OWL extended with Meta-modelling

Regina Motz[1], Edelweis Rohrer[1], Paula Severi[2⋆] and Ignacio Vidal[1]

[1] Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay
[2] Department of Computer Science, University of Leicester

Abstract. In this paper we explain how we extended the Web Ontology Language (OWL) with meta-modelling. In order to express meta-modelling, we extended the OWL syntax with a new constructor that allows us to equate individuals with classes. In order to perform the main task of checking consistency of ontologies that have meta-modelling, we extended the free and open source reasoner for the Semantic Web called *Pellet*.

Keywords: OWL, Meta-modelling, Reasoner, Consistency checking

1 Motivation

In different application scenarios the need of linking ontologies of different domains arises. But the way how these ontologies are related is not always the same. Sometimes it is required to map classes or individuals of two ontologies, or link individuals of two ontologies through a new property. In these cases, elements of the same granularity are mapped, i. e., classes to classes or individuals to individuals. But there are some scenarios where mapping elements of different granularity is needed, for instance when the same real object is represented as an individual in one ontology and as a class in other ontology. This kind of relation between ontologies is called meta-modelling and is the main motivation of our work. Our extension of OWL comes up from a real-world application on geographic objects that requires to reuse existing ontologies and relate them through meta-modelling [1]. Figure 1 describes a simplified scenario of this application in order to illustrate the meta-modelling relationship. It shows two ontologies separated by a horizontal line. The two ontologies conceptualize

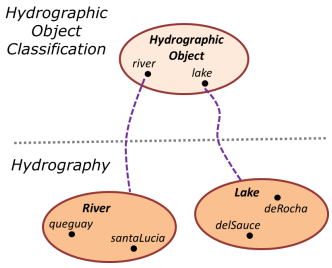

Fig. 1. Two ontologies on Hydrography

⋆ Daphne Jackson fellowship sponsored by EPSRC and the University of Leicester.

Copyright © 2016 for this paper by its authors. Copying permitted for private and academic purposes.

the same entities at different levels of granularity. In the ontology above the horizontal line, rivers and lakes are represented as individuals while in the one below the line they are classes. 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 class *River* as the same real object. Similarly for *lake* and *Lake*. Our solution consists in equating the individual *river* to the class *River* and the individual *lake* to the class *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 class *HydrographicObject* is now also a *meta-class* because it is a class that contains an individual which is also a class. The kind of meta-modelling we consider in this paper can be expressed in OWL Full, which allows to equate any two resources, for instance a class to an individual or a property to an individual. However, 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 [2]. 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 class [3]. 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 reasoner will not detect that the interpretation of *River* is not a well founded set (it is a set that belongs to itself). That is, the interpretation of *River* (which is equal to *river* by meta-modelling) belongs to that of *HydrographicObject* and, by the introduced axiom, the interpretation of *HydrographicObject* is a subset of that of *River*.

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 reasoner does not detect that there is a contradiction.

2 Extending OWL with Meta-modelling Axioms

In order to express meta-modelling, we extended OWL with *meta-modelling axioms*. A meta-modelling axiom $a =_{\mathsf{m}} A$ is an equation between an individual a and an atomic class A. The semantics of the above axiom is that the individual a and the class A have the same interpretation. For example, the meta-modelling axiom that equates the individual river with the class River is expressed in OWL/XML as follows.

```
<MetaModelling>
    <NamedIndividual IRI="river"/>
    <Class IRI = "River" />
</Metamodelling>
```

The individual *river* is interpreted as the set $\{queguay, santaLucia\}$ since this is exactly the interpretation of the class *River*. Extending the syntax is the easy part. The most difficult part consists in extending the Tableau algorithm in order to check consistency of ontologies that have meta-modelling [4, 5]. For an introduction on the basics of the Tableau algorithm, we suggest the reader to see [3]. The Tableau algorithm constructs a graph to represent a possible model of the knowledge base. The nodes of the initial graph are the individuals of the ontology and the edges are the properties between the individuals. Equality between individuals is recorded using $a \approx b$ and each node x has associated a set of class expressions $\mathcal{L}(x)$. In [4, 5] we extended the Tableau algorithm for checking consistency of ontologies with meta-modelling by adding:

1. new expansion rules: to deal with equalities and inequalities of individuals with meta-modelling which are given in Table 1.
2. a new condition to deal with circularities with respect to the membership $\in$ relation. This condition detects the presence of non well founded sets.

Equality Rule:
If $a =_{\mathsf{m}} A$, $b =_{\mathsf{m}} B$, $a \approx b$ and $A \equiv B$ does not belong to the Tbox then,
add $A \equiv B$ to the Tbox.

Inequality Rule:
If $a =_{\mathsf{m}} A$, $b =_{\mathsf{m}} B$, $a \not\approx b$ and there is no z, $A \sqcap \neg B \sqcup B \sqcap \neg A \in \mathcal{L}(z)$ then,
create a new node z with $\mathcal{L}(z) = \{A \sqcap \neg B \sqcup B \sqcap \neg A\}$.

Close Rule:
If $a =_{\mathsf{m}} A$, $b =_{\mathsf{m}} B$ and neither $a \approx b$ nor $a \not\approx b$ then,
add either $a \approx b$ or $a \not\approx b$.

Table 1. Expansion rules for meta-modelling

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 classes. Any equality at the level of individuals should be transferred as an equality between classes and similarly with the difference. A particular case of the application of Equality Rule is when $a =_{\mathsf{m}} A$ and $a =_{\mathsf{m}} B$. In this case, the algorithm also adds $A \equiv B$. In the Inequality Rule, the inequality $a \not\approx b$ should be transferred to the level of classes as $A \not\equiv B$. However, 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 witnesses this difference.
The rules for equality and inequality are not sufficient to detect all inconsistencies coming from meta-modelling. The idea is that we also need to transfer the equality $A \equiv B$ between classes as an equality $a \approx b$ between individuals and we also need to transfer the semantic consequences, e.g. $\mathcal{O} \models A \equiv B$. Unfortunately, a recursive call of the form $\mathcal{O} \models A \equiv B$ is not possible. Otherwise we

will be captured in a vicious circle 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 $a \not\approx b$. It is similar to the choose-rule which adds either C or $\neg C$ [3].
Our Tableau algorithm for meta-modelling has a Tbox rule:

> Tbox Rule:
> If C is a TBox statement and $C \notin \mathcal{L}(x)$, then add C to $\mathcal{L}(x)$.

where C is a TBox axiom $C \sqsubseteq D$ codified as $\neg C \sqcup D$.
It is not so straightforward to extend the code of Pellet [6, 7] to include the new expansions rules for meta-modelling. This is because we have to adapt our new rules to cope with the optimizations of Pellet [8–10]. The main complication is given by the equality rule which has the unusual characteristic of changing the Tbox, since we add new TBox axioms. This lead us to re-run actions and optimizations on the TBox, which were originally executed at the beginning, each time the equality rule is applied, because of the addition of new axioms.
First of all, Pellet divides the Tbox $\mathcal{T}$ in two disjoint sets: an unfoldable part $\mathcal{T}_u$ that contains unique, acyclical definition axioms and a general Tbox $\mathcal{T}_g = \mathcal{T} \setminus \mathcal{T}_u$. Then, it tries to absorb subclass axioms from the $\mathcal{T}_g$ into the $\mathcal{T}_u$.
After that, in the rule application step, Pellet uses the expansion rules of lazy unfolding [8] shown below.

> Unfolding Rule I:
> If $A \sqsubseteq C \in \mathcal{T}_u$, $A \in \mathcal{L}(x)$ and $C \notin \mathcal{L}(x)$ then add C to $\mathcal{L}(x)$.

> Unfolding Rule II:
> If $\neg A \sqsubseteq C \in \mathcal{T}_u$, $\neg A \in \mathcal{L}(x)$ and $C \notin \mathcal{L}(x)$ then add C to $\mathcal{L}(x)$.

The general Tbox $\mathcal{T}_g$ is expressed as $\top = \bigsqcap \mathcal{T}_g$, which is the conjunction of all $\mathcal{T}_g$ axioms. Then, Pellet also applies lazy unfolding to the nodes x that have $\top$ in $\mathcal{L}(x)$ by adding $\sqcap \mathcal{T}_g$ to $\mathcal{L}(x)$. But this works only if we ensure that $\top$ is in $\mathcal{L}(x)$ for all nodes x of the graph. The initialisation step in Pellet guarantees that $\top$ is added to $\mathcal{L}(a)$ for all the individuals a of the ontology and every expansion rule that creates a new node z (such as $\exists$-rule) adds $\top$ to $\mathcal{L}(z)$.

We also apply optimization techniques to our meta-modelling rules. The optimized rules for meta-modelling are shown in Table 2. The Optimized Inequality Rule adds $\top$ to $\mathcal{L}(z)$ to the new node z that it creates. The Optimized Equality Rule adds $A \equiv B$ to the $\mathcal{T}_g$ and applies the algorithm of absorption in order to absorb the new axiom $A \equiv B$. This axiom could be absorbed completely or partially. Some part of this axiom may pass to the $\mathcal{T}_u$ and another part may remain in the $\mathcal{T}_g$. Finally, the Optimized Equality Rule forces the application of the unfolding rule for $\top$, which ensures that the parts not absorbed of the axioms $A \equiv B$ be applied to all nodes.
In order to ensure termination, the expansion rules should not be applied more

than once under the same conditions. For the Optimized Equality Rule, we keep record of the fact that we applied this rule and equated A with B by means of $A \approx B$. It would be wrong to have the condition "$A \equiv B$ does not belong to $\mathcal{T}_g$" because parts of that axiom could have been absorbed. For the Optimized Inequality Rule, we keep record of the fact that we applied this rule and set A different from B by means of $A \not\approx B$. This is more efficient than having the alternative condition "there is no root node z such that $A \sqcap \neg B \sqcup B \sqcap \neg A \in \mathcal{L}(z)$" of Table 1.

Optimized Equality Rule:
If $a =_{\mathsf{m}} A$, $b =_{\mathsf{m}} B$, $a \approx b$ and $A \approx B$ does not hold then
1. add $A \equiv B$ to the $\mathcal{T}_g$.
2. apply the algorithm of absorption to $(\mathcal{T}_u, \mathcal{T}_g)$.
3. apply the unfolding rule for $\top$.

Optimized Inequality Rule:
If $a =_{\mathsf{m}} A$, $b =_{\mathsf{m}} B$, $a \not\approx b$ and $A \not\approx B$ does not hold then
create a new root node z with $\mathcal{L}(z) = \{A \sqcap \neg B \sqcup B \sqcap \neg A, \top\}$

Table 2. Optimized equality and inequality rules

The implementation of Pellet extended with meta-modelling can be found in `http://www.cs.le.ac.uk/people/ps56/pelletM.xml`.

3 Conclusions and Related Work

OWL 2 DL has a very restricted form of meta-modelling called *punning* [3]. In this approach, the same identifier can be used simultaneously as an individual and as a concept, but they are semantically treated as different real objects. So, it does not detect certain inconsistencies as the ones illustrated in Examples 1 and 2. Moreover, 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 $a =_{\mathsf{m}} A$.
In the literature there are other approaches proposed to deal with meta-modelling in Description Logic [2, 11–17]. The approaches which define fixed layers or levels of meta-modelling [11, 13, 15, 16] impose a very strong limitation to the ontology engineer. 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. In a real scenario of evolving ontologies, that need to be integrated, not all individuals of a given concept need to have meta-modelling and hence, they do not have to belong to

the same level in the hierarchy.
The key feature in our semantics is to interpret a and A as the same object when a and A are connected through meta-modelling, i.e., if $a =_{\mathsf{m}} A$ then $a^{\mathcal{I}} = A^{\mathcal{I}}$. This allows us to detect inconsistencies in the ontologies which is not possible under the Hilog semantics [2, 14–17]. Our semantics also requires that the domain of the interpretation be a well-founded set. A domain such as $\Delta^{\mathcal{I}} = \{X\}$ where $X = \{X\}$ is a set that belongs to itself which cannot represent any real object from our usual applications in Semantic Web.

References

1. Servicio Geográfico Militar del Uruguay. Catálogo de objetos y símbolos geográficos. At http://www.sgm.gub.uy/index.php/documentos/doc_download/92-catalogo-de-objetos-geograficos-y-simbolos-v1. Last date accessed September 2015.
2. B. Motik, On the properties of metamodeling in OWL, in: International Semantic Web Conference, 2005, pp. 548–562.
3. P. Hitzler, M. Krötzsch, S. Rudolph, Foundations of Semantic Web Technologies, Chapman & Hall/CRC, 2009.
4. R. Motz, E. Rohrer, P. Severi. Reasoning for *ALCQ* extended with a flexible metamodelling hierarchy, in: Semantic Technology-4th Joint International Conference, JIST 2014. Revised Selected Papers, pp. 47–62.
5. R. Motz, E. Rohrer, P. Severi. The Description Logic *SHIQ* with a flexible meta-modelling hierarchy, in Journal of Web Semantics, doi:10.1016/j.websem.2015.05.002, open access 2015.
6. Clark, Parsia, Pellet, http://clarkparsia.com/pellet/. Last date accessed September 2015.
7. E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL reasoner, Journal of Web Semantics 5(2):51-53 (2007).
8. I. Horrocks, Implementation and Optimisation Techniques., in: The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003, pp. 306–346.
9. Ian Horrocks, Stephan Tobies: Optimisation of Terminological Reasoning. Description Logics 2000.
10. Alexander K. Hudek, Grant E. Weddell: Binary Absorption in Tableaux-Based Reasoning for Description Logics. Description Logics 2006.
11. J. Z. Pan, I. Horrocks, G. Schreiber, OWL FA: A metamodeling extension of OWL DL, in: OWLED, 2005.
12. B. Glimm, S. Rudolph, J. Völker, Integrated metamodeling and diagnosis in OWL 2, in: International Semantic Web Conference, 2010, pp. 257–272.
13. N. Jekjantuk, G. Gröner, J.Z. Pan, Modelling and reasoning in metamodelling enabled ontologies, Int. J. of Software and Informatics 4 (3) (2010) 277–290.
14. G. De Giacomo, M. Lenzerini, R. Rosati, Higher-order description logics for domain metamodeling, in: AAAI, 2011.
15. M. Homola, J. Kluka, V. Svátek, M. Vacura, Towards Typed Higher-Order Description Logics, in: Description Logics, 2013.
16. M. Homola, J. Kluka, V. Svátek, M. Vacura, Typed Higher-Order Variant of $\mathcal{SROIQ}$ - Why Not?, in: Description Logics, 2014.
17. M. Lenzerini, L. Lepore, A. Poggi, Making Metaquerying Practical for Hi(DLLiteR) Knowledge Bases, in: On the Move to Meaningful Internet Systems: OTM 2014 Conferences, 2014, 8841 580–596.