.

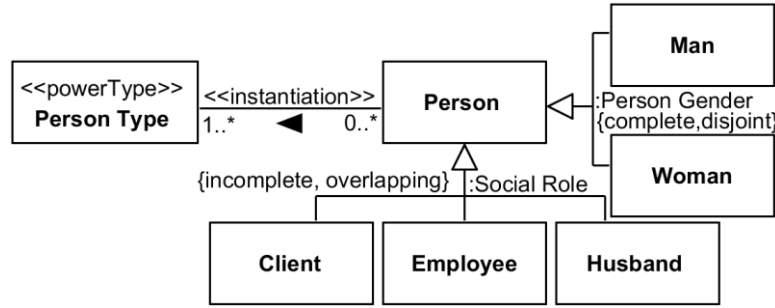


Figure 61 - Using «powerType» and «instantiation» to denote *is power type of* relations.

6.2.4 Syntactic Constraints Motivated by MLT Rules

An important aspect of the proposed interpretation is that it allows us to define syntactic constraints that reflect in the profile the axioms and theorems of the MLT formalization. These constraints having the purpose of reflecting the reference theory rules in the abstract syntax of a language are called *semantically-motivated syntactic constraints* (CARVALHO; ALMEIDA; GUIZZARDI, 2014). The main purpose of semantically-motivated syntactic constraints is to guarantee that the syntactically admissible models are sound according to the reference theory (CARVALHO; ALMEIDA; GUIZZARDI, 2014). Thus, the constraints presented in this section play a key role on guiding the modelers in producing sound models using the UML profile we propose.

For instance, given the definition of the *is power type of* relation of MLT, a type may not have more than one *power type* and a higher order *type* may be a *power type of* at most one other type. This suggests a clear syntactic constraint: a class stereotyped «powerType» can only be target of at most one «instantiation» association and a regular class can only be the source of at most one «instantiation» association having as target a class stereotyped «powerType». Further, the MLT theorem stating that if a type *t* *specializes* a type *t'* then the *power type of t specializes the power type of t'* may be used to check the syntax of power type hierarchies, and to generate the power types hierarchy corresponding to the base types hierarchy. For example, in Figure 62 the conjunction of the facts that: (i) “Employee” *specializes* “Person”, (ii) “Person Type” *is power type of* “Person” and (iii) “Employee Type” *is power type of* “Employee” implies that “Employee Type” must *specialize* “Person Type”.

Considering the MLT definitions of *power type*, *categorization* and *proper specialization* we conclude that if a type t' is *power type of* a type t and a type t'' *categorizes* the same base type t then all instances of t'' are also *instances of* t' and, thus, t'' *proper specializes* t' . This theorem also suggests a syntactic constraint. For example, in Figure 62 “Management Role” *categorizes* “Employee” and *specializes* “Employee Type”, whereas “Person Gender” *categorizes* “Person” and *specializes* “Person Type”. In this case, if the modeler fails to include any of the specializations between the higher-order types, it would be possible to infer them automatically.

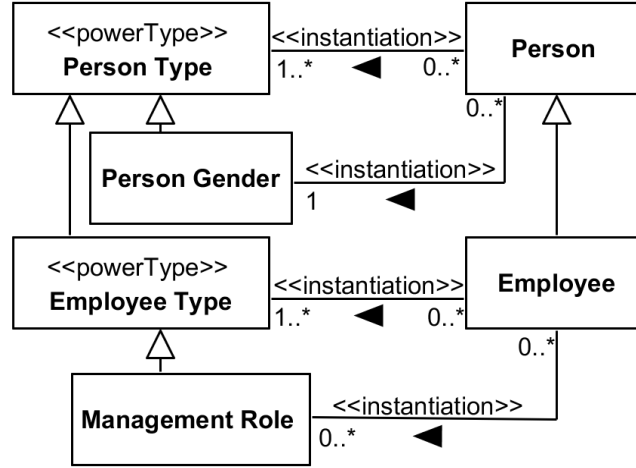


Figure 62 - Illustrating syntactic constraints concerning hierarchies of higher-order types.

Another MLT theorem states that if two types t' and t'' both *partition* the same type t then it is not possible for t' to specialize t'' . Again this suggests a clear syntactic constraint. For example, in Figure 63, “Person Age Phase” *partitions* “Person” according to their age having “Child” and “Adult” (and other non-represented types) as instances. “Person Gender”, in turn, *partitions* “Person” according to their gender having “Man” and “Woman” as instances. Thus, to be syntactically valid, the model may not include a specialization between “Person Age Phase” and “Person Gender”.

Recall that the MLT cross-level relations (*categorization* and *is power type of*) hold between a higher-order type and another type at one order lower. Thus, if two types are linked through an «instantiation» association, the type at the source association end is at an order lower than the one in the target (e.g. in Figure 63 “Person” is one order lower than “Person Age Phase”). Hence, cycles of associations stereotyped «instantiation» are not allowed. For example, suppose A is the target in an «instantiation» association in which B is the source, while B is the target in another «instantiation» association in which A is the source. This scenario is absurd since A must be at one order lower than B and, simultaneously, B must be at one order lower than A .

Finally, we consider that all higher-order types represented in diagrams must have cross-level relations with other types. Thus, we can determine the order of a type considering the «instantiation» associations in which they are involved as target. Types that are not targets of any «instantiation» association are first-order types (e.g. “Person”, “Man”, “Woman”, “Adult” and

“Child” in Figure 63). Types that are target in «instantiation» associations in which the sources are first-order types are second-order types (e.g. “Person Gender” and “Person Age Phase” in Figure 63), and so on. The MLT axiom that states that each domain type must be instance of exactly one MLT basic type (being thus at only one order) can be syntactically verified in our models. Further, the MLT theorem saying that specialization relations may only hold between two types at the same order may also be syntactically verified. For example, in Figure 63 there may not be specialization relations between a first-order type (i.e., “Person”, “Man”, “Woman”, “Adult” or “Child”) and a second-order type (i.e. “Person Gender” or “Person Age Phase”). Otherwise, the model would be considered syntactically invalid.

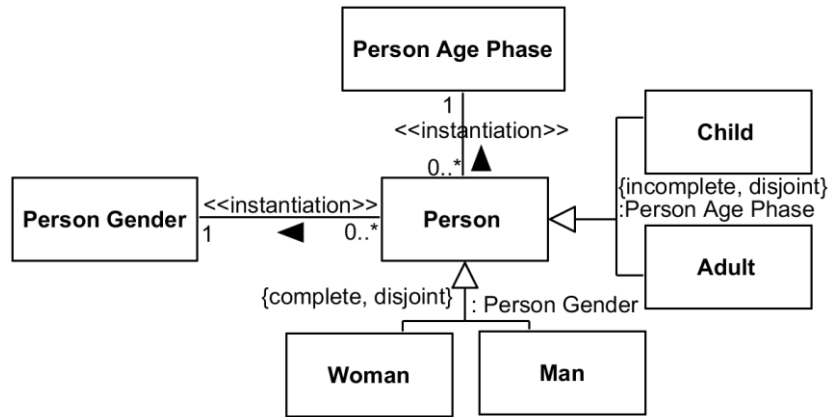


Figure 63 - Illustrating syntactic constraints concerning types orders.

A prototype plugin for the Visual Paradigm modeling tool that implements the proposed profile and performs syntactic verification of MLT rules is available at <http://github.com/nemo-ufes/MLT-VP-plugin>.

6.3 Final Considerations

In this chapter, we have used MLT as a reference theory to support the analysis and revision of the UML support to model the power type pattern, demonstrating that the current support lacks *expressivity*, *clarity*, and *parsimony*. By employing the results of this analysis, we propose a UML profile to address the exposed limitations and a tool set implementing the proposed profile. The proposal of the UML profile illustrates the application of MLT as a reference theory in the redesign of a modeling language while the implemented tool set enables the practical application of MLT by the conceptual modeling community.

The profile proposed here define constructs having specialized semantics to denote the cross-level relations between types defined in MLT, being, thus, able to distinguish properties of the relation between higher-order and base types that are required to represent multi-level classification schemes but that cannot be expressed in plain UML. The definition of these constructs with specialized semantics also allows us to systematically incorporate syntactic

constraints to reflect, in the profile, the formal rules of MLT. These syntactic constraints guide the modeler to produce sound multi-level models.

The analysis conducted here is focused on enhancing the UML support to the representation of the power type pattern. A natural extension is the incorporation of the MLT notion of *subordination* to the proposed UML profile. Our strategy to denote the MLT cross-level relation is based on the use of stereotypes to mark associations that have instantiation semantics. A similar approach can be adopted to denote *subordination* relations by defining a special stereotype as, for example, «specialization», to mark the associations between types involved in subordination relations. In this setting, an association stereotyped «specialization» would represent that instances of the target type are specialized by instances of the source type and, thus, it would denote a *subordination* relation between the involved types. This new extension must encompass the inclusion of new syntactic constraints to prevent inconsistencies in models applying the notion of subordination.

Other important issue for further investigation concerns the support to the representation of regularity attributes in UML. As previously discussed, regularity attributes are features of higher-order types that influences the intension of its instances, which can be reflected as constraints over the possible values for attributes of the base type. Future investigation may focus on providing mechanisms to capture the relations between attributes defined by types in one order and attributes of types in one order lower. Desirable features to represent this phenomena includes language constructs to provide visual representation of the relations between attributes as well as support to the formal expression of the constraints involving the related attributes.

In (GUIZZARDI, 2005), a fragment of UML has been evaluated at light of the Unified Foundational Ontology (UFO). Based on this analysis, a UML extension for the purposes of conceptual modeling (dubbed OntoUML) has been proposed. The ontology was used as a theory to inform the definition of a profile with syntactic constraints that reflect the UFO axioms. In this chapter, we have applied a similar approach to extend UML class diagrams using MLT as a theory to incorporate distinctions and constraints for multi-level modeling. We believe that a similar approach can be applied using MLT to analyze and enrich the semantics of other multi-level modeling approaches as, for example, the so-called deep modeling approaches (ATKINSON; KÜHNE, 2008; NEUMAYR; GRÜN; SCHREFFL, 2009). Furthermore, a natural extension of this work is to enrich OntoUML with the support for the power type pattern as discussed here.

Chapter 7. Final Considerations

In this work, we have explored the synergy between ontology-based conceptual modeling and multi-level modeling, with contributions for both areas. First, we developed MLT: a well-founded theory that captures the conceptualization underlying multi-level phenomena. Aiming to provide a precise conceptual foundation to multi-level modeling, we formalized MLT into an axiomatic theory and applied a lightweight formal method to verify the theory's consistency and the validity of its theorems.

Second, we combined MLT with UFO to leverage the benefits of ontology-based conceptual modeling to domains that include categories of categories, providing support to what we call *multi-level ontology-based conceptual modeling*. We have shown how the elements of MLT can be used: (i) to serve as the topmost layer of a hierarchy of multi-level conceptual models, from a foundational ontology to conceptual domain models (a contribution to ontology-based modeling area); and (ii) to explain the UFO's taxonomies in terms of higher-order types (contributing, thus, to the engineering of UFO).

Finally, following a systematic approach commonly applied in the ontology-based conceptual modeling community, we used MLT as a reference theory to analyze and improve the UML support for representing multi-level domains. We believe that similar approaches can be applied to analyze other multi-level modeling languages, bringing benefits to the multi-level modeling community. Section 7.1 summarizes the contributions of this thesis, and Section 7.2 discusses some research opportunities that arise from it.

7.1 Contributions

More specifically, the contributions of the work described here can be summarized as follows:

- **The design of an axiomatic theory for multi-level modeling (MLT).** The theory formally characterizes the nature of classification levels, and precisely defines structural relations that may occur between elements in the same classification level and relations that may occur between elements of different classification levels (addressing thereby the specific objective SO1). MLT can be considered a reference top-level ontology for types in multi-level conceptual modeling, providing basic concepts and patterns to articulate domains that require multiple levels of classification, as well as to inform the development or redesign of well-founded languages for multi-level conceptual modeling. In this sense, the theory fills a gap in the literature of multi-level modeling, contributing to the establishment of a more rigorous foundation for the discipline.

- **The proposal of a modeling approach founded on the combination of MLT and UFO.** The MLT-UFO combination serves as a foundation for conceptual models that can benefit from the ontological distinctions of UFO as well as MLT's basic concepts and patterns for multi-level modeling. Therefore, the proposed modeling approach extends the ontology-based conceptual modeling foundations to consider multiple classification issues, and contributes with multi-level modeling field by proposing a multi-level approach that addresses ontological distinctions (addressing the specific objective SO2). To the best of our knowledge, this is the first initiative that considers ontological distinctions such as *existential dependence*, *rigidity* and *sortality* on identifying patterns and constraints for higher-order types.
- **The design of a core organizational structure ontology founded on MLT-UFO combination.** The proposed core ontology contributes to the enterprise modeling area, defining a semantic foundation for the organizational structure domain that reflects the domain's multi-level nature and provides the required support to cope with the large diversity in organizational structures and structuring approaches. Further, it illustrates how the application of our modeling approach leads to a core ontology that can be extended with more specific conceptual models giving rise to a hierarchy of models in a spectrum of generality (addressing the specific objective SO3).
- **The proposal of a UML extension that enhances the language's support for the representation of multi-level phenomena.** This extension is the result of a principled approach to language revision, where MLT serves as a reference theory to analyze UML's support for the representation of the power type pattern (addressing the specific objective SO4). The proposed UML profile incorporates MLT rules, and is accompanied by tool support that automates the verification of rules. This initiative has two main contributions in the overall context of this work: (i) it demonstrates how MLT can be used as a reference theory to analyze the support modeling languages provide to represent multi-level phenomena, and (ii) it enables the practical application of MLT by the conceptual modeling community.

All the aforementioned research contributions were published (or are accepted to be published) in peer-review conferences or journals. Such publications are the following:

CARVALHO, Victorio Albani; ALMEIDA, João Paulo A. **Toward a Well-Founded Theory for Multi-Level Conceptual Modeling.** Software & Systems Modeling, 2016. DOI:10.1007/s10270-016-0538-9.

CARVALHO, Victorio Albani; ALMEIDA, João Paulo A.; FONSECA, Claudenir Moraes; GUIZZARDI, Giancarlo. **Extending the Foundations of Ontology-based Conceptual Modeling with a Multi-Level Theory.** In: Proceedings of the 35th International

Conference on Conceptual Modeling, ER 2015. DOI: 10.1007/978-3-319-25264-3_9.
(Best Paper Award)

CARVALHO, Victorio Albani; ALMEIDA, João Paulo A.; FONSECA, Claudenir Moraes;
GUIZZARDI, Giancarlo. **Multi-Level Ontology-based Conceptual Modeling**. Data
& Knowledge Engineering (accepted for publication, in review process).

CARVALHO, Victorio Albani; ALMEIDA, João Paulo A. **A Semantic Foundation for
Organizational Structures: A Multi-level Approach**. In: Proceedings of the 19th
International Enterprise Distributed Object Computing Conference, EDOC 2015. DOI:
10.1109/EDOC.2015.18.

CARVALHO, Victorio Albani; ALMEIDA, João Paulo A.; GUIZZARDI, Giancarlo
**Using a Well-Founded Multi-Level Theory to Support the Analysis and
Representation of the Powertype Pattern in Conceptual Modeling**. In: Proceedings
of the 28th International Conference on Advanced Information System Engineering,
CAiSE 2016. DOI: 10.1007/978-3-319-39696-5_19.

Other work developed in the scope of this doctoral research include:

CARVALHO, Victorio Albani; ALMEIDA, João Paulo A.; GUIZZARDI, Giancarlo.
**Using Reference Domain Ontologies to Define the Real-World Semantics of
Domain-Specific Languages**. In: Proceedings of the 26th International Conference on
Advanced Information System Engineering, CAiSE 2014. DOI: 10.1007/978-3-319-
07881-6_33.

GUIZZARDI, Giancarlo; ALMEIDA, João Paulo A.; GUARINO, Nicola; CARVALHO,
Victorio Albani. **Towards an Ontological Analysis of Powertypes**. In: Proceedings of
the International Workshop on Formal Ontologies for Artificial Intelligence, FOFAI
2015.

BRASILEIRO, Freddy; ALMEIDA, João Paulo A.; CARVALHO, Victorio Albani;
GUIZZARDI, Giancarlo. **Applying a Multi-Level Modeling Theory to Assess
Taxonomic Hierarchies in Wikidata**. In: Proceedings of the 5th International
Conference Companion on World Wide Web, Wiki Workshop 2016. DOI:
10.1145/2872518.2891117

BRASILEIRO, Freddy; ALMEIDA, João Paulo A.; CARVALHO, Victorio Albani;
GUIZZARDI, Giancarlo. **Expressive Multi-level Modeling for the Semantic Web**.
In: Proceedings of the 15th International Semantic Web Conference, ISWC 2016. DOI:
10.1007/978-3-319-46523-4_4

The last two publications deal with the implications of MLT to *Data on the Web* and *Semantic Web* technologies and have been the result of joint work whose bulk has been reported in a master’s thesis (BRASILEIRO, 2016).

In (BRASILEIRO et al., 2016a), we show that MLT rules can be used to detect a large number of problems in multi-level taxonomies in Wikidata (VRANDEČIĆ; KRÖTZSCH, 2014), a large structured knowledge base on the web. We have queried Wikidata for three anti-patterns that violate the strict stratification schema defined in MLT. Over 22,000 occurrences of these anti-patterns were found. The identified violations are usually associated with semantic problems in the models. For example, one of the hierarchies that violate the stratification rules states that “Computer Scientist” is, simultaneously, a specialization of “Profession” and an instance of “Profession”. Thus, according to this, every instance of “Computer Scientist” (e.g. “Tim Berners-Lee”) can be also considered an instance of “Profession”, which clearly violates our sense of what a Profession is. The high number of violations identified in Wikidata clearly indicates the need of an expressive support for multi-level conceptual modeling in Semantic Web. Further, in (BRASILEIRO et al., 2016b), we propose an OWL vocabulary that can be used as a basis for multi-level (operational) ontologies in Semantic Web. The proposed vocabulary is founded on MLT: the axioms and theorems of MLT are used to derive integrity constraints for multi-level vocabularies, offering guidance to prevent the construction of inconsistent vocabularies. Further, MLT rules are used to infer knowledge about the relations between types that are not explicitly stated. By defining a taxonomy of reusable relations between types founded in MLT, the vocabulary enables the expression of domain rules that would otherwise not be captured. A comparison with Semantic Web technologies that have some support for the representation of multi-level domains is available in (BRASILEIRO, 2016) and (BRASILEIRO et al., 2016b), showing the fruitfulness of MLT beyond the conceptual modeling community.

7.2 Future Perspectives

In this section we discuss some research opportunities that arise from this work:

- **Accounting for types with instances at different orders.** The current version of MLT regiments types into neatly stratified orders, and does not admit types that have instances at different orders. While this creates useful guidelines for domain modelers, it excludes from the domain of enquiry abstract notions such as a universal “Type” and, an even more abstract notion such as “Thing”. These notions may be relevant in abstract models, and in fact are widely employed in language engineering and in semantic web. A natural extension to this work is to provide an extended version of MLT that can account for these types. We believe that such an extended theory should distinguish between individuals and types, classifying the types as “stratified” or “non-stratified”. All the rules

discussed in the current version of MLT would apply to the “stratified” types. Rules concerning the behavior of “non-stratified” types as well as the possible relations involving “non-stratified” and “stratified” types are issues for further investigation.

- **Providing support to formally capture the intension of types.** The formalization of MLT presented here considers instantiation as a primitive notion, not appealing to the “internals” of intensions. This choice results in a less complex theory which is independent of modeling choices or ontological commitments concerning the nature of intensions of types. Nevertheless, because the “internals” of intensions are not addressed formally, the analysis concerning the influence of structural relations over the intensions of types could only be conducted informally. For example, although we discuss that in a proper specialization the intension of the specializing type adds some constraint(s) to the intension of the specialized type, it is not captured in MLT axiomatization. We believe that an extension of MLT providing support to capture the nature of the intension of types may allow us to identify and (formally) discuss patterns that arise from the influence of structural relations over the intensions of types.
- **Using MLT as a reference theory to inform analysis of multi-level modeling approaches.** As discussed in this thesis, we have applied MLT to revisit the UML support to represent the power type pattern. In the future, we expect to apply MLT to evaluate the notions underlying prominent multi-level modeling approaches. It may allow us to clarify the semantics of these multi-level modeling approaches, to position and harmonize different approaches and to propose enhancements in their mechanism to support multi-level modeling. Given its relevant influence in the multi-level modeling community, we have special interest in analyzing deep instantiation-based approaches. Further, we aim at evaluating the semantics of the modeling language defined by the Melanie framework (ATKINSON; GERBIG, 2012; KENNEL, 2012). We aim at clarifying its real-world semantics and at proposing the incorporation of some MLT concepts to the modeling language to enhance its expressivity.
- **Extending the MLT-UFO combination to encompass the UFO portion dealing with events as well as UFO’s social layer.** In this thesis, we have focused on the UFO portion dealing with endurants (objects). A future research agenda includes extending the proposed modeling approach to encompass the UFO portion dealing with events as well as UFO’s social layer. Such an extended approach could provide appropriate ontological support to the development of ontologies dealing with types of events, types of participations in an event, types of agents, types of intentions, types of situations, and so on. It could also be used to found the ontological analysis of modeling languages dealing with these concepts, such as goal-oriented modeling languages (QUARTEL et al., 2009; VAN LAMSWEERDE; LETIER, 2004; YU, 2009).

- **Improving the formalization of UFO-based ontologies.** The modeling approach proposed in this thesis can be applied to improve the formalization of UFO-based ontologies whose conceptualizations span multiple levels of classifications, as, for example, the O3 ontology (PEREIRA; ALMEIDA, 2014). Further, we believe that our modeling approach can be used to expand some UFO-based ontologies, such as UFO-S (NARDI et al., 2015), to include in their conceptualization second-order notions that are not discussed in their current versions. Finally, we believe the E-OPL pattern language (FALBO et al., 2014) may also benefit from our modeling approach, since it encompass higher-order concepts.
- **Designing a multi-level ontology-based modeling language.** A natural application for the MLT-UFO combination is to inform the design of an ontologically well-founded multi-level conceptual modeling language or to promote the redesign of a language, such as OntoUML, into a multi-level modeling language. Considering that OntoUML is proposed as a UML profile, this could be approached by extending OntoUML to incorporate the improvements to the UML support to the representation of the power type pattern we propose in this thesis. Further, the rules and patterns for the introduction of second-order types in ontology-based domain conceptual models we discuss in Chapter 4. may be incorporated by a multi-level version of OntoUML.

References

ALMEIDA, J. P. A.; GUIZZARDI, G. An ontological analysis of the notion of community in the RM-ODP enterprise language. **Computer Standards & Interfaces**, fev. 2012.

ALMEIDA, J. P. A.; GUIZZARDI, G.; SANTOS JR., P. S. Applying and Extending a Semantic Foundation for Role-Related Concepts in Enterprise Modeling. **Enterprise Information Systems**, v. 3, 2009.

ATKINSON, C. **Meta-modelling for distributed object environments**. Proceedings of the First International Enterprise Distributed Object Computing Workshop. Brisbane, Australia: IEEE Comput. Soc, 1997.

ATKINSON, C.; GERBIG, R. **Melanie: multi-level modeling and ontology engineering environment**. Proceedings of the 2nd International Master Class on Model-Driven Engineering Modeling Wizards - MW '12. New York, New York, USA: ACM Press, 2012.

ATKINSON, C.; GERBIG, R.; KÜHNE, T. **Comparing multi-level modeling approaches**. Proceedings of the 1st International Workshop on Multi-Level Modelling (MULTI 2014), co-located with MODELS 2014. 2014.

ATKINSON, C.; KUHNE, T. Processes and Products in a Multi-level Metamodeling Architecture. **Int. Journal of Software Engineering and Knowledge Engineering**, v. 11, n. 6, p. 761–784, 2001.

ATKINSON, C.; KUHNE, T. Model-driven development: a metamodeling foundation. **IEEE Software**, v. 20, n. 5, p. 36–41, set. 2003.

ATKINSON, C.; KÜHNE, T. **Meta-level Independent Modeling**. Proceedings of the International Workshop “Model Engineering” (in conjunction with ECOOP'2000). Cannes, France: 2000.

ATKINSON, C.; KÜHNE, T. **The Essence of Multilevel Metamodeling**. Proceedings of the 4th International Conference on the Unified Modeling Language. Toronto, Canada: 2001.

ATKINSON, C.; KÜHNE, T. Model-Driven Development: A Metamodeling Foundation. **IEEE Software**, v. 20, n. 5, p. 36–41, 2003.

ATKINSON, C.; KÜHNE, T. Reducing accidental complexity in domain models. **Software & Systems Modeling**, v. 7, n. 3, p. 345–359, 30 jun. 2008.

- BEALER, G. **Quality and Concept**. Oxford: Clarendon Press, 1982.
- BENEVIDES, A. B. et al. Validating Modal Aspects of OntoUML Conceptual Models Using Automatically Generated Visual World Structures. **Journal of Universal Computer Science**, v. 16, n. 20, p. 2904–2933, 2011.
- BENEVIDES, A. B.; GUIZZARDI, G. **A Model-Based Tool for Conceptual Modeling and Domain Ontology Engineering in OntoUML**. Proceedings of 11th ICEIS. Milan: 2009.
- BRAGA, B. F. B. et al. Transforming OntoUML into Alloy: towards conceptual model validation using a lightweight formal method. **Innovations in Systems and Software Engineering**, v. 6, n. 1–2, p. 55–63, 2 mar. 2010.
- BRASILEIRO, F. et al. **Applying a Multi-Level Modeling Theory to Assess Taxonomic Hierarchies in Wikidata**. Proceedings of the Wiki Workshop 2016 at 25th Int. Conference Companion on World Wide Web. 2016a.
- BRASILEIRO, F. **Representation of multi-level domains on the web**. Federal University of Espírito Santo, 2016.
- BRASILEIRO, F. et al. Expressive Multi-level Modeling for the Semantic Web. In: **Proceedings of the 15th International Semantic Web Conference**. p. 53–69. 2016b.
- CARDELLI, L. Structural subtyping and the notion of power type. **Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages - POPL '88**, p. 70–79, 1988.
- CARRARETTO, R. **Separating Ontological and Informational Concerns: A Model-driven Approach for Conceptual Modeling**. Federal University of Espírito Santo, 2012.
- CARVALHO, V. A. et al. **Extending the Foundations of Ontology-based Conceptual Modeling with a Multi-Level Theory**. Proceedings of the 34th Intl. Conf. on Conceptual Modeling (ER 2015). 2015
- CARVALHO, V. A.; ALMEIDA, J. P. A. **A Semantic Foundation for Organizational Structures: A Multi-Level Approach**. Proceedings of the Enterprise Computing Conference (EDOC 2015). 2015
- CARVALHO, V. A.; ALMEIDA, J. P. A.; GUIZZARDI, G. **Using Reference Domain Ontologies to Define the Real-World Semantics of Domain-Specific Languages**. Proceedings of the 26th International CAiSE Conference (CAiSE 2014). Springer, 2014.
- CHEN, P. P.-S. The entity-relationship model---toward a unified view of data. **ACM Transactions on Database Systems**, v. 1, n. 1, p. 9–36, 1 mar. 1976.

COQUAND, T. Type Theory. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. 2014. Disponível em: <http://plato.stanford.edu/archives/fall2014/entries/type-theory/>

DE LARA, J.; GUERRA, E. **Deep Meta-modelling with MetaDepth**. Proceedings of the 48th International Conference, TOOLS 2010. Málaga, Spain: 2010.

ERESHEFSKY, M. Species. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. 2010. Disponível em: <http://plato.stanford.edu/archives/spr2010/entries/species/>

ERIKSSON, O.; HENDERSON-SELLERS, B.; ÅGERFALK, P. J. Ontological and linguistic metamodelling revisited: A language use approach. **Information and Software Technology**, v. 55, n. 12, p. 2099–2124, dez. 2013.

ETZIONI, A. **Modern organizations**. Prentice-Hall, 1964.

FALBO, R. A. et al. **Organizing Ontology Design Patterns as Ontology Pattern Languages**. Proceedings of the 10th Extended Semantic Web Conference (ESWC 2013). Montpellier - France.: 2013.

FALBO, R. DE A. et al. **Towards an enterprise ontology pattern language**. Proceedings of the 29th Annual ACM Symposium on Applied Computing - SAC '14. New York, New York, USA: ACM Press,

FINE, K. Things and their Parts. **Midwest Studies in Philosophy**, v. 23, n. 1, p. 61–74, 1999.

FOWLER, M. **Analysis Patterns: Reusable Object Models**. 1. ed. [s.l.] Addison-Wesley Professional, 1997.

FOX, M. S. et al. An Organisation Ontology for Enterprise Modeling. In: PRIETULA, M.; CARLEY, K.; GASSER, L. (Eds.). . **Simulating Organizations: Computational Models of Institutions and Groups**. MIT Press, 1998. p. 131–152.

FRANK, U. Multilevel Modeling. **Business & Information Systems Engineering**, v. 6, n. 6, p. 319–337, 6 dez. 2014.

GÄRDENFORS, P. **Conceptual Spaces: the Geometry of Thought**. Cambridge, USA: MIT Press, 2000.

GONZALEZ-PEREZ, C.; HENDERSON-SELLERS, B. A powertype-based metamodelling framework. **Software & Systems Modeling**, v. 5, n. 1, p. 72–90, 2006.

GRUBER, T. R. A Translation Approach to Portable Ontologies. **Knowledge Acquisition**, v. 5, n. 2, p. 199–220, 1993.

GUARINO, N. The Ontological Level. In: CASATI, R.; SMITH, B.; WHITE, G. (Eds.). **. Philosophy and the Cognitive Science**. Vienna: 1994. p. 453–456.

GUARINO, N. **Formal Ontology and Information Systems**. Proceedings of International Conference on Formal Ontology in Information Systems (FOIS). IOS Press, 1998

GUARINO, N.; GUIZZARDI, G. “We Need to Discuss the Relationship”: Revisiting Relationships as Modeling Constructs. In: **Advanced Information Systems Engineering - 27th International Conference, CAiSE 2015**. p. 279–294.

GUARINO, N.; WELTY, C. Evaluating Ontological Decisions with OntoClean. **Communications of the ACM**, v. 45, n. 2, p. 61–65, 2002a.

GUARINO, N.; WELTY, C. Identity and Subsumption. **The Semantics of Relationships**. p. 111–126. Springer Netherlands, 2002.

GUIZZARDI, G. **Ontological Foundations for Structural Conceptual Models**. University of Twente, 2005.

GUIZZARDI, G. On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. In: VASILECAS, O.; EDLER, J.; CAPLINSKAS, A. (Eds.). **. Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV**. Amsterdã: IOS Press, 2007. v. 15p. 18–39.

GUIZZARDI, G. et al. **Towards an Ontological Analysis of Powertypes**. Proceedings of the International Workshop on Formal Ontologies for Artificial Intelligence (FOFAI 2015), 24th International Joint Conference on Artificial Intelligence (IJCAI 2015). 2015a

GUIZZARDI, G. et al. Towards ontological foundations for conceptual modeling: The unified foundational ontology (UFO) story. **Applied Ontology**, v. 10, n. 3–4, p. 259–271, 15 dez. 2015b.

GUIZZARDI, G.; GUIZZARDI, R. S. S.; FALBO, R. A. **Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The case of the ODE Software Process Ontology**. Proceedings of the XI Iberoamerican Workshop on Requirements Engineering and Software Environments (IDEAS 2008). 2008

GUIZZARDI, G.; SALES, T. P. Detection, Simulation and Elimination of Semantic Anti-patterns in Ontology-Driven Conceptual Models. In: **Conceptual Modeling: Proc. of the 33rd International Conference, ER 2014**. p. 363–376.

GUIZZARDI, G.; ZAMBORLINI, V. **An Ontologically-Founded Reification Approach for Representing Temporally Changing Information in OWL**. Proceedings of the 11th International Symposium on Logical Formalizations of Commonsense Reasoning (COMMONSENSE 2013). 2013

HALPIN, T.; MORGAN, T. **Information Modeling and Relational Databases**. 2. ed. Morgan Kaufmann, 2008.

HENDERSON-SELLERS, B. **On the Mathematics of Modelling, Metamodelling, Ontologies and Modelling Languages**. Springer, 2012.

ISO/IEC. **ISO/IEC 24744: Software Engineering –Metamodel for Development Methodologies**. Geneva: ISO, 2007.

JACKSON, D. **Software Abstractions: Logic, Language and Analysis**. MIT Press, 2006.

JARKE, M. et al. ConceptBase - A deductive object base for meta data management. **Journal of Intelligent Information Systems**, v. 4, n. 2, p. 167–192, mar. 1995.

JEUSFELD, M. A.; NEUMAYR, B. DeepTelos: Multi-level Modeling with Most General Instances. In: **35th International Conference, ER 2016**. Springer International Publishing, 2016. p. 198–211.

KENNEL, B. **A Unified Framework for Multi-Level Modeling**. University of Mannheim, 2012.

KÜHNE, T. **Contrasting Classification With Generalisation** (M. . Kirchberg, S. Link, Eds.)APCCM '09 Proc. Sixth Asia-Pacific Conference on Conceptual Modelling. Wellington, New Zealand: CRPIT, 2009

LARA, J. et al. Extending Deep Meta-Modelling for Practical Model-Driven Engineering. **The Computer Journal**, 2013.

LARA, J. DE; GUERRA, E.; CUADRADO, J. S. When and How to Use Multilevel Modelling. **ACM Transactions on Software Engineering and Methodology**, v. 24, n. 2, p. 1–46, 23 dez. 2014.

MAYR, E. **The Growth of Biological Thought: Diversity, Evolution, and Inheritance**. The Belknap Press, 1982.

MONTAGUE, R. **Formal Philosophy: Selected Papers of Richard Montague**. Yale University Press, 1974.

MYLOPOULOS, J. et al. Telos: Representing Knowledge About Information Systems. **ACM Trans. Inf. Syst**, v. 8, n. 4, p. 325–362, 1990.

MYLOPOULOS, J. Conceptual Modeling and Telos. In: LOUCOPOULOS, P.; ZICARI, R. (Eds.). . **Conceptual modeling, databases and CASE**. Wiley, 1992. p. 49–68.

NARDI, J. C. et al. A commitment-based reference ontology for services. **Information Systems**, v. 51, fev. 2015.

NARDI, J. C.; FALBO, R. DE A.; ALMEIDA, J. P. A. **An Ontological Analysis of Service Modeling at ArchiMate's Business Layer** 2014 IEEE 18th International Enterprise Distributed Object Computing Conference. IEEE, set. 2014.

NEUMAYR, B. et al. Dual Deep Instantiation and Its ConceptBase Implementation. Proceedings of the 26th International CAiSE Conference (CAiSE 2014). Springer, 2014. p. 503–517.

NEUMAYR, B.; GRÜN, K.; SCHREFL, M. **Multi-level domain modeling with m-objects and m-relationships** Proc. of the 6th Asia-Pacific Conference on Conceptual Modeling. Wellington, New Zealand: 2009

NEUMAYR, B.; SCHREFL, M.; THALHEIM, B. Modeling Techniques for Multi-level Abstraction. In: **The Evolution of Conceptual Modeling**. Springer, 2012.

ODELL, J. Power types. **Journal of Object-Oriented Programing**, v. 7, n. 2, p. 8–12, 1994.

OLIVÉ, A. **Conceptual Modeling of Information Systems**. Springer, 2007.

OMG. **UML Superstructure Specification – Version 2.4.1**, 2011.

PARTRIDGE, C. **Business Objects: Re-Engineering for Re-Use**. 2nd Edition. ed. London: BORO Centre, 2005.

PARTRIDGE, C. et al. Formalization of the classification pattern: survey of classification modeling in information systems engineering. **Software & Systems Modeling**, 16 abr. 2016.

PEREIRA, D.; ALMEIDA, J. P. A. **Representing Organizational Structures in an Enterprise Architecture Language**. Proceedings of the 6th Workshop on Formal Ontologies meet Industry (FOMI 2014). Rio de Janeiro: 2014

PEREIRA, D. C. **Representing Organizational Structures in Enterprise Architecture: An Ontology-Based Approach**. Federal University of Espírito Santo, 2015.

PIROTTE, A. et al. **Materialization: a powerful and ubiquitous abstraction pattern** (J. Bocca, M. Jarke, C. Zaniolo, Eds.) Procs. 20th Int. Conf. Very Large DataBases (VLDB '94). 1994

QUARTEL, D. et al. **A goal-oriented requirements modelling language** Proceedings of the 13th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2009, 1-4 September 2009, Auckland, New Zealand. Auckland, New Zealand: IEEE Computer Society, 2009

- RECKER, J. et al. Do Ontological Deficiencies in Modeling Grammars Matter? **MIS Quarterly**, v. 35, n. 1, p. 1–9, 2011.
- ROSSINI, A. et al. A formalisation of deep metamodeling. **Formal Aspects of Computing**, v. 26, n. 6, p. 1115–1152, nov. 2014.
- SANTOS, P. S.; ALMEIDA, J. P. A.; GUIZZARDI, G. An ontology-based analysis and semantics for organizational structure modeling in the ARIS method. **Information Systems**, jul. 2013.
- SCHEER, A.-W. **Business process modeling**. 3rd ed. Springer, 1999.
- SCHERP, A. et al. Designing core ontologies. **Applied Ontology**, v. 6, n. 3, p. 177–221, 2011.
- STEINBERG, D.; BUDINSKY, F. **EMF: Eclipse Modeling Framework**. 2nd Edition. Addison-Wesley Professional, 2008.
- SWOYER, C.; ORILIA, F. Properties. In: ZALTA, E. N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Fall 2014 ed. 2014.
- THE OPEN GROUP. **ArchiMate Version 2.1 Technical Standard**, 2012. Disponível em: <<http://pubs.opengroup.org/architecture/archimate2-doc/>>
- U.S. DEPARTMENT OF DEFENSE. **Data Modeling Guide (DMG) for an Enterprise Logical Data Model (ELDM)**, 2016. Disponível em: <http://www.omgwiki.org/architecture-ecosystem/lib/exe/fetch.php?media=dmg_for_enterprise_ldm_v2_3.pdf>
- US DEPARTMENT OF DEFENSE. **DoDAF Specification**, 2015. Disponível em: <http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_v2-02_web.pdf>
- USCHOLD, M.; KING, M.; MORALEE, S. **Enterprise Ontology specification**. Disponível em: <<http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.htm>>.
- VAN LAMSWEERDE, A.; LETIER, E. From Object Orientation to Goal Orientation: A Paradigm Shift for Requirements Engineering. In: **Radical Innovations of Software and Systems Engineering in the Future**. Springer Berlin Heidelberg, 2004. p. 325–340.
- VRANDEČIĆ, D.; KRÖTZSCH, M. Wikidata: A Free Collaborative Knowledgebase. **Communications of the ACM**, v. 57, p. 78–85, 2014.
- W3C. **OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (Second Edition)**. Disponível em: <<https://www.w3.org/TR/2012/REC-owl2-syntax-20121211>>.

W3C. **Org Ontology specification**. Disponível em: <<http://www.w3.org/TR/2014/REC-vocab-org-20140116/>>.

WAND, Y.; WEBER, R. On the ontological expressiveness of information systems analysis and design grammars. **Journal of Information Systems**, v. 3, p. 217–237, 1993.

YU, E. S. Social Modeling and i*. In: BORGIDA, ALEXANDER T.; CHAUDHRI, VINAY K.; GIORGINI, P. ET AL. (Ed.). . **Conceptual Modeling: Foundations and Applications**. Springer, 2009. p. 99–121.

Appendix A. Specification of MLT in Alloy – The basic theory

This appendix presents a specification of MLT in Alloy. This encompasses the axioms, definitions and theorems discussed in Sections 3.2, 3.3, and 3.4. Therefore, it does not address dynamic classification. An Alloy specification of MLT addressing dynamic classification is presented in Appendix B.

The whole specification is defined in one Alloy *module*. A *signature* “Entity” is defined to represent all entities in the domain of enquiry. Further, one *signature* is created to represent each basic type. Both axioms and definitions are represented as *facts* while the theorems are represented as *predicates*. Finally, *assertions* are created to group all the axioms to be checked.

```
module mlt

sig Entity{
  iof: set Entity,
  specializes: set Entity,
  properSpecializes: set Entity,
  isSubordinateTo: set Entity,
  powertypeOf: set Entity,
  categorizes: set Entity,
  compCategorizes: set Entity,
  disjCategorizes: set Entity,
  partitions: set Entity
}

//-----Basic types represented as singletons-----
-
//Representing the basic type "Individual"

one sig Individual extends Entity{}

//Representing the basic type "1stOT"
one sig FOT extends Entity{}

//Representing the basic type "2ndOT"
one sig SOT extends Entity{}

//Representing the basic type "3rdOT"
one sig TOT extends Entity{}
```



```

//-----End of Basic types representations-----
-

//-----Axioms and Definitions represented as facts-----
--

/*Axiom A1 - An entity is an instance of "Individual" iff does not possibly play
the role of type in instantiation relations.*/
fact individualDef{
    all x:Entity, i:Individual | ( i in x.iof iff no iof.x)
}

/*Axiom A2 - Two types are equal iff the sets of all their possible instances are
the same*/
fact typesEqualityDef{
    all x,y:Entity, i:Individual|
        i not in y.iof and i not in x.iof implies (iof.x = iof.y iff x=y)
}

/*Axiom A3 - An entity t is an instance first-order type ("FOT") iff all its
instances are Individuals (i.e., instances of "Individual")*/
fact firstOrderTypeDef{
    all t:Entity, f:FOT| f in t.iof iff (all x:Entity, i:Individual|
        some iof.t and (t in x.iof implies i in x.iof))
}

/*Axiom A4 - An entity t is an instance second-order type ("SOT") iff all its
instances are first-order types (i.e., instances of "FOT")*/
fact secondOrderTypeDef{
    all t:Entity, s:SOT| s in t.iof iff (all t':Entity, f:FOT|
        some iof.t and (t in t'.iof implies f in t'.iof))
}

/*Axiom A5 - An entity t is an instance third-order type ("TOT") iff all its
instances are second-order types (i.e., instances of "SOT")*/
fact thirdOrderTypeDef{
    all t:Entity, th:TOT| th in t.iof iff (all t':Entity, s:SOT|
        some iof.t and (t in t'.iof implies s in t'.iof))
}

/*Axiom A6 - Each entity in our domain of enquiry is necessarily an instance of
"Individual", "1stOT", "2ndOT" or "3rdOT" (except "3rdOT" whose type is outside
the scope of the formalization).*/
fact completenessAxiom{
    all s:SOT, f:FOT, i:Individual, t:TOT, x:Entity|

```

```

    f in x.iof or i in x.iof or s in x.iof or t in x.iof or x=t
}

/*Definition D1 - Specialization Definition: t1 specializes t2 iff all instances
of t1 are also instances of t2.*/
fact specializationDef{
    all t1,t2:Entity | t2 in t1.specializes iff all e:Entity, i:Individual |
        i not in t1.iof and i not in t2.iof and (t1 in e.iof implies t2 in e.iof)
}

/*Definition D2 - Proper Specialization Definition: t1 proper specializes t2 iff
t1 specializes t2 and is different from it.*/
fact properSpecializationDef{
    all t1,t2:Entity |
        t2 in t1.properSpecializes iff (t2 in t1.specializes and t1!=t2)
}

/*Definition D3 - Subordination Definition: t1 is subordinate to t2 iff every
instance of t1 specializes an instance of t2.*/
fact subordinationDef{
    all t1,t2:Entity | t2 in t1.isSubordinateTo iff (all i:Individual|
        i not in t1.iof and (all t3:Entity | (t1 in t3.iof implies (some t4:Entity|
            t2 in t4.iof and t4 in t3.properSpecializes))))
}

/*Definition D4 - Powertype Definition: iff a type t1 is power type of a type t2
all instances of t1 are specializations of t2 and all possible specializations of
t2 are instances of t1.*/
fact powertypeOfDef{
    all t1,t2:Entity | t2 in t1.powertypeOf iff (all t3:Entity, i:Individual |
        i not in t1.iof and (t1 in t3.iof iff t2 in t3.specializes))
}

/*Definition D5 - Categorization Definition: a type t1 categorizes a type t2 iff
all instances of t1 are properSpecializations of t2.*/
fact categorizationDef{
    all t1,t2:Entity | t2 in t1.categorizes iff (all t3:Entity , i:Individual |
        i not in t1.iof and (t1 in t3.iof implies t2 in t3.properSpecializes))
}

/*Definition D6 - Complete Categorization Definition: a type t1 completely
categorizes a type t2 iff t1 categorizes t2 and every instance of t2 is instance
of some instance of t1.*/
fact completeCategorizationDef{

```

```

all t1,t2:Entity | t2 in t1.compCategorizes iff
  (t2 in t1.categorizes and (all e:Entity | t2 in e.iof implies
    (some t3:Entity | t3 in e.iof and t1 in t3.iof)))
}

/*Definition D7 - Disjoint Categorization Definition: a type t1 disjointly
categorizes a type t2 iff t1 categorizes t2 and every instance of t2 is instance
of, at most, one instance of t1.*/
fact disjointCategorizationDef{
  all t1,t2:Entity | t2 in t1.disjCategorizes iff
    (t2 in t1.categorizes and (all e,t3,t4:Entity |
      (t1 in t3.iof and t1 in t4.iof and t3 in e.iof and t4 in e.iof) implies
        (t3=t4)))
}

/*Definition D8 - Partition Definition: a type t1 partitions a type t2 iff t1
completely categorizes t2 and t1 disjointly categorizes t2.*/
fact partitionsDef{
  all t1,t2:Entity | t2 in t1.partitions iff
    (t2 in t1.disjCategorizes and t2 in t1.compCategorizes)
}

//-----End of Axioms and Definitions representations-----
--

//Command to simulate the theory considering a scope of 20 elements
run {} for 20

//-----Theorems represented as predicates-----
--

//Theorems T1, T2 and T3
pred theoremsT1T2T3{
  //T1: "Individual" is an instance of "1stOT"
  all i:Individual, f:FOT | f in i.iof
  //T2: "1stOT" is an instance of "2ndOT"
  all f:FOT, s:SOT | s in f.iof
  //T3: "2ndOT" is an instance of "3rdOT"
  all t:TOT, s:SOT | t in s.iof
}

/*Theorem T4: "Individual", "1stOT", "2ndOT" and "3rdOT" have no instances in
common (i.e., their extensions are disjoint).*/
pred theoremT4{
  all t:TOT, s:SOT, f:FOT, i:Individual| no x:Entity|
    (f in x.iof and i in x.iof) or (s in x.iof and i in x.iof) or

```

```

    (t in x.iof and i in x.iof) or (f in x.iof and s in x.iof) or
    (f in x.iof and t in x.iof) or (t in x.iof and s in x.iof)
}

/*Theorems T5 and T6: The instance of relation is irreflexive, asymmetric and
anti-transitive */
pred theoremsT5T6{
    //Assymmetric
    all x,y:Entity | x in y.iof => y not in x.iof
    //Irreflexive
    all x:Entity | x not in x.iof
    //Anti-transitive
    all x,y,z:Entity | (y in x.iof and z in y.iof) => z not in x.iof
    //Acyclic
    all x:Entity | x not in x.^iof
}

/* Theorems T7, T8 and T9: Any instance of a higher-order type (any instance of
“1stOT”, “2ndOT”, and “3rdOT”) specializes the basic type at an immediately lower
order.*/
pred theoremsT7T8T9{
    //T7: Every instance of “1stOT” specializes “Individual”
    all t:Entity, i:Individual, f:FOT | f in t.iof iff i in t.specializes
    //T8: Every instance of “2ndOT” specializes “1stOT”
    all t:Entity, f:FOT, s:SOT | s in t.iof iff f in t.specializes
    //T9: Every instance of “3rdOT” specializes “2ndOT”
    all t:Entity, s:SOT, th:TOT | th in t.iof iff s in t.specializes
}

// Theorems T10, T11 and T2
pred theoremsT10T11T12{
    //T10: “1stOT” is powertype of “Individual”
    all i:Individual, f:FOT | i in f.powertypeOf
    //T11: “2ndOT” is powertype of “1stOT”
    all f:FOT, s:SOT | f in s.powertypeOf
    //T12: “3rdOT” is powertype of “2ndOT”
    all s:SOT, t:TOT | s in t.powertypeOf
}

//Theorem T13: each type has at most one power type
pred theoremT13{
    all t:Entity| lone powertypeOf.t
}

//Theorem T14: each type is power type of, at most, one other type

```

```

pred theoremT14{
  all t:Entity| lone t.powertypeOf
}

/*Theorem T15: if a type t2 specializes a type t1 then the power type of t2
specializes the power type of t1.*/
pred theoremT15{
  all t1,t2,t3,t4:Entity |
    (t1 in t2.specializes and t2 in t4.powertypeOf and t1 in t3.powertypeOf)
    implies t3 in t4.specializes
}

/*Theorem T17: If a type t2 is power type of a type t1 and a type t3 categorizes
the same base type t1 then all instances of t3 are also instances of the power
type t2 and, thus, t3 proper specializes t2.*/
pred theoremT17{
  all t1,t2,t3:Entity | (t1 in t2.powertypeOf and t1 in t3.categorizes)
    implies t2 in t3.properSpecializes
}

/*Theorem T18: if two types t1 and t2 both partitions the same type t3 then it is
not possible for t1 to specialize t2*/
pred theoremT18{
  all t1,t2,t3:Entity | (t3 in t1.partitions and t3 in t2.partitions)
    implies (t2 not in t1.properSpecializes)
}

//-----End of theorems representation-----
--
/*Assertion to verify all theorems*/
assert allTheorems{
  theoremsT1T2T3
    and theoremT4
    and theoremsT5T6
    and theoremsT7T8T9
    and theoremsT10T11T12
    and theoremT14
    and theoremT15
    and theoremT17
    and theoremT18
}

//Command to check the theorems considering a scope of 20 elements

```

```

/*To run the verification, uncomment the line bellow and comment the command used
to simulate the model as well as the command to run the verification of the other
theorems.*/

//check allTheorems for 20

/*----- Rules cited on the text but not formally stated as Theorems are tested
here as theorems-----
-----*/

/*Instantiation relations hold between two elements such that the last is one
order higher than the former.*/
pred iofCrossLevel{
  all x,y:Entity, i:Individual, f:FOT, s:SOT, t:TOT | y in x.iof implies
    ((i in x.iof and f in y.iof) or (f in x.iof and s in y.iof) or
      (s in x.iof and t in y.iof) or (t in x.iof))
}

/*Specialization is a partial order relation (i.e., a reflexive, transitive and
antisymmetric relation). */
pred specializationProperties{
  //Antissymmetric
  all x,y:Entity | (x in y.specializes and x!=y) => y not in x.specializes
  //Reflexive
  all x:Entity, i:Individual | i not in x.iof => x in x.specializes
  //Transitive
  all x,y,z:Entity |
    (y in x.specializes and z in y.specializes) => z in x.specializes
}

/*Specializations and proper Specializations may only hold between types of the
same order*/
pred specializationIntraLevel{
  all x,y:Entity, f:FOT, s:SOT, t:TOT | y in x.specializes implies
    ((f in x.iof and f in y.iof) or (s in x.iof and s in y.iof) or
      (t in x.iof and t in y.iof) or (t in x.specializes and t in y.specializes))
}

//Subordinations can only hold between higher-order types of equal order
pred subordinationIntraLevel{
  all x,y:Entity, s:SOT, t:TOT | y in x.isSubordinateTo implies
    ((s in x.iof and s in y.iof) or (t in x.iof and t in y.iof) or
      (t in x.specializes and t in y.specializes))
}

```

```

// PowertypeOf relations only occur between types of adjacent levels
pred powertypeOfCrossLevel{
  all x,y:Entity, f:FOT, s:SOT, t:TOT | x in y.powertypeOf implies
    ((f in x.iof and s in y.iof) or (s in x.iof and t in y.iof) or
    (t in x.iof and t in y.specializes))
}

//Categorization relations only occur between types of adjacent levels
pred categorizationCrossLevel{
  all x,y:Entity, f:FOT, s:SOT, t:TOT | x in y.categorizes implies
    ((f in x.iof and s in y.iof) or (s in x.iof and t in y.iof) or
    (t in x.iof and t in y.specializes))
}

//Individual, FOT, SOT and TOT do not have supertypes
pred supertypesOfBasicTypes{
  all i:Individual, s:SOT, f:FOT, t:TOT | some i.specializes or
    some f.specializes or some s.specializes or some t.specializes
}

//-----End of additional theorems representation-----
--
/*Assertion to verify all additional theorems*/
assert allAdditionalTheorems{
  iofCrossLevel
  and specializationProperties
  and specializationIntraLevel
  and subordinationIntraLevel
  and powertypeOfCrossLevel
  and categorizationCrossLevel
  and supertypesOfBasicTypes
}

//Command to check all the additional theorems considering a scope of 20 elements
/*To run the verification, uncomment the line bellow and comment the command used
to simulate the model as well as the command to run the verification of the other
theorems.*/

//check allAdditionalTheorems for 20

```

Appendix B. Specification of MLT in Alloy - Addressing Dynamic Classification

This appendix presents a specification of MLT, in Alloy, which addresses dynamic classification. This encompasses the axioms, definitions and theorems discussed in Section 3.7.

The whole specification is defined in one Alloy *module*. A *signature* “Entity” is defined to represent all entities in the domain of enquiry and a *signature* “World” is defined to represent the possible worlds (states-of-affairs). To capture the notion that the instantiation relation is world-indexed, we define a set of instantiations in each world, i.e. a world is seen as a set of instantiation relations holding between entities. On the other hand, since the other structural relations are not world-indexed, they are represented as properties of entities.

Further, one *signature* is created to represent each basic type. Both axioms and definitions are represented as *facts* while the theorems are represented as *predicates*. Finally *assertions* are created to group all the axioms to be checked.

```
module mltDynamicClassification

sig Entity{
  specializes: set Entity,
  properSpecializes: set Entity,
  isSubordinateTo: set Entity,
  powertypeOf: set Entity,
  categorizes: set Entity,
  compCategorizes: set Entity,
  disjCategorizes: set Entity,
  partitions: set Entity
}

some sig World{
  iof: set Entity -> Entity
}

//-----Basic types represented as singletons-----
//Representing the basic type "Individual"
one sig Individual extends Entity{}

//Representing the basic type "1stOT"
one sig FOT extends Entity{}
```



```

//Representing the basic type "2ndOT"
one sig SOT extends Entity{}

//Representing the basic type "3rdOT"
one sig TOT extends Entity{}

//-----End of Basic types representations-----

//-----Axioms and Definitions represented as facts-----
/*Axiom A1 - To be considered an instance of "Individual", an entity must have no
possible instance in any admissible world.*/
fact individualDef{
  all x:Entity, w1:World, i:Individual |
    (x in (w1.iof).i) iff (all w2:World | no (w2.iof).x)
}

/*Axiom A2 - Two types are considered the same iff they have the same instances in
all possible worlds */
fact typesEqualityDef{
  all t1,t2:Entity|
    ((some w1:World| some (w1.iof).t1) and (some w2:World| some (w2.iof).t2))
implies
  (t1 = t2 iff (all w:World, x:Entity| x in (w.iof).t1 iff x in (w.iof).t2))
}

/*Axiom A3 - An entity t is an instance first-order type ("FOT") iff all its instances
in all possible worlds are Individuals (i.e., instances of "Individual")*/
fact firstOrderTypeDef{
  all t:Entity, w1:World, f:FOT| (t in (w1.iof).f) iff
    ((some w2:World | some (w2.iof).t) and (all w3:World, x:Entity, i:Individual|
      x in (w3.iof).t implies x in (w3.iof).i))
}

/*Axiom A4 - An entity t is an instance second-order type ("SOT") iff all its instances
in all possible worlds are first-order types (i.e., instances of "FOT")*/
fact secondOrderTypeDef{
  all t:Entity, w1:World, s:SOT| (t in (w1.iof).s) iff
    ((some w2:World | some (w2.iof).t) and (all w3:World, x:Entity, f:FOT|
      x in (w3.iof).t implies x in (w3.iof).f))
}

/*Axiom A5 - An entity t is an instance third-order type ("TOT") iff all its instances
in all possible worlds are second-order types (i.e., instances of "SOT")*/
fact thirdOrderTypeDef{

```

```

all t:Entity, w1:World, th:TOT | (t in (w1.iof).th) iff
  ((some w2:World | some (w2.iof).t) and (all w3:World, x:Entity, s:SOT |
    x in (w3.iof).t implies x in (w3.iof).s))
}

/*Axiom A6 - Each entity in our domain of enquiry is necessarily an instance of
"Individual", "1stOT", "2ndOT" or "3rdOT" in all possible worlds(except "3rdOT" whose
type is outside the scope of the formalization).*/
fact completenessAxiom{
  all t:TOT, s:SOT, f:FOT, i:Individual, x:Entity, w:World |
    x in (w.iof).i or x in (w.iof).f or x in (w.iof).s or x in (w.iof).t or x=t}

/*Definition D1 - Specialization Definition: t1 specializes t2 iff, in all possible
worlds, all instances of t1 are also instances of t2.*/
fact specializationDef{
  all t1,t2:Entity | t2 in t1.specializes
  iff (all e:Entity, i:Individual, w:World |
    t1 not in (w.iof).i and t2 not in (w.iof).i and
    (e in (w.iof).t1 implies e in (w.iof).t2))
}

/*Definition D2 - Proper Specialization Definition: t1 proper specializes t2 iff t1
specializes t2 and is different from it.*/
fact properSpecializationDef{
  all t1,t2:Entity |
    t2 in t1.properSpecializes iff (t2 in t1.specializes and t1!=t2)
}

/*Definition D3 - Subordination Definition: t1 is subordinate to t2 iff, in all
possible worlds, every instance of t1 specializes an instance of t2.*/
fact subordinationDef{
  all t1,t2:Entity | t2 in t1.isSubordinateTo iff (all i:Individual, w:World |
    t1 not in (w.iof).i and (all t3:Entity | (t3 in (w.iof).t1 implies
    (some t4:Entity | t4 in (w.iof).t2 and t4 in t3.properSpecializes))))
}

/*Definition D4 - Powertype Definition: iff a type t1 is power type of a type t2 all
instances of t1 are specializations of t2 and all possible specializations of t2 are
instances of t1.*/
fact powertypeOfDef{
  all t1,t2:Entity | t2 in t1.powertypeOf iff (all t3:Entity, i:Individual, w:World |
    t1 not in (w.iof).i and (t3 in (w.iof).t1 iff t2 in t3.specializes))
}

```

```

/*Definition D5 - Categorization Definition: a type t1 categorizes a type t2 iff, in
all possible worlds, all instances of t1 are properSpecializations of t2.*/
fact categorizationDef{
    all t1,t2:Entity | t2 in t1.categorizes iff (all t3:Entity, i:Individual, w:World|
        t1 not in (w.iof).i and (t3 in (w.iof).t1 implies t2 in t3.properSpecializes))
}

/*Definition D6 - Complete Categorization Definition: a type t1 completely categorizes
a type t2 iff t1 categorizes t2 and, in all possible worlds, every instance of t2 is
instance of some instance of t1.*/
fact completeCategorizationDef{
    all t1,t2:Entity | t2 in t1.compCategorizes iff
        (t2 in t1.categorizes and (all e:Entity, w:World| e in (w.iof).t2 implies
            (some t3:Entity | e in (w.iof).t3 and t3 in (w.iof).t1)))
}

/*Definition D7 - Disjoint Categorization Definition: a type t1 disjointly categorizes
a type t2 iff t1 categorizes t2 and, in all possible worlds, every instance of t2 is
instance of, at most, one instance of t1.*/
fact disjointCategorizationDef{
    all t1,t2:Entity | t2 in t1.disjCategorizes iff
        (t2 in t1.categorizes and (all e:Entity, w:World|
            e in (w.iof).t2 implies
                (lone t3:Entity | e in (w.iof).t3 and t3 in (w.iof).t1)))
}

/*Definition D8 - Partition Definition: a type t1 partitions a type t2 iff t1
completely categorizes t2 and t1 disjointly categorizes t2.*/
fact partitionsDef{
    all t1,t2:Entity | t2 in t1.partitions iff
        (t2 in t1.disjCategorizes and t2 in t1.compCategorizes)
}

//-----End of Axioms and Definitions representations-----

//Command to simulate the theory considering a scope of 20 elements
run {} for 20

//-----Theorems represented as predicates-----
//Theorems T1, T2 and T3
pred theoremsT1T2T3{
    //T1: "Individual" is an instance of "1stOT"
    all i:Individual, f:FOT, w:World| i in w.iof.f
    //T2: "1stOT" is an instance of "2ndOT"
    all f:FOT, s:SOT, w:World| f in w.iof.s
}

```

```

    //T3: "2ndOT" is an instance of "3rdOT"
    all t:TOT, s:SOT, w:World | s in w.iof.t
}

/*Theorem T4: In all possible worlds, "Individual", "1stOT", "2ndOT" and "3rdOT" have
no instances in common (i.e., their extensions are disjoint).*/
pred theoremT4{
    all t:TOT, s:SOT, f:FOT, i:Individual, w:World | no x:Entity |
        (x in (w.iof).i and x in (w.iof).f) or (x in (w.iof).i and x in (w.iof).s) or
        (x in (w.iof).i and x in (w.iof).t) or (x in (w.iof).f and x in (w.iof).s) or
        (x in (w.iof).f and x in (w.iof).t) or (x in (w.iof).s and x in (w.iof).t)
}

/*Theorems T5 and T6: The instance of relation is irreflexive, asymmetric and anti-
transitive */
pred theoremsT5T6{
    //Assymmetric
    all x,y:Entity, w:World | x in w.iof.y => y not in w.iof.x
    //Irreflexive
    all x:Entity, w:World | x not in w.iof.x
    //Anti-transitive
    all x,y,z:Entity, w:World | (y in w.iof.x and z in w.iof.y) => z not in w.iof.x
}

/* Theorems T7, T8 and T9: Any instance of a higher-order type (any instance of
"1stOT", "2ndOT", and "3rdOT") specializes the basic type at an immediately lower
order.*/
pred theoremsT7T8T9{
    //T7: Every instance of "1stOT" specializes "Individual"
    all t:Entity, i:Individual, f:FOT, w:World | t in w.iof.f iff i in t.specializes
    //T8: Every instance of "2ndOT" specializes "1stOT"
    all t:Entity, f:FOT, s:SOT, w:World | t in w.iof.s iff f in t.specializes
    //T9: Every instance of "3rdOT" specializes "2ndOT"
    all t:Entity, s:SOT, th:TOT, w:World | t in w.iof.th iff s in t.specializes }

/* Theorems T10, T11 and T2
pred theoremsT10T11T12{
    //T10: "1stOT" is powertype of "Individual"
    all i:Individual, f:FOT | i in f.powertypeOf
    //T11: "2ndOT" is powertype of "1stOT"
    all f:FOT, s:SOT | f in s.powertypeOf
    //T12: "3rdOT" is powertype of "2ndOT"
    all s:SOT, t:TOT | s in t.powertypeOf
}

```

```

//Theorem T13: each type has at most one power type
pred theoremT13{
  all t:Entity| lone powertypeOf.t
}

//Theorem T14: each type is power type of, at most, one other type
pred theoremT14{
  all t:Entity| lone t.powertypeOf
}

/*Theorem T15: if a type t2 specializes a type t1 then the power type of t2 specializes
the power type of t1.*/
pred theoremT15{
  all t1,t2,t3,t4:Entity |
    (t1 in t2.specializes and t2 in t4.powertypeOf and t1 in t3.powertypeOf)
    implies t3 in t4.specializes
}

/*Theorem T17: If a type t2 is power type of a type t1 and a type t3 categorizes the
same base type t1 then all instances of t3 are also instances of the power type t2
and, thus, t3 proper specializes t2.*/
pred theoremT17{
  all t1,t2,t3:Entity | (t1 in t2.powertypeOf and t1 in t3.categorizes)
    implies t2 in t3.properSpecializes
}

/*Theorem T18: if two types t1 and t2 both partitions the same type t3 then it is
not possible for t1 to specialize t2*/
pred theoremT18{
  all t1,t2,t3:Entity | (t3 in t1.partitions and t3 in t2.partitions)
    implies (t2 not in t1.properSpecializes)
}

//-----End of theorems representation-----
/*Assertion to verify all theorems*/
assert allTheorems{
  theoremsT1T2T3
  and theoremT4
  and theoremsT5T6
  and theoremsT7T8T9
  and theoremsT10T11T12
  and theoremT14
  and theoremT15
  and theoremT17
  and theoremT18
}

```

```

}

//Command to check the theorems considering a scope of 15 elements
/*To run the verification, uncomment the line bellow and comment the command used to
simulate the model as well as the command to run the verification of the other
theorems.*/

//check allTheorems for 15

/*----- Rules cited on the text but not formally stated as Theorems are tested here
as theorems-----*/

/*Instantiation relations hold between two elements such that the last is one order
higher than the former.*/
pred iofCrossLevel{
  all x,y:Entity, i:Individual, f:FOT, s:SOT, t:TOT, w:World |
    y in (w.iof).x implies
      ((y in (w.iof).i and x in (w.iof).f) or
       (y in (w.iof).f and x in (w.iof).s) or
       (y in (w.iof).s and x in (w.iof).t) or (y in (w.iof).t and x=t))
}

/*Specialization is a partial order relation (i.e., a reflexive, transitive and
antisymmetric relation). */
pred specializationProperties{
  //Antissymmetric
  all x,y:Entity | (x in y.specializes and x!=y) => y not in x.specializes
  //Reflexive
  all x:Entity | (some w:World, y: Entity | y in (w.iof).x) => x in x.specializes
  //Transitive
  all x,y,z:Entity |
    (y in x.specializes and z in y.specializes) => z in x.specializes
}

/*Specializations and proper Specializations may only hold between types of the same
order*/
pred specializationIntraLevel{
  all x,y:Entity, f:FOT, s:SOT, t:TOT, w:World | y in x.specializes implies
    ((x in (w.iof).f and y in (w.iof).f ) or (x in (w.iof).s and y in (w.iof).s)
or
    (x in (w.iof).t and y in (w.iof).t) or
    (t in x.specializes and t in y.specializes))
}

```

```

//Subordinations can only hold between higher-order types of equal order
pred subordinationIntraLevel{
  all x,y:Entity, s:SOT, t:TOT, w:World| y in x.isSubordinateTo implies
    ((x in (w.iof).s and y in (w.iof).s) or (x in (w.iof).t and y in (w.iof).t) or
      (t in x.specializes and t in y.specializes))
}

// PowertypeOf relations only occur between types of adjacent levels
pred powertypeOfCrossLevel{
  all x,y:Entity, f:FOT, s:SOT, t:TOT, w:World| x in y.powertypeOf implies
    ((x in (w.iof).f and y in (w.iof).s) or (x in (w.iof).s and y in (w.iof).t) or
      (x in (w.iof).t and t in y.specializes))
}

//Categorization relations only occur between types of adjacent levels
pred categorizationCrossLevel{
  all x,y:Entity, f:FOT, s:SOT, t:TOT, w:World | x in y.categorizes implies
    ((x in (w.iof).f and y in (w.iof).s) or (x in (w.iof).s and y in (w.iof).t) or
      (x in (w.iof).t and t in y.specializes))
}

//Individual, FOT, SOT and TOT do not have supertypes
pred supertypesOfBasicTypes{
  all i:Individual, s:SOT, f:FOT, t:TOT | some i.specializes or
    some f.specializes or some s.specializes or some t.specializes
}

//-----End of additional theorems representation-----
/*Assertion to verify all additional theorems*/
assert allAdditionalTheorems{
  iofCrossLevel
  and specializationProperties
  and specializationIntraLevel
  and subordinationIntraLevel
  and powertypeOfCrossLevel
  and categorizationCrossLevel
  and supertypesOfBasicTypes
}

//Command to check all the additional theorems considering a scope of 20 elements
/*To run the verification, uncomment the line bellow and comment the command used to
simulate the model as well as the command to run the verification of the other
theorems.*/

//check allAdditionalTheorems for 20

```

