Practical Conforming Datatype Groups♦

David Turner, Jeremy Carroll
Digital Media Systems Laboratory
HP Laboratories Bristol
HPL-2007-37
March 14, 2007*

datatype group, decidability, OWL

Datatype groups are an extension to certain Description Logics (DLs) that permit the user to reason about *n*-tuples of data, where $n \geq 1$, and thus to express complex constraints on multiple properties of objects. A conforming datatype group has the appropriate computational properties to preserve the soundness, completeness, and compactness of the reasoning procedure of the underlying DL. We consider examples of datatype groups that appear useful in practice, and illustrate the care that must be taken to ensure that datatype groups are both conforming and practical.

* Internal Accession Date Only
♦ 20th International Workshop on Description Logics (DL-2007) 8-10 June 2007, Brixen-Bressanone, Italy
Approved for External Publication

Practical Conforming Datatype Groups

Id: DL-2007.tex 31 2007-02-28 09:14:59Z turdavid

Dave Turner and Jeremy Carroll

{DavidT,Jeremy.Carroll}@hp.com
HP Labs Bristol
Filton Road
Stoke Gifford
Bristol BS34 8QZ

Abstract. Datatype groups are an extension to certain Description Logics (DLs) that permit the user to reason about n-tuples of data, where $n \geq 1$, and thus to express complex constraints on multiple properties of objects. A conforming datatype group has the appropriate computational properties to preserve the soundness, completeness, and compactness of the reasoning procedure of the underlying DL. We consider examples of datatype groups that appear useful in practice, and illustrate the care that must be taken to ensure that datatype groups are both conforming and practical.
Keywords: Datatype Group, Decidability, OWL

1 Introduction

OWL 1.0 [1] is a standardised ontology language for machine interpretability of Web content, with strong theoretical foundations in Description Logics (DLs). Currently under development is a proposal for an improved language, OWL 1.1, with extensions over OWL 1.0 that reflect both the theoretical state-of-the-art in DLs and end-user experiences of OWL 1.0.

The proposed OWL 1.1 language is based on the DLs $\mathcal{SROIQ}$ [2] and $\mathcal{SHOQ}(\mathrm{D_n})$ [3]. The DL name fragment '$(\mathrm{D_n})$' represents datatype groups, which are essentially collections of predicates over n-tuples of data where $n \geq 1$ [4]. To ensure satisfactory computational properties, attention is restricted to *conforming* datatype groups: a datatype group $\mathcal{G}$ is conforming if it satisfies some closure properties and, importantly, the appropriate satisfiability problem over $\mathcal{G}$ is decidable. The actual solution of this satisfiability problem is performed by a black-box decision procedure known as a type-checker.

This modular approach to datatype groups makes for a smooth theoretical presentation. Indeed, the word 'conforming' appears just once in all the discussions of $\mathcal{SHOQ}(\mathrm{D_n})$ [3] and $\mathcal{SROIQ}$ [2], and does not appear at all in the current OWL 1.1 proposal. We will consider examples of datatype groups that appear useful in practice, and illustrate the care that must be taken to ensure that one does not introduce an unconforming system.

2 Definitions

A datatype group $\mathcal{G}$ is [4] a tuple $(\Delta_{\mathbf{D}}, \mathbf{D}_{\mathcal{G}}, \Phi^1_{\mathcal{G}}, \Phi_{\mathcal{G}})$, where Δ_D is the datatype domain covering all datatypes, and $\mathbf{D}_{\mathcal{G}}$, $\Phi^1_{\mathcal{G}}$ and $\Phi_{\mathcal{G}}$ are (presumably) disjoint sets of names of base datatypes, derived datatypes, and predicates respectively. A base datatype name $d \in \mathbf{D}_{\mathcal{G}}$ is interpreted as $d^{\mathbf{D}} \subseteq \Delta_{\mathbf{D}}$; a datatype $d' \in \Phi^1_{\mathcal{G}}$ derived from d is interpreted as $d'^{\mathbf{D}} \subseteq d^{\mathbf{D}}$. Each predicate $P \in \Phi_{\mathcal{G}}$ is given an arity $n > 1$ and interpreted as $P^{\mathbf{D}} \subseteq d_1^{\mathbf{D}} \times \ldots \times d_n^{\mathbf{D}} \subseteq \Delta_{\mathbf{D}}^n$. A finite predicate conjunction over $\mathcal{G}$ is a statement $\mathscr{C}$ of the form

$$\bigwedge_{j=1}^{k} p_j(v_1^{(j)}, \ldots, v_{n_j}^{(j)}) \tag{1}$$

where p_j is an n_j-ary predicate in $\mathbf{D}_{\mathcal{G}} \cup \Phi^1_{\mathcal{G}} \cup \Phi_{\mathcal{G}}$ and the $v_i^{(j)}$ are variables. A *solution* for $\mathscr{C}$ is a function mapping the variables to elements of $\Delta_{\mathbf{D}}$ such that $\langle \delta(v_1^{(j)}, \ldots, \delta(v_{n_j}^{(j)}) \rangle \in p_j^{\mathbf{D}}$ for each j. $\mathscr{C}$ is said to be *satisfiable* if it has a solution. The datatype group $\mathcal{G}$ is said to be *conforming* if

1. $\mathbf{D}_{\mathcal{G}}$, $\Phi^1_{\mathcal{G}}$ and $\Phi_{\mathcal{G}}$ are closed under negation,
2. a binary inequality predicate $\neq_d \in \Phi_{\mathcal{G}}$ is defined for each datatype $d \in \mathbf{D}_{\mathcal{G}}$, and importantly
3. the satisfiability of finite predicate conjunctions over $\mathcal{G}$ is decidable.

Given two datatype groups $\mathcal{G}_1$, $\mathcal{G}_2$, one can merge them to form a datatype group

$$\mathcal{G}_1 \oplus \mathcal{G}_2 = (\Delta_{\mathbf{D}}, \mathbf{D}_{\mathcal{G}_1} \cup \mathbf{D}_{\mathcal{G}_2}, \Phi^1_{\mathcal{G}_1} \cup \Phi^1_{\mathcal{G}_2}, \Phi_{\mathcal{G}_1} \cup \Phi_{\mathcal{G}_2}). \tag{2}$$

3 Examples

3.1 Miles and Kilometers

Pan and Horrocks [4] present a datatype group containing a predicate for converting between kilometers and miles as follows.

$$\begin{aligned} \mathcal{G}_1 = (&\Delta_{\mathbf{D}}, \\ &\{\texttt{float}, \overline{\texttt{float}}\}, \\ &\{\texttt{lengthInMile}, \overline{\texttt{lengthInMile}}, \texttt{lengthInKMtr}, \overline{\texttt{lengthInKMtr}}\}, \\ &\{\texttt{kmtrsPerMile}, \overline{\texttt{kmtrsPerMile}}, =_{\texttt{float}}, \neq_{\texttt{float}}\}) \end{aligned} \tag{3}$$

The datatype `float` is a standard part of XML Schema Part 2 [5]; the derived datatypes `lengthInMile` and `lengthInKmtr` consist of nonnegative `float`s and these may be declared using XML Schema syntax too. The following is their suggested XML syntax for declaring the `kmtrsPerMile` predicate.

```
<predicate name="kmtrsPerMile" arity="2">
    <par var="i" base="lengthInKmtr" />
    <par var="j" base="lengthInMile" />
    <constraint val="i=1.6*j" />
</predicate>
```

For clarity, we will abbreviate declarations of this kind as follows.

$$\texttt{kmtrsPerMile} = D(\texttt{i}, \texttt{j}, \texttt{"i=1.6*j"}) \tag{4}$$

3.2 Small Objects

Pan and Horrocks [3] present an alternative motivating example, where an object is classified as 'small' for postal purposes if its height is less than `5cm` and the sum of its length and width is less than `10cm`. No datatype group is explicitly given for this example, so we suggest the following.

$$\begin{aligned}\mathcal{G}_2 = (&\Delta_{\mathbf{D}},\\ &\{\texttt{float}, \overline{\texttt{float}}\},\\ &\{\texttt{heightInCm}, \texttt{widthInCm}, \texttt{lengthInCm}, \texttt{heightLessThan5cm},\\ &\quad \overline{\texttt{heightInCm}}, \overline{\texttt{widthInCm}}, \overline{\texttt{lengthInCm}}, \overline{\texttt{heightLessThan5cm}}\},\\ &\{\texttt{sumLengthAndWidthLessThan10cm}, =_{\texttt{float}},\\ &\quad \overline{\texttt{sumLengthAndWidthLessThan10cm}}, \neq_{\texttt{float}}\})\end{aligned} \tag{5}$$

As before, it is possible to declare all of the unary parts of this datatype group using the existing tools of XML Schema. These tools do not extend to the declaration of the binary predicate `sumLengthAndWidthLessThan10cm`. Adapting the syntax of the previous example might result in the following.

$$\texttt{sumLengthAndWidthLessThan10cm} = D(\texttt{i}, \texttt{j}, \texttt{"i+j<10"}) \tag{6}$$

4 Discussion

That datatype groups are conforming is important to ensure the decidability of reasoning in Description Logics such as $\mathcal{SHOQ}(\mathrm{D_n})$ [3] and by extension in $\mathcal{SROIQ}$ [2] and hence in the proposed OWL 1.1 standard. There has been only limited discussion regarding how to construct and deal with conforming datatype groups in practice. In particular, the permitted syntax of the arithmetic expressions in the declarations above is unspecified. While small expressions of the form `i=1.6*j` and `i+j<10` may seem innocuous, one does not have to work much harder to construct non-conforming datatype groups. For example, consider a datatype group with the following ternary predicates,

$$\texttt{integerAddition} = D(\texttt{i}, \texttt{j}, \texttt{k}, \texttt{"i=j+k"}) \tag{7}$$

and

$$\texttt{integerMultiplication} = D(\texttt{i}, \texttt{j}, \texttt{k}, \texttt{"i=j*k"}) \tag{8}$$

viewed as predicates over $\left(\texttt{integer}^{\mathbf{D}}\right)^3$. With this datatype group one can ask for integer solutions to the polynomial equation $3x^2 + 5y - 11 = 0$ using the DL

concept

$$\begin{array}{l}\exists\langle n_0, c_1, n_1\rangle.\texttt{integerAddition} \sqcap \\ \exists\langle n_1, n_2, n_3\rangle.\texttt{integerAddition} \sqcap \\ \exists\langle n_2, c_2, n_4\rangle.\texttt{integerMultiplication} \sqcap \\ \exists\langle n_3, c_3, y\rangle.\texttt{integerMultiplication} \sqcap \\ \exists\langle n_4, x, x\rangle.\texttt{integerMultiplication} \sqcap \\ \exists c_1.=_{-11} \sqcap \exists c_2.=_3 \sqcap \exists c_3.=_5 \sqcap \exists n_0.=_0\end{array} \tag{9}$$

A type-checker for this datatype group must therefore be able to find integer roots of arbitrary integer polynomials; unfortunately, this problem is known to be undecidable [6]. Notice that the datatype groups

$$\begin{array}{l}\mathcal{G}_3 = (\Delta_{\mathbf{D}}, \\ \quad \{\texttt{decimal}, \overline{\texttt{decimal}}\}, \\ \quad \{\texttt{integer}, \overline{\texttt{integer}}\}, \\ \quad \{\texttt{integerAddition}, =_{\texttt{decimal}}, \\ \quad\ \overline{\texttt{integerAddition}}, \neq_{\texttt{decimal}}\})\end{array} \tag{10}$$

and

$$\begin{array}{l}\mathcal{G}_4 = (\Delta_{\mathbf{D}}, \\ \quad \{\texttt{decimal}, \overline{\texttt{decimal}}\}, \\ \quad \{\texttt{integer}, \overline{\texttt{integer}}\}, \\ \quad \{\texttt{integerMultiplication}, =_{\texttt{decimal}}, \\ \quad\ \overline{\texttt{integerMultiplication}}, \neq_{\texttt{decimal}}\})\end{array} \tag{11}$$

are conforming, but their merge $\mathcal{G}_3 \oplus \mathcal{G}_4$ is not, which demonstrates that conformingness is not preserved by merging in general.

Lemma 6 of [4] says that $\mathcal{G}_3 \oplus \mathcal{G}_4$ is conforming if both $\mathcal{G}_3$ and $\mathcal{G}_4$ are conforming, and $\mathbf{D}_{\mathcal{G}_3} \cap \mathbf{D}_{\mathcal{G}_4} = \varnothing$, but in this case $\texttt{decimal} \in \mathbf{D}_{\mathcal{G}_3} \cap \mathbf{D}_{\mathcal{G}_4}$. This is not an artificial situation: in the context of XML Schema Part 2 [5] there are a small, finite number of base datatypes and in practice one would expect to have to merge datatype groups over intersecting sets of base datatypes reasonably often. It would not be possible to completely forbid intersecting merges of this kind without compromising the open-world assumption of the Semantic Web. Furthermore, without both multiplication and addition, end-users would not even be able to convert between °F and °C:

$$\texttt{farenheitToCelsius} = D(\texttt{f}, \texttt{c}, \texttt{"f=1.8*c+32.0"}) \tag{12}$$

A suggested practical alternative to arithmetic expressions is to use URIs to refer to a fixed collection of predicates. Sirin [7] suggests

`http://www.w3.org/2003/11/swrlb#greaterThan = swrlb:greaterThan`

or

`http://www.w3.org/TR/xpath-functions/#func-numeric-greater-than`

instead of

$$D(\texttt{i}, \texttt{j}, \texttt{"i>j"}) \tag{13}$$

However, `swrlb:add` and `swrlb:multiply` (and their XPath equivalents) are URIs for addition and multiplication predicates that can be restricted to the problematic `integerAddition` and `integerMultiplication` above. Therefore datatype groups built from too much of the `swrlb` or XPath function namespaces cannot be conforming.

4.1 Conforming, but Impractical, Datatype Groups

There are a number of known decidability results about arithmetic that have less practical impact than they might appear to have on first inspection. Firstly, in contrast with integer arithmetic, arithmetic over (the existential theory of) $\mathbb{R}$ is known to be decidable [8]. However, XML Schema does not attempt to implement a `real` type, only `decimal`, and the status of arithmetic over `decimal` is unknown. Secondly, the satisfiability of finite predicate conjunctions over a datatype group which only uses the XML Schema datatypes `float`, `double`, `long` (and its derived types `int`, `short` and `byte`) and `unsignedLong` (and its derived types `unsignedInt`, `unsignedShort` and `unsignedByte`) is trivially decidable since each of those datatypes is finite. In practice this result does not give rise to a usable decision procedure: an exhaustive search for a solution is not feasible.

4.2 XML Schema-specific Implementation Constraints

An example from [4] declares that the Yangtze river is 3937.5 miles long and uses the `kmtrsPerMile` predicate to deduce that it is also 6300.0km long. In other words,

$$\langle 6300.0, 3937.5 \rangle \in [\![\texttt{kmtrsPerMile}]\!]. \tag{14}$$

This example uses the XML Schema datatype `float` to represent lengths. Suppose that the Yangtze was declared instead to be 3937.501 miles long, then

$$\langle 6300.0015, 3937.501 \rangle \in [\![\texttt{kmtrsPerMile}]\!] \tag{15}$$

so the Yangtze river may be deduced to be 6300.0015km long. However,

$$\langle 6300.0015, 3937.5007 \rangle \in [\![\texttt{kmtrsPerMile}]\!], \tag{16}$$

so that the Yangtze river may also be deduced to be 3937.5007 miles long. This would be inconsistent with the user's expectation that a river has only one length.

As pointed out in [4], there are well over a hundred length units, and rounding errors caused by round-tripping values through all of the associated conversions can accumulate into significant errors. We implemented a system to do conversions between `float`s representing lengths in kilometers, meters, centimeters, millimeters, micrometers, inches, feet, yards, fathoms, poles, chains, furlongs, statute miles, leagues and nautical miles and deduced the length of the Yangtze

to be both 6335.3584km and 6361.8555km[1], and nearly 800,000 other values, starting from a declaration that its length in miles is 3937.5. These rounding errors were highly dependent on the structure of the definitions of the units, as multiplication in `float` is not associative so scalar multiplication operators on `float` do not commute. This lack of associativity also demonstrates that the (necessarily associative) composition of two datatypes like `kmtrsPerMile` and, say, `milesPerLeague` cannot be the same as the composition of the underlying arithmetic operations; again, this is likely to be inconsistent with a user's expectations. In short, the behaviour of fixed-precision floating-point datatypes with arithmetic in OWL is likely to be a source of confusion amongst users.

Additionally, suppose the Volga river were declared to be 3668.8003km long, then it would have no value for its `lengthInMile` property at all, since

$$\begin{array}{l} \langle 3668.8000, 2293.0000 \rangle \in [\![\texttt{kmtrsPerMile}]\!] \\ \langle 3668.8005, 2293.0002 \rangle \in [\![\texttt{kmtrsPerMile}]\!] \\ \text{and } \nexists x \in \texttt{float}.\ 2293.0 < x < 2293.0002 \end{array} \tag{17}$$

Again, this situation would be contrary to the user's expectation that one can always convert freely (albeit possibly inaccurately) between miles and kilometers. Notice that this cannot be remedied by using the arbitrary-precision `decimal` instead of the fixed-precision `float`: for example the temperature 75.0°F has no corresponding `decimal` representation in °C.

In practice, many applications do not require the declarative style of arithmetic that datatypes like `kmtrsPerMile` would allow. Instead, a procedural approach is adequate. For example, a user may be happy that the Volga can be deduced to be 2293.0km long, and may be equally happy with 2293.0002km, as long as only one of the options is chosen. One method that has been used to achieve this would be to embed conversion instructions as literals in an ontology[9], which makes it clear to a user that the semantics of arithmetic is separated from that of the DL.

5 Conclusion

The extension of a DL with conforming datatype groups extends its expressive power whilst preserving its decidability and the efficacy of known reasoning techniques. Datatype groups are motivated by the requirements of DL users to be able to express complex constraints simultaneously on multiple data values.

However, there has been little discussion regarding instances of datatype groups that may be capable of satisfying these user requirements whilst also being conforming and computationally feasible. We have shown that this question is far from trivial, and that care is also necessary to keep within the open-world assumption of the Semantic Web by allowing these datatype groups to freely merge with each other. Furthermore although the proposed OWL 1.1 standard

[1] These figures are different from the 6300km quoted previously because the conversion factor from miles to kilometers is not quite 1.6.

includes support for datatype groups, it offers no suitable syntax and we have shown that some 'obvious' choices turn out to be undecidable or otherwise unsuitable.

Finally we have demonstrated that (XML Schema-based) datatypes that represent notionally bijective functions are in general neither injective nor surjective because of implementation constraints. In a procedural context, such functions are 'sufficiently' bijective for many applications, but in a declarative context like that of DLs this imprecision manifests itself more seriously. We have shown that, for example, it may lead to serious rounding errors, and may cause properties to fail to satisfy expected cardinality constraints and exhibit other unexpected behaviour.

References

1. McGuinness, D.L., van Harmelen, F.: OWL web ontology language overview. W3C recommendation, W3C (2004) http://www.w3.org/TR/2004/REC-owl-features-20040210/.
2. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006), AAAI Press (2006) 57–67
3. Pan, J.Z.: Web Ontology Reasoning in the SHOQ(Dn) Description Logic. In: Carlos Areces and Maartin de Rijke, editors,Proceedings of the Methods for Modalities 2 (M4M-2). (2001) ILLC, University of Amsterdam.
4. Pan, J.Z., Horrocks, I.: Web Ontology Reasoning with Datatype Groups. In Fensel, D., Sycara, K., Mylopoulos, J., eds.: Proc. of the 2nd International Semantic Web Conference (ISWC2003). (2003)
5. Malhotra, A., Biron, P.V.: XML schema part 2: Datatypes second edition. W3C recommendation, W3C (2004) http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/.
6. Davis, M.: Hilbert's tenth problem is unsolvable. American Mathematical Monthly **80** (1973) 233–269
7. Sirin, E.: Re: n-ary datatypes (2007) http://lists.w3.org/Archives/Public/public-owl-dev/2007JanMar/0106.html.
8. Björner, A., Las Vergnas, M., Sturmfels, B., White, N., Ziegler, G.M.: Oriented Matroids (Second Edition). Cambridge University Press (2006)
9. van der Veen, M., Reynolds, D., Seaborne, A.: Re: Mathematics (2006-7) http://tech.groups.yahoo.com/group/jena-dev/messages/25376.