DeaLing with Ontologies using CODs

Chiara Del Vescovo[1]* and Rafael Peñaloza[2]**

[1] School of Computer Science, The University of Manchester, UK
delvescc@cs.man.ac.uk
[2] Theoretical Computer Science, TU Dresden, Germany
Center for Advancing Electronics Dresden
penaloza@tcs.inf.tu-dresden.de

Abstract. A major challenge in knowledge representation is to manage the access to knowledge: users should not be presented with knowledge that is irrelevant to their topic of interest, or have no right to access.
Two general strategies exist for providing access restrictions: (1) the ontology engineers describe the conditions that allow access to specific fragments of the ontology, or (2) fragments are automatically identified through their logical properties. The former is prone to miss logical connections between axioms, while the latter can fail to capture relevant knowledge that has no logical connection with the topic of interest.
We define the Context-Oriented Decomposition (COD) of an ontology as a technique that combines the benefits of both approaches: it allows authors to identify relevant fragments, while guaranteeing the strong semantic properties of the logic-based Atomic Decomposition.

1 Introduction

Description Logics (DLs) [1] have been successfully used to represent the knowledge of an application domain in a structured and usable manner. This knowledge is encoded through an *ontology*: a finite set of axioms that restrict the possible interpretations of the terms that are relevant for the domain. Different reasoning tasks can then be applied to extract knowledge that is implicitly encoded in this ontology.

In general, ontologies can be very large and contain a high level of detail of different topics within the same general domain. For example, the large-scale ontology Snomed CT [16] describes knowledge about anatomy, disorders, and processes, among many others. A user that is interested in only one topic or a general view might be overwhelmed by the additional information, irrelevant to her, available. It would thus be helpful to provide this user with just the knowledge that is relevant with her, without obscuring any relevant knowledge. A related, dual problem in using ontologies consists of restricting the access to sensitive knowledge: users have reasoning capabilities, thus removing an explicit axiom that describes sensitive knowledge is not sufficient to preserve secrecy, as it might be the case that the same knowledge can be inferred from the rest of

* Funded by the EPS Strategic Fund 2014 provided by the University of Manchester
** Partially supported by the DFG within the Cluster of Excellence 'cfAED'

the ontology. These two problems offer the same kind of challenge: to provide different users with the view to this knowledge that best suits their needs.

One strategy for tacking these issues is to decompose the ontology into meaningful fragments that can be provided to or hidden from users, depending on their interests and access rights. Current approaches for decomposing an ontology can be divided in two main categories: manual author-defined decompositions like [3,2], where the strong human component makes the fragments obtained liable to miss key information, or to include irrelevant one; and automated logic-based approaches, where the decomposition obtained satisfies logical properties, but it is likely to be very different from the one that authors have in mind. The main drawbacks of these two approaches are clear. The manual approach requires an expert to explicitly describe the full decomposition in detail, which is infeasible, specially for large or complex ontologies. The automated approach, on the other hand, must rely on semantic properties of the ontology without the intervention of an expert. Such an automated procedure cannot, e.g., combine two axioms with two disparate signatures into one common topic.

In this paper we present a new approach to ontology partitioning, called *context-oriented (atomic) decomposition* (*COD*), that combines the benefits of both kinds of approaches, while minimizing their drawbacks: it allows authors to identify fragments of an ontology that may be provided to users under some conditions; and it is based on a well-defined ontology partitioning, namely the *atomic decomposition*, that can be efficiently computed and guarantees the fragments obtained to satisfy strong logical properties.

2 Preliminaries

We assume familiarity with Description Logics (DLs) [1], and discuss the central notions of locality-based modularity [4] and Atomic Decompositions (ADs) [6]. For the rest of this paper, $\mathcal{L}$ denotes an arbitrary but fixed DL language, e.g. $\mathcal{ALC}$, and $\mathcal{O}$, $\mathcal{M}$, or $\mathcal{R}$, stand for an $\mathcal{L}$-ontology, i.e., a finite set of $\mathcal{L}$-axioms. For an axiom α or ontology $\mathcal{O}$, we denote as $\widetilde{\alpha}$ and $\widetilde{\mathcal{O}}$ the signature of α and $\mathcal{O}$, respectively; i.e., $\widetilde{\mathcal{O}}$ is the set of concept-, role-, and individual-names that appear in $\mathcal{O}$, and similarly for $\widetilde{\alpha}$. A *seed signature* is any user-selected or application-driven set Σ of terms occurring in $\widetilde{\mathcal{O}}$. A *logical module* $\mathcal{M}$ for a seed signature Σ in $\mathcal{O}$ is a subset of $\mathcal{O}$ such that, for all axioms α with $\widetilde{\alpha} \subseteq \Sigma$, $\mathcal{M} \models \alpha$ iff $\mathcal{O} \models \alpha$; i.e., $\mathcal{M}$ preserves all the entailments of $\mathcal{O}$ over the seed signature Σ. This property is usually known as *coverage*.

In general, we call *module* any subset $\mathcal{M}$ of an ontology $\mathcal{O}$ such that there exists a signature $\Sigma \subseteq \widetilde{\mathcal{O}}$ for which $\mathcal{M}$ is a logical module in $\mathcal{O}$. This notion of module is based upon the well-known notion in logics of *deductive Conservative Extensions* (dCEs), first discussed for DLs in [7]. Unfortunately, deciding whether a set of axioms is a module is, in general, hard or even impossible for expressive DLs [12]. In particular, it is hard to identify *minimal* such modules: due to the monotonicity of DLs, once a module $\mathcal{M}$ for Σ in $\mathcal{O}$ is found, then all its supersets in $\mathcal{O}$ are also modules for Σ in $\mathcal{O}$. Clearly, all the axioms in a minimal module play some role in preserving the entailments over Σ.

Some (not necessarily minimal) modules satisfy two additional properties: (1) a module $\mathcal{M}$ is *depleting* if $\mathcal{O} \setminus \mathcal{M}$ entails only tautologies over the signature Σ; and (2) $\mathcal{M}$ is *self-contained* if it is a module in $\mathcal{O}$ for the signature $\Sigma \cup \widetilde{\mathcal{M}}$. Intuitively, depleting modules encapsulate the knowledge of $\mathcal{O}$ about Σ, and no information about Σ is disclosed if the remainder $\mathcal{O} \setminus \mathcal{M}$ is published; and self-containment guarantees that all the terms in $\widetilde{\mathcal{M}}$ have the same status w.r.t. entailments, so that all the terms in $\widetilde{\mathcal{M}}$ are as much constrained in $\mathcal{M}$ as they are in $\mathcal{O}$. Modules satisfying these two properties also satisfy *uniqueness*: for each ontology $\mathcal{O}$, a depleting and self-contained module for a signature Σ is uniquely determined, under some mild conditions on the language used that are always satisfied by DL ontologies [13]. These properties, originally introduced in [13] are important for knowledge reuse, as discussed in [15].

The hardness results for minimal modules transfer to the computation of minimal self-contained and depleting modules. To identify feasible module extraction algorithms, one line of research in module extraction has been to restrict the expressivity of the DL to regain decidability; this approach has led to the identification of a polynomial algorithm to extract modules from *DL-Lite* ontologies using QBF solvers [13], and to the MEX system that allows for the extraction of modules from terminological $\mathcal{ELI}$ ontologies [11].

A different approach, aiming to allow module extraction from ontologies as expressive as $\mathcal{SROIQ}$, consists of identifying conditions under which computing a module is cheap, and the resulting modules are "not too large"; i.e., do not contain too many superfluous axioms. A prominent approach in this direction exploits the notion of *syntactic locality* [4], or *locality* for short. Locality-based modules (LBMs) come in many flavours, but for the sake of this paper we will focus on the three main notions named $\bot$, $\top$, and $\top\bot^*$.

Let $\diamond \in \{\top, \bot\}$, and let α' denote the formula obtained by replacing with the concept $\diamond$ all the terms in α that are not in Σ. A *syntactic check for $\diamond$-locality* can be performed by testing whether α' matches one of the syntactical patterns that guarantee that α is a tautology, that can be found in [4]. In contrast to determining whether a formula is a tautology, which is as hard as reasoning, the syntactic approximation provided by the patterns table makes the locality check purely syntactic, and hence the check can be performed in linear time. As an example, consider the axiom $\alpha = \mathtt{C} \sqsubseteq \mathtt{D}$. Then, α is $\bot$-local w.r.t. $\Sigma_1 = \{\mathtt{D}\}$ since $\alpha' = \bot \sqsubseteq \mathtt{D}$ is clearly a tautology for any (possibly complex) concept $\mathtt{D}$. Similarly, α is $\top$-local w.r.t. $\Sigma_2 = \{\mathtt{C}\}$ since $\alpha' = \mathtt{C} \sqsubseteq \top$ is a tautology for any concept $\mathtt{C}$. The following proposition lays down the foundation for using locality to identify modules in ontologies.

Proposition 1 ([4]). *If $\mathcal{M} \subseteq \mathcal{O}$ is such that all the axioms in $\mathcal{O} \setminus \mathcal{M}$ are $\diamond$-local w.r.t. $\Sigma \cup \widetilde{\mathcal{M}}$, then $\mathcal{M}$ is a depleting module for Σ in $\mathcal{O}$.*

The extraction of a module can be thus performed as follows: (1) define $\mathcal{M}$ as the set consisting of all the axioms in $\mathcal{O}$ that are non $\diamond$-local w.r.t. Σ; (2) enlarge Σ with $\widetilde{\mathcal{M}}$; (3) repeat steps (1) and (2) until a fixpoint is reached. By Proposition 1, the set $\mathcal{M}$ obtained is a depleting module. Moreover, the enlargement of Σ by

$\widetilde{\mathcal{M}}$ guarantees that $\mathcal{M}$ is also self-contained. The resulting module is called the $\diamond$-*module* of Σ in $\mathcal{O}$, and is denoted by $\diamond$-mod$(\Sigma, \mathcal{O})$.

Due to the coverage property, the extraction of the $\bot$-module for a signature Σ from the $\top$-module for Σ in $\mathcal{O}$ is still a module for Σ in $\mathcal{O}$. The converse is also true; that is, the $\top$-module for Σ in the $\bot$-module for Σ in $\mathcal{O}$ is still a module for Σ in $\mathcal{O}$. Hence, by iteratively nesting $\bot$- and $\top$-extraction until a fixpoint is reached one can get rid of more axioms that are not needed to preserve the entailments of $\mathcal{O}$ over Σ. It can be shown that the fixpoint does not depend on which notion is applied first. This uniquely determined new notion of module is called the $\top\bot^*$-*module* of Σ in $\mathcal{O}$, and is denoted by $\top\bot^*$-mod$(\Sigma, \mathcal{O})$.

Roughly speaking, a $\top$-module for Σ in $\mathcal{O}$ gives a view "from above" to the knowledge about Σ: it contains all the named sub-concepts of any concept name in Σ; dually, a $\bot$-module for Σ in $\mathcal{O}$ gives a view "from below" since it contains all the named super-concepts of any concept name in Σ; a $\top\bot^*$-module, instead, is a subset of both the corresponding $\top$- and $\bot$-modules, containing all the axioms needed to entail that two concepts in Σ are one a sub-concept of the other, but not necessarily all their sub- or super-concepts. The results in this paper can be easily extended to all notions of self-contained and depleting modules, but for simplicity we restrict our attention to LBMs. For a deeper discussion on LBMs, see [4].

OWL [10] and its successor OWL 2 are syntactic variants of DLs. For OWL ontologies, module extraction systems have been implemented and are available, both as part of the ontology editor Protégé,[3] and online.[4]

The Atomic Decomposition of an Ontology. For the purposes of this paper, it is relevant to investigate on the logical relationships between modules. This task, though, cannot be performed by pairwise comparisons of modules, as there are, in principle and in practice, exponentially many modules in the number of axioms of an ontology $\mathcal{O}$ [14].

We can restrict our attention to the *building blocks* of modules, as defined in [6]: a $\diamond$-*atom* is a maximal set of axioms $\mathfrak{a} \subseteq \mathcal{O}$ such that, for each $\diamond$-module $\mathcal{M}$, either $\mathfrak{a} \subseteq \mathcal{M}$, or $\mathfrak{a} \cap \mathcal{M} = \emptyset$ (for $\diamond \in \{\bot, \top, \top\bot^*\}$). Atoms are thus fragments of an ontology that show a tight logical interrelation as they never split across two or more modules. Notably, atoms are disjoint, and cover the whole ontology. As a consequence, the set $\mathcal{A}$ of all the atoms of an ontology $\mathcal{O}$ is a partition of $\mathcal{O}$, thus there are at most as many atoms as axioms in $\mathcal{O}$.

For each atom $\mathfrak{a}$, there is a unique smallest module $\mathcal{G}_{\mathfrak{a}}$ that contains $\mathfrak{a}$. Such a module is called *genuine*. Notably, the set $\mathfrak{G}$ of the genuine modules of $\mathcal{O}$ forms a base of the set $\mathfrak{F}$ of all the modules of $\mathcal{O}$. In particular, every module of $\mathcal{O}$ can be decomposed in a unique way as the union of one or more genuine modules. Moreover, $\mathfrak{G}$ induces a partial order relation $\succ$ over the set $\mathcal{A}$ which is defined as follows: we say that an atom $\mathfrak{a}$ *depends on* another, distinct atom $\mathfrak{b}$, and write $\mathfrak{a} \succ \mathfrak{b}$, iff for every genuine module $\mathcal{G} \in \mathfrak{G}$ such that $\mathfrak{a} \subseteq \mathcal{G}$, we have that $\mathfrak{b} \subseteq \mathcal{G}$. The partially ordered set $(\mathcal{A}, \succ)$ is called the *Atomic Decomposition* (AD)

[3] `http://www.co-ode.org/downloads/protege-x`

[4] `http://owlapi.sourceforge.net/`

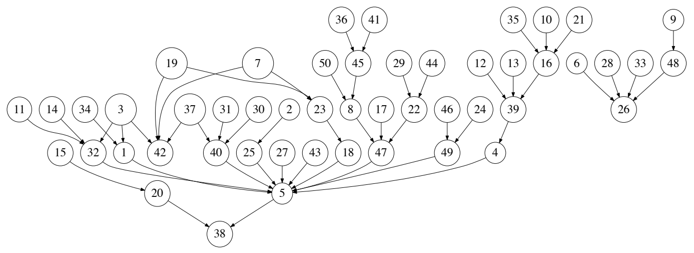

Fig. 1. The $\perp$-AD of the CARO ontology

of $\mathcal{O}$, and its well-foundedness derives from the depletion and self-containment properties of modules. ADs are usually represented by their Hasse diagrams as the one shown in Figure 1. Each genuine module $\mathcal{G}_{\mathfrak{a}}$ can be found by identifying the corresponding atom $\mathfrak{a}$ and by taking the union of atoms obtained by chasing the dependence relation $\succ$.

Example 2. The Common Anatomy Reference Ontology (CARO) is a simple ontology, available in the ontology repository BioPortal,[5] that, in the words of its developers, aims at "facilitating interoperability between existing ontologies for different species, and at providing a template for building new anatomy ontologies" [9]. CARO consists of 54 axioms written in $\mathcal{EL}$ with transitive roles.

Figure 1 depicts the $\perp$-AD of CARO, which reflects the hierarchical structure of the ontology. As it can be seen, CARO decomposes into 50 different $\perp$-atoms. 46 of these atoms contain only one axiom, and the remaining atoms ($\mathfrak{a}_3, \mathfrak{a}_7, \mathfrak{a}_{19}$, and $\mathfrak{a}_{37}$ in the figure) contain 2 axioms each. Looking at the graph, we can identify the different genuine modules. For instance, $\mathfrak{a}_{20}$ defines the genuine module $\mathcal{G}_{20} = \mathfrak{a}_{20} \cup \mathfrak{a}_{38}$. The atom $\mathfrak{a}_{39}$ contains only $\texttt{epithelium} \sqsubseteq \texttt{portion\_of\_tissue}$. Three atoms depend on $\mathfrak{a}_{39}$:

$$\begin{aligned}
&\mathfrak{a}_{12} = \{\texttt{atypical\_epithelium} \sqsubseteq \texttt{epithelium}\},\\
&\mathfrak{a}_{13} = \{\texttt{multilaminar\_epithelium} \sqsubseteq \texttt{epithelium}\}, \text{ and}\\
&\mathfrak{a}_{16} = \{\texttt{unilateral\_epithelium} \sqsubseteq \texttt{epithelium}\}.
\end{aligned}$$

Although $\mathfrak{a}_{31}$ contains the axiom $\texttt{epithelial\_cell} \sqsubseteq \texttt{cell}$, no logical dependence among $\mathfrak{a}_{31}$ and $\mathfrak{a}_{39}$ is shown since the ontology does not describe any logical relationship between an `epithelial_cell` and the `epithelium`.

3 Contexts and Rhemes

Modules are widely used for ontology reuse as they provide a powerful tool to identify a part of an ontology preserving the meaning of a signature of interest. For other tasks, though, the user must understand how the module $\mathcal{M}$ relates to the rest of the ontology. In the following, we describe two such tasks, and discuss the limitations of using purely logical modules for them.

[5] `http://bioportal.bioontology.org/`

Knowledge Completion (KC). An ontology is an explicit, formal representation that accounts for the ontological commitment of a particular conceptualization of the knowledge about a domain [8]. As such, it can only approximate the intended models; moreover, even if the difference between the intensional and extensional models could be eliminated, for many purposes this would mean superfluous work, as a partial representation may suffice for a given application. In other words, some relations between the objects of the domain can be underspecified in the logical formalisation, as it occurs in the CARO ontology (see Example 2). However, the users may still want their fragments to preserve not only the logical aspects, but also the ontological commitment of their sub-domain of interest, which purely logical modules are not designed to preserve. In this case, domain knowledge is required to include the relevant terms for the sub-domain of interest in the seed signature of the module.

Access Control (AC). As discussed in Section 2, the signature of a module $\mathcal{M}$ is usually larger than the seed signature Σ, as the reasons for an entailment over Σ to hold can arise from axioms where no term in Σ is mentioned. Since users are capable of reasoning, they can then derive logical consequences over the signature $\widetilde{\mathcal{M}}$. Thus, the knowledge engineer must ensure that users can only access those modules that do not implicitly encode any sensitive information, whose access should be denied. This task is in general infeasible given the exponential number of possible modules. Restricting the inspection only to genuine modules can still be impractical since their number is usually close to the number of axioms of the ontology [5]. Most importantly, even distinct genuine modules can be dealing with related topics as shown in Example 2, and users are likely to have the right to access a combination of them.

Technically, both KC and AC can be carried out with the help of contexts, and more precisely, modules that preserve the knowledge from a context.

Definition 3. *Let $\mathcal{O}$ be an ontology, $\mathfrak{L}$ a finite set of* labels *and* $\mathsf{lab} : \mathcal{O} \to 2^{\mathfrak{L}}$ *a function that maps every axiom to a set of labels. A* context *is a finite subset $\mathcal{C} = \{C_1, \ldots, C_\kappa\} \subseteq \mathfrak{L}$. Every context $\mathcal{C}$ defines the subontology*

$$\mathcal{R}^{\mathcal{C}} := \{\alpha \in \mathcal{O} \mid \mathsf{lab}(\alpha) \cap \mathcal{C} \neq \emptyset\}.$$

This set $\mathcal{R}^{\mathcal{C}}$ is called the rheme *of the context $\mathcal{C}$.*[6]

Intuitively, the set $\mathcal{C}$ expresses the conditions under which the ontology $\mathcal{R}^{\mathcal{C}}$ is accessible to a user. Every user will be associated also to a set of labels from $\mathfrak{L}$. If she has at least one label from $\mathcal{C}$, then she can access the whole rheme $\mathcal{R}^{\mathcal{C}}$.

If there is no ambiguity regarding the context $\mathcal{C}$, or specifying it is irrelevant in the discussion, we will usually drop the symbol $\mathcal{C}$ and write $\mathcal{R}$. In some cases, we may deal with sets of contexts $\mathcal{C}^1, \ldots, \mathcal{C}^\kappa$; to ease the notation, we will then write $\mathcal{R}^i$ to denote the rheme of the context $\mathcal{C}^i$, for $i \in \{1, \ldots, \kappa\}$.

Definition 4. *Let $\mathcal{O}$ be an ontology, $\diamond$ a notion of module, and $\mathcal{C}$ a context. The* contextual $\diamond$-module for $\mathcal{C}$ in $\mathcal{O}$ *is the set* $\diamond\text{-mod}(\mathcal{C}, \mathcal{O}) = \diamond\text{-mod}(\widetilde{\mathcal{R}^{\mathcal{C}}}, \mathcal{O}) \cup \mathcal{R}^{\mathcal{C}}$. *The signature of this set, $\Sigma^{\mathcal{C}}$, is called the* contextual signature *of $\mathcal{C}$.*

[6] The term rheme is borrowed from linguistics; it refers to "what is said about a theme", in contrast to the theme itself, which may be seen as the title for the rheme.

Algorithm 1 Computation of the contextual $\diamond$-module for $\mathcal{C}$ in $\mathcal{O}$

```
    Input:  An ontology $\mathcal{O}$; a notion $\diamond$ of module; a context $\mathcal{C}$.
    Output: The contextual $\diamond$-module $\mathcal{M}^{\mathcal{C}}_{\diamond} = \diamond\text{-mod}(\mathcal{C}, \mathcal{O})$ for $\mathcal{C}$.
1: $\mathcal{M}^{\mathcal{C}} \leftarrow \emptyset$
2: repeat
3:   $\Sigma prev \leftarrow \widetilde{\mathcal{R}^{\mathcal{C}}} \cup \widetilde{\mathcal{M}^{\mathcal{C}}}$
4:   for each $\alpha \in \mathcal{O}$ do
5:     if (the $\diamond$-locality check of $\alpha$ against $\widetilde{\mathcal{R}^{\mathcal{C}}} \cup \widetilde{\mathcal{M}^{\mathcal{C}}}$ is negative) or ($\alpha \in \mathcal{R}^{\mathcal{C}}$) then
6:       $\mathcal{M}^{\mathcal{C}} \leftarrow \mathcal{M}^{\mathcal{C}} \cup \{\alpha\}$
7: until $\widetilde{\mathcal{R}^{\mathcal{C}}} \cup \widetilde{\mathcal{M}^{\mathcal{C}}} = \Sigma prev$
8: return $\mathcal{M}^{\mathcal{C}}$
```

If the ontology $\mathcal{O}$ is clear, we will use the notation $\mathcal{M}^{\mathcal{C}}_{\diamond}$. If, instead, we have a set of contexts $\mathcal{C}_1, \ldots, \mathcal{C}_\kappa$, then we denote the corresponding contextual signatures by $\Sigma^1, \ldots, \Sigma^\kappa$. As the name suggests, a contextual module is also a module.

Proposition 5. *The contextual $\diamond$-module for a context $\mathcal{C}$ in an ontology $\mathcal{O}$ is a self-contained and depleting logical module in $\mathcal{O}$ for the signature $\widetilde{\mathcal{R}^{\mathcal{C}}}$.*

Proof. By definition, $\diamond$-mod$(\widetilde{\mathcal{R}^{\mathcal{C}}}, \mathcal{O})$ is a module for $\widetilde{\mathcal{R}^{\mathcal{C}}}$. Then, the result follows by monotonicity of DLs since the contextual module $\mathcal{M}^{\mathcal{C}}_{\diamond}$ is a superset of $\diamond$-mod$(\widetilde{\mathcal{R}^{\mathcal{C}}}, \mathcal{O})$, and hence since $\widetilde{\mathcal{R}^{\mathcal{C}}}$ is contained in the signature of the module, it is also a module for $\widetilde{\mathcal{R}^{\mathcal{C}}}$ with the same properties as $\diamond$-mod$(\widetilde{\mathcal{R}^{\mathcal{C}}}, \mathcal{O})$. □

Notice that in general $\mathcal{M}^{\mathcal{C}}_{\diamond} \neq \diamond$-mod$(\widetilde{\mathcal{R}^{\mathcal{C}}}, \mathcal{O})$: in principle the rheme $\mathcal{R}^{\mathcal{C}}$ could contain tautologies, like OWL *declarations*, which have no logical influence on the represented knowledge, but can be helpful for simplifying the work of the reasoners. Since tautologies are logically unnecessary for preserving entailments over $\widetilde{\mathcal{R}^{\mathcal{C}}}$, minimal modules over this signature will not include any of them.[7] Tautologies, though, can be of interest for users, for example if they are annotated with useful information. For this reason, it is crucial to leave at the discretion of the ontology engineer the choice of including them in rhemes or not.

Contextual modules can be computed by making use of Algorithm 1, which we prove to compute in finite time the contextual module $\mathcal{M}^{\mathcal{C}}_{\diamond}$ for a context $\mathcal{C}$.

Proposition 6. *Let $\mathcal{O}$ be an ontology, $\diamond$ a notion of module, and $\mathcal{C}$ a context. Algorithm 1 computes the contextual $\diamond$-module for the context $\mathcal{C}$.*

Proof. Termination follows from the finiteness of the ontology: the **repeat** loop on lines 2-7 terminates when no more axioms can be added to the set $\mathcal{M}^{\mathcal{C}}_{\diamond}$, and this eventually happens since there is a finite number of axioms in $\mathcal{O}$. Lines 5 and 6 ensure that the output $\mathcal{M}^{\mathcal{C}}_{\diamond}$ of the algorithm contains $\diamond$-mod$(\widetilde{\mathcal{R}^{\mathcal{C}}}, \mathcal{O}) \cup \mathcal{R}^{\mathcal{C}}$. We need only to prove that the converse inclusion holds.

Suppose that there is some axiom $\alpha \in \mathcal{M}^{\mathcal{C}}_{\diamond} \setminus (\diamond$-mod$(\widetilde{\mathcal{R}^{\mathcal{C}}}, \mathcal{O}) \cup \mathcal{R}^{\mathcal{C}})$. W.l.o.g., assume α to be $\diamond$-local against the signature of $\diamond$-mod$(\widetilde{\mathcal{R}^{\mathcal{C}}}, \mathcal{O})$, but not $\diamond$-local

[7] In syntactic approximations of modules, like e.g., syntactic locality, some tautologies can still appear since syntactic checks may fail to identify them as irrelevant.

against the signature of $\diamond$-mod$(\widetilde{\mathcal{R}^{\mathcal{C}}}, \mathcal{O}) \cup \mathcal{R}^{\mathcal{C}}$. Notice that the signature of the second term $\mathcal{R}^{\mathcal{C}}$ equals (thus, it is contained in) the seed signature of the first term $\diamond$-mod$(\widetilde{\mathcal{R}^{\mathcal{C}}}, \mathcal{O})$. Thus, the set of all the axioms that are non $\diamond$-local against the signature of $\diamond$-mod$(\widetilde{\mathcal{R}^{\mathcal{C}}}, \mathcal{O}) \cup \mathcal{R}^{\mathcal{C}}$ is already in $\diamond$-mod$(\widetilde{\mathcal{R}^{\mathcal{C}}}, \mathcal{O})$. □

Computing a contextual module requires a polynomial number of locality checks in the number n of axiom of an ontology $\mathcal{O}$. Since we focus only on the syntactic notions which can be performed in polynomial time on n, it is clear that the whole extraction is overall polynomial in n.

We now take a closer look at the notion of context and its properties. From Definition 3, an axiom can belong to as many contexts as the number ℓ of possible labels, which could, in principle, be arbitrarily large. On the other hand, ontologies are always finite by definition. Let n be the cardinality of the ontology. If $\ell > 2^n$, then some contexts have the same content; that is, there exists two different contexts $\mathcal{C}^1 = \{C_1^1, \dots, C_r^1\}$ and $\mathcal{C}^2 = \{C_1^2, \dots, C_s^2\}$ such that $\mathcal{R}^1 = \mathcal{R}^2$. In other words, we have two syntactically different contexts sharing the same rheme. Since we cannot differentiate among them, we can equivalently define a new context $\mathcal{C} := \mathcal{C}^1 \cup \mathcal{C}^2$. It is a simple consequence of Definition 3 and the equivalence of $\mathcal{R}^1$ and $\mathcal{R}^2$ that the rheme for $\mathcal{C}$ is $\mathcal{R} = \mathcal{R}^1 = \mathcal{R}^2$.

From what we just discussed, it is clear that the number of contexts can be bounded by 2^n. Such method leads to a potentially exponential number of different contexts, and hence of contextual modules. In this case, it is hard to check that all of them keep the promise of either completing knowledge or restricting the access to sensitive knowledge. However, ontology engineers can focus their attention to a much smaller sample of contextual modules; there is a base, linear in n, of contextual modules, called *genuine*, such that every contextual module is the union of a suitable selection of genuine contextual modules. Attentive readers may have noticed that we are reusing the term "genuine" as in the context of ADs. This choice is not due to chance; it is rather a way to emphasize the similarities between the two notions. We formalize and analyse this next.

Definition 7. *Let $\mathcal{O}$ be an ontology, and $\mathfrak{C}$ the set of all the contexts over $\mathcal{O}$. For each axiom $\alpha \in \mathcal{O}$, define the* rheme of α *as $\mathcal{R}_\alpha^{\mathcal{C}} = \{\beta \in \mathcal{O} \mid \mathsf{lab}(\alpha) \subseteq \mathsf{lab}(\beta)\}$ if $\mathsf{lab}(\alpha)$ is non-empty, or $\{\alpha\}$ otherwise. A* genuine contextual module *is a contextual module $\mathcal{G}_\alpha^{\mathcal{C}} := \mathrm{mod}(\widetilde{\mathcal{R}_\alpha^{\mathcal{C}}}, \mathcal{O})$ for some $\alpha \in \mathcal{O}$.*

For each two axioms α, β, the condition $\mathsf{lab}(\alpha) \subseteq \mathsf{lab}(\beta)$ means that whoever can access α can also access β. In particular, $\mathcal{G}_\alpha^{\mathcal{C}}$ is the smallest contextual module containing α.

Theorem 8 *Let $\mathcal{O}$ be an ontology. The set of the genuine contextual modules $\mathfrak{G}^{\mathcal{C}} = \{\mathcal{G}_\alpha^{\mathcal{C}} \mid \alpha \in \mathcal{O}\}$ is a base for the set of all the contextual modules in $\mathcal{O}$.*

Proof. We prove that every contextual module $\mathcal{M}^{\mathcal{C}}$ coincides with the union of all the modules $\mathcal{G}_\alpha^{\mathcal{C}}$ for $\alpha \in \mathcal{M}^{\mathcal{C}}$. The inclusion $\mathcal{M}^{\mathcal{C}} \subseteq \bigcup_{\alpha \in \mathcal{M}^{\mathcal{C}}} \mathcal{G}_\alpha^{\mathcal{C}}$ is trivial since every $\mathcal{R}_\alpha^{\mathcal{C}}$ contains at least α. The converse inclusion also holds since $\mathcal{M}^{\mathcal{C}}$ contains, for each $\alpha \in \mathcal{M}^{\mathcal{C}}$, the rheme $\mathcal{R}_\alpha^{\mathcal{C}}$ by construction. □

Since there are at most as many genuine contextual modules as axioms, we have that $\mathfrak{G}^{\mathcal{C}}$ is a linear (in $|\mathcal{O}|$) base of the set of all contextual modules of $\mathcal{O}$. As for ADs, the genuine contextual modules are those contextual modules that never split as the union of two or more contextual modules, which are incomparable by $\subseteq$. We want to find the building blocks of contextual modules.

Definition 9. *A* contextual atom *is a maximal subset of axioms $\mathfrak{a}^{\mathcal{C}} \subseteq \mathcal{O}$ such that, for each genuine contextual module $\mathcal{G}^{\mathcal{C}} \in \mathfrak{G}^{\mathcal{C}}$, either $\mathfrak{a}^{\mathcal{C}} \subseteq \mathcal{G}^{\mathcal{C}}$, or $\mathfrak{a}^{\mathcal{C}} \cap \mathcal{G}^{\mathcal{C}} = \emptyset$.*

Contextual atoms are pairwise disjoint by construction; as for modules, contextual modules are then a finite union of contextual atoms. Since contextual modules are also modules, atoms cannot split across two contextual modules. Similarly, contextual atoms do not split across two contextual modules by construction. Thus, the set $\mathcal{A}^{\mathcal{C}}$ of all the contextual atoms of $\mathcal{O}$ is a coarsening of the set $\mathcal{A}$ of all the atoms of $\mathcal{O}$. Moreover, the preservation of logical interrelations between the terms of a contextual atom requires that this atom is always provided to users as a whole. Thus, rhemes can be, ultimately, redefined to coincide with contextual atoms. Rhemes then contain axioms that are logically or contextually interrelated; that is, $\mathcal{R}^{\mathcal{C}}$ includes axioms that belong to a same topic according to the knowledge engineer.

Definition 10. *Let $\mathcal{O}$ be an ontology, $\mathfrak{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_\kappa\}$ a set of contexts with rhemes $\mathcal{R}_1^{\mathcal{C}}, \ldots, \mathcal{R}_\kappa^{\mathcal{C}}$, and $\mathcal{A}^{\mathcal{C}}$ the set of all the corresponding contextual atoms. A partial order relation is induced as follows: for each pair of distinct contextual atoms $\mathfrak{a}^{\mathcal{C}}, \mathfrak{b}^{\mathcal{C}}$, we say that $\mathfrak{a}^{\mathcal{C}} \succ \mathfrak{b}^{\mathcal{C}}$ iff, if a contextual module $\mathcal{G}^{\mathcal{C}}$ contains $\mathfrak{a}^{\mathcal{C}}$, then $\mathcal{G}^{\mathcal{C}}$ contains also $\mathfrak{b}^{\mathcal{C}}$. The pair $(\mathcal{A}^{\mathcal{C}}, \succ)$ is called the* Context-Oriented Decomposition *(COD) of $\mathcal{O}$.*

The COD of an ontology can be represented by a dependency graph of contextual atoms. From this graph, ontology engineers may want to identify the fragments of the ontology that are of interest for the needs of its users, and provide them with suitable contexts to access the corresponding subontology. Thus, the issue of identifying the right subontology for each user can be reduced to the definition of contexts in such a way that the corresponding COD reflects the knowledge engineer modelling of the domain.

We discuss a feasible methodology to generate a suitable set of contexts $\mathfrak{C}$ and the corresponding fragments of the ontology $\mathcal{O}$ from which users are assigned the right to access to the relevant part of $\mathcal{O}$. Rather than defining all the modules directly, the knowledge engineer may proceed in the following way. First, she can identify relevant but small rhemes $\mathcal{R}_i^{\mathcal{C}}$. Second, the COD $(\mathcal{A}^{\mathcal{C}}, \succ)$ is computed, and rhemes are redefined to coincide with contextual atoms. Finally, she defines suitable labels for each contextual atoms that will form the set of contexts.

4 Applications of CODs

In this section, we discuss the two applications, Knowledge Completion and Access Control, introduced before, and show how COD can be of help for practical use of ontologies.

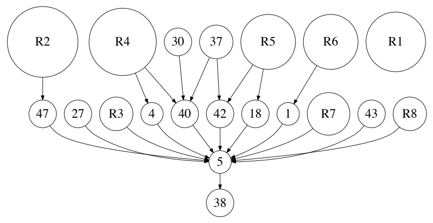

Fig. 2. ⊥-COD of the CARO ontology

COD for KC. An analysis of the ⊥-AD of the CARO ontology reveals how hard it can be to identify which fragment captures everything that an ontology says about a certain topic. The most emblematic example in CARO can be observed already in Example 2: even by chasing sub- and super-concepts, incoming or outgoing role-links to the concept `epithelium`, a user not familiar with CARO cannot find the portion of the ontology dealing with `epithelial_cells`. To reveal that these terms are related in the ontological commitment of the CARO ontology, we have grouped the 8 atoms $\mathfrak{a}_{39}, \mathfrak{a}_{12}, \mathfrak{a}_{13}, \mathfrak{a}_{16}, \mathfrak{a}_{35}, \mathfrak{a}_{10}$ and $\mathfrak{a}_{21}$ in rheme R4, and named the context C4 as `Epithelium`.

In general, two axioms dealing with the same topic may still belong to different atoms even when the ontology is supposedly well-modelled. Ideally, we might want that two contextual atoms are distinct only when they deal with different topics. To obtain this situation, we can adopt the strategy just described, and merge into a broader contextual atoms all those atoms that deal with the same topic according to the domain experts' knowledge.

In Figure 2 we show the ⊥-COD of CARO where the atoms are grouped according to the principle just discussed. In specific: rheme R1, that deals with `Immaterial anatomical entity`, consists of the atom $\mathfrak{a}_{26}$ plus all the atoms that depend on it; R2, dealing with `Sexual traits`, contains the atoms $\mathfrak{a}_{8}, \mathfrak{a}_{17}, \mathfrak{a}_{22}$ plus all those that depend on any of these; R3= $\mathfrak{a}_{25} \cup \mathfrak{a}_{2}$, and deals with `Acellular` anatomical objects; R5= $\mathfrak{a}_{23} \cup \mathfrak{a}_{7} \cup \mathfrak{a}_{19}$, and deals with `Neuron projection`; R6, dealing with `Compound organ`, consists of the atom $\mathfrak{a}_{32}$ plus all those that depend on it; R7= $\mathfrak{a}_{49} \cup \mathfrak{a}_{24} \cup \mathfrak{a}_{46}$, and deals with `Anatomic group`; finally, R8=$\mathfrak{a}_{20} \cup \mathfrak{a}_{15}$ deals with `Portion of substance`.

From the ⊥-COD of CARO, the knowledge engineer will provide the axioms in R4, $\mathfrak{a}_{4}, \mathfrak{a}_{40}, \mathfrak{a}_{5}$ and $\mathfrak{a}_{38}$ to those interested in getting the fragment of CARO that deals with `Epithelium`.

COD for AC. To exploit the use of COD for AC, the main idea is that users are assigned a set of genuine contextual signatures, which specify the logical symbols over which the user can query the ontology. The coverage properties of logical modules guarantees that reasoning over these modules yields the exact same answers as reasoning over the whole ontology, provided that they refer only to symbols in the genuine contextual signature. Thus, the knowledge engineer can verify that the user cannot derive a consequence, simply by comparing the signature of the consequence with the contextual signatures assigned to the user: if the former is contained in any of the latter, then (and only then) the user can derive the consequence.

Notice that, by definition, users are not assigned one module, but a set of them. This means that each user can derive the consequences of the union of all the contexts that she can access. For each of these contexts, it is guaranteed that all the consequences of the ontology that refer to its signature are obtained. However, we might want to restrict the access a specific piece of knowledge, but not to completely obstruct access to its signature. In this case, we can exploit the fact that the union of two modules is not, in general, a module. For example, suppose that we are trying to hide a consequence $\mathtt{C} \sqsubseteq \mathtt{D}$ from a user. To achieve this, it is not necessary to completely prevent the user from accessing none of the concept names $\mathtt{C}$ or $\mathtt{D}$. In fact, she can be assigned two signatures Σ_1 and Σ_2 such that $\mathtt{C} \in \Sigma_1$ and $\mathtt{D} \in \Sigma_2$, as long as $\{\mathtt{C}, \mathtt{D}\}$ is not contained in any of them. In this way, the user can still derive meaningful consequences concerning each of the concepts $\mathtt{C}$ and $\mathtt{D}$, while the subsumption $\mathtt{C} \sqsubseteq \mathtt{D}$ remains hidden.

To fully specify an AC scenario, the knowledge engineer needs only to identify which genuine contextual modules do not cover the signature of the piece of knowledge that some users cannot access. Users are then assigned only contextual signatures that do not contain the signature of any consequence that they are not allowed to see, or which are irrelevant for their intended use.

5 Conclusions

We have proposed a new approach for decomposing ontologies, capable of dealing with contexts defined by a user or knowledge engineer. This context-oriented decomposition is based on the notions of atomic decompositions, and preserves several of its important properties. Most notably, the COD can be efficiently computed and provides a compact representation of all genuine modules that preserve the given contexts. Intuitively, one can think of contexts as a manner of combining axioms that do not necessarily share logical properties into a single entity. This combination can be motivated by the presence of axioms with disparate signatures, but referring to a common theme.

As working examples, we showed how COD can be used to complete knowledge and to restrict access to some consequences of the ontology in a structured manner. Using COD, a knowledge engineer can easily define contexts that can be assigned to different users, guaranteeing access to all relevant knowledge, while hiding all unauthorized consequences from them.

Although we have focused our discussion on locality-based modules, and in particular on the three notions of $\top$-, $\bot$-, and $\top\bot^*$-modules, it should be noted that our approach is general and can be applied to any desired notion of module. Obviously, the choice of a different notion of module can affect the efficiency of the method; in particular, if modules are hard to compute, then the decomposition will necessarily be also hard to find.

As future work we plan to further study the logical and computational properties of COD and its applicability to other context-dependent reasoning problems. We are also in the process of developing a tool for computing these decompositions. This tool will allow us to evaluate the practical feasibility of our approach for solving different reasoning tasks.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Baader, F., Knechtel, M., Peñaloza, R.: Context-dependent views to axioms and consequences of semantic web ontologies. J. of Web Sem. 12–13, 22–40 (2012)
3. Baader, F., Knechtel, M., Peñaloza, R.: A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms. In: et al., A.B. (ed.) Proc. of ISWC-09. LNCS, vol. 5823, pp. 49–64 (2009)
4. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. of Artif. Intell. Research 31(1), 273–318 (2008)
5. Del Vescovo, C., Gessler, D., Klinov, P., Parsia, B., Sattler, U., Schneider, T., Winget, A.: Decomposition and modular structure of BioPortal ontologies. In: Proc. of ISWC-11. LNCS, vol. 7031, pp. 130–145 (2011)
6. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The modular structure of an ontology: Atomic decomposition. In: Proc. of IJCAI-11. pp. 2232–2237 (2011)
7. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in Description Logics. In: Proc. of KR-06. pp. 187–197. AAAI Press/The MIT Press (2006)
8. Guarino, N.: Formal ontology in information systems. In: Proc. of FOIS-98. pp. 3–15. IOS Press (1998)
9. Haendel, M.A., Neuhaus, F., Osumi-Sutherland, D., Mabee, P.M., Mejino, J.L.V.J., Mungall, C.J., Smith, B.: CARO - The Common Anatomy Reference Ontology. In: Anatomy Ontologies for Bioinformatics: Principles and Practice, pp. 327–350. No. 6 in Computational Biology, Springer-Verlag (2008)
10. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. J. of Web Sem. 1(1), 7–26 (2003)
11. Konev, B., Lutz, C., Walther, D., Wolter, F.: Semantic modularity and module extraction in description logics. In: Proc. of ECAI-08. pp. 55–59 (2008)
12. Konev, B., Lutz, C., Walther, D., Wolter, F.: Formal properties of modularization. In: Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.) Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization, LNCS, vol. 5445, pp. 25–66. Springer-Verlag (2009)
13. Kontchakov, R., Pulina, L., Sattler, U., Schneider, T., Selmer, P., Wolter, F., Zakharyaschev, M.: Minimal module extraction from DL-Lite ontologies using QBF solvers. In: Proc. of IJCAI-09. pp. 836–841 (2009)
14. Parsia, B., Schneider, T.: The modular structure of an ontology: an empirical study. In: Proc. of KR-10. pp. 584–586. AAAI Press/The MIT Press (2010)
15. Sattler, U., Schneider, T., Zakharyaschev, M.: Which kind of module should I extract? In: Proc. of DL 2009. `ceur-ws.org`, vol. 477 (2009)
16. Spackman, K.: Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. Journal of the American Medical Informatics Association (2000), fall Symposium Special Issue.