Artificial Intelligence 174 (2010) 1093–1141

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Logic-based ontology comparison and module extraction, with an application to *DL-Lite*

Roman Kontchakov [a,*], Frank Wolter [b], Michael Zakharyaschev [a]

[a] *Department of Computer Science and Information Systems, Birkbeck College London, UK*
[b] *Department of Computer Science, University of Liverpool, UK*

ARTICLE INFO

Article history:
Received 31 July 2009
Received in revised form 30 April 2010
Accepted 17 June 2010
Available online 23 June 2010

Keywords:
Description logic
Ontology
Module extraction
Entailment
Computational complexity
Uniform interpolation
Forgetting

ABSTRACT

We develop a formal framework for comparing different versions of ontologies, and apply it to ontologies formulated in terms of *DL-Lite*, a family of 'lightweight' description logics designed for data-intensive applications. The main feature of our approach is that we take into account the vocabulary (= signature) with respect to which one wants to compare ontologies. Five variants of difference and inseparability relations between ontologies are introduced and their respective applications for ontology development and maintenance discussed. These variants are obtained by generalising the notion of conservative extension from mathematical logic and by distinguishing between differences that can be observed among concept inclusions, answers to queries over ABoxes, by taking into account additional context ontologies, and by considering a model-theoretic, language-independent notion of difference. We compare these variants, study their meta-properties, determine the computational complexity of the corresponding reasoning tasks, and present decision algorithms. Moreover, we show that checking inseparability can be automated by means of encoding into QBF satisfiability and using off-the-shelf general purpose QBF solvers.

Inseparability relations between ontologies are then used to develop a formal framework for (minimal) module extraction. We demonstrate that different types of minimal modules induced by these inseparability relations can be automatically extracted from real-world medium-size *DL-Lite* ontologies by composing the known tractable syntactic locality-based module extraction algorithm with our non-tractable extraction algorithms and using the multi-engine QBF solver AQME. Finally, we explore the relationship between uniform interpolation (or forgetting) and inseparability.

1. Introduction

In computer science, ontologies are used to provide a common vocabulary (or, in logic parlance, signature) for a domain of interest, together with a description of certain relationships between terms built from the vocabulary. Ontology languages based on description logics represent ontologies as 'TBoxes' (terminological boxes) containing inclusions between complex concepts over the vocabulary [2]. An increasingly important application of ontologies is management of large amounts of data, where ontologies are used to provide flexible and efficient access to repositories consisting of data sets of instances of concepts and relations. In description logics, such repositories are typically modelled as 'ABoxes' (assertion boxes) [2].

* Corresponding author.
E-mail addresses: roman@dcs.bbk.ac.uk (R. Kontchakov), frank@csc.liv.ac.uk (F. Wolter), michael@dcs.bbk.ac.uk (M. Zakharyaschev).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2010.06.003

Developing and maintaining ontologies for this and other purposes is a rather difficult task. When using description logics (including the description logic based dialects of the Web Ontology Language OWL[1]), the ontology designer is supported by efficient reasoning tools for classification, instance checking and a variety of other reasoning tasks. However, this support is generally recognised to be insufficient when ontologies are developed not as 'monolithic entities' but by means of importing, merging, combining, refining and extending already existing ontologies. In all those cases, reasoning support for analysing the impact of the respective operation on the ontology would be extremely useful. Typical examples of such 'unorthodox' reasoning services include the following:

Comparing versions of ontologies. The standard syntactic `diff` utility is an indispensable tool for comparing different versions of text files, and it would be very helpful to have a similar versioning tool for ontologies. However, a purely syntactic operation of computing the difference between ontologies is of little value [3] because our concern now is not the syntactic form of the ontologies, but their differing logical consequences. Moreover, instead of comparing arbitrary logical consequences, it is more useful and informative to compare logical consequences over the *common vocabulary* Σ of the versions, or even such consequences regarding a certain subject matter corresponding to some subvocabulary of Σ. Thus, the reasoning service we need in this case should be able to compare the logical consequences of different versions of ontologies over some vocabulary Σ.

Ontology refinement. When refining an ontology by adding new axioms, one usually wants to preserve the relationships between terms of a certain part Σ of its vocabulary. The reasoning service required in such a case is to check whether the refined ontology has precisely the same logical consequences over Σ as the original one.

Ontology re-use. When importing an ontology, one wants to *use* its vocabulary Σ as originally defined. However, relationships between terms over Σ may change due to interaction with some axioms in the importing ontology. So again we need a reasoning service capable of checking whether new logical consequences over Σ are derivable (this service has been termed *safety checking* in [4]).

In all these and many other cases, we are interested in comparing logical consequences over some vocabulary Σ that can be drawn from two different ontologies. This gives rise to the three main notions we investigate in this paper: *Σ-difference*, *Σ-entailment*, and *Σ-inseparability*. Roughly, the Σ-difference between two ontologies is the set of 'formulas' over Σ that are derivable from one ontology but not from the other; one ontology Σ-entails another one if all Σ-formulas derivable from the latter are also derivable from the former; and two ontologies are Σ-inseparable if they Σ-entail each other.

In the discussion so far, we have not specified the language from which the logical consequences over Σ are drawn. This language depends on the application. For example, if one is mainly interested in terminological reasoning and differences visible in applications that use relationships between concepts, then an appropriate language is the set of all concept inclusions. The Σ-difference then consists of all concept inclusions over Σ derivable from one ontology but not from the other. And one ontology Σ-entails another ontology if every concept inclusion over Σ derivable from the latter is derivable from the former. If, however, one is mainly interested in using ontologies to query instance data, then it is more appropriate to consider a language for consequences over Σ that reflects, in some way, answers to queries in the signature Σ (or Σ-queries) over instance data in Σ. In this case, two ontologies should be Σ-inseparable if, and only if, they give the same answers to every Σ-query in the chosen language for any instance data over Σ. Even this language may be insufficient for applications where different versions of ontologies are imported into a context ontology, in which case two ontologies should be deemed Σ-inseparable only if *after* importing them into another ontology over Σ, the resulting extensions still give the same answers to Σ-queries.

The *first aim* of this paper is to give precise formalisations of five variants of Σ-difference, Σ-entailment and Σ-inseparability for ontologies given in the *DL-Lite* logics $\textit{DL-Lite}^{\mathcal{N}}_{bool}$ and $\textit{DL-Lite}^{\mathcal{N}}_{horn}$. These variants of Σ-difference and Σ-entailment are obtained by distinguishing between differences visible among concept inclusions, answers to queries over ABoxes, by taking additional context ontologies into account, and by considering model-theoretic, language-independent notions of Σ-difference and Σ-entailment.

The *DL-Lite* family of description logics [5–8] has been originally designed with the aim of providing query access to large amounts of data via a high-level conceptual (ontological) interface. Thus, the *DL-Lite* logics result from various compromises between (i) the necessity of retaining the data complexity of query answering as close as possible to the complexity of standard database query evaluation and (ii) the desire of having the expressive means for representing various constraints of data modelling formalisms such as the ER model and UML class diagrams [9]. For example, the logic $\textit{DL-Lite}^{\mathcal{N}}_{bool}$ [10] (containing many other *DL-Lite* logics) can express is-a hierarchies of concepts, disjointness and covering constraints for concepts, and domain, range and cardinality constraints for binary relations. Instance checking in $\textit{DL-Lite}^{\mathcal{N}}_{bool}$ is in AC^0 for data complexity (i.e., of the same complexity as database query evaluation); however, answering conjunctive queries is coNP-complete. On the other hand, $\textit{DL-Lite}^{\mathcal{N}}_{horn}$ cannot express covering constraints, but boasts AC^0 query answering (under the unique name assumption) [11]. To simplify presentation, in this paper we do not consider *DL-Lite* logics with role inclusions, focusing mainly on the impact of the Boolean constructs in concept inclusions as well as number restrictions.

[1] http://www.w3.org/2007/OWL/.