Extending DL Reasoning Support for the OWL Datatyping (or "Why Datatype Groups?")

Jeff Z. Pan and Ian Horrocks
Department of Computer Science,
University of Manchester, UK M13 9PL
{last-name}@cs.man.ac.uk

The OWL [2] datatype formalism (or simply *OWL datatyping*) presents some new requirements for DL reasoning services, in terms of semantics (to allow the use of so-called 'un-supported' datatypes), expressive power (to support enumerated datatypes) and datatype construction mechanism (both datatypes and datatype expressions). On the other hand, OWL datatyping is expected to be extended to include more expressive power. E.g., OWL datatyping does not provide a general framework for user-defined datatypes, such as XML Schema derived datatypes, nor does it support n-ary datatype predicates (such as the binary predicate $>$ for integers), not to mention user-defined datatype predicates (such as the binary predicate $>$ for non-negative integers). In this poster, we explain why it is necessary to extend the existing datatype approaches to the datatype group approach, in order to meet the above new requirements.

It was Baader and Hanschke [1] who first presented a rigorous treatment of datatype predicates (or simply *predicates*). In their approach, a concrete domain [1, 4] is composed of a set of datatype values (such as integers) and a set of *n-ary* predicates (such as '$<$') defined over these values with obvious (fixed) extensions. Horrocks and Sattler [3] proposed the so called 'type system approach', which can be seen as a simplified version of the concrete domain approach, where the datatype domain (of a datatype interpretation) is regarded as a universal concrete domain and datatypes are treated as unary predicates in the universal concrete domain. In short, in the above two approaches, datatypes are nothing but unary predicates.

In OWL datatyping, however, people take another view. A Datatype d distinguishes from a predicate in that it is characterised not only by the value spaces $V(d)$, but also a lexical space, $L(d)$, which is a set of Unicode strings, and a total mapping $L2V(d)$ from the lexical space to the value space. E.g., *boolean* is a datatype with value space $\{true, false\}$, lexical space {T,F,1,0} and lexical-to-value mapping {T$\mapsto$ $true$, F$\mapsto$ $false$, 1$\mapsto$ $true$, 0$\mapsto$ $false$}. Data values can be represented by typed literals or plain literals, where *typed literals* are combinations of string and datatype URIs, while *plain literals* are simply strings, with optional language tag. E.g., "1"^^xsd:boolean is a typed literal, while "1" is a plain literal. Therefore, when we extend OWL datatyping to support predicates, we should not simply replace datatypes with predicates,

but let them co-exist in a proper framework.

Secondly, an OWL datatype interpretation is relativised to a datatype map, which is a partial mapping from datatype URIs to datatypes; e.g., $\mathbf{M}_{d1} = \{\langle \text{xsd:string}, \mathit{string}\rangle, \langle \text{xsd:integer}, \mathit{integer}\rangle\}$. *Unsupported datatypes*, which are not included in a given datatype map, are interpreted as any subsets of the datatype domain. Therefore, the datatype domain (of a datatype interpretation) is expected to be unfixed, which is different from the (datatype) domain in existing approaches.

Thirdly, OWL advocates a more user-friendly style of datatyping than what the existing approaches provide. OWL provides a kind of datatype expressions, called *enumerated datatypes*, of the form oneOf$(l_1, \ldots, l_n)$, where $l_1, \ldots, l_n$ are literals, which is interpreted as the union of all the interpretation of l_i $(1 \leq i \leq n)$. It is expected that it supports more expressive datatype expressions, to represent user-defined datatypes and user-defined predicates. Furthermore, it is desirable that the interpretation of negated predicate is relativised to the value space of the related datatypes; e.g., $\overline{>5}$ is interpreted as $V(\mathit{integer}) \backslash >_5^{\mathbf{D}}$ but not $\Delta_{\mathbf{D}} \backslash >_5^{\mathbf{D}}$. Therefore, the interpretation of $\overline{>5}$ will not be affected by the existence of other datatypes in a datatype map.

We extend OWL datatyping with datatype predicates by a revised definition of datatype groups, which was first presented in [5] and was meant to be an extension of DAML+OIL datatyping. Unlike the original definition, the revised definition of datatype groups is completely compatible with OWL datatyping. We show that the predicate conjunctions over datatype groups can be easily reduced to those over concrete domains. We then propose OWL-E, a language extending OWL DL with datatype expression axioms, as well as the datatype group-based class constructors to allow the use of datatype expressions in class restrictions. The novelty of OWL-E is that it enhances OWL DL with much more datatype expressiveness and it is still decidable. Of course, we will need a full paper to present details of the above.

References

[1] Franz Baader and Philipp Hanschke. A Schema for Integrating Concrete Domains into Concept Languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 452–457, 1991.

[2] Sean Bechhofer, Frank van Harmelen, James Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein eds. OWL Web Ontology Language Reference. URL `http://www.w3.org/TR/owl-ref/`, Feb 2004.

[3] Ian Horrocks and Ulrike Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204, 2001.

[4] C. Lutz. *The Complexity of Reasoning with Concrete Domains*. PhD thesis, Teaching and Research Area for Theoretical Computer Science, RWTH Aachen, 2001.

[5] Jeff Z. Pan and Ian Horrocks. Web Ontology Reasoning with Datatype Groups. In *Proc. of the 2nd International Semantic Web Conference (ISWC2003)*, 2003.