DLP – An introduction

Denny Vrandečić, Peter Haase, Pascal Hitzler, York Sure, Rudi Studer
Institute AIFB
University of Karlsruhe, Germany
dlp@aifb.uni-karlsruhe.de

Abstract

DLP – *Description Logic Programs* – is the name for the common language that is able to integrate knowledge bases described in Description Logic with Logic Programs. In this introduction, we offer a very short overview of DLP, the motivation for it, the benefits it offers and how to use it.

1 Introduction

In the past few years, the W3C has established 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 only very basic elements for describing concepts and their relationships – and the powerful Web Ontology Language OWL, with the sublanguages OWL DL and OWL Lite, which are based on Description Logics [Baader *et al.*, 2003]. But OWL DL (and even OWL Lite) suffer from a very high computational complexity, and efficient reasoners able to deal with them in a sufficiently scalable way are still one of the main challenges for the community.

As these standards are still very new, quite a number of different knowledge modelling languages continue to exist, mostly based on knowledge representation paradigms other than Description Logics. Another major paradigm is F-Logic [Kifer *et al.*, 1995; Angele and Lausen, 2004], which is strongly related to logic and object-oriented programming, and allows for the modelling of knowledge by implication rules, a feature missing in OWL.

2 Motivation

When choosing a knowledge representations language today, one has either to go for a not yet efficiently implemented, but standardized language, thus committing to wait until reasonable efficient tools are available, or for efficient systems building on non-standardized languages, which are already well established. For applying semantic technologies, the practitioner is faced with the question, which of the above — or other — paradigms should be adopted. For performance reasons, simpler paradigms are often preferable, but equally important are the interoperability between different systems and the future reusability.

This is where DLP enters the scene. Knowledge bases in DLP – roughly speaking – lie within the intersection of Decription Logics and Logic Programming. So DLP provides the formal link between the two major knowledge representation paradigms, and serve multiple purposes in theoretical research and practical applications. Following the path of increasing expressivity, DLP defines precisely the point of no return before committing to one of the paradigms. Since their first introduction [Grosof *et al.*, 2003], a variety of research projects have begun to investigate and use the language.

The major advantage of DLP is its obvious future stability with both currently existing major trends for ontology representation, description logics (OWL DL) and logic programming (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. DLP offers this framework, heading for maximal flexibility, as it can easily be translated from one paradigm to the other. We can decide freely to use the most suitable approach based on the task at hand, or even mix both paradigms as we like. With DLP we gain the ability to use well-tested and efficient implementations with a modern W3C standardized language.

DLP is not yet another knowledge representation language, but rather a bridge to bring two paradigms together. 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, but it does not define new semantics or syntax.

3 Benefits

As we have seen, by using DLP we offer maximal flexibility towards possible future developments in knowledge representation. This instantly brings another major benefit: as the semantics remain the same within DLP, we can use tools from both worlds – the well-equipped world of F-Logic and the heavily researched world of DL – in order to generate, evolve, present, work with and reason over DLP knowledge bases.

By providing syntactical converters we are able to join the power of the newest OWL tools and the efficient and stable logic program interpreters like OntoBroker, SWI- or XSB-Prolog. Efficient reasoners like these, existing already today, will provide sufficient support in order to work with OWL

immediately. The price we pay are a few limitation, which in practise will only very rarely limit the actual modelling possibilities, as we will show in the next section.

On the other side, every OWL DLP ontology is a OWL DL ontology as well. DLP is a proper subset of OWL DL, staying totally truthful to the standardised semantics – unlike other existing proprietary extensions, which change the agreed-on semantics and thus break the available tool and hamper compatibility and exchange. Thus we remain faithful to some identified basic requirements for the Semantic Web, like the Open World Assumption, Classical Negation and the No Unique Name Assumption.

Finally, the constraints imposed on DLP regarding the more expressive OWL DL, lead to DLP enjoying a much better data complexity (polynomial) and combined complexity (exptime). This allows to expect more efficient and responsive tools than full OWL DL reasoning will ever be able to achieve due to the complexity of DL reasoning algorithms. Not only do we achieve to have an immediate tool support, but we can also state that reasoning over DLP will always lead to a lower complexity than reasoning over DL could.

In the next section we will take a look at how restricting the constraints that lead from DL to DLP really are in applied ontologies.

4 Usage

By restricting attention to DLP, we make significant gains in reasoning complexity, and get access to a much broader support with tools. At the same time, the semantic expressivity lost due to the restriction to DLP hardly matters in practice: [Volz, 2004] analysed that 99% of the axioms in ontologies taken from the daml.org repository are within the DLP fragment.

In order to use DLP, there is no need for special software. Standard OWL editors can be used to create valid DLP ontologies. As most of the common used DL constructors, especially those used for light-weight ontologies or taxonomies with relations, are within DLP anyway, DLP has a head-start on users compared to alternative approaches to gain higher efficiency in reasoning.

We provide a KAON2 based tool that converts DLP ontologies written in OWL/XML or RDF/XML syntax to Logic program syntax, retaining the semantics[1], thus allowing the use of standard out-of-the-box LP reasoners for DLP reasoning. Further we plan to integrate DLP into our OWL evolution framework evOWLution [Haase *et al.*, 2004] in order to provide the means to develop OWL ontologies with the guarantee to remain within DLP. Finally adapting existing ontology editors in order to help the user to account for DLP automatically is planned, thus enabling the standard user to fully leverage the power of DLP. Due to the semantic entailment of DLP in DL, there is no need for recreating the whole range of ontology management tools, but only small adjustments – if any – of existing software will allow to reap full benefit of the existing systems and methodologies.

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

5 Language Description

OWL DLP is not easily defined, but in order to facilitate the usage of OWL DLP, in this section a description of an easy to use sublanguage of DLP is provided by listing all the constructors you may use freely in an OWL ontology without running the risk of leaving DLP (taking for granted that you stick to the usual OWL DL constraints).

This doesn't mean that other constructors are forbidden in OWL DLP. It is just that their use underlies further constraints which can't be described in this short introduction.

Allowed OWL constructors: *Class, Thing, subClassOf, Property, subPropertyOf, domain, range, Individual, equivalentClass, equivalentProperty, sameAs, differentFrom, AllDifferent, ObjectProperty, DatatypeProperty, inverseOf, TransitiveProperty, SymmetricProperty, FunctionalProperty. InverseFunctionalProperty, intersectionOf*

As stated, this is by no means a description of the whole DLP language. But it is supposed to help tremendously by making you feel safe with the usage of the constructors.

6 Further Reading

The basic DLP language was described initially in [Grosof *et al.*, 2003]. [Volz, 2004] provides a much deeper definition and analysis of DLP, especially evaluating the scope of the identified constraints and their practical meaning. A website with pointers to further resources is being maintained at `http://logic.aifb.uni-karlsruhe.de`. Questions may be sent to dlp@aifb.uni-karlsruhe.de.

References

[Angele and Lausen, 2004] Jürgen Angele and Georg Lausen. Ontologies in F-logic. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 29–50. Springer, 2004.

[Baader *et al.*, 2003] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[Grosof *et al.*, 2003] Benjamin Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logics. In *Proc. of WWW 2003, Budapest, Hungary, May 2003*, pages 48–57. ACM, 2003.

[Haase *et al.*, 2004] Peter Haase, York Sure, and Denny Vrandecic. Ontology management and evolution – survey, methods and prototype. SEKT formal deliverable D3.1.1, Institute AIFB, University of Karlsruhe, December 2004.

[Kifer *et al.*, 1995] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843, 1995.

[Volz, 2004] Raphael Volz. *Web Ontology Reasoning with Logic Databases*. PhD thesis, AIFB, University of Karlsruhe, 2004. `http://www.ubka.uni-karlsruhe.de/vvv/2004/wiwi/2/2.pdf`.