Extending Datatype Support for Tractable Reasoning with OWL 2 EL Ontologies

Pospishnyi Oleksandr[1],

[1] National Technical University of Ukraine "Kyiv Polytechnic Institute", Kiev, Ukraine
pospishniy@kpi.in.ua

Abstract. It was mentioned on multiple occasions that datatype expressions are a necessary component of any production quality knowledge base and will surely play a major role in the upcoming Semantic Web. This paper briefly summarizes the work on improving the support for tractable reasoning with datatype expressions in ELK - a highly efficient $\mathcal{EL}$ reasoner. Tests have shown an exceptional speed of ontology classification of great size which opens up new perspectives for applying ontologies with datatype expressions in practice.

Keywords: ontology, dataypes, reasoning, EL++, ELK

1 Background

In Description Logics, *datatypes* (also called concrete domains) can be used to define new concepts by referring to particular values, such as strings or integers. For example the following axioms from computer hardware ontology provide definitions for the notions of dual-, quad-, and many-core processors:

$$\mathbf{DualCoreCPU} \equiv \mathbf{CPU} \sqcap \exists \mathbf{hasCores}.(=, \mathbf{2})$$
$$\mathbf{QuadCoreCPU} \equiv \mathbf{CPU} \sqcap \exists \mathbf{hasCores}.(=, \mathbf{4})$$
$$\mathbf{MultiCoreCPU} \equiv \mathbf{CPU} \sqcap \exists \mathbf{hasCores}.(>, \mathbf{1})$$

In mentioned example, $(>, \mathbf{1})$ refers to the domain of natural numbers and the relation > is used to constrain possible values to those larger than 1. Restriction $(=, \mathbf{2})$ uses the relation = to constrain the value to element 2, and similarly for $(=, \mathbf{4})$.

Any ontology reasoner, with a support of datatype expressions, as presented above, should be able to derive new axioms such as:

$$\mathbf{DualCoreCPU} \sqsubseteq \mathbf{MultiCoreCPU}$$
$$\mathbf{QuadCoreCPU} \sqsubseteq \mathbf{MultiCoreCPU}$$

In order to ensure that reasoning remains polynomial, logic $\mathcal{EL}^{++}$ allows only for datatype restrictions that *cannot* implicitly express concept disjunction, which is a well-known cause of intractability. Consider, for example, the axioms:

$$\mathbf{CPU} \sqsubseteq \exists \mathbf{hasCores}.(\geq, \mathbf{1})$$
$$\mathbf{SingleCoreCPU} \equiv \mathbf{CPU} \sqcap \exists \mathbf{hasCores}.(=, \mathbf{1})$$
$$\mathbf{MultiCoreCPU} \equiv \mathbf{CPU} \sqcap \exists \mathbf{hasCores}.(\geq, \mathbf{2})$$

It could be seen, that these axioms imply the disjunction:

$$\mathbf{CPU} \sqsubseteq \mathbf{SingleCoreCPU} \sqcup \mathbf{MulticoreCPU}$$

To prevent such situations, the EL Profile of OWL 2, which is based on $\mathcal{EL}^{++}$, admits only equality $(=)$ in datatype restrictions. Unfortunately, it would be almost impossible to adequately model knowledge about real world domains using ontologies with such severe limitations on allowed datatype expressions.

It was recently demonstrated by D. Magka, Y. Kazakov and I. Horrocks [1] that the mentioned restrictions could be significantly relaxed without losing tractability. This paper introduced a notion of "safety" for datatype restrictions and classified all safe combinations of datatype restrictions for the domain of natural numbers, integers, rational and real numbers. Conducted theoretical work opened up an opportunity to implement datatype support for $\mathcal{EL}$ reasoners that would be both, safe and practically useful.

But more work still needs to be done. Namely, a large amount of ontologies relies on other common datatypes, such as date/time, strings, binary data, URIs, etc:

$$\mathbf{E30\text{-}1280} \sqsubseteq \mathbf{Xeon} \sqcap \exists \mathbf{releaseDate}.(=, \mathbf{2011\text{-}06\text{-}03})$$
$$\mathbf{E30\text{-}1280} \equiv \exists \mathbf{hasPartNo}.(=, \text{"}\mathbf{BX80623E31280}\text{"})$$
$$\mathbf{E30\text{-}1280} \equiv \exists \mathbf{hasCPUID}.(=, \mathbf{0206D7h}),$$

restrictions on them:

$$\exists \mathbf{hasPartNo}.(\mathbf{pattern}, \mathbf{'CM806230*'}) \sqsubseteq \mathbf{Xeon}$$
$$\mathbf{IntelCPU} \sqsubseteq \exists \mathbf{hasPartNo}.(\boldsymbol{length}_{=}, \mathbf{15}),$$

and interval relations, such as:

$$\exists \mathbf{releaseDate}.(\geq \mathbf{2011\text{-}01\text{-}01}, \leq \mathbf{2011\text{-}12\text{-}31})$$
$$\exists \mathbf{hasCores}.(\geq \mathbf{2}, < \mathbf{8}).$$

Thus, a goal was set to use the approach, presented in [1], and extend it to include all datatypes from the OWL 2 EL profile, as well as complex datatype restrictions, all without compromising the tractability.

Recent studies [2] showed that the abovementioned datatype expressions are already widely used in different ontologies all over the Internet, despite their poor support by most reasoners. It also has been often mentioned that datatype assertions would be a major part of the Semantic Web and lack of their support would greatly hamper its development.

Newly developed reasoning procedures were implemented and tested in ELK – a state of the art tractable $\mathcal{EL}$ reasoner [3].

2 Technical approach

According to Web Ontology Language specification, OWL 2 EL profile provides for 19 various datatypes, most of which are defined in XML Schema Definition Language (XSD) specification. Figure 1 provides a summary of all datatypes, allowed by the OWL 2 EL profile and displays their inheritance and a set of allowed facet restrictions.

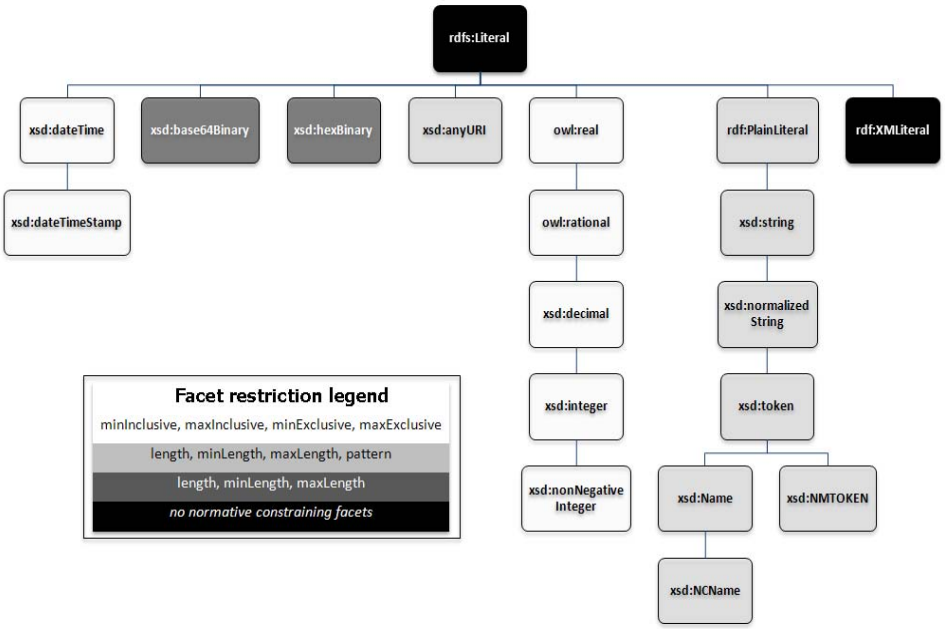

Fig. 1. Datatypes allowed by the OWL 2 EL profile

It is worth mentioning, that OWL 2 EL profile does not allow the following datatypes: *xsd:double, xsd:float, xsd:nonPositiveInteger, xsd:positiveInteger, xsd:negativeInteger, xsd:long, xsd:int, xsd:short, xsd:byte, xsd:unsignedLong, xsd:unsignedInt, xsd:unsignedShort, xsd:unsignedByte, xsd:language* and *xsd:boolean.* The set of supported datatypes has been designed such that the intersection of the value spaces of any set of these datatypes is either empty or infinite, which is necessary to obtain the desired computational properties.

In order to support reasoning with abovementioned allowed datatypes, a set of new inference rules was implemented for the ELK reasoner:

$$\frac{C \sqsubseteq \exists R.\, r_+}{C \sqsubseteq D} \qquad \exists R.\, r_- \sqsubseteq D \in \mathcal{O},\ \ r_+ \subseteq r_-$$

$$\frac{}{C \sqsubseteq \bot} \qquad C \sqsubseteq \exists R.\, r_+ \in \mathcal{O},\ \ r_+ = \emptyset$$

where $\mathcal{O}$ is an ontology, C and D are the concepts, R is a datatype property and $\exists R.\, r_{\pm}$ denotes an existential datatype expression created with constraining facets. Symbols $_+$ and $_-$ indicate that an expression is occurring right or left of $\sqsubseteq$ respectively.

With $r_+ \subseteq r_-$ we represent a fact that a value space constrained by the datatype restriction r_+ is a subset of a value space constrained by the r_- datatype restriction, or in other words r_- includes r_+. With $r_+ = \emptyset$ we represent an empty value space produced by the datatype restriction r_+.

In the proposed implementation, all datatype expressions are parsed with respect to their lexical form and then transformed to internal representation that reflects the nature of their respective value space. All possible value spaces, created by the datatype restrictions, could be conventionally divided into 3 categories:

- *Values*: binary value, date/time value, literal value and numeric value.
- *Intervals*: numeric interval, date/time interval and length restriction
- *Other*: empty value space, entire value space and pattern

During the computation of all conclusions under the inference rules, for each encountered $\exists R.\, r_+$ expression a search is conducted for all $\exists R.\, r_-$ expressions in the ontology where $r_+ \subseteq r_-$. To increase the efficiency of reasoning, a special datatype index is used to optimize the search for all such $\exists R.\, r_-$ expressions.

Table 1 summarizes all possible $r_+ \subseteq r_-$ scenarios where *A* represents r_- datatype restriction, and *B* represents r_+ datatype restriction.

My means of $B_D \subseteq A_D$ expression we state that the datatype of a restriction *B* is equal to or inherited from the datatype of a restriction *A*, for example $\mathrm{xsd{:}\,integer} \subseteq \mathrm{owl{:}\,real}$. Expression A_{low} and A_{high} denote a minimum and maximum value implied by the restriction *A*, while B_{len} denotes a length of a corresponding value. Finally, by $B \xrightarrow{A} \notin \emptyset$ expression we represent a fact that literal value *A* matches a corresponding pattern restriction *B*.

3 Evaluation

The evaluation of proposed modifications was conducted using a large OWL 2 EL ontology that was generated by the Grid-DL Semantic Grid information service [4].

Two test ontologies were considered. The full ontology consisted of 1,087,124 axioms, 65 classes, 33 object properties, 109 datatype properties and 131,637 individual assertions. Truncated version of the ontology connsisted of only 230,670 axioms and 36,191 individual assertions.

Three classes were added to act as a "query" to the knowledge base:

```
UK_Site ≡ Site and hasLocation some
    (Location and hasName some string[pattern ".*, UK"])
Idle_CE ≡ ComputingElement and hasState some
    (CEState and hasRunningJobs value 0 and hasWaitingJobs value 0
     and hasFreeJobSlots some integer[>0])
     and hasState some (CEState and hasStatus value Production)
x64_Cluster ≡ SubCluster and (describedBy some
     (hasPlatformType value "x86_64"^^string) and
     (describedBy some (hasRAMSize some integer[>= 4096, <= 8192])))
```

Table 2 presents the test results. For comparison the Pellet [5] and HermiT [6] reasoners were included, both capable of reasoning on OWL 2 EL ontologies with datatype expressions. The following test setup was used: Intel Core 2 Duo T9300 @ 2.50GHz, 4 Gb RAM, OpenSUSE 12.1 (Linux 3.1.10), Java 1.7.0_05 (-Xmx3200M -Xms3200M), Pellet 2.2.0, HermiT 1.3.6. All tests were conducted three times and then averaged.

Table 2. Datatype restrictions subsumption matrix

A \ B	Empty ValueSp.	Entire ValueSp.	Binary Value	DateTime Value	Literal Value	Numeric Value	DateTime Interval	Length Restriction	Numeric Interval	Pattern
Empty ValueSp	-	-	-	-	-	-	-	-	-	-
Entire ValueSp	$B_D \subseteq A_D$	$B_D \subseteq A_D$	$B_D \subseteq A_D$	$B_D \subseteq A_D$	$B_D \subseteq A_D$	$B_D \subseteq A_D$	$B_D \subseteq A_D$	$B_D \subseteq A_D$	$B_D \subseteq A_D$	$B_D \subseteq A_D$
Binary Value	-	-	$B_D = A_D$ $A = B$	-	-	-	-	-	-	-
DateTime Value	-	-	-	$A =^1 B$	-	-	-	-	-	-
Literal Value	-	-	-	-	$A =^2 B$	-	-	-	-	-
Numeric Value	-	-	-	-	-	$A =^3 B$	-	-	-	-
DateTime Interval	-	-	-	$B_D \subseteq A_D$ $A_{low} \leq B$ $A_{high} \geq B$	-	-	$B_D \subseteq A_D$ $A_{low} \leq B_{low}$ $A_{high} \geq B_{high}$	-	-	-
Length Restrict.	-	-	$B_D \subseteq A_D$ $A_{low} \leq B_{len}$ $A_{high} \geq B_{len}$	-	$B_D \subseteq A_D$ $A_{low} \leq B_{len}$ $A_{high} \geq B_{len}$	-	-	$B_D \subseteq A_D$ $A_{low} \leq B_{low}$ $A_{high} \geq B_{high}$	-	$B_D \subseteq A_D$ $B \subseteq^4 A$
Numeric Interval	-	-	-	-	-	$B_D \subseteq A_D$ $A_{low} \leq^3 B$ $A_{high} \geq^3 B$	-	-	$B_D \subseteq A_D$ $A_{low} \leq^3 B_{low}$ $A_{high} \geq^3 B_{high}$	-
Pattern	-	-	-	-	$B_D \subseteq A_D$ $B \xrightarrow{A} \notin \emptyset$	-	-	$B_D \subseteq A_D$ $B \subseteq^4 A$	-	$B_D \subseteq A_D$ $B \subseteq^4 A$

[1] Equality evaluation takes into account a position of A and B on the timeline with respect to specified time zones. If time zone is specified only for one parameter, a ± 14:00 offset is used in evaluation.

[2] Literals are considered to be equal when all characters of their lexical form are equal to each other, both missing a language tag and/or datatype tag or (if present) they are equal for both literals.

[3] During comparison all numbers are cast to common, most specific datatype.

[4] Verify that every interpretation of regular expression (restriction) B will satisfy a regular expression (restriction) A. All restrictions are viewed as regular expressions.

Table 2. Evaluation results

Reasoner	Truncated ontology classification time, ms	Full ontology classification time, ms
ELK	7366	54912
ELK[5]	5598	12567
Pellet	165419	out of mem
HermiT	timeout	timeout

Unfortunately, HermiT reasoner could not classify the ontology within a 1 hour timeout constraint and Pellet ran out of memory while processing the full ontology. Evidently, ELK greatly outperformed abovementioned reasoners due to its efficient reasoning procedures.

The profiling analysis showed that it is possible to considerably speed up the reasoning procedures by using only one type of literals in the ontology, for example *xsd:string*. If such requirements are met, simplified literal handling algorithm could yield a considerable performance boost.

Currently the work is focused on designing and implementing an ontology analysis tool that would be capable of detecting unsafe datatype expressions in the processed ontology. It would ensure that ontology being reasoned upon does not contain conflicting datatype expressions that might cause incomplete results. In case if implicit disjunction is detected, a warning would be issued to the user, informing him about the source of the problem.

The source code of modified version of ELK with the support of tractable datatype reasoning can be found in `elk-parent-datatypes` branch in the official ELK repository: https://code.google.com/p/elk-reasoner.

4 References

1. *D. Magka, Y. Kazakov, I. Horrocks.* Tractable Extensions of the Description Logic $\mathcal{EL}$ with Numerical Datatypes. Journal of Automated Reasoning 47(4), Pp. 427–450, Springer, 2011.
2. *B. Glimm, A. Hogan, M. Krötzsch, A. Polleres.* OWL: Yet to Arrive on the Web of Data?. arXiv preprint, 2012.
3. *Y. Kazakov, M. Krötzsch, F. Simančík.* Concurrent Classification of EL Ontologies. In: Proc. of the 10th International Semantic Web Conference (ISWC-11). LNCS 7032, Springer, 2011.
4. *O. Pospishniy, S. Stirenko.* GRID-DL: Semantic GRID Information Service. In Proc. 9th OWL Experiences and Directions Workshop (OWLED). Heraklion, Crete, 2012.
5. *B. Parsia and E. Sirin.* Pellet: An OWL-DL Reasoner. In Proc. ISWC 2004, Hiroshima, Japan. 2004.
6. *R. Shearer, B. Motik, and I. Horrocks*. HermiT: a Highly-Efficient OWL Reasoner. In Proc. 5th OWL Experiences and Directions Workshop (OWLED). Karlsruhe, Germany, 2008. Pp. 26-27

[5] Simplified literal reasoning algorithm