Modeling Hetero-Homogeneous Hierarchies and Mapping to OWL

DIPLOMARBEIT

zur Erlangung des akademischen Grades

MAG.RER.SOC.OEC.

im Diplomstudium

WIRTSCHAFTSINFORMATIK

Angefertigt am *Institut für Wirtschaftsinformatik - Data & Knowledge Engineering*

Betreuung/Begutachter:
o.Univ.–Prof. Dr. Michael Schrefl

Mitbetreuung:
Mag. Dr. Bernd Neumayr

Eingereicht von:
Dipl.-Ing. Dr.techn. Karl Rieger, MSc.

Linz, im *Juni 2010*.

Abstract

The diploma thesis is devoted to a conceptual modeling approach for *hetero-homogeneous hierarchies* with multiple levels of abstraction. These hierarchies are homogeneous with respect to a common schema, defined by a single root node and prescribed for all sub-hierarchies, and are heterogeneous with respect to refined schemata for sub-hierarchies. In particular, the approach relies on so-called *Generalized Multi-Level Objects* and *Generalized Multi-Level Relationships*, which permit to represent objects and relationships among these objects at different levels. Both constructs can be arranged in concretization hierarchies, which compromise different aspects of abstraction hierarchies like e.g. aggregation, classification and generalization.

In addition, a semantic-preserving mapping of such hierarchies to the decidable fragment of OWL (Web Ontology Language) is presented to render the approach from conceptual modeling to the semantic web and ontological engineering. By suitably combining closed world assumption and open world assumption, a meta-modeling facility in OWL is obtained, where OWL semantic reasoners allow e.g. consistency checks and respective query executions. Finally, data can be accessed in the semantic web at different abstraction levels of hetero-homogeneous hierarchies, while its quality is improved.

In the diploma thesis we mainly continue the ideas of Neumayr et al. [2009a,b, 2010], which are partially revised, compared to and complemented with own ideas and results.

Kurzfassung

Die vorliegende Diplomarbeit widmet sich einem konzeptionellen Modellierungsansatz für *hetero-homogene Hierarchien* mit mehreren Abstraktionsebenen. Diese Hierarchien sind homogen im Hinblick auf ein gemeinsames Schema, welches durch ein Wurzelelement definiert und für alle Unterhierarchien vorgeschrieben wird, und heterogen im Hinblick auf verfeinerte Schemata für Unterhierarchien. Im Konkreten basiert der Ansatz auf so genannten *Generalized Multi-Level Objects* und *Generalized Multi-Level Relationships*, welche es gestatten, Objekte und Beziehungen zwischen diesen Objekten in verschiedenen Abstraktionsebenen darzustellen. Beide Konstrukte können in Konkretisierungshierarchien angeordnet werden, welche unterschiedliche Aspekte von Abstraktionshierarchien, wie z.B. Aggregation, Klassifikation und Generalisierung, in sich vereinigen.

Zusätzlich wird eine Semantik erhaltende Abbildung von diesen Hierarchien auf den entscheidbaren Teil von OWL (Web Ontology Language) vorgestellt, um den konzeptionellen Modellierungsansatz auf das Semantische Web und das Arbeiten mit Ontologien zu übertragen. Durch geeignetes Kombinieren der Closed World Assumption und der Open World Assumption entsteht eine Meta-Modellierungstechnik in OWL, wobei OWL Reasoners z.B. Konsistenzprüfungen und entsprechende Abfragen erlauben. Schlussendlich kann auf Daten im Semantischen Web in unterschiedlichen Abstraktionsebenen innerhalb hetero-homogener Hierarchien zugegriffen und gleichzeitig deren Qualität verbessert werden.

In dieser Arbeit werden im Wesentlichen die Ideen von Neumayr et al. [2009a,b, 2010] aufgegriffen, teilweise überarbeitet und mit eigenen Ideen und Resultaten verglichen und ergänzt.

Contents

List of Figures

1 Introduction

In this introductory chapter some background information about the semantic web, ontologies and description logic is provided, and in this context the relevance of multi-level modeling is revealed. Afterward the purpose and main goals of this diploma thesis are presented. The chapter finishes with an outline of the present work.

1.1 Semantic Web, Ontologies and OWL

Let us first sketch why the semantic web, see, e.g., Berners-Lee et al. [2001], requires ontologies and why description logic is well-suited for ontology languages. Semantic web aims at making intelligent web pages meaningful for machines. Thus, it intends to provide machine-understandable and -processable web resources, which can be shared by agents, like automated tools, search engines, human users, etc. In order to ensure a common understanding of the agents an ontology is required attached with well-defined formal semantics. An ontology is a collection of concepts, where all agents have the same interpretation of the concepts with respect to the ontology. The use of ontologies requires a suitable ontology language. Its semantic should be formally specified and its expressive power should be adequate for defining relevant concepts at a detail level of interest. In summary, ontologies play a pivotal role, because it provides a common and shared agreement of a specific domain. In current computer science, ontology is said to be an "*agreement about a shared, formal, explicit and partial account of a conceptualization*", see Spyns et al. [2002] and references therein. The ontology contains the vocabulary (concepts and terms) and the definition of these concepts and their relationships for a specific domain.

Currently, the Web Ontology Language (OWL), a W3C recommendation, is to be the *de-facto* standard for modeling ontologies in the semantic web. The logical underpinning for OWL is provided by description logics (DLs) (as the decidable fragment of first-order logics (FOL) in terms of predicate logic), which represents a family of knowledge representation formalisms with well-understood semantics and computational properties. OWL can be seen as an expressive schema language that can be used to provide flexible access to data. High quality ontologies are crucial for the semantic web and their construction and evolution greatly depends on the availability of well-defined semantics and efficient reasoning tools. DL contains both such that they are typical candidates for ontology languages, i.e., it can be used as a formalism for representing and reasoning about ontologies. In particular, the design of OWL is based on the SH family of DLs, where, in particular, OWL 2 DL is based on SHROIQ, see, e.g., Hitzler et al. [2008], the OWL semantics and syntax definitions of W3C[1], or the OWL

[1]See http://www.w3c.org/TR/owl-semantics/.

technical report of W3C [2].

However, to access data using the semantic web such data must often be converted to OWL. Much research has been performed on the topic of information integration and knowledge combination. The main goal of these researches is to make a connection between users and heterogeneous information systems that can be applied in intranet and internet environments. Some approaches tend to create a new ontology and other approaches create a mapping to an already existing ontology. In particular, there are several possible solutions available to create mappings from relational database schema to web ontologies, see, e.g., Albarrak and Sibley [2009], Levshin [2009], Sane and Shirke [2009], Xu et al. [2006] and references therein, or from XML and XML Schema to OWL, see, e.g., Anicic et al. [2007], Kobeissy et al. [2007], and references therein. Hence, schema statements in OWL (with the *open world assumption* (OWA)) are interpreted in a different way to similar statements in a relational database setting (with the *closed world assumption* (CWA)). To bridge the gap between OWA and CWA Motik et al. [2007, 2009] proposes an extension of OWL to *extended DL knowledge bases* that mimic the intuition behind integrity constraints in relational databases. In particular, the modeler is allowed to designate a subset of TBox axioms as integrity constraints. For TBox (schema) reasoning these axioms are treated as usual axioms, whereas for ABox (data) reasoning they are treated as checks and do not derive additional information. Thus, the approach provides a clear semantic relationship between the roles that integrity constraints play during TBox and ABox reasoning.

The distinction between classes and instances has gathered great importance whether done automatically or manually, and underlies many present discussions in different knowledge representation systems. There are inherent requests to involve the capability to incorporate a distinction between classes and instances, driven by, e.g., ontologies. Hence, it turns out that the clear distinction is not always possible, see, e.g., Miller and Hristea [2006]. Mainly, the distinction is not drawn explicitly since there is only a "*is a*" relation, where a difference between classes and instances is made.

1.2 Ontological Multi-level Modeling

In the recent years multi-level modeling has attracted an increased attention in the field of semantic data modeling since objects are often organized in hierarchies with multiple levels. Original data sources are often heterogeneous in an inherent manner, while current modeling and implementation techniques often discard these heterogeneities. Typical examples are production hierarchies, dimension hierarchies in data warehouses, and taxonomies in general. In particular, *ontological multi-level modeling* or meta-modeling is achieved by the representation of domain objects at different levels of abstraction, which are part of an abstraction hierarchy. In contrast to linguistic meta-modeling, which is used to define or extend modeling languages, ontological multi-level modeling is used to model complex domains, where the

[2]See http://www.w3c.org/TR/owl-features.

distinction between classes and instances is contextual in general.

Hence, a multiple representation normally accompanies an increased complexity and complicates modeling, maintenance and extension. These supplementary effects may result from the description of objects with non-uniform structure at the same level. Due to the rapidly growth of information systems in size and the requirement of system integration, a need to handle levels of similar, but heterogeneous objects arise.

Several modeling techniques have been proposed to overcome these undesirable effects and unnecessary complexity. Main techniques are *Power Types*, *Potency-Deep Instantiation*, *Materialization* and *Multi-Level Objects* (m-objects), where the interested reader is referred to, e.g., Neumayr and Schrefl [2008], Neumayr et al. [2009b] and references therein for an overview and a comparison of the methods. These approaches allow to describe domain concepts with members at multiple levels of abstraction. In contrast to the rest of the mentioned multi-level modeling techniques, which focus more or less on a single type of semantic abstraction hierarchies, especially m-objects (together with *Multi-Level Relationships* (m-relationships)) intend to present domain objects at levels of abstraction in more or less very general abstraction hierarchies, see, e.g., Neumayr et al. [2009a]. The main idea is to encapsulate the levels of abstraction, which relate to a single domain concept, into a m-object and to represent general hierarchies by a single concretization hierarchy. Moreover, the approach intends to integrate aspects of different abstraction hierarchies, including aggregation, classification and generalization, in a single concretization hierarchy. This fact should lead to an improvement of readability and a simplified modeling and maintenance.

If a concept hierarchy is considered, which states if concepts are specialization of other concepts or if they are synonyms, again OWL provides a language that can be applied to describe classes and relations between them that are inherent in web documents and applications. Reasoning can be applied, e.g., to test whether concepts are non-contradictory or to derive hidden relations. This procedure is highly relevant to guarantee the quality of an ontology. If the ontology is deployed, even the consistency of facts or instance relationships can be inferred. Reasoning means deciding satisfiability and subsumption of concepts with respect to TBoxes and role hierarchies. In Neumayr and Schrefl [2009] a semantic-preserving mapping of m-objects and m-relationships to OWL is presented, especially, to transfer the formal conceptual modeling approach to ontological engineering. Then, previously-mentioned advantages from ontological engineering can be utilized like e.g. consistency checks, query facilities.

1.3 Main Ideas and Aims of the Diploma Thesis

The diploma thesis aims to provide a conceptual multi-level modeling approach, which can be combined with ontological engineering. The novelty of the present work is stated by an appropriate extension of the m-object and m-relationship approach in order to integrate further important aspects of general abstraction hierarchies, see, e.g., Neumayr et al. [2010]. In this context so-called *Generalized Multi-Level Objects* (gm-objects) and *Generalized Multi-Level*

Relationships (gm-relationships) are introduced. The intention is that the proposed modeling technique inherits the advantages of m-objects/m-relationships and, simultaneously, incorporates additional essential facets of more general abstraction hierarchies. In contrast to m-objects/m-relationships the presented multi-level modeling approach is characterized by its multi-dimensional character. The modeling approach is especially devoted to *Hetero-Homogeneous Hierarchies* (HH-Hierarchies), which are hierarchies with a single root node, being

- homogeneous with respect to a minimal common schema, which is shared by any sub-hierarchy, whose root note is realized by a child node of the hierarchy, and
- heterogeneous with respect to refined schemata of sub-hierarchies.

In addition, analogous to Neumayr and Schrefl [2009] a semantic-preserving mapping of gm-objects and gm-relationships to the decidable part of OWL is presented, while combining OWA and CWA. This instrument enhances ontological engineering and, consequently, data access in the semantic web.

Remark 1. In the present diploma thesis we mainly continue the ideas of Neumayr et al. [2009a], Neumayr and Schrefl [2009], Neumayr et al. [2010], which are partially revised, compared to and complemented with own ideas.

1.4 Outline of the Diploma Thesis

The rest of the diploma thesis is organized in chapters and can briefly be sketched as follows.

Chapter 2 In this chapter gm-objects and gm-relationships are introduced, and their relevance for hetero-homogeneous hierarchies is discussed.

Chapter 3 An appropriate semantic-preserving mapping to OWL in form of algorithms for gm-objects and gm-relationships is presented. Relevant issues regarding, e.g., CWA and OWA are discussed and (mapping) results are compared to Neumayr and Schrefl [2009].

Chapter 4 Additional related work with respect to meta-modeling support in OWL and suggestions for further work are revealed, and some conclusions of our results are provided.

The notation and important mathematical preliminaries are introduced when necessary. In addition, several examples are arranged in the diploma thesis to illustrate the results.

2 Modeling Hetero-Homogeneous Hierarchies

In this chapter *Generalized Multi-Level Objects* (gm-objects) and *Generalized Multi-Level Relationships* (gm-relationships) are introduced. It is highlighted that by using these objects and relationships as well as associated concretizations and concretization hierarchies a conceptual modeling approach is obtained, which is capable to model hetero-homogeneous hierarchies, see chapter 1.3. Several examples demonstrate the applicability of the proposed technique.

2.1 Generalized Multi-Level Objects

In this section the notions of a gm-object, a consistent (individual) concretization of a gm-object, and a consistent concretization hierarchy of gm-objects are defined.

2.1.1 Definition of GM-Object

With respect to m-objects the arrangement of abstraction levels is restricted to linear order from the most abstract to the most concrete one, see, e.g., Neumayr et al. [2009a], whereas now gm-objects permit to encapsulate and arrange abstraction levels in an almost arbitrary order. Hence, the levels are still arranged from the most abstract one to the most concrete ones. In addition, there can be references to abstraction levels of other gm-objects, which are called *level references*. A gm-object describes itself and the common properties of the objects at each level of the concretization hierarchy beneath itself. A gm-object specifies concrete values for the properties of its top-level. This top-level has a special role in that it describes the gm-object itself.

Let us start with a formal definition of a gm-object, allowing a partial (non-linear) order of levels.

Definition 2 (GM-Object with Level References)**.** A gm-object

$$o = (L_o, LR_o, A_o, p_o, pr_o, l_o, d_o, v_o)$$

consists of a set of *levels* L_o, a set of *level references* LR_o and a set of *attributes* A_o, which are taken from an universe of levels L, level references LR and attributes A, respectively. The order of levels L_o is given by the partial function *parent*[1]

$$p_o : L_o \to \mathcal{P}(L_o);\ l \mapsto U_l \subset L_o\,,$$

[1] $\mathcal{P}(L_o)$ denotes the power set $\mathcal{P}(L_o) := \{U : U \subset L_o\}$ of L_o. Another typical notation is 2^{L_o}.

which associates with each level its *parent levels*, where for all $l \in L_o$ the relation $l \notin p_o(l) = U_l$ holds. Further, the partial function *parent reference*

$$pr_o : L_o \to \mathscr{P}(LR_o);\ l \mapsto \bar{U}_l \subset LR_o\,,$$

associates with each level its *parent level references*. A gm-object has a single top level $\hat{l}_o \in L_o$ with $p_o(\hat{l}_o) = \{\}$ and $pr_o(\hat{l}_o) = \{\}$. The set $P_o \subseteq L_o \times L_o$, where $(l, \bar{l}) \in P_o$ iff $\bar{l} \in p_o(l)$ such that $P_o = \bigcup_{l \in L_o \setminus \{\hat{l}_o\}} (\{l\} \times \{p_o(l)\})$, is denoted *parent relation*. Each attribute is associated with one level, which is defined by the function *level*

$$l_o : A_o \to L_o; a \mapsto l \in L_o\,,$$

and has domain, which is defined by the function *domain*

$$d_o : A_o \to D; a \mapsto d \in D\,,$$

where D is an universe of data types. Optionally, an attribute has a value from its domain, which is defined by the partial function *value*

$$v_o : A_o \to V; a \mapsto v \in V\,,$$

where V is an universe of data values. The attributes $\hat{A}_o$ of the top-level $\hat{l}_o$ are specified entirely by values of their domain.

In Neumayr et al. [2010] a different definition[2] of a similar object is provided, where in comparison to Definition 2 level references are neglected. Therefore, we also state an alternative definition of a gm-object here.

Definition 3 (GM-Object without Level References)**.** A gm-object

$$o = (L_o, A_o, p_o, l_o, d_o, v_o)$$

consists of a set of *levels* L_o and a set of *attributes* A_o, which are taken from an universe of levels L and attributes A. The order of levels L_o is given by the partial function *parent*

$$p_o : L_o \to \mathscr{P}(L_o);\ l \mapsto U_l \subset L_o\,,$$

which associates with each level its *parent levels*, where for all $l \in L_o$ the relation $l \notin p_o(l) = U_l$ holds. A gm-object has a single top level $\hat{l}_o \in L_o$ with $p_o(\hat{l}_o) = \{\}$. The set $P_o \subseteq L_o \times L_o$, where $(l, \bar{l}) \in P_o$ iff $\bar{l} \in p_o(l)$ such that $P_o = \bigcup_{l \in L_o \setminus \{\hat{l}_o\}} (\{l\} \times \{p_o(l)\})$, is denoted *parent relation*. Each attribute is associated with one level, which is defined by the function *level*

$$l_o : A_o \to L_o; a \mapsto l \in L_o\,,$$

[2]In Neumayr et al. [2010] the object is denoted m-object.

and has domain, which is defined by the function *domain*

$$d_o : A_o \to D; a \mapsto d \in D,$$

where D is an universe of data types. Optionally, an attribute has a value from its domain, which is defined by the partial function *value*

$$v_o : A_o \to V; a \mapsto v \in V,$$

where V is an universe of data values. The attributes $\hat{A}_o$ of the top-level $\hat{l}_o$ are specified entirely by values of their domain.

The parent relation P_o represents an alternative description for the order of abstraction levels. Its transitive closure is P_o^+ and its transitive and reflexive closure is P_o^*. Moreover, a level $\bar{l}$ is called a child of a level l iff $(\bar{l}, l) \in P_o$, and $\bar{l}$ is a descendant of, or below, l iff $(\bar{l}, l) \in P_o^+$ and $\bar{l}$ is a descendant of or the same as l iff $(\bar{l}, l) \in P_o^*$. Moreover, level references are written in the form $o : l$, where o states the gm-object and l the appealed level of abstraction of the gm-object o.

It is worth noting that concretize-relationships[3] between levels describe direct relations to descendant or ancestor levels and do not consider implicit relations between levels. The set of second top-levels of a gm-object o is given by

$$\{l \in L_o : p_o(l) = \hat{l}_o\}$$

and the set, which has the level $\bar{l}_o \in L_o$ and all of its direct and indirect descendants as its elements, is denoted by

$$L_o|_{\bar{l}_o} := \{\bar{l}_o\} \cup \{l \in L_o : \exists k \in \mathbb{N}\ p_o(\ldots(p_o(l))) = (p_o)^k(l) = \bar{l}_o\},$$

where $(p_o)^k(\cdot)$ denotes the k-times application of p_o. In addition, the set of attributes related to $L_o|_{\bar{l}_o}$ reads

$$A|_{L_o|_{\bar{l}_o}} = \{a \in A_o : l_o(a) \in L_o|_{\bar{l}_o}\}.$$

Let O denote a set of gm-objects, then a gm-object $o \in O$ is said to be at level l if l is its top-level $\hat{l}_o$. In particular, gm-objects, levels, and attributes have names, which are defined by the function *name*

$$n : O \cup L \cup A \to N,$$

where N is an universe of names.

In order to exemplify gm-objects we take a look at different ranges of applications.

[3]In this work we restrict to 1 : 1 concretize-relationships.

Example 4. Let us consider the gm-object *Product* as depicted in figure 2.1, which consists of four levels

$$L_{\text{Product}} = \{\text{Catalog, Category, Model, PhysicalEntity}\}$$

with one attribute each. The levels are arranged in linear order, where the parent level of *Category* is the level *Catalog*,

$$p_{\text{Product}}(\text{Category}) = \text{Catalog}\,,$$

the parent level of *Model* is *Category*,

$$p_{\text{Product}}\left(\text{Model}\right) = \text{Category}\,,$$

and the parent level of *PhysicalEntity* is *Model*,

$$p_{\text{Product}}\left(\text{PhysicalEntity}\right) = \text{Model}\,.$$

The top-level *Catalog*,

$$\hat{l}_{\text{Product}} = \text{Catalog}\,,$$

defines the attribute *description*,

$$l_{\text{Product}}(\text{description}) = \text{Catalog}\,,$$

with domain String,

$$d_{\text{Product}}(\text{description}) = \text{String}\,,$$

and assigns a specific value,

$$v_{\text{Product}}(\text{description}) = \text{„Our products“}\,.$$

In [Neumayr et al., 2009a, p. 2] the same example is used for m-objects, which shows that gm-objects are a generalization of m-objects.

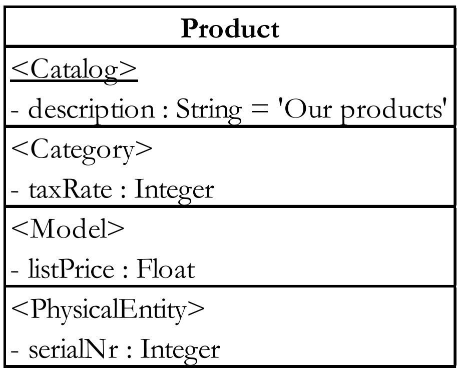

Figure 2.1: Gm-object *Product* with linear order of levels

In comparison to m-objects gm-objects are even capable to treat more general hierarchies of abstraction levels.

Example 5. Let us consider the gm-object *Location* as depicted in figure 2.2, which consists of four levels

$$L_{\text{Location}} = \{\text{T, country, region, city}\}\,.$$

The levels are not arranged in linear order. The parent-child dependencies are given by

$$\begin{aligned} p_{\text{Location}}(\text{country}) &= \text{T}, \\ p_{\text{Location}}(\text{region}) &= \text{T}, \\ p_{\text{Location}}(\text{city}) &= \{\text{country, region}\}\,. \end{aligned}$$

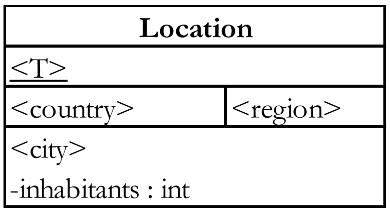

Figure 2.2: Gm-object *Location* with nonlinear order of levels

With regard to Definition 2 level references can arise, i.e., where $LR_o \neq \{\}$ for a gm-object o. This may happen if we concretize gm-objects with a general (non-linear) order of abstraction levels.

Example 6. Let us consider the gm-objects *Austria* and *Alps* as depicted in figure 2.3, which consist of the levels

$$L_{\text{Austria}} = \{\text{country, city}\}$$

and

$$L_{\text{Alps}} = \{\text{region, city}\}\,.$$

The gm-object *Austria* contains a level reference,

$$pr_{\text{Austria}}(\text{city}) = \{\text{Alps:region}\}\,.$$

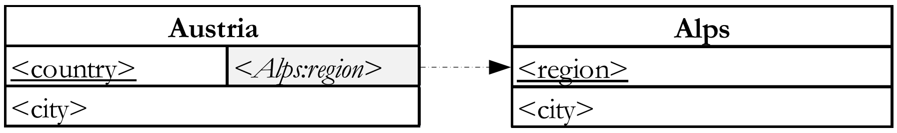

Figure 2.3: Gm-objects *Austria* and *Alps* with level reference

2.1.2 Concretizations

A gm-object can concretize other gm-objects, which are referred as its parents. A concretization of gm-objects, which is a gm-object itself, can be regarded as an instance of its parent gm-objects. Then, the top-level of a gm-object is an instance of a specific second top-levels of all parent gm-objects. Thus, a child gm-object selects its single top-level from the common second top levels of its parent gm-objects and in the case of multiple concretization the top-level of the child gm-object must be a common second-top level of its parent gm-objects.

The concretize-relationship especially compromise e.g. classification, generalization and aggregation. However, a concretize-relationship between gm-objects does not reflect that one m-object is at the same time an *instance of*, *component of*, and *subclass of* another gm-object as a whole. The gm-object inherits the second top-levels, all direct and indirect descendant levels together with the relative order of these common levels and, eventually, level references. It can even specify values for its attributes. Moreover, the concretize-relationships to the parents must compromise the neglected concretize-relationships of the parents. The level descriptions of a gm-object correspond to subclasses of the corresponding levels of its parents. The gm-object can define new levels or add/concretize attributes as well as concretization-relationships to other levels. In any case a gm-object must not change the relative order of levels, which are inherited from its parents.

In the following we define what we mean with a consistent (individual) concretization, hence, we do not explicitly describe the inheritance mechanism for simplicity. First, gm-objects with level references are considered, see Definition 2.

Definition 7 (Consistent Concretization of GM-Objects with Level References)**.** A gm-object $\bar{o}$ is a consistent concretization of another gm-object o iff

1. (Second Top-Level Instantiation) The top-level $\hat{l}_{\bar{o}}$ of $\bar{o}$ is a second top-level $\bar{l}_o$ of o,
$$(\hat{l}_{\bar{o}}, \hat{l}_o) \in P_o\,.$$

2. (Level Containment) The second top level $\bar{l}_o$ and all of its (direct and indirect) descendant levels are also levels of $\bar{o}$,
$$\bar{L}_o = L_o|_{\bar{l}_o} \subseteq L_{\bar{o}}\,.$$

3. (Level References Containment) All level references related to $\bar{L}_o \setminus \{\bar{l}_o\}$ are levels or level references in $\bar{o}$,
$$pr_o(\bar{L}_o \setminus \{\bar{l}_o\}) \subseteq (L_{\bar{o}} \cup LR_{\bar{o}}),$$
and neglected concretization relationships related to $\bar{L}_o \setminus \{\bar{l}_o\}$ are added as level references,
$$(p_o(\bar{L}_o \setminus \{\bar{l}_o\}) \setminus \bar{L}_o) \subseteq (L_{\bar{o}} \cup LR_{\bar{o}})\,.$$

4. (Attribute Containment) All attributes related to $\bar{L}_o$ also exist in $\bar{o}$, i.e.,
$$A|_{\bar{L}_o} \subseteq A_{\bar{o}} .$$

5. (Level Order Compatibility) The relative order of common levels of o and $\bar{o}$ is the same, i.e.,
$$l, \bar{l} \in (L_o \cap L_{\bar{o}}) : (l, \bar{l}) \in P^+_{\bar{o}} \Rightarrow (l, \bar{l}) \in P^+_o .$$

6. (Local Level Order) Levels newly introduced in $\bar{o}$ have parents only within $\bar{o}$, i.e.,
$$\forall (l, \bar{l}) \in P_{\bar{o}} : l \in (L_{\bar{o}} \setminus L_o) \Rightarrow \bar{l} \in L_{\bar{o}} .$$

7. Common attributes are associated with the same level, have the same domain, and the same value, if defined, i.e., for $a \in (A_o \cap A_{\bar{o}})$ it follows

   (a) (Stability of Attribute Level) $l_o(a) = l_{\bar{o}}(a)$,

   (b) (Stability of Attribute Domains) $d_o(a) = d_{\bar{o}}(a)$,

   (c) (Compatibility of Attribute Values) If $v_o(a)$ is defined, it follows $v_o(a) = v_{\bar{o}}(a)$.

If gm-objects without level references are applied according to Definition 3, we have a different definition of a consistent (individual) concretization.

Definition 8 (Consistent Concretization of GM-Objects without Level References)**.** A gm-object $\bar{o}$ is a consistent concretization of another gm-object o iff

1. (Second Top-Level Instantiation) The top-level $\hat{l}_{\bar{o}}$ of $\bar{o}$ is a second top-level $\bar{l}_o$ of o,
$$(\hat{l}_{\bar{o}}, \hat{l}_o) \in P_o .$$

2. (Level Containment) The second top level $\bar{l}_o$ and all of its (direct and indirect) descendant levels are also levels of $\bar{o}$,
$$\bar{L}_o = L_o|_{\bar{l}_o} \subseteq L_{\bar{o}} .$$

3. (Attribute Containment) All attributes related to $\bar{L}_o$ also exist in $\bar{o}$, i.e.,
$$A|_{\bar{L}_o} \subseteq A_{\bar{o}} .$$

4. (Level Order Compatibility) The relative order of common levels of o and $\bar{o}$ is the same, i.e.,
$$l, \bar{l} \in (L_o \cap L_{\bar{o}}) : (l, \bar{l}) \in P^+_{\bar{o}} \Rightarrow (l, \bar{l}) \in P^+_o .$$

5. (Local Level Order) Levels newly introduced in $\bar{o}$ have parents only within $\bar{o}$, i.e.,

$$\forall (l, \bar{l}) \in P_{\bar{o}} : l \in (L_{\bar{o}} \setminus L_o) \Rightarrow \bar{l} \in L_{\bar{o}}.$$

6. Common attributes are associated with the same level, have the same domain, and the same value, if defined, i.e., for $a \in (A_o \cap A_{\bar{o}})$ it follows

   (a) (Stability of Attribute Levels) $l_o(a) = l_{\bar{o}}(a)$,

   (b) (Stability of Attribute Domains) $d_o(a) = d_{\bar{o}}(a)$,

   (c) (Compatibility of Attribute Values) If $v_o(a)$ is defined, it follows $v_o(a) = v_{\bar{o}}(a)$.

Example 9. Let us consider the gm-object *Product* from Example 4, which is concretized by the gm-object *Car*. The gm-object *Car* instantiates *Category*, i.e., it has *Category* as its top-level, and specifies the value for the attribute *taxRate*, as depicted in figure 2.4. Moreover, an extra level *Brand* is introduced between the levels *Category* and *Model*. According to Definition 7 and Definition 8 the gm-object *Car* is a consistent (individual) concretization of the gm-object *Product* because all required conditions are fulfilled.

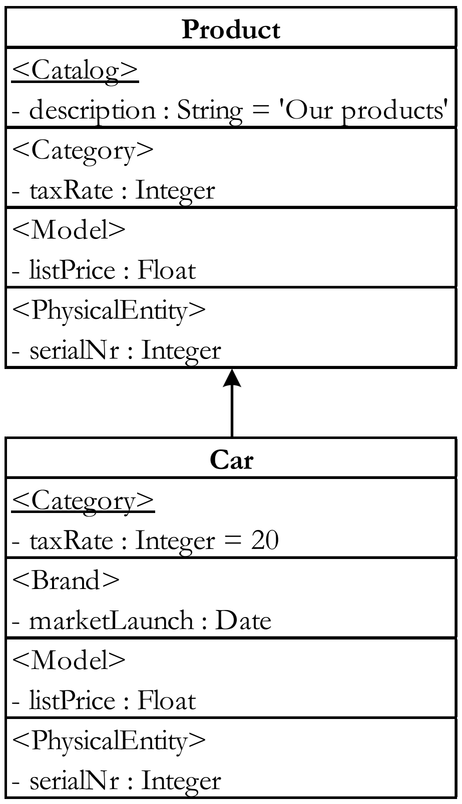

Figure 2.4: Consistent concretization of gm-object *Product* by gm-object *Car*

Example 10. Let us consider the gm-object *Location* from Example 5, which is concretized by the gm-objects *Austria* and *Alps* with the corresponding top-levels *country* and *region* and level references *Location:country* and *Location:region*, respectively, as depicted in figure 2.5. Both gm-objects are consistent (individual) concretizations of the gm-object *Location* in the sense of Definition 7.

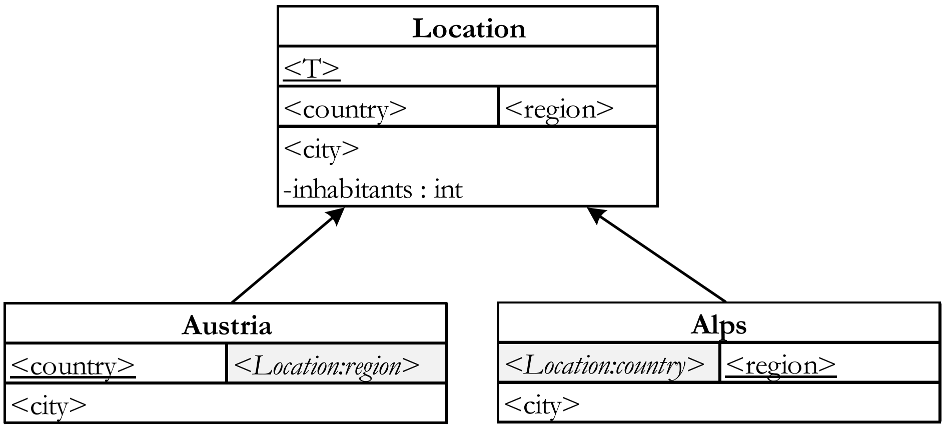

Figure 2.5: Consistent concretizations of gm-object *Location* with level references

Example 11. Let us again consider the gm-object *Location* from Example 5, which is concretized by the gm-objects *Austria* and *Alps* with the corresponding top-levels *country* and *region*, as depicted in figure 2.6. Both gm-objects are consistent (individual) concretizations of the gm-object *Location* with respect to Definition 8, but not with respect to Definition 7.

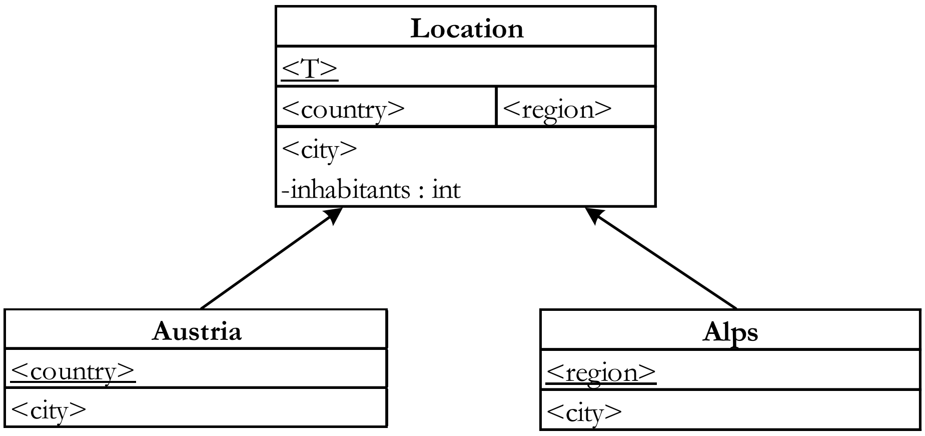

Figure 2.6: Consistent concretizations of gm-object *Location*

2.1.3 Concretization Hierarchies

Gm-objects can be organized in a hierarchy. The concretization path between gm-objects of different levels expresses a hierarchy at the instance level. At the schema level the level hierarchy is given by the order of levels of a gm-object. In order to avoid conflicts, the names of gm-objects, attributes and levels are supposed to be unique within the considered context.

Gm-objects organized in a concretization hierarchy inherit properties and functions from their parent gm-objects and, thus, may only be partially defined, i.e., a child gm-object inherits from its parent gm-objects several properties, i.e., certain levels, attributes and function definitions. The inherited properties are at the child's disposal in all function definitions. These respective functions, if only partially specified at the child, are extended for undefined arguments using the corresponding functions of its parents.

Apparently, the definition of a consistent (individual) concretization of gm-objects leads us directly to the notion of a concretization hierarchy.

Definition 12 (Concretization Hierarchy of GM-Objects)**.** A concretization hierarchy of a set of gm-objects O is an acyclic relation $\mathscr{H}_O \subseteq O \times O$. If $(o, \bar{o}) \in \mathscr{H}_O$, then o is a direct concretization of $\bar{o}$. If $(o, \bar{o}) \in \mathscr{H}_O^+$ and $(o, \bar{o}) \notin \mathscr{H}_O$, then o is an indirect concretization of $\bar{o}$.

The class of descendant gm-objects of a gm-object o at level l is denoted $o\,\langle l \rangle$. In particular, in this work we are interested in consistent concretization hierarchies. In the case of level hierarchies, which are not in total order, but in partial order, we especially demand that each attribute and level is induced at exactly one gm-object in order to avoid conflicts due to multiple inheritance.

Definition 13 (Consistent Concretization Hierarchy of GM-Objects)**.** A concretization hierarchy $\mathscr{H}_O$ of a set of gm-object O is consistent, iff

1. Each $o \in O$ is a gm-object according to Definition 2 resp. Definition 3.
2. For each pair of gm-objects $(o, \bar{o}) \in \mathscr{H}_O$, o is a consistent concretization of $\bar{o}$ according to Definition 7 resp. Definition 8.
3. Each attribute and level is introduced at only one gm-object:
   (a) (Unique Induction Rule for Attribute) If $a \in (A_{\bar{o}} \cap A_o)$ and there exists a gm-object $\tilde{o} \in O$ such that $(o, \tilde{o}) \in \mathscr{H}_O^*$, $(\tilde{o}, \bar{o}) \in \mathscr{H}_O^*$ then $a \in A_{\tilde{o}}$.
   (b) (Unique Induction Rule for Levels) If $l \in (L_{\bar{o}} \cap L_o)$ and there exists a gm-object $\tilde{o} \in O$ such that $(o, \tilde{o}) \in \mathscr{H}_O^*$, $(\tilde{o}, \bar{o}) \in \mathscr{H}_O^*$ then $l \in L_{\tilde{o}}$.

4. If a gm-object $\bar{o}$ with top level l is a direct or indirect concretization of a gm-object o, where $(l, \bar{l}) \in P_o$, then $\bar{o}$ must concretize a gm-object $\hat{o}$ with top-level $\bar{l}$.

Remark 14. Apparently, hetero-homogeneous hierarchies, as described in section 1.3, can be modeled by means of such consistent concretization hierarchies. Here, we confine to hierarchies with a single root node, which may not be the case in the general setting.

Because of the unique induction rule for levels and the level order compatibility, levels in a concretization hierarchy are partially ordered implicitly. A level $\bar{l} \in L$ is called a descendant of $l \in L$, written as $\bar{l} \prec l$, if there is a gm-object $o \in O$ in which $\bar{l}$ is a descendant of l. In addition, we write $\bar{l} \preceq l$ if $\bar{l}$ is either descendant of or equal to l.

Remark 15. With regard to m-objects, introduced in Neumayr et al. [2009a], and gm-objects parent relations P_o and concretization hierarchies $\mathscr{H}_O$ imply certain types of graphs. For m-objects

- the concretization hierarchy $\mathscr{H}_O$ is a directed acyclic graph or, especially, tree with partial order (and[4] one root node), and
- P_o^* is a directed acyclic graph with total order.

Whereas for gm-objects

- the concretization hierarchy $\mathscr{H}_O$ is a directed acyclic graph with partial order (and[5] one root node), and
- P_o^* is a directed acyclic graph with partial order.

Example 16. Let us consider a consistent concretization hierarchy with level references as depicted in figure 2.7, which is represented by the set of the gm-objects $O = \{$Location, Austria, Alps, Salzburg$\}$. It can easily be checked that all conditions of Definition 13 are satisfied.

[4]This fact must be ensured by an additional assumption.

[5]For gm-objects also a further assumption must be placed to have only a single root node.

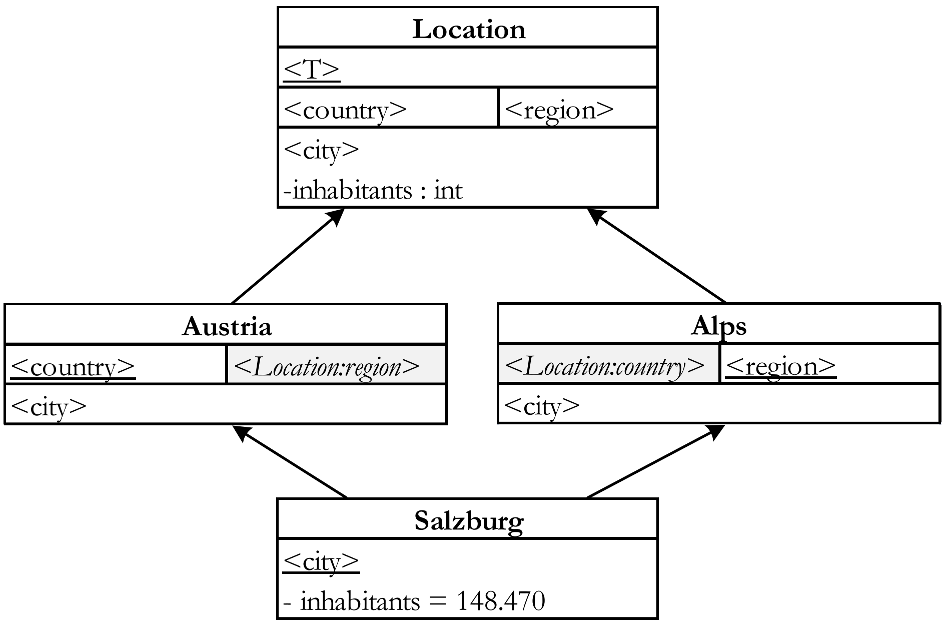

Figure 2.7: Consistent gm-object concretization hierarchy with level references

Example 17. Let us consider a consistent concretization hierarchy without level references as depicted in figure 2.8, which is represented by the set of gm-objects O = {Location, Austria, Alps, Salzburg}. Again, it can easily be verified that all conditions of Definition 13 are satisfied.

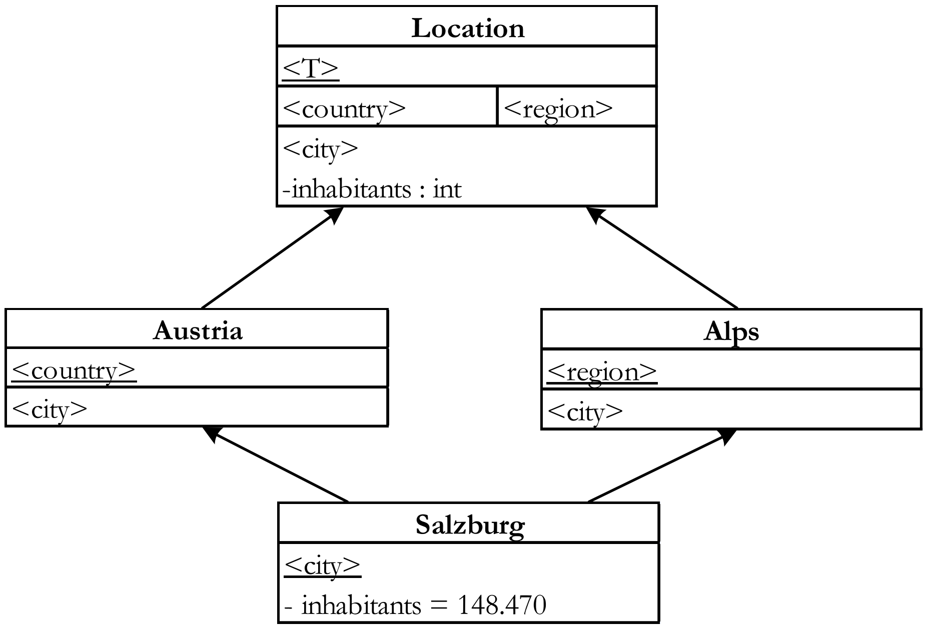

Figure 2.8: Consistent gm-object concretization hierarchy without level references

2.2 Generalized Multi-Level Relationships

In this section the notions of a gm-relationship, a consistent (individual) concretization of a gm-relationship, and a consistent concretization hierarchy of gm-relationships are defined.

2.2.1 Definition of GM-Relationship

Gm-relationships are analogous to gm-objects since they describe relationships between gm-objects at multiple levels of abstraction. In addition, similar to gm-objects they can even be arranged at different levels of abstraction in concretization hierarchies. Thus, a gm-relationship represents different abstraction levels of a relationship, which connects gm-objects at the respective levels, and it implies extensional constraints for its concretizations at multiple levels. Gm-relationships can cope with hetero-homogeneous hierarchies and, moreover, they can be applied for querying and navigation.

The proposed approach covers n-ary m-relationships, which can optionally be described by attributes. It is worth mentioning that in Neumayr et al. [2010] measures are associated with m-relationships in the context of data warehouses, hence, which have a different meaning compared to attributes.

Definition 18 (GM-Relationship). A gm-relationship

$$r = (O_r, C_r, A_r, c_r, d_r, v_r, n_r)$$

relates a sequence of gm-objects $O_r = (o_1, \ldots, o_m)$, which also defines its coordinate (denoted by coord(r)), and has a set of connection-levels $C_r \subset L_O = L_{o_1} \times \cdots \times L_{o_m}$. Its top-connection-level $\hat{l}_r$ is implicitly given by the top-levels of the referenced gm-objects, i.e., $\hat{l}_r := \left(\hat{l}_{o_1}, \ldots, \hat{l}_{o_m}\right) \in L_O$. Each gm-object o_i, $i = 1, \ldots, n$, is associated with a label, which is defined by the total function *name*

$$n_r : [1, m] \to LB\,,$$

having the one-to-one property and where LB is an universe of labels. Each attribute $a \in A_r$ is described by a connection-level, which is defined by the total function *connection*

$$c_r : A_r \to C_r; a \mapsto l \in C_r\,,$$

its domain, which is defined by the function *domain*

$$d_r : A_r \to D; a \mapsto d \in D\,,$$

where D is an universe of data types and, optionally, by a value from its domain, which is defined by the partial function *value*

$$v_r : A_r \to V; a \mapsto v \in V\,,$$

where V is an universe of data values. The attributes of the top-connection-level $\hat{l}_r$ are specified entirely by values of their domain.

Remark 19. The label $n_r(i)$ of a gm-object o_i is typically the name of the root node in the associated concretization hierarchy. If two or more gm-objects share the same root node, i.e., they belong to the same concretization hierarchy, different names have to be introduced. Otherwise, the function n_r will not be injective. This labeling approach (for gm-relationships) is required since a gm-object can be related in several different manners to other gm-objects within the same gm-relationship, see, e.g., Example 23, where a *Person* might be a *Seller* and a *Buyer* simultaneously. This technique is equivalent to approaches for relational databases, see, e.g., Codd [1970].

Remark 20. In contrast to Neumayr et al. [2010] we do not suppose that each elements o_i of a coordinate $O = (o_1, \ldots, o_m)$ belongs to a separate concretization hierarchy.

Example 21. Let us consider the gm-relationships *producedBy* and *designedBy* as depicted in figure 2.9. They involve the gm-objects *Product* and *Company* resp. *Product* and *Person,*

$$\begin{aligned} O_{\text{producedBy}} &= (\text{Product}, \text{Company}) \, , \\ O_{\text{designedBy}} &= (\text{Product}, \text{Person}) \, , \end{aligned}$$

with labels

$$\begin{aligned} n_{\text{producedBy}}(1) &= \text{"Product"}, \\ n_{\text{producedBy}}(2) &= \text{"Company"}, \\ n_{\text{designedBy}}(1) &= \text{"Product"}, \\ n_{\text{designedBy}}(2) &= \text{"Person"}, \end{aligned}$$

and (multiple) connection-levels

$$\begin{aligned} C_{\text{producedBy}} &= \{(\text{category}, \text{industrial sector}) \, , \\ &\qquad (\text{model}, \text{enterprise}) \, , \\ &\qquad (\text{physical entity}, \text{factory})\} \, , \\ C_{\text{designedBy}} &= \{(\text{model}, \text{individual})\} \, . \end{aligned}$$

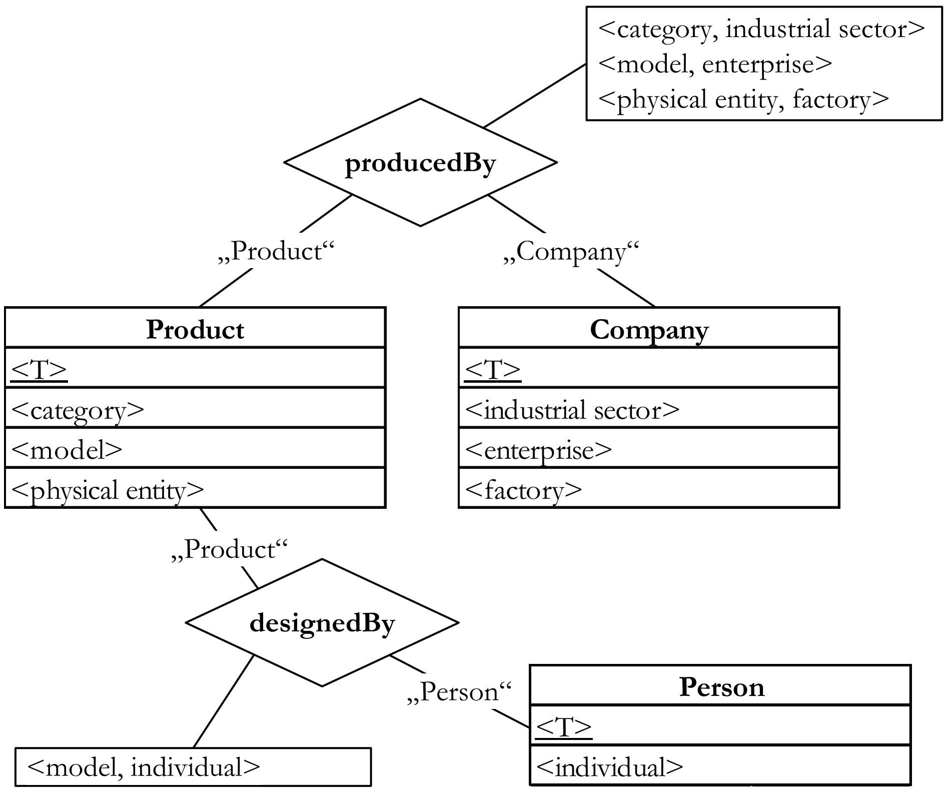

Figure 2.9: Gm-relationships *producedBy* and *designedBy*

Example 22. Let us consider a gm-relationship *sales* as depicted in figure 2.10. It links the gm-objects *Product*, *Time* and *Location*,

$$O_{\text{sales}} = (\text{Product}, \text{Time}, \text{Location}) ,$$

with labels

$$\begin{aligned} n_{\text{sales}}(1) &= \text{"Product"}, \\ n_{\text{sales}}(2) &= \text{"Time"}, \\ n_{\text{sales}}(3) &= \text{"Location"}, \end{aligned}$$

has connection-level *<model, month, city>*

$$C_{\text{sales}} = \{(\text{model}, \text{month}, \text{city})\} ,$$

and defines the attribute *discount,*

$$\begin{aligned} c_{\text{sales}}(\text{discount}) &= (\text{model}, \text{month}, \text{city}) , \\ d_{\text{sales}}(\text{discount}) &= \text{Float} . \end{aligned}$$

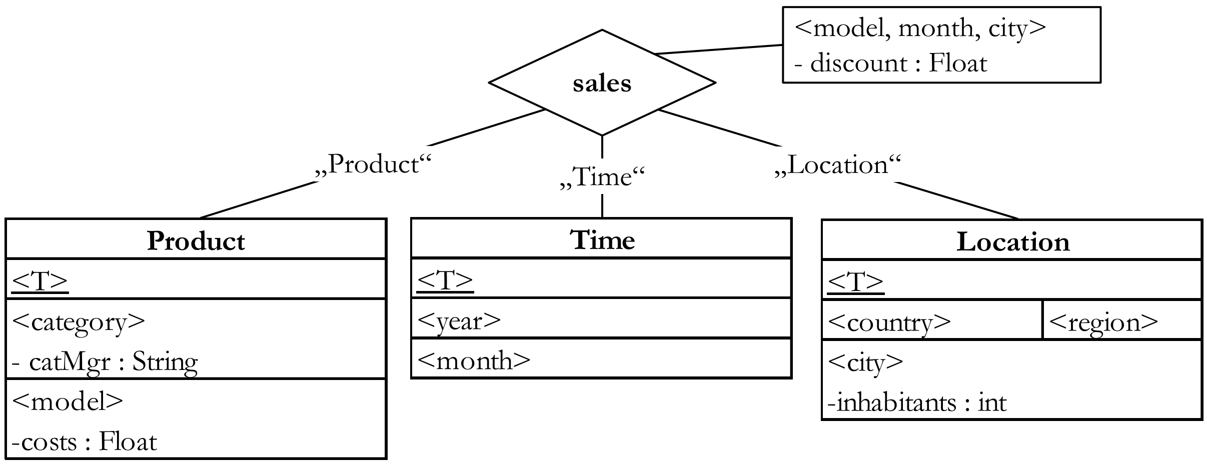

Figure 2.10: Gm-relationship *sales*

Example 23. Let us consider a gm-relationship *order* as depicted in figure 2.11. It links the gm-objects *Product*, *Time* and *Location*,

$$O_{\text{order}} = (\text{Product}, \text{Person}, \text{Person}) \ ,$$

with labels

$$\begin{aligned} n_{\text{order}}(1) &= \text{"Product"}, \\ n_{\text{order}}(2) &= \text{"Seller"}, \\ n_{\text{order}}(3) &= \text{"Buyer"}, \end{aligned}$$

has the connection-level *<model, individual, individual>*,

$$C_{\text{order}} = \{(\text{model}, \text{individual}, \text{individual})\}$$

and defines the attribute *minPrice,*

$$\begin{aligned} c_{\text{order}}(\text{minPrice}) &= (\text{model}, \text{individual}, \text{individual}) \ , \\ d_{\text{order}}(\text{minPrice}) &= \text{Float} \, . \end{aligned}$$

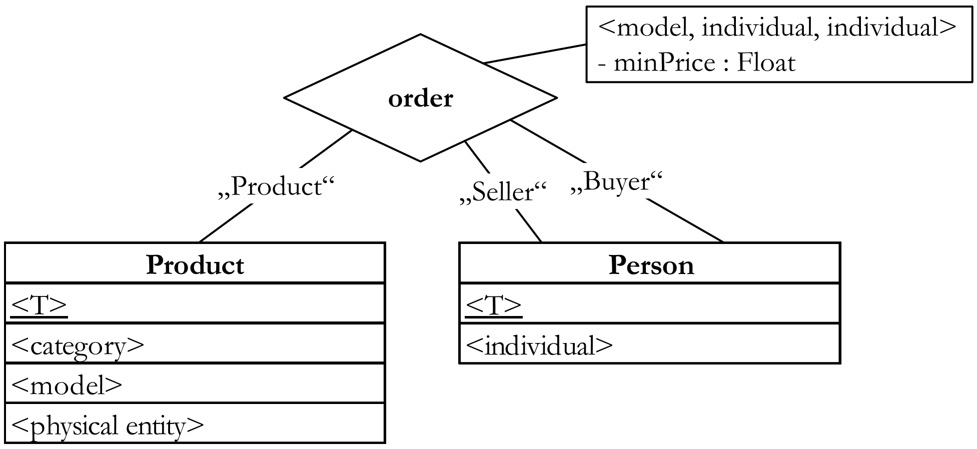

Figure 2.11: Gm-relationship *order*

2.2.2 Concretizations

Gm-relationships can be concretized like gm-objects. If at least one linked gm-object is substituted by a (direct or indirect) descendant gm-object, a gm-relationship is concretized in its coordinate. The concretize-relationship between two m-relationships states instantiation and/or specialization. The descendant gm-relationship must provide values for the attributes at its top-connection-level and may add additional connection-levels and attributes.

Hence, first of all we require a suitable (partial) ordering for connection-levels and coordinates of gm-relationships. Let us suppose a set of gm-objects O and connection-levels $l_r = (l_1, \dots, l_m), \bar{l}_r = (\bar{l}_1, \dots, \bar{l}_m) \in L_O$. $\bar{l}_r$ is a descendant of l_r, written $\bar{l}_r \preceq l_r$, iff $\bar{l}_i \preceq l_i$ for $i = 1, \dots, m$. Let us suppose two coordinates (of gm-relationships) $O = (o_1, \dots, o_m)$ and $\bar{O} = (\bar{o}_1, \dots, \bar{o}_m)$, whose elements belong to concretization hierarchies $\mathcal{H}_i$. The coordinate $\bar{O}$ is a descendant of or equal to coordinate O, written $\bar{O} \preceq O$, iff $(\bar{o}_i, o_i) \in \mathcal{H}_i^*$ for $i = 1, \dots, m$. The coordinate $\bar{O}$ is a proper descendant of O, written as $\bar{O} \prec O$, iff o_i is a descendant of or equal to o_i for $i = 1, \dots, m$ and at least one gm-object $\bar{o}_i$ concretizes o_i.

The following definition for a consistent (individual) concretization of a gm-relationship is applied in the diploma thesis.

Definition 24 (Consistent Concretization of GM-Relationships)**.** A gm-relationship $\bar{r} = (O_{\bar{r}}, C_{\bar{r}}, A_{\bar{r}}, c_{\bar{r}}, d_{\bar{r}}, v_{\bar{r}})$ is a consistent concretization of another gm-relationship $r = (O_r, C_r, A_r, c_r, d_r, v_r)$ iff

1. $O_{\bar{r}} \prec O_r$.
2. (Connection-Level Containment) Every connection-level with base-level that is below or equal the top-level of $\bar{r}$ is also a connection-level of $\bar{r}$, i.e.,

$$\bar{C}_r = \left\{ l_r \in C_r : l_r \preceq \hat{l}_{\bar{r}} \right\} \subseteq C_{\bar{r}},$$

and every other connection-level of r is not a connection-level of $\bar{r}$, i.e.,

$$\left\{ l_r \in C_r : l_r \preceq \hat{l}_{\bar{r}} \right\} \cap C_{\bar{r}} = \varnothing .$$

3. (Label Stability) Labels associated with gm-objects must be the same, i.e.,

$$n_r(i) = n_{\bar{r}}(i),\ i = 1, \ldots, m .$$

4. (Attribute Containment) Every attribute a of r related to a connection-level in $\bar{C}_r$ is also a attribute of $\bar{r}$, i.e.,

$$\left\{ a \in A_r : c_r(a) \in \bar{C}_r \right\} \subseteq A_{\bar{r}} ,$$

and every other attribute of r is not an attribute of $\bar{r}$, i.e.,

$$\left\{ a \in A_r : c_r(a) \notin \bar{C}_r \right\} \cap A_{\bar{r}} = \varnothing .$$

5. Common attributes are associated with the same connection-level, have the same domain, and the same value, if defined, i.e., for $a \in \left(A_r \cap A_{\bar{r}}\right)$ it follows

   (a) (Stability of Attribute Connection-Level) $c_r(a) = c_{\bar{r}}(a)$,

   (b) (Stability of Attribute Domains) $d_r(a) = d_{\bar{r}}(a)$,

   (c) (Compatibility of Attribute Values) If $v_r(a)$ is defined, it follows $v_r(a) = v_{\bar{r}}(a)$.

Example 25. Let us consider a gm-relationship *sales-HarryPotter4-Feb.09-Salzburg* between the gm-objects *HarryPotter4*, *Feb.09* and *Salzburg as* depicted in figure 2.12. It concretizes the gm-relationship *sales-Product-Time-Location* between the gm-objects *Product*, *Time* and *Location* and its top-connection-level *<model, month, city>*, i.e., it defines a value for the attribute *discount*.

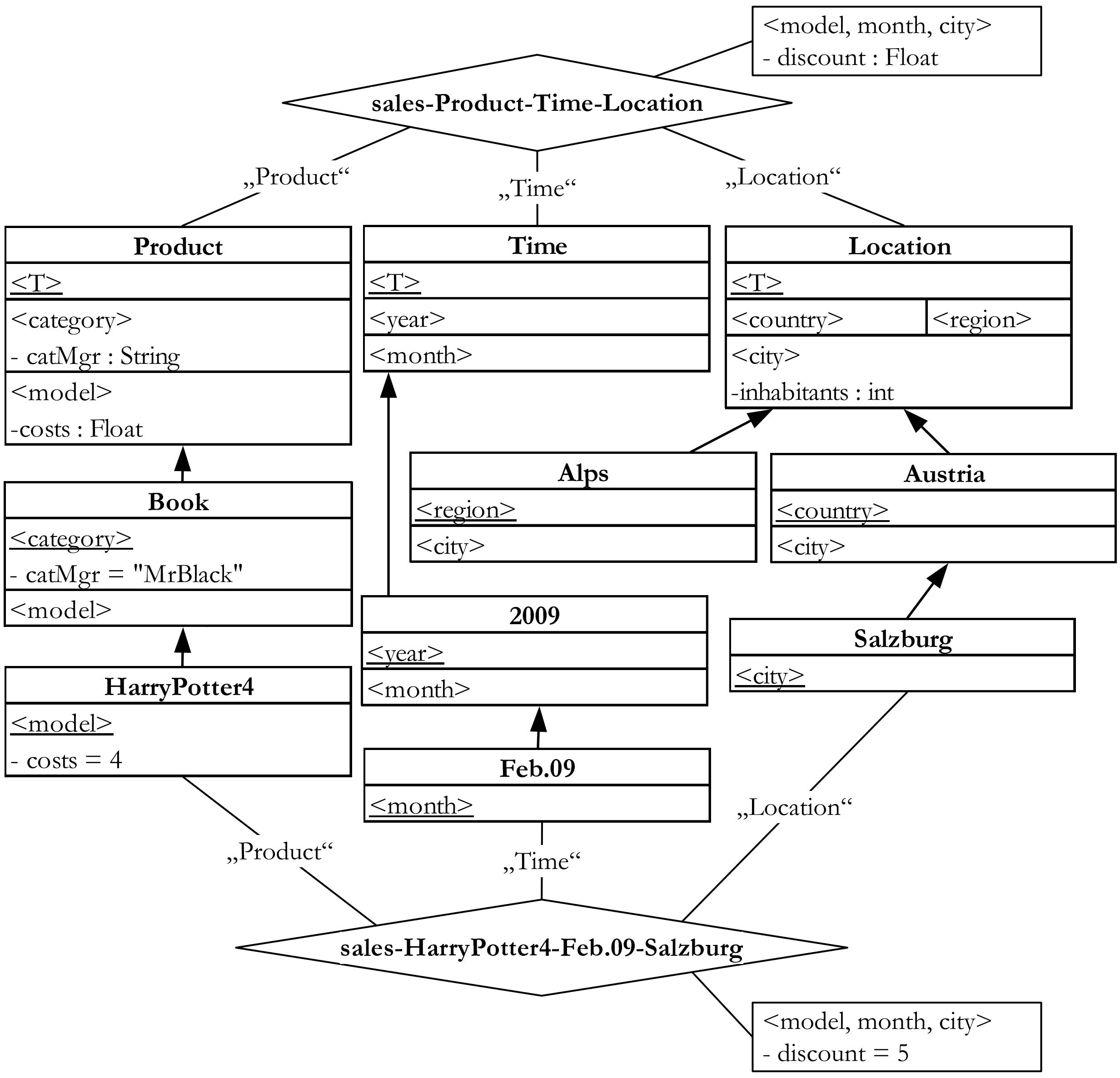

Figure 2.12: Concretization of a gm-relationship

Example 26. Let us consider a gm-relationship *sales-Car-2009-Switzerland* between the gm-objects *Car*, *2009* and *Switzerland as* depicted in figure 2.13. It concretizes the gm-relationship *sales-Product-Time-Location* between the gm-objects *Product*, *Time* and *Location* by introducing new connection-levels and attributes. I.e., the top-connection-level becomes *<category, year, country>* with the attribute *minPrice* and defined value, and the connection-level *<brand, month, city>* is introduced with attribute *minQty* and defined domain as well as defined value.

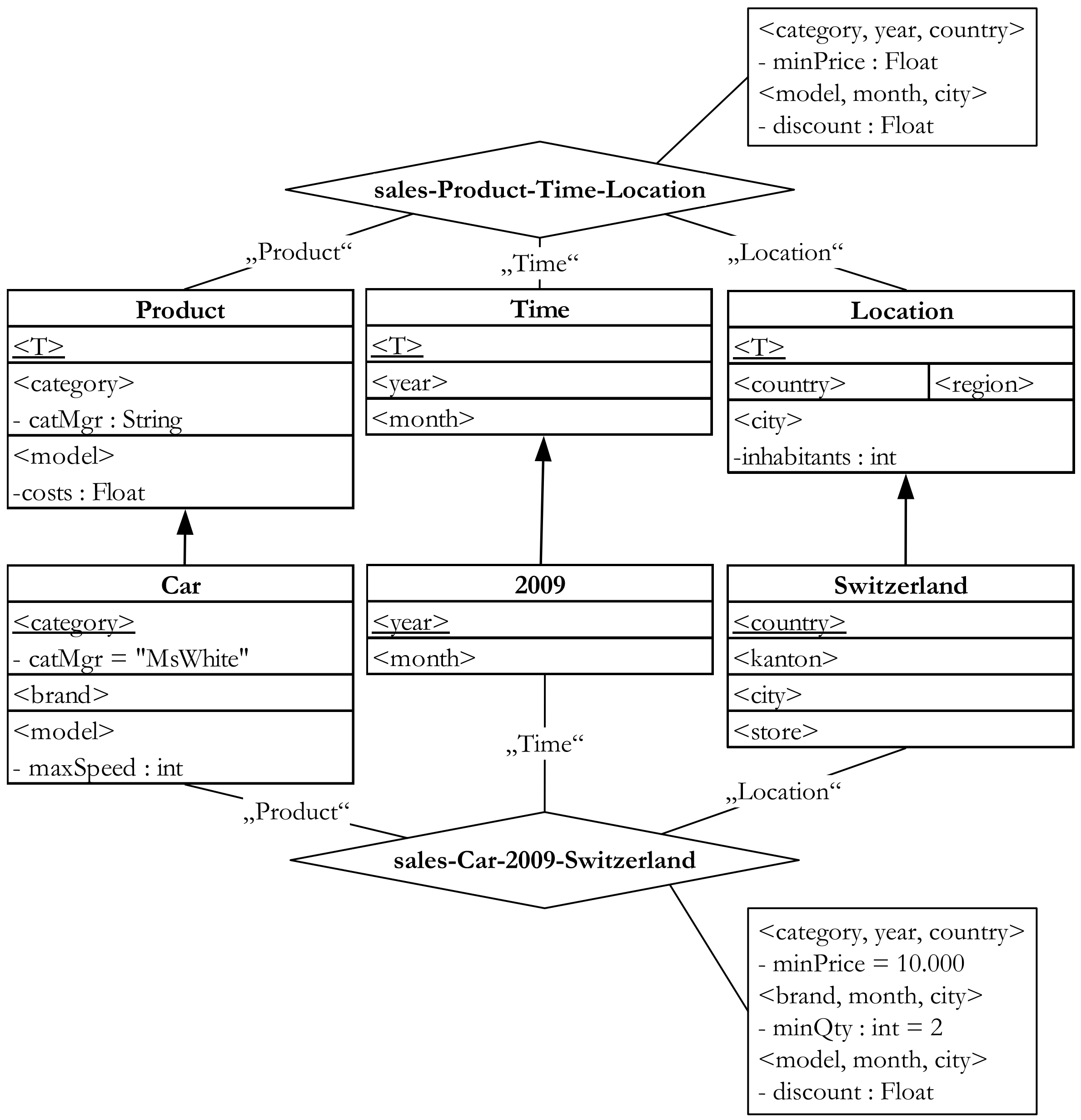

Figure 2.13: Concretization of a gm-relationship with new connection-level and attributes

If connection-levels are allowed to be moved to a more specific connection-level, we obtain the following extended definition of a consistent concretization of gm-relationships.

Definition 27 (Extended Consistent Concretization of GM-Relationships)**.** A gm-relationship $\bar{r} = (O_{\bar{r}}, C_{\bar{r}}, A_{\bar{r}}, c_{\bar{r}}, d_{\bar{r}}, v_{\bar{r}})$ is a consistent concretization of another gm-relationship $r = (O_r,$ $C_r, A_r, c_r, d_r, v_r)$ iff

1. $O_{\bar{r}} \prec O_r$.

2. (Connection-Level Containment) Every connection-level with base-level that is below or equal the top-level of $\bar{r}$ is also a connection-level of $\bar{r}$, i.e.,

$$\bar{C}_r = \left\{ l_r \in C_r : l_r \preceq \hat{l}_{\bar{r}} \right\} \subseteq C_{\bar{r}},$$

and every other connection-level of r is not a connection-level of $\bar{r}$, i.e.,

$$\left\{ l_r \in C_r : l_r \preceq \hat{l}_{\bar{r}} \right\} \cap C_{\bar{r}} = \emptyset.$$

3. (Attribute Containment) Every attribute a of r related to a connection-level in $\bar{C}_r$ is also a attribute of $\bar{r}$, i.e.,

$$\left\{ a \in A_r : c_r(a) \in \bar{C}_r \right\} \subseteq A_{\bar{r}},$$

and every other attribute of r is not an attribute of $\bar{r}$, i.e.,

$$\left\{ a \in A_r : c_r(a) \notin \bar{C}_r \right\} \cap A_{\bar{r}} = \emptyset.$$

4. (Assured Granularity) For each connection-level l shared by r and $\bar{r}$ the base-level of l at $\bar{r}$ is the same or below the base-level of l at r, i.e.

$$\forall l \in \left(C_r \cap C_{\bar{r}} \right) : c_{\bar{r}}(l) \preceq c_r(l).$$

5. Common attributes are associated with the same connection-level, have the same domain, and the same value, if defined, i.e., for $a \in \left(A_r \cap A_{\bar{r}} \right)$ it follows

   (a) (Stability of Attribute Connection-Levels) $c_r(a) = c_{\bar{r}}(a)$,

   (b) (Stability of Attribute Domains) $d_r(a) = d_{\bar{r}}(a)$,

   (c) (Compatibility of Attribute Values) If $v_r(a)$ is defined, it follows $v_r(a) = v_{\bar{r}}(a)$.

2.2.3 Concretization Hierarchies

Analogous to gm-objects gm-relationships can be organized in a hierarchy and the definition of a consistent (individual) concretization of gm-relationships directly induces to the notion of a concretization hierarchy.

Definition 28 (Concretization Hierarchy of GM-Relationships)**.** A concretization hierarchy of a set of gm-relationships R is an acyclic relation $\mathscr{H}_R \subseteq R \times R$. If $(r, \bar{r}) \in \mathscr{H}_R$, then r is a direct concretization of $\bar{r}$. If $(r, \bar{r}) \in \mathscr{H}_R^+$ and $(r, \bar{r}) \notin \mathscr{H}_R$, then r is an indirect concretization of $\bar{r}$.

In order to avoid conflicts due to multiple inheritance we confine to consistent concretization hierarchies.

Definition 29 (Consistent Concretization Hierarchy of GM-Relationships)**.** A concretization hierarchy $\mathscr{H}_R$ of a set of gm-object R is consistent, iff

1. Each $r \in R$ is a gm-relationship according to Definition 18.
2. For each pair of gm-objects $(r, \bar{r}) \in \mathscr{H}$, r is a consistent concretization of $\bar{r}$ according to Definition 24 resp. Definition 27.
3. Each attribute and level is introduced at only one gm-object:
   (a) (Unique Induction Rule for Attributes) If $a \in (A_{\bar{r}} \cap A_r)$ and there exists a gm-relationship $\tilde{r} \in R$ such that $(r, \tilde{r}) \in \mathscr{H}_R^*$, $(\tilde{r}, \bar{r}) \in \mathscr{H}_R^*$ then $a \in A_{\tilde{r}}$.
   (b) (Unique Induction Rule for Connection-Levels) If $l \in (C_{\bar{r}} \cap C_r)$ and there exists a gm-relationship $\tilde{r} \in R$ such that $(r, \tilde{r}) \in \mathscr{H}_R^*$, $(\tilde{r}, \bar{r}) \in \mathscr{H}_R^*$ then $l \in C_{\tilde{r}}$.

Remark 30. Analogue to gm-objects the concretization hierarchy $\mathscr{H}_R$ of gm-relationships is a directed acyclic graph with partial order (and[6] one root node).

Example 31. Let us consider a concretization hierarchy as depicted in figure 2.14, which is represented by the set of gm-relationships

$$\begin{aligned} R \ = \ \{&\text{order-Producer-Seller-Buyer}, \\ &\quad\text{order-Car-MrBlack-Person}, \\ &\quad\text{order-Porsche911-MrBlack-MsWhite}\}, \end{aligned}$$

where the gm-relationship *order-Car-MrBlack-Person* concretizes the gm-relationship *order-Producer-Seller-Buyer* and the gm-relationship *order-Porsche911-MrBlack-MsWhite* concretizes the gm-relationship *order-Car-MrBlack-Person*. The hierarchy is a consistent concretization hierarchy since all criteria of Definition 29 are fulfilled.

[6]This circumstance must be guaranteed by an additional assumption.

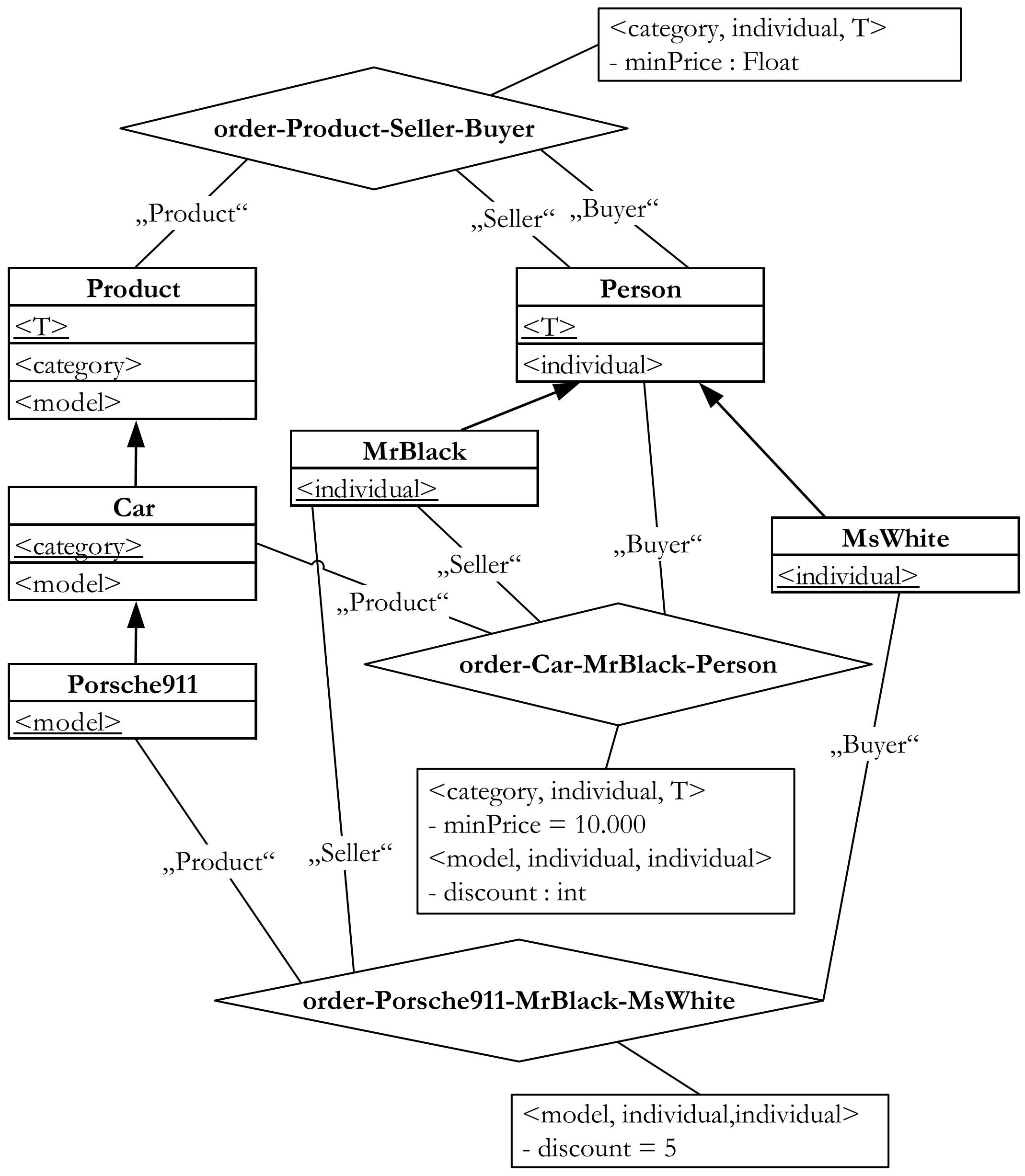

Figure 2.14: Consistent gm-relationship concretization hierarchy

3 OWL Mapping

In chapter 2 a multi-level modeling technique for hetero-homogeneous hierarchies based upon generalized multi-level objects (gm-objects) as well as generalized multi-level relationships (gm-relationships) was introduced. Supplementarily, the present chapter aims to provide a possibility to transfer the concepts and ideas from conceptual modeling to ontological engineering, where in particular we refer to OWL. In this context a semantic-preserving mapping from gm-objects and gm-relationships to the decidable fragment of OWL, which is augmented with suitable integrity constraints to take important aspects into account. Thus, we confine here to a certain OWL 2 profile, i.e., a sub-language (syntactic subset) of OWL 2, namely OWL 2 DL. The OWL mapping is restricted and motivated by the following main arguments regarding the conceptual modeling and the ontology engineering:

- The mapping should preserve the semantics of the conceptual modeling approach using gm-objects and gm-relationships.
- The mapping output should be in a form such that OWL reasoners are able to
  - execute queries at different levels of abstraction and
  - to decidably check for consistency or inconsistency for multi-level conceptual modeling, i.e., with respect to hetero-homogeneous hierarchies.
- The mapping approach should lead to a pattern for ontological meta-modeling within the decidable fragment of OWL 2. I.e., the mapping should extend the meta-modeling features of OWL 2 ("punning", see section 4.1) to objects that represent classes at multiple levels of abstraction, while remaining within the decidable and first-order fragment of OWL.

The final result of the mapping serves as the basis for ontology engineering and can optionally be combined with additional OWL axioms and ontologies. In Neumayr and Schrefl [2009] it was sketched how to transfer m-objects and m-relationships to OWL. Herein, this line of work is continued and extended to case of gm-objects and gm-objects, which were introduced in chapter 2. In addition, for certain examples the final mapping-output is provided.

Remark 32. In chapter 2 different definitions of gm-objects/gm-relationships and their consistent (individual) concretizations were provided, respectively. For the OWL mapping we confine to Definition 3, Definition 8 and Definition 13 for gm-objects, their consistent (individual) concretization and consistent concretization hierarchies, as well as Definition 18, Definition 24 and Definition 29 for gm-relationships, their consistent (individual) concretization and consistent hierarchies.

For brevity and readability Description Logic (DL) syntax[1] is applied and entities (individuals, classes and properties) are not introduced explicitly because they should be clear from the context. It is worth noting that the conceptual modeling approach based on gm-objects and gm-relationships uses the closed world assumption (CWA). Since in OWL this is not the case, i.e., we have the open world assumption (OWA), the mapping has to ensure that there are no unwanted drawbacks in form of wrong classifications. As already mentioned in section 1.1, an existing approach from Motik et al. [2007, 2009] is applied, which allows to combine open and closed world reasoning. Thereby, TBox axioms are added, which are interpreted as integrity constraints. For TBox (schema) reasoning these axioms are treated as usual axioms, whereas for ABox (data) reasoning they are treated as checks and do not derive additional information.

Remark 33. The OWL mapping procedure is provided in form of algorithms for both gm-objects and gm-relationships. Thereby, [·] denotes that a variable is substituted by its actual value. Moreover, "**assert:**" adds an OWL axiom to the mapping output. "**IC:**" denotes an OWL axiom which is interpreted as an integrity constraint.

3.1 Mapping Example

In order to illustrate the mapping procedure and its result we apply a running example, which is shown in the figures 3.1 - 3.4. It is represented by three (hetero-homogeneous) gm-object concretization hierarchies *Product* (see figure 3.1), *Time* (see figure 3.2) and *Location* (see figure 3.3) and one (hetero-homogeneous) gm-relationship concretization hierarchy *sales* (see figure 3.4), respectively. The final entire mapping result is provided in section 3.4.

Remark 34. With respect to the gm-object concretization hierarchies the concretize relationship expresses different semantics, e.g., a *materializationOf* relationship in the *Product* hierarchy.

[1]DL syntax does not differentiate between data properties and object properties.

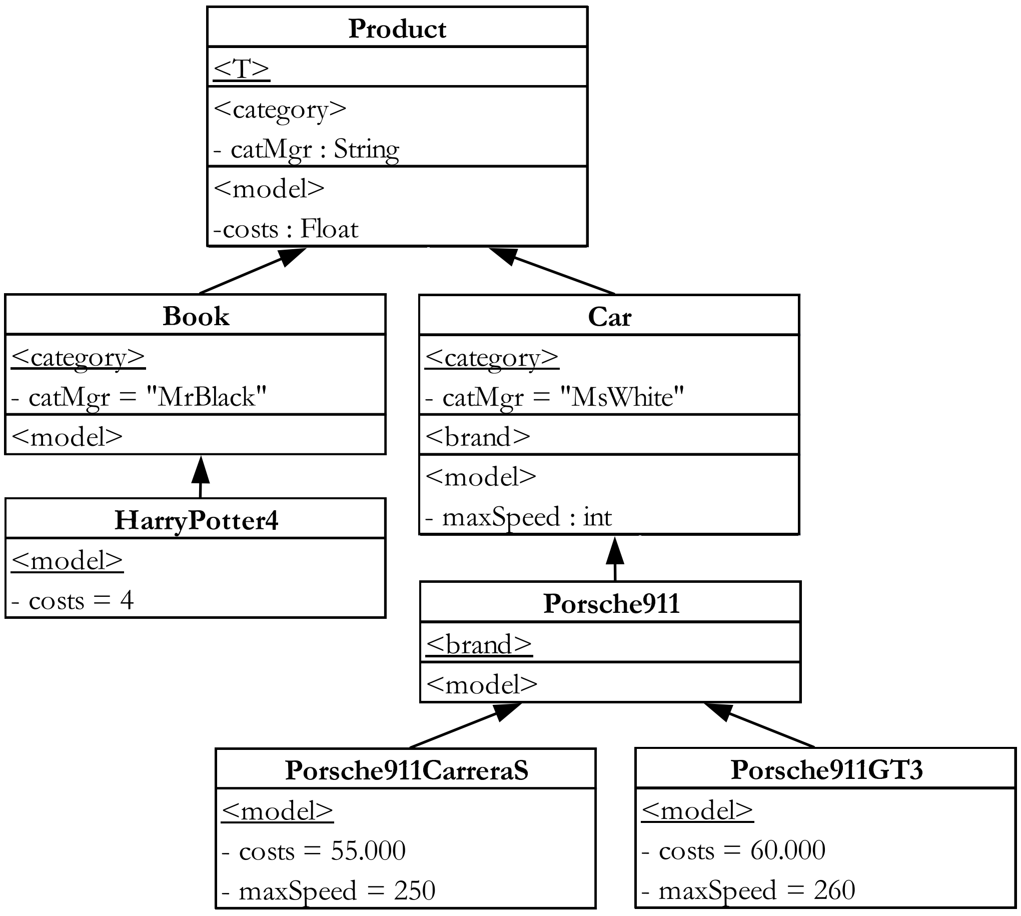

Figure 3.1: OWL mapping example 1 - gm-objects

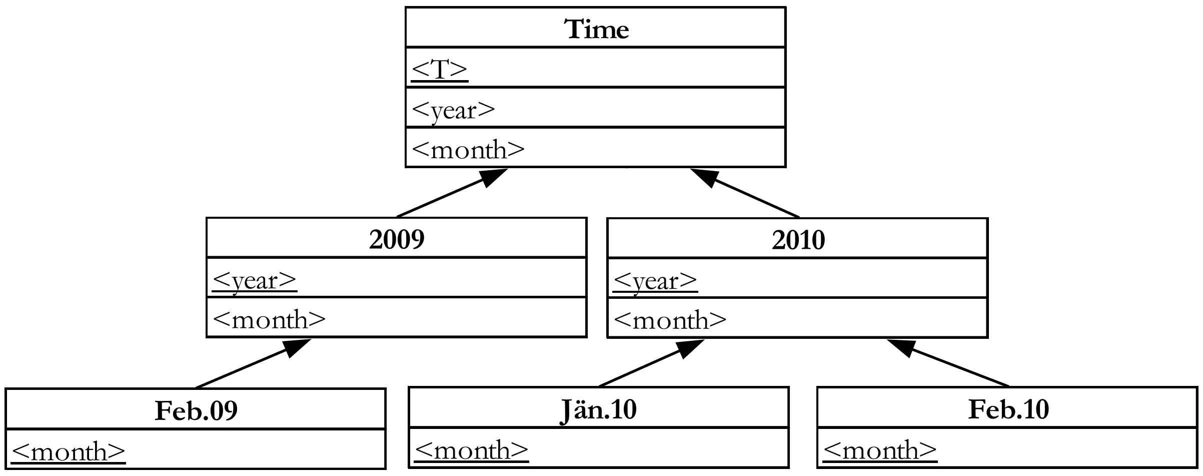

Figure 3.2: OWL mapping example 2 - gm-objects

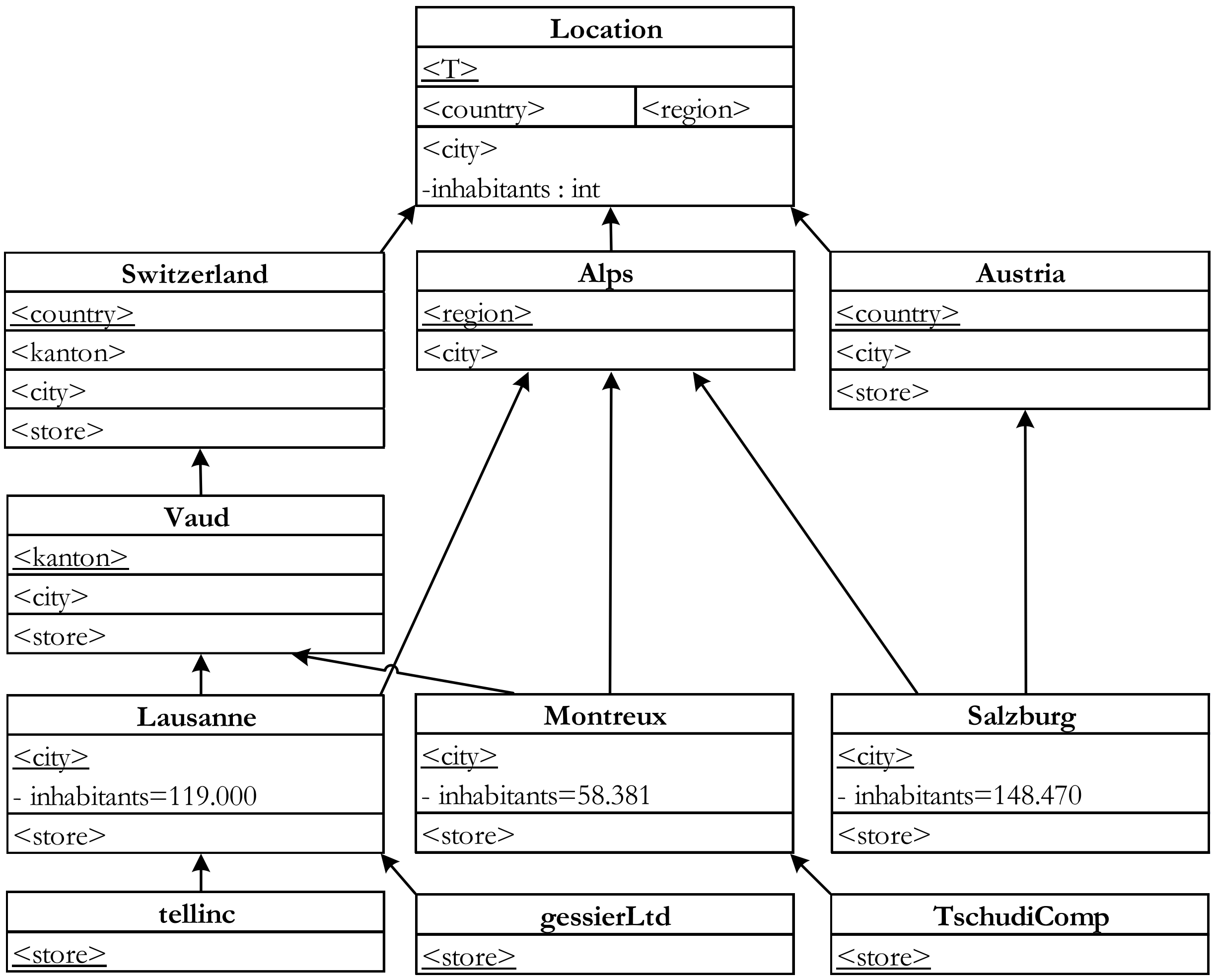

Figure 3.3: OWL mapping example 3 - gm-objects

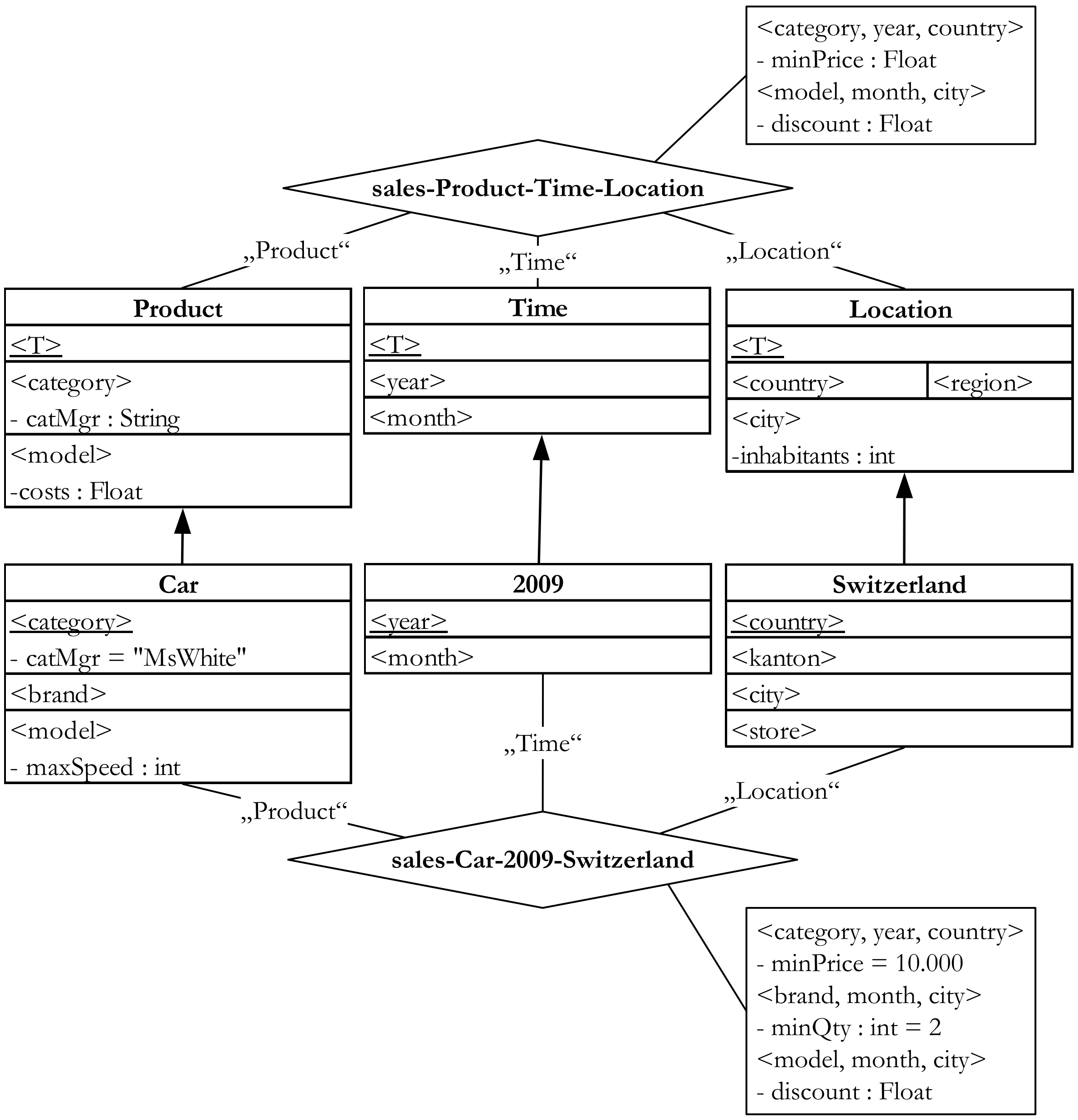

Figure 3.4: OWL mapping example - gm-relationships

3.2 Mapping Algorithm for GM-Objects

In this section each step of the mapping for gm-objects to OWL is explained. To exemplify in this section the mapping output for the gm-object *Salzburg*, see figure 3.3, is provided. The mapping procedure is summarized as algorithm, see Algorithm 1.

```
input : A set $O$ of gm-objects $o = (L_o, A_o, p_o, l_o, d_o, v_o)$, a concretization hierarchy $\mathscr{H}_O$ and
        an universe of levels $L$.
output: A set of OWL axioms.
assert: concretize $\sqsubseteq$ concretize_t
assert: concretize_t$^+$ $\sqsubseteq$ concretize_t
for all $o \in O$ do
    assert: $[\hat{l}_o]([o])$
    if $\exists \bar{o} : (o, \bar{o}) \in \mathscr{H}_O$ then
        concretize$([o], [\bar{o}])$
    end
    for all $a \in \hat{A}_o$ do
        assert: $[a]\left([o], [v_o(a)]\right)$
    end
    for all $a \in A_o \setminus \hat{A}_o$ do
        assert IC: $\exists$concretize_t$.\{[o]\} \sqcap [l_o(a)] \sqsubseteq \forall [a].[d_o(a)] \sqcap {=}1\ [a].\top$
        if $v_o(a)$ is defined then
            assert IC: $[l_o(a)] \sqcap \exists$concretize_t$.\{[o]\} \sqsubseteq \exists [a].\{[v_o(a)]\}$
        end
    end
    for all $(l, \bar{l}) \in P_o : \bar{l} \neq \hat{l}_o \wedge \left(\nexists \bar{o} \in O : (o, \bar{o}) \in \mathscr{H}_O \wedge (l, \bar{l}) \in P_{\bar{o}}\right)$ do
        assert IC: $\exists$concretize_t$.\{[o]\} \sqcap [l] \sqsubseteq \exists$concretize_t$.\{[o]\} \sqcap [\bar{l}]$
    end
    for all $a \in A_o : \nexists \bar{o} \in O : (o, \bar{o}) \in \mathscr{H}_O \wedge a \in A_{\bar{o}}$ do
        assert IC: $\exists [a].\top \sqsubseteq \exists$concretize_t$.\{[o]\} \sqcap [l_o(a)]$
    end
    for all $l \in L_o : l \neq \hat{l}_o \wedge (\nexists \bar{o} \in O : (o, \bar{o}) \in \mathscr{H}_O \wedge l \in L_{\bar{o}})$ do
        assert IC: $[l] \sqsubseteq$ concretize_t$.\{[o]\}$
    end
end
for all $l \in L, \bar{l} \in L \setminus l$ do
    $[l] \sqcap [\bar{l}] \sqsubseteq \bot$
end
for all $o \in O, \bar{o} \in O \setminus o$ do
    $[o] \not\approx [\bar{o}]$
end
```

Algorithm 1: Mapping GM-Objects to OWL

Gm-objects and concretizations

In order to map concretization hierarchies, which are composed of gm-objects, to OWL each gm-object is represented as individual, e.g., *Salzburg*, and each abstraction level as primitive class, e.g., *city*. Moreover, each gm-object is assigned to an abstraction level by a class assertion (see Algorithm 1, line 4). The level of abstraction corresponds to the top level of the gm-object. In addition, each gm-object is connected to its parent gm-objects, if they exist, by applying a property concretize (see Algorithm 1, line 5-7). Here the gm-object *Salzburg* is associated to the individual *Salzburg*, which is a member of the class *city* and concretizes the individuals *Alps* and *Austria*.

```
city(Salzburg)
concretize(Salzburg,Alps)
concretize(Salzburg,Austria)
```

Top-level attributes

An attribute must have a value from its domain if the level of the attribute is equivalent to the top-level of the gm-object. Property assertions are applied to handle values of top-level attributes since those are attributes, which describe the gm-object itself (see Algorithm 1, line 8-10). For example, city *Salzburg* has a number of inhabitants.

```
inhabitants(Salzburg,148.470)
```

Levels of abstraction and navigation

A level of a gm-object is interpreted as a container for all direct or indirect concretizations of the gm-object at the respective level. That is why the mapping considers any level of abstraction as class. Such an approach has many advantages, because it allows

- to define common characteristics of the class members,
- to define additional constraints,
- to provide entry points for queries.

The class of individuals $o\langle l\rangle$ that belong to the abstraction level l and are direct or indirect concretizations of individual o correspond to the class expression

$$\boxed{\exists \text{concretize\_t}.\{[o]\} \sqcap [l]}$$

where concretize_t is defined as transitive super-property of concretize. For example, the class of all *Location cities*, *Location<city>* corresponds to the class expression

$$\boxed{\exists \text{concretize\_t}.\{\text{Location}\} \sqcap \text{city}}$$

Subsumption hierarchies

A concretization hierarchy induces several subsumption hierarchies, i.e., there is one subsumption hierarchy for each level of abstraction. For example, from

$$\boxed{\text{concretize(Salzburg,Austria)}}$$

an OWL reasoner should conclude that the term

$$\boxed{\exists \text{concretize\_t}.\{\text{Austria}\} \sqcap \text{store}}$$

subsumes

$$\boxed{\exists \text{concretize\_t}.\{\text{Salzburg}\} \sqcap \text{store}}$$

This particular feature (of OWL) allows for consistency checks as well as inheritance between classes.

Common characteristics

Class axioms are used to define common characteristics for members of a particular level of a gm-object. Attributes are represented by data properties with values and number restrictions (see Algorithm 1, line 11-16). Values of attributes could be shared with gm-objects at lower levels (see Algorithm 1, line 11-16). The axioms are interpreted as integrity constraints. Otherwise semantics of the gm-object approach is lost. For example, the container *Location<city>* has a number of inhabitants of type Integer.

$$\boxed{\text{IC: } \exists \text{concretize\_t}.\{\text{Location}\} \sqcap \text{city} \sqsubseteq \forall \text{inhabitants.Integer} \sqcap = 1\ \text{inhabitants}.\top}$$

A level l of a gm-object o guarantees that concretizations of o at lower levels also concretize a concretization of o at the level l (see Algorithm 1, line 17-19). These integrity constraints allow stable upward navigation and enhance heterogeneous level hierarchies since we allow that gm-objects can introduce new levels of abstraction. It is worth mentioning that the new levels are valid for descendants of the corresponding gm-object, hence, which are not valid for descendants of other gm-objects. For example, all *Switzerland cities* belong to a *Switzerland kanton* since the gm-object introduces the new abstraction level *kanton.*

$$\text{IC: } \exists \text{concretize\_t.} \{\text{Switzerland}\} \sqcap \text{city} \sqsubseteq \exists \text{concretize\_t.} (\exists \text{concretize\_t.} \{\text{Switzerland}\} \sqcap \text{kanton})$$

Each attribute is allowed to be introduced exactly at one level of one gm-object (see Algorithm 1, line 20-22) and each level is allowed to be introduced exactly at one gm-object (see Algorithm 1, line 23-25).

Gm-objects and levels

The mapping has to guarantee that a gm-object belongs to maximum one abstraction level. Thus, all levels are supposed to be pairwise disjoint (see Algorithm 1, line 27-29). For example, an individual at level *country* cannot be at the level *city*.

$$\text{country} \sqcap \text{city} \sqsubseteq \bot$$

Unique name assumption

To confine to the unique name assumption we state that each pair of gm-objects refers to different individuals (see Algorithm 1, line 30-32). For example, *Salzburg* and *Montreux* are different individuals.

$$\text{Salzburg} \not\approx \text{Montreux}$$

Main extension in comparison to m-object mapping approach

For the mapping of gm-objects there is a main difference with respect to the approach of Neumayr and Schrefl [2009] restricted to m-objects. Since gm-objects can concretize a set of gm-objects in hetero-homogeneous systems we do not have

~~$\top \sqsubseteq$ 1 concretize~~

3.3 Mapping Algorithm for GM-Relationships

For the mapping of gm-relationships there are actually two different approaches available. On the one-hand gm-relationships can be represented as properties (*property approach*) or as individuals (*objectification approach*) in OWL. But property assertions do not have identifiers and so it is impossible to directly represent concretization links. If we follow the property approach each connection-level of a gm-relationship is represented as property and redundantly a concretization-link between two gm-relationships is represented by several sub-property-axioms. Thus, in order to avoid any redundancy we propose the objectification approach, similar to Neumayr and Schrefl [2009], which maps gm-relationships to individuals and allows to directly represent concretization links between gm-objects in contrast to the previous mapping approach.

Similar to the mapping of gm-objects each step of the mapping for gm-relationships to OWL is explained. In order to exemplify the mapping, in this section the mapping output for the gm-relationship *sales* is provided. The mapping procedure is summarized as algorithm, see Algorithms 2 - 3.

Gm-relationships and concretizations

Because of the objectification approach each gm-relationship is represented as individual, which is linked to its parent gm-relationships, if they exist, and to its gm-objects by property assertions. Thereby, the property *concretize* and sub-properties of (the property) *relatedTo* are used, respectively (see Algorithm 2, line 4-10). The gm-relationship *sales* between the gm-objects *Car*, *2009* and *Switzerland*, named *sales-Car-2009-Switzerland*, concretizes the gm-relationship *sales* between the gm-objects *Product*, *Time* and *Location*, named *sales-Product-Time-Location*. Thus, we have

concretize(sales-Car-2009-Switzerland,sales-Product-Time-Location)
relatedToProduct(sales-Car-2009-Switzerland,Car)
relatedToProduct $\sqsubseteq$ relatedTo
relatedToTime(sales-Car-2009-Switzerland,2009)
relatedToTime $\sqsubseteq$ relatedTo
relatedToLocation(sales-Car-2009-Switzerland,Switzerland)
relatedToLocation $\sqsubseteq$ relatedTo

A gm-relationship constrains the relatives (see Algorithm 2, line 11-19). At least one relative of a (direct or indirect) concretization of the gm-relationship *sales-Car-2009-Switzerland* must concretize a relative of the gm-relationship *sales-Car-2009-Switzerland*.

```
input : A set $R$ of gm-relationships $r = (O_r, C_r, A_r, c_r, d_r, v_r)$ and a concretization
        hierarchy $\mathscr{H}_R$.
output: A set of OWL axioms.
assert: concretize $\sqsubseteq$ contretize_t
assert: concretize_t$^+$ $\sqsubseteq$ contretize_t
for all $r \in R$ do
    if $\exists \bar{r} : (r, \bar{r}) \in \mathscr{H}_R$ then
        assert: concretize$([r], [\bar{r}])$
    end
    for $i = 1, \ldots, m$ do
        assert: relatedTo$[n_r(i)]\,([r], [o_i])$
        assert: relatedTo$[n_r(i)] \sqsubseteq$ relatedTo
    end
    aux = „$\bot$“
    for all $i = 1, \ldots, m$ do
        aux = aux + „$\sqcup$ ($\forall$relatedTo$[n_r(i)]$. ($\exists$concretize_t.$\{[o_i]\}$) “
        for all $j = 1, \ldots, i-1, i+1, \ldots, m$ do
            aux = aux + „$\sqcap$ ($\forall$relatedTo$[n_r(j)]$. ($\exists$concretize_t.$\{[o_j]\} \sqcup \{[o_j]\}$))“
        end
        aux = aux + „)“
    end
    assert IC: $\exists$concretize_t.$\{[r]\} \sqsubseteq$ [aux]
    for all $a \in A_r : c_r(a) = \hat{l}_r$ do
        assert: $[a]\,([r], [v_r(a)])$
    end
    for all $a \in A_r : c_r(a) = (l_1, \ldots, l_m) \neq \hat{l}_r$ do
        aux = „$\exists$concretize_t.$\{[r]\} \sqcap$“
        aux = aux + „$\exists$relatedTo$[n_r(1)].[l_1] \sqcap \ldots \sqcap \exists$relatedTo$[n_r(m)].[l_m]$“
        assert IC: [aux] $\sqsubseteq \forall[a].[d_r(a)] \sqcap = 1\,[a].\top$
        if $v_r(a)$ is defined then
            [aux] $\sqsubseteq \exists[a].\{[v_r(a)]\}$
        end
    end
end
```

Algorithm 2: Mapping GM-Relationships to OWL - Part I

```
input  : A set $R$ of gm-relationships $r = (O_r, C_r, A_r, c_r, d_r, v_r)$ and a concretization
         hierarchy $\mathscr{H}_R$.
output: A set of OWL axioms.
for all $r \in R$ do
    for all $l = (l_1, \ldots, l_m) \in C_r : l \neq \hat{l}_r$ do
        assert IC: $\exists$concretize_t.$\{[r]\} \ldots$
        assert IC: $\ldots \sqcap (\exists$relatedTo$[n_r(1)].(\exists$concretize_t.$[l_1] \sqcup [l_1]) \ldots$
        assert IC: $\ldots \sqcup \ldots \sqcup \exists$relatedTo$[n_r(m)].(\exists$concretize_t.$[l_m] \sqcup [l_m])) \ldots$
        assert IC: $\ldots \sqsubseteq \exists$concretize_t.$( \ldots$
        assert IC: $\ldots \exists$concretize_t.$\{[r]\} \sqcap \ldots$
        assert IC: $\ldots \exists$relatedTo$[n_r(1)].[l_1] \sqcap \ldots \sqcap \exists$relatedTo$[n_r(m)].[l_m]) \ldots$
        assert IC: $\ldots \sqcup (\exists$relatedTo$[n_r(1)].[l_1] \sqcap \ldots \sqcap \exists$relatedTo$[n_r(m)].[l_m])$
    end
    for all $a \in A_r : \nexists \bar{r} \in R : (r, \bar{r}) \in \mathscr{H}_R \wedge a \in A_{\bar{r}}$ do
        aux = „$\exists$concretize_t.$\{[r]\} \sqcap$“
        aux = aux + „$\exists$relatedTo$[n_r(1)].[l_1] \sqcap \ldots \sqcap \exists$relatedTo$[n_r(m)].[l_m]$“
        assert IC: $\exists[a].\top \sqsubseteq [\text{aux}]$
    end
    for all $l = (l_1, \ldots, l_m) \in C_r : l \neq \hat{l}_r \wedge (\nexists \bar{r} \in R : (r, \bar{r}) \in \mathscr{H}_R \wedge l \in C_{\bar{r}})$ do
        assert IC: $(\exists$relatedTo$[n_r(1)].[l_1] \sqcap \ldots \sqcap \exists$relatedTo$[n_r(m)].[l_m]) \ldots$
        assert IC: $\ldots \sqsubseteq$ concretize_t.$\{[r]\}$
    end
end
for all $r \in R, \bar{r} \in R \setminus r$ do
    $[r] \not\approx [\bar{r}]$
end
```

Algorithm 3: Mapping GM-Relationships to OWL - Part II

$\text{IC: } \exists \text{concretize\_t.}\{\text{sales-Car-2009-Switzerland}\} \sqsubseteq \ldots$
$\ldots (\forall \text{relatedToProduct.}(\exists \text{concretize\_t.}\{\text{Car}\}) \ldots$
$\ldots \sqcap \forall \text{relatedToTime.}(\exists \text{concretize\_t.}\{2009\} \sqcup \{2009\}) \ldots$
$\ldots \sqcap \forall \text{relatedToLocation.}(\exists \text{concretize\_t.}\{\text{Switzerland}\} \sqcup \{\text{Switzerland}\}))\ldots$
$\ldots \sqcup \ldots$
$\ldots (\forall \text{relatedToProduct.}(\exists \text{concretize\_t.}\{\text{Car}\} \sqcup \{\text{Car}\}) \ldots$
$\ldots \sqcap \forall \text{relatedToTime.}(\exists \text{concretize\_t.}\{2009\}) \ldots$
$\ldots \sqcap \forall \text{relatedToLocation.}(\exists \text{concretize\_t.}\{\text{Switzerland}\} \sqcup \{\text{Switzerland}\}))\ldots$
$\ldots \sqcup \ldots$
$\ldots (\forall \text{relatedToProduct.}(\exists \text{concretize\_t.}\{\text{Car}\} \sqcup \{\text{Car}\}) \ldots$
$\ldots \sqcap \forall \text{relatedToTime.}(\exists \text{concretize\_t.}\{2009\} \sqcup \{2009\}) \ldots$
$\ldots \sqcap \forall \text{relatedToLocation.}(\exists \text{concretize\_t.}\{\text{Switzerland}\}))\ldots$

Top-connection-level attributes

An attribute must have a value from its domain if the connection-level of the attribute is equivalent to the top-connection-level of the gm-relationship, otherwise it is optional. Property assertions are applied to handle values of attributes since those are attributes, which describe the gm-relationship itself (see Algorithm 2, line 20-22). For example, the gm-relationship *sales-Car-2009-Switzerland* has a minimal price, i.e.,

minPrice(sales-Car-2009-Switzerland,10.000)

Common characteristics

Class axioms are used to define common characteristics for members of a particular connection-level of a gm-relationship. Attributes are represented by data properties with values and number restrictions (see Algorithm 2, line 23-30). Values of attributes could be shared with gm-relationships at lower levels (see Algorithm 2, line 23-30). The axioms are interpreted as integrity constraints. Otherwise semantics of the gm-relationships approach is lost. For example, *sales-Product-Time-Location* at connection level *<category, year, country>* has an attribute *minPrice* of type *Float*.

$\text{IC: } \exists \text{concretize\_t.}\{\text{sales-Product-Time-Location}\} \ldots$
$\ldots \sqcap \exists \text{relatedToProduct.category} \sqcap \ldots$
$\ldots \exists \text{relatedToTime.year} \sqcap \ldots$
$\ldots \exists \text{relatedToLocation.country} \ldots$
$\ldots \sqsubseteq \forall \text{minPrice.Float} \sqcap = 1\ \text{minPrice.}\top$

Each attribute is allowed to be introduced exactly at one connection-level of one gm-relationship (see Algorithm 3, line 11-15) and each connection-level is allowed to be introduced exactly at one relationship (see Algorithm 3, line 16-19).

Connection levels

For abstraction levels of gm-objects safe upward navigation must be guaranteed, whereas for connection-levels of gm-relationships safe navigation along gm-relationships at higher level must be ensured. Constraints concerning connection-levels are considered in form of integrity constraints. A connection-level $l = (l_1, \ldots, l_n) \in C_r$ of a gm-relationship r ensures that concretizations of r at levels below l concretize a gm-relationship at level l that concretize r. The axioms are also interpreted as integrity constraints (see Algorithm 3, line 2-10).

IC: $\exists$concretize_t.$\{$sales-Product-Time-Location$\}$ $\ldots$
$\ldots$ $\sqcap$ ($\exists$relatedToProduct.($\exists$concretize_t.category $\sqcup$ category) $\ldots$
$\ldots$ $\sqcup$ $\exists$relatedToTime.($\exists$concretize_t.year $\sqcup$ year) $\ldots$
$\ldots$ $\sqcup$ $\exists$relatedToLocation.($\exists$concretize_t.country $\sqcup$ country)) $\ldots$
$\ldots$ $\sqsubseteq$ $\exists$concretize_t.($\exists$concretize_t.$\{$sales-Product-Time-Location$\}$ $\ldots$
$\ldots$ $\sqcap$ $\exists$relatedToProduct.category $\ldots$
$\ldots$ $\sqcap$ $\exists$relatedToTime.year $\ldots$
$\ldots$ $\sqcap$ $\exists$relatedToLocation.country) $\ldots$
$\ldots$ $\sqcup$($\exists$relatedToProduct.category $\ldots$
$\ldots$ $\sqcap$ $\exists$relatedToTime.year $\ldots$
$\ldots$ $\sqcap$ $\exists$relatedToLocation.country)

Unique name assumption

To confine to the unique name assumption we state that each pair of gm-relationships refers to different individuals (see Algorithm 3, line 21-23). For example, *sales-Product-Time-Location* and *sales-Car-2009-Switzerland* are different individuals.

sales-Product-Time-Location $\not\approx$ sales-Car-2009-Switzerland

Navigation

Gm-relationships can be queried using OWL reasoners. Thereby, navigation along gm-relationships can be used in terms of class expressions. In order to query for cars, which were sold in Switzerland in the year 2009, we ask for the members of the class

$$\begin{aligned}&\text{Car} \sqcap \exists\text{relatedToProduct}^{-}.((\exists\text{concretize\_t}.\{\text{sales-Product-Time-Location}\} \sqcup \\ &\{\text{sales-Product-Time-Location}\})\ \ldots \\ &\ldots \sqcap \exists\text{relatedToTime}.(\exists\text{concretize\_t}^{-}.\{2009\} \sqcup \{2009\})\ \ldots \\ &\ldots \sqcap \exists\text{relatedToLocation}.(\exists\text{concretize\_t}^{-}.\{\text{Switzerland}\} \sqcup \{\text{Switzerland}\}))\end{aligned}$$

Main extension in comparison to m-object mapping approach

For the mapping of gm-relationships there are significant differences with respect to the approach of Neumayr and Schrefl [2009] restricted to m-objects. A gm-relationships can concretize a set of gm-relationships in hetero-homogeneous systems. Thus, we do not have

$$\cancel{\top \sqsubseteq \leq 1\ \text{concretize}}$$

Moreover, attributes are included, which describe the gm-relationships themselves.

3.4 Entire Mapping Result

Let us illustrate the mapping procedure for gm-objects and gm-relationships by presenting the full mapping result of the running example.

3.4.1 GM-Object Concretization Hierarchy *Product*

The mapping result of the gm-object concretization hierarchy *Product* is given by the following mapping outputs.

$$\begin{aligned}&\text{concretize} \sqsubseteq \text{concretize\_t} \\ &\text{concretize\_t}^{+} \sqsubseteq \text{concretize\_t}\end{aligned}$$

Mapping Output 4: General Axioms

T(Product)
IC: $\exists$concretize_t.$\{$Product$\} \sqcap$ category $\sqsubseteq \forall$catMgr.String $\sqcap = 1$ catMgr.$\top$
IC: $\exists$concretize_t.$\{$Product$\} \sqcap$ model $\sqsubseteq \forall$costs.Float $\sqcap = 1$ costs.$\top$
IC:
$\exists$concretize_t.$\{$Product$\} \sqcap$ model $\sqsubseteq \exists$concretize_t.$(\exists$concretize_t.$\{$Product$\} \sqcap$ category$)$
IC: $\exists$catMgr.$\top \sqsubseteq (\exists$concretize_t.$\{$Product$\} \sqcup \{$Product$\}) \sqcap$ category
IC: $\exists$costs.$\top \sqsubseteq (\exists$concretize_t.$\{$Product$\} \sqcup \{$Product$\}) \sqcap$ model
IC: category $\sqsubseteq \exists$concretize_t.$\{$T$\}$
IC: model $\sqsubseteq \exists$concretize_t.$\{$T$\}$

Mapping Output 5: GM-Object *Product*

category(Book)
concretize(Book,Product)
catMgr(Book,„MrBlack“)

Mapping Output 6: GM-Object *Book*

country(Car)
concretize(Car,Product)
catMgr(Book,„MsWhite “)
IC: $\exists$concretize_t.$\{$Car$\} \sqcap$ model $\sqsubseteq \forall$maxSpeed.Integer $\sqcap = 1$ maxSpeed.$\top$
IC: $\exists$concretize_t.$\{$Car$\} \sqcap$ model $\sqsubseteq \exists$concretize_t.$(\exists$concretize_t.$\{$Car$\} \sqcap$ brand$)$
IC: $\exists$maxSpeed.$\top \sqsubseteq (\exists$concretize_t.$\{Car\} \sqcup \{Car\}) \sqcap$ model
IC: brand $\sqsubseteq \exists$concretize_t.$\{$Car$\}$

Mapping Output 7: GM-Object *Car*

model(HarryPotter4)
concretize(HarryPotter4,Book)
costs(HarryPotter,4)

Mapping Output 8: GM-Object *HarryPotter4*

brand(Porsche911)
concretize(Porsche911,Car)

Mapping Output 9: GM-Object *Porsche911*

```
model(Porsche911CarreraS)
concretize(Porsche911CarreraS,Prosche911)
costs(Porsche911CarreraS,55.000)
maxSpeed(Porsche911CarreraS,250)
```

Mapping Output 10: GM-Object *Porsche911CarreraS*

```
model(Porsche911GT3)
concretize(Porsche911GT3,Prosche911)
costs(Porsche911GT3,60.000)
maxSpeed(Porsche911GT3,260)
```

Mapping Output 11: GM-Object *Porsche911GT3*

$T \sqcap category \sqsubseteq \bot$
$T \sqcap brand \sqsubseteq \bot$
$T \sqcap model \sqsubseteq \bot$
$category \sqcap T \sqsubseteq \bot$
$category \sqcap brand \sqsubseteq \bot$
$category \sqcap model \sqsubseteq \bot$
$brand \sqcap T \sqsubseteq \bot$
$brand \sqcap category \sqsubseteq \bot$
$brand \sqcap model \sqsubseteq \bot$
$model \sqcap T \sqsubseteq \bot$
$model \sqcap category \sqsubseteq \bot$
$model \sqcap brand \sqsubseteq \bot$

Mapping Output 12: Disjoint Levels

$Porsche911 \not\approx Product$
$Porsche911 \not\approx Book$
$Porsche911 \not\approx Car$
$Porsche911 \not\approx HarryPotter4$
$Porsche911 \not\approx Porsche911CarreraS$
$Porsche911 \not\approx Porsche911GT3$

Mapping Output 13: Unique Name Assumption (only for *Porsche911*)

3.4.2 GM-Object Concretization Hierarchy *Time*

The mapping result of the gm-object concretization hierarchy *Time* is given by the following mapping outputs.

concretize $\sqsubseteq$ concretize_t
concretize_t^{+} $\sqsubseteq$ concretize_t

Mapping Output 14: General Axioms

T(Time)
IC: $\exists$concretize_t.{Time} $\sqcap$ month $\sqsubseteq$ $\exists$concretize_t.($\exists$concretize_t.{Time} $\sqcap$ year)
IC: year $\sqsubseteq$ $\exists$concretize_t.{T}
IC: month $\sqsubseteq$ $\exists$concretize_t.{T}

Mapping Output 15: GM-Object *Time*

year(2009)
concretize(2009,Time)

Mapping Output 16: GM-Object *2009*

year(2010)
concretize(2010,Time)

Mapping Output 17: GM-Object *2010*

month(Feb.09)
concretize(Feb.09,2009)

Mapping Output 18: GM-Object *Feb.09*

month(Jän.10)
concretize(Jän.10,2010)

Mapping Output 19: GM-Object *Jän.10*

```
month(Feb.10)
concretize(Feb.10,2010)
```

Mapping Output 20: GM-Object *Feb.10*

$\mathrm{T} \sqcap \mathrm{year} \sqsubseteq \bot$
$\mathrm{T} \sqcap \mathrm{month} \sqsubseteq \bot$
$\mathrm{year} \sqcap \mathrm{T} \sqsubseteq \bot$
$\mathrm{year} \sqcap \mathrm{month} \sqsubseteq \bot$
$\mathrm{month} \sqcap \mathrm{T} \sqsubseteq \bot$
$\mathrm{month} \sqcap \mathrm{year} \sqsubseteq \bot$

Mapping Output 21: Disjoint Levels

$\mathrm{Feb.09} \not\approx \mathrm{Time}$
$\mathrm{Feb.09} \not\approx 2009$
$\mathrm{Feb.09} \not\approx 2010$
$\mathrm{Feb.09} \not\approx \mathrm{Jän.10}$
$\mathrm{Feb.09} \not\approx \mathrm{Feb.10}$

Mapping Output 22: Unique Name Assumption (only for *Feb.09*)

3.4.3 GM-Object Concretization Hierarchy *Location*

The mapping result of the gm-object concretization hierarchy *Location* is given by the following mapping outputs.

$\mathrm{concretize} \sqsubseteq \mathrm{concretize\_t}$
$\mathrm{concretize\_t}^{+} \sqsubseteq \mathrm{concretize\_t}$

Mapping Output 23: General Axioms

$\top$(Location)
IC: $\exists$concretize_t.$\{$Location$\}\sqcap$city $\sqsubseteq$ $\forall$inhabitants.Integer $\sqcap = 1$ inhabitants.$\top$
IC: $\exists$concretize_t.$\{$Location$\}\sqcap$city $\sqsubseteq$ $\exists$concretize_t.($\exists$concretize_t.$\{$Location$\}\sqcap$country)
IC: $\exists$concretize_t.$\{$Location$\}\sqcap$city $\sqsubseteq$ $\exists$concretize_t.($\exists$concretize_t.$\{$Location$\}\sqcap$region)
IC: $\exists$inhabitants.$\top$ $\sqsubseteq$ ($\exists$concretize_t.$\{$Location$\}\sqcup\{$Location$\})\sqcap$city
IC: country $\sqsubseteq\exists$concretize_t.$\{$T$\}$
IC: region $\sqsubseteq\exists$concretize_t.$\{$T$\}$
IC: city $\sqsubseteq\exists$concretize_t.$\{$T$\}$

Mapping Output 24: GM-Object *Location*

country(Switzerland)
concretize(Switzerland,Location)
IC:
$\exists$concretize_t.$\{$Switzerland$\}\sqcap$city $\sqsubseteq\exists$concretize_t.($\exists$concretize_t.$\{$Switzerland$\}\sqcap$kanton)
IC: kanton $\sqsubseteq\exists$concretize_t.$\{$Switzerland$\}$

Mapping Output 25: GM-Object *Switzerland*

region(Alps)
concretize(Alps,Location)

Mapping Output 26: GM-Object *Alps*

country(Austria)
concretize(Austria,Location)

Mapping Output 27: GM-Object *Austria*

kanton(Vaud)
concretize(Vaud,Switzerland)

Mapping Output 28: GM-Object *Vaud*

city(Lausanne)
concretize(Lausanne,Vaud)
concretize(Lausanne,Alps)

Mapping Output 29: GM-Object *Lausanne*

```
city(Montreux)
concretize(Montreux,Vaud)
concretize(Montreux,Alps)
```

Mapping Output 30: GM-Object *Montreux*

```
city(Salzburg)
concretize(Salzburg,Alps)
concretize(Salzburg,Austria)
```

Mapping Output 31: GM-Object *Salzburg*

```
store(tellInc)
concretize(tellInc,Lausanne)
```

Mapping Output 32: GM-Object *tellInc*

```
store(gessierLtd)
concretize(gessierLtd,Lausanne)
```

Mapping Output 33: GM-Object *gessierLtd*

```
store(TschudiComp)
concretize(TschudiComp,Montreux)
```

Mapping Output 34: GM-Object *TschudiComp*

$T \sqcap \text{country} \sqsubseteq \bot$
$T \sqcap \text{region} \sqsubseteq \bot$
$T \sqcap \text{kanton} \sqsubseteq \bot$
$T \sqcap \text{city} \sqsubseteq \bot$
$\text{country} \sqcap T \sqsubseteq \bot$
$\text{country} \sqcap \text{region} \sqsubseteq \bot$
$\text{country} \sqcap \text{kanton} \sqsubseteq \bot$
$\text{country} \sqcap \text{city} \sqsubseteq \bot$
$\text{region} \sqcap T \sqsubseteq \bot$
$\text{region} \sqcap \text{country} \sqsubseteq \bot$
$\text{region} \sqcap \text{kanton} \sqsubseteq \bot$
$\text{region} \sqcap \text{city} \sqsubseteq \bot$
$\text{kanton} \sqcap T \sqsubseteq \bot$
$\text{kanton} \sqcap \text{country} \sqsubseteq \bot$
$\text{kanton} \sqcap \text{region} \sqsubseteq \bot$
$\text{kanton} \sqcap \text{city} \sqsubseteq \bot$
$\text{city} \sqcap T \sqsubseteq \bot$
$\text{city} \sqcap \text{country} \sqsubseteq \bot$
$\text{city} \sqcap \text{region} \sqsubseteq \bot$
$\text{city} \sqcap \text{kanton} \sqsubseteq \bot$

Mapping Output 35: Disjoint Levels

$\text{Salzburg} \not\approx \text{Location}$
$\text{Salzburg} \not\approx \text{Switzerland}$
$\text{Salzburg} \not\approx \text{Alps}$
$\text{Salzburg} \not\approx \text{Salzburg}$
$\text{Salzburg} \not\approx \text{Vaud}$
$\text{Salzburg} \not\approx \text{Lausanne}$
$\text{Salzburg} \not\approx \text{Montreux}$
$\text{Salzburg} \not\approx \text{tellInc}$
$\text{Salzburg} \not\approx \text{gessierInc}$
$\text{Salzburg} \not\approx \text{TschudiComp}$

Mapping Output 36: Unique Name Assumption (only for *Salzburg*)

3.4.4 GM-Relationship Concretization Hierarchy *sales*

The mapping result of the gm-relationship concretization hierarchy *sales* is given by the following mapping outputs.

concretize $\sqsubseteq$ concretize_t
concretize_t$^+$ $\sqsubseteq$ concretize_t

Mapping Output 37: General Axioms

relatedToProduct(sales-Product-Time-Location,Product)
relatedToProduct $\sqsubseteq$ relatedTo
relatedToTime(sales-Product-Time-Location,Time)
relatedToTime $\sqsubseteq$ relatedTo
relatedToLocation(sales-Product-Time-Location,Location)
relatedToLocation $\sqsubseteq$ relatedTo
IC: $\exists$concretize_t.$\{$sales-Product-Time-Location$\}$ $\sqsubseteq$ …
… ($\forall$relatedToProduct.($\exists$concretize_t.$\{$Product$\}$) …
… $\sqcap$ $\forall$relatedToTime.($\exists$concretize_t.$\{$Time$\}$ $\sqcup$ $\{$Time$\}$) …
… $\sqcap$ $\forall$relatedToLocation.($\exists$concretize_t.$\{$Location$\}$ $\sqcup$ $\{$Location$\}$))…
… $\sqcup$ …
… ($\forall$relatedToProduct.($\exists$concretize_t.$\{$Product$\}$ $\sqcup$ $\{$Product$\}$) …
… $\sqcap$ $\forall$relatedToTime.($\exists$concretize_t.$\{$Time$\}$) …
… $\sqcap$ $\forall$relatedToLocation.($\exists$concretize_t.$\{$Location$\}$ $\sqcup$ $\{$Location$\}$))…
… $\sqcup$ …
… ($\forall$relatedToProduct.($\exists$concretize_t.$\{$Product$\}$ $\sqcup$ $\{$Product$\}$) …
… $\sqcap$ $\forall$relatedToTime.($\exists$concretize_t.$\{$Time$\}$ $\sqcup$ $\{$Time$\}$) …
… $\sqcap$ $\forall$relatedToLocation.($\exists$concretize_t.$\{$Location$\}$))…
IC: $\exists$concretize_t.$\{$sales-Product-Time-Location$\}$ …
… $\sqcap$ $\exists$relatedToProduct.category $\sqcap$ …
… $\exists$relatedToTime.year $\sqcap$ …
… $\exists$relatedToLocation.country …
… $\sqsubseteq$ $\forall$minPrice.Float $\sqcap = 1$ minPrice.$\top$
IC: $\exists$concretize_t.$\{$sales-Product-Time-Location$\}$ …
… $\sqcap$ $\exists$relatedToProduct.model $\sqcap$ …
… $\exists$relatedToTime.month $\sqcap$ …
… $\exists$relatedToLocation.city …
… $\sqsubseteq$ $\forall$discount.Float $\sqcap = 1$ discount.$\top$

Mapping Output 38: GM-Relationship *sales-Product-Time-Location* - Part I

IC: $\exists \text{concretize\_t}.\{\text{sales-Product-Time-Location}\} \ldots$
$\ldots \sqcap (\exists \text{relatedToProduct}.(\exists \text{concretize\_t.category} \sqcup \text{category}) \ldots$
$\ldots \sqcup \exists \text{relatedToTime}.(\exists \text{concretize\_t.year} \sqcup \text{year}) \ldots$
$\ldots \sqcup \exists \text{relatedToLocation}.(\exists \text{concretize\_t.country} \sqcup \text{country})) \ldots$
$\ldots \sqsubseteq \exists \text{concretize\_t}.(\exists \text{concretize\_t}.\{\text{sales-Product-Time-Location}\} \ldots$
$\ldots \sqcap \exists \text{relatedToProduct.category} \ldots$
$\ldots \sqcap \exists \text{relatedToTime.year} \ldots$
$\ldots \sqcap \exists \text{relatedToLocation.country}) \ldots$
$\ldots \sqcup (\exists \text{relatedToProduct.category} \ldots$
$\ldots \sqcap \exists \text{relatedToTime.year} \ldots$
$\ldots \sqcap \exists \text{relatedToLocation.country})$
IC: $\exists \text{concretize\_t}.\{\text{sales-Product-Time-Location}\} \ldots$
$\ldots \sqcap (\exists \text{relatedToProduct}.(\exists \text{concretize\_t.model} \sqcup \text{model}) \ldots$
$\ldots \sqcup \exists \text{relatedToTime}.(\exists \text{concretize\_t.month} \sqcup \text{month}) \ldots$
$\ldots \sqcup \exists \text{relatedToLocation}.(\exists \text{concretize\_t.city} \sqcup \text{city})) \ldots$
$\ldots \sqsubseteq \exists \text{concretize\_t}.(\exists \text{concretize\_t}.\{\text{sales-Product-Time-Location}\} \ldots$
$\ldots \sqcap \exists \text{relatedToProduct.model} \ldots$
$\ldots \sqcap \exists \text{relatedToTime.month} \ldots$
$\ldots \sqcap \exists \text{relatedToLocation.city}) \ldots$
$\ldots \sqcup (\exists \text{relatedToProduct.model} \ldots$
$\ldots \sqcap \exists \text{relatedToTime.month} \ldots$
$\ldots \sqcap \exists \text{relatedToLocation.city})$
$\exists \text{minPrice}.\top \sqsubseteq \exists \text{concretize\_t}.\{\text{sales-Product-Time-Location}\} \ldots$
$\ldots \sqcap \exists \text{relatedToProduct.category} \sqcap \ldots$
$\ldots \exists \text{relatedToTime.year} \sqcap \ldots$
$\ldots \exists \text{relatedToLocation.country} \ldots$
$\exists \text{discount}.\top \sqsubseteq \exists \text{concretize\_t}.\{\text{sales-Product-Time-Location}\} \ldots$
$\ldots \sqcap \exists \text{relatedToProduct.model} \sqcap \ldots$
$\ldots \exists \text{relatedToTime.month} \sqcap \ldots$
$\ldots \exists \text{relatedToLocation.city} \ldots$
$\exists \text{relatedToProduct.category} \sqcap \ldots$
$\ldots \exists \text{relatedToTime.year} \sqcap \ldots$
$\ldots \exists \text{relatedToLocation.country} \ldots$
$\ldots \sqsubseteq \exists \text{concretize\_t}.\{\text{sales-Product-Time-Location}\}$
$\exists \text{relatedToProduct.model} \sqcap \ldots$
$\ldots \exists \text{relatedToTime.month} \sqcap \ldots$
$\ldots \exists \text{relatedToLocation.city} \ldots$
$\ldots \sqsubseteq \exists \text{concretize\_t}.\{\text{sales-Product-Time-Location}\}$

Mapping Output 39: GM-Relationship *sales-Product-Time-Location* - Part II

concretize(sales-Car-2009-Switzerland,sales-Product-Time-Location)
relatedToProduct(sales-Car-2009-Switzerland,Car)
relatedToProduct $\sqsubseteq$ relatedTo
relatedToTime(sales-Car-2009-Switzerland,2009)
relatedToTime $\sqsubseteq$ relatedTo
relatedToLocation(sales-Car-2009-Switzerland,Switzerland)
relatedToLocation $\sqsubseteq$ relatedTo
IC: $\exists$concretize_t.$\{$sales-Car-2009-Switzerland$\}$ $\sqsubseteq \ldots$
$\ldots$ ($\forall$relatedToProduct.($\exists$concretize_t.$\{$Car$\}$) $\ldots$
$\ldots \sqcap$ $\forall$relatedToTime.($\exists$concretize_t.$\{2009\} \sqcup \{2009\}$) $\ldots$
$\ldots \sqcap$ $\forall$relatedToLocation.($\exists$concretize_t.$\{$Switzerland$\} \sqcup \{$Switzerland$\}$))$\ldots$
$\ldots \sqcup \ldots$
$\ldots$ ($\forall$relatedToProduct.($\exists$concretize_t.$\{$Car$\} \sqcup \{$Car$\}$) $\ldots$
$\ldots \sqcap$ $\forall$relatedToTime.($\exists$concretize_t.$\{2009\}$) $\ldots$
$\ldots \sqcap$ $\forall$relatedToLocation.($\exists$concretize_t.$\{$Switzerland$\} \sqcup \{$Switzerland$\}$))$\ldots$
$\ldots \sqcup \ldots$
$\ldots$ ($\forall$relatedToProduct.($\exists$concretize_t.$\{$Car$\} \sqcup \{$Car$\}$) $\ldots$
$\ldots \sqcap$ $\forall$relatedToTime.($\exists$concretize_t.$\{2009\} \sqcup \{2009\}$) $\ldots$
$\ldots \sqcap$ $\forall$relatedToLocation.($\exists$concretize_t.$\{$Switzerland$\}$))$\ldots$
minPrice(sales-Car-2009-Switzerland,10.000)
IC: $\exists$concretize_t.$\{$sales-Car-2009-Switzerland$\}$ $\ldots$
$\ldots \sqcap$ $\exists$relatedToProduct.brand $\sqcap \ldots$
$\ldots$ $\exists$relatedToTime.month $\sqcap \ldots$
$\ldots$ $\exists$relatedToLocation.city $\ldots$
$\ldots$ $\sqsubseteq$ $\forall$minQty.Integer $\sqcap = 1$ minQty.$\top$

Mapping Output 40: GM-Relationship *sales-Car-2009-Switzerland* - Part I

IC: $\exists$concretize_t.$\{$sales-Car-2009-Switzerland$\}$ …
… $\sqcap$ ($\exists$relatedToProduct.($\exists$concretize_t.brand $\sqcup$ brand) …
… $\sqcup$ $\exists$relatedToTime.($\exists$concretize_t.month $\sqcup$ month) …
… $\sqcup$ $\exists$relatedToLocation.($\exists$concretize_t.city $\sqcup$ city)) …
… $\sqsubseteq$ $\exists$concretize_t.($\exists$concretize_t.$\{$sales-Car-2009-Switzerland$\}$ …
… $\sqcap$ $\exists$relatedToProduct.brand …
… $\sqcap$ $\exists$relatedToTime.month …
… $\sqcap$ $\exists$relatedToLocation.city) …
… $\sqcup$($\exists$relatedToProduct.brand …
… $\sqcap$ $\exists$relatedToTime.month …
… $\sqcap$ $\exists$relatedToLocation.city)
$\exists$minQty.$\top$ $\sqsubseteq$ $\exists$concretize_t.$\{$sales-Car-2009-Switzerland$\}$ …
… $\sqcap$ $\exists$relatedToProduct.brand $\sqcap$ …
… $\exists$relatedToTime.month $\sqcap$ …
… $\exists$relatedToLocation.city …
$\exists$relatedToProduct.brand $\sqcap$ …
… $\exists$relatedToTime.month $\sqcap$ …
… $\exists$relatedToLocation.city …
… $\sqsubseteq$ $\exists$concretize_t.$\{$sales-Car-2009-Switzerland$\}$

Mapping Output 41: GM-Relationship *sales-Car-2009-Switzerland* - Part II

sales-Product-Time-Location $\not\approx$ sales-Car-2009-Switzerland

Mapping Output 42: Unique Name Assumption

4 Related Work, Suggestions for Further Work and Conclusions

In this last chapter additional related work with respect to meta-modeling support in OWL and several suggestions for further work are revealed, and some conclusions of the proposed conceptual modeling approach and the associated OWL mapping are provided.

4.1 Related Work for Meta-Modeling in OWL

A well-known meta-modeling technique in the current version of OWL, namely OWL 2, is "punning". It allows to use a single symbol to refer to any or all of an individual, a class, or a property and, thus, the introduction of ontological meta-classes to express relationships between instances and classes. Punning is a simple way to provide decidable meta-modeling facilities, which is supported by several OWL reasoners and typically seems to satisfy the semantic requirements.

A conceptual modeling approach, which is closely related to and contained in our approach, is *Materialization*, as already mentioned in section 1.2. In Borgida and Brachman [2003] a suitable mapping of the materialization relationship to DLs is presented in term of roles and/or sub-roles of *materializationOf*, which can directly be applied to obtain a meta-modeling facility in OWL.

4.2 Suggestions for Further Work

In the opinion of the author there are some issues, which can be considered in further work:

- The identification of potential for optimization with respect to reasoning performance, e.g., in term of a suitable pre-calculation of subsumption hierarchies. In the present diploma thesis a conceptual modeling approach was discussed, which can suitably be mapped to OWL for ontological engineering. Hence, optimization issues have not been discussed at all.

- Software support for modeling hetero-homogeneous hierarchies with gm-objects and gm-relationships, e.g., in form of a plug-in for the software tool Protégé[1], such that an additional meta-modeling technique in OWL is facilitated.

[1]See http://www.co-ode.org/resources/ for the ontology engineering with the Protégé plugin.

- Software support for fully automated mapping of gm-objects and gm-relationships to OWL and corresponding OWL reasoning if the original hierarchy model does not rely on OWL directly.

- Extension of gm-objects to allow for parent relations, which are represented by multigraphs, i.e., levels of abstraction can be related by multiple concretize-relationships.

- Different approaches to combine closed-world assumption and open-world assumption regarding the mapping to OWL, while preserving the semantics of gm-objects and gm-relationships.

4.3 Conclusions

In chapter 2 a multi-level modeling approach based on gm-objects and gm-relationships was introduced to particularly model hetero-homogeneous hierarchies. The main idea was to encapsulate levels of abstraction by means of gm-objects, which can again be related by gm-relationships at certain connection-levels. Gm-objects as well as gm-relationships can be organized in (multi-dimensional) concretization hierarchies. As shown by plenty of examples the modeling approach seems to be rather intuitive and incorporates existing multi-level modeling techniques and essential aspects of abstraction hierarchies. Thus, it is very promising that this technique allows to suitably tackle the problem of increasing complexity and amount of ancillary effects of information systems.

Supplementarily, in chapter 3 a mapping between the multi-level modeling approach and OWL was proposed. The ideas and concepts were transferred from conceptual modeling to ontological engineering, while preserving their semantics. So existing reasoning techniques can be used to analyze the data to improve the quality of the underlying ontology, and to query and access data in the semantic web at multiple levels of abstraction.

Finally, in this chapter we have stated some important issues, which have not been considered yet. So there is potential for extending the proposed modeling framework in terms of both the conceptual modeling part and the ontological engineering part.

Bibliography

Khalid M. Albarrak and Edgar H. Sibley. Translating relational & object-relational database models into owl models. In *IRI'09: Proceedings of the 10th IEEE international conference on Information Reuse & Integration*, pages 336–341, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-4114-3.

Nenad Anicic, Nenad Ivezic, and Zoran Marjanovic. Mapping xml schema to owl. In Guy Doumeingts, Jörg Müller, Gérard Morel, and Bruno Vallespir, editors, *Enterprise Interoperability*, chapter 23, pages 243–252. Springer London, London, 2007. ISBN 978-1-84628-713-8. doi: http://dx.doi.org/10.1007/978-1-84628-714-5_23.

T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5): 28–37, 2001.

Alex Borgida and Ronald J. Brachman. Conceptual modeling with description logics. pages 349–372, 2003.

E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6): 377–387, 1970.

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, and York Sure. *Semantic Web: Grundlagen*. Springer, Berlin, 2008. ISBN 978-3-540-33993-9. doi: 10.1007/978-3-540-33994-6.

Nassim Kobeissy, Marc GIROD Genet, and Djamal Zeghlache. Mapping xml to owl for seamless information retrieval in context-aware environments. pages 361 –366, july 2007. doi: 10.1109/PERSER.2007.4283940.

Dmitry V. Levshin. Mapping relational databases to the semantic web with original meaning. In *KSEM '09: Proceedings of the 3rd International Conference on Knowledge Science, Engineering and Management*, pages 5–16, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-10487-9. doi: http://dx.doi.org/10.1007/978-3-642-10488-6_5.

George A. Miller and Florentina Hristea. Wordnet nouns: Classes and instances. *Comput. Linguist.*, 32(1):1–3, 2006. ISSN 0891-2017. doi: http://dx.doi.org/10.1162/coli.2006.32.1.1.

Boris Motik, Ian Horrocks, and Ulrike Sattler. Bridging the Gap Between OWL and Relational Databases. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *Proc. of the 16th International World Wide Web Conference (WWW 2007)*, pages 807–816, Banff, AB, Canada, May 8–12 2007. ACM Press.

Boris Motik, Ian Horrocks, and Ulrike Sattler. Bridging the Gap Between OWL and Relational Databases. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 7(2):74–89, 2009.

Bernd Neumayr and Michael Schrefl. Comparison Criteria for Ontological Multi-Level Modeling. In *Technical Report 08.03.*, chapter Modeling Technique for Multi-Level Abstraction. Department of Business Informatics - Data & Knowledge Engineering, Johannes Kepler University Linz, Altenberger Str. 69, 4040 Linz, Austria, 2008.

Bernd Neumayr and Michael Schrefl. Multi-level Conceptual Modeling and OWL. In *ER '09: Proceedings of the ER 2009 Workshops (CoMoL, ETheCoM, FP-UML, MOST-ONISW, QoIS, RIGiM, SeCoGIS) on Advances in Conceptual Modeling - Challenging Perspectives*, pages 189–199, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-04946-0. doi: http://dx.doi.org/10.1007/978-3-642-04947-7_23.

Bernd Neumayr, Katharina Grün, and Michael Schrefl. Multi-Level Domain Modeling with M-Objects and M-Relationships. In Sebastian Link and Markus Kirchberg, editors, *Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009)*, volume 96 of *CRPIT*, pages 107–116, Wellington, New Zealand, 2009a. ACS.

Bernd Neumayr, Michael Schrefl, and Bernhard Thalheim. Modeling Techniques for Multi-Level Abstraction. chapter Modeling Technique for Multi-Level Abstraction. Department of Business Informatics - Data & Knowledge Engineering, Johannes Kepler University Linz, Altenberger Str. 69, 4040 Linz, Austria, 2009b.

Bernd Neumayr, Michael Schrefl, and Bernhard Thalheim. Hetero-Homogeneous Hierarchies in Data Warehouses. In Sebastian Linz, K. Aditya, and K. Ghose, editors, *Proceedings of the seventh Asia-Pacific Conference on Conceptual Modelling (APCCM 2010)*, volume 110 of *CRPIT*, Brisbane, Australia, 2010. ACS.

Sunita S. Sane and Archana Shirke. Generating OWL ontologies from a relational databases for the semantic web. In *ICAC3 '09: Proceedings of the International Conference on Advances in Computing, Communication and Control*, pages 157–162, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-351-8. doi: http://doi.acm.org/10.1145/1523103.1523136.

Peter Spyns, Robert Meersman, and Mustafa Jarrar. Data modelling versus ontology engineering. *SIGMOD Rec.*, 31(4):12–17, 2002. ISSN 0163-5808. doi: http://doi.acm.org/10.1145/637411.637413.

Zhuoming Xu, Shichao Zhang, and Yisheng Dong. Mapping between relational database schema and owl ontology for deep annotation. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 548–552, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2747-7. doi: http://dx.doi.org/10.1109/WI.2006.114.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Linz, Juni 2010 Karl Rieger