Calculations in OWL

Luigi Iannone and Alan Rector

School of Computer Science
University of Manchester
Manchester
M13 9PL UK
{iannone,rector}@cs.manchester.ac.uk

Abstract. The current OWL specification does not include a syntax for specifying calculated values for data type properties. Their introduction, although acknowledged among the desiderata, seems yet unlikely to appear in the forthcoming OWL 2.0. This paper shows one way of working around this absence using annotations. The aim here is not to suggest a standard but, rather, to individuate the potential issues that a full fledged standard solution will have to tackle.

1 Motivation

OWL 1.1/2.0's ability to deal with numeric ranges and values is a major step forward for many applications. However, very often, in the biological domain, it is necessary that numeric data data are transformed before being used. While there are many more complex arithmetic operations that would be useful, the ability to take two or more numeric inputs and calculate composite result, is needed to make progress at all. Typical examples are calculation of Body Mass Index from height and weight (BMI = Weight/Height^2), units conversion (the weight in the preceding formula must be in Kg and the height in meters), drug dose calculations which are often expressed as total daily dose per weight or per body surface area. Although theoretical negative results [1] have discouraged the incorporation of complex arithmetic operations into inference engines, a limited, reasoner independent, solution that supplements the OWL specification in a systematic way is needed. In this paper we propose a possible way forward.

2 OWL Calculations

Our framework offers the possibility to attach one or more alternative formulas[1] to a data property so that, after a preprocessing phase, each individual admitting such properties in an OWL ontology, could have a filler computed using the appropriate formula. Our initial assumption is that formulas ought to be attached to properties. The most direct way to do that is by means of annotations on the

[1] Please notice that the words formula, calculations, and their plurals are used henceforth interchangeably as synonyms for the same concept

data property themselves. This has the direct consequence that, in the frequent case in which different classes of individuals require different formulas for computing the value for the same data property, such property will present as many annotations as there are alternatives. Moreover, this requires a mechanism for deciding which one should be applied. A more object-oriented approach would have, on the contrary, suggested to attach formulas to classes. However, we decided against the latter because it would have been convenient only for formulas that applied to named classes[2], as there is no way in OWL to annotate a non primitive class. The only workaround would be to define such complex classes by asserting their equivalence with a named one, and then annotate the latter with the formula. However, this would not scale and would introduce new class names whose relevance in terms of domain model is at least dubious. Furthermore, OWL is axiom, rather than, object-oriented and this makes the annotation of properties more natural than the annotation of classes with formulas.

In Fig. 1 we summarise graphically the components of a formula in our framework:

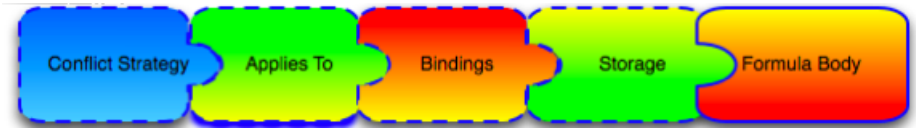

Fig. 1. Formula components - dashed lines mean optional component

A formula can be decomposed into the following blocks:

- Conflict Strategy: Indicates the strategy to adopt when a value is calculated for a functional property that already has got a filler for for the individual under consideration. The possible strategies are:
  - OVERRIDING - The engine will replace the existing value with the newly computed one.
  - OVERRIDDEN - The engine will discard the computed value preserving the existing one.
  - EXCEPTION - The engine will raise a specific exception relinquishing control to an external application or to the user.
- Applies to: Class expression that restricts the application scope of the formula to the instances of such description (Optional, if omitted the property domain will be the scope). In case more than one formula is applicable to an individual (i.e.: the individual is in more than one Applies to restriction for the same property, and such scopes are incomparable w.r.t. subsumption) an exception is raised and no value is computed.

[2] Named class here is used to denote either defined classes or primitive ones, as opposed to unnamed, anonymous, complex class

– Bindings: A binding is an assignment of a formula variable. Each binding is a property chain that yields to a value (or a set of values) to be assigned to the corresponding variable in the formula.
– Storage: Like bindings, storage paths are property chains that determine where to store the computed value(s). They are made of object properties only and the final data property is omitted as it is the data property annotated by the formula itself. If omitted, for each computed value, a filler for the annotated data property will be attached directly to the each individual in the formula application scope.
– Formula Body: The formula itself. It can be a simple formula or an aggregate. In case of multiple bindings for a variable, the former will output multiple results, the latter ones will out put the corresponding aggregate value.

We developed a syntax for encoding all the above inside OWL annotations attached to data properties. The complete grammar can be found at `http://www.cs.man.ac.uk/~iannonel/owlcalculations/syntax.html`. Besides dealing with cases of different formulas for the same data property inherited by an individual on account of multiple inheritance, our framework **does not** solve any of the theoretical problems referenced above (see [1]). Unedicidability remains a problem that formula users must deal with. This framework, in fact, is meant to be used as a preprocessing facility that populates the ontology with values, before it is sent to a reasoner for classification purposes. However, the framework itself uses the reasoner inferences to apply its formulas. Hence, the only safe situations are those where data property values do not interact with terminological axioms, i.e.: instance classification does not depend on its data property fillers.

3 Related Work

To the best of our knowledge, there is not, at the state of the art, a standard way to express, in OWL, that the fillers of a data property can (or should) be derived from others by means of a formula. The immediate plans for OWL 2.0 specifications drafted so far [2] do not provide any support either. Nonetheless, the issue has been raised and discussed during the drafting phase [3] and it will be resumed in the preparation of the following versions of OWL. Ours, then, should be considered a **practical**, **temporizing** solution and an early exploration of the issues that a fully OWL-based answer to the problem of including calculations into ontologies will have to solve. The only comparable alternative consists of utilising a rule based formalism to represent calculations. In this perspective, DL-safe rules [4] are currently the most expressive decidable rule language compatible with OWL-DL. Rules implementing the calculations could be encoded in the DL-safe fragment of the Semantic Web Rule Language (SWRL) and used on top of ontologies. The actual formulas can be translated using what in the SWRL specification are called *built-ins*. Unfortunately, all the implementation at the state of the art of reasoners that deal with DL-safe rules offer a very limited

or no built-in support (see, for instance, Pellet - http://pellet.owldl.com/faq/rules). Furthermore, there is a limited support also for SWRL rules that have anonymous OWL classes in their bodies, hence, in order to encode the applicability of a formula to an individual, one would have to refer it to a named class. This has the same shortcoming of annotating classes with formulas rather than data property (see the discussion at the beginning section 2), i.e.: the undesirable proliferation of defined class created just for calculation rather than modeling purposes.

4 Conclusions and Future Work

We presented a framework that enables the attachment of formulas to a data property in OWL and the computation of its values. This has been done in view of a more thorough standard specification covering this functionality for OWL and represents a temporizing solution. We also developed an API and a plug-in for Proótegé 4 ontology editor that can be found along with essential documentation at http://www.cs.man.ac.uk/~iannonel/owlcalculations/. Future work in this direction will focus on layering formulas on top of each other, augmenting the set of arithmetic operators available, devising extension for custom strategies for conflicts between computed values and pre-existing ones, and heuristics to detect dangerous (potentially undecidable) interactions with the reasoners.

References

1. Baader, F., Sattler, U.: Description logics with concrete domains and aggregation. In Prade, H., ed.: Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98), John Wiley & Sons Ltd (1998) 336–340
2. Horrocks, I., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2 Web Ontology Language: Model-Theoretic Semantics W3C Working Draft April 2008.
3. W3C OWL Working Group: OWL Working Group Issue Tracking - Issue 5 - Doubt about n-ary types http://www.w3.org/2007/OWL/tracker/issues/5.
4. Motik, B., Sattler, U., Studer, R.: Query Answering for OWL-DL with Rules. In McIlraith, S.A., Plexousakis, D., van Harmelen, F., eds.: International Semantic Web Conference. Volume 3298 of Lecture Notes in Computer Science., Springer (2004) 549–563