dlpconvert

Converting OWL DLP statements to logic programs

Boris Motik
FZI
Karlsruhe, Germany
motik@fzi.de

Denny Vrandecic
AIFB, University of Karlsruhe
Karlsruhe, Germany
denny@aifb.uni-karlsruhe.de

Pascal Hitzler
AIFB, University of Karlsruhe
Karlsruhe, Germany
hitzler@aifb.uni-karlsruhe.de

York Sure
AIFB, University of Karlsruhe
Karlsruhe, Germany
sure@aifb.uni-karlsruhe.de

Rudi Studer
AIFB, University of Karlsruhe
FZI
Karlsruhe, Germany
studer@aifb.uni-karlsruhe.de

1. MOTIVATION

In the past few years, the W3C has been establishing standards for the fundamental building blocks of the Semantic Web. With RDF a common data model has been provided. A major step was the introduction of the ontology definition languages RDFS — offering very basic semantics — and the Web Ontology Language OWL As these standards are still very new, only a limited support of tools for inferencing and querying exists. Still, the availability of such tools will be crucial for the development of the Semantic Web.

The OWL fragments OWL DL and even OWL Lite suffer from a very high computational complexity, and the efficient reasoners able to deal with these fragments in a sufficiently scalable way are one of the main challenges for the community. In the meantime, efficient tools existing for other knowledge representation paradigms may provide sufficient support in order to work with OWL, provided suitable translation tools are at hand. Embracing the idea of "small can be beautiful" [1], we can indeed gain access to a strong and almost immediate tool support — the price we pay are a few limitation, which in practise will only very rarely limit the actual modelling possibilities, as we will see.

Our approach rests on choosing the DLP fragment [1], of OWL DL, described in [2, 3]. It is the intersection — in an intuitive sense — of OWL DL and (Horn) logic programming. As such it imposes certain constraints on OWL DL in order to guarantee that all axioms stated are transformable in an efficient way to Horn clauses, i.e. rules in the sense of traditional logic programming. These logic programs may then be interpreted efficiently with standard Prolog systems like XSB-Prolog or SWI-Prolog. They may also be fed to F-Logic based systems like Ontobroker. These systems are already implemented and can be used out of the box.

By restricting attention to DLP, we make significant gains in reasoning complexity, and can use standard Prolog systems e.g. for efficient (ABox) query answering. At the same time, the semantic expressivity lost due to the restriction to DLP hardly matters in practice. Indeed [3] analysed that 99% of the axioms in ontologies taken from the daml.org repository are within the DLP fragment.

Another major advantage of DLP is its future stability. Currently there exist two major trends for ontology representation, namely description logics — with OWL being the most prominent one — and logic programming, particularly F-Logic. Both have benefits and drawbacks and allow for different usage scenarios, which is why there are currently several efforts being undertaken for joining both paradigms into a single framework. Even though the W3C can generate huge impact due to its role in the standardisation of web technologies, it remains to be seen how OWL will develop in future years, and whether it provides for sufficient practical use. The semantic incompatibility of OWL DL and RDFS may also hamper its development DLP on the other hand provides maximal flexibility, as it can easily be translated from one paradigm to the other. We can even decide to use the most suitable approach based on the task at hand. It remains fully reusable, as DLP is a proper subset of the W3C standard OWL, but is also a natural subset of other paradigms like F-Logic.

2. TOOL

In order to facilitate the use of DLP, and to display its full potential, we provide the syntax conversion tool `dlpconvert`[2]. It allows to convert OWL encoded DLP fragments into (Edinburgh) logic programming syntax, as used by standard Prolog systems.

`dlpconvert` is based on the algorithms for reducing description logics to Datalog implemented in KAON2[3] and described in [5]

[1] `http://logic.aifb.uni-karlsruhe.de`

[2] `logic.aifb.uni-karlsruhe.de/dlpconvert/`
[3] `http://kaon2.semanticweb.org`

and, in greater length, in [6]. It reads in an OWL ontology, reduces it to disjunctive Datalog and finally serialises it into a logic program, which can be used for easier reading and thus understanding by people with an appropriate logic background or as input for Prolog interpreters.

`dlpconvert` comes as a command line tool, implemented in Java 5, with numerous switches for different kinds of name transformations and serialisation options. It can be used to convert an OWL DL file directly into a Prolog program file, that can be consumed by a Prolog interpreter as it is. In addition to the command line tool, there is a Tomcat-powered online conversion available on the DLP website, which is a thin wrapper around the `dlpconvert` java package. You may choose to either supply a URL for an ontology, upload a file from your local hard disk or even write (or copy and paste) an ontology directly into the website. Your ontology will be converted and the result shown within an HTML page.

3. EXAMPLE

We provide an example to display the capabilities of `dlpconvert`. In philosophy, the probably most classical example for inferencing is the following syllogism:

All humans are mortal.
Socrates is a human.
Therefore Socrates is mortal.

This syllogism, along with another fact surrounding Socrates – that the author of the *Politeia* is actually Plato, and not Socrates – is formalized in the following OWL file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Ontology [
 <!ENTITY ex
  "http://logic.uni-karlsruhe.de/dlpconvert/example1#">]>
<owlx:Ontology owlx:name="&ex;"
    xmlns:owlx="http://www.w3.org/2003/05/owl-xml#">
  <owlx:Class owlx:name="#human" owlx:complete="false">
    <owlx:Class owlx:name="#mortal"/>
  </owlx:Class>
  <owlx:Individual owlx:name="#Socrates">
    <owlx:type owlx:name="#human" />
  </owlx:Individual>
  <owlx:ObjectProperty owlx:name="#isauthor">
    <owlx:domain owlx:class="#human"/>
    <owlx:range owlx:class="#book"/>
  </owlx:ObjectProperty>
  <owlx:Individual owlx:name="#Plato">
    <owlx:ObjectPropertyValue owlx:property="#isauthor">
      <owlx:Individual owlx:name="#Politeia" />
    </owlx:ObjectPropertyValue>
  </owlx:Individual>
</owlx:Ontology>
```

This example is also provided on the `dlpconvert` website. Processing it with `dlpconvert` yields the following result.

```
mortal(X) :- human(X).
book(Y) :- isauthor(X, Y).
human(X) :- isauthor(X, Y_0).
isauthor(plato, politeia).
human(socrates).
```

Both representations actually have the same meaning, which is the point of the syntactic translation. The second is undoubtedly shorter, and for many readers the syntax is much easier to understand, and for every person with a background in logic programming the meaning of the second representation is immediately clear.

We can directly feed this result to a Prolog engine, like XSB-Prolog[4]. This way, the XSB system understands the semantics of the original OWL file and we can ask questions about the given ontology, as in the following example.

```
?- mortal(socrates).
yes
?- isauthor(X, politeia).
X = plato
yes
?- human(plato).
yes
```

The system dutifully gives us the correct answer: Socrates is a mortal. Asking XSB about the author of the *Politeia*, it will answer correctly. Better yet, it knows that Plato too is a human, because the domain of the *isauthor* relationship is *human* (note that we never stated explicitly, that Plato is a human). This can be used in several ways, for example for questioning knowledge bases (as we did) or for checking the consistency of an ontology.

4. FUTURE WORK

As of writing this document, `dlpconvert` is still in a beta version and quite a number of features are waiting to be implemented. This includes minor changes like adding more sophisticated error messages and switches for the website version. More substantially, we intend to provide an F-Logic serialization for the output in order to facilitate interoperability with F-Logic based systems like Ontobroker, and thereby an integration of OWL DLP into F-Logic reasoners. We also intend to improve the implemented heuristics for generating names. The goal is to provide better and more flexible naming support, especially for automatically generated ontologies that usually have insignificant names for concepts and instances. Also, namespaces should be considered in order to avoid clashes.

5. REFERENCES

[1] Rousset, M.C.: Small can be beautiful in the semantic web. In: Proceedings of the 3rd International Semantic Web Conference (ISWC 2004), Hiroshima, Japan (2004)

[2] Grosof, B., Horrocks, I., Volz, R., Decker, S.: Description logic programs: Combining logic programs with description logics. In: Proc. of WWW 2003, Budapest, Hungary, May 2003, ACM (2003) 48–57

[3] Volz, R.: Web Ontology Reasoning with Logic Databases. PhD thesis, Institute AIFB, University of Karlsruhe (2004)

[4] Pan, J.: Description Logics: Reasoning Support for the Semantic Web. PhD thesis, University of Manchester (2004)

[5] Hustadt, U., Motik, B., Sattler, U.: Reducing SHIQ-Description Logic to Disjunctive Datalog Programs. In Dubois, D., Welty, C., Williams, M.A., eds.: Proc. of the 9th Int. Conf. on Knowledge Representation and Reasoning (KR2004), Menlo Park, California, USA, AAAI Press (2004) 152–162

[6] Hustadt, U., Motik, B., Sattler, U.: Reasoning for Description Logics around SHIQ in a Resolution Framework. Technical Report 3-8-04/04, FZI, Karlsruhe, Germany (2004) http://www.fzi.de/wim/publikationen.php?id=1172.

[4] `http://xsb.sourceforge.net`