Formal Properties of Modularisation

Boris Konev¹, Carsten Lutz², Dirk Walther¹, and Frank Wolter¹

¹ University of Liverpool, UK
{kon, dwalther, wolter}@liverpool.ac.uk
² University of Bremen, Germany
clu@informatik.uni-bremen.de

Summary. Modularity of ontologies is currently an active research field, and many different notions of a module have been proposed. In this paper, we review the fundamental principles of modularity and identify formal properties that a robust notion of modularity should satisfy. We explore these properties in detail in the contexts of description logic and classical predicate logic and put them into the perspective of well-known concepts from logic and modular software specification such as interpolation, forgetting and uniform interpolation. We also discuss reasoning problems related to modularity.

2.1 Introduction

The benefits of modular ontologies are manifold. In ontology design, modularity supports the structured and controlled development of large ontologies, enables ontology design by multiple, possibly distributed designers and allows the re-use of (parts of) already existing ontologies. In ontology deployment and usage, modularity can be exploited for right-sizing large ontologies (by selecting and using only the relevant part) and to speed up reasoning. Alas, making full use of modularity is hampered by the fact that there are many different definitions of what a module in an ontology actually is. In fact, it seems unlikely that there can be a unique and generally accepted such definition because the desirable properties of a module strongly depends on the intended application.

In this paper, our aim is to provide guidance for choosing the right notion of modularity. In particular, we give a survey of possible options and identify three formal properties that a notion of modularity may or may not enjoy and that can be used to evaluate the robustness of this notion in the context of a given application. We analyze whether the surveyed notions of modularity satisfy these robustness properties and provide further guidance by discussing the computational complexity of central decision problems associated with modularity.

To make different notions of modularity comparable to each other, we have to agree on some general framework for studying ontologies and modules. Generally
speaking, a module is a part of a complex system that functions independently from this system. To define what a module in an ontology is, we thus have to specify what it means for such a module to function independently from the containing ontology. It is not a priori obvious how this can be done. We start with adopting the following, abstract view of an ontology: an ontology $O$ can be regarded as a black box that provides answers to queries about some vocabulary $S$ of interest. The form that such queries take is one of the main distinguishing factors between different applications. Important cases include the following.

**Classification.** Classifying an ontology means to compute the sub-/superclass relationships between all atomic classes in the ontology. For example, if $S$ is a vocabulary for buildings and architecture, then the query Church $\sqsubseteq$ Building asks whether every church is a building.

**Subsumption queries.** In other applications, one is interested in computing subsumption between complex class expressions. For example, if $S$ is a biological vocabulary, then the query Father $\sqsubseteq$ Living being $\sqcap$ $\exists$ has child $\top$ asks whether every father is a living being having a child.

**Instance data.** A popular application of ontologies is their use for providing a background theory when querying instance data. In this case, one is interested in instance queries that are posed to a knowledge base, which consists of an ontology and an ABox that stores instances of classes and relations. Note that we do not consider the ABox to be part of the ontology. To represent this setup in terms of queries posed to the ontology, we consider queries that consist of an instance query together with an ABox. For example, if $S$ is a geographical vocabulary, then a query might consist of the instance data

$$\mathcal{A} = \{ \text{Country(France), Country(Columbia), \ldots, LocatedinEurope(France), \ldots} \}$$

together with the conjunctive query EuropeanCountry(France). This query asks whether it follows from the instance data $\mathcal{A}$ and the ontology that France is a European country. The answer is yes if, for example, the ontology states that every country LocatedinEurope is a EuropeanCountry.

These examples show that, to define what it means for a part of an ontology to “function independently”, we first have to fix a query language and a vocabulary of interest. Once this is done, two ontologies can be regarded equivalent if they give the same answers to all queries that can be built in the fixed query language with the fixed vocabulary. Similarly, given an ontology and its module, we can say that the module functions independently from the ontology if any query built in the fixed query language with the vocabulary associated with the module has the same answer when querying the module and the whole ontology.

Formally, these ideas can be captured by the concept of inseparability: given a query language $\mathcal{QL}$ and a vocabulary $S$, two ontologies $O_1$ and $O_2$ are $S$-inseparable w.r.t. $\mathcal{QL}$ if they give the same answers to queries in $\mathcal{QL}$ over $S$. There are various