Reasoning Support for OWL-E

(Extended Abstract)

Jeff Z. Pan

Department of Computer Science,
University of Manchester, UK M13 9PL
pan@cs.man.ac.uk

1 Introduction

The vision of the Semantic Web [2] is to make Web resources (not just HTML pages, but a wide range of Web accessible data and services) more readily accessible to machines (or 'intelligent agents'). Ontologies play a key role in the Semantic Web by providing formal semantics to machine understandable annotations.

OWL is a newly emerged W3C recommendation of expressing ontologies in the Semantic Web, which provides three increasingly expressive sub-languages: OWL Lite, OWL DL and OWL Full. Like DAML+OIL [6], OWL Lite and OWL DL are basically very expressive DLs with an RDF [4] syntax. OWL Lite and OWL DL are very close to the $\mathcal{SHIF}(\mathbf{D}^+)$ and $\mathcal{SHOIN}(\mathbf{D}^+)$ DLs; they can thus exploit existing DL research, e.g., define the semantics of the language and to understand its formal properties, in particular the decidability and complexity of key reasoning services. OWL Full presents a more complete integration with RDF(S), but its formal properties are less well understood, and it is clearly undecidable. Therefore, when we talk about OWL reasoning, we mean OWL Lite and OWL DL.

Although OWL is very expressive, the OWL datatype formalism (or simply *OWL datatyping*) is far not expressive enough. E.g., OWL datatyping does not provide a general framework for user-defined datatypes, such as XML Schema derived datatypes [3], 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 extended abstract, we show how to extend OWL datatyping to a datatype expression formalism based on datatype groups, which provides a general way to represent user-defined datatypes and datatype predicates, based on datatype groups. The result is OWL-E, a language extending OWL DL with 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.

2 OWL and OWL Datatyping

In OWL, A *datatype* d is characterised by a lexical space, $L(d)$, which is a set of Unicode strings; a value space, $V(d)$; 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.

Datatype interpretations are relativised to a datatype map, which is a partial mapping from datatype URIs to datatypes. E.g., $\mathbf{M}_{d1} = \{\langle \textit{xsd:string}, string\rangle, \langle \textit{xsd:integer}, integer\rangle\}$ is a datatype map. A datatype URI u is called *supported* if $\mathbf{M}_p(u)$ is defined; otherwise, u is called *unsupported*. E.g., *xsd:string* is a supported datatype URI w.r.t. $\mathbf{M}_{d1}$, while *xsd:boolean* is an unsupported datatype URI w.r.t. $\mathbf{M}_{d1}$.

In OWL, $\Delta_\mathbf{D}$ (the datatype domain) w.r.t. to a datatype map is the union of all the value spaces of all the supported datatypes together with the value space of all the plain literals. Supported datatypes are interpreted as their value space, unsupported datatypes are interpreted as any subsets of $\Delta_\mathbf{D}$. Type literals are interpreted as a member of the interpretation of the associated datatypes, while plain literals are interpreted as themselves.

OWL provides a form 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 $(1 \leq i \leq n)$.

3 Motivating Examples

We provide some examples which cannot be represented by OWL datatyping.

- Ontologies and applications in the Semantic Web need to represent datatype constraints over multiple datatype properties; e.g., the sum of the $height$, $length$ and $width$ of an item should less than 15cm, or the $itemPrice$ times $quantity$ of items should equal to $totalPrice$ etc. We will extend OWL datatyping with predicates by introducing datatype groups in the next section.
- Ontologies and applications in the Semantic Web need to define user-defined datatypes and user-defined datatype predicates for their own purposes. OWL doesn't support user-defined datatypes, such as derived XML Schema, because there is no standard access mechanism for derived XML Schema datatypes, i.e., there is no standard way to access a derived XML Schema datatype via a URI reference. Although the enumerated datatypes supports some sorts of user-defined datatypes, they are far less than necessary: it is difficult to use them to define derived datatypes like the integer range between 1 to 1,000,000; furthermore, they cannot be used to define infinite

datatypes (e.g., integers less than 12 or larger than 17), not to mention user-defined predicates (such $>$ for non-negative integers). In next section, we will further introduce a more general datatype expression formalism.

4 Datatype Groups

In this section, we first extend OWL datatyping to support datatype predicates. A *datatype predicate* (or simply *predicate*) p is characterised by an arity $a(p)$, and a predicate extension $E(p)$. E.g., the integer equality '$=^{int}$' is a predicate with arity $a(=^{int}) = 2$ and predicate extension $E(=^{int}) = \{\langle i_1, i_2\rangle \in E(integer)^2 \mid i_1 = i_2\}$. It is obvious that datatypes can be seen as predicates with arity 1, and their predicate extensions are equal to their value spaces. Similar to their datatype counterparts, we have predicate maps, supported and unsupported predicates.

A *datatype group* $\mathcal{G}$ is a tuple ($\mathbf{M}_p$,$\mathbf{D}_\mathcal{G}$,dom), where $\mathbf{M}_p$ is the *predicate map* of $\mathcal{G}$, $\mathbf{D}_\mathcal{G}$ is the set of *base datatypes* URI references of $\mathcal{G}$, and dom is the *declared domain function* of $\mathcal{G}$. For convenience, we call $\Phi_\mathcal{G}$ the set of supported predicate URI references in $\mathcal{G}$, i.e., for each $u \in \Phi_\mathcal{G}$, $\mathbf{M}_p(u)$ is defined; we require $\mathbf{D}_\mathcal{G} \subseteq \Phi_\mathcal{G}$. We call $\mathbf{U}_\mathcal{G}$ the set of unsupported datatype URI references of $\mathcal{G}$, i.e., for any $u \in \mathbf{U}_\mathcal{G}$, $u \notin \Phi_\mathcal{G}$. We assume that there exists a unary predicate URI reference owlx:DatatypeBottom in $\mathbf{U}_\mathcal{G}$. The declared domain function dom is a mapping s.t. $\forall u \in \mathbf{D}_\mathcal{G}$: $\mathsf{dom}(u) = u$, and $\forall u \in \Phi_\mathcal{G}$, $\mathsf{dom}(u) \in (\mathbf{D}_\mathcal{G})^n$, where $n = a(\mathbf{M}_p(u))$. A *datatype interpretation* $\mathcal{I}_\mathbf{D}$ of a datatype group $\mathcal{G} = (\mathbf{M}_p, \mathbf{D}_\mathcal{G}, \mathsf{dom})$ is a pair $(\Delta_\mathbf{D}, \cdot^\mathbf{D})$, where $\Delta_\mathbf{D}$ (the datatype domain) is a non-empty set and $\cdot^\mathbf{D}$ is a datatype interpretation function.[1] The supported predicate URIs are interpreted as their predicate extensions, while unsupported predicates are interpreted as any subsets of $(\Delta_\mathbf{D})^n$, given an arity n.[2]

Intuitively speaking, a datatype group is a group of supported predicates ('included' in a predicate map $\mathbf{M}_p$), which can potentially be divided into different sub-groups, so that predicates in each sub-group are about the base datatype of the sub-group, where the declared domain function is used to relate supported predicate URIs to base datatype URIs. Therefore, we can make use of known decidability results about the satisfiability problem of predicate (about some datatype) conjunctions, such as the admissible/computable concrete domains [1, 5] presented in Section 2.4 of [5]. Similar to admissible concrete domains, we can define conforming datatype groups, with extra conditions in order to guarantee the possibility of constructing a corresponding concrete domain for each sub-group.

Now we introduce a datatype expression formalism based on datatype groups. Let $\mathcal{G} = (\mathbf{M}_p, \mathbf{D}_\mathcal{G}, \mathsf{dom})$ be a datatype group, the set of *$\mathcal{G}$-datatype expressions*, abbreviated $\mathbf{Dexp}(\mathcal{G})$, is inductively defined as follows:

[1] Because of limited of space, we cannot fully describe the interpretation of a datatype group in this extended abstract, which will surely be available for the full paper.

[2] n is the number of variables used with an unsupported predicate URI u; note that we do not even know the arity of the predicate u represents.

Abstract Syntax	DL Syntax	Semantics
domain$(u_1, \ldots, u_n)$	$(u_1, \ldots, u_n)$	$u_1^{\mathbf{D}} \times \ldots \times u_n^{\mathbf{D}}$
$And(P, Q)$	$P \sqcap Q$	$P^{\mathbf{D}} \cap Q^{\mathbf{D}}$
$Or(P, Q)$	$P \sqcup Q$	$P^{\mathbf{D}} \cup Q^{\mathbf{D}}$

Table 1. Semantics of $\mathcal{G}$-Datatype Expressions

Abstract Syntax	DL Syntax	Semantics
restriction($\{T\}$ someTuplesSatisfy(P))	$\exists T_1, \ldots, T_n.P$	$\{x \in \Delta^{\mathcal{I}} \mid \exists t_1, \ldots, t_n.\langle x, t_i \rangle \in T^{\mathcal{I}}$ (for all $1 \leq i \leq m) \wedge \langle t_1, \ldots, t_n \rangle \in P^{\mathbf{D}}\}$
restriction($\{T\}$ allTuplesSatisfy(P))	$\forall T_1, \ldots, T_n.P$	$\{x \in \Delta^{\mathcal{I}} \mid \forall t_1, \ldots, t_n.\langle x, t_i \rangle \in T^{\mathcal{I}}$ (for all $1 \leq i \leq m) \rightarrow \langle t_1, \ldots, t_n \rangle \in P^{\mathbf{D}}\}$
restriction($\{T\}$ maxCardinality(m) someTuplesSatisfy(P))	$\geqslant m T_1, \ldots, T_n.P$	$\{x \in \Delta^{\mathcal{I}} \mid \sharp\{\langle t_1, \ldots, t_n \rangle \mid \langle x, t_i \rangle \in T^{\mathcal{I}}$ (for all $1 \leq i \leq m) \wedge \langle t_1, \ldots, t_n \rangle \in P^{\mathbf{D}}\} \geq m\}$
restriction($\{T\}$ minCardinality(m) someTuplesSatisfy(P))	$\leqslant m T_1, \ldots, T_n.P$	$\{x \in \Delta^{\mathcal{I}} \mid \sharp\{\langle t_1, \ldots, t_n \rangle \mid \langle x, t_i \rangle \in T^{\mathcal{I}}$ (for all $1 \leq i \leq m) \wedge \langle t_1, \ldots, t_n \rangle \in P^{\mathbf{D}}\} \leq m\}$

Table 2. Datatype Group-based Concepts

1. $\forall u \in \Phi_{\mathcal{G}} \cup \mathbf{U}_{\mathcal{G}} : u, \in \mathbf{Dexp}(\mathcal{G})$ (with arity $a(\mathbf{M}_p(u))$ if $u \in \Phi_{\mathcal{G}}$);
2. $\forall u_1, \ldots, u_n \in$ sub-group(w) where $w \in \mathbf{D}_{\mathcal{G}}$ and $a(\mathbf{M}_p(u_i)) = 1$ (for all $1 \leq i \leq n$): $(u_1, \ldots, u_n) \in \mathbf{Dexp}(\mathcal{G})$ (with arity n);
3. $\forall P, Q \in \mathbf{Dexp}(\mathcal{G})$ with the same arity n: $P \sqcap Q, P \sqcup Q \in \mathbf{Dexp}(\mathcal{G})$.

For any $u \in \Phi_{\mathcal{G}} \cup \mathbf{U}_{\mathcal{G}}$, the semantics of u has been described above. The abstract syntax and semantics of the rest $\mathcal{G}$-datatype expressions are given in Table 1. Let P be a $\mathcal{G}$-datatype expression with an arity n, the negation of P is of the form $\neg P$, and is interpreted as $(\Delta_{\mathbf{D}})^n \setminus P^{\mathbf{D}}$. It can be proved that for a conforming datatype group $\mathcal{G}$, the satisfiability problem of $\mathcal{G}$-datatype expression conjunctions is decidable.

5 OWL-E

We can extend OWL DL to OWL-E by introducing four new datatype group-based concept constructors listed in Table 2 (where $T_1, \ldots, T_n$ are datatype properties and P is a $\mathcal{G}$-datatype expression). Even though OWL-E is much more expressive than OWL DL, in the sense of datatype expressiveness, OWL-E is still decidable. In fact, it can be proved that we can combine any decidable description logics that provide (at least) the conjunction and bottom constructors with the four datatype group-based concept constructors (listed in Table 2), the combined logics are still decidable.

Bibliography

[1] F. Baader and P. 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] T. Berners-Lee. Realising the Full Potential of the Web. W3C Document, `URL http://www.w3.org/1998/02/Potential.html`, Dec 1997.

[3] P. V. Biron and A. Malhotra. Extensible Markup Language (XML) Schema Part 2: Datatypes – W3C Recommendation 02 May 2001. Technical report, World Wide Web Consortium, 2001. Available at `http://www.w3.org/TR/xmlschema-2/`.

[4] D. Brickley and R. Guha. Resource Description Framework (RDF) Schema Specification 1.0. W3C Recommentdation, `URL http://www.w3.org/TR/rdf-schema`, Mar. 2000.

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

[6] F. van Harmelen, P. F. Patel-Schneider, and I. Horrocks. Reference Description of the DAML+OIL(March 2001) Ontology Markuk Language. DAML+OIL Document. Available at `http://www.daml.org/2000/12/reference.html`, Mar. 2001.