Extracting Modules from Ontologies: A Logic-based Approach

Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov and Ulrike Sattler
The University of Manchester
School of Computer Science
Manchester, M13 9PL, UK
{bcg, horrocks, ykazakov, sattler}@cs.man.ac.uk

1 Introduction

The design, maintenance, reuse, and integration of ontologies are highly complex tasks—especially for ontologies formulated in a logic-based language such as OWL. Like software engineers, "ontology engineers" need to be supported by tools and methodologies that help them to minimise the introduction of errors, i.e., to ensure that ontologies have appropriate consequences. In order to develop this support, important notions from software engineering, such as *module*, *black-box behavior*, and *controlled interaction*, need to be adapted so as to take into account the fact that an OWL ontology is, in essence, a logical theory; due to the expressive power of OWL, this turns out to be difficult.

In this paper, we focus on the use of modularity to support the *partial reuse* of ontologies. Given a foreign ontology from which we are reusing a set of symbols, our goal is to *extract*, from the foreign ontology, a small fragment that captures the meaning of the terms we use in our ontology. For example, when building an ontology describing research projects, we may use terms such as Cystic_Fibrosis and Genetic_Disorder in our descriptions of medical research projects. In order to improve the precision of our ontology, we may want to add more detail about the meaning of these terms; for reasons of cost and accuracy, we would prefer to do this by reusing information from a medical ontology. Such ontologies are, however, typically very large, and importing the whole ontology would make the consequences of the additional information costly to compute and difficult for our ontology engineers (who are not medical experts) to understand. Thus, in practice, we need to extract a (hopefully small) module that includes just the relevant information. More specifically, we will require that, when answering a query (expressed in a fixed query language) against our project ontology, importing the module would give us *exactly the same answers* as if we had imported the whole medical ontology. In this case, importing the module instead of the whole ontology will have no observable effect on our ontology.

In this paper, we propose a definition of a *module* $\mathcal{Q}_1$ within a given ontology $\mathcal{Q}$ for a given vocabulary $\mathbf{S}$. We show that, for OWL DL, which is a syntactical variant of the Description Logic $\mathcal{SHOIN}$, checking whether $\mathcal{Q}_1$ is a module in $\mathcal{Q}$ for $\mathbf{S}$ is an undecidable problem. Given this negative result, we propose sufficient conditions for $\mathcal{Q}_1$ to be a module in $\mathcal{Q}$—that is, if $\mathcal{Q}_1$ satisfies our conditions then we can guarantee that it is a module for $\mathbf{S}$ in $\mathcal{Q}$, but not vice-versa. The conditions we present here are based on the notion of *syntactic locality*, first introduced in [3], and can be checked

Ontology of medical research projects $\mathcal{P}$:	
P1	Genetic_Disorder_Project $\equiv$ Project $\sqcap$ $\exists$has_Focus.Genetic_Disorder
P2	Cystic_Fibrosis_EUProject $\equiv$ EUProject $\sqcap$ $\exists$has_Focus.Cystic_Fibrosis
P3	EUProject $\sqsubseteq$ Project
Ontology of medical terms $\mathcal{Q}$:	
M1	Cystic_Fibrosis $\equiv$ Fibrosis $\sqcap$ $\exists$located_In.Pancreas $\sqcap$ $\exists$has_Origin.Genetic_Origin
M2	Genetic_Fibrosis $\equiv$ Fibrosis $\sqcap$ $\exists$has_Origin.Genetic_Origin
M3	Fibrosis $\sqcap$ $\exists$located_In.Pancreas $\sqsubseteq$ Genetic_Fibrosis
M4	Genetic_Fibrosis $\sqsubseteq$ Genetic_Disorder
M5	DEFBI_Gene $\sqsubseteq$ Immuno_Protein_Gene $\sqcap$ $\exists$associated_With.Cystic_Fibrosis

Table 1: Reusing medical terminology in an ontology on research projects

in polynomial time for OWL DL ontologies. We propose an algorithm for computing locality-based modules and present our experimental results on a set of real-world ontologies of varying size and complexity. We show that the modules we obtain are much smaller than the ones computed using existing techniques.

We refer the interested reader to the extended version of this paper [2] for a more detailed presentation of our results.

2 Modules for Knowledge Reuse

For exposition, suppose that an ontology engineer wants to build an ontology about research projects. The ontology defines different types of projects according to the research topics they focus on. Suppose that the ontology engineer defines two concepts Genetic_Disorder_Project and Cystic_Fibrosis_EUProject in his ontology $\mathcal{P}$. The first one describes projects about genetic disorders; the second one describes European projects about cystic fibrosis, as given by the axioms P1 and P2 in Table 1.

The ontology engineer is an expert on research projects: he knows, for example, that an EUProject is a Project (axiom P3). He might be unfamiliar, however, with most of the topics the projects cover and, in particular, with the terms Cystic_Fibrosis and Genetic_Disorder mentioned in P1 and P2. In this case, he decides to reuse the knowledge about these subjects from a well-established and widely-used medical ontology.

The most straightforward way to reuse these concepts is to import the foreign medical ontology. This may be, however, a large ontology, which deals with other matters in which the ontology engineer is not interested, such as genes, anatomy, surgical techniques, etc. Ideally, one would like to extract a (hopefully small) fragment of the medical ontology—a *module*—that describes in detail the reused concepts. Intuitively, importing the module $\mathcal{Q}_1$ into $\mathcal{P}$ instead of the full ontology $\mathcal{Q}$ should have the same impact on the modeling of the ontology $\mathcal{P}$ as importing $\mathcal{Q}$.

Continuing with the example, suppose that Cystic_Fibrosis and Genetic_Disorder are described in an ontology $\mathcal{Q}$ containing axioms M1-M5 in Table 1. If we include

in the module $\mathcal{Q}_1$ just the axioms that mention Cystic_Fibrosis or Genetic_Disorder, namely M1, M4 and M5, we lose the following dependency:

$$\mathsf{Cystic\_Fibrosis} \sqsubseteq \mathsf{Genetic\_Disorder} \tag{1}$$

The dependencies $\mathsf{Cystic\_Fibrosis} \sqsubseteq \mathsf{Genetic\_Fibrosis} \sqsubseteq \mathsf{Genetic\_Disorder}$ follow from axioms M1-M5, but not from M1, M4, M5, since the dependency $\mathsf{Cystic\_Fibrosis} \sqsubseteq \mathsf{Genetic\_Fibrosis}$ does not hold after removing M2 and M3 from $\mathcal{Q}$. The dependency (1), however, is crucial for our ontology $\mathcal{P}$ as it (together with axiom P3) implies the following axiom:

$$\mathsf{Cystic\_Fibrosis\_EUProject} \sqsubseteq \mathsf{Genetic\_Disorder\_Project} \tag{2}$$

This means that all the projects annotated with Cystic_Fibrosis_EUProject must be included in the answer for a query on Genetic_Disorder_Project. Importing a part of $\mathcal{Q}$ containing only axioms that mention the terms used in $\mathcal{P}$ instead of $\mathcal{Q}$ results in an underspecified ontology. We stress that the ontology engineer might be unaware of dependency (2), even though it concerns the concepts of his primary scope.

The example above suggests that the central requirement for a module $\mathcal{Q}_1 \subseteq \mathcal{Q}$ to be reused in our ontology $\mathcal{P}$ is that $\mathcal{P} \cup \mathcal{Q}_1$ should yield the *same* logical consequences in the vocabulary of $\mathcal{P}$ as $\mathcal{P} \cup \mathcal{Q}$ does. Furthermore, the fact that $\mathcal{Q}_1$ is a module in $\mathcal{Q}$ should be independent from the particular $\mathcal{P}$ under consideration—that is, if $\mathcal{Q}_1$ "behaves" in the same way as $\mathcal{Q}$ for a given ontology $\mathcal{P}$ but not for a different ontology $\mathcal{P}'$, then $\mathcal{Q}_1$ should not be a module in $\mathcal{Q}$. Based on this intuition, we can formalize the notion of a *module* as follows:

Definition 1 (Module). *Let $\mathcal{Q}_1 \subseteq \mathcal{Q}$ be two ontologies and $\mathbf{S}$ a signature. We say that $\mathcal{Q}_1$ is an* $\mathbf{S}$-module in $\mathcal{Q}$ *w.r.t. an ontology language* L*, if for every ontology $\mathcal{P}$ and axiom α expressed in* L *with $\mathsf{Sig}(\mathcal{P}) \cap \mathsf{Sig}(\mathcal{Q}) \subseteq \mathbf{S}$ and $\mathsf{Sig}(\alpha) \subseteq \mathsf{Sig}(\mathcal{P})$, we have that $\mathcal{P} \cup \mathcal{Q} \models \alpha$ if and only if $\mathcal{P} \cup \mathcal{Q}_1 \models \alpha$.*

Definition 1 implies that, for any $\mathcal{P}$, the axioms in $\mathcal{Q}$ excluded from the module $\mathcal{Q}_1$ do not cause new consequences in $\mathcal{P}$ and therefore can be disregarded. In Definition 1 the signature $\mathbf{S}$ acts as the *interface* signature between $\mathcal{P}$ and $\mathcal{Q}$ in the sense that it contains the symbols that $\mathcal{P}$ and may share with $\mathcal{Q}$. Unfortunately, it can be shown that, for the description logic $\mathcal{ALCO}$—the fragment of OWL DL that disallows transitive roles, role hierarchies, inverse roles, and cardinality restrictions—checking whether $\mathcal{Q}_1$ is a module in $\mathcal{Q}$ for $\mathbf{S}$ is an undecidable problem (see [1]).

Theorem 1. *Given a signature $\mathbf{S}$, an $\mathcal{ALC}$-ontology $\mathcal{Q}$ and an $\mathcal{ALCO}$ ontology $\mathcal{Q}_1 \subseteq \mathcal{Q}$, it is undecidable whether $\mathcal{Q}_1$ is an $\mathbf{S}$-module in $\mathcal{Q}$ w.r.t.* $\mathrm{L} = \mathcal{ALCO}$.

It is a consequence of this theorem that many relevant tasks, such as the extraction of minimal modules or the extraction of all the modules, cannot be algorithmically solved in OWL DL. Theorem 1, however, does *not* imply the impossibility of designing an algorithm for finding *some* (not necessarily minimal) modules. In particular, it is always possible to extract an $\mathbf{S}$-module in $\mathcal{Q}$ since one can simply return $\mathcal{Q}$ which is always an $\mathbf{S}$-module in $\mathcal{Q}$. From a practical point of view, our goal is to find a procedure for computing "reasonably small" modules. Obviously, by Theorem 1, the set of all modules and the set of all minimal modules in $\mathcal{Q}$ for a given signature $\mathbf{S}$ cannot be computed using our procedure.

3 Modules Based on Locality

Consider axiom M5 from Table 1. Suppose that we are given an interpretation $\mathcal{I}_{\mathbf{S}}$ for the vocabulary $\mathbf{S} = \{\mathsf{Cystic\_Fibrosis}\}$. Suppose that we define a new interpretation $\mathcal{I}$ by taking $\mathcal{I}_{\mathbf{S}}$ and interpreting the remaining symbols in M5, namely DEFB_Gene, Inmuno_Protein_Gene and associated_With as the empty set. It is not hard to see that $\mathcal{I}$ is a model of M5 regardless of the interpretation of Cystic_Fibrosis. For module extraction, this means that M5 is irrelevant to the meaning of Cystic_Fibrosis and hence this axiom need not be included in a module for this symbol.

This intuition can be used to define a particular kind of modules, which are based on the notion of *syntactic locality*, first introduced in [3].

Definition 2 (Syntactic Locality for $\mathcal{SHOIQ}$).
Let $\mathbf{S}$ *be a signature. The following grammar recursively defines two sets of concepts* $\mathcal{C}_{\mathbf{S}}^{\perp}$ *and* $\mathcal{C}_{\mathbf{S}}^{\top}$ *for a signature* $\mathbf{S}$*:*

$$\mathcal{C}_{\mathbf{S}}^{\perp} ::= A^{\perp} \mid (\neg C^{\top}) \mid (C \sqcap C^{\perp}) \mid (\exists R^{\perp}.C) \mid (\exists R.C^{\perp}) \mid (\geqslant n\, R^{\perp}.C) \mid (\geqslant n\, R.C^{\perp})\,.$$
$$\mathcal{C}_{\mathbf{S}}^{\top} ::= (\neg C^{\perp}) \mid (C_1^{\top} \sqcap C_2^{\top})\,.$$

where $A^{\perp} \notin \mathbf{S}$ *is a atomic concept,* R *is a role, and* C *is a concept,* $C^{\perp} \in \mathcal{C}_{\mathbf{S}}^{\perp}$*,* $C_{(i)}^{\top} \in \mathcal{C}_{\mathbf{S}}^{\top}$*,* $i = 1, 2$*, and* $R^{\perp} \notin \mathsf{Rol}(\mathbf{S})$ *is a role.*

An axiom α *is* syntactically local w.r.t. $\mathbf{S}$ *if it is of one of the following forms:* (1) $R^{\perp} \sqsubseteq R$*, or* (2) $\mathsf{Trans}(R^{\perp})$*, or* (3) $C^{\perp} \sqsubseteq C$ *or* (4) $C \sqsubseteq C^{\top}$*. A* $\mathcal{SHOIQ}$ *ontology* $\mathcal{O}$ *is syntactically local w.r.t.* $\mathbf{S}$ *if all its axioms are syntactically local w.r.t.* $\mathbf{S}$*.*

Intuitively, every concept in $\mathcal{C}_{\mathbf{S}}^{\perp}$ becomes equivalent to $\perp$ if we replace every symbol $A^{\perp}$ or $R^{\perp}$ not in $\mathbf{S}$ with the bottom concept $\perp$ and the empty role respectively, which are both interpreted as the empty set under every interpretation. Similarly, the concepts from $\mathcal{C}_{\mathbf{S}}^{\top}$ are equivalent to $\top$ under this replacement. Syntactically local axioms become tautologies after these replacements.

For example, the axiom M5 from Table 1 is local w.r.t. $\mathbf{S} = \{\mathsf{Fibrosis}, \mathsf{has\_Origin}\}$: if we replace the remaining symbols in this axiom with $\perp$, we obtain a tautology $\perp \equiv \perp$:

$$\overbrace{\mathsf{DEFBI\_Gene}}^{\perp} \equiv \overbrace{\mathsf{Inmuno\_Protein\_Gene}}^{\perp} \sqcap \exists \overbrace{\mathsf{associated\_With}}^{\text{empty role}}.\underline{\mathsf{Cystic\_Fibrosis}}$$

Given a signature $\mathbf{S}$ and an ontology $\mathcal{O}$, syntactic locality can be used as a sufficient condition for a fragment $\mathcal{O}_1 \subseteq \mathcal{O}$ to be an $\mathbf{S}$-module in $\mathcal{O}$, as given by the following proposition:

Proposition 1. *Let* $\mathcal{O}_1, \mathcal{O}_2$ *be* $\mathcal{SHOIQ}$ *ontologies and* $\mathbf{S}$ *a signature such that* $\mathcal{O}_2$ *is local w.r.t.* $\mathbf{S} \cup \mathsf{Sig}(\mathcal{O}_1)$*. Then* $\mathcal{O}_1$ *is an* $\mathbf{S}$*-module in* $\mathcal{O}_1 \cup \mathcal{O}_2$*.*

Based on Proposition 1, we can define a particular modules that can be identified using locality:

Definition 3 (Modules based on Locality Condition).
Given an ontology $\mathcal{Q}$ *and a signature* $\mathbf{S}$*, we say that* $\mathcal{Q}_1 \subseteq \mathcal{Q}$ is a locality-based $\mathbf{S}$-module in $\mathcal{Q}$ *if* $\mathcal{Q} \setminus \mathcal{Q}_1$ *is local w.r.t* $\mathbf{S} \cup \mathsf{Sig}(\mathcal{Q}_1)$.

Algorithm 1 extract_module($\mathcal{Q}, \mathbf{S}$)

Input:
$\mathcal{Q}$: ontology
$\mathbf{S}$: signature
Output:
$\mathcal{Q}_1$: a locality-based $\mathbf{S}$-module in $\mathcal{Q}$

```
1: Q1 ← ∅   Q2 ← Q
2: while not empty(Q2) do
3:     α ← select_axiom(Q2)
4:     if locality_test( α, S ∪ Sig(Q1) ) then
5:         Q2 ← Q2 \ {α}                          ▷ α is processed
6:     else
7:         Q1 ← Q1 ∪ {α}                          ▷ move α into Q1
8:         Q2 ← Q \ Q1                            ▷ reset Q2 to the complement of Q1
9:     end if
10: end while
11: return Q1
```

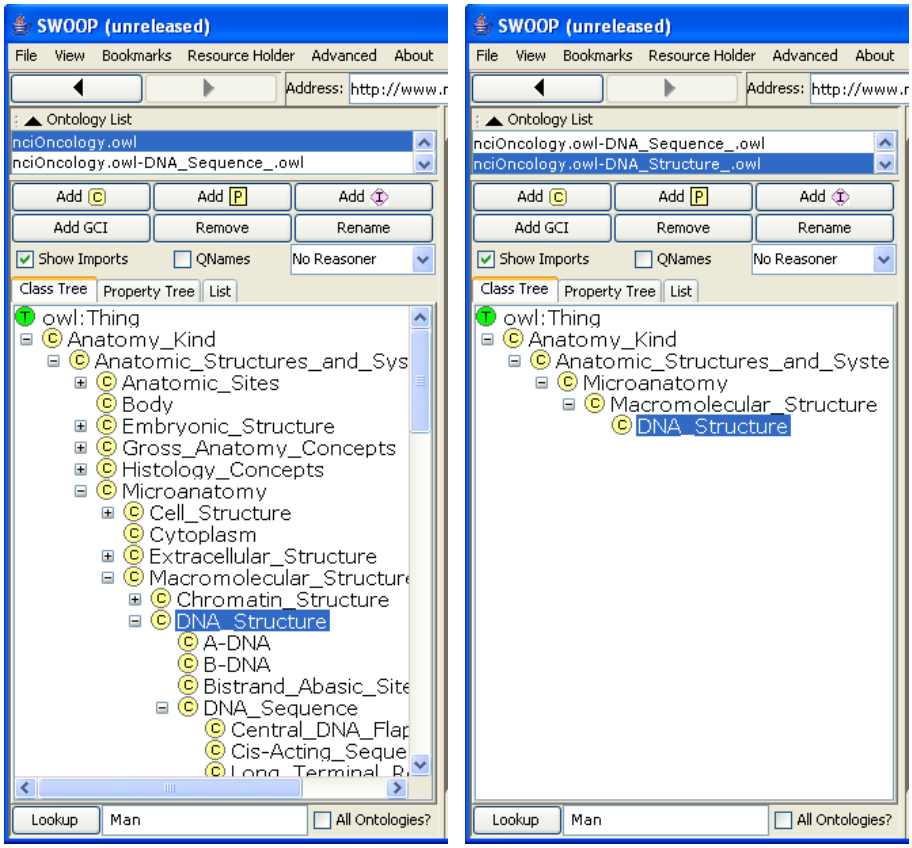
For example, we have seen that axiom M5 is local w.r.t. $\mathsf{Sig}(\mathcal{Q}_1)$ with $\mathcal{Q}_1 = \{\mathrm{M1}, \ldots, \mathrm{M4}\}$. Hence, according to Definition 3, $\mathcal{Q}_1$ is a locality-based $\mathbf{S}$-module in $\mathcal{Q}$ for every $\mathbf{S} \subseteq \mathsf{Sig}(\mathcal{Q}_1)$.

In order to construct a locality-based $\mathbf{S}$-module in an ontology $\mathcal{Q}$, it suffices to partition the ontology $\mathcal{Q}$ as $\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$ such that $\mathcal{Q}_2$ is local w.r.t. $\mathbf{S} \cup \mathsf{Sig}(\mathcal{Q}_1)$. Algorithm 1 outlines a simple procedure which performs this task. Assuming there is an effective locality test locality_test$(\alpha, \mathbf{S})$, which returns true only for axioms α that are local w.r.t. $\mathbf{S}$, the algorithm first initializes the partition to the trivial one: $\mathcal{Q}_1 = \emptyset$ and $\mathcal{Q}_2 = \mathcal{Q}$, and then repeatedly moves to $\mathcal{Q}_1$ those axioms from $\mathcal{Q}_2$ that are not local w.r.t. $\mathbf{S} \cup \mathsf{Sig}(\mathcal{Q}_1)$ until no such axioms are left in $\mathcal{Q}_2$.

In Table 2 we provide a trace of Algorithm 1 for the input $(\mathcal{Q}, \mathbf{S})$, where $\mathcal{Q}$ consists of the axioms M1-M5 from Table 1 and $\mathbf{S} = \{\mathsf{Cystic\_Fibrosis}, \mathsf{Genetic\_Disorder}\}$. Each row in the table corresponds to an iteration of the while loop in Algorithm 1. The

	$\mathcal{Q}_1$	$\mathcal{Q}_2$	New elements in $\mathbf{S} \cup \mathsf{Sig}(\mathcal{Q}_1)$	α	local?
1	–	$\mathrm{M1} - \mathrm{M5}$	Cystic_Fibrosis, Genetic_Disorder	M1	No
2	M1	$\mathrm{M2} - \mathrm{M5}$	Fibrosis, located_In, Pancreas, has_Origin, Genetic_Origin	M2	No
3	$\mathrm{M1}, \mathrm{M2}$	$\mathrm{M3} - \mathrm{M5}$	Genetic_Fibrosis	M3	No
4	$\mathrm{M1} - \mathrm{M3}$	$\mathrm{M4}, \mathrm{M5}$	–	M4	No
5	$\mathrm{M1} - \mathrm{M4}$	M5	–	M5	Yes
6	$\mathrm{M1} - \mathrm{M4}$	–	–	–	

Table 2: A trace of Algorithm 1 for the input $\mathcal{Q} = \{\mathrm{M1}, \ldots, \mathrm{M5}\}$ and $\mathbf{S} = \{\mathsf{Cystic\_Fibrosis}, \mathsf{Genetic\_Disorder}\}$



(a) Concept DNA_Structure in NCI

(b) Syntactic locality-based module for the concept DNA_Structure in NCI

Fig. 1: The Module Extraction Functionality in Swoop

last column of the table provides the result of the locality test in line 4. It can be shown [2] that Algorithm 1 computes, for each $\mathcal{Q}, \mathbf{S}$, a locality-based $\mathbf{S}$-module in $\mathcal{Q}$.

Note that there is an implicit non-determinism in Algorithm 1, namely, in line 3 in which an axiom from $\mathcal{Q}_2$ is selected. It might well be the case that several choices for α are possible at this moment. For example, in Table 2 at step 2 we might have selected axiom M3 instead of M2: It is possible to show (see [1] for detail) that the output of Algorithm 1 is uniquely determined by its input $\mathcal{Q}$ and $\mathbf{S}$.

In many Semantic Web applications, reuse often boils down to the following high-level task: given an atomic concept in the ontology that we want to borrow, provide the axioms in the ontology that are relevant to its meaning. Consequently, it is interesting to determine the scope of a locality-based module $\mathcal{Q}^{loc}_{\{A\}}$ for A in $\mathcal{Q}$.

Proposition 2. *Let $\mathcal{Q}$ be ontology, A and B atomic concepts, and $\mathcal{Q}^{loc}_{\{A\}}$ a locality-based module in $\mathcal{Q}$ for $\mathbf{S} = \{A\}$. Then, $\mathcal{Q} \models (A \sqsubseteq B)$ iff $\mathcal{Q}^{loc}_{\{A\}} \models (A \sqsubseteq B)$.*

Proposition 2 (see [1] for a proof) implies that any locality-based module for a single atomic concept A provides a complete representation of all the super-classes of A.

As an illustration, consider in Figure 1 the locality-based module for the atomic concept DNA_Structure in the NCI ontology, as obtained in SWOOP [5]. The user interface of SWOOP allows for the selection of an input signature and the retrieval of the corresponding module. As given in Proposition 2, the locality-based module $\mathcal{O}^{loc}_{\{A\}}$ for every atomic concept $A \in \mathsf{Sig}(\mathcal{O})$ contains all the necessary axioms for, at least, all the (entailed) super-concepts of A in $\mathcal{O}$. Thus $\mathcal{O}^{loc}_{\{A\}}$ can be seen as the "upper ontology" for A. In fact, Figure 1 shows that the locality-based module for DNA_Structure contains only the concepts in the "path" from DNA_Structure to the top level concept Anatomy_Kind. This suggests that the knowledge in NCI about the particular concept DNA_Structure is very shallow in the sense that NCI only "knows" that a DNA_Structure is a macromolecular structure, which, in the end, is an anatomical concept.

4 Related Work

The problem of extracting modular fragments of ontologies has recently been addressed in [9], [7] and [8]. In [9], the authors propose an algorithm for partitioning the concepts in an ontology. The goal is to facilitate the visualization of and navigation through the ontology. The algorithm uses a set of heuristics for measuring the degree of dependency between the concepts in the ontology and outputs a graphical representation of these dependencies. The algorithm is intended as a visualization technique, and does not establish a correspondence between the nodes of the graph and sets of axioms in the ontology. The algorithms in [7] and [8] use structural traversal techniques to extract modules from ontologies for a given signature. None of these approaches provides a characterization of the logical properties of the extracted modules, nor do they establish a notion of correctness of the modularization. In fact, in general, they do not produce modules according to Definition 1.

In [4], the authors propose a definition of a module and an algorithm for extracting modules based on that definition. The notion of a module in an ontology $\mathcal{Q}$ for a signature $\mathbf{S}$ also provides logical guarantees concerning the preservation of the meaning of the reused symbols. The logical requirements in [4] lead, in many cases, to modules which are larger than one may wish. The reason is that, for every atomic concept $A \in \mathbf{S}$, the module $\mathcal{Q}_1$ for A in $\mathcal{Q}$ must be a module for all its sub-concepts and super-concepts.

Our notion module is closely related to the notion of a conservative extension, which has been recently investigated in the context of ontologies [6]; we refer the reader to [2] for further details about this relationship. Finally, in our previous work [3] we have used locality as a sufficient condition for safe integration of ontologies.

5 Evaluation

Given an input ontology and an input signature, locality-based modules are not the only possible modules we can obtain. It remains to be shown that the locality-based modules obtained in realistic ontologies are *small enough* to be useful in practice.

Ontology	♯ Atomic	A1: Prompt-Factor [7]		A2: Mod. in [4]		A3: Loc.-based mod.	
	Concepts	Max.(%)	Avg.(%)	Max.(%)	Avg.(%)	Max.(%)	Avg.(%)
NCI	27772	87.6	75.84	55	30.8	0.8	0.08
SNOMED	255318	100	100	100	100	0.5	0.05
GO	22357	1	0.1	1	0.1	0.4	0.05
SUMO	869	100	100	100	100	2	0.09
GALEN-Small	2749	100	100	100	100	10	1.7
GALEN-Full	24089	100	100	100	100	29.8	3.5
SWEET	1816	96.4	88.7	83.3	51.5	1.9	0.1
DOLCE-Lite	499	100	100	100	100	37.3	24.6

Table 3: Comparison of Different Modularization Algorithms

For evaluation and comparison, we have implemented the following algorithms using Manchester's OWL API:[1]

A1: The PROMPT-FACTOR algorithm, as described in [7];
A2: The algorithm for extracting modules described in [4];
A3: Our algorithm for extracting modules (Algorithm 1), based on syntactic locality.

As a test suite, we have collected a set of well-known ontologies available on the Web, which can be divided into two groups:

Simple. In this group, we have included the National Cancer Institute (NCI) Ontology,[2] the SUMO Upper Ontology,[3] the Gene Ontology (GO),[4] and the SNOMED Ontology[5]. These ontologies use a simple ontology language and are of a simple structure; in particular, they contain only definitions.

Complex. This group contains the well-known GALEN ontology (GALEN-Full),[6] the DOLCE upper ontology (DOLCE-Lite),[7] and NASA's Semantic Web for Earth and Environmental Terminology (SWEET)[8]. These ontologies are complex since they use many constructors from OWL DL and/or include a significant number of general inclusion axioms and cyclic dependencies. In the case of GALEN, we have also considered a version GALEN-Small that has commonly been used as a benchmark for OWL reasoners. This ontology is almost 10 times smaller than the original GALEN-Full ontology, but it is similar in structure.

[1] http://sourceforge.net/projects/owlapi
[2] http://www.mindswap.org/2003/CancerOntology/nciOncology.owl
[3] http://ontology.teknowledge.com/
[4] http://www.geneontology.org
[5] http://www.snomed.org
[6] http://www.openclinical.org/prj_galen.html
[7] http://www.loa-cnr.it/DOLCE.html
[8] http://sweet.jpl.nasa.gov/ontology/

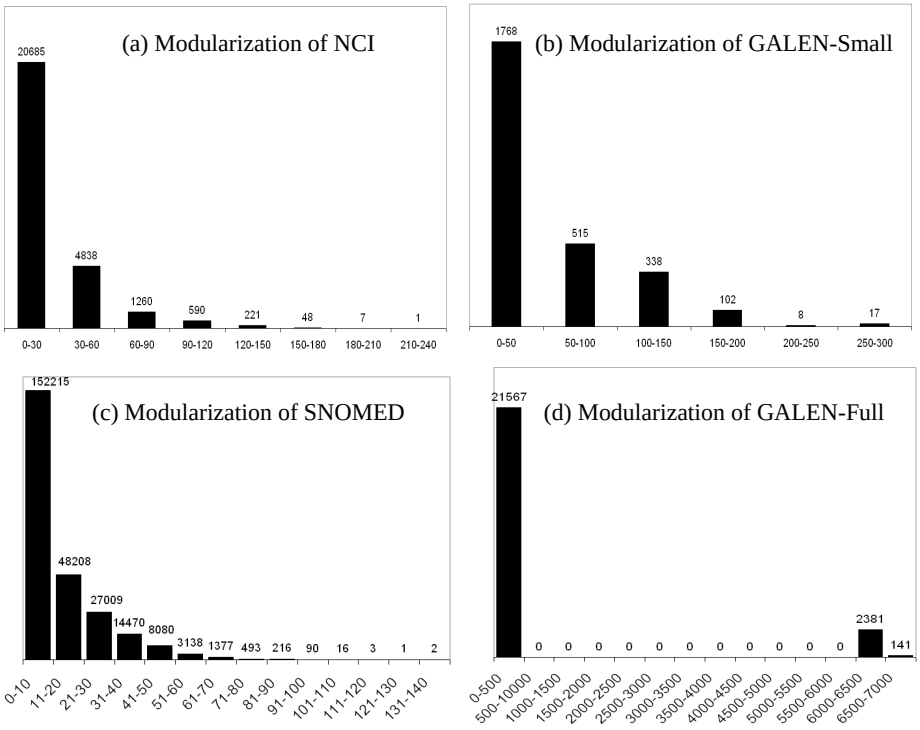


Fig. 2: Distribution for the sizes of syntactic locality-based modules for atomic concepts: the X-Axis gives the number of concepts in the modules and the Y-Axis the number of modules for each size range.

For each of these ontologies, and for each atomic concept in their signature, we have extracted the corresponding modules using algorithms A1-A3 and measured their size. We use modules for single atomic concepts to get an idea of the typical size of locality-based modules compared to the size of the whole ontology.

The results we have obtained are summarized in Table 3. The table provides the size of the largest module and the average size of the modules obtained using each of these algorithms. In the table, we can clearly see that locality-based modules are significantly smaller than the ones obtained using the other methods; in particular, in the case of SUMO, DOLCE, GALEN and SNOMED, the algorithms A1 and A2 retrieve the whole ontology as the module for each atomic concept. In contrast, the modules we obtain using our algorithm are significantly smaller than the size of the input ontology. In fact, our modules are not only smaller, but are also strict subsets of the respective modules computed using A1 and A2.

For NCI, SNOMED, GO and SUMO, we have obtained very small locality-based modules. This can be explained by the fact that these ontologies, even if large, are simple in structure and logical expressivity. For example, in SNOMED, the largest locality-

based module obtained is approximately 1/10000 of the size of the ontology, and the average size of the modules is 1/10 of the size of the largest module. In fact, most of the modules we have obtained for these ontologies contain less than 40 atomic concepts.

For GALEN, SWEET and DOLCE, the locality-based modules are larger. Indeed, the largest module in GALEN-Small is 1/10 of the size of the ontology, as opposed to 1/10000 in the case of SNOMED. For DOLCE, the modules are even bigger—1/3 of the size of the ontology—which indicates that, either the dependencies between the different concepts in the ontology are very strong and complicated and thus actual minimal modules are big themselves, or that no small locality-based modules exist. The SWEET ontology is an exception: even though the ontology uses most of the constructors available in OWL, the ontology is heavily underspecified, which yields small modules.

In Figure 2, we have presented a more detailed analysis of the modules for each concept name in NCI, SNOMED, GALEN-Small and GALEN-Full. Here, the X-axis represents the size ranges of the obtained modules and the Y-axis the number of modules whose size is within the given range. The plots thus give an idea of the distribution for the sizes of the modules.

For SNOMED, NCI and GALEN-Small, we can observe that the size of the modules follows a smooth distribution. In contrast, for GALEN-Full, we have obtained a many small modules and many big ones, but no medium-sized modules in-between. This abrupt distribution indicates the presence of a big cycle of dependencies in the ontology, which involves all the concepts with large modules.

References

1. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Extracting modules from ontologies: Theory and practice. Technical report, University of Manchester, 2007. Available from: http://www.cs.man.ac.uk/~bcg/Publications.html.
2. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount: Extracting modules from ontologies. In *Proc. of WWW-2007*, 2007.
3. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. A logical framework for modularity of ontologies. In *Proc. IJCAI-2007*, pages 298–304, 2007.
4. B. Cuenca Grau, B. Parsia, E. Sirin, and A. Kalyanpur. Modularity and Web Ontologies. In *Proc. KR-2006*, pages 198–209, 2006.
5. A. Kalyanpur, B. Parsia, E.Sirin, B. Cuenca Grau, and J. Hendler. SWOOP: A web editing browser. *Elsevier's Journal Of Web Semantics*, 4(2):144–153, 2006.
6. C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. In *Proc. of IJCAI-2007*, pages 453–459, 2007.
7. N. Noy and M. Musen. The PROMPT suite: Interactive tools for ontology mapping and merging. *Int. Journal of Human-Computer Studies*, 6(59), 2003.
8. J. Seidenberg and A. Rector. Web ontology segmentation: Analysis, classification and use. In *Proc. WWW-2006*, 2006.
9. H. Stuckenschmidt and M. Klein. Structure-based partitioning of large class hierarchies. In *Proc. ISWC-2004*, pages 289–303, 2004.