The Modular Structure of an Ontology: Atomic Decomposition

Chiara Del Vescovo[1] and **Bijan Parsia**[1] and **Uli Sattler**[1] and **Thomas Schneider**[2]
[1] The University of Manchester, Oxford Road, Manchester, M13 9PL, United Kingdom
[2] Universität Bremen, FB 03, Postfach 330 440, 28334 Bremen, Germany
{delvescc, bparsia, sattler}@cs.man.ac.uk tschneider@informatik.uni-bremen.de

Abstract

Extracting a subset of a given ontology that captures all the ontology's knowledge about a specified set of terms is a well-understood task. This task can be based, for instance, on locality-based modules. However, a single module does not allow us to understand neither topicality, connectedness, structure, or superfluous parts of an ontology, nor agreement between actual and intended modeling.

The strong logical properties of locality-based modules suggest that the family of all such modules of an ontology can support comprehension of the ontology as a whole. However, extracting that family is not feasible, since the number of locality-based modules of an ontology can be exponential w.r.t. its size.

In this paper we report on a new approach that enables us to efficiently extract a polynomial representation of the family of all locality-based modules of an ontology. We also describe the fundamental algorithm to pursue this task, and report on experiments carried out and results obtained.

1 Introduction

In software engineering, modularly structured systems are desirable, all other things being equal. Given a well-designed modular program, it is generally easier to process, modify, and analyze it and to reuse parts by exploiting the modular structure. As a result, support for modules (or components, classes, objects, packages, aspects) is a commonplace feature in programming languages.

Ontologies are computational artefacts akin to programs and, in notable examples, can get quite large and complex, which suggests that exploiting modularity might be fruitful. Research into modularity for ontologies has been an active area for ontology engineering. Recently, a lot of effort has gone into developing *logically sensible* modules, that is, modules which offer strong logical guarantees for intuitive modular properties [Cuenca Grau *et al.*, 2008]. One such guarantee is called *coverage* and means that the module captures all the ontology's knowledge about a given set of terms (signature)—a kind of dependency isolation or encapsulation. This guarantee is provided by modules based on conservative extensions, but also by efficiently computable approximations, such as locality-based modules.

The task of extracting one such module given a signature (GetOne) is well understood and starting to be deployed in standard ontology development environments, such as Protégé 4,[1] and online.[2] The extraction of locality-based modules has already been effectively used in the field for ontology reuse [Jimeno *et al.*, 2008] as well as a subservice for incremental reasoning [Cuenca Grau *et al.*, 2010]. Now GetOne requires the user to know in advance the right set of terms to input to the extractor: we call this a *seed* signature for the module and note that one module can have several such seed signatures. Since there are non-obvious relations between the final signature of a module and its seed signature, users are often unsure how to generate a proper request and confused by the results.

The task GetOne is an important and useful service but, by itself, it tells us nothing about the modular structure of the ontology as a whole. The modular structure is determined by the set of *all* modules and their inter-relations, or at least a suitable subset thereof. The task of a-posteriori determining the modular structure of an ontology is called GetStruct, and the task of extracting all modules is called GetAll. While GetOne is well-understood and often computationally cheap, GetAll and GetStruct have not been examined for module notions with strong logical guarantees, with a few preliminary exceptions [Cuenca Grau *et al.*, 2006; Del Vescovo *et al.*, 2010]. If ontology engineers had access to the overall modular structure of the ontology determined by GetStruct, they might be able to use it to guide their extraction choices and, supported by the experience described in [Cuenca Grau *et al.*, 2006], to understand its topicality, connectedness, structure, superfluous parts, or agreement between actual and intended modeling. For example, by inspecting the modular structure and observing un-connected parts that are intended to be connected, ontology designers could learn of weakly modeled parts of their ontology.

In the worst case, however, the number of all modules of an ontology is exponential in the number of terms or axioms in the ontology [Del Vescovo *et al.*, 2010]. More importantly,

[1] http://www.co-ode.org/downloads/protege-x
[2] http://owl.cs.manchester.ac.uk/modularity

even for very small ontologies, the number of all modules is far too large for them to be inspected by a user or even computed; e.g., Koala is an ontology with 42 axioms that has 3,660 modules, and GetAll fails even on many ontologies consisting of less than one hundred axioms.

In this paper, we report on new insights regarding the modular structure of ontologies which leads to a new, polynomial algorithm for GetStruct (provided that module extraction is polynomial) that generates a linear (in the size of the ontology), partially ordered set of modules and *atoms* which succinctly represent *all* (potentially exponentially many) modules of an ontology. We also report on some experiments carried out with an implementation of this algorithm. For full proofs, the reader is referred to [Del Vescovo *et al.*, 2011], and for more detail about the experiments and data can be found at `http://bit.ly/i4olY0`.

2 Preliminaries

We assume the reader to be familiar with Description Logics [Baader *et al.*, 2003], and only briefly sketch here some of the central notions around locality-based modularity. We use $\mathcal{L}$ for a Description Logic, e.g., $\mathcal{SHIQ}$, and $\mathcal{O}, \mathcal{M}$, etc., for a knowledge base, i.e., a finite set of axioms. Moreover, we use $\widetilde{\mathcal{O}}$ for the signature of $\mathcal{O}$, i.e., the set of concept, role, and individual names used in $\mathcal{O}$.

Conservative extensions (CEs) capture the above described encapsulation of knowledge. They are defined as follows.

Definition 2.1. *Let $\mathcal{L}$ be a DL, $\mathcal{M} \subseteq \mathcal{O}$ be $\mathcal{L}$-ontologies, and Σ be a signature.*

1. *$\mathcal{O}$ is a* deductive Σ-conservative extension (Σ-dCE) *of $\mathcal{M}$ w.r.t. $\mathcal{L}$ if for all axioms α over $\mathcal{L}$ with $\widetilde{\alpha} \subseteq \Sigma$, it holds that $\mathcal{M} \models \alpha$ if and only if $\mathcal{O} \models \alpha$.*
2. *$\mathcal{M}$ is a* dCE-based module for Σ *of $\mathcal{O}$ if $\mathcal{O}$ is a Σ-dCE of $\mathcal{M}$ w.r.t. $\mathcal{L}$.*

Unfortunately, CEs are hard or even impossible to decide for many DLs [Ghilardi *et al.*, 2006; Konev *et al.*, 2009; Sattler *et al.*, 2009]. Therefore, approximations have been devised. We focus on *syntactic locality* (here for short: locality). Locality-based modules can be efficiently computed and provide coverage; that is, they capture *all* the relevant entailments, but not necessarily *only* those [Cuenca Grau *et al.*, 2008; Jiménez-Ruiz *et al.*, 2008]. Although locality is defined for the DL $\mathcal{SHIQ}$, it is straightforward to extend it to $\mathcal{SHOIQ}$(D) (see [Cuenca Grau *et al.*, 2008; Jiménez-Ruiz *et al.*, 2008]), and a locality-based module extractor is implemented in the OWL API.[3]

It has been shown in [Cuenca Grau *et al.*, 2008] that $\mathcal{M} \subseteq \mathcal{O}$ and all axioms in $\mathcal{O} \setminus \mathcal{M}$ being local w.r.t. $\Sigma \cup \widetilde{M}$ is sufficient for $\mathcal{O}$ to be a Σ-dCE of $\mathcal{M}$. Various notions of locality are described in the literature, including so-called $\top$-, $\bot$-, and $\top\bot^*$-locality.

Given an ontology $\mathcal{O}$, a *seed signature* Σ and a module notion $x \in \{\top, \bot, \top\bot^*\}$, we denote the x-module of $\mathcal{O}$ w.r.t. Σ by x-mod$(\Sigma, \mathcal{O})$. Tractable, locality-based module extractors are described in [Cuenca Grau *et al.*, 2008]. If we do not specify x, we generally speak of a *locality-based* module.

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

The following properties of locality-based modules will be of interest for our modularization [Cuenca Grau *et al.*, 2008; Sattler *et al.*, 2009].

Proposition 2.2. *Let $\mathcal{O}$ be an ontology, Σ a signature and $x \in \{\top, \bot, \top\bot^*\}$. Then the following properties hold:*
(i) for any Σ', x-mod$(\Sigma, \mathcal{O}) \subseteq x$-mod$(\Sigma \cup \Sigma', \mathcal{O})$ (monotonicity)
(ii) for Σ' with $\Sigma \subseteq \Sigma' \subseteq Sigma \cup \widetilde{\mathcal{M}}$, x-mod$(\Sigma', \mathcal{O}) = x$-mod$(\Sigma, \mathcal{O})$ (self-containedness)
(iii) each axiom α entailed by $\mathcal{O} \setminus x$-mod$(\Sigma, \mathcal{O})$ and such that $\widetilde{\alpha} \subseteq \Sigma$ is a tautology (depletingness).

2.1 Fields of sets and atoms

We want to describe the relationships between an ontology $\mathcal{O}$ and a family $\mathfrak{F}(\mathcal{O})$ of subsets thereof by means of a well-understood structure. To this end, we introduce in what follows some notions of algebra.

Definition 2.3. *A* field of sets *is a pair (O, F), where O is a set and F is an algebra over O i.e., set of subsets of O that is closed under intersection, union and complement. Elements of O are called* points*, while those of F are called* complexes*.*

We will make use of a *partial order* $\leq$, i.e., a reflexive, transitive, and antisymmetric binary relation. Two elements a, b of a poset are called *comparable* if $a \leq b$ or $b \leq a$, otherwise they are *incomparable*.

Given a finite set O and a family F of subsets of O, we can build the *induced field of sets* $B(O, F)$ by closing the family under union, intersection and complement. Then $B(O, F)$ is obviously a field of sets and its elements are called *induced complexes*. Also, $B(O, F)$ inherits a partial order relation defined by the inclusion relation "$\subseteq$". Next, we define minimal elements in a slightly non-standard way.

Definition 2.4. *Given a poset $(O, \leq)$, an element $a \in O$ is called* minimal *if there exists no element b of $O \setminus a$ with $b \leq a$.*

The minimal elements of the $B(O, F) \setminus \emptyset$ with respect to the inclusion relation "$\subseteq$" are called atoms.[4]

For an element $a \in O$, the set $(a] := \{x \in O \mid x \leq a\}$ is called the principal ideal *of a.*

Every finite poset $(O, \leq)$ (and every lattice) can be depicted in a graph, called *Hasse diagrams*, where nodes are elements of O and edges connect two elements $a \leq b$ if there is no element c distinct from a and b such that $a \leq c \leq b$; for $a \leq b$, we will draw b in a position higher than a's.

3 Modules and atoms

In what follows, we are using the notion of $\top\bot^*$-locality from [Sattler *et al.*, 2009]. However, the approach we present can be applied to any notion of a module that is monotonic, self-contained, and depleting, and we know from [Kontchakov *et al.*, 2009] that robustness under replacement and depletingness implies self-containedness. These properties have a deep impact on the modules generated, as described in Proposition 3.1. See [Kontchakov *et al.*, 2009] for more details.

[4] Slightly abusing notation, we use $B(O, F)$ here for the set of complexes in $B(O, F)$.

Proposition 3.1. *Any notion of locality-based module that satisfies monotonicity, self-containedness, and depletingness is such that any given signature generates a unique module.*

We are going to define a correspondence among ontologies with relative families of modules and fields of sets as defined in Definition 2.3. Axioms correspond to points. Let then $\mathfrak{F}(\mathcal{O})$ denote the family of $\top\bot^*$-modules of $\mathcal{O}$ (or let $\mathfrak{F}_x(\mathcal{O})$ be such family for each corresponding notion x of module if not univocally specified). Then $\mathfrak{F}(\mathcal{O})$ is not, in general, closed under union, intersection and complement: given two modules, neither their union nor their intersection nor the complement of a module is, in general, a module; hence, only some complexes correspond to modules. Next, we introduce the (induced) field of modules, that is the field of sets over $\mathfrak{F}(\mathcal{O})$. This enables us to use properties of fields of sets also for ontologies.

Definition 3.2. *Given an ontology $\mathcal{O}$ and the family $\mathfrak{F}(\mathcal{O})$ of $\top\bot^*$-modules of $\mathcal{O}$, we define the* (induced) field of modules $\mathcal{B}(\mathfrak{F}(\mathcal{O}))$ *as the closure of the set $\mathfrak{F}(\mathcal{O})$ under union, intersection and complement.*

Definition 3.3. *We call*

- syntactic tautologies *the axioms that do not occurr in any module and hence belong to $\mathcal{O} \setminus \top\bot^*\text{-mod}(\widetilde{\mathcal{O}}, \mathcal{O})$;*
- global axioms *those occurring in each module, and in particular to $\top\bot^*\text{-mod}(\emptyset, \mathcal{O})$.*

Remark 3.4. *To make the presentation easier, we assume, without loss of generality,*[5] *that $\mathcal{O}$ contains no syntactic tautologies or global axioms.*

An (induced) field of modules is, by construction, a field of sets. It is partially ordered by $\subseteq$ and, due to the finiteness of $\mathcal{O}$, and can thus be represented via its Hasse diagram.

Next, we define *atoms* of our field of modules as building blocks of modules of an ontology; recall that these are the $\subseteq$-minimal complexes of $\mathcal{B}(\mathfrak{F}(\mathcal{O})) \setminus \{\emptyset\}$.

Definition 3.5. *The family of atoms from $\mathcal{B}(\mathfrak{F}(\mathcal{O}))$ is denoted by $\mathcal{A}(\mathfrak{F}(\mathcal{O}))$ and is called* atomic decomposition.

An atom is a set of axioms such that, for any module, it either contains all axioms in the atom or none of them. Moreover, every module is the union of atoms. Next, we show how atoms can provide a succinct representation of the family of modules. Before proceeding further, we summarize in Table 1 the four structures introduced so far and, for each, its elements, source, maximal size, and structure.

3.1 Atoms and their structure

The family $\mathcal{A}(\mathfrak{F}(\mathcal{O}))$ of atoms of an ontology, as in Definition 3.5, has many properties of interest for us.

Lemma 3.6. *The family $\mathcal{A}(\mathfrak{F}(\mathcal{O}))$ of atoms of an ontology $\mathcal{O}$ is a partition of $\mathcal{O}$, and thus $\#\mathcal{A}(\mathfrak{F}(\mathcal{O})) \leq \#\mathcal{O}$.*

Hence the atomic decomposition is *succinct*; we will see next whether its computation is tractable and whether it is indeed a representation of $\mathfrak{F}(\mathcal{O})$.

[5] We can always remove those unwanted axioms that occur in either all or no module, and consider them separately.

$\mathcal{O}$	$\mathfrak{F}(\mathcal{O})$	$\mathcal{B}(\mathfrak{F}(\mathcal{O}))$	$\mathcal{A}(\mathfrak{F}(\mathcal{O}))$
axioms α	modules $\mathcal{M}$	complexes $\mathcal{K}_{i,j}$	atoms $\mathfrak{a}, \mathfrak{b}, \ldots$
ontology engineers	module extractor	closure of $\mathfrak{F}(\mathcal{O})$	atoms of $\mathcal{B}(\mathfrak{F}(\mathcal{O}))$
baseline	exponential	exponential	linear
set	family of sets	complete lattice	poset

Table 1: Four ways for looking at ontology fragments

The following definition aims at defining a notion of "logical dependence" between axioms: the idea is that an axiom α depends on another axiom β if, whenever α occurs in a module $\mathcal{M}$ then β also belongs to $\mathcal{M}$. A slight extension of this argument allows us to generalize this idea because, by definition of atoms, whenever α occurs in a module, all axioms belonging to α's atom $\mathfrak{a}$ occur. Hence, we can formalize this idea by defining a relation between atoms.

Definition 3.7. ***(Relations between atoms)*** *Let $\mathfrak{a}$ and $\mathfrak{b}$ be two distinct atoms of an ontology $\mathcal{O}$. Then:*

- *$\mathfrak{a}$ is* dependent *on $\mathfrak{b}$ (written $\mathfrak{a} \succeq \mathfrak{b}$) if, for every module $\mathcal{M} \in \mathfrak{F}(\mathcal{O})$ such that $\mathfrak{a} \subseteq \mathcal{M}$, we have $\mathfrak{b} \subseteq \mathcal{M}$.*
- *$\mathfrak{a}$ and $\mathfrak{b}$ are* independent *if there exist two disjoint modules $\mathcal{M}_1, \mathcal{M}_2 \in \mathfrak{F}(\mathcal{O})$ such that $\mathfrak{a} \subseteq \mathcal{M}_1$ and $\mathfrak{b} \subseteq \mathcal{M}_2$.*
- *$\mathfrak{a}$ and $\mathfrak{b}$ are* weakly dependent *if they are neither independent nor dependent; in this case, there exists an atom $\mathfrak{c} \in \mathcal{A}(\mathfrak{F}(\mathcal{O}))$ which both $\mathfrak{a}$ and $\mathfrak{b}$ are dependent on.*

Proposition 3.8. *Definition 3.7 describes the all and only relations between atoms.*

The logical dependence between atoms can, in general, be incomplete: for example, let us consider the following (hypothetical) family of modules: $\mathfrak{F}(\mathcal{O}) = \{\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4\}$ where $\mathcal{M}_1 = \{\mathfrak{a}, \mathfrak{b}\}$, $\mathcal{M}_2 = \{\mathfrak{a}, \mathfrak{c}\}$, $\mathcal{M}_3 = \{\mathfrak{a}, \mathfrak{b}, \mathfrak{d}\}$ and $\mathcal{M}_4 = \{\mathfrak{a}, \mathfrak{c}, \mathfrak{d}\}$. Following Definition 3.7, the atoms $\mathfrak{b}$, $\mathfrak{c}$ and $\mathfrak{d}$ depend on $\mathfrak{a}$. However, we want our structure to reflect that $\mathfrak{b}$ and $\mathfrak{c}$ act as "intermediates" in the dependency of $\mathfrak{d}$ on $\mathfrak{a}$, i.e., that $\mathfrak{d}$ depends via "$\mathfrak{c}$ or $\mathfrak{b}$" on $\mathfrak{a}$. Since our definition does not capture disjunctions of occurrences of atoms, we call the pairs $(\mathfrak{d}, \mathfrak{b})$ and $(\mathfrak{d}, \mathfrak{c})$ *problematic*. Fortunately, problematic pairs of atoms do not exist in an atomic decomposition obtained via locality-based modules.

Lemma 3.9. *Since the $\top\bot^*$ notion of module is monotonic, self-contained, and depleting, there are no problematic pairs in the set $\mathcal{A}(\mathfrak{F}(\mathcal{O}))$ of atoms over $\mathcal{O}$.*

The key to proving Lemma 3.9 is the following remark:

Remark 3.10. *Let $\mathfrak{a} \in \mathcal{A}(\mathfrak{F}(\mathcal{O}))$ an atom induced over $\mathcal{O}$ by $\top\bot^*$-mod. Then, for every nonempty set of axioms $\{\alpha_1, \ldots, \alpha_k\} \subseteq \mathfrak{a}$ we have that $\top\bot^*\text{-mod}(\{\widetilde{\alpha}_1, \ldots, \widetilde{\alpha}_k\}, \mathcal{O})$ is the smallest module containing $\mathfrak{a}$.*

Proof. Let $\alpha \in \mathfrak{a}$ be an axiom, and let us consider the module $\mathcal{M}_\alpha := \top\bot^*\text{-mod}(\widetilde{\alpha}, \mathcal{O})$. We recall $\top\bot^*$-mod is self-contained and monotonic. Then:

(1) $\mathcal{M}_\alpha$ is not empty since it contains α (recall Remark 3.4).

(2) $\mathcal{M}_\alpha \supseteq \mathfrak{a}$, by the definition of atoms.

(3) $\mathcal{M}_\alpha$ is the unique and thus smallest module for the seed signature $\widetilde{\alpha}$.

(4) by monotonicity, enlarging the seed signature $\widetilde{\alpha}$ results in a superset of $\mathcal{M}_\alpha$.

(5) by self-containedness and monotonicity, any module $\mathcal{M}'$ that contains α needs to contain also $\mathcal{M}_\alpha$: $\mathcal{M}' = \top\bot^*\text{-mod}(\widetilde{\mathcal{M}'}, \mathcal{O}) = \top\bot^*\text{-mod}(\widetilde{\mathcal{M}'} \cup \widetilde{\alpha}, \mathcal{O}) \supseteq \top\bot^*\text{-mod}(\widetilde{\alpha}, \mathcal{O})$.

(6) because of (2), we have that $\mathcal{M}_\alpha \supseteq \top\bot^*\text{-mod}(\widetilde{S}, \mathcal{O})$ for every non empty set of axioms $S = \{\alpha_1, \ldots, \alpha_k\} \subseteq \mathfrak{a}$; in particular, this holds if $S = \{\alpha_i\}$ for any $\alpha_i \in \mathfrak{a}$.

(7) the inverted inclusion $\top\bot^*\text{-mod}(\widetilde{\alpha_i}, \mathcal{O}) \supseteq \mathcal{M}_\alpha$ also holds by the arbitrarity of choice of α in $\mathfrak{a}$. □

Corollary 3.11. *Given an atom $\mathfrak{a}$, for any axiom $\alpha \in \mathfrak{a}$ we have that $\mathcal{M}_\alpha = \top\bot^*\text{-mod}(\widetilde{\mathfrak{a}}, \mathcal{O})$. Moreover, $\mathfrak{a}$ is dependent on all atoms belonging to $\mathcal{M}_\alpha \setminus \mathfrak{a}$.*

Lemma 3.9 has interesting consequences on the dependency relation on atoms.

Proposition 3.12. *The binary relation " $\succeq$" is a partial order over the set $\mathcal{A}(\mathfrak{F}(\mathcal{O}))$ of atoms of an ontology $\mathcal{O}$.*

Definition 3.7 and Proposition 3.12 allow us to draw a Hasse diagram also for the atomic decomposition $\mathcal{A}(\mathfrak{F}(\mathcal{O}))$, where independent atoms belong to different chains, see Figure 1 for the Hasse diagramm of Koala. As an atom can be dependent on more that one atom; hence, we will have some nodes with more than one outgoing edge.

3.2 Atoms as a module base

As an immediate consequence of our observations so far, a module is a disjoint finite union of atoms. Conversely, it is not true that arbitrary unions of atoms are modules. However, we can compute the modules from $\mathcal{A}(\mathfrak{F}(\mathcal{O}))$, and thus the latter is indeed a succinct *representation* of all modules.

Definition 3.13. *The* principal ideal of an atom $\mathfrak{a}$ *is the set* $(\mathfrak{a}] = \{\alpha \in \mathfrak{b} \mid \mathfrak{b} \prec \mathfrak{a}\} \subseteq \mathcal{O}$.

Proposition 3.14. *For every atom $\mathfrak{a}$, $(\mathfrak{a}]$ is a module.*

To get modules from $\mathcal{A}(\mathfrak{F}(\mathcal{O}))$, we need, for each atom $\mathfrak{a}$, to store the $\subseteq$-minimal seed signatures that lead to $(\mathfrak{a}]$: we say that an atom $\mathfrak{a}$ is *relevant* for a module $\top\bot^*\text{-mod}(\Sigma, \mathcal{O})$ if there is a seed signature Σ' for $(\mathfrak{a}]$ with $\Sigma' \subseteq \Sigma$.

Proposition 3.15. *Let $\mathfrak{a}_1, \ldots \mathfrak{a}_k$, $k \in \mathbb{N}$, be all atoms that are relevant for Σ. Then the module $\top\bot^*\text{-mod}(\Sigma, \mathcal{O})$ is the union of principal ideals of these atoms:*

$$\top\bot^*\text{-mod}(\Sigma, \mathcal{O}) = \bigcup_{i=1}^{k} (\mathfrak{a}_i].$$

4 Computing the atomic decomposition

As we have seen, the atomic decomposition is a succinct representation of all modules of an ontology: its linearly many atoms represent all its worst case exponentially many modules. Next, we will show how we can compute the atomic decomposition in polynomial time, i.e., without computing all modules, provided that module extraction is polynomial (which is the case, e.g., for syntactic locality-based modules). Our approach relies on modules "generated" by a single axioms, which can be used to generate all others.

Definition 4.1. *A module $\mathcal{M}$ is called:*

1) compact *if there exists an atom $\mathfrak{a}$ in the atomic decomposition $\mathcal{A}(\mathfrak{F}(\mathcal{O}))$ such that $\mathcal{M} = (\mathfrak{a}]$.*

2) α-module *if there is an axiom $\alpha \in \mathcal{O}$ such that $\mathcal{M} = \top\bot^*\text{-mod}(\widetilde{\alpha}, \mathcal{O})$.*

3) fake *if there exist two incomparable (w.r.t. set inclusion) modules $\mathcal{M}_1 \neq \mathcal{M}_2$ with $\mathcal{M}_1 \cup \mathcal{M}_2 = \mathcal{M}$; a module is called* genuine *if it is not fake.*

Please note that our notion of genuinity is different from the one in [Parsia & Schneider, 2010], where the incomparable "building" modules were also required to be disjoint.

The following lemma provides the basis for our polynomial algorithm for the computation of the atomic decomposition since it allows us to construct $\mathcal{A}(\mathfrak{F}(\mathcal{O}))$ via α-modules only.

Lemma 4.2. *The notions of compact, α and genuine modules coincide.*

Algorithm 1 Atomic decomposition algorithm

Input: An ontology $\mathcal{O}$.
Output: The set $\mathfrak{G}$ of genuine $\top\bot^*$-modules; the poset of atoms $(\mathcal{A}(\mathfrak{F}(\mathcal{O})), \succeq)$; the set of generating axioms GenAxioms; for $\alpha \in$ GenAxioms, the cardinality CardAtom(α) of its atom.

ToDoAxioms $\leftarrow \top\bot^*\text{-mod}(\widetilde{\mathcal{O}}, \mathcal{O}) \setminus \top\bot^*\text{-mod}(\emptyset, \mathcal{O})$
GenAxioms $\leftarrow \emptyset$
for each $\alpha \in$ ToDoAxioms **do**
  Module$(\alpha) \leftarrow \top\bot^*\text{-mod}(\widetilde{\alpha}, \mathcal{O})$ % $\neq \emptyset$ due to line 3
  $new \leftarrow true$
  for each $\beta \in$ GenAxioms **do**
    if Module$(\alpha) =$ Module(β) **then**
      Atom$(\beta) \leftarrow$ Atom$(\beta) \cup \{\alpha\}$
      CardAtom$(\beta) \leftarrow$ CardAtom$(\beta) + 1$
      $new \leftarrow false$
    end if
  end for
  if $new = true$ **then**
    Atom$(\alpha) \leftarrow \{\alpha\}$
    CardAtom$(\alpha) \leftarrow 1$
    GenAxioms $\leftarrow$ GenAxioms $\cup \{\alpha\}$
  end if
end for
for each $\alpha \in$ GenAxioms **do**
  for each $\beta \in$ GenAxioms **do**
    if $\beta \in$ Module(α) **then**
      Atom$(\beta) \succeq$ Atom(α)
    end if
  end for
end for
$\mathcal{A}(\mathfrak{F}(\mathcal{O})) \leftarrow \{$Atom$(\alpha) \mid \alpha \in$ GenAxioms$\}$
$\mathfrak{G} \leftarrow \{$Module$(\alpha) \mid \alpha \in$ GenAxioms$\}$
return $[(\mathcal{A}(\mathfrak{F}(\mathcal{O})), \succeq), \mathfrak{G},$ GenAxioms, CardAtom$(\cdot)]$

Algorithm 1 sketches our algorithm for computing atomic decompositions that runs in polynomial time in the size of $\mathcal{O}$ (provided that module extraction is polynomial), and calls a module extractor as many times as there are axioms in $\mathcal{O}$.

Name	#Axioms	DL	#Gen. mods	#Con. comp.	#max. mod.	#max. atom
Koala	42	$\mathcal{ALCON}(\mathcal{D})$	23	5	18	7
Mereology	44	$\mathcal{SHIN}$	17	2	11	4
University	52	$\mathcal{SOIN}(\mathcal{D})$	31	11	20	11
People	108	$\mathcal{ALCHOIN}$	26	1	77	77
miniTambis	173	$\mathcal{ALCN}(\mathcal{D})$	129	85	16	8
OWL-S	277	$\mathcal{SHOIN}(\mathcal{D})$	114	1	57	38
Tambis	595	$\mathcal{ALCN}(\mathcal{D})$	369	119	236	61
Galen	4,528	$\mathcal{ALEHF}+$	3,340	807	458	29

Table 2: Experiments summary; only logical axioms are counted

Proposition 4.3. *Algorithm 1 is correct.*

Algorithm 1 considers, in `ToDoAxioms`, all axioms that are neither tautologies nor global (as in Remark 3.4), and computes all genuine modules, all atoms with their dependency relation and, for each module and atom, their cardinality. For each axiom α "generating" a module, the algorithm stores that module in $\texttt{Module}(\alpha)$ and the corresponding atom is constructed in $\texttt{Atom}(\alpha)$; those functions are undefined for axioms outside `GenAxioms`.

5 Empirical evaluation

We have run the atomic decomposition algorithm on a selection of ontologies, including those that were used in [Del Vescovo *et al.*, 2010; Parsia & Schneider, 2010], and indeed managed to compute the atomic decomposition of all ontologies, even for ontologies for which a complete modularization was not possible so far. Table 2 shows summary data for each ontology: size, expressivity, number of genuine modules, number of connected components, size of largest module and of largest atom. Our tests were obtained on a 2.16 GHz Intel Core 2 Duo Macbook with 2 GB of memory running Mac OS X 10.5.8; each atomic decomposition was computed within a couple of seconds, apart from that for Galen, which took less than 3 minutes.

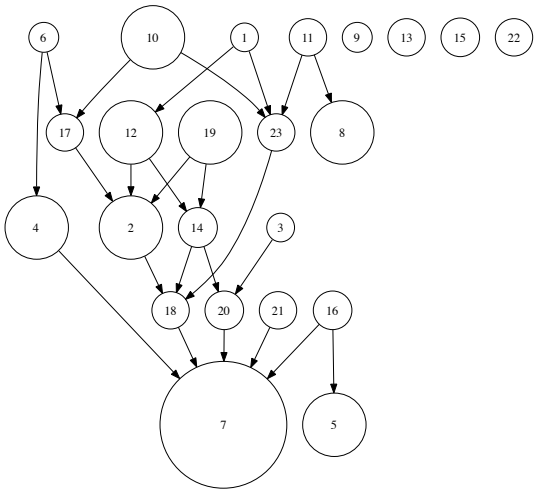

Figure 1: The atomic decomposition of Koala

We have also generated a graphical representation by using GraphViz[6], a package of open source tools initiated by AT&T Labs Research. Our atomic decompositions show atom size as node size, see Figure 1 for example. We notice that it shows four isolated atoms, e.g., Atom 22 in the top right corner, which consists of the axiom `DryEucalyptForest` $\sqsubseteq$ `Forest`. This means that, even though other modules may use terms from Atom 22, they do not "need" the axioms in Atom 22 for any entailments; i.e., removing (the axioms in) these isolated atoms from the ontology would not result in the loss of any entailments regarding other modules or terms. Of course, for entailments regarding both `DryEucalyptForest` and `Forest` and possibly other terms, this axiom is required. A similar structure is observable in all ontologies considered. The material on `http://bit.ly/i4olY0` includes graphs for all ontologies considered.

6 Related work

One solution to GetStruct is described in [Cuenca Grau *et al.*, 2006] via partitions related to $\mathcal{E}$-connections. When this approach succeeds, it divides an ontology into three kinds of modules: (A) those which import vocabulary (and axioms) from others, (B) those whose vocabulary (and axiom set) is imported, and (C) isolated parts. In various experiments, the numbers of parts extracted were quite low and the structure often corresponded usefully to user understanding. For instance, the tutorial ontology Koala, consisting of 42 logical axioms, is partitioned into one A-module about animals and three B-modules about genders, degrees and habitats.

It has also been shown in [Cuenca Grau *et al.*, 2006] that certain combinations of these parts provide coverage. For Koala, such a combination would still be the whole ontology (though smaller parts have coverage as well).

Ontology partitions based on $\mathcal{E}$-connections require rather strong conditions to ensure modular separation and have been observed to force together axioms and terms which are logically separable. As a consequence, it has been observed that ontologies with fairly elaborate modular structure have impoverished $\mathcal{E}$-connections based structures. Furthermore, the robustness properties of the parts (e.g., under vocabulary extension) are not as well-understood as those of locality-based modules. Partitions ensure, however, a linear upper bound on the number of parts.

Another approach to GetStruct is described in [Bezerra *et al.*, 2008]. It underlies the tool ModOnto, which aims at providing support for working with ontology modules that borrows intuitions from software modules. To the best of our knowledge, however, it has not been examined whether such modules provide coverage in the above sense. Furthermore, ModOnto does not aim at obtaining *all* modules.

A further procedure for partitioning an ontology is described in [Stuckenschmidt & Klein, 2004]. However, this method only takes the concept hierarchy of the ontology into account and can therefore not provide the strong logical guarantee of coverage.

In [Konev *et al.*, 2010], it was shown how to decompose the *signature* of an ontology to obtain the dependencies be-

[6] `http://www.graphviz.org/About.php`

tween its terms. In contrast to previous such approaches, this one is syntax-independent. While gaining information about term dependencies is one goal of our approach, we are also interested in the *modules* of the ontology.

Among the a-posteriori approaches to GetOne, some provide logical guarantees such as coverage, and others do not. The latter are not of interest for this paper. The former are usually restricted to DLs of low expressivity, where deciding conservative extensions—which underly coverage—is tractable. Prominent examples are the module extraction feature of CEL [Suntisrivaraporn, 2008] and the system MEX [Konev *et al.*, 2008]. However, we aim at an approach that covers DLs up to OWL 2.

7 Conclusion and outlook

We have presented the *atomic decomposition* of an ontology, and shown how it is a succinct, tractable representation of the modular structure of an ontology: it is of polynomial size and can be computed in polynomial time in the size of the ontology (provided module extraction is polynomial), whereas the number of modules of an ontology is exponential in the worst case and prohibitely large in cases so far investigated. Moreover, it can be used to assemble all other modules without touching the whole ontology and without invoking a direct module extractor.

Future work is three-fold: first, we will try to compute, from the atomic decomposition, good upper and lower bounds for the number of all modules to answer an open question from [Parsia & Schneider, 2010]. Second, we will investigate suitable labels for atoms, e.g., suitable representation of seed and module signatures, and how to employ the atomic decomposition for ontology engineering, e.g., to compare the modular structure with their intuitive understanding of the domain and thus detect modelling errors, and to identify suitable modules for reuse. Third, we will investigate when module extraction from the atomic decomposition is faster than extracting it using a module extractor.

References

[Baader *et al.*, 2003] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[Bao *et al.*, 2009] J. Bao, G. Voutsadakis, G. Slutzki, and V. Honavar. Package-based description logics. In *Modular Ontologies*, vol. 5445 of *LNCS*, pages 349–371. Springer-Verlag, 2009.

[Bezerra *et al.*, 2008] C. Bezerra, F. Freitas, A. Zimmermann, and J. Euzenat. ModOnto: A tool for modularizing ontologies. In *Proc. WONTO-08*, vol. 427 of *CEUR*. CEUR-WS.org/Vol-427/, 2008.

[Borgida & Serafini, 2003] A. Borgida and L. Serafini. Distributed description logics: Assimilating information from peer sources. In *J. Data Semantics I*, vol. 2800 of *LNCS*, pages 153–184. Springer-Verlag, 2003.

[Cuenca Grau *et al.*, 2006] B. Cuenca Grau, B. Parsia, E. Sirin, and A. Kalyanpur. Modularity and web ontologies. In *Proc. of KR-06*, pages 198–209. AAAI Press, 2006.

[Cuenca Grau *et al.*, 2008] B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular reuse of ontologies: Theory and practice. *J. of Artificial Intelligence Research*, 31:273–318, 2008.

[Cuenca Grau *et al.*, 2010] B. Cuenca Grau, C. Halaschek-Wiener, Y. Kazakov, and B. Suntisrivaraporn. Incremental classification of description logics ontologies. *J. of Automated Reasoning*, 44(4):337–369, 2010.

[Del Vescovo *et al.*, 2010] C. Del Vescovo, B. Parsia, U. Sattler, and T. Schneider. The modular structure of an ontology: an empirical study. In *Proc. of DL 2010*. CEUR-WS.org/Vol-573/, 2010.

[Del Vescovo *et al.*, 2011] C. Del Vescovo, B. Parsia, U. Sattler, and T. Schneider. The modular structure of an ontology: atomic decomposition. Technical report, 2011. Available at http://bit.ly/i4olY0.

[Ghilardi *et al.*, 2006] S. Ghilardi, C. Lutz, and F. Wolter. Did I damage my ontology? A case for conservative extensions in description logics. In *Proc. of KR-06*, pages 187–197. AAAI Press, 2006.

[Jiménez-Ruiz *et al.*, 2008] E. Jiménez-Ruiz, B. Cuenca Grau, U. Sattler, T. Schneider, and R. Berlanga Llavori. Safe and economic re-use of ontologies: A logic-based methodology and tool support. In *Proc. of ESWC-08*, vol. 5021 of *LNCS*, pages 185–199. Springer-Verlag, 2008.

[Jimeno *et al.*, 2008] A. Jimeno, E. Jiménez-Ruiz, R. Berlanga, and D. Rebholz-Schuhmann. Use of shared lexical resources for efficient ontological engineering. In *Proc. of SWAT4LS-08*, vol. 435 of *CEUR*, 2008.

[Konev *et al.*, 2008] B. Konev, C. Lutz, D. Walther, and F. Wolter. Logical difference and module extraction with CEX and MEX. In *Proc. of DL 2008*, vol. 353 of *CEUR*, 2008.

[Konev *et al.*, 2009] B. Konev, C. Lutz, D. Walther, and F. Wolter. Formal properties of modularization. In *Modular Ontologies*, vol. 5445 of *LNCS*, pages 25–66. 2009.

[Konev *et al.*, 2010] B. Konev, C. Lutz, D. Ponomaryov, and F. Wolter. Decomposing Description Logic Ontologies. In *Proc. of KR-10*, pages 236–246. AAAI Press, 2010.

[Kontchakov *et al.*, 2009] R. Kontchakov, L. Pulina, U. Sattler, T. Schneider, P. Selmer, F. Wolter, and M. Zakharyaschev. Minimal module extraction from DL-Lite ontologies using QBF solvers. In *Proc. of IJCAI-09*, pages 836–841, 2009.

[Kutz *et al.*, 2004] O. Kutz, C. Lutz, F. Wolter, and M. Zakharyaschev. $\mathcal{E}$-connections of abstract description systems. *Artificial Intelligence*, 156(1):1–73, 2004.

[Parsia & Schneider, 2010] B. Parsia and T. Schneider. The modular structure of an ontology: an empirical study. In *Proc. of KR-10*, pages 584–586. AAAI Press, 2010.

[Sattler *et al.*, 2009] U. Sattler, T. Schneider, and M. Zakharyaschev. Which kind of module should I extract? In *Proc. of DL 2009*, vol. 477 of *CEUR*. CEUR-WS.org/Vol-477/, 2009.

[Stuckenschmidt *et al.*, 2004] H. Stuckenschmidt, F. van Harmelen, P. Bouquet, F. Giunchiglia, and L. Serafini. Using C-OWL for the alignment and merging of medical ontologies. In *Proc. KR-MED*, vol. 102 of *CEUR*, pages 88–101. CEUR-WS.org/Vol-102/, 2004.

[Stuckenschmidt & Klein, 2004] H. Stuckenschmidt and M. Klein. Structure-based partitioning of large concept hierarchies. In *Proc. of ISWC-04*, vol. 3298 of *LNCS*, pages 289–303. Springer-Verlag, 2004.

[Suntisrivaraporn, 2008] B. Suntisrivaraporn. Module extraction and incremental classification: A pragmatic approach for $\mathcal{EL}^+$ ontologies. In *Proc. of ESWC-08*, vol. 5021 of *LNCS*, pages 230–244. Springer-Verlag, 2008.