Integrated Metamodeling and Diagnosis in OWL 2

Birte Glimm, Sebastian Rudolph, and Johanna Völker

Oxford University Computation Laboratory, UK
birte.glimm@comlab.ox.ac.uk
Institute AIFB, Karlsruhe Institute of Technology, DE
rudolph@kit.edu
KR & KM Research Group, University of Mannheim, DE
voelker@informatik.uni-mannheim.de

Abstract. Ontological metamodeling has a variety of applications yet only very restricted forms are supported by OWL 2 directly. We propose a novel encoding scheme enabling class-based metamodeling inside the domain ontology with full reasoning support through standard OWL 2 reasoning systems. We demonstrate the usefulness of our method by applying it to the OntoClean methodology. En passant, we address performance problems arising from the inconsistency diagnosis strategy originally proposed for OntoClean by introducing an alternative technique where sources of conflicts are indicated by means of marker predicates.

1 Introduction

Applications of metamodeling in Ontology Engineering are manifold, including the representation of provenance or versioning information as well as the documentation of modeling decisions. Roughly speaking, metamodeling allows for referring to *predicates* (classes and properties in OWL) as if they were domain individuals. This way it is possible to assert the membership of classes in metaclasses and interconnect them via metaroles.

Consider, for example, the following extract of a knowledge base about animals and the respective species they belong to.

$$(\mathsf{GoldenEagle} \sqcup \mathsf{HaastsEagle})(\mathit{harry}) \qquad \mathsf{HouseMouse}(\mathit{jerry})$$

Intuitively, we specify that the individual *harry* is a golden or a Haast's eagle and *jerry* is a common house mouse. Now, assume the knowledge base also expresses taxonomic relationships assigning species to orders of animals.[1]

$$\mathsf{GoldenEagle} \sqsubseteq \mathsf{Falconiformes} \qquad \mathsf{HouseMouse} \sqsubseteq \mathsf{Rodentia}$$
$$\mathsf{HaastsEagle} \sqsubseteq \mathsf{Falconiformes}$$

If, additionally, we were to specify which of the zoological terms actually denote species and which denote orders, we could introduce the classes `Species` and `Order`.

[1] Species is the most specific level within the biological classification and order is a more general one, e.g., Golden Eagle (A. chrysaetos) is a *species*, whereas Falconiformes is the *order* of Golden Eagle. In Europe the Falconiformes order is commonly split into Falconiformes and Accipitriformes, but we neglect that here.

Treating those classes on a level with Rodentia etc. by subclass statements like Rodentia $\sqsubseteq$ Order leads to consequences that are doubtful (like Order(*jerry*)) or outright unwanted (like HouseMouse $\sqsubseteq$ Order). Therefore, species and order should be treated as *metaclasses* the members of which are themselves classes, i.e., we would like to make statements such as

Species(GoldenEagle)	Order(Falconiformes)
Species(HaastsEagle)	Order(Rodentia)
Species(HouseMouse)	

Likewise we may think of *metaroles* that interrelate classes instead of individuals. In particular, the subclass relationship between classes can be seen as such a metarole (one with a built-in meaning instead of one that can be freely defined). In fact, many evaluation or design criteria for ontologies [12, 4] directly refer to the hierarchy of classes in an ontology. Considering our example, one obvious design criterion would be that for A a species and B an order, $B \sqsubseteq A$ must not hold as this would contradict the conventional organization of zoological taxonomies.

Current ontology languages differ with respect to their support for metamodeling. While it is supported by OWL Full, this high expressivity leads to undecidability as shown by Motik [9], who also discusses milder variants of metamodeling. One variant, which is also supported by OWL 2 DL, is called *punning*. Punning allows for using the same identifier, e.g., for an individual and a class. The class and its corresponding individual are, however, treated as entirely independent, which disallows many of the intended usage scenarios of metamodeling. As another lightweight metamodeling feature, OWL 2 allows for annotation properties, which may associate information to classes, roles, and even axioms. In OWL 2 DL and all its subprofiles, these properties do not carry any semantics and are not used for reasoning.

One way to facilitate more expressive metamodeling while still supporting the use of off-the-shelf reasoning tools for OWL is to maintain two (or more) ontologies, keeping the basic domain knowledge separate from the meta knowledge. In that case, the two ontologies must be kept in sync by additional external mechanisms. Thereby, information obtained from reasoning in the basic ontology (like its subclass hierarchy) is fed into the metaontology as explicit statements. Based on this, reasoning in the enriched metaontology can be carried out. Clearly, this approach comes with increased maintenance efforts. Examples for this strategy are [10] and [14].

We extend this state of the art in two ways, which are independent from each other, but can be combined:

1. We introduce a technique that enables class-based metamodeling within one ontology. Thereby, subclass relationships between classes are axiomatically synchronized with role memberships of class-representing individuals. Meta-level constraints on classes and their subsumption relationships can then be expressed as OWL axioms in the same ontology as the actual content.
2. We propose a way of expressing meta-level constraints in a way that does not lead to inconsistency, but rather indicates constraint violations by auxiliary classes or roles. Thus, the origins of these violations can be localized by comparably cheap instance retrieval operations instead of costly debugging strategies.

We proceed as follows: The next section introduces the necessary preliminaries of the description logic $\mathcal{SROIQ}$ underlying the OWL 2 standard. We use the DL notation for its brevity. Section 3 introduces our technique enabling ontology-inherent metamodeling. Section 4 sketches the OntoClean methodology as one possible metamodeling use case. Section 5 describes the original OWL-based OntoClean constraint checking approach as well as our metamodeling-based modification of it. Section 6 introduces another modification of the methodology by suggesting to use marker predicates instead of explanations. Finally, Section 7 provides an evaluation of the proposed techniques before we conclude in Section 8. A more detailed treatise can be found in the extended version of the paper [3].

2 Preliminaries

We just recall the basic definitions for the description logic $\mathcal{SROIQ}$ [6]. For further details on DLs we refer interested readers to the Description Logic Handbook [1]. As our definitions are based on DLs, we use the terms *ontology* and *knowledge base* interchangeably.[2]

Definition 1. *Let* N_R, N_C, *and* N_I *be three disjoint sets of* role names *containing the* universal role $U \in \mathsf{N}_R$, class names, *and* individual names, *respectively. A* $\mathcal{SROIQ}$ *RBox for* N_R *is based on a set* $\mathbf{R}$ *of* roles *defined as* $\mathbf{R} \coloneqq \mathsf{N}_R \cup \{R^- \mid R \in \mathsf{N}_R\}$, *where we set* $\mathrm{Inv}(R) \coloneqq R^-$ *and* $\mathrm{Inv}(R^-) \coloneqq R$ *to simplify notation. In the sequel, we will use the symbols* R, S, *possibly with subscripts, to denote roles.*

A generalised role inclusion axiom *(RIA) is a statement of the form* $S_1 \circ \ldots \circ S_n \sqsubseteq R$, *and a set of such RIAs is a generalised* role hierarchy. *A role will be called* non-simple *for some role hierarchy if it can be implied by some role chain, otherwise it is* simple.

A role disjointness assertion *is a statement of the form* $\mathsf{Dis}(S, S')$, *where* S *and* S' *are simple. A* $\mathcal{SROIQ}$ RBox *is the union of a set of role disjointness assertions together with a role hierarchy. A* $\mathcal{SROIQ}$ *RBox is regular if its role hierarchy is regular.*

For brevity, we omit a precise definition of *simple* roles and role hierarchy *regularity*, and refer interested readers to [6]. Note that number restrictions (defined below) can only be formed with simple roles to guarantee the decidability of the standard reasoning tasks such as checking knowledge base consistency.

Definition 2. *Given a* $\mathcal{SROIQ}$ *RBox* $\mathcal{R}$, *the set of* class expressions $\mathbf{C}$ *is defined as follows:*

- $\mathsf{N}_C \subseteq \mathbf{C}$, $\top \in \mathbf{C}$, $\bot \in \mathbf{C}$,
- *if* $\mathsf{C}, \mathsf{D} \in \mathbf{C}$, $R \in \mathbf{R}$, $S \in \mathbf{R}$ *a simple role,* $a \in \mathsf{N}_I$, *and* n *a non-negative integer, then* $\neg\mathsf{C}$, $\mathsf{C} \sqcap \mathsf{D}$, $\mathsf{C} \sqcup \mathsf{D}$, $\{a\}$, $\forall R.\mathsf{C}$, $\exists R.\mathsf{C}$, $\exists S.\mathsf{Self}$, $\leq n\, S.\mathsf{C}$, *and* $\geq n\, S.\mathsf{C}$ *are also class expressions.*

[2] Moreover, we use the term *classes* instead of concepts for unary predicates, whereas we refer to binary predicates as *roles* instead of properties in order to avoid confusion with the term *metaproperties* introduced by OntoClean.

Table 1. Semantics of class expressions in $\mathcal{SROIQ}$ for an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$

Name	Syntax	Semantics
inverse role	R^-	$\{\langle x, y\rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \langle y, x\rangle \in R^{\mathcal{I}}\}$
universal role	U	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
negation	$\neg \mathsf{C}$	$\Delta^{\mathcal{I}} \setminus \mathsf{C}^{\mathcal{I}}$
conjunction	$\mathsf{C} \sqcap \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}}$
disjunction	$\mathsf{C} \sqcup \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cup \mathsf{D}^{\mathcal{I}}$
nominals	$\{a\}$	$\{a^{\mathcal{I}}\}$
univ. restriction	$\forall R.\mathsf{C}$	$\{x \in \Delta^{\mathcal{I}} \mid \langle x, y\rangle \in R^{\mathcal{I}}$ implies $y \in \mathsf{C}^{\mathcal{I}}\}$
exist. restriction	$\exists R.\mathsf{C}$	$\{x \in \Delta^{\mathcal{I}} \mid$ for some $y \in \Delta^{\mathcal{I}}$, $\langle x, y\rangle \in R^{\mathcal{I}}$ and $y \in \mathsf{C}^{\mathcal{I}}\}$
Self construct	$\exists S.\mathsf{Self}$	$\{x \in \Delta^{\mathcal{I}} \mid \langle x, x\rangle \in S^{\mathcal{I}}\}$
qualified number	$\leq n\, S.\mathsf{C}$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid \langle x, y\rangle \in S^{\mathcal{I}}$ and $y \in \mathsf{C}^{\mathcal{I}}\} \leq n\}$
restriction	$\geq n\, S.\mathsf{C}$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid \langle x, y\rangle \in S^{\mathcal{I}}$ and $y \in \mathsf{C}^{\mathcal{I}}\} \geq n\}$

In the remainder, we use C *and* D *to denote class expressions. A* $\mathcal{SROIQ}$ TBox *is a set of* general class inclusion axioms *(GCIs) of the form* $\mathsf{C} \sqsubseteq \mathsf{D}$. *We use* $\mathsf{C} \equiv \mathsf{D}$ *to abbreviate* $\mathsf{C} \sqsubseteq \mathsf{D}$ *and* $\mathsf{D} \sqsubseteq \mathsf{C}$. *An* individual assertion *can have the form* $\mathsf{C}(a)$ *or* $R(a, b)$ *with* $a, b \in \mathsf{N}_I$ *individual names. A* $\mathcal{SROIQ}$ ABox *is a set of individual assertions.*

A $\mathcal{SROIQ}$ ontology $\mathcal{O}$ *is the union of a regular RBox* $\mathcal{R}$*, an ABox* $\mathcal{A}$ *and TBox* $\mathcal{T}$ *for* $\mathcal{R}$. *The* vocabulary *of an ontology, denoted* $\mathsf{voc}(\mathcal{O})$*, is a triple* $(\mathsf{O}_C, \mathsf{O}_R, \mathsf{O}_I)$ *with* O_C *the set of class names occurring in* $\mathcal{O}$*,* O_R *the set of role names occurring in* $\mathcal{O}$*, and* O_I *the set of individual names occurring in* $\mathcal{O}$.

The semantics of $\mathcal{SROIQ}$ ontologies is given by means of interpretations.

Definition 3. *An interpretation* $\mathcal{I}$ *consists of a set* $\Delta^{\mathcal{I}}$ *called* domain *(the elements of it being called* individuals*) together with a function* $\cdot^{\mathcal{I}}$ *mapping individual names to elements of* $\Delta^{\mathcal{I}}$*, class names to subsets of* $\Delta^{\mathcal{I}}$*, and role names to subsets of* $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

The function $\cdot^{\mathcal{I}}$ *is inductively extended to role and class expressions as shown in Table 1. An interpretation* $\mathcal{I}$ satisfies *an axiom* φ *if we find that* $\mathcal{I} \models \varphi$*:*

- $\mathcal{I} \models S \sqsubseteq R$ *if* $S^{\mathcal{I}} \subseteq R^{\mathcal{I}}$,
- $\mathcal{I} \models S_1 \circ \ldots \circ S_n \sqsubseteq R$ *if* $S_1^{\mathcal{I}} \circ \ldots \circ S_n^{\mathcal{I}} \subseteq R^{\mathcal{I}}$ *(*$\circ$ *being overloaded to denote the standard composition of binary relations here),*
- $\mathcal{I} \models \mathsf{Dis}(R, S)$ *if* $R^{\mathcal{I}}$ *and* $S^{\mathcal{I}}$ *are disjoint,*
- $\mathcal{I} \models \mathsf{C} \sqsubseteq \mathsf{D}$ *if* $\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$.

An interpretation $\mathcal{I}$ satisfies $\mathsf{C}(a)$ *if* $a^{\mathcal{I}} \in C^{\mathcal{I}}$ *and* $R(a, b)$ *if* $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. *An interpretation* $\mathcal{I}$ satisfies *an ontology* $\mathcal{O}$ *(we then also say that* $\mathcal{I}$ *is a* model *of* $\mathcal{O}$ *and write* $\mathcal{I} \models \mathcal{O}$*) if it satisfies all axioms of* $\mathcal{O}$*. An ontology* $\mathcal{O}$ *is* satisfiable *if it has a model. An ontology* $\mathcal{O}$ entails *an axiom* φ*, if every model of* $\mathcal{O}$ *is a model of* φ.

Further details on $\mathcal{SROIQ}$ can be found in [6]. We have omitted here several syntactic constructs that can be expressed indirectly, especially RBox assertions for transitivity, reflexivity of simple roles, and symmetry.

In the remainder, we use the following notational convention: individual names are written in italic, e.g., *jerry*. Class names are written in sans serif font, e.g., HouseMouse and role names are written in normal serif font, e.g., eats, unless they are used to denote metaclasses or metaroles for which we use typewriter font, e.g., `Species` or `subClassOf`.

3 Ontology-Inherent Metamodeling for Classes

We will now show how to define a metamodeling-enabled version $\mathcal{O}^{\text{meta}}$ for a given ontology $\mathcal{O}$. The converted ontology $\mathcal{O}^{\text{meta}}$ will be such that each model of the converted ontology has two different kinds of individuals: the *class individuals* are individuals that represent classes and each such individual is an instance of the newly introduced metaclass `Class`. On the other hand, the model also contains *proper individuals* and all these are instances of the newly introduced class `Inst`. Subclass relationships between a class C and a class D in the given ontology $\mathcal{O}$ are materialized as role instances: the individual that represents the class C, say o_{C}, and the individual that represents D, say o_{D}, are interconnected by the newly introduced metarole `subClassOf`. Similarly, a class membership of an individual a in a class C in the given ontology becomes manifest in a `type` relationship between a and o_{C}, for `type` also a freshly introduced role in $\mathcal{O}^{\text{meta}}$. We further introduce an auxiliary role $R_{\texttt{Inst}}$, which is used to localize the universal role. These correspondences can then be used to check for modeling errors and to examine quality properties of the ontology.

Definition 4. *Let $\mathcal{O}$ be a domain ontology with vocabulary* $\mathsf{voc}(\mathsf{O}_C, \mathsf{O}_R, \mathsf{O}_I)$*. The* vocabulary of the metamodeling-enabled version $\mathcal{O}^{\text{meta}}$ of $\mathcal{O}$ *is:*

$$\begin{aligned}
\mathsf{O}_C^{\text{meta}} &:= \mathsf{O}_C \cup \{\texttt{Inst}, \texttt{Class}\} \\
\mathsf{O}_R^{\text{meta}} &:= \mathsf{O}_R \cup \{\texttt{type}, \texttt{subClassOf}, R_{\texttt{Inst}}\} \\
\mathsf{O}_I^{\text{meta}} &:= \mathsf{O}_I \cup \{o_{\mathsf{C}} \mid \mathsf{C} \in \mathsf{O}_C\}
\end{aligned}$$

where all the newly introduced names are fresh, i.e., they are not part of $\mathsf{voc}(\mathcal{O})$.

We define the functions bound(·), **SepDom**(·), **Typing**(·), *and* **MatSubClass**(·), *which take an ontology, i.e., a set of axioms, and return a set of axioms. The function* bound(·) *returns its input after rewriting it as follows: first, every occurrence of X having one of the forms* $\top$, $\neg\mathsf{C}$, $\forall R.\mathsf{C}$, $\leq n\, R.\mathsf{C}$, $\exists U.\mathsf{Self}$ *is substituted by* $\texttt{Inst} \sqcap X$*, where we explicitly allow for complex classes* C*. Next, the universal role is localized by substituting every* $\forall U.\mathsf{C}$ *by* $\forall U.(\neg\texttt{Inst} \sqcup \mathsf{C})$ *and every U occurring on the left hand side of a role chain axiom by* $R_{\texttt{Inst}} \circ U \circ R_{\texttt{Inst}}$ *where* $R_{\texttt{Inst}}$ *is axiomatized via* $\exists R_{\texttt{Inst}}.\mathsf{Self} \equiv \texttt{Inst}$.[3] *We extend* bound(·) *in the obvious way to also rewrite an axiom or a class expression. The functions* **SepDom**, **Typing**, *and* **MatSubClass** *return a set of axioms as specified in Table 2. The* metamodeling-enabled version $\mathcal{O}^{\text{meta}}$ *of* $\mathcal{O}$ *is*

$$\text{bound}(\mathcal{O}) \cup \textbf{SepDom}(\mathcal{O}) \cup \textbf{Typing}(\mathcal{O}) \cup \textbf{MatSubClass}(\mathcal{O})$$

[3] It is not hard to check that none of these transformations harms the global syntactic constraints.

Table 2. Returned axioms for an ontology $\mathcal{O}$ by **SepDom**, **Typing**, and **MatSubClass**

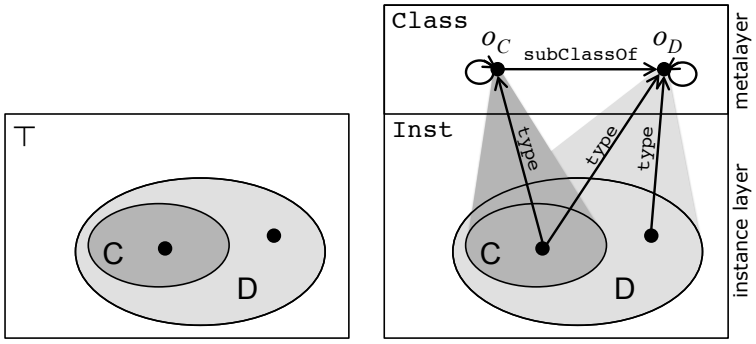
SepDom($\mathcal{O}$):	$\texttt{Inst} \equiv \neg\texttt{Class}$		(1)
	$\texttt{Class}(o_{\mathsf{C}})$	for all $\mathsf{C} \in \mathsf{O}_C$	(2)
	$\texttt{Inst}(i)$	for all $i \in \mathsf{O}_I$	(3)
	$\exists R.\top \sqsubseteq \texttt{Inst}$	for all $R \in \mathsf{O}_R$	(4)
	$\top \sqsubseteq \forall R.\texttt{Inst}$	for all $R \in \mathsf{O}_R$	(5)
	$\exists\texttt{type}.\top \sqsubseteq \texttt{Inst}$		(6)
	$\top \sqsubseteq \forall\texttt{type}.\texttt{Class}$		(7)
	$\exists\texttt{subClassOf}.\top \sqsubseteq \texttt{Class}$		(8)
	$\top \sqsubseteq \forall\texttt{subClassOf}.\texttt{Class}$		(9)
Typing($\mathcal{O}$):	$\mathsf{C} \equiv \exists\texttt{type}.\{o_{\mathsf{C}}\}$	for all $\mathsf{C} \in \mathsf{O}_C$	(10)
MatSubClass($\mathcal{O}$):	$\texttt{Class} \sqcap \forall\texttt{type}^{-}.\exists\texttt{type}.\{o_{\mathsf{C}}\} \equiv \texttt{Class} \sqcap \exists\texttt{subClassOf}.\{o_{\mathsf{C}}\}$	for all $\mathsf{C} \in \mathsf{O}_C$	(11)

Roughly speaking, given an ontology $\mathcal{O}$, the function bound($\mathcal{O}$) ensures that the complete domain of $\mathcal{O}$ is "squeezed" into the class `Inst` and also class construction is forced to only involve individuals from `Inst`. The axioms constructed by **SepDom**($\mathcal{O}$) have the following purpose: Axiom (1) makes sure that the newly established metalayer does not interfere with the instance layer. Axiom (2) ensures that all class-representative individuals lie in the metalayer. Axiom (3) forces every named individual of the original ontology to be in the instance layer. Axioms (4) and (5) state that every role of the original ontology is forced to start and end only in the instance layer. Axioms (6) and (7) stipulate that the `type`-role starts in the instance layer and ends in the metalayer. Finally, Axiom (8) and (9) specify that the $\texttt{subClassOf}(\mathcal{O})$ role is allowed to interconnect only individuals from the metalayer. The axioms from **Typing**($\mathcal{O}$) ensure that class members of C are exactly those domain individuals which are connected to C's representative o_{C} via the `type` role, while axioms from **MatSubClass**($\mathcal{O}$) finally synchronize actual subclass relationships in the instance layer with the `subClassOf` links between the corresponding representatives in the metalayer.

Note that the size of $\mathcal{O}^{\text{meta}}$ is linearly bounded by the size of $\mathcal{O}$. Intuitively, the conversion from $\mathcal{O}$ to $\mathcal{O}^{\text{meta}}$ realizes a model conversion: given a model of $\mathcal{O}$, the transformation endows the model with a metalayer containing reified atomic classes o_{C}. As in RDF, class membership of the original individuals is now indicated by the newly introduced `type` role and class subsumption by the `subClassOf` role which is axiomatically synchronized with the actually valid subclass relation in the considered model. Thereby, we materialize the hierarchy among classes of a particular model in the metalayer. Figure 1 depicts the established correspondences in a schematic way.

Note that no original model is ruled out by this process which also ensures that the conversion does not cause new (unwanted) consequences. In the sequel, we characterize the above mentioned properties of $\mathcal{O}^{\text{meta}}$ more formally:

Theorem 1. *Let $\mathcal{O}$ be an OWL ontology and $\mathcal{O}^{\text{meta}}$ its metamodeling-enabled version as specified in Definition 4. Then the following properties hold:*



Model of $\mathcal{O}$. Corresponding model of $\mathcal{O}^{\text{meta}}$.

Fig. 1. Sketch of the established interdependencies in the models of $\mathcal{O}^{\text{meta}}$.

1. *For any OWL axiom a containing only names from O_C, O_R and O_I, we have that $\mathcal{O} \models a$ iff $\mathcal{O}^{\text{meta}} \models \text{bound}(a)$.*
2. *For any class name $\mathsf{C} \in \mathsf{O}_C$ and instance name $i \in \mathsf{O}_I$, we have that $\mathcal{O} \models \mathsf{C}(i)$ iff $\mathcal{O}^{\text{meta}} \models \mathtt{type}(i, o_{\mathsf{C}})$.*
3. *For any two named classes $\mathsf{C}, \mathsf{D} \in \mathsf{O}_C$, we have that $\mathcal{O} \models \mathsf{C} \sqsubseteq \mathsf{D}$ iff $\mathcal{O}^{\text{meta}} \models \mathtt{subClassOf}(o_{\mathsf{C}}, o_{\mathsf{D}})$.*

Proof. For the first claim, given a model $\mathcal{I}$ of $\mathcal{O}$, we construct a model $\mathit{meta}(\mathcal{I}) = \mathcal{J}$ of $\mathcal{O}^{\text{meta}}$ as follows:

$$\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}} \cup \{\delta_{\mathsf{C}} \mid \mathsf{C} \in \mathsf{O}_C\}$$
$$\varsigma^{\mathcal{J}} = \varsigma^{\mathcal{I}} \quad \text{for all } \varsigma \in \mathsf{O}_C \cup \mathsf{O}_R \cup \mathsf{O}_I$$
$$\mathtt{type}^{\mathcal{J}} = \{\langle \delta, \delta_{\mathsf{C}} \rangle \mid \delta \in \mathsf{C}^{\mathcal{I}}\} \qquad \mathtt{Inst}^{\mathcal{J}} = \Delta^{\mathcal{I}}$$
$$\mathtt{Class}^{\mathcal{J}} = \{\delta_{\mathsf{C}} \mid \mathsf{C} \in \mathsf{O}_C\} \qquad \mathtt{subClassOf}^{\mathcal{J}} = \{\langle \delta_{\mathsf{C}}, \delta_{\mathsf{D}} \rangle \mid \mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}\}$$

By construction $\mathcal{J}$ satisfies all axioms from **SepDom**($\mathcal{O}$) ∪ **Typing**($\mathcal{O}$) ∪ **MatSubClass**($\mathcal{O}$). By induction, we obtain, for every class C containing only names from $\mathsf{voc}(\mathcal{O})$, that $\text{bound}(\mathsf{C})^{\mathcal{J}} = \mathsf{C}^{\mathcal{I}}$ (claim †). This in turn guarantees that, for every axiom Ax using only terms from $\mathsf{voc}(\mathcal{O})$, we obtain $\mathcal{I} \models Ax$ iff $\mathcal{J} \models \text{bound}(Ax)$. In particular, $\mathcal{J}$ also satisfies $\text{bound}(\mathcal{O})$, whence it is a model of $\mathcal{O}^{\text{meta}}$ as claimed.

Using this transformation, we can show that $\mathcal{O}^{\text{meta}} \models \text{bound}(Ax)$ implies $\mathcal{O} \models Ax$. We demonstrate the case for GCIs. Suppose $\mathcal{O}^{\text{meta}} \models \text{bound}(\mathsf{C}) \sqsubseteq \text{bound}(\mathsf{D})$ but $\mathcal{O} \not\models \mathsf{C} \sqsubseteq \mathsf{D}$. Then there is a model $\mathcal{I}$ of $\mathcal{O}$ with $\mathsf{C}^{\mathcal{I}} \not\subseteq \mathsf{D}^{\mathcal{I}}$. But then there is a model $\mathcal{J} = \mathit{meta}(\mathcal{I})$ with $\text{bound}(\mathsf{C})^{\mathcal{J}} \not\subseteq \text{bound}(\mathsf{D})^{\mathcal{J}}$ (according to †) contradicting our assumption. For the other axiom types, the correspondence can be shown along the same lines.

The other direction ($\mathcal{O} \models Ax$ implying $\mathcal{O}^{\text{meta}} \models \text{bound}(Ax)$) is shown analogously using the transformation converting models $\mathcal{J}$ of $\mathcal{O}^{\text{meta}}$ to models $\mathcal{I}$ of $\mathcal{O}$ as follows:

$$\Delta^{\mathcal{I}} = \mathtt{Inst}^{\mathcal{J}} \qquad \varsigma^{\mathcal{I}} = \varsigma^{\mathcal{J}} \text{ for all } \varsigma \in \mathsf{O}_C \cup \mathsf{O}_R \cup \mathsf{O}_I$$

Note that the additional axioms of $\mathcal{O}^{\text{meta}}$ ensure that only individuals from $\Delta^{\mathcal{I}}$ occur in every $\varsigma^{\mathcal{J}}$, whence $\mathcal{I}$ is well-defined. Again we can establish $\text{bound}(\mathsf{C})^{\mathcal{J}} = \mathsf{C}^{\mathcal{I}}$ by induction and use this to show $\mathcal{O}^{\text{meta}} \models \text{bound}(Ax)$ implying $\mathcal{O} \models Ax$.

Table 3. An example ontology $\mathcal{O}$ and its metamodeling-enabled version $\mathcal{O}^{\text{meta}}$

	Ontology $\mathcal{O}$:
	$\mathsf{HouseMouse} \sqsubseteq \exists \mathsf{eats}^{-}.\mathsf{GoldenEagle}$ $\quad \mathsf{Prey} \equiv \exists \mathsf{eats}^{-}.\top$
	$\mathsf{HouseMouse}(\mathit{jerry})$ $\quad (\mathsf{GoldenEagle} \sqcup \mathsf{HaastsEagle})(\mathit{harry})$
	Metaontology $\mathcal{O}^{\text{meta}}$:
bound($\mathcal{O}$):	$\mathsf{HouseMouse} \sqsubseteq \exists \mathsf{eats}^{-}.\mathsf{GoldenEagle}$ $\quad \mathsf{Prey} \equiv \exists \mathsf{eats}^{-}.\mathtt{Inst}$
	$\mathsf{HouseMouse}(\mathit{jerry})$ $\quad (\mathsf{GoldenEagle} \sqcup \mathsf{HaastsEagle})(\mathit{harry})$
SepDom($\mathcal{O}$):	$\mathtt{Inst} \equiv \neg \mathtt{Class}$
	$\mathtt{Class}(o_{\mathsf{HouseMouse}})$ $\quad \mathtt{Class}(o_{\mathsf{GoldenEagle}})$ $\quad \mathtt{Class}(o_{\mathsf{Prey}})$ $\quad \mathtt{Class}(o_{\mathsf{HaastsEagle}})$
	$\mathtt{Inst}(\mathit{jerry})$ $\quad \mathtt{Inst}(\mathit{harry})$
	$\exists \mathsf{eats}.\top \sqsubseteq \mathtt{Inst}$ $\quad \top \sqsubseteq \forall \mathsf{eats}.\mathtt{Inst}$ $\quad \exists \mathtt{type}.\top \sqsubseteq \mathtt{Inst}$ $\quad \top \sqsubseteq \forall \mathtt{type}.\mathtt{Class}$
	$\exists \mathtt{subClassOf}.\top \sqsubseteq \mathtt{Class}$ $\quad \top \sqsubseteq \forall \mathtt{subClassOf}.\mathtt{Class}$
Typing($\mathcal{O}$):	$\mathsf{HouseMouse} \equiv \exists \mathtt{type}.\{o_{\mathsf{HouseMouse}}\}$ $\quad \mathsf{Prey} \equiv \exists \mathtt{type}.\{o_{\mathsf{Prey}}\}$
	$\mathsf{GoldenEagle} \equiv \exists \mathtt{type}.\{o_{\mathsf{GoldenEagle}}\}$ $\quad \mathsf{HaastsEagle} \equiv \exists \mathtt{type}.\{o_{\mathsf{HaastsEagle}}\}$
MatSubClass($\mathcal{O}$):	$\mathtt{Class} \sqcap \forall \mathtt{type}^{-}.\exists \mathtt{type}.\{o_{\mathsf{HouseMouse}}\} \equiv \mathtt{Class} \sqcap \exists \mathtt{subClassOf}.\{o_{\mathsf{HouseMouse}}\}$
	$\mathtt{Class} \sqcap \forall \mathtt{type}^{-}.\exists \mathtt{type}.\{o_{\mathsf{Prey}}\} \equiv \mathtt{Class} \sqcap \exists \mathtt{subClassOf}.\{o_{\mathsf{Prey}}\}$
	$\mathtt{Class} \sqcap \forall \mathtt{type}^{-}.\exists \mathtt{type}.\{o_{\mathsf{GoldenEagle}}\} \equiv \mathtt{Class} \sqcap \exists \mathtt{subClassOf}.\{o_{\mathsf{GoldenEagle}}\}$
	$\mathtt{Class} \sqcap \forall \mathtt{type}^{-}.\exists \mathtt{type}.\{o_{\mathsf{HaastsEagle}}\} \equiv \mathtt{Class} \sqcap \exists \mathtt{subClassOf}.\{o_{\mathsf{HaastsEagle}}\}$

For the second claim, given $\mathcal{O} \models \mathsf{C}(i)$ we can conclude $\mathcal{O}^{\text{meta}} \models \text{bound}(\mathsf{C}(i))$ and hence $\mathcal{O}^{\text{meta}} \models \mathsf{C}(i)$ from which by **Typing**($\mathcal{O}$) follows $\mathtt{type}(i, o_{\mathsf{C}})$. The argument holds in both directions.

For the third claim, we have that from $\mathcal{O} \models \mathsf{C} \sqsubseteq \mathsf{D}$ follows $\mathcal{O}^{\text{meta}} \models \text{bound}(\mathsf{C}) \sqsubseteq \text{bound}(\mathsf{D})$ and, therefore, $\mathcal{O}^{\text{meta}} \models \mathsf{C} \sqsubseteq \mathsf{D}$. Considering a model $\mathcal{J}$ of $\mathcal{O}^{\text{meta}}$, **MatSubClass**($\mathcal{O}$) ensures that $\mathcal{J} \models \mathtt{subClassOf}(o_{\mathsf{C}}, o_{\mathsf{D}})$ iff $o_{\mathsf{C}} \in (\mathtt{Class} \sqcap \forall \mathtt{type}^{-}.\exists \mathtt{type}.\{o_{\mathsf{D}}\})^{\mathcal{J}}$. This can be simplified to $\{\delta \mid \langle \delta, o_{\mathsf{C}}^{\mathcal{J}} \rangle \in \mathtt{type}^{\mathcal{J}}\} \subseteq \{\delta \mid \langle \delta, o_{\mathsf{D}}^{\mathcal{J}} \rangle \in \mathtt{type}^{\mathcal{J}}\}$ which, by **Typing**($\mathcal{O}$), coincides with $\mathsf{C}^{\mathcal{J}} \subseteq \mathsf{D}^{\mathcal{J}}$ and is true by assumption. Again, the argument holds in both ways. □

As an example, consider the ontology $\mathcal{O}$ and its metamodeling-enabled version $\mathcal{O}^{\text{meta}}$ from Table 3. We find that $\mathsf{HouseMouse} \sqsubseteq \mathsf{Prey}$ is a consequence of $\mathcal{O}$, whence $\mathcal{O}^{\text{meta}}$ entails $\mathtt{subClassOf}(o_{\mathsf{HouseMouse}}, o_{\mathsf{Prey}})$. In $\mathcal{O}^{\text{meta}}$ we can further make statements such as $\mathtt{ExtinctSpecies}(o_{\mathsf{HaastsEagle}})$ where $\mathtt{ExtinctSpecies}$ is a metaclass used to state that Haast's eagle is an extinct species. If we then add the axiom $\mathtt{ExtinctSpecies} \sqsubseteq \forall \mathtt{type}^{-}.\bot$ to say that extinct species cannot have instances, an OWL 2 DL reasoner can deduce $\mathsf{GoldenEagle}(\mathit{harry})$.

In the following, we will illustrate the benefits of our approach on the basis of a more concrete application scenario: the evaluation of ontologies with respect to the OntoClean methodology.

4 OntoClean

This section gives a brief introduction to OntoClean (for a more thorough description refer, e.g., to Guarino and Welty [4]), a methodology developed in order to ensure the

correctness of taxonomies with respect to the philosophical principles of Formal Ontology. Central to OntoClean are the notions of **rigidity**, **unity**, **dependence** and **identity**, commonly known as *metaproperties*. Note that in the OntoClean terminology, *properties* are what is called *classes* in OWL. Metaproperties are, therefore, "properties of properties." Consequently, OntoClean can be considered a very natural application of metamodeling in ontology engineering and evaluation.

In the following, we will explain the process of applying the OntoClean methodology by making reference to the OntoClean example ontology introduced by Guarino and Welty ([4], Figure 1). This ontology, which consists of 22 classes such as Apple, Food, Person or Agent, illustrates some of the most frequent modeling errors in terms of OntoClean.

The process of applying the OntoClean methodology to a given ontology consists of two essential phases:

Phase 1: First, every single class of the ontology to be evaluated or redesigned is tagged with respect to the aforementioned metaproperties. This way, every class gets assigned a particular tagging such as +R-D+I+U, denoting the fact that this class is rigid (+R), non-dependent (-D), a sortal (+I, i.e. it carries an identity criterion) and that it has unity (+U).

Phase 2: In the second phase, after the metaproperty tagging has been completed, all the subsumption relationships of the ontology are checked according to a predefined set of OntoClean *constraints*. Any violation of such a constraint potentially indicates a fundamental misconceptualization in the subsumption hierarchy.

Hence, after performing the two steps, the result is a tagged ontology and a (potentially empty) list of misconceptualizations by whose means an ontology engineer can "clean" the ontology. In a nutshell, the key idea underlying OntoClean is to constrain the possible taxonomic relationships by disallowing subsumption relations between specific combinations of tagged classes. Welty et al. [13] show that analyzing and modifying an ontology according to the quality criteria defined by OntoClean can have a positive impact on the performance of an ontology-based application.

Metaproperties. As mentioned above, the original version of OntoClean is based on four metaproperties: rigidity, unity, identity and dependence – abbreviated as R, U, I and D, respectively. For brevity, we focus on rigidity omitting detailed explanations of the other metaproperties and referring the interested reader to, e.g., [4].

Rigidity is based on the notion of *essence*. A class is essential for an individual *iff* the individual is necessarily a member of this class, in all worlds and at all times. *Iff* a class is essential to all of its individuals, the class is called *rigid* and is tagged with +R. *Non-rigid* classes, i.e., classes which are not essential to some of their individuals, are tagged with -R. An anti-rigid class is one that is not essential to all of its individuals and thus tagged with ~R. Hence, every anti-rigid class is also a non-rigid class. Apple is a typical example for a rigid class, because the property of being an apple is *essential* to all of its individuals, or to put it differently: an apple is necessarily an apple and cannot stop being one. In this respect, Apple differs from classes such as Food which is mostly considered anti-rigid. Note, however, that the tagging of Food crucially hinges

on the intended semantics of this class. Welty [14] nicely illustrates this by an example: If Food is the class of all things edible by humans then it should be tagged as rigid (+R). If, in contrast, Food is a role that can be played by any individual while it is being eaten, we must consider it anti-rigid (~R). The latter sense was assumed by Guarino and Welty when they designed the aforementioned example ontology.

Constraints. The following formulation of the OntoClean constraints is literally taken from Welty [14]. We adhere to this version rather than to the more stringent formulation provided by Guarino and Welty [4] as it directly maps to the axiomatization in the original OntoClean metaontology (cf. Section 5).

1. A rigid class (+R) cannot be a subclass of an anti-rigid class (~R).
2. A class with unity (+U) cannot be a subclass of a class with anti-unity (~U).
3. All subclasses of a sortal are sortals (+I).
4. All subclasses of a dependent class are dependent (+D).

What seems a matter of merely philosophical consideration can in fact have practical implications. Imagine, for instance, that a rigid class Apple is subsumed by Food which is tagged as anti-rigid. Thus, Apple(a) would imply Food(a). It might also appear reasonable to model the class Poisoned as disjoint to Food. But now, as the ontology evolves and further class instantiations are added, we could state, for example, that a has been poisoned (formally, Poisoned(a)) – and the ontology turns logically inconsistent.

5 OWL-based Constraint Checking

Despite the fact that OntoClean is the single most well-known and theoretically founded methodology for evaluating the formal correctness of subsumption hierarchies, there has always been a lot of criticism regarding the high costs for tagging and constraint checking. To address this criticism and to make the OntoClean methodology more easily applicable in practical ontology engineering settings, Welty suggested an OWL-based formalization of metaproperty assignments and constraints [14], which leverages logical inconsistencies as indicators of constraint violations. We discuss this framework next and then introduce our novel metamodeling in this setting.

5.1 The Original OntoClean Metaontology

Welty's formalization, occasionally referred to as *OntOWLClean*, axiomatizes the aforementioned OntoClean constraints as domain-range restrictions on a transitive object property `subClassOf`, which serves as a replacement for the normal subclass relation ($\sqsubseteq$) and enables a reification of every subsumption relationship in a given domain ontology. For example, instead of writing Apple $\sqsubseteq$ Food to state that Apple is a subclass of Food, we write `subClassOf`($o_{\mathsf{Apple}}, o_{\mathsf{Food}}$) introducing fresh individuals for the classes (e.g., o_{Apple} for Apple). Metaproperty assignments then become class membership assertions, e.g., we write `RigidClass`(o_{Apple}). Similarly, we can state that o_{Apple} is a class by adding the assertion `Class`(o_{Apple}). Note that in order to have true metamodeling one would have to state that `subClassOf` is the same as $\sqsubseteq$ and that `Class`

Table 4. Fragment of the OntoClean metaontology [14]

$$
\begin{aligned}
\texttt{RigidClass} &\sqsubseteq \texttt{Class}\\
\texttt{NonRigidClass} &\sqsubseteq \texttt{Class}\\
\texttt{AntiRigidClass} &\sqsubseteq \texttt{NonRigidClass}\\
\texttt{Class} &\equiv (\texttt{NonRigidClass} \sqcup \texttt{RigidClass}) \sqcap\\
&\quad\ (\texttt{NonDependentClass} \sqcup \texttt{DependentClass}) \sqcap\\
&\quad\ (\texttt{SortalClass} \sqcup \texttt{NonSortalClass}) \sqcap\\
&\quad\ (\texttt{UnityClass} \sqcup \texttt{NonUnityClass})\\
\texttt{NonRigidClass} \sqcap \texttt{RigidClass} &\sqsubseteq \bot\\
\texttt{RigidClass} &\sqsubseteq \forall \texttt{subClassOf}.\neg\texttt{AntiRigidClass}\\
&\ldots
\end{aligned}
$$

really implies that its instances are classes. This is possible in OWL Full, but not in OWL DL. Since the reasoning support for OWL Full is limited, the axioms required to equate `subClassOf` and `Class` with their OWL modeling constructs are available as a complementary OWL Full ontology, which is kept separate from the core of the OntOWLClean ontology.[4] Table 4 shows an excerpt from the OWL DL part of the ontology focussing on the axioms for rigidity.

Note that if we now state that Apple is rigid and Food is anti-rigid the metaontology becomes inconsistent as witnessed by the following set of axioms:

$$
\begin{aligned}
\texttt{RigidClass} \sqsubseteq \forall \texttt{subClassOf}.\neg\texttt{AntiRigidClass} \qquad & \texttt{RigidClass}(o_{\mathsf{Apple}})\\
\texttt{subClassOf}(o_{\mathsf{Apple}}, o_{\mathsf{Food}}) \qquad & \texttt{AntiRigidClass}(o_{\mathsf{Food}})
\end{aligned}
$$

OntOWLClean has been a great step forward when it comes to the practical applicability of the OntoClean methodology, because it enables the use of standard DL reasoning for detecting constraint violations (see also [11]). However, the axiomatization of metaproperty assignments and constraints suggested by Welty has at least two drawbacks which can be overcome now that OWL 2 is available:

First, while syntactically metaproperty assignments and constraints could be part of the same ontology as classes and subsumption axioms, there is no semantic link between a class and its corresponding individual (e.g., the class Apple and the individual o_{Apple} being a member of `RigidClass` in the metaontology). Hence without the OWL Full part of the axiomatization, OntOWLClean does not allow for integrated reasoning over classes and their metaproperties. Furthermore, any changes to the subsumption hierarchy (in case of constraint violations, for example) involve modifications of two logically unrelated taxonomies, possibly maintained in two different files.

Second, the computational costs of determining the reasons for logical inconsistencies and thus constraint violations can be very high. The typical way of debugging an inconsistent ontology is to compute minimal subsets of the ontological axioms, which preserve the inconsistency. These subsets are called *explanations*, *justifications*, or *minAs*. In order to compute the explanations, axioms are removed from the original ontology

[4] Note that the last axiom in Table 4 is specified as an equivalence in http://www.ontoclean.org/ontoclean-dl-v1.owl. We assumed this to be a mistake as equivalence is not used for modeling any of the constraints in Welty's paper [14], and corrected the ontology accordingly.

Table 5. An explanation for the conflict caused by Apple being rigid and Food being anti-rigid

$$
\begin{aligned}
\mathsf{Apple} &\sqsubseteq \mathsf{Food}\\
\mathtt{RigidClass} &\sqsubseteq \forall \mathtt{subClassOf}.(\neg \mathtt{AntiRigidClass})\\
\exists \mathtt{type}.\top &\sqsubseteq \mathtt{Inst}\\
\mathsf{Apple} &\equiv \exists \mathtt{type}.\{o_{\mathsf{Apple}}\}\\
\mathsf{Food} &\equiv \exists \mathtt{type}.\{o_{\mathsf{Food}}\}\\
\mathtt{Class} \sqcap \forall \mathtt{type}^{-}.\exists \mathtt{type}.\{o_{\mathsf{Food}}\} &\equiv \mathtt{Class} \sqcap \exists \mathtt{subClassOf}.\{o_{\mathsf{Food}}\}\\
&\quad \mathtt{RigidClass}(o_{\mathsf{Apple}})\\
&\quad \mathtt{AntiRigidClass}(o_{\mathsf{Food}})\\
&\quad \mathtt{Class}(o_{\mathsf{Apple}})
\end{aligned}
$$

in a step by step manner, while after each removal a reasoner is used to check whether the remaining set of axioms is still inconsistent. This process is repeated until no further axiom can be removed without turning the ontology consistent. Users presented with the explanations can then decide how to fix the ontology. In particular computing *all* such explanations is a computationally hard task. Due to the high costs, only limited tool support for inconsistency diagnosis in OWL is available.

5.2 Towards OntOWL2Clean

In order to address the first issue, we extend $\mathcal{O}^{\text{meta}}$ from Table 3 by a set of classes and axioms which enable us to express all of the OntoClean metaproperty assignments and constraints. In particular, we have the following axioms for the constraints:

$$
\begin{aligned}
\mathtt{RigidClass} &\sqsubseteq \forall \mathtt{subClassOf}.\neg \mathtt{AntiRigidClass} && \text{(C1)}\\
\mathtt{UnityClass} &\sqsubseteq \forall \mathtt{subClassOf}.\neg \mathtt{AntiUnityClass} && \text{(C2)}\\
\mathtt{SortalClass} &\sqsubseteq \forall \mathtt{subClassOf}^{-}.\mathtt{SortalClass} && \text{(C3)}\\
\mathtt{DependentClass} &\sqsubseteq \forall \mathtt{subClassOf}^{-}.\mathtt{DependentClass} && \text{(C4)}
\end{aligned}
$$

We can now add the OntoClean taggings to the classes making use of the class individuals in $\mathcal{O}^{\text{meta}}$. For example, since we assume Food to be anti-rigid while Apple is rigid, we add the following facts to $\mathcal{O}^{\text{meta}}$:

$$\mathtt{AntiRigidClass}(o_{\mathsf{Food}}) \qquad \mathtt{RigidClass}(o_{\mathsf{Apple}})$$

Note that we do not have to add $\mathtt{subClassOf}(o_{\mathsf{Apple}}, o_{\mathsf{Food}})$ explicitly since the subClassOf role between o_{Apple} and o_{Food} is implied in $\mathcal{O}^{\text{meta}}$. Since constraint (C1) prevents an anti-rigid class from being a subclass of a rigid one, the ontology becomes inconsistent, witnessed by the explanation shown in Table 5.

One should be aware that adding constraints might have a "backward" impact on the semantics of the "original part" of the ontology in that they could rule out certain models thereby leading to additional consequences. To see this, assume our ontology has been corrected by removing $\mathsf{Apple} \sqsubseteq \mathsf{Food}$. Axiom (C1) still enforces that Apple must not be a subclass of Food and we have as a consequence that the extension of Apple must be nonempty in every model. This is because the empty set is trivially a subset of every set and, in particular, a subset of the extension of Food. Depending on the concrete scenario, these ramifications might be unwanted or intended. They can be avoided by using the approach based on marker predicates described next.

6 Marker Predicates for Pinpointing Constraint Violations

The above method implements Welty's approach of specifying the constraints in a way that their violation results in an inconsistent knowledge base. In order to actually find and identify the reasons for these inconsistencies, diagnosis techniques [7, 5] have to be employed. Typically these diagnosis techniques are rather costly as they require numerous calls to a reasoning system.

We argue that in certain cases, an alternative approach can be employed, wherein violations of OntoClean constraints do not cause the ontology to become inconsistent but lead to the creation of *marker classes* or *roles* that indicate which ontology elements are involved in a constraint violation. This alternative method can be combined with the original two-ontology approach as well as with our metamodeling technique.

Consider a constraint that prohibits $\mathsf{C} \sqsubseteq \mathsf{D}$ whenever C is endowed with the metaproperty $\mathtt{T}_1$ and D is endowed with $\mathtt{T}_2$. By specifying the axiom $\mathtt{T}_1 \sqsubseteq \forall \mathtt{subClassOf}.\neg\mathtt{T}_2$, we would turn an ontology inconsistent whenever it entails $\mathtt{T}_1(o_\mathsf{C})$, $\mathtt{T}_2(o_\mathsf{D})$, and $\mathtt{subClassOf}(o_\mathsf{C}, o_\mathsf{D})$. Consequently, diagnosis would be required to locate the violated constraint. Instead, we propose to establish an auxiliary marker role `conflictsWith` between o_C and o_D in this case. Thereby, all conflicts can be readily spotted by simply retrieving all entailed `conflictsWith` role memberships. This wanted correspondence can be logically enforced in OWL 2 using an encoding introduced independently in [8] and [2] which makes use of additional auxiliary roles t_1, t_2 as well as some of the advanced features of $\mathcal{SROIQ}$:

$$\mathtt{T}_1 \sqsubseteq \exists t_1.\mathsf{Self} \qquad \mathtt{T}_2 \sqsubseteq \exists t_2.\mathsf{Self} \qquad t_1 \circ \mathtt{subClassOf} \circ t_2 \sqsubseteq \mathtt{conflictsWith}$$

In order to axiomatize the OntoClean constraints, we introduce a fresh role `mp` for each OntoClean metaproperty `mpClass` (e.g., `rigid` for `RigidClass`) and an axiom

$$\mathtt{mpClass} \sqsubseteq \exists \mathtt{mp}.\mathsf{Self} \tag{M1}$$

where `mp` is the fresh role associated with `mpClass`. We can then axiomatize the four OntoClean constraints (C1) to (C4) with the following role chain axioms, where we use one marker per conflict type:

$$\mathtt{rigid} \circ \mathtt{subClassOf} \circ \mathtt{antiRigid} \sqsubseteq \mathtt{rigidityConflict} \tag{M2}$$

$$\mathtt{unity} \circ \mathtt{subClassOf} \circ \mathtt{antiUnity} \sqsubseteq \mathtt{unityConflict} \tag{M3}$$

$$\mathtt{nonDependent} \circ \mathtt{subClassOf} \circ \mathtt{dependent} \sqsubseteq \mathtt{dependencyConflict} \tag{M4}$$

$$\mathtt{nonSortal} \circ \mathtt{subClassOf} \circ \mathtt{sortal} \sqsubseteq \mathtt{sortalConflict} \tag{M5}$$

For each role `r` on the right-hand side of Axioms (M2) to (M5)

$$\mathtt{r} \sqsubseteq \mathtt{conflictsWith} \tag{M6}$$

Note that for (C3) and (C4) we use an equivalent formulation which is better suited for using the marker properties. In order to allow for retrieving all conflicts at once, we further introduce `conflictsWith` as a superrole of all the roles on the right-hand side of the above axioms.

7 Evaluation

We used the OntoClean example ontology [4] to test the different approaches. This leaves us with four settings: we first test Welty's metamodeling (see Section 5) and our metamodeling (see Section 3) with Explanations for discovering the modeling mistakes in settings Ex1 and Ex2, then we test the two approaches with the new Marker predicates (see Section 6) in settings Ma1 and Ma2.

Ex1 Our baseline is the metamodeling part of the example OntoClean ontology, i.e., we use the metamodeling as proposed by Welty. It is worth noting that we use the weakened OWL DL version of the original OWL Full meta ontology to be able to use OWL DL reasoners. For each axiom, e.g., $\mathsf{Apple} \sqsubseteq \mathsf{Food}$ in the original ontology, the meta version contains an assertion $\texttt{subClassOf}(o_{\mathsf{Apple}}, o_{\mathsf{Food}})$ with o_{Apple} and o_{Food} individuals and `subClassOf` a role. Furthermore, the meta ontology contains the taggings such as $\texttt{RigidClass}(o_{\mathsf{Apple}})$ and the OntoClean constraints from Axioms (C1) to (C4).

Ex2 The second ontology is the metamodeling version of the OntoClean ontology according to our novel metamodeling technique. Since we have no separation between the metamodeling part and the original axioms, the ontology contains assertions of either kind: adjusted axioms from the source ontology as well as the taggings and the OntoClean constraints from Axioms (C1) to (C4).

Ma1 In this setting, we use the marker predicates to manifest modeling errors instead of inconsistencies. The ontology uses Welty's metamodeling as in setting Ex1 and contains the taggings, but instead of causing inconsistencies by adding Axioms (C1) to (C4), we use marker predicates as described in Section 6 and add Axioms (M1) to (M6).

Ma2 The last setting uses our novel metamodeling approach in combination with the marker predicates, i.e., we use an ontology as in setting Ex2, but instead of causing inconsistencies by adding Axioms (C1) to (C4), we again use marker predicates and add Axioms (M1) to (M6).

In order to find all potential modeling errors in the settings Ex1 and Ex2, we use the explanation framework by Horridge et al. [5] and we generate all minimal subsets $\mathcal{O}'$ of the ontology such that $\mathcal{O}'$ is inconsistent. For the settings Ma1 and Ma2, we retrieve instances of the roles that indicate a conflict. All tests have been performed using the OWL 2 DL reasoner HermiT. The ontologies, HermiT 1.2.4, the obtained results, and the program used to produce the results are available online.[5] The tests have been performed on a MacBook Air with Java 1.6, assigning 1GB memory to Java.

Both explanation approaches ran out of memory after 2.5 days, generating 51 explanations for setting Ex1 and 46 explanations for setting Ex2, making the approach not really feasible in practice. Although the first explanations are generated quickly, the later ones can take significant time and memory. We repeated the tests of setting Ex1 and Ex2 on a node of the Oxford Supercomputing Centre, assigning 24GB of main memory to Java. We terminated the programs after one week, getting 53 explanations for setting Ex1 and 66 for setting Ex2. By analyzing the ontology manually, we find that

[5] http://www.hermit-reasoner.com/2010/metamodeling/metamodeling.zip

Table 6. Time in seconds for retrieving instances of the marker properties

	conflict				
setting	rigidity	unity	dependency	sortal	all
Ma1	< 1	< 1	< 1	< 1	< 1
Ma2	21	355	20	56	452

there should be 53 explanations for setting Ex1 and we assume that the code attempted to find more explanations without success.

There are more explanations for the new metamodeling approach since an explanation might contain the meta axioms for the involved classes only partially. In such a case, the explanation contains additional axioms that are not directly related to the conflict, but which contain enough meta axioms to cause the clash for the only partially axiomatized real inconsistency cause.

The novel marker approaches (Ma1 and Ma2) both find 40 conflicts: 10 rigidity conflicts, 16 unity conflicts, 12 dependency conflicts, and 2 identity conflicts. The timings are given in Table 6 and were averaged over 3 runs of the reasoner. It can be observed that the times for our new metamodeling approach are significantly slower than the ones for the original approach. This is a consequence of the more complex axiomatization that is required in order to achieve real metamodeling in OWL 2 DL, whereas in setting Ma1, the reasoner only works on a part of the ontology that suffices to detect these conflicts. Full metamodeling in the settings Ex1 and Ma1 requires an OWL Full reasoner and, as Welty states [14], a satisfactory implementation that can handle the full meta ontology is not (yet) available.

The number of marker conflicts is lower than the number of explanations because for several indirect subclass relationships, there are different ways of deriving the subsumption. E.g., the ontology contains:

$$\mathsf{Organisation} \sqsubseteq \mathsf{SocialEntity} \qquad \mathsf{SocialEntity} \sqsubseteq \mathsf{Agent}$$
$$\mathsf{Organisation} \sqsubseteq \mathsf{LegalAgent} \qquad \mathsf{LegalAgent} \sqsubseteq \mathsf{Agent}$$

In both cases, we have that Organisation is a subclass of Agent. Since we further have that Organisation is a rigid class, while Agent is anti-rigid, we have one explanation using the first set of subclass axioms and another explanation using the second set of subclass axioms, whereas the marker approach does not distinguish the two cases.

8 Conclusion

We have presented a novel approach to ontology-inherent metamodeling for classes in OWL based on an axiomatization of class reification. This approach allows for associating information to classes and asserting constraints on the subclass hierarchy in a way that allows for the usage of standard OWL reasoning tools. We demonstrated our approach by applying it to the OntoClean methodology. We found that the benefits of our approach in terms of maintenance and tight logical integration may come at the cost of runtime performance. On the other hand, we showed that performance can be increased

by several orders of magnitude if the explanation-based diagnosis originally proposed for OntoClean is substituted by a novel consistency-preserving approach working with marker predicates that indicate potential modeling flaws in the ontology. The runtime improvements thus obtained outweigh the metamodeling-induced slowdown by far.

Acknowledgements Birte Glimm is funded by the EPSRC project HermiT: Reasoning with Large Ontologies. Sebastian Rudolph is supported by the German Research Foundation (DFG) under the ExpresST project. Johanna Völker is financed by a Margarete-von-Wrangell scholarship of the European Social Fund (ESF) and the Ministry of Science, Research and the Arts Baden-Württemberg. The evaluation has been performed on computers of the Oxford Supercomputing Centre.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2007)
2. Gasse, F., Sattler, U., Haarslev, V.: Rewriting rules into $\mathcal{SROIQ}$ axioms. Poster at 21st International Workshop on Description Logics (DL) (2008)
3. Glimm, B., Rudolph, S., Völker, J.: Integrated metamodeling and diagnosis in owl 2. Tech. Rep. 3006, Institut AIFB, KIT, Karlsruhe (September 2010), available at http://www.aifb.kit.edu/web/Techreport3006
4. Guarino, N., Welty, C.A.: An Overview of OntoClean, pp. 201–220. International Handbook on Information Systems, Springer, 2nd edn. (2009)
5. Horridge, M., Parsia, B., Sattler, U.: Explaining inconsistencies in OWL ontologies. In: 3rd International Conference on Scalable Uncertainty Management (SUM) (2009)
6. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006). pp. 57–67. AAAI Press (2006)
7. Ji, Q., Haase, P., Qi, G., Hitzler, P., Stadtmüller, S.: RaDON – Repair and diagnosis in ontology networks. In: Proceedings of the European Semantic Web Conference (ESWC). pp. 863–867. Springer (2009)
8. Krötzsch, M., Rudolph, S., Hitzler, P.: Description logic rules. In: Ghallab, M., Spyropoulos, C.D., Fakotakis, N., Avouris, N.M. (eds.) ECAI. Frontiers in Artificial Intelligence and Applications, vol. 178, pp. 80–84. IOS Press (2008)
9. Motik, B.: On the properties of metamodeling in OWL. Journal of Logic and Computation 17(4), 617–637 (2007)
10. Tran, T., Haase, P., Motik, B., Grau, B.C., Horrocks, I.: Metalevel information in ontology-based applications. In: Fox, D., Gomes, C.P. (eds.) AAAI. pp. 1237–1242. AAAI Press (2008)
11. Völker, J., Vrandečić, D., Sure, Y., Hotho, A.: AEON – An approach to the automatic evaluation of ontologies. Journal of Applied Ontology 3(1-2), 41–62 (2008)
12. Vrandečić, D.: Ontology Evaluation, pp. 293–313. International Handbook on Information Systems, Springer, 2nd edn. (2009)
13. Welty, C., Mahindru, R., Chu-Carroll, J.: Evaluating ontology cleaning. In: McGuinness, D.L., Ferguson, G. (eds.) Proc. 19th National Conf. on AI (AAAI) and 16th Conf. Innovative Applications of AI (IAAI). MIT Press (July 2004)
14. Welty, C.: OntOWLClean: Cleaning OWL ontologies with OWL. In: Bennet, B., Fellbaum, C. (eds.) Proceedings of Formal Ontologies in Information Systems (FOIS). pp. 347–359. IOS Press, Amsterdam, The Netherlands (2006)