Annotated Literals for Standard Units of Measurement

Ryan Blace, Andrew Perez-Lopez

[1] BBN Technologies, 1300 N. 17th Street, Suite 400, Arlington, VA 22209
{rblace, aperezlo}@bbn.com

Abstract. In our experiences building systems that use Semantic Web technologies, we have often identified a requirement for a consistent way to associate data values with standard units. We have tried a number of solutions to this issue, each of which has its own benefits and drawbacks in terms of complexity, expressivity, and clarity. In this paper, we present a discussion of the issue and a proposal for a simple extension to OWL/RDF to support the annotation of literal values with standard SI units that meets our needs without introducing many of the drawbacks of alternate approaches.

Keywords: OWL, RDF, International Standard Units, Annotation

1 Introduction

At BBN Technologies, we use Semantic Web technologies in both prototype and operational systems that help our customers address their information integration and data management challenges. Most of our projects use OWL to represent various domain knowledge models and to map between the concepts within each. We define classes and properties, arrange them taxonomically, and use restrictions to specify structure expectations and to perform input validation.

Our knowledge models vary from network-based models which are composed of entities and relationships to measurement-based models that contain entities with many data values. For almost any measurement data, proper interpretation depends on an awareness of the units of the measurement. This is particularly true when integrating measurement data from various sources or knowledge models.

We've experimented with, and observed in relevant literature, many solutions to expressing units of measurement in OWL. The most common approach involves the inclusion in the knowledge model of unit-specific properties or data types. Other approaches include statement reification or the use of value containers that associate values with units and other relevant information. While any of these approaches is sufficient for capturing units, none has proved satisfactory for our purposes.

In this paper, we present a brief discussion of units in information management along with a number of approaches to solving it. Each approach is assessed critically and the benefits and drawbacks to each are identified. Next, we present a proposal for a solution for representing units for standard measurements that addresses our requirements, and discuss its strengths and weaknesses.

2 Units in Information Management

One of the insights of OWL/RDF is that data is only useful when combined with a relevant context, or collection of metadata. A model that is expressive enough to represent both TBox information such as classes, properties, and the relationships among them, and ABox information about instances is one of the major successes of this area of research. However, while OWL and RDF allow for a great deal of metadata about instances, they do not similarly allow a rich description of literal values. Just as knowing the type of an instance allows machines to interpret RDF information more usefully, so too would metadata about literal values.

Maintaining information about the units associated with literal values is critically important in many real-world systems. Proper comparisons, model consistency checks, and a better user experience with input validation and more customization are some of the system requirements that demand unit specification.

There are a few approaches from the literature that we have attempted to apply, but none is satisfactory. In the following section, we describe each approach we are considering.

2.1 Unit-specific Properties and Datatypes

One of the most common ways to express measurement unit information is to define unit-specific properties in an ontology. For example, an ontology about people may define multiple properties representing height, including heightMeters, heightFeet, heightInches, heightCentimeters, etc. The property itself designates the unit information about the value to which it refers. Consider the example statement: *Steve hasHeightInches 72*. This clearly states that Steve has a height of 72 inches.

Unit-specific datatypes are a similar approach that uses typed literals to specify units. Consider the statement *Steve hasHeight 72^^int-inches.*

2.2 Statement Reification and Value Containers

Statement reification involves annotating reification resources with unit information that describes the value of the statement. An example is to say that the statement *Steve hasHeight 77* has units inches.

A final approach to this problem is the use of value containers [1]. This involves replacing literal values with resources that in turn refer to the associated literal value and appropriate unit. Consider the following example: *Steve hasHeight _height, _height hasValue 72, _height hasUnits inches.*

2.3 Benefits and Drawbacks to Each Approach

Each approach we have discussed has its own benefits and drawbacks. In general, each approach suffers from at least one of the following issues: complexity, expressivity, or clarity.

The unit-specific property approach is simple and clear. Its drawbacks are in expressivity. It mixes the concept of height with a feature of the data value (meters, feet, inches, etc.) to which it refers. The concept of height does not depend on the units in which the value is expressed. The inappropriate conflation of the semantics and the representation produces a plethora of new predicates representing every permutation of property and unit. Using a super-property relationship to establish the semantic similarity between height and all of its unit-specific versions is of questionable utility because the unit information is lost when the super-property statements are entailed, rendering the entailed statements unit-less.

The unit-specific datatype approach is also simple and clear, and it correctly associates unit information with values rather than properties so that units are correctly transferred with entailed statements. It eliminates the necessity of an unruly number of predicates, but trades it for the only slightly less egregious problem of type/unit combinations. With unit-specific datatypes, system builders must cope with int-meters and float-meters, and int-liters and float-liters, etc. This also introduces the problem of comparisons - according to standard RDF literal equality, a height of "1.8"^^float-meters is not the same as "1.8"^^double-meters. Additionally, types cannot easily be combined, as is often required by real-world systems. Additional, custom, unit-specific types are required to support, for example, kilometers per liter, or per hour, or per year.

The statement reification approach suffers from a number of problems. First, it once again associates unit information with statements, rather than with the objects of statements, so entailments lose the unit information. Second, it's a complex and potentially expensive method for modeling this information because it includes additional structure that must be maintained and queried in the model.

The value container approach is probably the most flexible, explicit, and correct approach we have observed. It allows you to associate almost anything with a data value, entailments correctly carry unit information with them, and it supports any sort of unit specification method you may desire. However, there are drawbacks to this approach as well. It trades expressiveness for complexity and interoperability, and it deemphasizes the role of datatype properties. The added structure for specifying data values requires more statements to model, and more complex queries. Interoperability suffers because this method requires a non-standard method for specifying data values. Finally, this method virtually eliminates the use of datatype properties for describing resources other than the value containers. In the most extreme case, where all data values require units, every data value must be encapsulated within these value containers and referred to using object properties.

5 Our Requirements and a Proposal

Most of the approaches we have discussed either associate unit information with the wrong concept (the predicate or the statement), they conflate two dimensions of data values, or they are overly complicated and non-standard. We don't need excessive expressivity and we don't want additional structure assumptions that will exacerbate interoperability challenges. We need a solution that correctly associates units with

data values and is correctly carried with entailed statements. We prefer a simple solution that provides the means to correctly convey unit information when it is interfaced with humans or external applications.

We propose extending the language tag to provide support for the annotation of non-string datatypes with semantics-free, composable *unit strings*. The following table represents a few examples of literals with unit strings maintained in the language tag:

Table 1. Literals using the proposed language tag representation for units

Literal	Interpretation
“57”@km	57 kilometers
“57000”@m	57000 meters
“78”@kg	78 kilograms
“122”@km.kg-1	57 kilometers / kilogram
“100”@km.h-1	100 kilometers / hour

Representing units of measurement like languages has several desirable characteristics. Language tags specify how humans and applications should interpret string datatype literals. This is very similar to our desire for specifying units. Using language tags as an annotation for units avoids any great increase in the number of required predicates or datatypes, because literal datatypes are already treated separately from the language tag. Additionally, existing parsers would only require slight changes, if any, to support the expanded use of the language tag, so it could be implemented quickly.

The use of language tag annotation would be semantics-free, so no additional complexity would be introduced into class definitions or descriptions. Just as the meaning of typed literals depends on information outside RDF, so too would the semantics of units. Reasoners could optionally support automatic conversions between units, or consistency checking if desired, but those details need not be defined by relevant OWL or RDF specifications. Additionally, units are specified directly on literals and will be propagated through entailment. The obvious drawback to this approach is that it is semantics-free and has limited utility in the knowledge model itself; however, in our experience that drawback has limited impact.

References

1. Chaudhri, V.K., Jarrold, B., Pacheco, J.. Exporting Knowledge Bases into OWL. OWLED Long Papers (2006)