Reasoning for $\mathcal{ALCQ}$ Extended with a Flexible Meta-Modelling Hierarchy

Regina Motz[2], Edelweis Rohrer[2], and Paula Severi[1(✉)]

[1] Department of Computer Science, University of Leicester, Leicester, England
ps330@leicester.ac.uk
[2] Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay
{rmotz,erohrer}@fing.edu.uy

Abstract. This works is motivated by a real-world case study where it is necessary to integrate and relate existing ontologies through *meta-modelling*. For this, we introduce the Description Logic $\mathcal{ALCQM}$ which is obtained from $\mathcal{ALCQ}$ by adding statements that equate individuals to concepts in a knowledge base. In this new extension, a concept can be an individual of another concept (called *meta-concept*) which itself can be an individual of yet another concept (called *meta meta-concept*) and so on. We define a tableau algorithm for checking consistency of an ontology in $\mathcal{ALCQM}$ and prove its correctness.

Keywords: Description logic · Meta-modelling · Meta-concepts · Well founded sets · Consistency · Decidability

1 Introduction

Our extension of $\mathcal{ALCQ}$ is motivated by a real-world application on geographic objects that requires to reuse existing ontologies and relate them through meta-modelling [10].

Figure 1 describes a simplified scenario of this application in order to illustrate the meta-modelling relationship. It shows two ontologies separated by a line. The two ontologies conceptualize the same entities at different levels of granularity. In the ontology above the line, rivers and lakes are formalized as individuals while in the one below the line they are concepts. If we want to integrate these ontologies into a single ontology (or into an ontology network) it is necessary to interpret the individual *river* and the concept *River* as the same real object. Similarly for *lake* and *Lake*.

Our solution consists in equating the individual *river* to the concept *River* and the individual *lake* to the concept *Lake*. These equalities are called *meta-modelling axioms* and in this case, we say that the ontologies are related through *meta-modelling*. In Figure 1, meta-modelling axioms are represented by dashed edges. After adding the meta-modelling axioms for rivers and lakes, the concept *HydrographicObject* is now also a *meta-concept* because it is a concept that contains an individual which is also a concept.

T. Supnithi et al.(Eds.): JIST 2014, LNCS 8943, pp. 47–62, 2015.
DOI: 10.1007/978-3-319-15615-6_4

The kind of meta-modelling we consider in this paper can be expressed in OWL Full but it cannot be expressed in OWL DL. The fact that it is expressed in OWL Full is not very useful since the meta-modelling provided by OWL Full is so expressive that leads to undecidability [11].

OWL 2 DL has a very restricted form of meta-modelling called *punning* where the same identifier can be used as an individual and as a concept [7]. These identifiers are treated as different objects by the reasoner and it is not possible to detect certain inconsistencies. We next illustrate two examples where OWL would not detect inconsistencies because the identifiers, though they look syntactically equal, are actually different.

Example 1. If we introduce an axiom expressing that *HydrographicObject* is a subclass of *River*, then OWL's reasoner will not detect that the interpretation of *River* is not a well founded set (it is a set that belongs to itself).

Example 2. We add two axioms, the first one says that *river* and *lake* as individuals are equal and the second one says that the classes *River* and *Lake* are disjoint. Then OWL's reasoner does not detect that there is a contradiction.

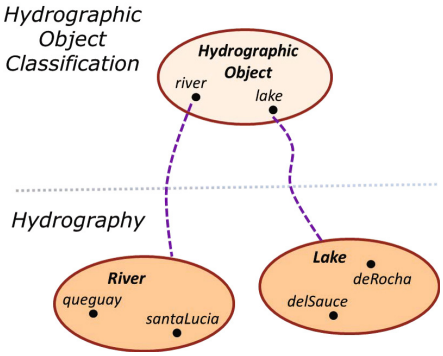

Fig. 1. Two ontologies on Hydrography

In this paper, we consider $\mathcal{ALCQ}$ ($\mathcal{ALC}$ with qualified cardinality restrictions) and extend it with *Mboxes*. An Mbox is a set of equalities of the form $a =_{\mathsf{m}} A$ where a is an individual and A is a concept. In our example, we have that $river =_{\mathsf{m}} River$ and these two identifiers are semantically equal, i.e. the interpretations of the individual *river* and the concept *River* are the same. The domain of an interpretation cannot longer consists of only basic objects but it must be any well-founded set. The well-foundness of our model is not ensured by means of fixing layers beforehand as in [8,12] but it is our reasoner which checks for circularities. Our approach allows the user to have any number of levels (or layers) (meta-concepts, meta meta-concepts and so on). The user does not have to write or know the layer of the concept because the reasoner will infer it for him. In this way, axioms can also naturally mix elements of different layers and the user has the flexibility of changing the status of an individual at any point without having to make any substantial change to the ontology.

We define a tableau algorithm for checking consistency of an ontology in $\mathcal{ALCQM}$ by adding new rules and a new condition to the tableau algorithm for $\mathcal{ALCQ}$. The new rules deal with the equalities and inequalities between individuals with meta-modelling which need to be transferred to the level of concepts as equalities and inequalities between the corresponding concepts. The new condition deals with circularities avoiding non well-founded sets. From the practical point of view, extending tableau for $\mathcal{ALCQ}$ has the advantage that one can easily change and reuse the code of existing OWL's reasoners. From the theoretical point of view, we give an elegant proof of correctness by showing an isomorphism between the canonical interpretations of $\mathcal{ALCQ}$ and $\mathcal{ALCQM}$. Instead of re-doing inductive proofs, we "reuse" and invoke the results of correctness of the tableau algorithm for $\mathcal{ALCQ}$ from [1] wherever possible.

Related Work. As we mentioned before, OWL 2 DL has a very restricted form of meta-modelling called *punning* [7]. In spite of the fact that the same identifier can be used simultaneously as an individual and as a concept, they are semantically different. In order to use the punning of OWL 2 DL in the example of Figure 1, we could change the name *river* to *River* and *lake* to *Lake*. In spite of the fact that the identifiers look syntactically equal, OWL would not detect certain inconsistencies as the ones illustrated in Examples 1 and 2, and in Example 4 which appears in Section 3. In the first example, OWL won't detect that there is a circularity and in the other examples, OWL won't detect that there is a contradiction. Apart from having the disadvantage of not detecting certain inconsistencies, this approach is not natural for reusing ontologies. For these scenarios, it is more useful to assume the identifiers be syntactically different and allow the user to equate them by using axioms of the form $a =_{\mathsf{m}} A$.

Motik proposes a solution for meta-modelling that is not so expressive as RDF but which is decidable [11]. Since his syntax does not restrict the sets of individuals, concepts and roles to be pairwise disjoint, an identifier can be used as a concept and an individual at the same time. From the point of view of ontology design, we consider more natural to assume that the identifiers for a concept and an individual that conceptualize the same real object (with different granularity) will be syntactically different (because most likely they will live in different ontologies). In [11], Motik also defines two alternative semantics: the context approach and the HiLog approach. The context approach is similar to the so-called punning supported by OWL 2 DL. The HiLog semantics looks more useful than the context semantics since it can detect the inconsistency of Example 2. However, this semantics ignores the issue on well-founded sets. Besides, this semantics does not look either intuitive or direct as ours since it uses some intermediate extra functions to interpret individuals with meta-modelling. The algorithm given in [11, Theorem 2] does not check for circularities (see Example 1) which is one of the main contributions of this paper.

De Giacomo et al. specifies a new formalism, "Higher/Order Description Logics", that allows to treat the same symbol of the signature as an instance, a concept and a role [4]. This approach is similar to punning in the sense that the three new symbols are treated as independent elements.

Pan et al address meta-modelling by defining different "layers" or "strata" within a knowledge base [8,12]. This approach forces the user to explicitly write the information of the layer in the concept. This has several disadvantages: the user should know beforehand in which layer the concept lies and it does not give the flexibility of changing the layer in which it lies. Neither it allows us to mix different layers when building concepts, inclusions or roles, e.g. we cannot express that the intersection of concepts in two different layers is empty or define a role whose domain and range live in different layers.

Glimm et al. codify meta-modelling within OWL DL [5]. This codification consists in adding some extra individuals, axioms and roles to the original ontology in order to represent meta-modelling of concepts. As any codification, this approach has the disadvantage of being involved and difficult to use, since adding new concepts implies adding a lot of extra axioms. This codification is not enough for detecting inconsistencies coming from meta-modelling (see Example 4). The approach in [5] has also other limitations from the point of view of expressibility, e.g. it has only two levels of meta-modelling (concepts and meta-concepts).

Organization of the Paper. The remainder of this paper is organized as follows. Section 2 shows a case study and explains the advantages of our approach. Section 3 defines the syntax and semantics of $\mathcal{ALCQM}$. Section 4 proposes an algorithm for checking consistency. Section 5 proves its correctness. Finally, Section 6 sets the future work.

2 Case Study on Geography

In this section, we illustrate some important advantages of our approach through the real-world example on geographic objects presented in the introduction.

Figure 2 extends the ontology network given in Figure 1. Ontologies are delimited by light dotted lines. Concepts are denoted by ovals and individuals by small filled circles. Meta-modelling between ontologies is represented by dashed edges. Thinnest arrows denote roles within a single ontology while thickest arrows denote roles from one ontology to another ontology.

Figure 2 has five separate ontologies. The ontology in the uppermost position conceptualizes the politics about geographic objects, defining *GeographicObject* as a meta meta-concept, and *Activity* and *GovernmentOffice* as concepts. The ontology in the left middle describes hydrographic objects through the meta-concept *HydrographicObject* and the one in the right middle describes flora objects through the meta-concept *FloraObject*. The two remaining ontologies conceptualize the concrete natural resources at a lower level of granularity through the concepts *River*, *Lake*, $Wetland$ and $NaturalForest$.

Note that horizontal dotted lines in Figure 2 do not represent meta-modelling levels but just ontologies. The ontology "Geographic Object Politics" has the meta meta-concept *GeographicObject*, whose instances are concepts which have also instances being concepts, but we also have the concepts *GovernmentOfice* and *Activity* whose instances conceptualize atomic objects. OWL has only one

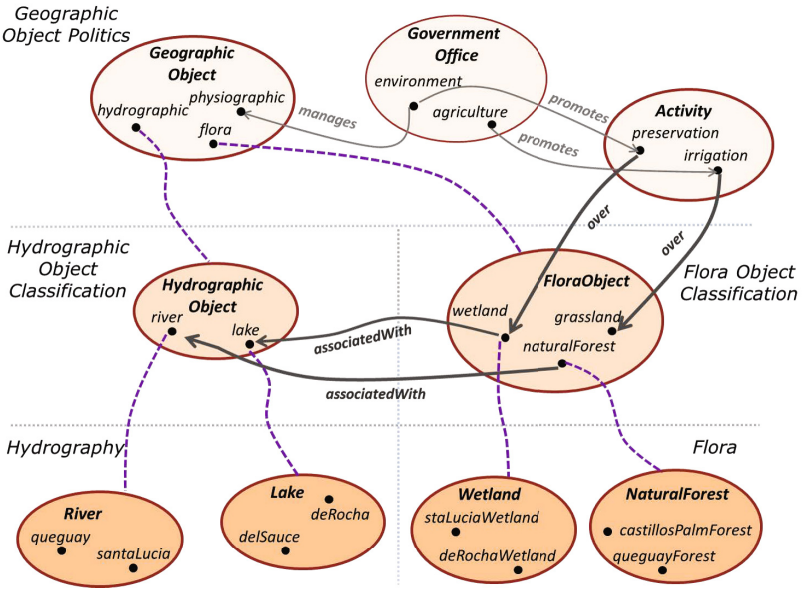

Fig. 2. Case Study on Geography

notion of hierarchy which classifies concepts with respect to the inclusion $\sqsubseteq$. Our approach has a new notion of hierarchy, called *meta-modelling hierarchy*, which classifies concepts with respect to the membership relation $\in$. The meta-modelling hierarchy for the concepts of Figure 2 is depicted in Figure 3. The concepts are *GovernmentOffice*, *Activity*, *River*, *Lake*, *Wetland* and *NaturalForest*, the meta-concepts are *HydrographicObject* and *FloraObject*, and the meta meta-concept is *GeographicObject*.

The first advantage of our approach over previous work concerns the reuse of ontologies when the same conceptual object is represented as an individual in one ontology and as a concept in the other. The identifiers for the individual and the concept will be syntactically different because they belong to different ontologies (with different URIs). Then, the ontology engineer can introduce an equation between these two different identifiers. This contrasts with previous approaches where one has to use the same identifier for an object used as a concept and as an individual. In Figure 2, *river* and *River* represent the same real object. In order to detect inconsistency and do the proper inferences, one has to be able to equate them.

The second advantage is about the flexibility of the meta-modelling hierarchy. This hierarchy is easy to change by just adding equations. This is illustrated in the passage from Figure 1 to Figure 2. Figure 1 has a very simple meta-modelling hierarchy where the concepts are *River* and *Lake* and the meta-concept is *HydrographicObject*. The rather more complex meta-modelling hierarchy for the ontology of Figure 2 (see Figure 3) has been obtained by combining the ontologies of Figure 1 with other ontologies and by simply adding some few meta-modelling axioms.

After adding the meta-modelling equations, the change of the meta-modelling hierarchy is *automatic* and *transparent* to the user. Concepts such as *GeographicObject* will automatically pass to be meta meta-concepts and roles such as *associatedWith* will automatically pass to be meta-roles, i.e. roles between meta-concepts.

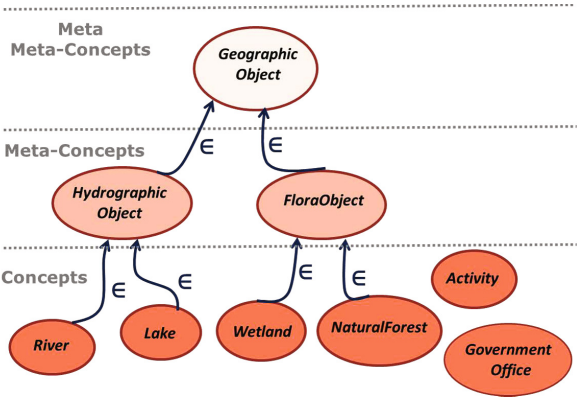

Fig. 3. Meta-modelling Hierarchy for the Ontology of Figure 2

The third advantage is that we do not have any restriction on the level of meta-modelling, i.e. we can have concepts, meta-concepts, meta meta-concepts and so on. Figure 1 has only one level of meta-modelling since there are concepts and meta-concepts. In Figure 2, there are two levels of meta-modelling since it has concepts, meta-concepts and meta meta-concepts. If we needed, we could extend it further by adding the equation $santaLucia =_{\mathsf{m}} SantaLucia$ for some concept *SantaLucia* and this will add a new level in the meta-modelling hierarchy: concepts, meta-concepts, meta meta-concepts and meta meta meta-concepts.

Moreover, the user does not have to know the meta-modelling levels, they are transparent for him. Our algorithm detects inconsistencies without burdening the user with syntactic complications such as having to explicitly write the level the concept belongs to.

The fourth advantage is about the possibility of mixing levels of meta-modelling in the definition of concepts and roles. We can build concepts using union or intersection between two concepts of different levels (layers). We can also define roles whose domain and range live in different levels (or layers). For example, in Figure 2, we have: 1) a role *over* whose domain is just a concept while the range is a meta-concept, 2) a role *manages* whose domain is just a concept and whose range is a meta meta-concept. We can also add axioms to express that some of these concepts, though at different levels of meta-modelling, are disjoint, e.g. the intersection of the concept *Activity* and the meta-concept *FloraObject* is empty.

3 $\mathcal{ALCQM}$

In this section we introduce the $\mathcal{ALCQM}$ Description Logics (DL), with the aim of expressing meta-modelling in a knowledge base. The syntax of $\mathcal{ALCQM}$ is obtained from the one of $\mathcal{ALCQ}$ by adding new statements that allow us to equate individuals with concepts. The definition of the semantics for $\mathcal{ALCQM}$ is the key to our approach. In order to detect inconsistencies coming from meta-modelling, a proper semantics should give *the same interpretation* to individuals and concepts which have been equated through meta-modelling.

Recall the formal syntax of $\mathcal{ALCQ}$ [2,7]. We assume a finite set of atomic individuals, concepts and roles. If A is an atomic concept and R is a role, the concept expressions C, D are constructed using the following grammar:
$C, D ::= A \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C \mid \geq nR.C \mid \leq nR.C$
Recall also that $\mathcal{ALCQ}$-statements are divided in two groups, namely TBox statements and ABox statements, where a TBox contains statements of the form $C \sqsubseteq D$ and an ABox contains statements of the form $C(a)$, $R(a, b)$, $a = b$ or $a \neq b$.
A *meta-modelling axiom* is a new type of statement of the form

$$a =_{\mathsf{m}} A \text{ where } a \text{ is an individual and } A \text{ is an atomic concept.}$$

which we pronounce as *a corresponds to A through meta-modelling*. An *Mbox* is a set $\mathcal{M}$ of meta-modelling axioms. We define $\mathcal{ALCQM}$ by keeping the same syntax for concept expressions as for $\mathcal{ALCQ}$ and extending it only to include MBoxes. An ontology or a knowledge base in $\mathcal{ALCQM}$ is denoted by $\mathcal{O} = (\mathcal{T}, \mathcal{A}, \mathcal{M})$ since it is determined by three sets: a Tbox $\mathcal{T}$, an Abox $\mathcal{A}$ and an Mbox $\mathcal{M}$. The set of all individuals with meta-modelling of an ontology is denoted by $\mathsf{dom}(\mathcal{M})$.

Figure 4 shows the $\mathcal{ALCQM}$-ontologies of Figure 1. In order to check for cycles in the tableau algorithm, it is convenient to have the restriction that A should be a concept name in $a =_{\mathsf{m}} A$. This restriction does not affect us in practice at all. If one would like to have $a =_{\mathsf{m}} C$ for a concept expression C, it is enough to introduce a concept name A such that $A \equiv C$ and $a =_{\mathsf{m}} A$.

Tbox

$River \sqcap Lake \sqsubseteq \bot$

Mbox

$river =_{\mathsf{m}} River$
$lake =_{\mathsf{m}} Lake$

Abox

$HydrographicObject(river)$	$HydrographicObject(lake)$
$River(queguay)$	$River(santaLucia)$
$Lake(deRocha)$	$Lake(delSauce)$

Fig. 4. The $\mathcal{ALCQM}$-ontology of Figure 1

Definition 1 (S_n for $n \in \mathbb{N}$). *Given a non empty set S_0 of atomic objects, we define S_n by induction on $\mathbb{N}$ as follows: $S_{n+1} = S_n \cup \mathcal{P}(S_n)$*

The sets S_n are clearly well-founded. Recall from Set Theory that *a relation R is well-founded on a class X* if every non-empty subset Y of X has a minimal element. Moreover, *a set X is well-founded* if the set membership relation is well-founded on the the set X.

Definition 2 (Model of an Ontology in $\mathcal{ALCQM}$). *An interpretation $\mathcal{I}$ is a* model of an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A}, \mathcal{M})$ in $\mathcal{ALCQM}$ *(denoted as $\mathcal{I} \models \mathcal{O}$) if the following holds:*

1. *the domain Δ of the interpretation is a subset of S_N for some $N \in \mathbb{N}$. The smallest N such that $\Delta \subseteq S_N$ is called the* level *of the interpretation $\mathcal{I}$.*
2. *$\mathcal{I}$ is a model of the ontology $(\mathcal{T}, \mathcal{A})$ in $\mathcal{ALCQ}$.*
3. *$\mathcal{I}$ is a model of $\mathcal{M}$, i.e. $\mathcal{I}$ satisfies each statement in $\mathcal{M}$. An interpretation $\mathcal{I}$ satisfies the statement $a =_{\mathsf{m}} A$ if $a^{\mathcal{I}} = A^{\mathcal{I}}$.*

Usually, the domain of an interpretation of an ontology is a set of atomic objects. In the first part of Definition 2 we redefine the domain Δ of the interpretation, so it does not consists only of atomic objects any longer. The domain Δ can now contain sets since the set S_N is defined recursively using the power-set operation. A similar notion of interpretation domain is defined in [9, Definition 1] for RDF ontologies.

It is sufficient to require that it is a subset of some S_N so it remains well-founded [1]. Note that S_0 does not have to be the same for all models of an ontology. The second part of Definition 2 refers to the $\mathcal{ALCQ}$-ontology without the Mbox axioms. In the third part of the definition, we add another condition that the model must satisfy considering the meta-modelling axioms. This condition restricts the interpretation of an individual that has a corresponding concept through meta-modelling to be equal to the concept interpretation.

Example 3. We define a model for the ontology of Figure 4 where

$$S_0 = \{queguay, santaLucia, deRocha, delSauce\}$$

Individuals and concepts equated through meta-modelling are semantically equal:

$$\begin{aligned} river^{\mathcal{I}} &= River^{\mathcal{I}} = \{queguay, santaLucia\} \\ lake^{\mathcal{I}} &= Lake^{\mathcal{I}} = \{deRocha, delSauce\} \end{aligned}$$

Definition 3 (Consistency of an Ontology in $\mathcal{ALCQM}$). *We say that an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A}, \mathcal{M})$ is* consistent *if there exists a model of $\mathcal{O}$.*

The $\mathcal{ALCQM}$-ontology defined in Figure 4 is consistent.

[1] In principle, non well-founded sets are not source of contradictions since we could work on non-well founded Set Theory. The reason why we exclude them is because we think that non well-founded sets do not occur in the applications we are interested in.

Example 4. We consider the ontology of Figure 2 and the axioms:

$$River \sqcap Lake \sqsubseteq \bot$$
$$Wetland \equiv NaturalForest$$

and the fact that $associatedWith$ is a functional property. Note that we have the following axioms in the Abox:

$$associatedWith(wetland, lake)$$
$$associatedWith(naturalForest, river)$$

As before, the $\mathcal{ALCQ}$-ontology (without the Mbox) is consistent. However, the $\mathcal{ALCQM}$-ontology (with the Mbox) is not consistent.

Example 1 illustrates the use of the first clause of Definition 2. Actually, this example is inconsistent because the first clause of this definition does not hold. Examples 2 and 4 illustrate how the second and third conditions of Definition 2 interact.

Definition 4 (Logical Consequence from an Ontology in $\mathcal{ALCQM}$). *We say that $\mathcal{S}$ is a* logical consequence of $\mathcal{O} = (\mathcal{T}, \mathcal{A}, \mathcal{M})$ *(denoted as $\mathcal{O} \models \mathcal{S}$) if all models of $\mathcal{O}$ are also models of $\mathcal{S}$ where $\mathcal{S}$ is any of the following $\mathcal{ALCQM}$-statements, i.e. $C \sqsubseteq D$, $C(a)$, $R(a, b)$, $a =_{\mathsf{m}} A$, $a = b$ and $a \neq b$.*

It is possible to infer new knowledge in the ontology with the meta-modelling that is not possible without it as illustrated by Examples 1, 2 and 4.

Definition 5 (Meta-concept). *We say that C is a meta concept in $\mathcal{O}$ if there exists an individual a such that $\mathcal{O} \models C(a)$ and $\mathcal{O} \models a =_{\mathsf{m}} A$.*

Then, C is a meta meta-concept if there exists an individual a such that $\mathcal{O} \models C(a)$, $\mathcal{O} \models a =_{\mathsf{m}} A$ and A is a meta-concept. Note that a meta meta-concept is also a meta-concept.
We have some new inference problems:

1. *Meta-modelling.* Find out whether $\mathcal{O} \models a =_{\mathsf{m}} A$ or not.
2. *Meta-concept.* Find out whether C is a meta-concept or not.

Most inference problems in Description Logic can be reduced to satisfiability by applying a standard result in logic which says that a formula ϕ is a semantic consequence of a set of formulas Γ if and only if $\Gamma \cup \neg\phi$ is not satisfiable. The above two problems can be reduced to satisfiability following this general idea. For the first problem, note that since $a \neq_m A$ is not directly available in the syntax, we have replaced it by $a \neq b$ and $b =_{\mathsf{m}} A$ which is an equivalent statement to the negation of $a =_{\mathsf{m}} A$ and can be expressed in $\mathcal{ALCQM}$.

Lemma 1. *$\mathcal{O} \models a =_{\mathsf{m}} A$ if and only if for some new individual b, $\mathcal{O} \cup \{a \neq b, b =_{\mathsf{m}} A\}$ is unsatisfiable.*

Lemma 2. *C is a meta-concept if and only if for some individual a we have that $\mathcal{O} \cup \{\neg C(a)\}$ is unsatisfiable and for some new individual b, $\mathcal{O} \cup \{a \neq b, b =_{\mathsf{m}} A\}$ is unsatisfiable.*

4 Checking Consistency of an Ontology in $\mathcal{ALCQM}$

In this section we will define a tableau algorithm for checking consistency of an ontology in $\mathcal{ALCQM}$ by extending the tableau algorithm for $\mathcal{ALCQ}$. From the practical point of view, extending tableau for $\mathcal{ALCQ}$ has the advantage that one can easily change and reuse the code of existing OWL's reasoners.

The tableau algorithm for $\mathcal{ALCQM}$ is defined by adding three expansion rules and a condition to the tableau algorithm for $\mathcal{ALCQ}$. The new expansion rules deal with the equalities and inequalities between individuals with meta-modelling which need to be transferred to the level of concepts as equalities and inequalities between the corresponding concepts. The new condition deals with circularities avoiding sets that belong to themselves and more generally, avoiding non well-founded sets.

Definition 6 (Cycles). *We say that the tableau graph $\mathcal{L}$ has a cycle with respect to $\mathcal{M}$ if there exist a sequence of meta-modelling axioms $A_0 =_{\mathsf{m}} a_0$, $A_1 =_{\mathsf{m}} a_1$, ... $A_n =_{\mathsf{m}} a_n$ all in $\mathcal{M}$ such that*

$$\begin{array}{ll} A_1 \in \mathcal{L}(x_0) & x_0 \approx a_0 \\ A_2 \in \mathcal{L}(x_1) & x_1 \approx a_1 \\ \vdots & \vdots \\ A_n \in \mathcal{L}(x_{n-1}) & x_{n-1} \approx a_{n-1} \\ A_0 \in \mathcal{L}(x_n) & x_n \approx a_n \end{array}$$

Example 5. Suppose we have an ontology $(\mathcal{T}, \mathcal{A}, \mathcal{M})$ with two individuals a and b, the individual assignments: $B(a)$ and $A(b)$; and the meta-modelling axioms:

$$a =_{\mathsf{m}} A \quad b =_{\mathsf{m}} B.$$

The tableau graph $\mathcal{L}(a) = \{B\}$ and $\mathcal{L}(b) = \{A\}$ has a cycle since $A \in \mathcal{L}(b)$ and $B \in \mathcal{L}(a)$.

Initialization for the $\mathcal{ALCQM}$-tableau is nearly the same as for $\mathcal{ALCQ}$. The nodes of the initial tableau graph will be created from individuals that occur in the Abox as well as in the Mbox. After initialization, the tableau algorithm proceeds by non-deterministically applying the **expansion rules** for $\mathcal{ALCQM}$. The expansion rules for $\mathcal{ALCQM}$ are obtained by adding the rules of Figure 5 to the expansion rules for $\mathcal{ALCQ}$.

We explain the intuition behind the new expansion rules. If $a =_{\mathsf{m}} A$ and $b =_{\mathsf{m}} B$ then the individuals a and b represent concepts. Any equality at the level of individuals should be transferred as an equality between concepts and similarly with the difference.

The $\approx$-rule transfers the equality $a \approx b$ to the level of concepts by adding two statements to the Tbox which are equivalent to $A \equiv B$. This rule is necessary to detect the inconsistency of Example 2 where the equality $river = lake$ is transferred as an equality $River \equiv Lake$ between concepts. A particular case of the application of the $\approx$-rule is when $a =_{\mathsf{m}} A$ and $a =_{\mathsf{m}} B$. In this case, the algorithm also adds $A \equiv B$.

$\approx$-rule: Let $a =_{\mathsf{m}} A$ and $b =_{\mathsf{m}} B$ in $\mathcal{M}$. If $a \approx b$ and $A \sqcup \neg B, B \sqcup \neg A$ does not belong to $\mathcal{T}$ then $\mathcal{T} \leftarrow A \sqcup \neg B, B \sqcup \neg A$.

$\not\approx$-rule: Let $a =_{\mathsf{m}} A$ and $b =_{\mathsf{m}} B$ in $\mathcal{M}$. If $a \not\approx b$ and there is no z such that $A \sqcap \neg B \sqcup B \sqcap \neg A \in \mathcal{L}(z)$ then create a new node z with $\mathcal{L}(z) = \{A \sqcap \neg B \sqcup B \sqcap \neg A\}$.

close-rule: Let $a =_{\mathsf{m}} A$ and $b =_{\mathsf{m}} B$ where $a \approx x$, $b \approx y$, $\mathcal{L}(x)$ and $\mathcal{L}(y)$ are defined. If neither $x \approx y$ nor $x \not\approx y$ are set then $\mathsf{equate}(a, b, \mathcal{L})$ or $\mathsf{differenciate}(a, b, \mathcal{L})$.

Fig. 5. Additional Expansion Rules for $\mathcal{ALCQM}$

The $\not\approx$-rule is similar to the $\approx$-rule. However, in the case that $a \not\approx b$, we cannot add $A \not\equiv B$ because the negation of $\equiv$ is not directly available in the language. So, what we do is to replace it by an equivalent statement, i.e. add an element z that witness this difference.

The rules $\approx$ and $\not\approx$ are not sufficient to detect all inconsistencies. With only these rules, we could not detect the inconsistency of Example 4. The idea is that we also need to transfer the equality $A \equiv B$ between concepts as an equality $a \approx b$ between individuals. However, here we face a delicate problem. It is not enough to transfer the equalities that are in the Tbox. We also need to transfer the semantic consequences, e.g. $\mathcal{O} \models A \equiv B$. Unfortunately, we cannot do $\mathcal{O} \models A \equiv B$. Otherwise we will be captured in a vicious circle [2] since the problem of finding out the semantic consequences is reduced to the one of satisfiability. The solution to this problem is to explicitly try either $a \approx b$ or $a \not\approx b$. This is exactly what the close-rule does. The close-rule adds either $a \approx b$ or $a \not\approx b$ through two new functions *equate* and *differenciate*. It is similar to the choose-rule which adds either C or $\neg C$. This works because we are working in Classical Logic and we have the law of excluded middle. For a model $\mathcal{I}$ of the ontology, we have that either $a^{\mathcal{I}} = b^{\mathcal{I}}$ or $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ (see also Lemma 5). Since the tableau algorithm works with canonical representatives of the $\approx$-equivalence classes, we have to be careful how we equate two individuals or make them different.
Note that the application of the tableau algorithm to an $\mathcal{ALCQM}$ knowledge base $(\mathcal{T}, \mathcal{A}, \mathcal{M})$ changes the Tbox as well as the tableau graph $\mathcal{L}$.

Definition 7 ($\mathcal{ALCQM}$-Complete). *$(\mathcal{T}, \mathcal{L})$ is $\mathcal{ALCQM}$-complete if none of the expansion rules for $\mathcal{ALCQM}$ is applicable.*

The algorithm terminates when we reach some $(\mathcal{T}, \mathcal{L})$ where either $(\mathcal{T}, \mathcal{L})$ is $\mathcal{ALCQM}$-complete, $\mathcal{L}$ has a contradiction or $\mathcal{L}$ has a cycle. The ontology $(\mathcal{T}, \mathcal{A}, \mathcal{M})$ is consistent if there exists some $\mathcal{ALCQM}$-complete $(\mathcal{T}, \mathcal{L})$ such that $\mathcal{L}$ has neither contradictions nor cycles. Otherwise it is inconsistent.

5 Correctness of the Tableau Algorithm for $\mathcal{ALCQM}$

In this section we prove termination, soundness and completeness for the tableau algorithm described in the previous section. We give an elegant proof of completeness by showing an isomorphism between the canonical interpretations of $\mathcal{ALCQ}$ and $\mathcal{ALCQM}$.

[2] Consistency is the egg and semantic consequence is the chicken.

Theorem 1 (Termination). *The tableau algorithm for $\mathcal{ALCQM}$ described in the previous section always terminates.*

Proof. Suppose the input is an arbitrary ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A}, \mathcal{M})$. We define

$$\mathsf{concepts}(\mathcal{M}) = \bigcup_{a=_{\mathsf{m}}A, b=_{\mathsf{m}}B} \{A \sqcap \neg B \sqcup B \sqcap \neg A, A \sqcup \neg B, B \sqcup \neg A\}$$

Suppose we have an infinite sequence of rule applications:

$$(\mathcal{T}_0, \mathcal{L}_0) \Rightarrow (\mathcal{T}_1, \mathcal{L}_1) \Rightarrow (\mathcal{T}_2, \mathcal{L}_2) \Rightarrow \dots \tag{1}$$

where $\Rightarrow$ denotes the application of one $\mathcal{ALCQM}$-expansion rule. In the above sequence, the number of applications of the $\approx, \not\approx$ and close-rules is finite as we show below:

1. The $\approx$ and $\not\approx$-rules can be applied only a finite number of times in the above sequence. The $\approx$ and $\not\approx$-rules add concepts to the Tbox and these concepts that can be added all belong to $\mathsf{concepts}(\mathcal{M})$ which is finite. We also have that $\mathcal{T}_i \subseteq \mathcal{T} \cup \mathsf{concepts}(\mathcal{M})$ for all i. Besides none of the other rules remove elements from the Tbox.
2. Since the set $\{(a, b) \mid a, b \in \mathsf{dom}(\mathcal{M})\}$ is finite, the close-rule can be applied only a finite number of times. This is because once we set $a \approx b$ or $a \not\approx b$, no rule can "undo" this.

This means that from some n onwards in sequence (1)

$$(\mathcal{T}_n, \mathcal{L}_n) \Rightarrow (\mathcal{T}_{n+1}, \mathcal{L}_{n+1}) \Rightarrow (\mathcal{T}_{n+2}, \mathcal{L}_{n+2}) \Rightarrow \dots \tag{2}$$

there is no application of the rules $\approx$, $\not\approx$ and close. Moreover, $\mathcal{T}_n = \mathcal{T}_i$ for all $i \geq n$. Now, sequence (2) contains only application of $\mathcal{ALCQ}$-expansion rules. This sequence is finite by [1, Proposition 5.2]. This is a contradiction.

The proof of the following theorem is similar to Soundness for $\mathcal{ALCQM}$ [1]..

Theorem 2 (Soundness). *If $\mathcal{O} = (\mathcal{T}, \mathcal{A}, \mathcal{M})$ is consistent then the $\mathcal{ALCQM}$-tableau graph terminates and yields an $\mathcal{ALCQM}$-complete $(\mathcal{T}_k, \mathcal{L}_k)$ such that $\mathcal{L}_k$ has neither cycles nor contradictions.*

The following definition of canonical interpretation is basically the one in [1, Definition 4.3]. Instead of $<$, we use the idea of descendants.

Definition 8 ($\mathcal{ALCQ}$-Canonical Interpretation). *We define the $\mathcal{ALCQ}$-canonical interpretation $\mathcal{I}_{\mathsf{c}}$ from a tableau graph $\mathcal{L}$ as follows.*

$$\begin{aligned}
\Delta^{\mathcal{I}_{\mathsf{c}}} &= \{x \mid \mathcal{L}(x) \text{ is defined}\} \\
(x)^{\mathcal{I}_{\mathsf{c}}} &= \begin{cases} x & \text{if } x \in \Delta^{\mathcal{I}_{\mathsf{c}}} \\ y & \text{if } x \approx y \text{ and } y \in \Delta^{\mathcal{I}_{\mathsf{c}}} \end{cases} \\
(A)^{\mathcal{I}_{\mathsf{c}}} &= \{x \in \Delta^{\mathcal{I}_{\mathsf{c}}} \mid A \in \mathcal{L}(x)\} \\
(R)^{\mathcal{I}_{\mathsf{c}}} &= \{(x, y) \in \Delta^{\mathcal{I}_{\mathsf{c}}} \times \Delta^{\mathcal{I}_{\mathsf{c}}} \mid R \in \mathcal{L}(x, y), \text{ } x \text{ is not blocked or} \\
&\qquad R \in \mathcal{L}(z, y), \text{ where } x \text{ is blocked by } z \text{ and } z \text{ is not blocked } \}
\end{aligned}$$

Note that the canonical interpretation is not defined on equivalence classes of $\approx$ but by choosing canonical representatives.

Lemma 3. *If the tableau algorithm for $\mathcal{ALCQM}$ with input $\mathcal{O} = (\mathcal{T}, \mathcal{A}, \mathcal{M})$ yields an $\mathcal{ALCQM}$-complete $(\mathcal{T}', \mathcal{L})$ such that $\mathcal{L}$ has no contradictions then $\mathcal{I}_c$ is a model of $(\mathcal{T}, \mathcal{A})$.*

Proof. We define $\mathsf{rel}(\mathcal{A}_\mathcal{L})$ as follows.

$$\begin{aligned}\mathsf{rel}(\mathcal{A}_\mathcal{L}) = &\{C(x) \mid C \in \mathcal{L}(y), y \approx x\} \cup \\ &\{R(a,b) \mid R \in \mathcal{L}(x,y), a \approx x, b \approx y, \{a,b\} \subseteq \mathcal{O}\} \cup \\ &\{x = y \mid x \approx y\} \cup \{x \neq y \mid x \not\approx y\}\end{aligned}$$

By [1, Lemma 5.5]), $\mathcal{I}_c$ is a model of $(\mathcal{T}', \mathsf{rel}(\mathcal{A}_\mathcal{L}))$. Since $\mathcal{T} \subseteq \mathcal{T}'$ and $\mathcal{A} \subseteq \mathsf{rel}(\mathcal{A}_\mathcal{L})$, we have that $\mathcal{I}_c$ is a model of $(\mathcal{T}, \mathcal{A})$.

So, how can we now make $\mathcal{I}_c$ into a model of the whole ontology $(\mathcal{T}, \mathcal{A}, \mathcal{M})$? We will transform $\mathcal{I}_c$ into a model of $(\mathcal{T}, \mathcal{A}, \mathcal{M})$ by defining a function set. The following lemma allows us to give a recursive definition of set.

Lemma 4. *If the tableau graph $\mathcal{L}$ has no cycles then $(\Delta^{\mathcal{I}_c}, \prec)$ is well-founded where $\prec$ is the relation defined as $y \prec x$ if $y \in (A)^{\mathcal{I}_c}$, $x \approx a$ and $a =_\mathsf{m} A \in \mathcal{M}$.*

Proof. Suppose $(\Delta^{\mathcal{I}_c}, \prec)$ is not well-founded. Since $\Delta^{\mathcal{I}_c}$ is finite, infinite descendent $\prec$-sequences can only be formed from $\prec$-cycles, i.e. they are of the form

$$y_n \prec y_1 \prec \ldots \prec y_n$$

It is easy to see that this contradicts the fact that $\mathcal{L}$ has no cycles.

Definition 9 (From Basic Objects to Sets: the function set). *Let $\mathcal{L}$ a tableau graph without cycles and $\mathcal{I}_c$ be the $\mathcal{ALCQ}$-canonical interpretation from $\mathcal{L}$. For $x \in \Delta^{\mathcal{I}_c}$ we define $\mathsf{set}(x)$ as follows.*

$$\begin{aligned}&\mathsf{set}(x) = \{\mathsf{set}(y) \mid y \in (A)^{\mathcal{I}_c}\} && \text{if } x \approx a \text{ for some } a =_\mathsf{m} A \in \mathcal{M} \\ &\mathsf{set}(x) = x && \text{otherwise}\end{aligned}$$

Lemma 5. *Let $\mathcal{L}$ be an $\mathcal{ALCQM}$-complete tableau graph without contradictions. If $a =_\mathsf{m} A$ and $a' =_\mathsf{m} A'$ then either $a \approx a'$ or $a \not\approx a'$. In the first case, $A^{\mathcal{I}_c} = A'^{\mathcal{I}_c}$ and in the second case, $A^{\mathcal{I}_c} \neq A'^{\mathcal{I}_c}$*

Lemma 6. *Let $\mathcal{L}$ be an $\mathcal{ALCQM}$-complete tableau graph that has neither contradictions nor cycles and let $\mathcal{I}_c$ be the canonical interpretation from $\mathcal{L}$. Then, set is an injective function, i.e. $x = x'$ if and only if $\mathsf{set}(x) = \mathsf{set}(x')$.*

Proof. We prove first that set is a function. It is enough to consider the case when $x \approx a =_\mathsf{m} A$ and $x \approx a' =_\mathsf{m} A'$. By Lemma 5, $a \approx a'$ and $(A)^{\mathcal{I}_c} = (A')^{\mathcal{I}_c}$. Hence, $\mathsf{set}(x)$ is uniquely determined.

To prove that set is injective, we do induction on $(\Delta^{\mathcal{I}_c}, \prec)$ which we know that is well-founded by Lemma 4. By Definition of set, we have two cases. The first

case is when $\mathsf{set}(x) = x$. We have that $\mathsf{set}(x') = x$ and x' is exactly x. This was the base case. In the second case, we have that for $x \approx a$ and $a =_{\mathsf{m}} A$,

$$\mathsf{set}(x) = \{\mathsf{set}(y) \mid y \in (A)^{\mathcal{I}_{\mathsf{c}}}\}$$

Since $\mathsf{set}(x) = \mathsf{set}(x')$, we also have that $x' \approx a'$ and $a' =_{\mathsf{m}} A'$ such that

$$\mathsf{set}(x') = \{\mathsf{set}(y') \mid y' \in (A')^{\mathcal{I}_{\mathsf{c}}}\}$$

Again since $\mathsf{set}(x) = \mathsf{set}(x')$, we have that $\mathsf{set}(y) = \mathsf{set}(y')$. By Induction Hypothesis, $y = y'$ for all $y \in (A)^{\mathcal{I}_{\mathsf{c}}}$. Hence, $(A)^{\mathcal{I}_{\mathsf{c}}} \subseteq (A')^{\mathcal{I}_{\mathsf{c}}}$. Similarly, we get $(A')^{\mathcal{I}_{\mathsf{c}}} \subseteq (A)^{\mathcal{I}_{\mathsf{c}}}$. So, $(A)^{\mathcal{I}_{\mathsf{c}}} = (A')^{\mathcal{I}_{\mathsf{c}}}$. It follows from Lemma 5 that $a \approx a'$. Then, $x = x'$ because the canonical representative of an equivalence class is unique.

We are now ready to define the canonical interpretation for an ontology in $\mathcal{ALCQM}$.

Definition 10 (Canonical Interpretation for $\mathcal{ALCQM}$). *Let $\mathcal{L}$ be an $\mathcal{ALCQM}$-complete tableau graph without cycles and without contradictions. We define the canonical interpretation $\mathcal{I}^m$ for $\mathcal{ALCQM}$ as follows:*

$$\begin{aligned}
\Delta^{\mathcal{I}_{\mathsf{m}}} &= \{\mathsf{set}(x) \mid x \in \Delta^{\mathcal{I}_{\mathsf{c}}}\} \\
(a)^{\mathcal{I}_{\mathsf{m}}} &= \mathsf{set}(a) \\
(A)^{\mathcal{I}_{\mathsf{m}}} &= \{\mathsf{set}(x) \mid x \in A^{\mathcal{I}_{\mathsf{c}}}\} \\
(R)^{\mathcal{I}_{\mathsf{m}}} &= \{(\mathsf{set}(x), \mathsf{set}(y)) \mid (x, y) \in (R)^{\mathcal{I}_{\mathsf{c}}}\}
\end{aligned}$$

Definition 11 (Isomorphism between interpretations of $\mathcal{ALCQ}$). *An isomorphism between two interpretations $\mathcal{I}$ and $\mathcal{I}'$ of $\mathcal{ALCQ}$ is a bijective function $f : \Delta \to \Delta'$ such that*

- $f(a^{\mathcal{I}}) = a^{\mathcal{I}'}$
- $x \in A^{\mathcal{I}}$ *if and only if* $f(x) \in A^{\mathcal{I}'}$
- $(x, y) \in R^{\mathcal{I}}$ *if and only if* $(f(x), f(y)) \in R^{\mathcal{I}'}$.

Lemma 7. *Let $\mathcal{I}$ and $\mathcal{I}'$ be two isomorphic interpretations of $\mathcal{ALCQ}$. Then, $\mathcal{I}$ is a model of $(\mathcal{T}, \mathcal{A})$ if and only if $\mathcal{I}'$ is a model of $(\mathcal{T}, \mathcal{A})$.*

To prove the previous lemma is enough to show that $x \in C^{\mathcal{I}}$ if and only if $f(x) \in C^{\mathcal{I}'}$ by induction on C.

Theorem 3 (Completeness). *If $(\mathcal{T}, \mathcal{A}, \mathcal{M})$ is not consistent then the $\mathcal{ALCQM}$-tableau algorithm with input $(\mathcal{T}, \mathcal{A}, \mathcal{M})$ terminates and yields an $\mathcal{ALCQM}$-complete $(\mathcal{T}', \mathcal{L})$ such that $\mathcal{L}$ that has either a contradiction or a cycle.*

Proof. By Theorem 1, the $\mathcal{ALCQM}$-tableau algorithm with input $(\mathcal{T}, \mathcal{A}, \mathcal{M})$ terminates. Suppose towards a contradiction that the algorithm yields an $\mathcal{ALCQM}$-complete $(\mathcal{T}', \mathcal{L})$ such that that $\mathcal{L}$ has neither a contradiction nor a cycle. We will prove that $\mathcal{I}_{\mathsf{m}}$ is a model of $(\mathcal{T}, \mathcal{A}, \mathcal{M})$. For this we have to check that $\mathcal{I}_{\mathsf{m}}$ satisfies the three conditions of Definition 2.

1. In order to prove that $\Delta^{\mathcal{I}_m} \subseteq S_N$ for some S_N and N, we define $S_0 = \{x \in \Delta^{\mathcal{I}_c} \mid \mathsf{set}(x) = x\}$.
2. We now prove that $\mathcal{I}_m$ is a model of $(\mathcal{T}, \mathcal{A})$. By Lemma 3, the canonical interpretation $\mathcal{I}_c$ is a model of $(\mathcal{T}, \mathcal{A})$. It follows from Lemma 6 that $\mathsf{set} : \Delta^{\mathcal{I}_c} \to \Delta^{\mathcal{I}_m}$ is a bijective map. It is also easy to show that $\mathcal{I}_c$ and $\mathcal{I}_m$ are isomorphic interpretations in $\mathcal{ALCQ}$. By Lemma 7, $\mathcal{I}_m$ is a model of $(\mathcal{T}, \mathcal{A})$.
3. Finally, we prove that $a^{\mathcal{I}_m} = (A)^{\mathcal{I}_m}$ for all $a =_m A \in \mathcal{M}$. Suppose that $a =_m A \in \mathcal{M}$. Then,

$$\begin{aligned} a^{\mathcal{I}_m} &= \mathsf{set}(a) && \text{by Definition 10} \\ &= \{\mathsf{set}(x) \mid x \in (A)^{\mathcal{I}_c}\} && \text{by Definition 9} \\ &= A^{\mathcal{I}_m} && \text{by Definition 10} \end{aligned}$$

A direct corollary from the above result is that $\mathcal{ALCQM}$ satisfies the finite model property.

6 Conclusions and Future Work

In this paper we present a tableau algorithm for checking consistency of an ontology in $\mathcal{ALCQM}$ and prove its correctness. In order to implement our algorithm, we plan to incorporate optimization techniques such as normalization, absorption or the use of heuristics [2, Chapter 9].

A first step to optimize the algorithm would be to impose the following order on the application of the expansion rules. We apply the rules that create nodes ($\exists$ and $\geq$) only if the other rules are not applicable. We apply the bifurcating rules ($\sqcup$, choose or close-rules) if the remaining rules (all rules except the $\exists$, $\geq$, $\sqcup$, choose and close-rules) are not applicable. One could prove that this strategy is correct similarly to Section 5.

A second step to optimize the algorithm would be to change the $\approx$-rule. Instead of adding $A \sqcup \neg B$ and $\neg A \sqcup B$, we could add $A \equiv B$ and treat this as a trivial case of lazy unfolding.

We would also like to study decidability of consistency for the kind of meta-modelling presented in this paper in more powerful Description Logics than $\mathcal{ALCQM}$.

We believe that consistency in $\mathcal{ALCQM}$ has the same complexity as $\mathcal{ALCQ}$, which is Exp-time complete [13]. We also plan to study worst-case optimal tableau algorithms for $\mathcal{ALCQM}$ [3,6].

Acknowledgments. We are grateful to Diana Comesaña for sharing with us the data from the ontology network on geographic objects she is developing in Uruguay [10].

References

1. Baader, F., Buchheit, M., Hollunder, B.: Cardinality restrictions on concepts. Artif. Intell. **88**(1–2), 195–213 (1996)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.) The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
3. Donini, F.M., Massacci, F.: Exptime tableaux for ALC. Artif. Intell. **124**(1), 87–138 (2000)
4. De Giacomo, G., Lenzerini, M., Rosati, R.: Higher-order description logics for domain metamodeling. In: AAAI (2011)
5. Glimm, B., Rudolph, S., Völker, J.: Integrated metamodeling and diagnosis in OWL 2. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 257–272. Springer, Heidelberg (2010)
6. Goré, R., Nguyen, L.A.: Exptime tableaux for ALC using sound global caching. In: Description Logics (2007)
7. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
8. Jekjantuk, N., Gröner, G., Pan, J.Z.: Modelling and reasoning in metamodelling enabled ontologies. Int. J. Software and Informatics **4**(3), 277–290 (2010)
9. Kaushik, S., Farkas, C., Wijesekera, D., Ammann, P.: An algebra for composing ontologies. In: Bennett, B., Fellbaum, C. (eds.) FOIS. Frontiers in Artificial Intelligence and Applications, vol 150, pp. 265–276. IOS Press (2006)
10. Militar, S.G., Catálogo de objetos y símbolos geográficos (2014). http://www.sgm.gub.uy/index.php/component/content/article/49-noticias/novedades/124-catalogoversion1/
11. Motik, B.: On the properties of metamodeling in OWL. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 548–562. Springer, Heidelberg (2005)
12. Pan, J.Z., Horrocks, I., Schreiber, G.: OWL FA: A metamodeling extension of OWL DL. In: OWLED (2005)
13. Tobies, S.: Complexity results and practical algorithms for logics in knowledge representation. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany (2001)