Available online at www.sciencedirect.com
ScienceDirect
Data & Knowledge Engineering 63 (2007) 200–223
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

Reasoning and change management in modular ontologies

Heiner Stuckenschmidt [a,*], Michel Klein [b]

[a] *Universität Mannheim, Germany*
[b] *Vrije Universiteit Amsterdam, The Netherlands*

Received 13 December 2006; accepted 5 February 2007
Available online 20 February 2007

Abstract

The benefits of modular representations are well known from many areas of computer science. While in software engineering modularization is mainly a vehicle for supporting distributed development and re-use, in knowledge representation, the main goal of modularization is efficiency of reasoning. In this paper, we concentrate on the benefits of modularization in the context of ontologies, explicit representations of the terminology used in a domain. We define a formal representation for modular ontologies based on the notion of Distributed Description Logics and introduce an architecture that supports local reasoning by compiling implied axioms. We further address the problem of guaranteeing the correctness and completeness of compiled knowledge in the presence of changes in different modules. We propose a heuristic for analyzing changes and their impact on compiled knowledge and guiding the process of updating compiled information that can often reduce the effort of maintaining a modular ontology by avoiding unnecessary recompilation.

Keywords: Ontologies; Reasoning; Distributed Knowledge Representation; Change management

1. Motivation

Currently, research in the area of the semantic web is in a state where ontologies are ready to be applied in real applications such as semantic web portals, information retrieval or information integration. In order to lower the effort of building ontology-based applications, there is a clear need for a representational and computational infrastructure in terms of general purpose tools for building, storing and accessing ontologies. A number of such tools have been developed, i.e. ontology editors [1,2], reasoning systems [3,4] and more recently storage and query systems (e.g. [5]). Most of these tools, however, treat ontologies as monolithic entities and provide little support for specifying, storing and accessing ontologies in a modular manner.

* Corresponding author. Address: Universität, Mannheim, Institut für Informatik, A5, 6, 68159 Mannheim, DE, Germany. Tel.: +49 621 181 2530.
E-mail address: heiner@informatik.uni-mannheim.de (H. Stuckenschmidt).

0169-023X/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2007.02.001

1.1. Why modularization?

There are many reasons for thinking about ontology modularization. Our work is mainly driven by three arguments. These also bias the solution we propose, as it focuses on the following aspects.

Distributed Systems: In distributed environments like the semantic web, the question for modularization arises naturally. Ontologies in different places are built independent of each other and can be assumed to be highly heterogeneous. Unrestricted referencing of concepts in a remote ontology can therefore lead to serious semantic problems as the domain of interpretation may differ even if concepts appear to be the same on a conceptual level. The introduction of modules with local semantics can help to overcome this problem.

Large Ontologies: Modularization is not only desirable in distributed environments, it also helps to manage very large ontologies that we find in medicine or biology. These ontologies, which sometimes contain more than a hundred thousand concepts, are hard to maintain as changes are not contained locally but can affect large parts of the model. Another argument for modularization in the presence of large ontologies is re-use: in most cases, we are not interested in the complete ontology when building a new system, but only in a specific part. Experiences from software engineering shows that modules provide a good level of abstraction to support maintenance and re-use.

Efficient reasoning: A specific problem with distributed ontologies as well as with very large models is the efficiency of reasoning. While the pure size of the ontologies causes problems in the latter case, hidden dependencies and cyclic references can cause serious problems in a distributed setting. The introduction of modules with local semantics and clear interfaces will help to analyze distributed systems and provides a basis for the development of methods for localizing inference.

1.2. Requirements

There are three requirements a modular ontology architecture has to fulfill in order to improve ontology maintenance and reasoning in the way suggested above. The requirements will be the main guidelines for the design of our solution proposed in this work.

Loose Coupling: In general, we cannot assume that two ontology modules have anything in common. This holds for the conceptualization as well as for the interpretation of objects, concepts or relations. Our architecture has to reflect this by providing an extremely loose coupling of modules. In particular, we have to prevent unwanted interactions between modules. For this purpose, mappings between modules have to be distinguished from local definitions on the semantic as well as the conceptual level.

Self-Containment: In order to facilitate the re-use of individual modules from a larger, possibly interconnected system, we have to make sure that modules are self-contained. In particular, it should be possible to perform certain reasoning tasks such as subsumption or query answering within a single module without having to access other modules. This is also important if we want to provide efficient reasoning. Further, we have to ensure correctness, and whenever possible completeness, of local reasoning, for obvious reasons.

Integrity: The advantages of having self-contained ontology modules have their price in terms of potential inconsistencies that arise from changes in other ontology modules. While there is in our architecture no need to access other modules at reasoning time, the correctness of reasoning within a self contained module may still depend on knowledge in other ontologies. If this knowledge changes, reasoning results in a self-contained module may become incorrect with respect to the overall system, and we will not even notice it. We have to provide mechanisms for checking whether relevant knowledge in other systems has changed and for adapting the reasoning process if needed to ensure correctness.

1.3. Related work

Our work relates to two main areas of research on representing and reasoning about ontological knowledge. The first is concerned with distributed and modular knowledge representation where we use ideas from